OBJECT ORIENTED MODEL

User's Guide

POWERDESIGNER 7.5

Contents

About This Book

Subject

This book describes the PowerDesigner Object-Oriented Model environment. It shows you how to do the following:

- Build an Object-Oriented Model (OOM)
- Use classes, packages, and other modeling objects
- Verify the model and import a Rose model
- Generate a Conceptual Data Model and a Physical Data Model from the OOM
- Reverse engineer Java files
- Generate Java source files

Audience

Anyone who will be designing or building an OOM with PowerDesigner Object-Oriented Model will find this book useful. It requires an understanding of object modeling, as well as familiarity with UML theory. Some experience with database structure and terminology, is helpful but not required.

Where to find information

This book focuses on the design and construction of an object-oriented model. General information about the PowerDesigner modeling environment, for example using many of the graphic tools, interface features, merging models, and using the Browser, can be found in the PowerDesigner General Features Guide.

To help you do your work more easily, this book is divided into chapters that focus on particular goals.

If you want to	Use these parts of the book
Learn about the environment	Object-Oriented Model Basics
Build an object-oriented model	Building a Object-Oriented Model
Verifying the model and importing a Rose model	Managing Object-Oriented Models
Generating a conceptual data model or a physical data model	The chapters on generating conceptual and physical models

CHAPTER 1

Object-Oriented Model Basics

About this chapter

This chapter presents the PowerDesigner Object-Oriented Model. It provides you with an introduction to the basic notions of object-oriented modeling and the Unified Modeling Language (UML).

Contents

Functional overview

PowerDesigner Object-Oriented Model is a powerful design tool for object-oriented modeling. It gives you all the advantages of a graphical object design implementation.

With this product, you can:

- Build an **Object-Oriented Model** (OOM)
- Generate Java class source files (.java)
- Generate PowerBuilder objects
- Reverse engineer Java files (.class, .java, or .jar)
- Reverse engineer PowerBuilder objects
- Import a Conceptual Data Model (CDM)
- Import a Physical Data Model (PDM)
- Generate a Conceptual Data Model (CDM)
- Generate a Physical Data Model (PDM)
- Customize the Object-Oriented Model to suit physical and performance considerations
- Customize and print model reports

UML and object-oriented modeling

What is UML?

UML (The Unified Modeling Language) is a modeling language aimed at defining standards for object-oriented modeling. UML has become a standardized language largely through the work of the OMG (Object Management Group), a group composed of individuals and representatives of companies involved in object-oriented projects. However, its original conception drew much of its inspiration from the work of G. Booch, J. Rumbaugh, and I. Jacobson.

UML has a vocabulary and rules that focus on the conceptual and physical representation of a system. You use UML symbols and notations to create your models and diagrams in an OOM.

Notational Terminology

UML has a well-defined syntax and semantics that is clear and easy to use in object modeling. All of the terminology used in the OOM interface is consistent with UML language notations.

What is object-oriented modeling?

Object-oriented modeling refers to the process of using objects as the basic building blocks for creating a software system. An object in this context usually means a class, that is, a description of a set of common objects. Each object or class has identity and behavior. You use these objects to build models in which the properties of each object interact to perform certain actions that together make up a system of information.

What is an OOM?

An OOM contains a set of packages, classes, interfaces, and their relationships. These objects together form a class structure that is the logical design view of all (or part of) a software system. An OOM is essentially a static conceptual model of a software system.

You use PowerDesigner Object-Oriented Model to build object-oriented models (OOM). You can build an OOM for purely object-oriented modeling purposes, to generate Java files or for PowerBuilder, or you can use objects from an OOM in a Physical Data Model (PDM), for relational database design analysis.

When modeling objects graphically, you use diagrams such as the class diagram.

OOM roles

You can use an OOM to:

- Represent the physical organization of objects in a graphic format
- Generate Java class source files
- Generate PowerBuilder objects
- Reverse engineer Java class source files
- Reverse engineer PowerBuilder objects
- Generate a Conceptual Data Model (CDM)
- Generate a Physical Data Model (PDM)

OOM creation

There are several ways to create an OOM:

- Create an OOM from scratch
- Import one or more existing OOM
- Generate an OOM from a Conceptual Data Model (CDM)
- Generate an OOM from Physical Data Model (PDM)
- Import a Rational Rose model (.mdl)

Objects in an OOM

An OOM represents the interaction of the following objects:

Object	Selection Tool	Description
Package		General purpose sub-set used to organize objects into groups
Class		Set of objects that share the same attributes, operations, methods, and relationships
Interface		Collection of operations used to specify the externally visible behavior of a class, object, or other entity
Attribute	—	Named property of a class
Operation	—	Service that can be requested from a class
Association		Structural relationship between objects of different classes
Dependency		Relationship between two modeling elements, in which a change to one modeling element will affect the other modeling element
Realization		Link between classes and interfaces and between components and interfaces
Generalization		Link between classes showing that the subclass shares the structure or behavior defined in one or more superclasses

Creating a new OOM

Creating an OOM requires that you do the following:

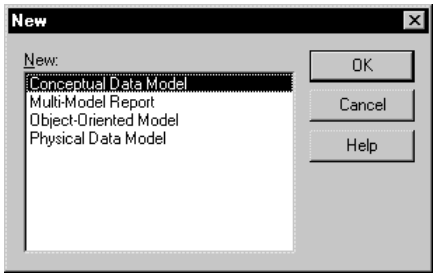
- Open a new file
- Give the OOM a name and a code

After you create an OOM, you can enrich its definition by entering properties and associating objects.

❖ To create an OOM:

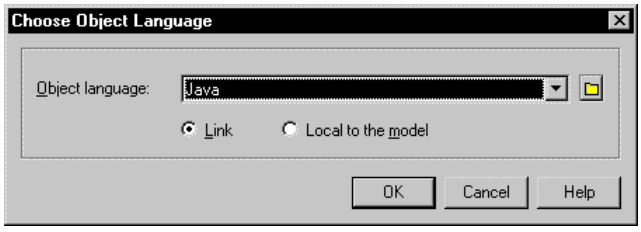
1 Select File➤New.
or
Click the New button in the toolbar.

The New window appears.



2 Select Object-Oriented Model and click OK.

The Choose Object Language window appears.



Every OOM is attached by default to one set of object language properties. When you create a new OOM, you choose a target language.

For more information on object language properties, see the chapter Object Language Properties.

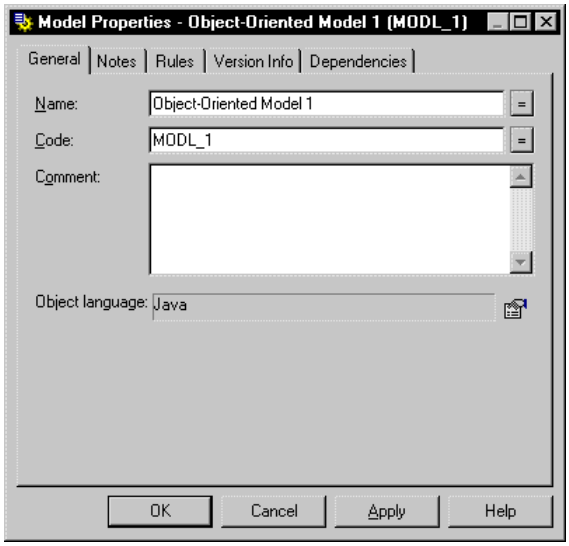
3 Select an object language from the Object language dropdown listbox.

4 Click OK.

If you were working on an existing workspace, PowerDesigner opens an new OOM. If there was no workspace open, PowerDesigner opens a new workspace and a new OOM.

5 Select Model➤Model Properties.
or
Right-click any empty space in the diagram and select Model Properties from the contextual menu.

The model property sheet appears.



6 Type a model name and model code.

7 Click OK.

Opening an existing OOM

An OOM has the file extension .OOM.

❖ **To open an existing OOM:**

1 Select File➤Open.
or
Click the Open tool.

A standard Windows file selection dialog box appears.

2 Select a file with the .OOM extension.

3 Click OK.

The model window displays the selected model.

Defining OOM model options

You can set model options and naming conventions that apply to all objects in the model. You can also set naming conventions for each type of object in your model.

You define OOM model options from the model options dialog box.

You can set options that apply to the following OOM objects:

- Classes
- Default data types
- Domain/Attribute

Classes

You can set the following option for classes in an OOM:

Option	Description
Show classes as datatypes	Includes classes that exist in the model in the list of data types that you can define for attributes, operations, or parameters

Default data types

The default data type is the data type that applies to attributes, operations and parameters if no data type is selected.

You can set the following options for default data type in an OOM:

Option	Description
Attribute Default Data Type	Defines the default data type for all new attributes
Operation Default Data Type	Defines the default return type for all new operations
Parameter Default Data Type	Defines the default data type for all new operation parameters

Domains/Attribute

From the Model Options dialog box, you can choose to enforce non-divergence between a domain definition and the attributes using the domain, for the following attribute properties:

Property	Attributes in the domain cannot have divergent
Data type	Data type, length, and precision
Check	Check parameters

Your choice of whether or not to enforce domain and attribute non-divergence has the following results:

Non-divergence	Result
Not enforced	Attributes that are divergent from the domain definition can remain attached to the domain
Enforced	Attributes that are divergent from the domain (for certain attribute properties) must be detached from the domain

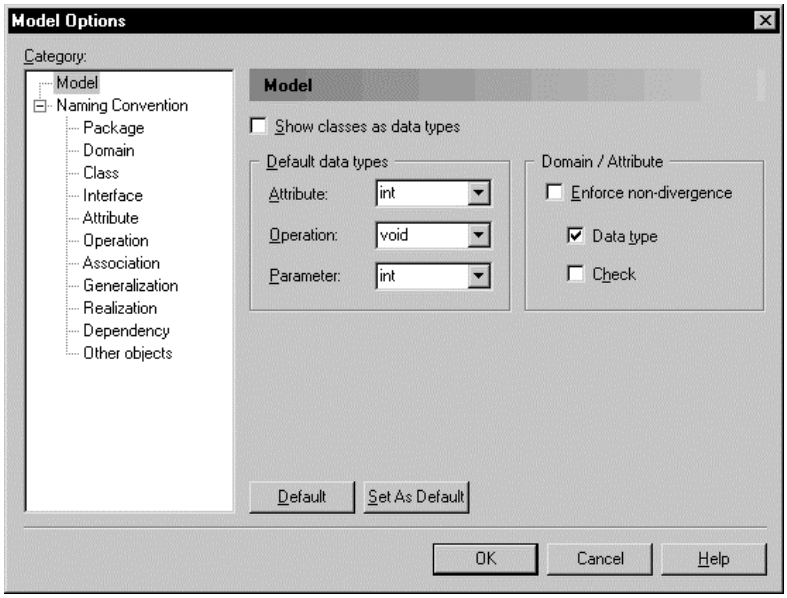
If you modify domain non-divergence options, these changes apply only to the current OOM.

For more information on PowerDesigner model options, see the PowerDesigner General Features Guide.

❖ To define OOM model options:

1 Select Tools➤Model Options.
or
Right-click any empty space in the diagram and select Model Options from the contextual menu.

The Model Options dialog box opens to the model page.



2 Select model options in the different boxes.

3 Click OK.

Defining OOM properties

The Model property sheet displays the definition of the current model. From this property sheet you can modify the model definition.

A OOM has the following model properties:

Property	Description	Length
Name	Name for the model	254
Code	Code for the model. This code is generated in database scripts	254
Comment	Descriptive label for the model	—
Object language	Current object language for the model. You can open the property sheet for the current object language by clicking the Properties tool to the right of the box	—

❖ **To modify the model properties:**

1 Select Model➤Model Properties.
or
Right click the diagram background and select Properties from the contextual menu.

The model property sheet appears.

2 Type changes to model properties.

3 Click OK.

CHAPTER 2

Building an Object-Oriented Model

About this chapter

This chapter describes how to build an Object-Oriented Model (OOM). It explains the role of each object in an OOM and how to create and modify objects.

Contents

Defining packages

A package is a general purpose mechanism for organizing elements into groups.

When you are working with large models, you can split any model into smaller subdivisions in order to avoid manipulating the entire set of data of the model. Packages can be useful to assign portions of a model, representing different tasks and subject areas, to different development teams.

You can create as many packages as you need in a model. The name of each package must be unique in the model.

Package hierarchy

You can create several packages at the same hierarchical level within a model. or decompose a package into other packages and continue this process without limitation in decomposition depth. At each level of decomposition you can create several diagrams.

Packages work as models, they can contain the following items:

- Model objects
- Other packages
- Diagrams, in order to have different views of the contents of the package. Each package appears with a default diagram window

For more information on packages, see the PowerDesigner Feature Guide.

Package properties

Packages have properties displayed on property sheets. All packages share the following common properties:

Property	Description	Length
Name	Names are like titles that clearly identify the package during the design process	254
Code	Codes are references for packages	254
Comment	A comment is an optional label that describes a package and provides more information than the name	—
Namespace	Option that defines the package as being the area in which the name of an object must be unique in order to be used.	—

Displaying text in package symbols

You can define the following display preferences for a package:

Preference	Description
Show stereotypes	When selected, displays the stereotype of the package
Show constraints	When selected, displays the constraints (types of business rule) that are attached to the package

You modify the display preferences for a package in the Display Preferences dialog box.

❖ **To modify the package display preferences:**

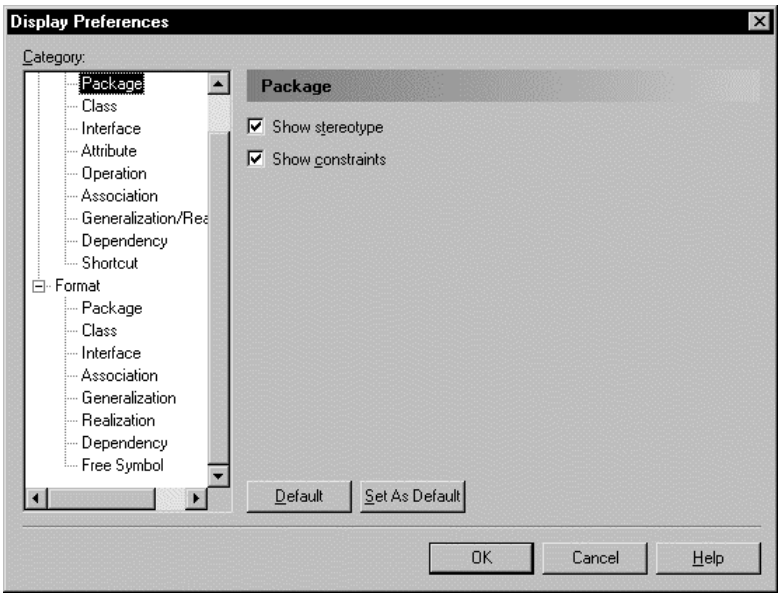
1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Package.

The package display preferences page appears.



4 Modify the package display preferences.

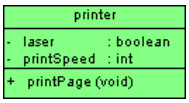
5 Click OK.

Defining classes

A class is a description of a set of objects that have a similar structure and behavior, and share the same attributes, operations, relationships, and semantics. A class usually implements one or more interfaces.

Classes are the main building blocks of an OOM. Classes, and the relationships that you create between them, form the basic structure of an OOM. Typically, classes represent either real, abstract or conceptual things that together make a whole or a part of a particular problem or system.

The following example shows the class Printer with its attributes and operations.



Class properties

A class has the following general properties:

Property	Description	Maximum length
Name	Name of the class	254
Code	Reference name for the class	254
Comment	Descriptive comment for the class	—
Stereotype	Subclassification of a class derived from an existing one. Extends the semantics of a class without changing it's structure	—
Type	Set of instances that share the same operations, abstract attributes, and relationships, and semantics	—
Visibility	Visibility of the class, whose value denotes how it may be seen and used by other objects	—
Cardinality	Specific number of instances that the class can have	—
Persistence	Lifetime of the instances of a class. An object can be persistent or transient. If it is persistent, it continues to exist after the process that created it has ceased to exist. If it is transient, then it ceases to exist when the process that created it ceases to exist	—
Abstract	Indicates that the class cannot be instantiated and therefore has no direct instances	—
Final	Specifies that the class cannot have any inherited objects	—
Generate	Indicates that the class will be automatically included among the objects generated from the model when you launch the generation process	—

A class definition also includes the following properties, which are defined on associated property sheets:

Property	Description
Attribute	Defines the characteristics of a class
Operation	Carries out a service that effects behavior
Rule	A business rule that your business follows. Business rules guide and document the creation of a model

Analyzing class properties

The following class properties each have several default values from which you can select from:

- Stereotype
- Type
- Visibility
- Cardinality

Stereotype

Stereotypes are classes that are derived from existing classes but that are specific to a particular problem. They enable you to extend the semantics of a class without changing its structure. In this way stereotypes must be based on existing classes but they allow you to provide additional distinctions for these classes. Stereotypes can be predefined or user-defined. They allow you to add additional information that may be specific to a project or process. They may extend the semantics, but not the structure of pre-existing classes.

Modify or creating new stereotypes

You can modify an existing stereotype or create a new one from the object language property sheet.

For more information on modifying and creating variables of an object language, see the chapter Object Language Properties.

Default stereotypes

You can declare a class to be one of the following stereotypes:

Stereotype	Description
actor	Coherent set of roles that users of use cases play when interacting with the use cases
enumeration	List of named values used as the range of a particular attribute type
exception	Exception class. Used mostly in relation to error messages
implementationClass	Class whose instances are statically typed, and that defines the physical data structure and methods of a class as implemented in traditional programming languages
process	Heavyweight flow that can execute concurrently with other processes
signal	Specification of an asynchronous stimulus communicated between instances
thread	Lightweight flow that can execute concurrently with other threads within the same process. Usually executes inside the address space of an enclosing process
type	Abstract class used only to specify the structure and behavior of a set of objects, not the implementation
utility	Class that has no instances

Type

You can declare a class to be one of the following types:

- Business Object
- Class
- Storage
- Utility
- Visual Object
- JavaBean

Visibility

The visibility of a class refers to the way in which it can be seen by other objects. A class that is visible to another object may influence the structure or behavior of the object, or similarly, its own properties may be affected by the other object.

Property	Visible
Private	Only to the class itself
Protected	Only to the class and its inherited objects
Package	To all objects contained within the same package
Public	To all objects in the model

Cardinality

The cardinality of a class specifies the number of instances that the class can have.

Cardinality	Number of instances
0..0	None
0..1	None or one
0..*	None to an unlimited number
1..1	One to one
1..*	One to an unlimited number
*	Unlimited number

Creating a class

There are three ways to create a class:

- Create a class symbol in the Browser
- Add a new class to the list of classes
- Create a class symbol directly in a diagram

Creating a class from the Browser

❖ **To create a class from the Browser:**

1 Right-click the Classes category in the Browser.

2 Select New from the contextual menu.

The property sheet of the class appears.

3 Type a class name and a class code.

4 Click OK.

A new class is created in the Classes category.

Creating a class from the list of classes

❖ **To create a class by inserting it in the list:**

1 Select Model➤Classes.

The list of classes appears.

> **Accessing the list of classes**
> The list of classes is accessible only from a diagram. If the current diagram is of a package, the list contains all the classes that exist in the package. If the current diagram is of the model, the list contains all the classes that exist in the model.

2 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

3 Type a name and code for the class.

4 Select a stereotype from the Stereotype dropdown listbox.

5 Select a visibility from the Visibility dropdown listbox.

6 Click OK.

A symbol for this class is inserted in the current model.

Creating a class from a diagram

❖ **To create a class in a diagram:**

1 Click the Class tool in the palette toolbar.

2 Click anywhere in the diagram.

The following symbol appears at the click position:

At creation, a class is named Class*n*, where *n* is a number assigned in the order of the creation of objects.

3 Click the Pointer tool in the palette toolbar.

4 Double-click the new class symbol in the diagram.

The class property sheet appears.

5 Type a class name and a class code.

6 Click OK.

The newly created class is visible in the Browser.

Inner classes

An inner class is a class that is defined within another (outer) class or interface. Inner classes are commonly used in Java. They help you to improve the overall visibility of your model by allowing you to group together classes that logically belong together.

You can add inner classes to a class or an interface.

Attaching an inner class to a class

You attach an inner class to a class (or interface) from the Inner Classes page of the class (or interface) property sheet.

❖ To declare an inner class within a class:

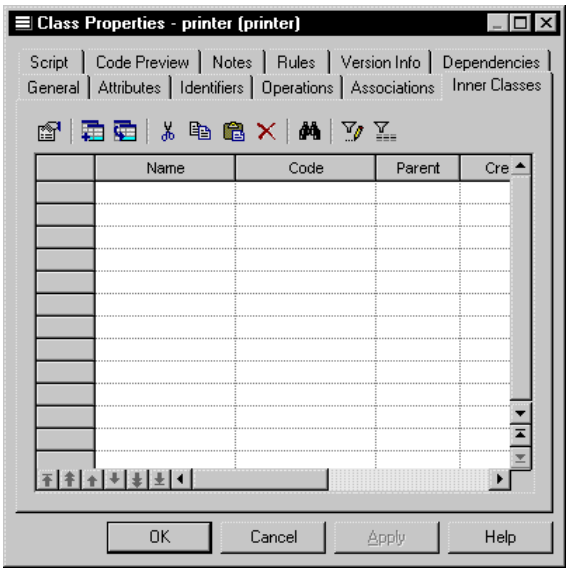
1 Double-click a class in the model.

The class property sheet opens to the General page.

> **Opening property sheets at last accessed page**
> Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

2 Click the Inner Classes tab.

The Inner Classes page appears.



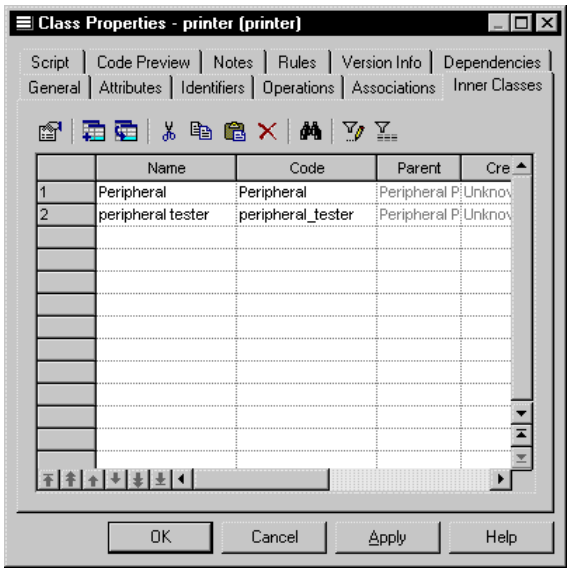
3 Click the Attach inner class tool.

A selection window appears.

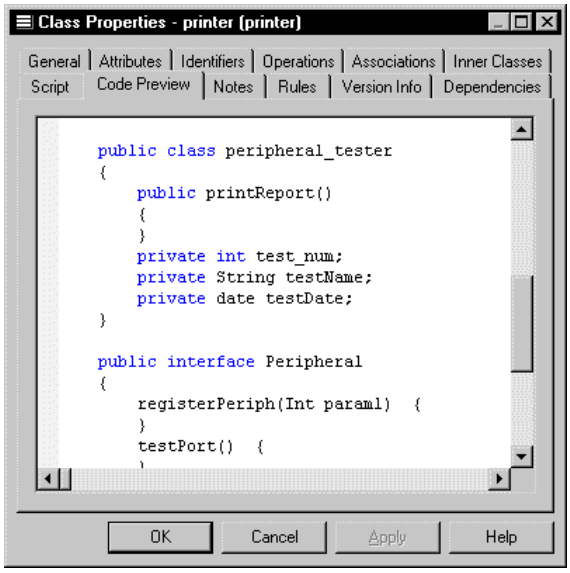
4 Click the classes you want to attach as inner classes in the current class.

5 Click OK.

The classes appear in the list of inner classes for the current class, and the definition of the classes are added to the current class definition.



6 Click the Code Preview tab to visualize the inner class definitions within the current class:



7 Click OK.

Detaching an inner class from a class

Once you have attached an inner class to a class, to remove its declaration from the class you must use detach it.

❖ **To detach an inner class from a class:**

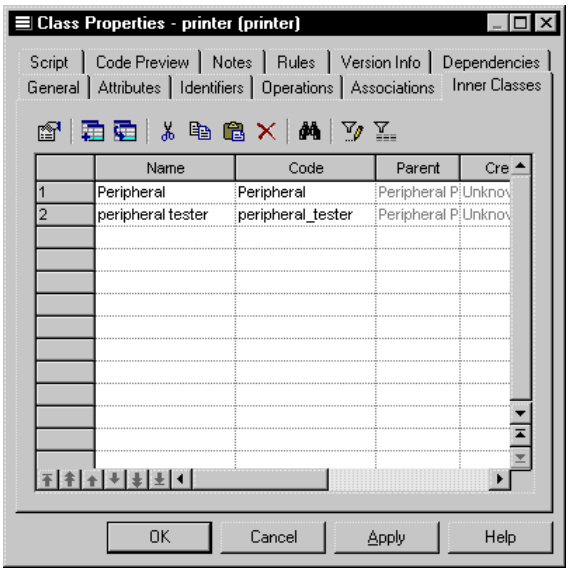
1 Double-click a class in the model.

The class property sheet opens to the General page.

> **Opening property sheets at last accessed page**
> Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

2 Click the Inner Classes tab.

The Inner Classes page appears.



3 Select an inner class from the list of inner classes.

4 Click the Detach inner class tool.

The inner class definition is detached from the current class definition and is removed from the list of inner classes of the current class.

5 Click OK.

Classifiers

A classifier, in UML terminology, is a mechanism that has structural (attributes) and behavioral (operations) features. A class is the most important classifier, but all objects that can have instances, such as interfaces or associations, are classifiers.

Modifying class properties

There are two approaches to modifying class properties:

- Modify the property sheet of the class
- Modify an entry in the list of classes

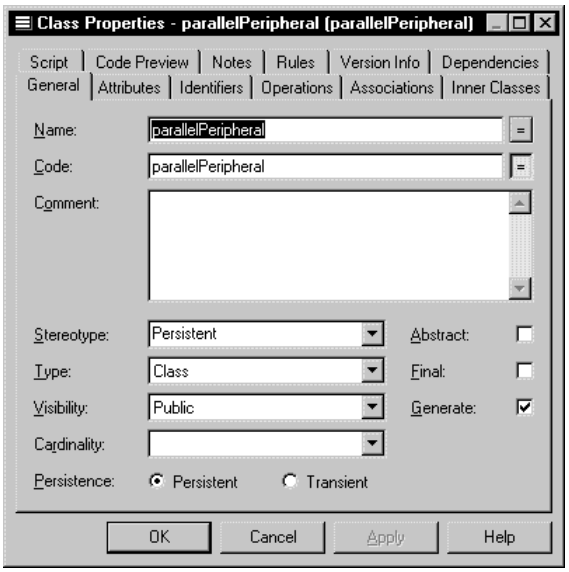
Modifying class properties from its property sheet

The class property sheet displays the definition of the class, which you can modify.

❖ **To modify class properties from its property sheet:**

1 Double-click the class in the Browser.
or
Double-click the class in the list of classes.
or
Double-click the class in a diagram.

The class property sheet appears.



2 Type or select class properties.
or
Click on a page tab.
Type or select class properties as required.

3 Click OK.

Modifying class properties from the list of classes

The list of classes includes all classes attached to the current model or package. You can modify the class properties from the list.

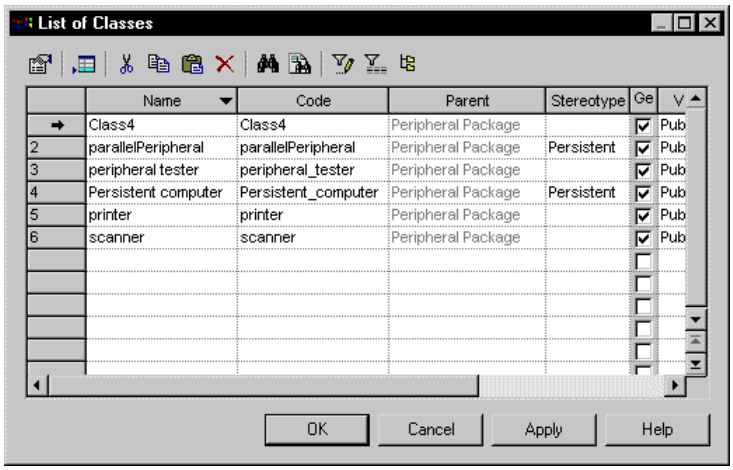
❖ **To modify class properties from the list of classes:**

1 Select Model➤Classes.

The list of classes appears.

2 Click the class that you want to modify.

An arrow appears at the beginning of the line.



3 Modify any of the properties of the class directly in the list.

4 Click OK.

Adding objects to a class

You can add an object to a class, that already exists in the model, but which belongs to another class.

You can add the following objects to an object:

Object	Description
Attribute	Named property of a class that defines the characteristics of a class
Operation	Implementation of a service that can be requested from any object of the class in order to affect behavior
Business rule	Written statement specifying what the information system must do or how it must be structured to support business needs

You add an object to a class from the list in the page corresponding to the object, in the class property sheet.

When you add an object to a class in this way, you in fact create a copy of the object. The new object exists as a unique object, and you can then make changes to it as you would to any object in the model.

Adding an attribute to a class

An attribute is a named property of an object that defines the characteristics of the object.

You can add attributes that already exist in the model and which belong to other objects.

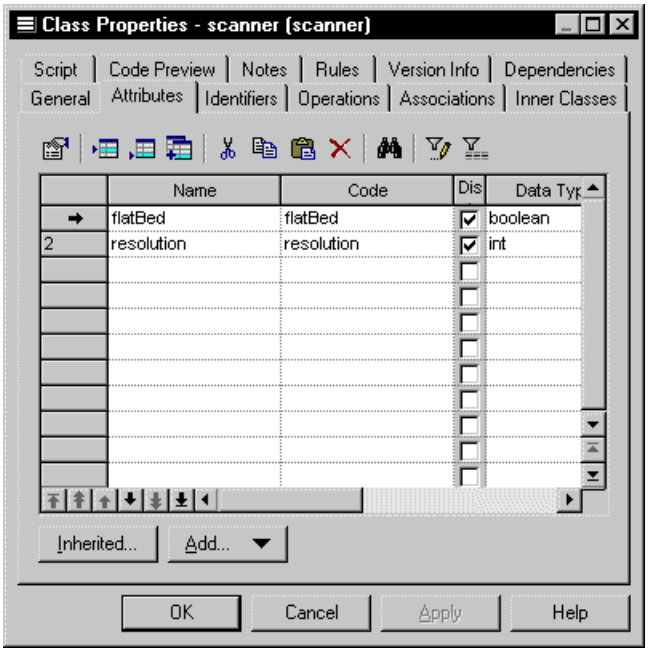
❖ To add an attribute to a class:

1 Double-click a class in the model.

The class property sheet appears.

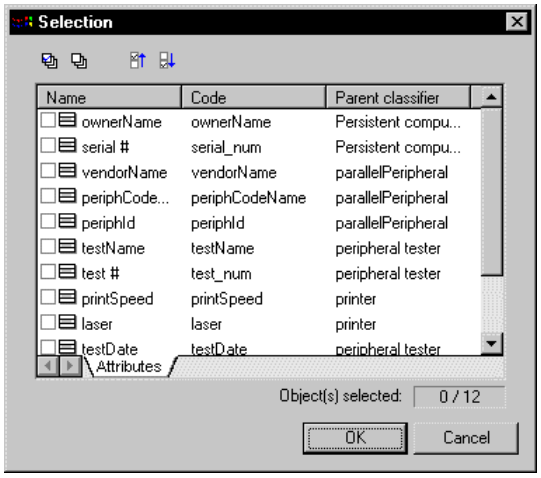
2 Click the Attributes tab.

The Attributes page appears.



3 Click the Add Attributes tool.

The Selection window appears. It contains a list of all the attributes that exist in the model, with the exception of those that already belong to the class.



4 Select the attributes that you want to add to the class.
or
Use the Select All tool to add all the attributes in the list to the class.

5 Click OK.

The attributes are added to the class and appear at the end of the list.

6 Click OK.

Adding an operation to a class

An operation is the implementation of a service that can be requested from any object of the class in order to affect behavior.

You can add operations that already exist in the model and which belong to other objects.

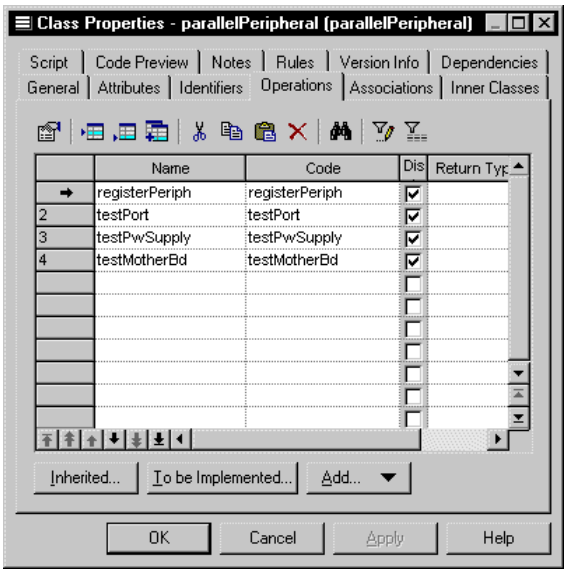
❖ To add an operation to a class:

1 Double-click a class in the model.

The class property sheet appears.

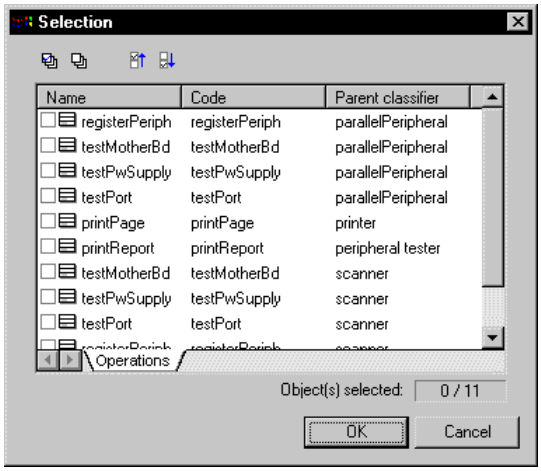
2 Click the Operations tab.

The Operations page appears.



3 Click the Add Operations tool.

The Selection window appears. It contains a list of all the operations that exist in the model, with the exception of those that already belong to the class.



4 Select the operations that you want to add to the class.

5 Click OK.

The operations are added to the class and appear in the list of operations for the class.

6 Click OK.

Preview the code of a class or an interface

You can preview the code of a class or an interface in the Code Preview page of the Property sheet of a class or an interface. You cannot edit the code in this window.

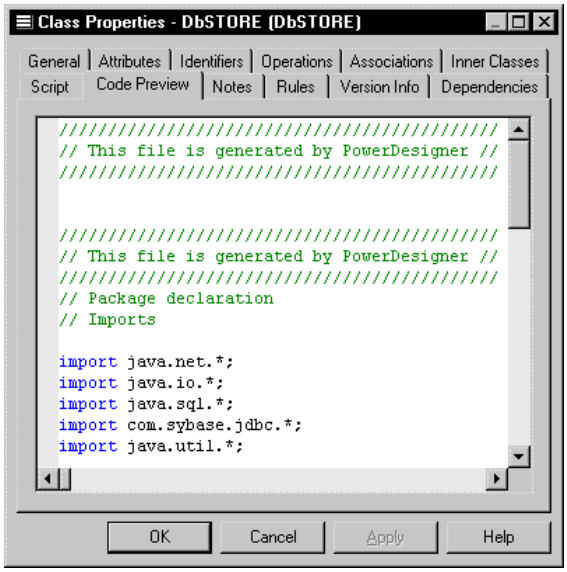
❖ To preview the code of a class:

1 Double-click a class in the model.

The class property sheet appears.

2 Click the Code Preview tab.

The Code Preview page appears.



3 Click OK.

Displaying text in class symbols

You can define the following display preferences for a class:

Preference	Description
Show attributes	Displays all the attributes of the class, or limits the number displayed to a maximum that you specify in the Limit box
Show operations	Displays all the operations of the class, or limits the number displayed to a maximum that you specify in the Limit box
Show stereotypes	When selected, displays the stereotype of the class
Show constraints	When selected, displays the constraints (types of business rule) that are attached to the class

You modify the display preferences for a class in the Display Preferences dialog box.

❖ **To modify the class display preferences:**

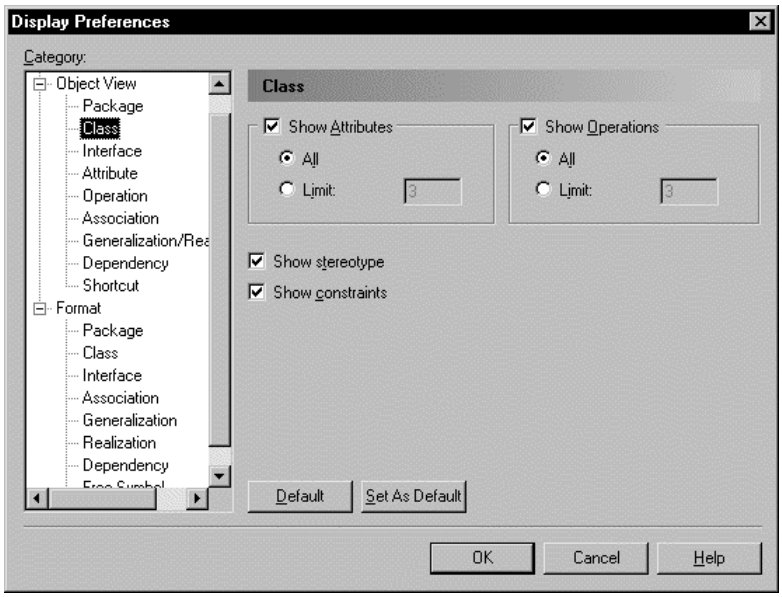
1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Class.

The class display preferences page appears.



4 Modify the class display preferences.

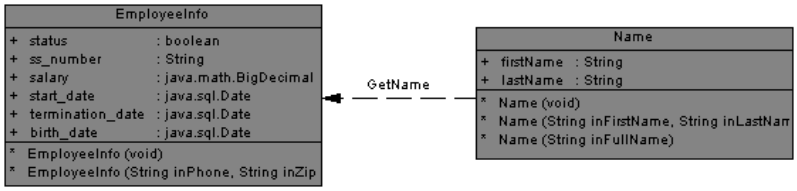
5 Click OK.

Defining interfaces

An interface is a type of class that is similar to a class but which is used to implement the specification of an abstraction of a class. An interface is a collection of operations used to specify the externally visible behavior of a class. It has no implementation of its own.

A class that implements all the operations in an interface is said to realize the interface. A class that requires one or more operations in an interface is said to use the interface. The interface includes the signatures of the operations of the class. Usually, an interface specifies only a limited part of the behavior of a class. A class can implement one or more interfaces.

The following example shows a Name (interface) that realizes the action GetName for an Employee (class).



Interface properties

An interface has the following properties:

Property	Description	Maximum length
Name	Name of the interface	254
Code	Reference name for the interface	254
Comment	Descriptive comment for the interface	—
Stereotype	Subclassification of an interface derived from an existing one. Extends the semantics of an interface without changing it's structure	—
Visibility	Visibility of the interface, whose value denotes how it may be seen outside its enclosing name space	—
Generate	Indicates that the class will be automatically included among the objects generated from the model when you launch the generation process	—

An interface definition also includes the following properties, which are defined on associated property sheets:

Property	Description
Attribute	Defines the characteristics of an interface
Operations	Carries out a service that effects behavior
Business rules	A rule that your business follows. Business rules guide and document the creation of a model

Analyzing interface properties

Visibility

The visibility of an interface refers to the way in which it can be seen by other objects. An interface that is visible to another object may influence the structure or behavior of the object, or similarly, its own properties may be affected by the other object.

Property	Visible
Private	Only to the interface itself
Protected	Only to the interface and its inherited objects
Package	To all objects contained within the same package
Public	To all objects in workspace

Creating an interface

There are three ways to create an interface:

- Create an interface symbol in the Browser
- Create an interface symbol directly in a diagram
- Add a new interface to the list of classes

Creating an interface from the Browser

❖ **To create an interface from the Browser:**

1 Right-click the Interfaces category in the Browser.

2 Select New from the contextual menu.

The property sheet of the interface appears.

3 Type an interface name and an interface code.

4 Click OK.

A new interface is created in the Interfaces category.

Creating an interface from the list of interfaces

❖ **To create an interface by inserting it in the list:**

1 Select Model➤Interfaces.

The list of interfaces appears.

> **Accessing the list of interfaces**
> The list of interfaces is accessible only from a diagram. If the current diagram is of a package, the list contains all the interfaces that exist in the package. If the current diagram is of the model, the list contains all the interfaces that exist in the model.

2 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

3 Type a name and code for the interface.

4 Select a stereotype from the Stereotype dropdown listbox.

5 Select a visibility from the Visibility dropdown listbox.

6 Click OK.

A symbol for this interface is inserted in the current model.

Creating an interface from a diagram

❖ **To create an interface in a diagram:**

1 Click the Interface tool in the palette toolbar.

2 Click anywhere in the interface diagram.

The following symbol appears at the click position:

At creation, an interface is named Intf*n*, where *n* is a number assigned in the order of the creation of objects.

3 Click the Pointer tool in the palette toolbar.

4 Double-click the new interface symbol in the diagram.

The interface property sheet appears.

5 Type an interface name and an interface code.

6 Click OK.

The newly created interface is visible in the Browser.

Modifying interface properties

There are two approaches to modifying interface properties:

- Modify an interface property sheet
- Modify an entry in the list of interface

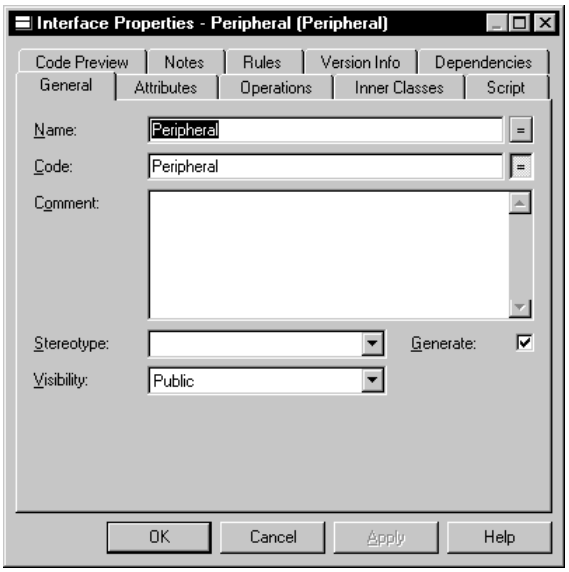
Modifying interface properties from its property sheet

The interface property sheet displays the definition of an interface, which you can modify.

❖ **To modify interface properties from its property sheet:**

1 Double-click the interface in the Browser.
or
Double-click the interface in the list of interfaces.
or
Double-click the interface in a diagram.

The interface property sheet opens to the General page.



Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

2 Type or select interface properties.
or
Click on a page tab.
Type or select interface properties as required.

3 Click OK.

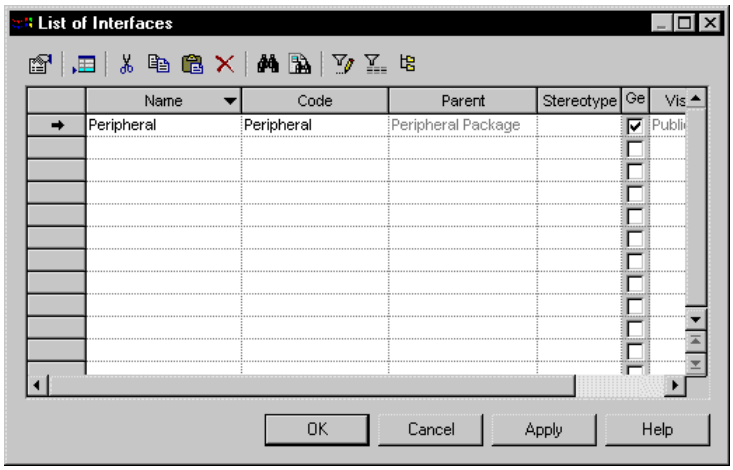
Modifying interface properties from the list of interfaces

The list of interfaces includes all interfaces attached to the current model. You can modify the interface properties from the list.

❖ **To modify interface properties from the list of interfaces:**

1 Select Model➤Interfaces.

The list of interfaces appears.



2 Click the interface that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the interface directly in the list.

4 Click OK.

Adding inner classes to an interface

An inner class is a class definition that is defined within another (outer) class definition. Inner classes are commonly used in Java. They help you to improve the overall visibility of your model by allowing you to group together classes that logically belong together.

You can add inner classes to a class or an interface.

For more information on inner classes, see the section Inner classes.

Adding objects to an interface

You can add an object to an interface, that already exists in the model, but which belongs to another object.

You can add the following objects to an interface:

Object	Description
Attribute	Named property of an interface that defines the characteristics of an interface
Operation	Implementation of a service that can be requested from any object of the interface in order to affect behavior
Business rule	Written statement specifying what the information system must do or how it must be structured to support business needs

You add an object to an interface from the list in the page corresponding to the object, in the interface property sheet.

When you add an object to an interface in this way, you in fact create a copy of the object. The new object exists as a unique object, and you can then make changes to it as you would to any object in the model.

Adding an attribute to an interface

An attribute is a named property of an object that defines the characteristics of the object.

You can add attributes to an interface that already exist in the model and which belong to other objects.

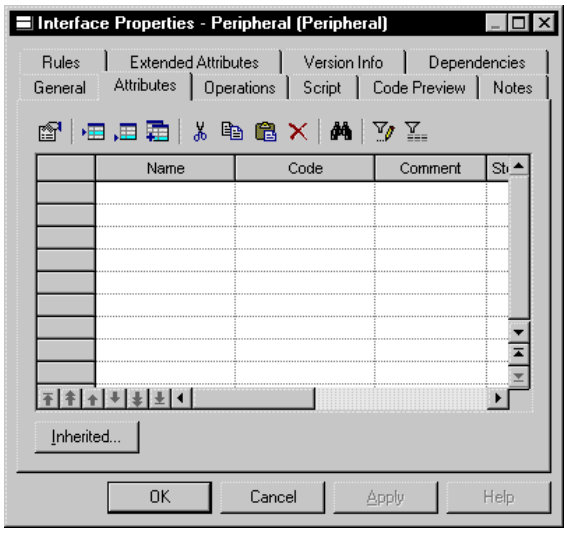
❖ **To add an attribute to an interface:**

1 Double-click an interface in the model.

The interface property sheet appears.

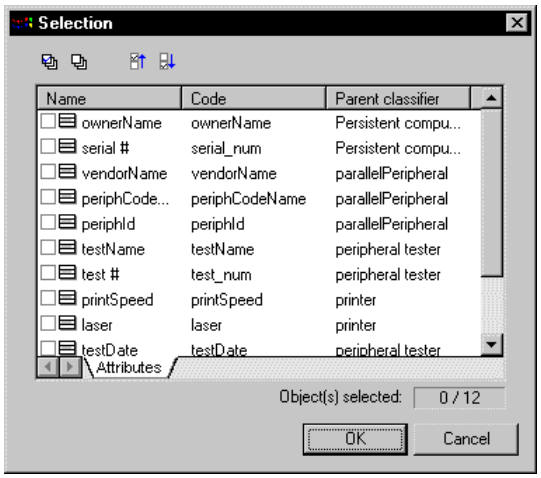
2 Click the Attributes tab.

The Attributes page appears.



3 Click the Add Attributes tool.

The Selection window appears. It contains a list of all the attributes that exist in the model, with the exception of those that already belong to the interface.



4 Select the attributes that you want to add to the interface.

5 Click OK.

The attributes are added to the interface and appear in the list of attributes for the interface.

6 Click OK.

Adding an operation to an interface

An operation is the implementation of a service that can be requested from any object of the class in order to affect behavior.

You can add operations that already exist in the model and which belong to other objects.

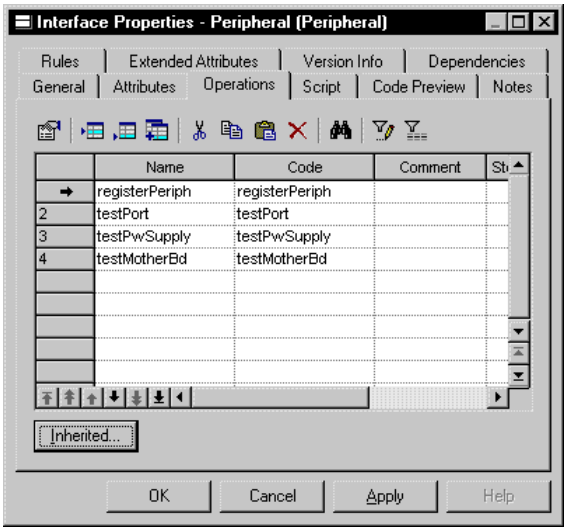
❖ To add an operation to an interface:

1 Double-click an interface in the model.

The interface property sheet appears.

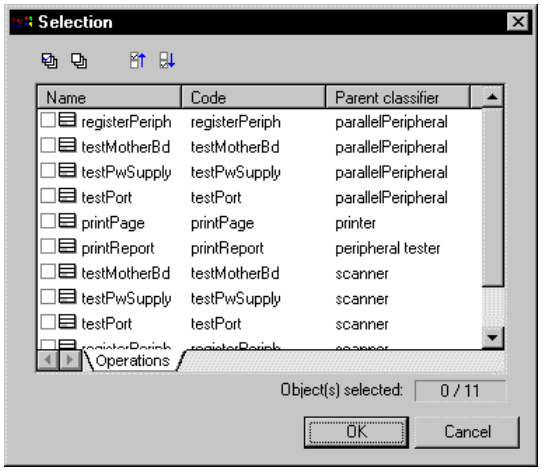
2 Click the Operations tab.

The Operations page appears.



3 Click the Add Operations tool.

The Selection window appears. It contains a list of all the operations that exist in the model, with the exception of those that already belong to the interface.



4 Select the operations that you want to add to the interface.

5 Click OK.

The operations are added to the interface and appear in the list of operations for the interface.

6 Click OK.

Preview the code of an interface

You can preview the code of an interface or a class in the Code Preview page of the Property sheet of an interface. You cannot edit the code in this window.

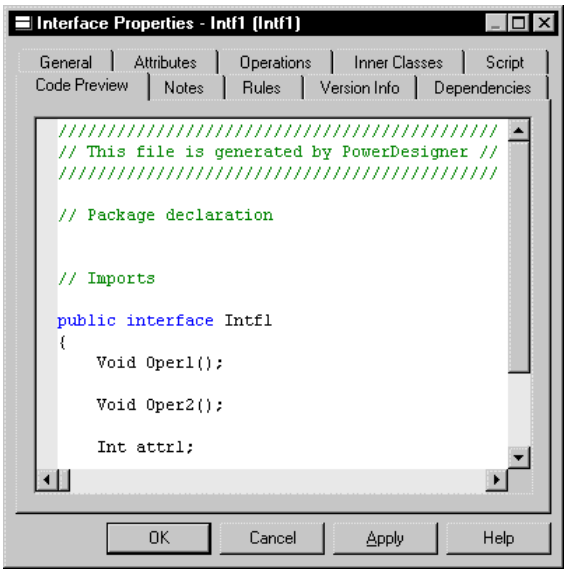
❖ **To preview the code of an interface:**

1 Double-click an interface in the model.

The interface property sheet appears.

2 Click the Code Preview tab.

The Code Preview page appears.



3 Click OK.

Displaying text in interface symbols

You can define the following display preferences for an interface:

Preference	Description
Show attributes	Displays all the attributes of the interface, or limits the number displayed to a maximum that you specify in the Limit box
Show operations	Displays all the operations of the interface, or limits the number displayed to a maximum that you specify in the Limit box
Show stereotypes	When selected, displays the stereotype of the interface
Show constraints	When selected, displays the constraints (types of business rule) attached to the interface

You modify the display preferences for an interface in the Display Preferences dialog box.

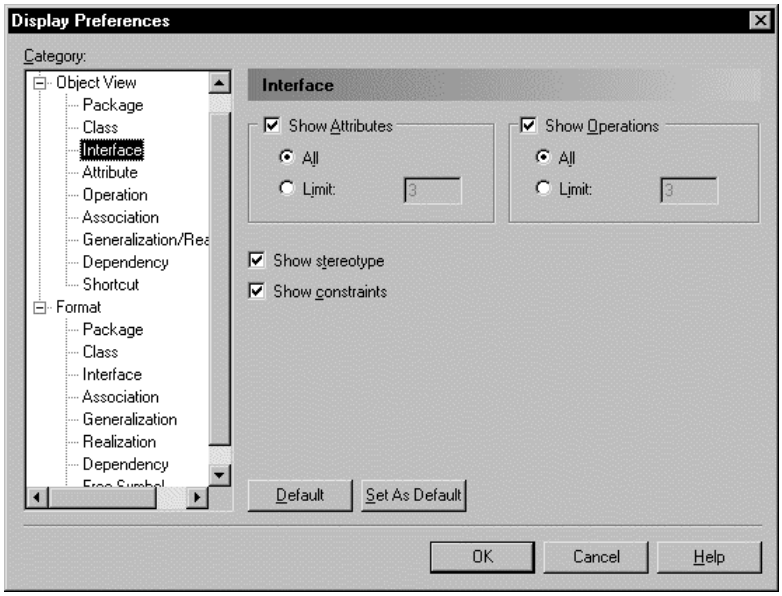
❖ **To modify the interface display preferences:**

1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Interface.



4 Modify the interface display preferences.

5 Click OK.

Defining attributes

Attributes define the characteristics of a class. A class may have none or several attributes. An attribute is a named property of a class that describes the range of values that instances of the property may hold. Each object in a class has the same attributes, but the values of the attributes may be different.

Attribute names within a class must be unique. You can give identical names to two or more attributes only if they exist in different classes.

You can create attributes for the following objects of an OOM:

- Class
- Interface

You can attach attributes to an Identifier.

Attribute properties

An attribute has the following properties:

Property	Description	Maximum length
Parent	Object to which the attribute belongs to	254
Name	Name of the attribute	254
Code	Reference name for the attribute	254
Comment	Descriptive comment for the attribute	—
Stereotype	Subclassification of an attribute derived from an existing one. Extends the semantics of an attribute without changing it's structure	—
Data Type	Set of instances that share the same operations, abstract attributes, and relationships, and semantics	—
Visibility	Visibility of the attribute, whose value denotes how it may be seen outside its enclosing name space	—
Multiplicity	Specification of the range of allowable cardinalities that a set may assume	—
Initial value	Initial value of the attribute	—
Changeability	Specifies that the value of the attribute cannot be modified once the object has been initialized	—
Length	Maximum number of characters	—
Precision	Number of places after the decimal point, for data values that can take a decimal point	—
Domain	Name of the associated domain	—
Static	Defines the attribute as static, meaning it cannot be modified	—
Derived	Indicates that the attribute is a calculated formula	—
Identifier	When selected, converts the attribute into a primary key after generation of the OOM to a PDM	—
Property	When selected, converts the attribute into a property after generating PowerBuilder objects from the OOM	—

An attribute definition also includes business rules, which are defined on associated property sheets.

Analyzing attribute properties

The following attribute properties each have several default values from which you can select from:

- Data Type
- Visibility
- Multiplicity

Data Type

You can select one of the following instances as a data type for an attribute:

Boolean
Byte
Char
Double
Float
Int
Long
Short

Visibility

Property	Visible
Private	Only to the attribute itself
Protected	Only to the attribute and its inherited objects
Package	To all objects contained within the same package
Public	To all objects

Multiplicity

The cardinality of each of an attribute is called the multiplicity.

Cardinality	Number of instances in relation
0..0	None
0..1	None or one
0..*	None to infinity
1..1	One to one
1..*	One to infinity
*	Infinity

You can change the default format of cardinalities from the registry:

```
HKEY_CURRENT_USER\Software\Sybase\PowerDesigner
7\ModelOptions\Cld
MultiplicityNotation = 1 (0..1) or 2 (0,1)
```

Creating an attribute

There are three ways to create an attribute:

- Create an attribute symbol in the Browser
- Add a new attribute to the list of attributes
- Create an attribute from a class in a diagram

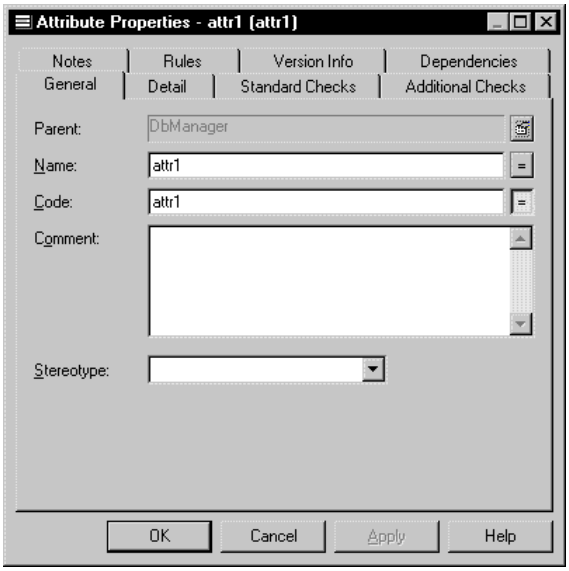
Creating an attribute from the Browser

❖ **To create an attribute from the Browser:**

1 Right-click the Attributes category in the Browser.

2 Select New from the contextual menu.

The property sheet of the attribute appears.



3 Type an attribute name and an attribute code.

4 Click OK.

A new attribute is created in the Attributes category.

Creating an attribute from the list of attributes

❖ **To create an attribute by inserting it in the list:**

1 Select Model➤Attributes.

The list of attributes appears.

> **Accessing the list of attributes**
> The list of attributes is accessible only from a diagram. If the current diagram is of a package, the list contains all the attributes that exist in the package. If the current diagram is of the model, the list contains all the attributes that exist in the model.

2 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

3 Type an attribute name and an attribute code.

4 Click the Stereotype column.

5 Select a stereotype from the Stereotype dropdown listbox.
or
Type a stereotype in the Stereotype column.

6 Click the Data Type column.

7 Select a data type from the Data Type dropdown listbox.
or
Type a data type in the Data Type column.

8 Click the Visibility column.

9 Select a value from the Visibility dropdown listbox.

10 Click the Multiplicity column.

11 Select a cardinality value from the Multiplicity dropdown listbox.
or
Type a cardinality value in the Multiplicity column.

12 Type the name of the class to which you want to associate the attribute in the Parent column.

13 Click OK.

The attribute is created for the class.

Creating an attribute from a class in a diagram

You can create an attribute from a class or an interface in a diagram in the same way.

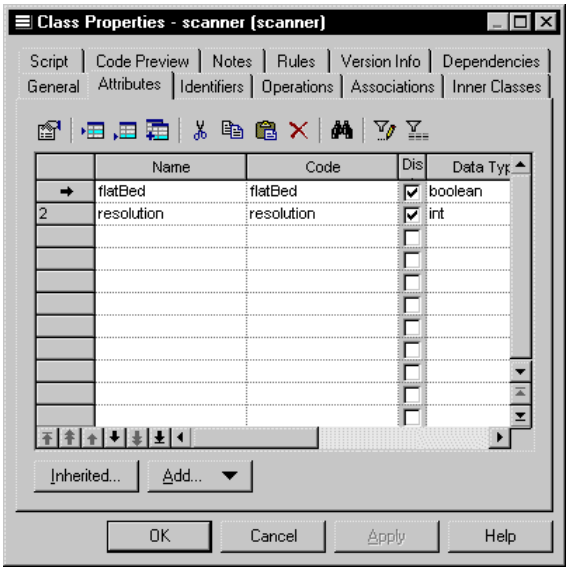
❖ **To create an attribute from a class in a diagram:**

1 Double-click a class in the model.

The class property sheet appears.

2 Click the Attributes tab.

The Attributes page appears. It lists attributes defined for the class.



3 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

4 Type an attribute name and an attribute code.

5 Click OK.

The attribute is created for the class and appears in the list of attributes for the class.

6 Click OK.

Modifying attribute properties

There are two approaches to modifying attribute properties:

- Modify the property sheet of an attribute
- Modify an entry in the list of attributes

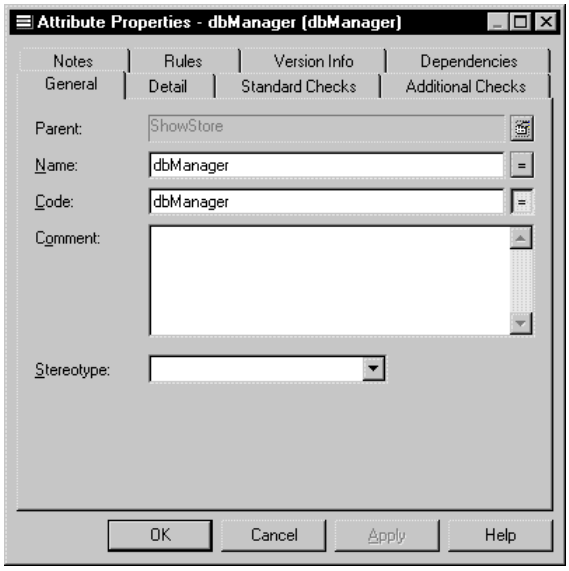
Modifying attribute properties from its property sheet

The attribute property sheet displays the definition of the attribute, which you can modify.

❖ **To modify attribute properties from its property sheet:**

1 Double-click the attribute in the model.

The attribute property sheet appears.

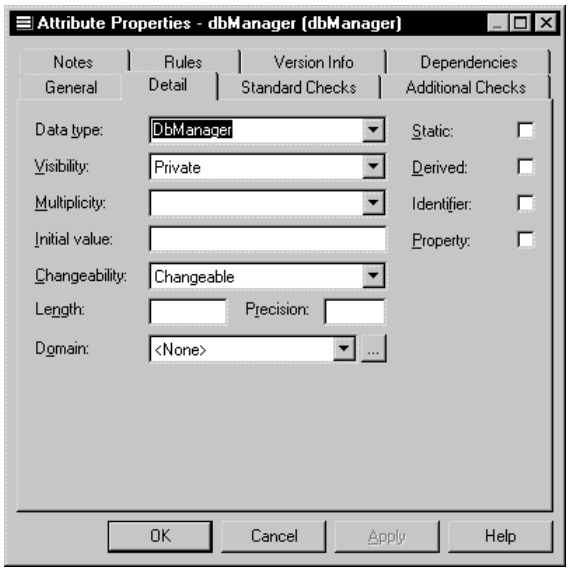


Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

2 Type or select attribute properties as required.

3 Click on the Detail tab.

The general properties of the attribute, in addition to those on the general page, appear.



4 Type or select attribute properties as required.

5 Click OK.

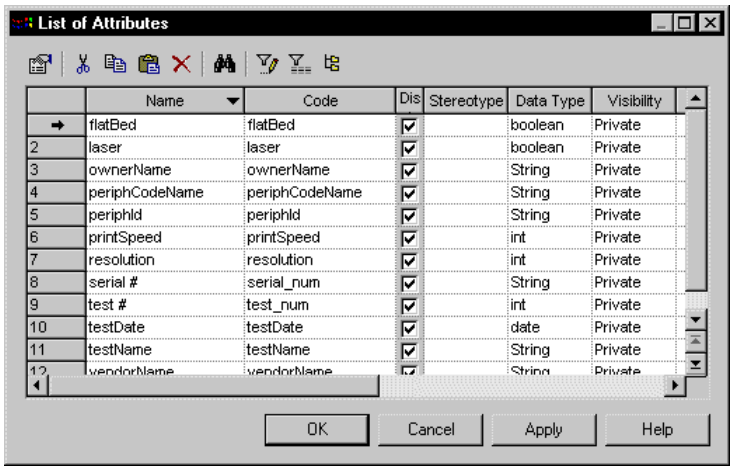
Modifying attribute properties from the list of attributes

The list of attributes includes all attributes attached to the current model. You can modify the attribute properties from the list.

❖ **To modify attribute properties from the list of attributes:**

1 Select Model➤Attributes.

The list of attributes appears.



2 Click the attribute that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the attribute directly in the list.

4 Click OK.

Attaching an attribute to a domain

If you attach an attribute to a domain, the domain supplies the data type and related data characteristics. It may also indicate check parameters, and business rules.

❖ **To attach an attribute to a domain:**

1 Double-click a class in the model.

The class property sheet appears.

2 Click the Attributes tab.

The Attributes page appears listing attributes associated with the class.

3 Click an attribute in the list.

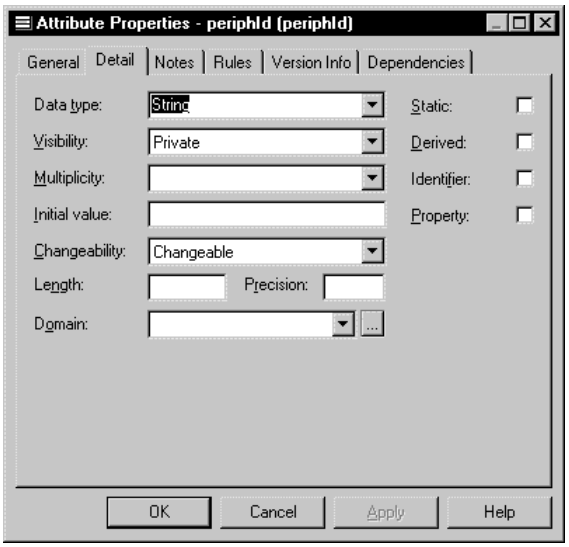
An arrow appears at the beginning of the line.

4 Click the Properties tool.
or
Double click the arrow at the beginning of the line.

The attribute property sheet opens to the General page.

> **Opening property sheets at last accessed page**
> Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

5 Click the Detail tab.



6 Select a domain from the Domain dropdown listbox at the bottom of the dialog box.

7 Click OK.

You return to the Attributes page. In the Data Type attribute, the domain's data type replaces the data type previously defined for the attribute.

8 Click OK.

Copying an attribute to another class

You can copy an attribute from one class and add it to another class. If the class already contains an attribute with the same name or code as the copied attribute, the copied attribute is renamed. For example the attribute PERIPHLD is renamed PERIPHLD2 when it is copied to a class which already contains an attribute PERIPHLD.

❖ **To copy an attribute to another class:**

1 Double-click a class in the model.

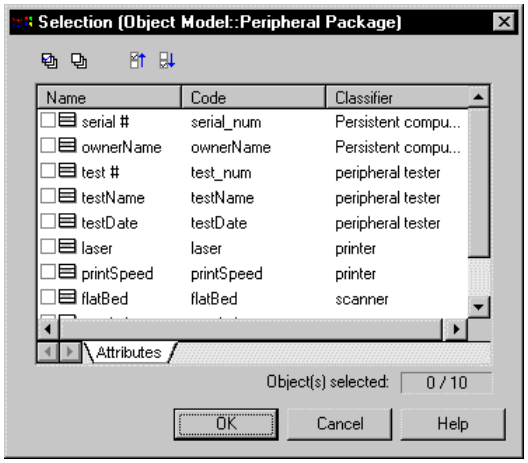
The class property sheet appears.

2 Click the Attributes tab.

The Attributes page appears.

3 Click the Add Attributes tool.

A selection box appears. It lists attributes attached to all other classes in the model.



4 Select one or more attributes in the list.

5 Click OK.

The copied attributes appear in the list of attributes for the current class.

6 Click OK.

Displaying text in attribute symbols

An attribute has the following display preferences:

Preference	Description
Show visibility	Displays the attribute as an icon, with markers, or using keywords
Show datatype	When selected, displays the datatype of the attribute in the attribute symbol
Show initial value	When selected, displays the initial value of the attribute in the attribute symbol

The visibility of an attribute in a class or an interface can be displayed in one of the following ways:

Visibility	When selected
Icon	Displays the attribute as an icon
Markers	Displays the visibility of the attribute as a marker: - (private), # (protected), + (public), or * (package)
Keywords	Displays the visibility of the attribute as a word: private, protected, public, or package

You modify the display preferences for an attribute in the Display Preferences dialog box.

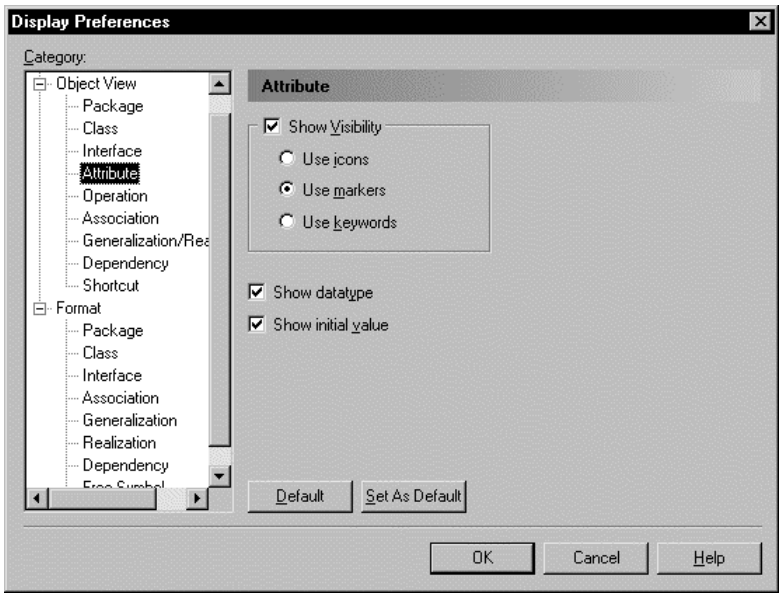
❖ **To modify the display preferences:**

1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Attribute.



4 Modify the attribute display preferences.

5 Click OK.

Defining identifiers

An identifier is a class attribute, or a combination of class attributes, whose values uniquely identify each occurrence of the class. An identifier is the OOM equivalent of a CDM identifier or a primary key or an alternate key in a PDM.

Each class must have at least one identifier. If a class has only one identifier, than it is designated by default as the primary identifier for the class. A primary identifier is the main identifier for a class.

You can attach attributes or business rules to an identifier.

Identifier properties

An identifier has the following properties:

Property	Description	Maximum length
Name	Name of the identifier	254
Code	Reference name for the identifier	254
Comment	Descriptive comment for the identifier	—
Class	Name of the class to which the identifier belongs to	254
Primary identifier	Indicates that the identifier is the primary identifier of the class. There can only be one primary identifier for a given class	—

An identifier definition also includes the following properties, which are defined on associated property sheets:

Property	Description
Attribute	Defines the characteristics of an identifier
Business rules	A rule that your business follows. Business rules guide and document the creation of a model

Creating an identifier

You can create an identifier from a class.

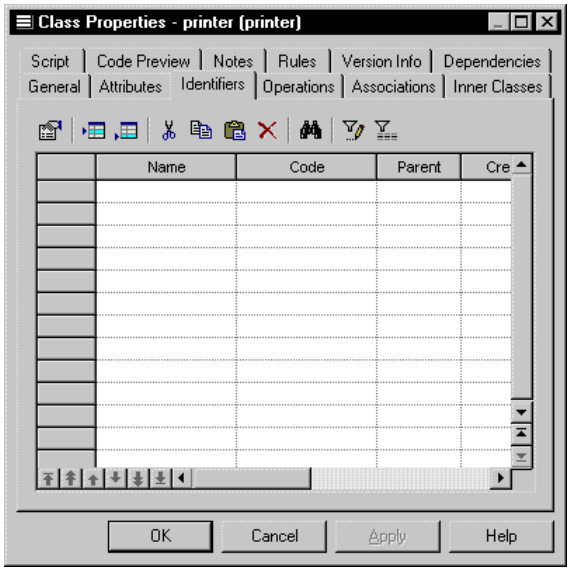
❖ **To create an identifier:**

1 Double-click a class in the model.

The class property sheet opens to the General page.

2 Click the Identifier tab.

The Identifier page appears.



3 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

4 Type a name and code for the identifier.

5 Click OK.

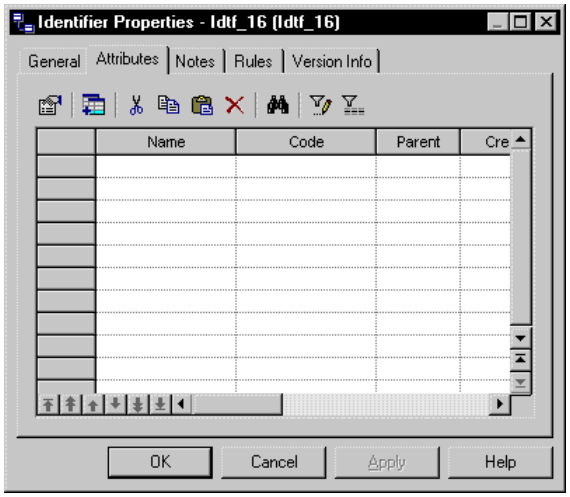
Adding attributes to an identifier

You can add attributes to an identifier.

❖ **To add attributes to an identifier:**

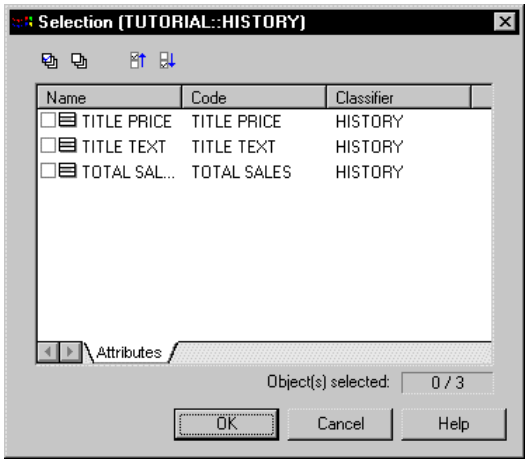
1 From the identifier property sheet, click the Attributes tab.

The Attributes page appears. It lists the attributes currently defined for the identifier.



2 Click the Add Attributes tool.

A list of attributes defined for the class appears.



3 Select checkboxes for one or more class attributes that you want to designate as an identifier.

4 Click OK in each of the dialog boxes.

Modifying identifier properties

There are two approaches to modifying identifier properties:

- Modify the property sheet of an identifier
- Modify an entry in the list of identifiers

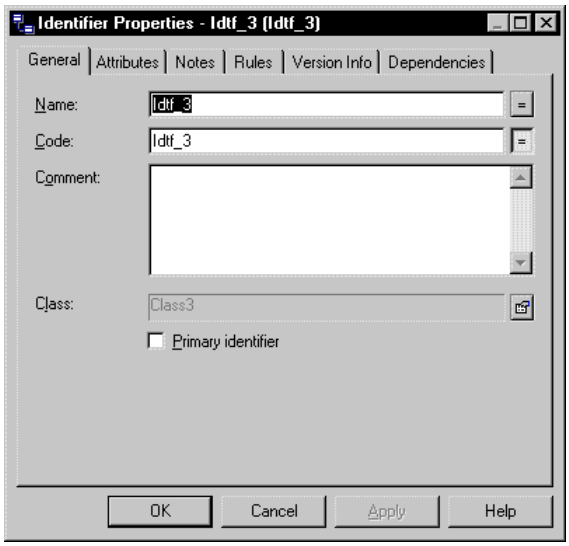
Modifying identifier properties from its property sheet

The identifier property sheet displays the definition of the identifier, which you can modify.

❖ **To modify identifier properties from its property sheet:**

1 Double-click the identifier in the model.

The identifier property sheet appears.



> **Opening property sheets at last accessed page**
> Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

2 Type or select identifier properties as required.

3 Click OK.

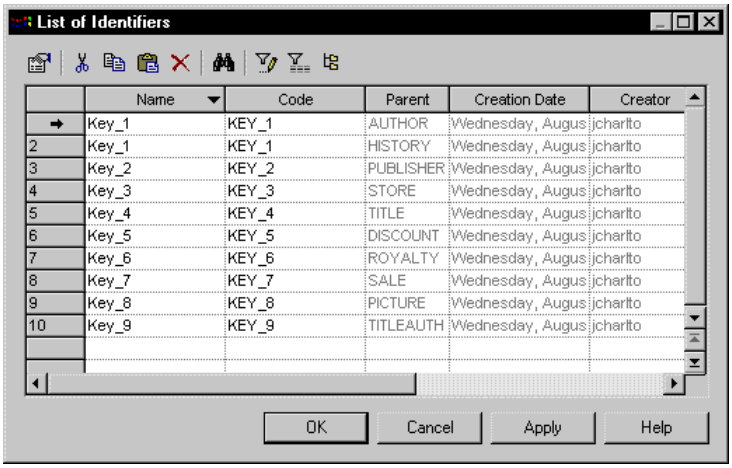
Modifying identifier properties from the list of identifiers

The list of identifiers includes all identifiers attached to the current model. You can modify the identifier properties from the list.

❖ **To modify identifier properties from the list of identifiers:**

1 Select Model➤Identifiers.

The list of Identifiers appears.



2 Click the identifier that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the identifier directly in the list.

4 Click OK.

Defining operations

An operation is a service that can be requested from an object to effect behavior. It has a name and a list of parameters. An operation is a specification of a transformation or query that an object may be called to execute.

Operation names within a class must be unique. You can give identical names to two or more operations only if they exist in different classes.

Operation properties

An operation has the following properties:

Property	Description	Maximum length
Parent	Object to which the operation belongs to	254
Name	Name of the operation	254
Code	Reference name for the operation	254
Comment	Descriptive comment for the operation	—
Stereotype	Subclassification of an operation derived from an existing one. Extends the semantics of an operation without changing it's structure	—
Return Type	A list of values returned by a call of the operation. If there are no values returned by the operation, the return type value is null	—
Visibility	Visibility of the operation, whose value denotes how it may be seen outside its enclosing name space	—
Event	Significant occurrence that has a location in time and space. An event can trigger a state transition	—
Abstract	Indicates that the operation cannot be instantiated and therefore has no direct instances	—
Final	Indicates that the operation cannot be redefined	—
Static	Defines the operation as static, meaning it cannot be modified	—

An operation definition also includes business rules, and parameters, which are defined on associated property sheets.

Analyzing operation properties

The following operation properties each have several default values from which you can select from:

- Visibility
- Stereotype

Visibility

Property	Visible
Private	Only to the operation itself
Protected	Only to the operation and its inherited objects
Package	To all objects contained within the same package
Public	To all objects

Stereotype

Stereotype	Description
constructor	Operation that creates and initializes an instance of a class

Creating an operation

There are three ways to create an operation:

- Create an operation symbol in the Browser
- Add a new operation to the list of operations
- Create an operation from a class in a diagram

Creating an operation from the Browser

❖ **To create an operation from the Browser:**

1 Right-click the Operations category in the Browser.

2 Select New from the contextual menu.

The property sheet of the operation appears.

3 Type an operation name and an operation code.

4 Click OK.

A new operation is created in the Operations category.

Creating an operation from the list of operations

❖ **To create an operation by inserting it in the list:**

1 Select Model➤Operations.

The list of operations appears.

> **Accessing the list of operations**
> The list of operations is accessible only from a diagram. If the current diagram is of a package, the list contains all the operations that exist in the package. If the current diagram is of the model, the list contains all the operations that exist in the model.

2 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

3 Type an operation name and an operation code.

4 Click the Stereotype column.

5 Select a stereotype from the Stereotype dropdown listbox.
or
Type a stereotype in the Stereotype column.

6 Click the Return Type column.

7 Select a return type from the Return Type dropdown listbox.
or
Type a return type in the Return Type column.

8 Click the Visibility column.

9 Select a value from the Visibility dropdown listbox.

10 Click OK.

The operation is created for the class.

Creating an operation from a class in a diagram

You can create an operation from a class or an interface in a diagram in the same way.

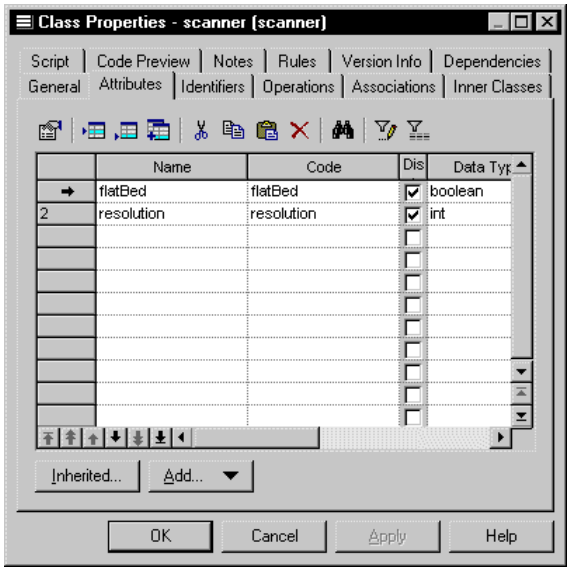
❖ **To create an operation from a class in a diagram:**

1 Double-click a class in the model.

The class property sheet appears.

2 Click the Operations tab.

The Operations page appears. It lists operations defined for the class.



3 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

4 Type an operation name and an operation code.

5 Click OK.

The operation is created for the class and appears in the list of operations for the class.

6 Click OK.

Modifying operation properties

There are two approaches to modifying operation properties:

- Modify the property sheet of an operation
- Modify an entry in the list of operations

Modifying operation properties from its property sheet

The operation property sheet displays the definition of the operation that you can modify.

❖ **To modify operation properties from its property sheet:**

1 Double-click a class in the model.

The class property sheet appears.

2 Click the Operations tab.

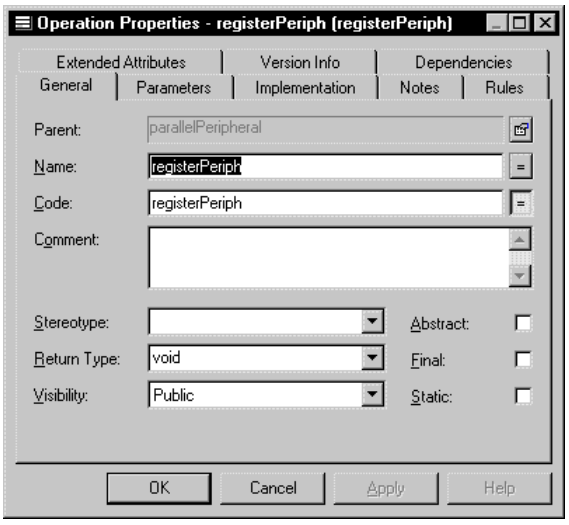
The Operations page appears. It lists operations associated with the class.

3 Click the operation that you want to define.

An arrow appears at the beginning of the line.

4 Click the Properties tool.
or
Double-click the arrow at the beginning of the line.

The operation property sheet opens to the General page.



Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

5 Type or select operation properties.
or
Click on a page tab.
Type or select operation properties as required.

6 Click OK.

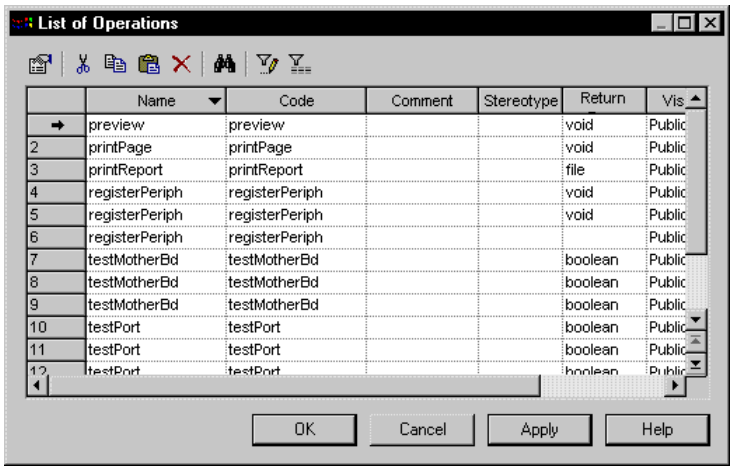
Modifying operation properties from the list of operations

The list of operations includes all operations attached to the current model. You can modify the operation properties from the list.

❖ **To modify operation properties from the list of operations:**

1 Select Model➤Operations.

The list of operations appears.



2 Click the operation that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the operation directly in the list.

4 Click OK.

Adding constructors and destructors to a class

A constructor is a special type of operation that creates and initializes an instance of a class.

A destructor, on the other hand, is the complement of a constructor in that it is an operation that it deinitializes and destroys the class instance. You can only create a default destructor for a given class, and only if the current object language for the OOM is PowerBuilder.

You can create constructors and destructors only from a class, from the Operations page of the class property sheet.

Constructor and destructor names are assigned automatically by PowerDesigner and you cannot modify them.

You cannot declare a Return Type for a constructor.

You can create two types of constructor for a given class:

Default
Copy

A Default constructor has no parameters.

Adding a Default constructor and destructor to a class

You can define only one Default constructor and one Default destructor (PowerBuilder only) for any given class.

If the current object language of the OOM is Analysis or Java, you can create only one Default constructor and no Default destructor for a class. The constructor has the same name as the class to which it belongs.

If the current object language of the OOM is PowerBuilder, you can create one Default constructor and one Default destructor for a class. The constructor has the name "constructor" and destructor has the name "destructor".

❖ **To add a Default constructor and destructor to a class:**

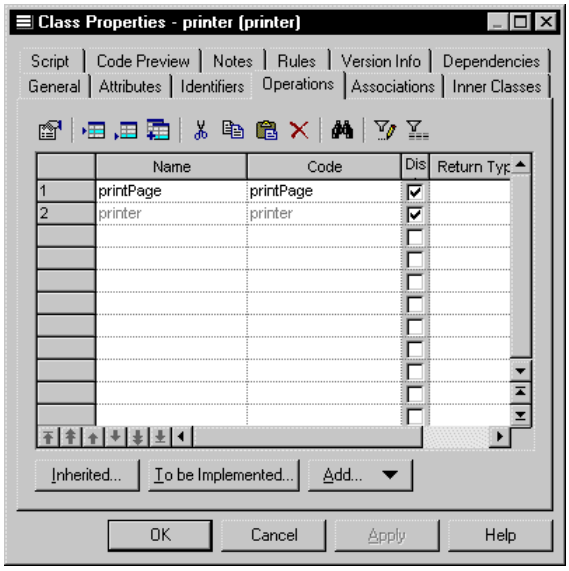
1 Double-click a class in the model.

2 Click the Operations tab.

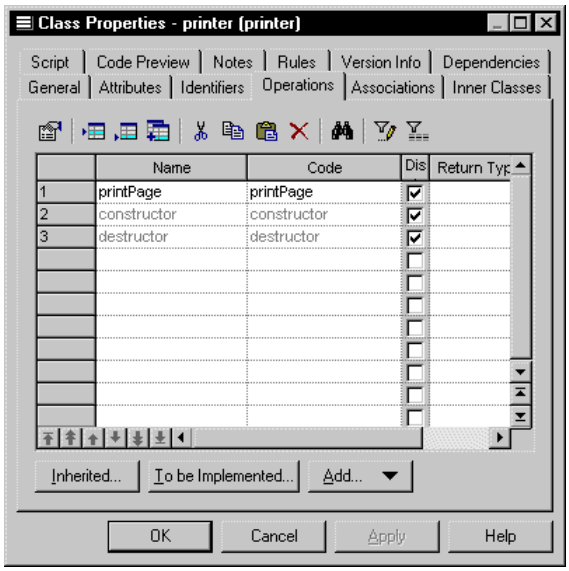
3 Click the Add button.

4 Select Default Constructor/Destructor from the dropdown list.

If the current object language of the OOM is Analysis or Java, a Default constructor is created at the end of the list of operations for the class. It has the same name as the class to which it belongs:



If the current object language of the OOM is PowerBuilder, a Default constructor and a Default destructor is created at the end of the list of operations for the class. They have the names constructor and destructor:



Adding a Copy constructor to a class

The body of a Copy constructor contains a copy of the attributes of the class that exist at the moment of the creation of the constructor.

When you create a Copy constructor, it has the same as that of the class, prefixed by the keyword *new*. It has a default parameter that you can modify, or you can add other parameters.

If the class is an instance of another class, the attribute names within the body of the Copy constructor operation are the same as those in the parent class.

You can define only one Copy constructor to any given class.

❖ **To add a Copy constructor to a class:**

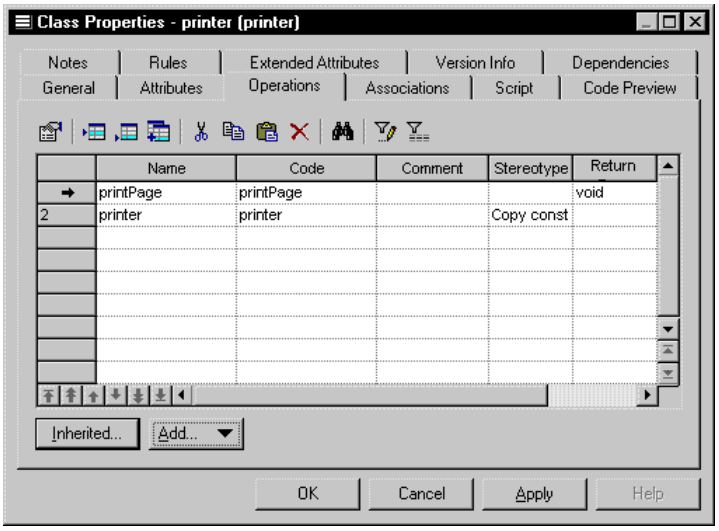
1 Double-click a class in the model.

2 Click the Operations tab.

3 Click the Add button.

4 Select Copy Constructor from the dropdown list.

A Copy constructor is created at the end of the list of operations for the class. It has the same name as the class to which it belongs.



Adding operations to a class

You can add an operation to a class in one of the following two ways:

- Add a duplicate operation
- Add an operation from a parent class

Adding a duplicate operation to a class

A duplicate operation is an operation that creates and initializes an instance of a class within the class.

When you create a duplicate operation, it has the name Duplicate, which you can modify.

You can define only one duplicate operation to any given class.

❖ **To add a duplicate operation to a class:**

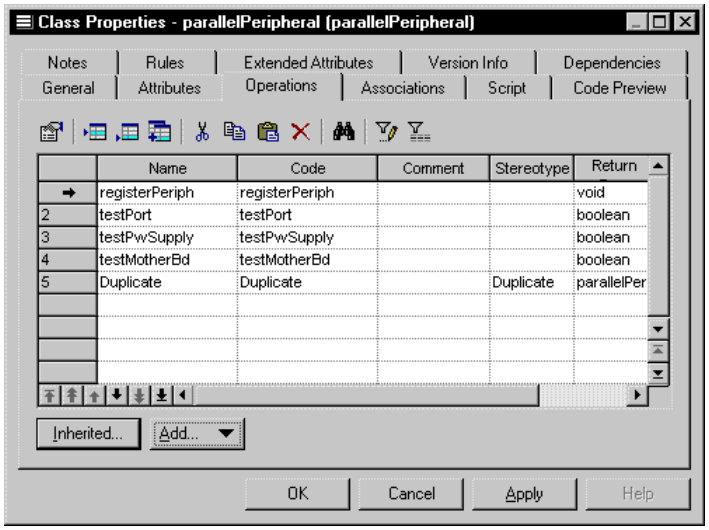
1 Double-click a class in the model.

2 Click the Operations tab.

3 Click the Add button.

4 Select Duplicate Operation from the dropdown list.

A duplicate operation, the name Duplicate, is created at the end of the list of operations for the class.



Adding an operation from a parent class

You can add to a class an operation that belongs to a parent class. The new operation has the same signature (name and parameters) as the original operation, but does not have its other properties.

Once you add an operation to a class in this way, you can modify only the code implementation of the operation. You cannot modify the signature of the operation.

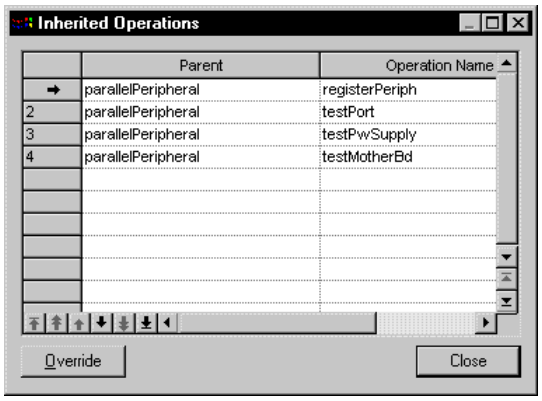
❖ To add an inherited operation to a class:

1 Double-click a class that is linked to a parent class in the model.

2 Click the Operations tab.

3 Click the Inherited button.

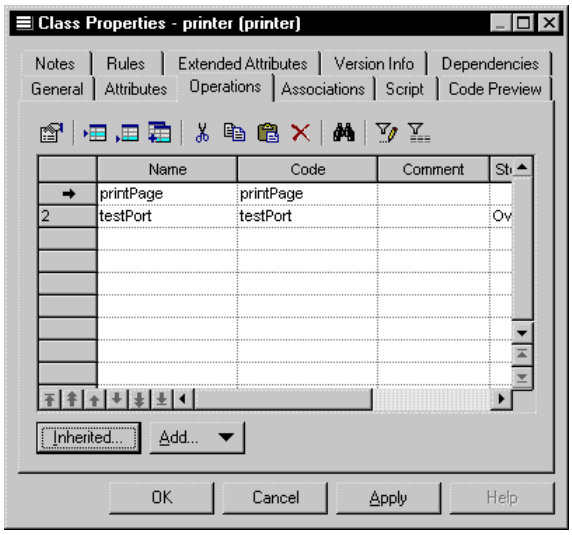
The Inherited Operations window appears showing the operations that belong to all the parent classes of the class.



4 Select an operation.

5 Click the Override button.

A copy of the operation is added to the list operations for the class.



Adding Getter and Setter operations to a class

Getter or a Setter operations are special types of operations that you create for an attribute. You create a Getter or a Setter operation type from the list of attributes of a class. For each attribute, you can create one Getter, one Setter operation, or both a Getter and a Setter operation.

You create Getter or a Setter operations for sending a receiving data values between fields.

Operation	Description
Getter	Returns a value from a field
Setter	Puts a value into a field

❖ **To add a Getter and a Setter operation to a class from an attribute:**

1 Double-click a class in the model.

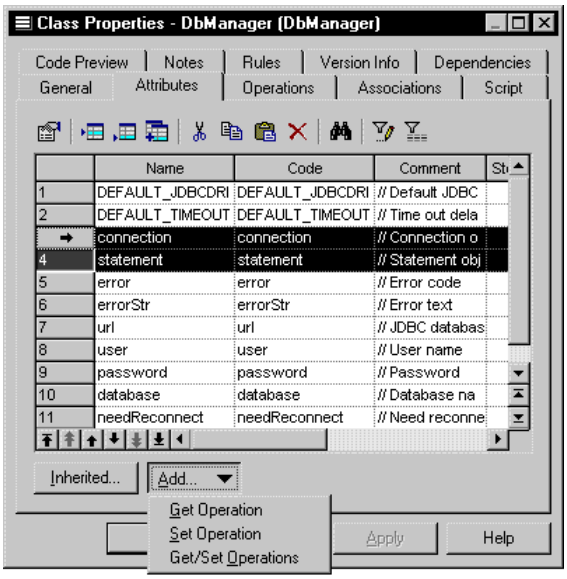
The class property sheet appears.

2 Click the Attributes tab.

The Attributes page appears.

3 Select one or more attributes.

4 Click the Add button.

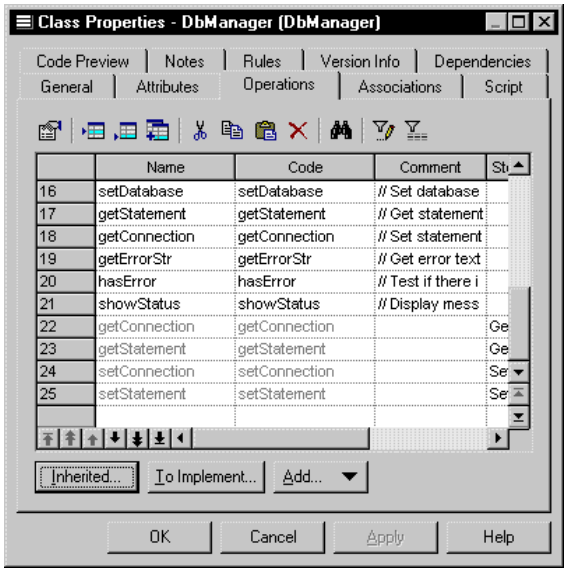


5 Select Get/Set Operations from the dropdown listbox.

The operations are created for the attributes. You can visualize them in the list of operations of the class.

6 Select the Operations tab.

The newly created operations appear at the bottom of the list of operations for the class. They are grayed indicating that their names can not be modified.



7 Click OK.

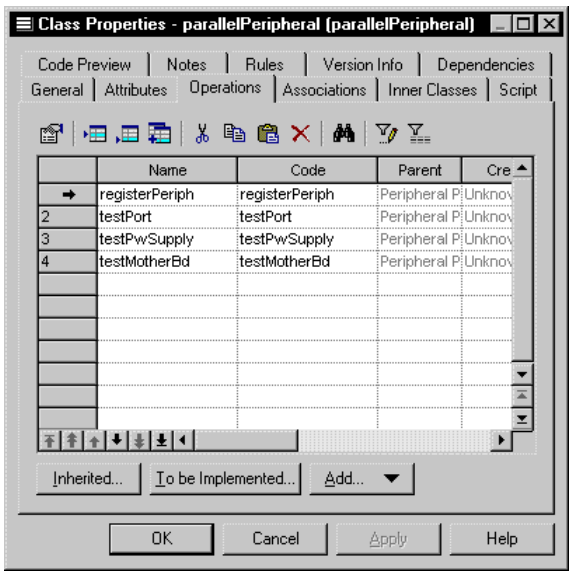
Creating an implementation operation

When you create a realization link between a class and an interface in which the class implements the interface, you create an operation in the class that implements the interface.

❖ **To create an implementation operation:**

1 Double-click a class that is linked to an interface by a realization link.

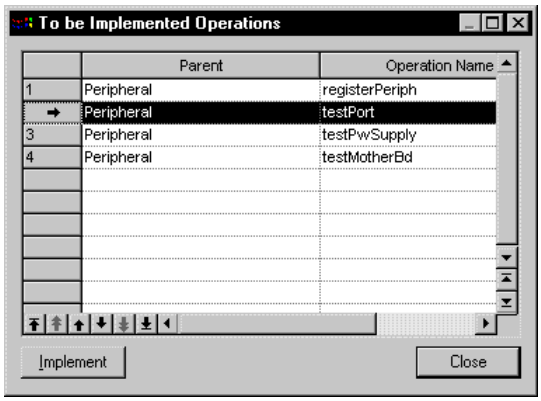
2 Click the Operations tab.



3 Click the To be implemented button.

The To Be Implemented Operations window appears. It contains a list of all the operations of the interface that can be implemented from the class.

4 Select an operation from the list.

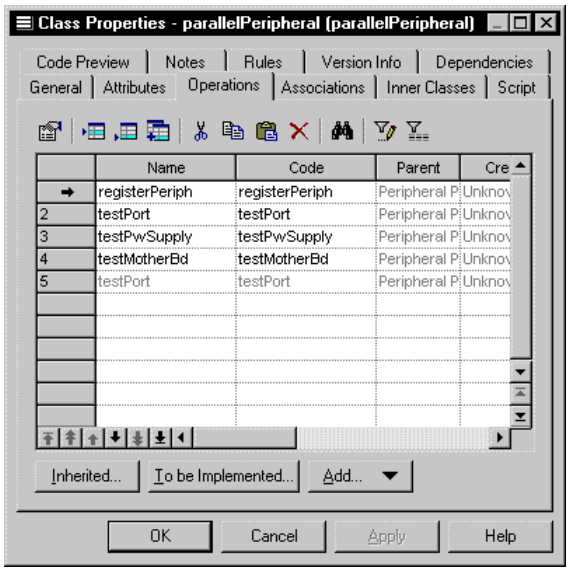


5 Click the Implement button.

A copy of the operation is created in the class.

6 Click Close.

The newly created operation is added to the end of the list of operations for the class. It is grayed, indicating that its name cannot be modified.



7 Click OK.

Modifying the code of an implementation operation

You can modify the code of an implementation operation from the Implementation page of the operation property sheet.

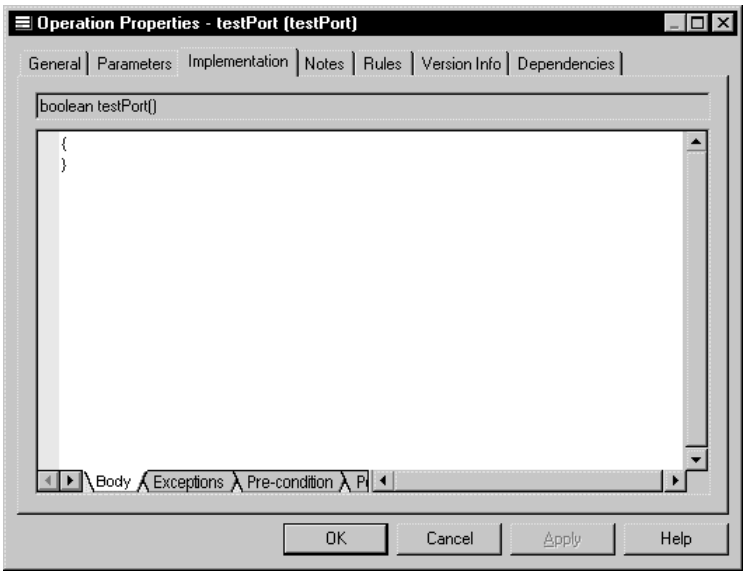
For information on how to create an implementation operation, see the section Creating an implementation operation.

❖ To modify the code of an implementation operation:

1 Double-click an implementation operation in the list of operations of a class that implements an interface.

2 Click the Implementation tab.

The Implementation page appears.



3 Type or modify code directly in the window.
or
Click a tab at the bottom of the edit window and type or modify code.

4 Click OK.

Copying an operation to another class

You can copy an operation from one class and add it to another class. If the class already contains an operation with the same name or code as the copied operation, the copied operation is renamed. For example the operation testPort is renamed testPort2 when it is copied to a class which already contains an operation testPort .

❖ **To copy an operation to another class:**

1 Double-click a class in the model.

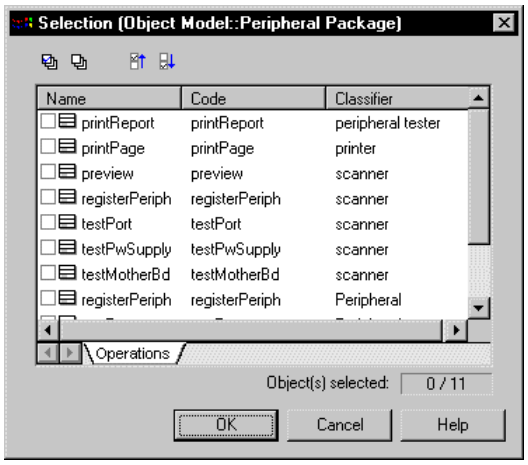
The class property sheet appears.

2 Click the Operations tab.

The Operations page appears.

3 Click the Add Operations tool.

A selection box appears. It lists operations attached to all other classes in the model.



4 Select one or more operations in the list.

5 Click OK.

The copied operations appear in the list of operations for the current class.

6 Click OK.

Displaying text in operation symbols

An operation has the following display preferences:

Preference	Description
Show visibility	Displays the operation as an icon, with markers, or using keywords
Show return type	When selected, displays the return type of the operation in the operation symbol
Show parameters	When selected, displays the parameters of the operation in the operation symbol

The visibility of an operation in a class or an interface can be displayed in one of the following ways:

Visibility	When selected
Icon	Displays the operation as an icon
Markers	Displays the visibility of the operation as a marker: - (private), # (protected), + (public), or * (package)
Keywords	Displays the visibility of the operation as a word: private, protected, public, or package

You modify the display preferences for an operation in the Display Preferences dialog box.

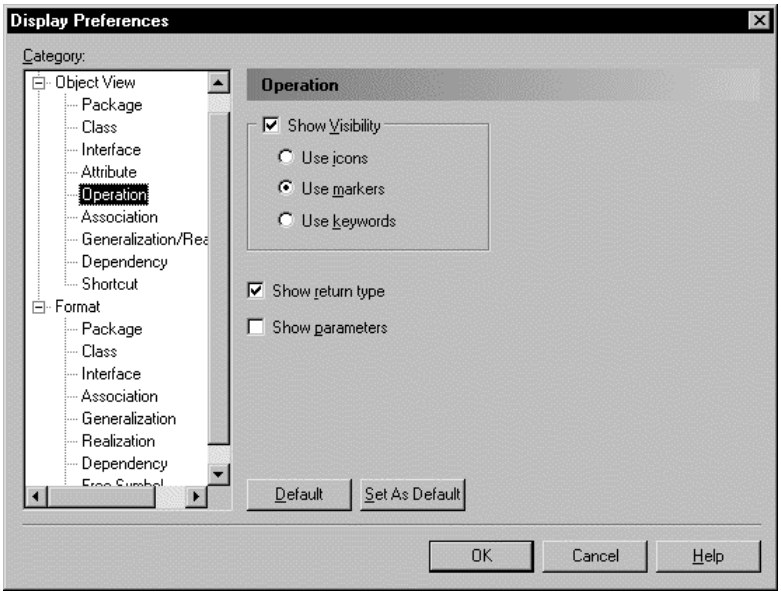
❖ **To modify the display preferences:**

1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Operation.



4 Modify the operation display preferences.

5 Click OK.

Defining parameters

A parameter is a specification of a variable that can be changed, passed, or returned. Parameters are used only for operations.

A parameter always has a direction, which indicates the flow of information.

Parameter properties

A parameter has the following properties:

Property	Description	Maximum length
Parent	Operation to which the parameter belongs to	254
Name	Name of the parameter	254
Code	Reference name for the parameter	254
Comment	Descriptive comment for the parameter	—
Data Type	Set of instances that share the same operations, abstract attributes, and relationships, and semantics	—
Array	Used in generating and reverse engineering for Java and PowerBuilder. When selected, arranges attributes into table format	—
Parameter Type	Direction of information flow of the parameter	254

Direction

The choice you make in the Direction dropdown listbox indicates what value is returned when the parameter is called by the operation during the execution process.

You can set the following values for the direction:

Value	Description
In	Input parameter passed by value. The final value may not be modified and information is not available to the caller
In\Out	Input parameter that may be modified. The final value may be modified to communicate information to the caller
Out	Output parameter. The final value may be modified to communicate information to the caller

Creating a parameter

You can create parameters only from an operation. You create parameters from the Parameters page in the operation property sheet.

❖ **To create a parameter:**

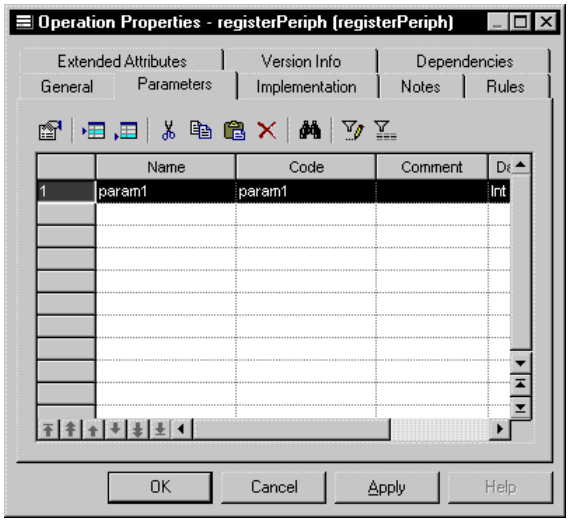
1 Double-click an operation in the model.

2 Click the Parameters tab.

The Parameters page appears.

3 Click the Add a Row tool
or
Click the first row.

A parameter is created.

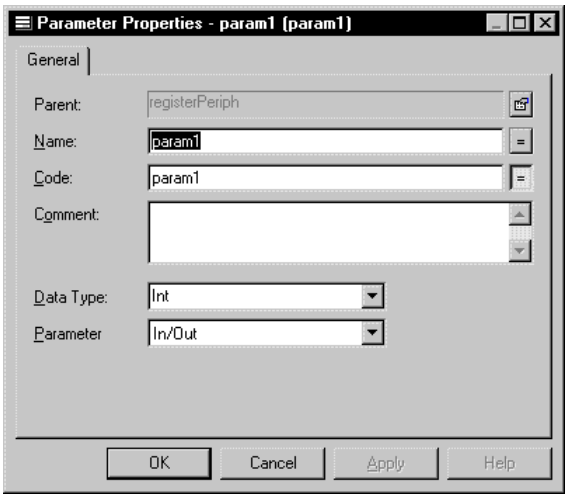


4 Double-click the arrow at the beginning of the line.

A confirmation box appears asking you if you to confirm the creation of the parameter.

5 Click OK

The parameter property sheet opens to the General page.



Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

6 Type a name and code.

7 Select a value from the Direction dropdown listbox.

In/Out is the default direction value.

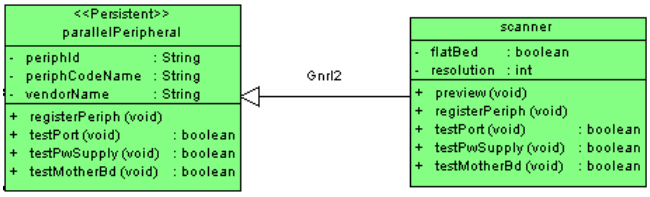
8 Click OK in all the open dialog boxes.

Defining generalizations

A generalization relationship between classes shows that the subclass shares the structure or behavior defined in one or more superclasses. You use a generalize to show a "is-a" relationship between classes.

You can create a generalization only from one class to another class, or from one interface to another interface. You can also create a generalization between a shortcut of a class to a class, or between a shortcut of a interface to a interface. If the link is oriented, only the parent object can be the shortcut.

You can create only one generalization between two given objects.



Generalization properties

A generalization has the following properties:

Property	Description	Maximum length
Name	Name of the generalization	254
Code	Reference name for the generalization	254
Comment	Descriptive comment for the generalization	—
Parent	Class or interface to which the generalization belongs to	254
Child	Class or interface that belongs to the generalization	254
Stereotype	Subclassification of a generalization derived from an existing one. Extends the semantics of a generalization without changing it's structure	—
Visibility	Visibility of the generalization, whose value denotes how it may be seen outside its enclosing name space	—
Virtual	Used in generation (except for Java and PowerBuilder)	—

Analyzing generalization properties

The following generalization properties each have several default values from which you can select from:

- Visibility
- Stereotype

Visibility

Property	Visible
Private	Only to the generalization itself
Protected	Only to the generalization and its inherited objects
Package	To all objects contained within the same package
Public	To all objects

Stereotype

Stereotype	Description
implementation	Specifies that the child object inherits the implementation of the parent object but that it does not make public its interfaces, nor support them, thus violating its substitutability

Creating a generalization

You can create a generalization only from a class to a class, or from an interface to an interface.

❖ **To create a generalization:**

1 Click the Generalization tool in the palette toolbar.

2 Drag the generalization from the child class to the parent class, or from the child interface to the parent interface.

The link appears between the two objects.

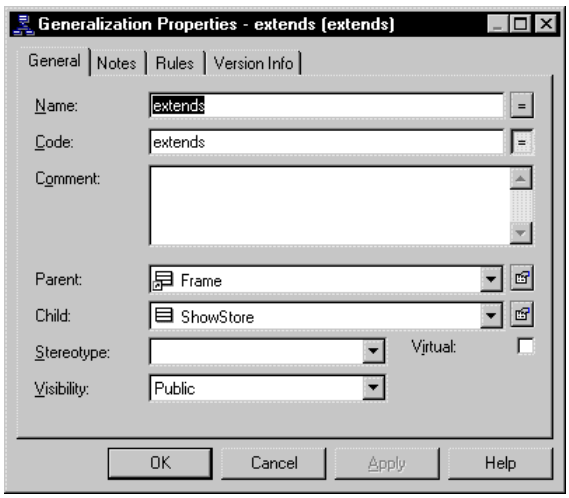
> **Dragging a generalization to a different class**
> You can change the class or interface at either end of a generalization by clicking the generalization to select it, pressing down CTRL, and dragging one of the attach points to a different class or interface.

3 Click the Pointer tool in the palette toolbar.
or
Click the right mouse button.

You release the Generalization tool.

4 Double-click the new generalization in the model.

The generalization property sheet opens to the General page.



> **Opening property sheets at last accessed page**
> Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

5 Type a generalization name and a generalization code.

6 Click OK.

Modifying generalization properties

There are two approaches to modifying generalization properties:

- Modify the property sheet of the generalization
- Modify an entry in the list of generalizations

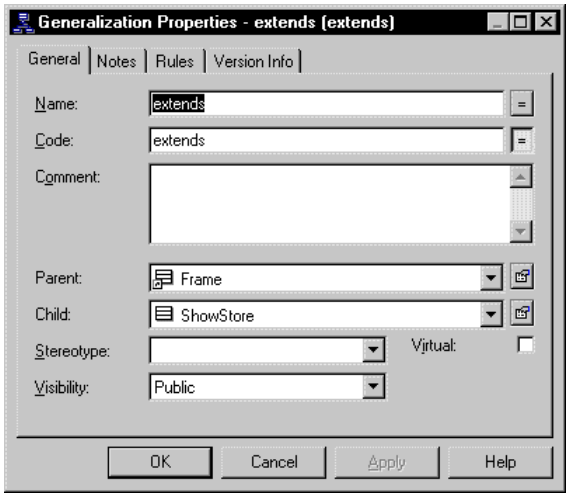
Modifying generalization properties from its property sheet

The generalization property sheet displays the definition of the generalization, which you can modify.

❖ **To modify generalization properties from its property sheet:**

1 Double-click the generalization in the model.

The generalization property sheet appears.



2 Type or select generalization properties.
or
Click on a page tab.
Type or select generalization properties as required.

3 Click OK.

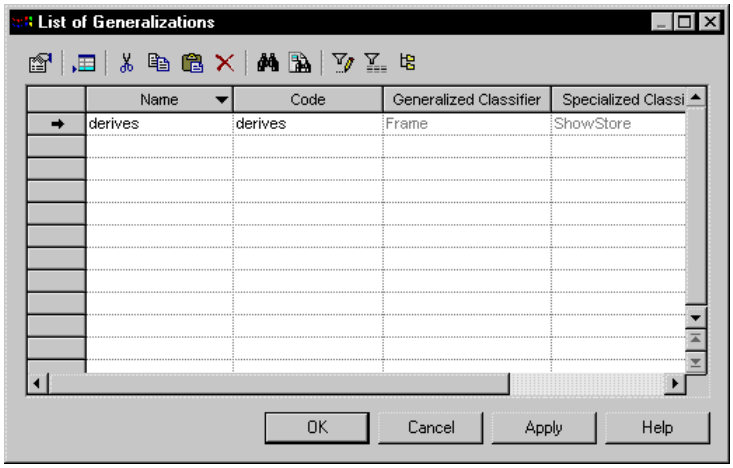
Modifying generalization properties from the list of generalizations

The list of generalizations includes all generalizations attached to the current model. You can modify the generalization properties from the list.

❖ **To modify generalization properties from the list of generalizations:**

1 Select Model➤Generalizations.

The list of generalizations appears.



2 Click the generalization that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the generalization directly in the list.

4 Click OK.

Displaying text in generalization symbols

A generalization has the following display preferences:

Preference	When selected
Show name	Displays the name of the generalization
Show stereotypes	Displays the stereotypes of the generalization
Show constraints	Displays the constraints (business rules) of the generalization

You modify the display preferences for a generalization in the Display Preferences dialog box.

❖ **To modify the display preferences:**

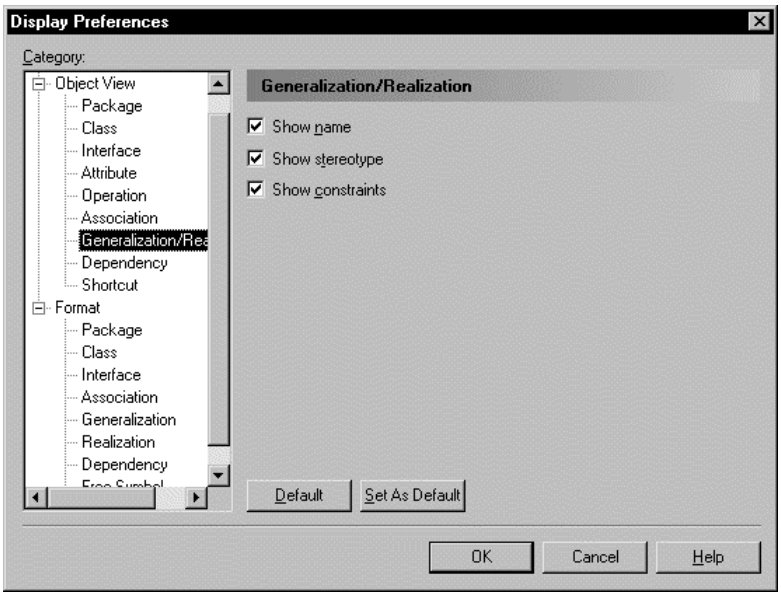
1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Generalization.

The Generalization display preferences appears.



4 Modify the generalization display preferences.

5 Click OK.

Defining associations

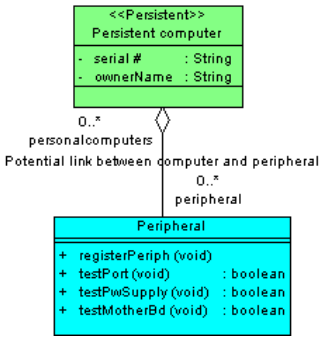
An association represents a structural relationship between objects of different classes. An association is drawn as a solid line between pairs of classes.

You can define an association between two classes, or between a class and an interface.

An association may have a name to clarify the nature of the relationship between the associated classes. The name of the association is usually omitted when end names are used.

Association Ends

Each end of an association may have a name that describes the role that each class plays as viewed by the opposite class.



Association properties

An association has the following properties:

Property	Description	Maximum length
Name	Name of the association	254
Code	Reference name for the association	254
Comment	Descriptive comment for the association	—
Stereotype	Subclassification of an association derived from an existing one. Extends the semantics of an association without changing it's structure	—
Aggregation/ composition	Indicates whether the association is an aggregation or a composition	—
Role A	One end of an association. Each role can have a name and a cardinality. You can make a role navigable or not, or change its visibility	254
Role B	One end of an association. Each role can have a name and a cardinality. You can make a role navigable or not, or change its visibility	254
Multiplicity	Minimum and maximum number of instances that the association can have	—
Ordering	Indicates that the association is included in the ordering that sorts the list of associations by their order of creation.	—
Navigable	Indicates whether or not information can be transmitted between the two objects that are linked by the relationship	—
Changeability	Specifies if the value of the association can be modified or not once the object has been initialized	—
Visibility	Visibility of the association, whose value denotes how it may be seen outside its enclosing name space	—

Creating an association

You can create an association between two classes or between a class and an interface:

- in a diagram
- from the list of associations
- from the Browser

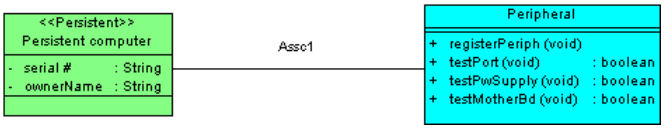
> **Creating an association outside of a diagram**
> When you create an association from the list of associations or from the Browser, you must select the two classes that are linked by the association.

❖ **To create an association in a diagram:**

1 Click the Association tool in the palette toolbar.

2 Drag the association link from one class to another.

The link appears between the two classes.



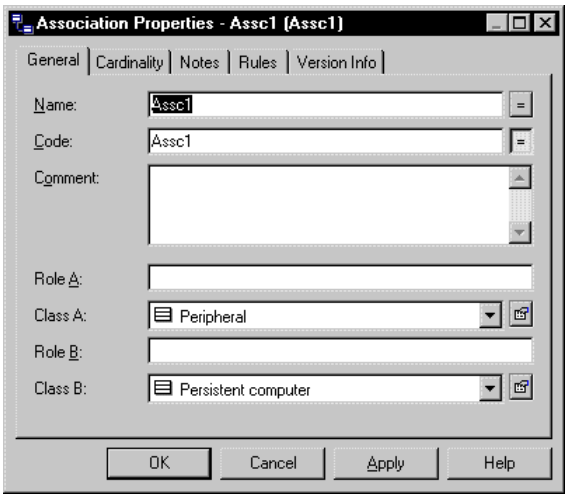
> **Dragging an association to a different class**
> You can change the class at either end of an association by clicking the association to select it, pressing down CTRL, and dragging one of the attach points to a different class.

3 Click the Pointer tool in the palette toolbar.
or
Click the right mouse button.

You release the Association tool.

4 Double-click the new association in the model.

The association property sheet opens to the General page.



Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

5 Type an association name and an association code.

6 Type a name and code for Role A.

7 Type a name and code for Role B.

8 Click OK.

Analyzing cardinality properties

Each end of an association is called a **Role**. A role has its own properties and cardinality. You can define one of the two roles as being an aggregation or a composition.

Association role properties

You can define the following properties for each of the two roles of an association:

- Multiplicity
- Ordering
- Visibility

Multiplicity

The cardinality of each of the two roles of an association is called the multiplicity. The multiplicity indicates the maximum and minimum number of values each role can have.

Cardinality	Number of instances
0..0	None
0..1	None or one
0..*	None to infinity
1..1	One to one
1..*	One to infinity
*	Infinity

Ordering

You can define the ends of an association as being ordered or sorted.

Property	Indicates
Sorted	That the set of objects at the end of an association are arranged according to the way they are defined in the model
Ordered	That the set of objects at the end of an association are arranged according to in a specific order
Unordered	That the end of the association is neither sorted nor ordered

Visibility

The visibility of an association refers to the way in which it can be seen by other objects. An association that is visible to another object may influence the structure or behavior of the object, or similarly, its own properties may be affected by the other object.

Property	Visible
Private	Only to the association itself
Protected	Only to the association and its inherited objects
Package	To all objects contained within the same package
Public	To all objects

Aggregation/composition of an Association

You can define one of the roles of an association as being either an aggregation or a composition in the Aggregation/composition group box.

Property	Description
Aggregation	An form of association that specifies the relationship between two classes of the same level
Composition	A form of aggregation of an association in which the class attached to the association role may be a part of only one composite at a time
Container	Specifies which of the two roles is an aggregation or a composition
Indicator	Indicates that the association is an aggregation or a composition

Changing an association into an associative class

You can transform an association into an associative class linked by two associations. Next, you can attach class attributes to this associative class, that you could not attach to the association.

The associative class gets the name and code of the association. You can define cardinality properties for each of the two associations created between the new class and the two existing classes.

❖ **To change an association into an associative class:**

1 Right-click an association.

The association context menu appears.

2 Select Change to Class from the context menu.

An associative class with two associations replaces the association. The associative class takes the name of the original association.

Modifying association properties

There are two approaches to modifying association properties:

- Modify the property sheet of an association
- Modify an entry in the list of associations

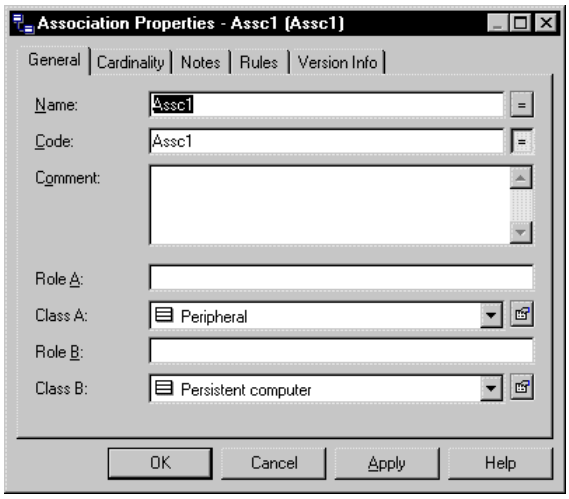
Modifying association properties from its property sheet

The association property sheet displays the definition of the association, which you can modify.

❖ **To modify association properties from its property sheet:**

1 Double-click the association in the model.

The association property sheet appears.



2 Type or select association properties.
or
Click on a page tab.
Type or select association properties as required.

3 Click OK.

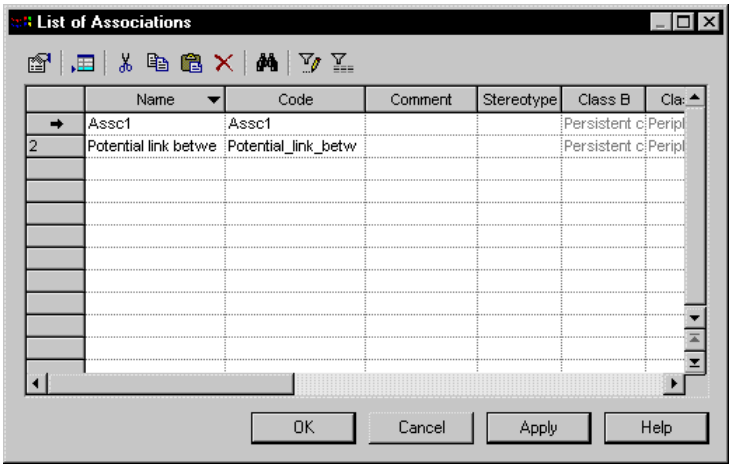
Modifying association properties from the list of associations

The list of associations includes all associations attached to the current model. You can modify the association properties from the list.

❖ **To modify association properties from the list of associations:**

1 Select Model➤Associations.

The list of associations appears.



2 Click the association that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the association directly in the list.

4 Click OK.

Modifying cardinality properties

The Cardinality page of the association property sheet displays the definition of the roles, which you can modify.

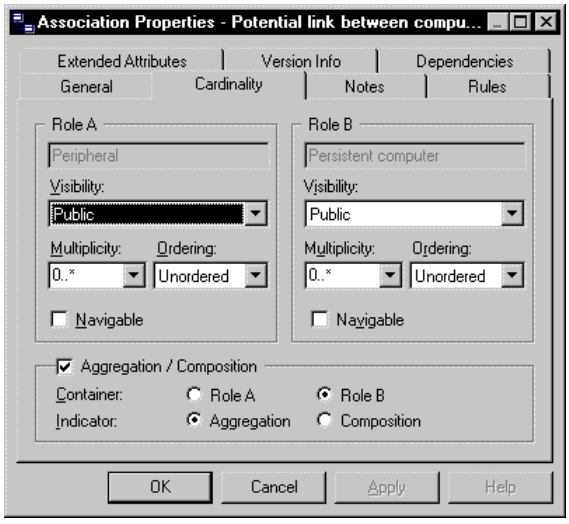
❖ **To modify association cardinality properties:**

1 Double-click the association in the model.

The association property sheet appears.

2 Click the Cardinality tab.

The Cardinality page appears.



3 Select properties for role A and for Role B.

4 Select the Aggregation/Composition checkbox.

5 Select Aggregation/Composition group box options.

6 Click OK.

Displaying text in association symbols

You can define the following display preferences for an association:

Preference	Description
Show name	When selected, displays the name of the association
Show constraints	When selected, displays the constraints (business rules) of the association
Show role names	When selected, displays the name of the association roles
Name attached to its symbol	When selected, the name of the role remains attached to it when it is moved. When not selected, role name can be moved anywhere in the model
Multiplicity	Displays the cardinality of the relationship. You can choose between showing the actual number of instances (String)or the symbol at the end of the relationship (Symbol)

You modify the display preferences for an association in the Display Preferences dialog box.

❖ **To modify the association display preferences:**

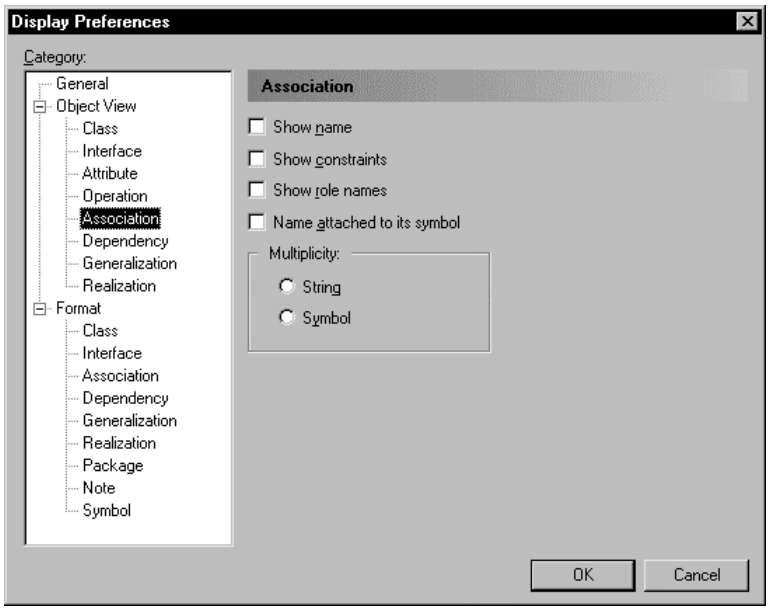
1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Association.

The Association display preferences appears.



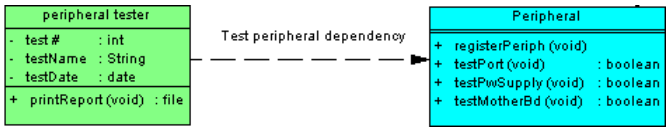
4 Modify the association display preferences.

5 Click OK.

Defining dependencies

A dependency is a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).

The dependency relationship indicates that one class or interface in a component diagram uses the services or facilities of another class or interface.



Dependency properties

A dependency has the following properties:

Property	Description	Maximum length
Name	Name of the dependency	254
Code	Reference name for the dependency	254
Comment	Descriptive comment for the dependency	—
Independent	Indicates that the two objects linked by the dependency are totally independent from one another	254
Dependant	Indicates that the two objects linked by the dependency are dependant, and therefore any changes in one object will affect the other	254
Stereotype	Pre-defined or user defined instance of the dependency	—
Parent	Name of the parent object of the dependency	254
Child	Name of the child object of the dependency	254

Analyzing dependency properties

Stereotype

You can select a stereotype for a dependency from the following several default values:

Stereotype	Description
access	Public contents of the target package that can by accessed by the source package
bind	Source object that instantiates the target template using the given actual parameters
call	Source operation that invokes the target operation
derive	Source object that can be computed from the target
friend	Source object that has special visibility towards the target
import	Everything that is declared as public in the target object becomes visible to the source object, as if it were part of the source object definition
include	Source use case incorporates the behavior of another use case at a location that is specified by the source
instantiate	Specifies that operations on the source class create instances of the target class
refine	Degree of abstraction of the source object is finer than that of the target object
trace	Specifies that there is an historical link between the source object and the target object
use	Specifies that the semantics of the source object are dependent on the semantics of the public part of the target object

Creating a dependency

You can create a dependency between two classes, two interfaces, or between a class and an interface. You create dependencies in a diagram.

❖ **To create a dependency:**

1 Click the Dependency tool in the palette toolbar.

2 Drag the dependency link from the child class or interface to the parent class or interface.

The link appears between the two objects.

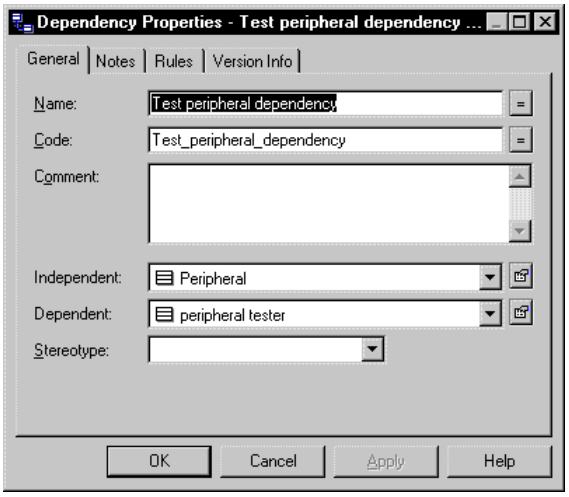
Dragging a dependency to a different class
You can change the class at either end of a dependency by clicking the dependency to select it, pressing down CTRL, and dragging one of the attach points to a different class.

3 Click the Pointer tool in the palette toolbar.
or
Click the right mouse button.

You release the Dependency tool.

4 Double-click the new dependency in the model.

The dependency property sheet opens to the General page.



Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

5 Type a dependency name and a dependency code.

6 Select a stereotype from the dropdown listbox.

7 Click OK.

Modifying dependency properties

There are two approaches to modifying dependency properties:

- Modify the property sheet of a dependency
- Modify an entry in the list of dependencies

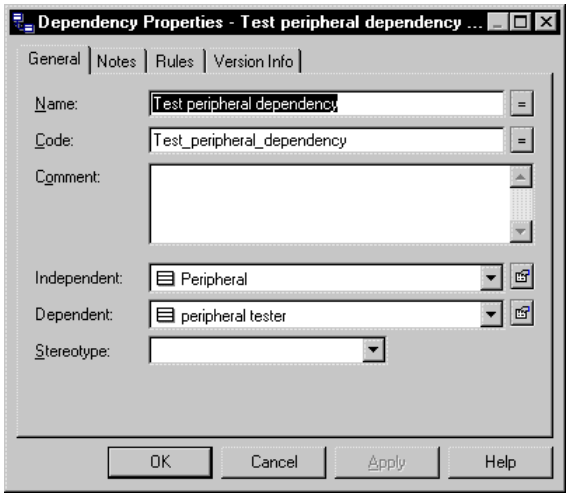
Modifying dependency properties from its property sheet

The dependency property sheet displays the definition of the dependency, which you can modify.

❖ **To modify dependency properties from its property sheet:**

1 Double-click the dependency in the model.

The dependency property sheet opens to the General page.



Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

2 Type or select dependency properties.
or
Click on a page tab.
Type or select dependency properties as required.

3 Click OK.

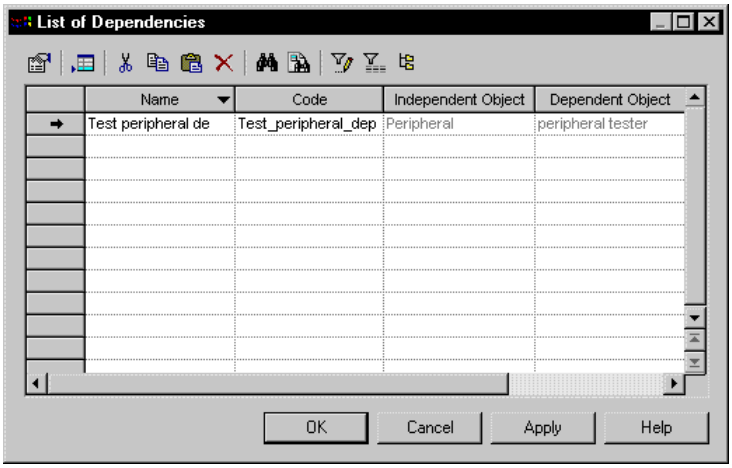
Modifying dependency properties from the list of dependencies

The list of dependencies includes all dependencies attached to the current model. You can modify the dependency properties from the list.

❖ **To modify dependency properties from the list of dependencies:**

1 Select Model➤Dependencies.

The list of dependencies appears.



2 Click the dependency that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the dependency directly in the list.

4 Click OK.

Displaying text in dependency symbols

A dependency has the following display preferences:

Preference	When selected
Show name	Displays the name of the dependency
Show stereotypes	Displays the stereotypes of the association
Show constraints	Displays the constraints (business rules) of the association

You modify the display preferences for a dependency in the Display Preferences dialog box.

❖ **To modify the display preferences:**

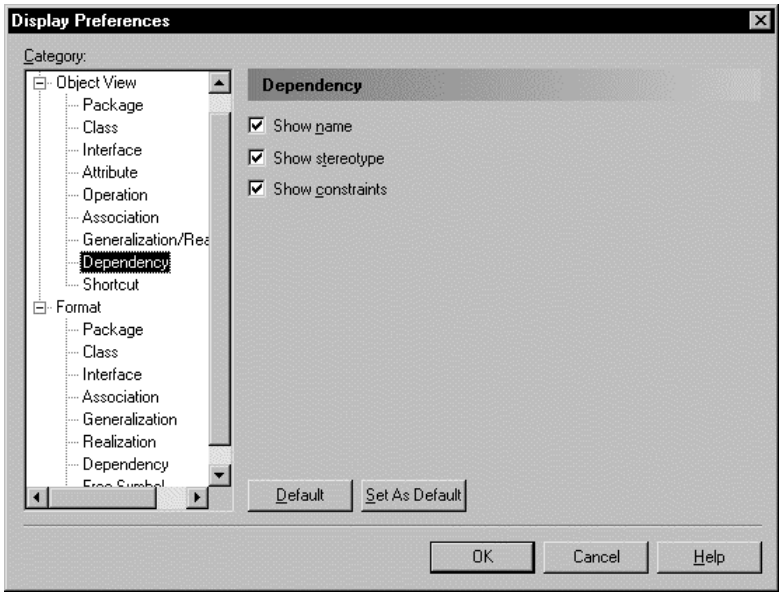
1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Dependency.

The Dependency display preferences appears.



4 Modify the dependency display preferences.

5 Click OK.

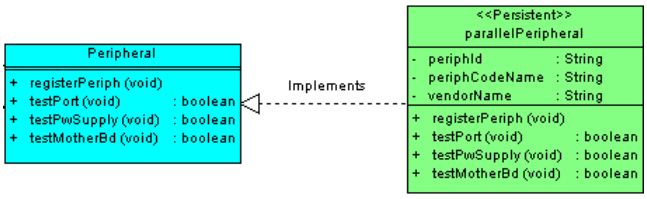
Defining realizations

A realization is a relationship between a class and an interface. It shows that the class realizes the operations offered by the interface. In this kind of relationship, the interface is called the specification element and the class is called the implementation element. The class implements the specification of the interface.

You can also create a realization between a shortcut of an interface and a class, or between a shortcut of a class and a interface. If the link is oriented, only the parent object can be the shortcut.

Although you can create more than one realization link between a class and an interface, you should create only one, because the interface can only realize one action for the class. When you generate from an OOM, if more than one realization exists between a class and an interface, a warning message is generated.

The arrowhead at one end of the realization always points towards the interface.



Realization properties

A realization has the following properties:

Property	Description	Maximum length
Name	Name of the realization	254
Code	Reference name for the realization	254
Comment	Descriptive comment for the realization	—
Interface	Name of the interface that carries out the realization	254
Class	Name of the class that for which the realization is carried out	254
Stereotype	Pre-defined or user defined instance of the realization	—

Creating a realization

You can create a realization only from a class to an interface.

❖ **To create a realization:**

1 Click the Realization tool in the palette toolbar.

2 Drag the realization from the class to the interface.

The link appears between the two objects.

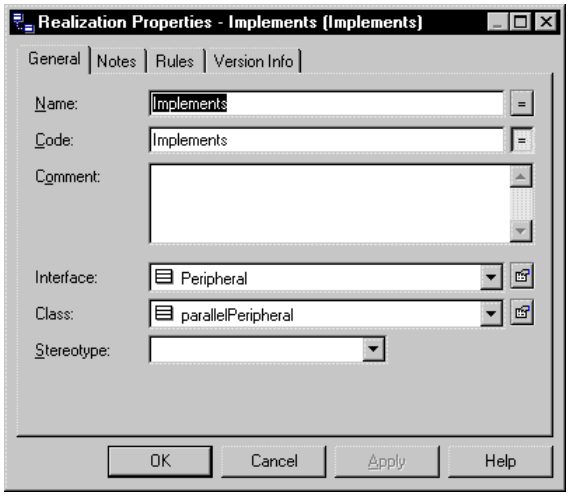
> **Dragging a realization to a different class**
> You can change the class or interface at either end of a realization by clicking the realization to select it, pressing down CTRL, and dragging one of the attach points to a different class or interface.

3 Click the Pointer tool in the palette toolbar.
or
Click the right mouse button.

You release the Realization tool.

4 Double-click the new realization in the model.

The realization property sheet opens to the General page.



Opening property sheets at last accessed page
Property sheets open to the General page by default. However, you can choose to open property sheets at the last page accessed by selecting Tools➤Options➤Dialog, and selecting the option Keep Last Tab in the Property Sheets groupbox.

5 Type a realization name and a realization code.

6 Select a stereotype from the dropdown listbox.

7 Click OK.

Modifying realization properties

There are two approaches to modifying realization properties:

- Modify the property sheet of a realization
- Modify an entry in the list of realizations

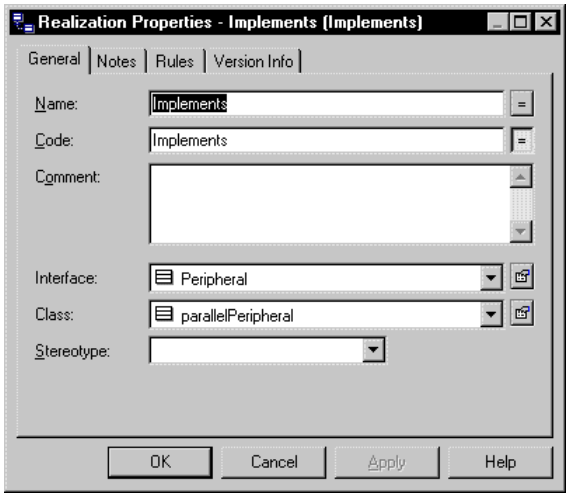
Modifying a realization from its property sheet

The realization property sheet displays the definition of the realization, which you can modify.

❖ **To modify realization properties from its property sheet:**

1 Double-click the realization in the model.

The realization property sheet appears.



2 Type or select realization properties.
or
Click on a page tab.
Type or select realization properties as required.

3 Click OK.

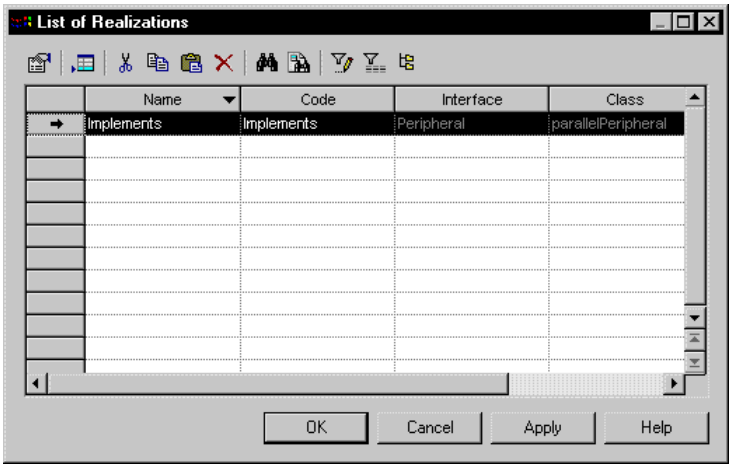
Modifying a realization from the list of realizations

The list of realizations includes all realizations attached to the current model. You can modify the realization properties from the list.

❖ **To modify realization properties from the list of realizations:**

1 Select Model➤Realizations.

The list of realizations appears.



2 Click the realization that you want to modify.

An arrow appears at the beginning of the line.

3 Modify any of the properties of the realization directly in the list.

4 Click OK.

Displaying text in realization symbols

A realization has the following display preferences:

Preference	When selected
Show name	Displays the name of the realization
Show stereotypes	Displays the stereotypes of the realization
Show constraints	Displays the constraints (business rules) of the realization

You modify the display preferences for a realization in the Display Preferences dialog box.

❖ **To modify the display preferences:**

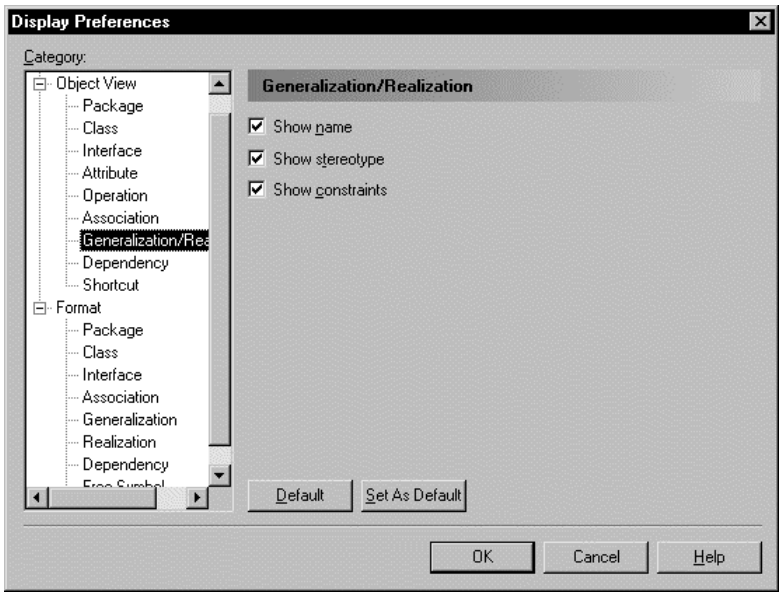
1 Select Tools➤Display Preferences.
or
Right-click the diagram background and select Display Preferences from the contextual menu.

The Display Preferences dialog box appears.

2 Expand the Object View node in the Category list.

3 Select Realization.

The Realization display preferences appears.



4 Modify the realization display preferences.

5 Click OK.

Defining domains

Domains help you identify the types of information in your project. They define the set of values for which an attribute is valid. Applying domains to attributes makes it easier to standardize data characteristics for attributes in different classes.

In an OOM, you can associate the following information with a domain:

- Data type, length, and precision
- Check parameters
- Business rules

Domain properties

Each domain definition includes the following properties:

Property	Description	Maximum length
Name	Name for the domain	254
Code	Reference name for the domain	254
Comment	Descriptive label for the domain	—
Data type	Form of the data corresponding to the domain, such as numeric, alphanumeric, Boolean, or others	—
Length	Maximum number of characters	—
Precision	Number of places after the decimal point, for data values that can take a decimal point	—

A domain definition can also include the following properties, which have associated values or information used by attributes attached to the class:

Property	Description
Standard checks	Check parameters defined for the domain
Additional checks	Domain constraints or validation rules not defined by standard check parameters
Rules	Business rules attached to the domain

Creating a domain

You create a domain from the list of domains.

> **Accessing the List of Domains**
> You can access the List of Domains from the current model, or by right clicking the appropriate model node in the Browser, and selecting New➤Domain from the contextual menu.

❖ **To create a domain:**

1 Select Model➤Domains.

The list of available domains appears.

2 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

3 Type a domain name and a domain code.

4 Click Apply.

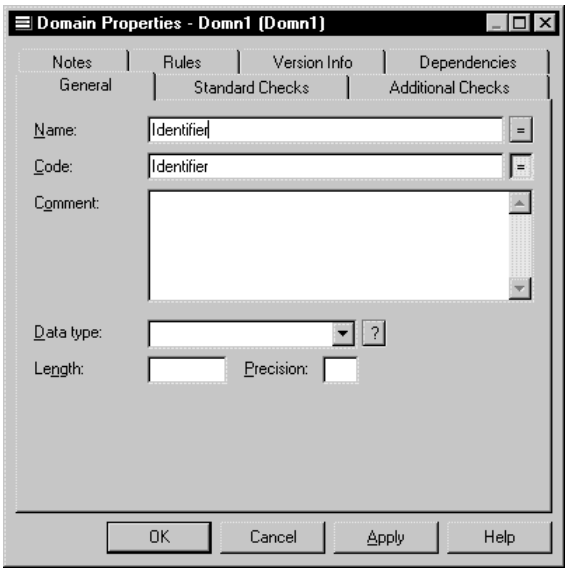
The creation of the new domain is committed.

5 Click the new domain line.

An arrow appears at the beginning of the line.

6 Click the Properties tool.
or
Double-click the arrow at the beginning of the line.

The property sheet for the new domain appears.



7 Select a data type.
Specify length and precision as required.

☞ For information on data types and selecting a data type for a domain see the following sections Indicating data type, length, and precision and Selecting a data type for a domain from the list.

8 Click on a page tab.
Type or select domain properties as required.

9 Click OK

You return to the List of Domains.

10 Click OK.
or
Click another domain line.

Indicating data type, length, and precision

The data types that you can select in a OOM depend on your current object language.

Length and precision

The properties length and precision do not apply to all data types. Depending on data type, length may indicate a maximum or a fixed number of characters.

In the list of available data types, a variable indicates where you have to type a length or precision, as follows:

Variable	Replace with
%n	Length
%s	Length with precision
%p	Decimal precision

Undefined data type

All object languages allow you to select the <undefined> data type. The <undefined> data type indicates which domains remain without data types. If an <undefined> data type is present when you generate your database, it is replaced by the default data type for your database.

Selecting a data type for a domain

You can select a data type for a domain in two ways:

- Directly from the List of Domains
- From the property sheet for the domain

The list of standard data types

When you select a data type for a domain from its property sheet, you can choose a data type from a list of standard data types available in PowerDesigner. This list presents the available data types in a more structured and complete format than the abbreviated format used in the data type dropdown listbox in the list of domains.

Selecting a data type for a domain from the list

❖ **To select a data type for a domain from the list:**

1 Select Model➤Domains.

The list of domains appears.

2 Click the domain that you want to define.

An arrow appears at the beginning of the line.

3 Click the Data Type attribute.

A dropdown listbox appears.

4 Select a data type from the dropdown listbox.

> **Undefined data type**
> If you do not want to select a data type immediately, you can choose the <Undefined> data type. When you generate Java or PowerBuilder objects, this data type is replaced by the default data type for your target object language.

5 Click OK.

Selecting a data type for a domain from its property sheet

❖ **To select a data type for a domain from its property sheet:**

1 Select Model➤Domains.

The List of Domains appears.

2 Click the domain to define.

An arrow appears at the beginning of the line.

3 Click the Properties tool.
or
Double-click the arrow at the beginning of the line.

The domain property sheet appears.

4 Select a data type from the Data Type dropdown list box.

> **Selecting a data type from a list of standard data types**
> You can select a data type from a list of standard data types by clicking the Question Mark button at the end of the Data Type dropdown listbox, and selecting the radio button for a data type from the list that appears.

5 Type the maximum number of characters for the data item in the Length box.

6 If the data type can include values that take a decimal point, type the number of places after the decimal point in the Precision box.

7 Click OK.

The change of data type appears in the list of domains.

Undefined data type
If you do not want to select a data type immediately, you can choose the <Undefined> data type. When you generate the database, this data type is replaced by the default data type for your target object language.

Selecting a data type from a list of standard data types

You can select a data type from a list of standard data types. This is the same list that is available in the Conceptual Data Model. PowerDesigner automatically maps the standard data type to an OOM data type.

The length and precision are properties that do not apply to all data types. Furthermore, depending on data type, length may indicate a maximum or a fixed number of characters.

The classes below indicates the data types for which you can specify:

- Fixed length
- Maximum length
- Decimal precision

Numeric data types

Conceptual data type	What it stores	Length?	Precision?
Integer	32-bit integer	—	—
Short Integer	16-bit integer	—	—
Long Integer	32-bit integer	—	—
Byte	256 values	—	—
Number	Numbers with a fixed decimal point	Fixed	✓
Decimal	Numbers with a fixed decimal point	Fixed	✓
Float	32-bit floating decimal numbers	Fixed	—
Short Float	Less than 32-bit floating decimal number	—	—
Long Float	64-bit floating decimal numbers	—	—
Money	Numbers with a fixed decimal point	Fixed	✓
Serial	Automatically incremented numbers	Fixed	—
Boolean	Two opposing values (true/false; yes/no; 1/0)	—	—

Character data types

Conceptual data type	What it stores	Length?
Characters	Character strings	Fixed
Variable Characters	Character strings	Maximum
Long Characters	Character strings	Maximum
Long Var Characters	Character strings	Maximum
Text	Character strings	Maximum
Multibyte	Multibyte character strings	Fixed
Variable Multibyte	Multibyte character strings	Maximum

Time data types

Conceptual data type	What it stores
Date	Day, month, year
Time	Hour, minute, and second
Date & Time	Date and time
Timestamp	System date and time

Other data types

Conceptual data type	What it stores	Length?
Binary	Binary strings	Maximum
Long Binary	Binary strings	Maximum
Image	Images	Maximum
Bitmap	Images in bitmap format (BMP)	Maximum
OLE	OLE links	Maximum
Other	User-defined data type	—
Undefined	Not yet defined data type	—

❖ **To select a data type from a list of standard data types:**

1 Select Model➤Domains.

The List of Domains appears.

2 Click the domain to define.

An arrow appears at the beginning of the line.

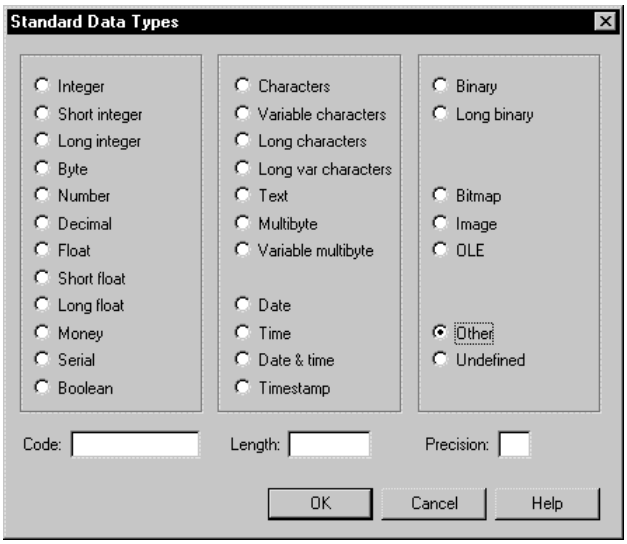
3 Click the Properties tool.
or
Double-click the arrow at the beginning of the line.

The domain property sheet appears.

4 Click the Question Mark button next to the Data Type dropdown listbox.

Selecting from the Data Type dropdown listbox
You can also select a data type directly from the Data Type dropdown listbox.

A list of standard data types appears.



5 Click the radio button corresponding to the data type you want to apply.

The code for the data type appears in the Code box.

Undefined data type
If you do not want to select a data type immediately, you can choose the Undefined data type.

6 Type the maximum number of characters for the data type in the Length box.

7 If the data type can include values that take a decimal point, type the number of places after the decimal point in the Precision box.

8 Click OK.

The change of data type appears in the Data Type box.

Modifying domain properties

You can modify domain properties from its property sheet.

When you modify a domain, you can choose to automatically update the following properties for attributes using the domain:

- Data type
- Check parameters
- Business rules

❖ **To modify domain properties:**

1 Select Model➤Domains.

The List of Domains appears.

2 Click a domain from the list.

An arrow appears at the start of the line.

3 Click the Properties tool.
or
Double-click the arrow at the start of the line.

> **Accessing a property sheet from the Browser**
> You can also access a domain property sheet by double-clicking the appropriate domain node in the Browser.

The Domain property sheet appears.

4 Type changes to domain properties.
or
Click on a page tab.
Type or select domain properties as required.

5 Click OK.

If the domain is used by one or more attributes, an update confirmation box appears asking if you want to modify domain properties for the attributes using the domain.

If the domain is not used by any attributes, then you do not receive the update confirmation box.

6 Select the properties that you want to be updated for all attributes using the domain.

7 Click Yes.

Defining check parameters

Check parameters are set of conditions which data must satisfy to remain valid. They are used principally in for use in a CDM or a PDM.

There are two types of check parameters:

Parameter type	Description	Can be attached to
Standard parameters	Common data constraints which define a data range. For example minimum and maximum values for an attribute	Attributes Domains
Additional check parameters	SQL expression defining a data constraint using the %MINMAX%, %LISTVAL%, and %RULES% variables that are instantiated with standard parameter values	Attributes Domains
Validation rule	Business rule that is defined as a server expression, and is attached to one of the following listed objects	Classes Attributes Domains

Setting standard check parameters for objects

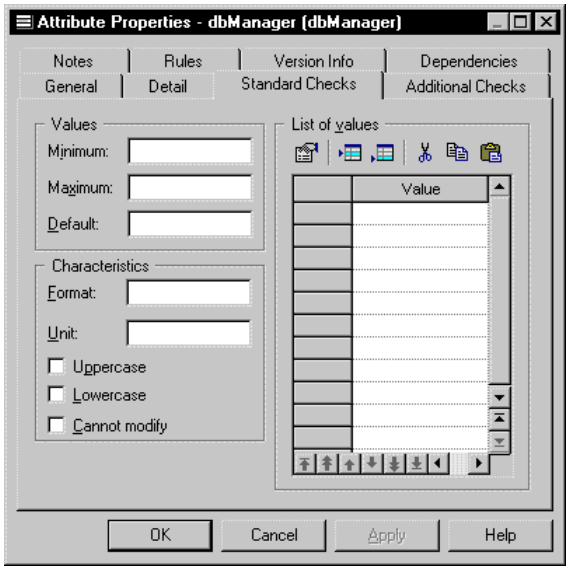
Standard parameters indicate common data constraints. The following table lists standard parameters:

Parameter	Description
Minimum	Lowest acceptable numeric value
Maximum	Highest acceptable numeric value
Default	Value assigned in absence of an expressly entered value
Format	Data format (for example, 9999.99)
Unit	Standard measure
Uppercase	Forces all alphabetical characters to uppercase
Lowercase	Forces all alphabetical characters to lowercase
Cannot Modify	Protects from changes, results in a nonmodifiable attribute in the class
List of Values	Authorized values

❖ **To set standard parameters:**

1 Click the Standard Checks tab in the property sheet of a domain or an attribute.

The Standard Checks page appears.



2 Type your choice of Standard Parameters.

3 Click OK.

Defining additional check parameters for objects

You can write an SQL statement using the following standard variables defined as standard check parameters and validation rules:

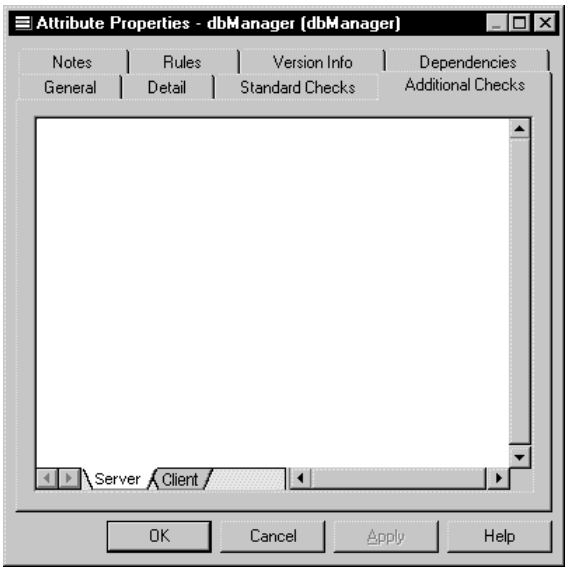
Variable	Description
%MINMAX%,	Minimum and maximum values defined in Values groupbox on Standard Checks page
%LISTVAL%	Customized values defined in List Values groupbox on Standard Checks page
%RULES%	Validation rule expression defined on Expression page of the Rules property sheet

You define additional check parameters for data constraints where standard check parameters are not sufficient.

❖ **To define additional check parameters:**

1 Click the Additional Checks tab in the property sheet of an attribute or domain.

The Additional Checks page appears.



2 Type SQL expression using the variables %MINMAX%, %LISTVAL%, and %RULES%.

3 Click OK.

Using a validation rule in check parameters

A validation rule is a rule that validates data based on a corresponding business rule. A validation rule can be generated as a check parameter when the following conditions apply:

- Validation rule is attached to a class, attribute, or domain
- Validation rule is defined as a server expression

At generation, validation rule variables are instantiated with the following values:

Variable	Value
%ATTRIBUTE%	Code of the attribute to which the business rule applies
%DOMAIN%	Code of the domain to which the business rule applies
%CLASS%	Code of the class to which the business rule applies
%MINMAX%	Minimum and maximum values for the attribute or domain
%LISTVAL%	List values for the entity attribute or domain
%RULES%	Server validation rules for the entity attribute or domain

For more information on defining business rules, see the chapter Using Business Rules.

❖ **To use a validation rule in check parameters:**

1 Click the Rules tab in the property sheet of a class, attribute, or domain.

The Rules page appears.

2 Click the Add Rules tool.

A list appears displaying the available business rules in the model.

3 Select a business rule in the list.

4 Click OK in each of the dialog boxes.

Validation rule expressions
You must click the Rules button to modify the expression attached to a validation rule. You can also modify validation rule expressions from the list of business rules, by clicking the Define button.

CHAPTER 3

Managing Object-Oriented Models

About this chapter

This chapter describes how to compare and merge Object-Oriented Models as well as how to check the validity of a Object-Oriented Model (OOM).

Contents

Checking an OOM

The procedure that generates .java Java source files or PowerBuilder objects starts by checking the validity of the OOM. If an error is found, the files are not generated.

Object parameters verified by Check model

The Check Model verifies the validity of the following objects in an OOM:

Object	Parameter
Classes	Class name and code uniqueness and length Class must have either attribute or operation Class is not declared as private Class constructor has no return type specified Class constructor cannot have modifiers Class constructor cannot be not declared as static, abstract or final
Interfaces	Interface name and code uniqueness and length Interface must have either attribute or operation Interface cannot have constructors
Attribute	Attribute name and code uniqueness and length Data type has a not-null and not-void data type Attribute must have value assigned Attribute must be initialized
Operation	Operation name and code uniqueness and length Operation must have a not-null return type Operation parameter must have a not-null and not-void data type Abstract operation cannot have a body Abstract operation declaration must appear only in an abstract class
Association	Role name and code uniqueness and length
Realization	Redundant realizations. Only one realization is needed to realize an interface
Generalization	Redundant generalization. Only one generalization is needed to generalize a classifier Generalization cannot have multiple inheritance (Java only) Generalization cannot have circular inheritance
BeanInfo class	BeanInfo class must have a corresponding JavaBean class
Views	View name and code uniqueness and length Incomplete query

OOM check options

When you check an OOM, if a parameter is found to be invalid, it can be displayed with one of two types of messages:

Message	Description
Error	Major problem that impedes Java or PowerBuilder generation
Warning	Minor problem or recommendation

These messages represent two different levels of problem severity. You can modify the level of problem severity for each object parameter that is verified by the Check model. This severity level can depend on the degree of normalization that you want to achieve in a your model.

You can also have certain problems automatically corrected.

Indicating error severity

You can use the following tools from the Check Model Parameters dialog box to indicate either an error or warning level of problem severity, and also if you want PowerDesigner to automatically correct an error:

Tool	Indicates	Description
	Error	Major problem that impedes generation
	Warning	Minor problem or recommendation
	Automatic correction	Indicates that PowerDesigner will correct the problem automatically

You must also choose one of the following correction options:

Option	Description
Manual correction	Displays error and warning messages
Automatic correction	Displays error and warning messages Corrects certain errors automatically

Object selection in the Check Model

You select objects to check from the Selection page.

You can list all objects in the current model, or package, by selecting the Include Sub-packages tool.

You have the following selection options:

Parent object	Include Sub-packages	Displays
Model	Selected	All objects in model including all objects contained in packages and sub-packages
Model	Not selected	All objects in model except objects contained in packages and sub-packages
Package	Selected	All objects contained in package including all objects contained in sub-packages
Package	Not selected	All objects in package except objects contained in sub-packages

Objects selected in the diagram
Graphically selected objects in your diagram can be automatically selected for verification by the Check Model by clicking the Use Graphical Selection tool in the Selection page tool bar.

Checking a OOM

You can check the validity of an OOM at any time.

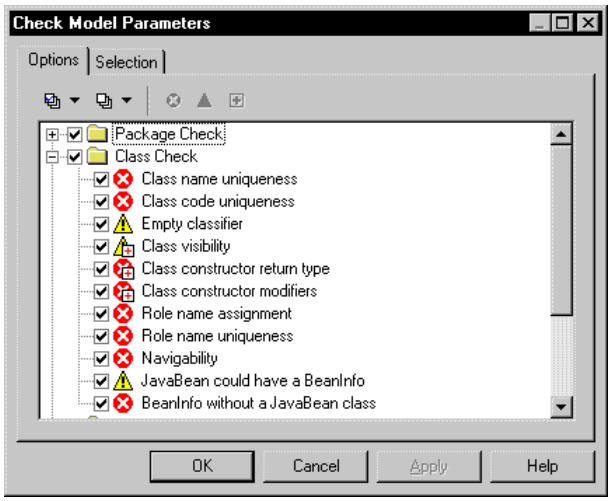
❖ **To check a OOM:**

1 Select Tools➤Check Model.
or
Right-click the diagram background and select Check Model from the context menu.

The Check Model Parameters dialog box opens to the Options page.

2 Expand an object parameter node.

The object parameters which are verified by the Check Model are displayed with the symbols indicating a degree of problem severity.



3 If you want to change a degree of problem severity, select the object parameter and then select either the Error or Warning tool.

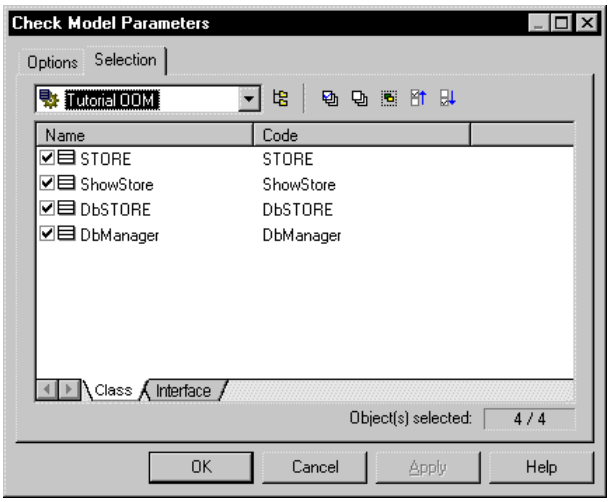
The symbol changes to the appropriate severity level.

4 If you want PowerDesigner to automatically correct a problem, select the object parameter and then select the Automatic Correction tool.

The Automatic Correction symbol appears superimposed on the Error or Warning symbol for that object parameter.

5 Click the Selection tab.

The Selection page appears.



6 Select a model from the dropdown list at the top of the dialog box.

7 Click an object tab.

The corresponding object page displays all the objects in the current OOM.

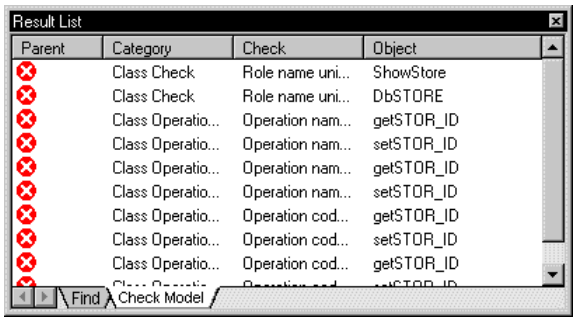
8 Select checkboxes for objects that you want to be checked.

9 Clear checkboxes for objects that you do not want to be checked.

> **Selecting all or clearing all checkboxes**
> You can select all object checkboxes by clicking the Select All tool. You can clear all object checkboxes by clicking the Deselect All tool.

10 Click OK.

The Check Model Result List displays errors and warnings based on the check options you have defined.



Dockable result window
When you right click an object parameter, a menu appears listing correction options. Among these, you can also select options to clear, dock or hide the result window.

Making corrections based on OOM check results

You can use the Check Model to locate and correct problems in the OOM.

You can choose one of the following correction options from the Check toolbar:

Symbol	Option	Description
	Manual correction	Displays property sheet of problem object
	Check detail	Displays description of the error and suggestion for correction
	Recheck	Checks selected object parameter, normally after a correction has been done
	Automatic correction	Automatically corrects: ♦ Non-unique names ♦ Code too long for generation (Java or PowerBuilder) ♦ Divergence in domain values, check parameters, and validation rules

Navigating in the error list

The Check tool bar also contains navigation tools that you can use to move to the first, previous, next, or last errors that are listed. You can also navigate in the list of errors by right-clicking an object parameter and selecting Go To First error, Previous error, Next error, or Last error from the context menu.

Right click menu
When you right click an object parameter a menu appears listing the correction options Manual Correction, Check Detail, Recheck, and Automatic Correction. You can also select options to clear, dock and hide the result window.

Making automatic corrections to the OOM

❖ **To make automatic corrections to an OOM:**

1 From the Check Result dialog box, select an object parameter.

2 Right-click the object parameter and select Auto-Correction from the contextual menu.

3 Right-click the object parameter and select Re-check from the contextual menu.

Verify that the problem has been corrected.

Making manual corrections to an OOM

Some errors cannot be corrected automatically and have to be corrected manually.

❖ **To make manual corrections to an OOM:**

1 From the Check Result dialog box, select an object parameter.

2 Right-click the object parameter and select Check from the contextual menu.

The object property sheet appears.

3 Select the appropriate tab and make the necessary correction.

4 Close the property sheet.

5 Re-select the object parameter.

6 Right-click the object parameter and select Re-check from the contextual menu.

Verify that the problem has been corrected.

Merging two OOM

You can merge two OOM. The merge makes it possible to form a single model that combines design efforts performed independently by several team members.

When the merge process finds two objects that have the same code, you can indicate whether or not the definition of the object in the source model should replace the definition in the target model.

For more information on merging models, see the PowerDesigner General Features Guide.

Opening a Rose model in an OOM

You can import a .mdl models built with Rational Rose in PowerDesigner. A new OOM is created for the Rose model, and the objects of the Rose model are translated into OOM objects.

This functionality provides you with greater scope and flexibility. You can create an OOM from a Rose model, from which you can generate Java files or objects for PowerBuilder to create applications. You can also use the OOM created from a Rose model to add to an existing OOM, or to generate a CDM or PDM for database analysis purposes.

❖ **To open a Rose model in PowerDesigner:**

1 Select File➤Open

An open file dialog box appears.

2 Select or browse to the directory that contains the Rose file.

3 Select Rose Model (*.mdl) file from the Files of type dropdown listbox.

The available Rose files are listed.

4 Select a file.

5 Click Open.

A message box lists all imported objects.

6 Click OK.

Objects imported

The following Rose objects are imported directly into the new OOM:

Package
Diagram
Class
Interface
Attribute
Operation
Generalization
Association
Dependency
Realization
Note
Note Link
Text

The import process translates some properties of imported objects into OOM properties as follows:

All objects

Property in an Rose model	Imported property in a OOM
Documentation	Comment
Export Control - Implementation	Visibility - Package

Class

Property in an Rose model	Imported property in a OOM
Cardinality - n	Cardinality - *
Cardinality - 1..n	Cardinality - 1..*

Generalization

Property in an Rose model	Imported property in a OOM
Virtual inheritance	Virtual

Association

Property in an Rose model	Imported property in a OOM
Cardinality	Multiplicity
Aggregate	Container
Aggregate by reference, by value, unspecified	Aggregation or Composition

Objects not imported

When you open a Rose model, the following properties are not imported into the new OOM:

Package

Global

Class

Rose Property	Rose Sub-property
Type	Parameterized Class Instanciated Class Parameterized Class Utility Instanciated Class Utility MetaClass
Formal arguments	—
Nested Class	—
Concurrency	—
Files	—

Interface

Rose Property	Rose Sub-property
Type	Class Class Utility Parameterized Class Instanciated Class Parameterized Class Utility Instanciated Class Utility MetaClass
Formal arguments	—
Cardinality	—
Persistence	—
Abstract	—
Nested Class	—
Concurrency	—

Attribute

Rose Property	Rose Sub-property
Containment	By Value By Reference Unspecified

Operation	Default Values of Arguments Protocol Qualification (language-specific) Exceptions Size (amount of storage) Time (to complete operation) Concurrency (sequential, guarded, synchronous) Preconditions Postconditions
Generalization	Friendship required (yes/No)
Association	Keys/qualifiers Constraints Stereotype Derived Static Friend
Dependency	Export control Friendship required Cardinality from Cardinality to

CHAPTER 4

Reverse Engineering

About this chapter

This chapter describes Java, PowerBuilder, and XML reverse engineering functions for an Object-Oriented Model (OOM). It also shows you how to create a new OOM by reverse engineering from a database.

Contents

What is reverse engineering?

Reverse engineering is the process of examining and recovering data or source code from a file that is then used to build or update an OOM. You reverse engineer objects to an OOM via a diagram. You can reverse engineer objects to a new model, or to an existing model. When you reverse engineer an object that already exists in a model, you can choose in an object comparison box either to replace the existing object, or to keep the existing object in the model.

> **Parsing**
> PowerDesigner uses parser software for reverse engineering XML, that was developed by the Apache Software Foundation (http://www.apache.org/).

You can reverse the following type of files into an OOM:

- Java
- PowerBuilder
- XML

Reverse engineering into a new OOM

You can reverse engineer an existing database into a new OOM. The data source can be either from a script file or an ODBC data source.

This functionality is accessible from the File➤Reverse Engineering menu.

Reverse engineering Java

You can reverse engineer files that contain Java classes into an OOM. For each existing class in a Java file, a corresponding class is created in the model, with the same name and containing the same information. When you reverse engineer a Java class that already exists in a model, you can choose in the Merge Model window either to replace the existing class, or to keep the existing class definition in the model.

Reverse engineered Java classes always keep their original names.

When you reverse engineer classes from Java files to a diagram, you can choose from one of the following four sources:

Source	Description	Extension
Java .java files	Each file contains one or several class definitions	.java
Java .class files	Files that contain one class definition that has the same name as the file	.class
Directory	Folder from which you can reverse all the Java files, including all those contained in it's sub-directories	—
Archived Java files	Compressed .jar or .zip files. Only the Java classes contained in these files are imported into your model. All other information is discarded	.zip and .jar

Inner Classes

An inner class is a class definition that is defined within another (outer) class definition. Inner classes are commonly used in Java. They help you to improve the overall visibility of your model by allowing you to group together classes that logically belong together.

When you reverse a Java class that contains one or more inner classes, one class is created for the outer class, and one class is created for each of the inner classes.

A dependency link is created between each inner class and the outer class to which it belongs. The name of each inner class is prefixed by the name of the outer class.

Java code comments

When you reverse engineer Java files, some comments may change form or position within the code.

Comment in original Java file	After reverse
Before the import declarations	Is lost from file
Beginning with /*	Begins with //
At the end of the file below all the code	Is lost from file
Within a class but not within an operation	Is attached to the attribute or operation that immediately follows it

Reverse engineering Java options

You define Java reverse engineering options from the Reverse Java dialog box.

You can define the following Java reverse engineering options:

Option	Result of selection
Ignore operation body	Reverses classes without including the body of the code
Ignore Comments	Reverses classes without including code comments
Create Symbols	Creates a symbol for each object in the diagram. Otherwise, reversed objects are visible in the browser
Mark Classifiers not to be generated	Reversed classifiers (classes and interfaces) cannot then be generated from the model. To be able to generate the classifier, you must select the Generate checkbox in its property sheet
Create Associations	Creates associations between classes and/or interfaces
Libraries	Opens a JDK model in the workspace. The Setup program installs these models with PowerDesigner. They contain the class libraries of each version of JDK and are useful to you in that you can load them quickly into PowerDesigner and thus save time reversing them

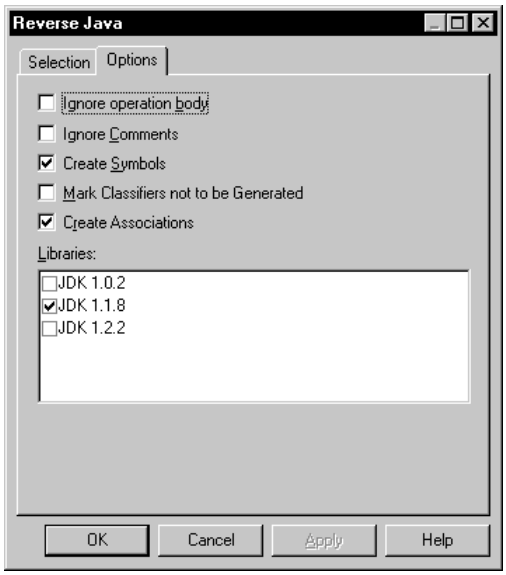
❖ **To define Java reverse engineering options:**

1 Select Language➤Reverse Engineer Java.

The Reverse Java dialog box opens.

2 Click the Options tab.

The Options page appears.



3 Select or clear options.

4 Click Apply.

5 Click Cancel.

Loading a JDK library model in the workspace

When you reverse engineer Java files, you can, at the same time, load one of the JDK models that contains the class libraries of a particular version of JDK. You can select to reverse a JDK library model from the Options page of the Reverse Java dialog box. The Setup program installs these models in the PowerDesigner LIBRARY folder.

You can open a JDK library model in the workspace from the PowerDesigner LIBRARY directory. You can then reference a class from the reversed JDK library model by creating shortcuts from another OOM.

❖ **To load a JDK library model:**

1 Select File➤Open.

An open file dialog box appears.

2 Select or browse to the PowerDesigner Library directory.

The available library files are listed. Each JDK file corresponds to a particular version of JDK.

3 Select the file JDK-1_1_8.OOM.

This file contains all the library class files of version 1_1_8 of JDK.

4 Click Open.

The OOM opens in the workspace.

Reverse engineering Java source files without code body

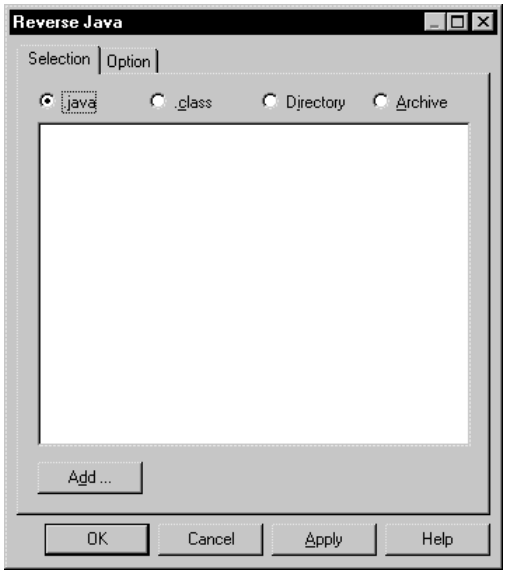
You can reverse engineer .java class source files without the body of the code. When you reverse classes in this way, the code contained within the operations of the class does not appear in the reversed class.

You use this option when you want to reverse objects for visualization or comparison purposes, or to limit the size of your model when you have a very large number of classes to reverse.

❖ **To reverse java without code body:**

1 Select Language➤Reverse Engineer Java.

The Reverse Java dialog box appears.



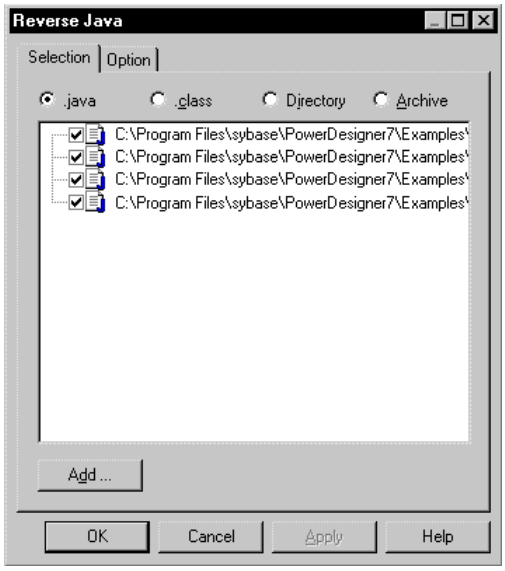
2 Select the .java radio button.

3 Click the Add button.

A standard Open dialog box appears.

4 Select the files that you want to reverse and click Open.

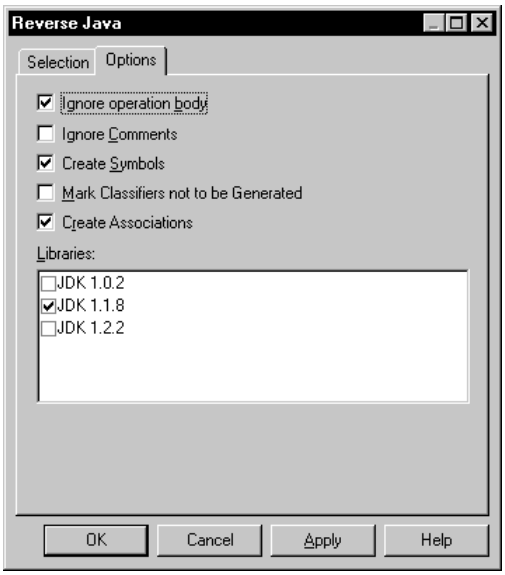
You return to the Reverse Java dialog box. It displays the files you selected.



5 Click the Options tab.

The Options page appears.

6 Select the Ignore operation body checkbox.



7 Click OK.

The classes reversed without the body of the code. The classes are automatically added to your model and are visible in the diagram.

Reverse engineering Java source files

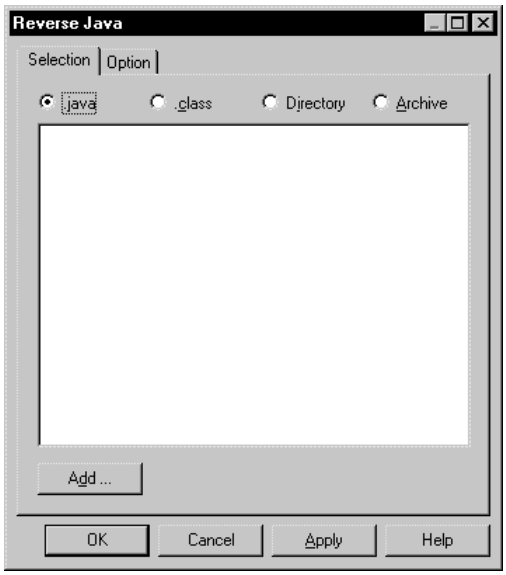
Each .java source file contains information on one or several class definitions. When you reverse engineer a .java file, PowerDesigner creates a class in the model corresponding to each class definition in the .java file. The newly created classes have the same name as in the .java file.

❖ **To reverse engineer .java files:**

1 Select Language➤Reverse Engineer Java.

The Reverse Java dialog box appears.

2 Select the .java radio button.



Reversing without the body of the code
You can choose reverse .java source files without the body of the code of the class by selecting the Ignore operation body checkbox in the Options page.

3 Click the Add button.

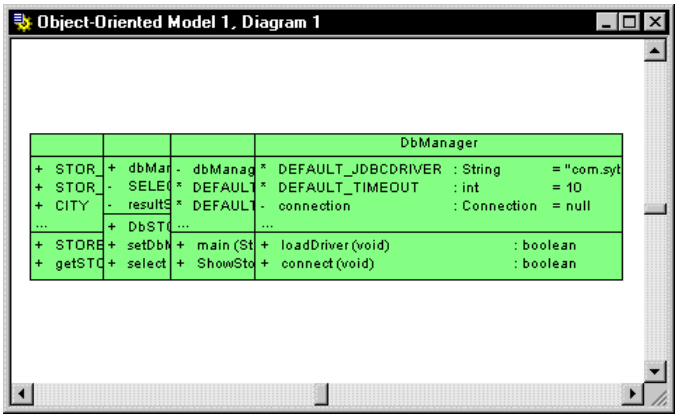
A standard Open dialog box appears.

4 Select the files that you want to reverse and click Open.

> **Multi-selection**
> You can select several files simultaneously by using the CTRL or SHIFT keys.

5 Click OK.

A Progress box appears and the classes are added to your model. The classes are visible in the diagram and in the Browser.



The reversed classes are listed in the Reverse page of the Output window, situated in the bottom part of the PowerDesigner main window.

Reverse engineering compiled Java files

A compiled .class file contains the definition of one unique class. A compiled .class file results from compiling a .java file, using an independent Java compiler. After compilation, each class definition in the .java file becomes an individual compiled .class file.

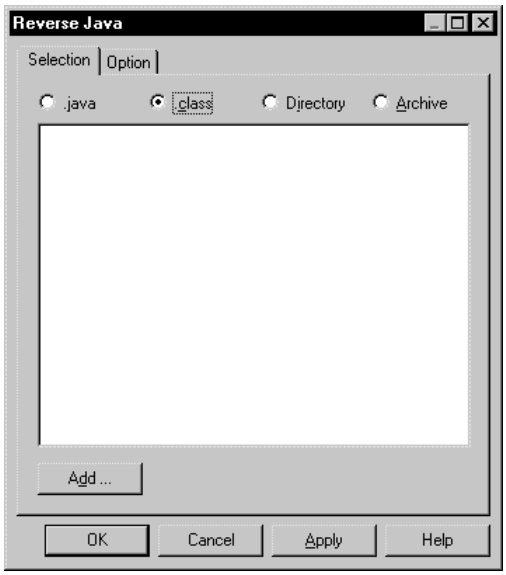
When you reverse engineer a .class file, PowerDesigner creates a class in the model that corresponds to the class definition in the .class file. The newly created class has the same name as the definition in the .class file.

❖ **To reverse engineer compiled Java files to a diagram:**

1 Select Language➤Reverse Engineer Java.

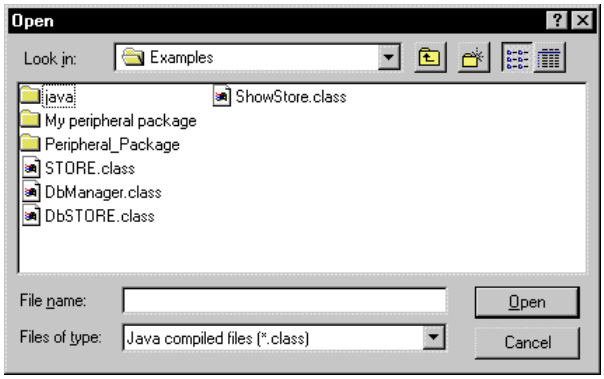
The Reverse Java dialog box appears.

2 Select the .class radio button.



3 Click the Add button.

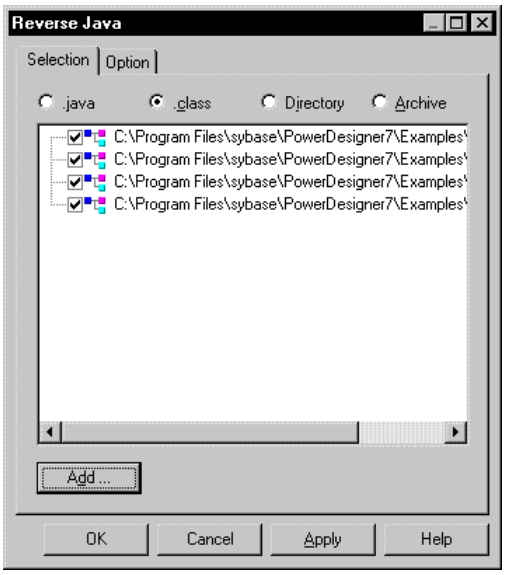
A standard Open dialog box appears.



4 Select the files that you want to reverse and click Open.

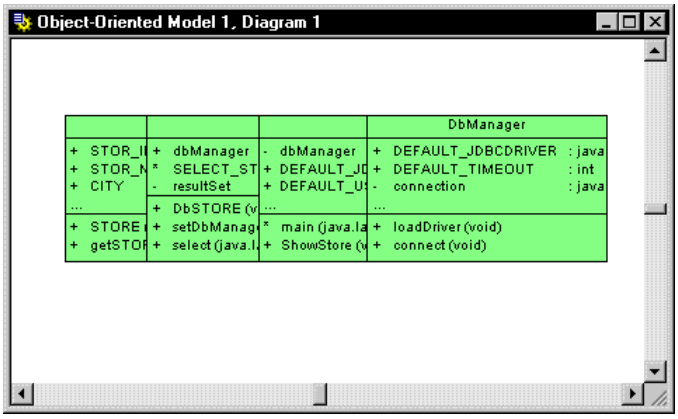
Multi-selection
You can select several files simultaneously by using the CTRL or SHIFT keys.

You return to the Reverse Java dialog box. It displays the files you selected.



5 Click OK.

A Progress box appears and the classes are added to your model. The classes are visible in the diagram and in the Browser.



The reversed classes are listed in the Reverse page of the Output window, situated in the bottom part of the PowerDesigner main window.

Reverse engineering Java files from a source directory

Reverse engineering .java source files from a source directory requires that you follow the same procedure as when you reverse engineer independent .java files, the only difference being that you select a directory in which several .java files are located and not individual files. This gives you the advantage of reversing groups of files that belong to the same model or package.

Often Java library files are interdependent as they belong to the same model and are therefore located in the same directory. In this case, if you do not reverse engineer all the library files located in the directory, your model may be incomplete.

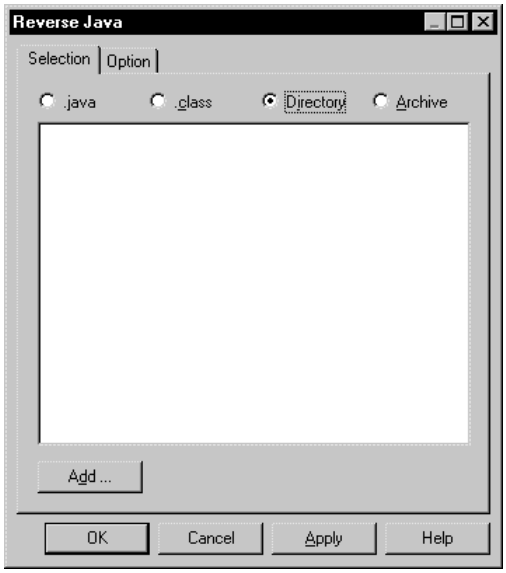
When you reverse engineer a directory, all the sub-directories and the Java files contained in them are reversed. In this case, each sub-directory becomes a package within the model.

❖ **To reverse engineer Java files from a source directory:**

1 Select Language➤Reverse Engineer Java.

The Reverse Java dialog box appears.

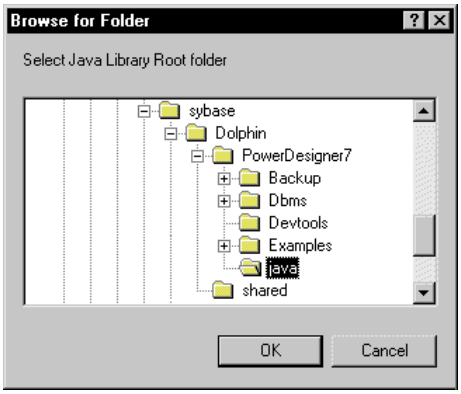
2 Select the Directory radio button.



Reversing without the body of the code
You can choose reverse .java source files without the body of the code of the class by deselecting the Ignore operation body checkbox.

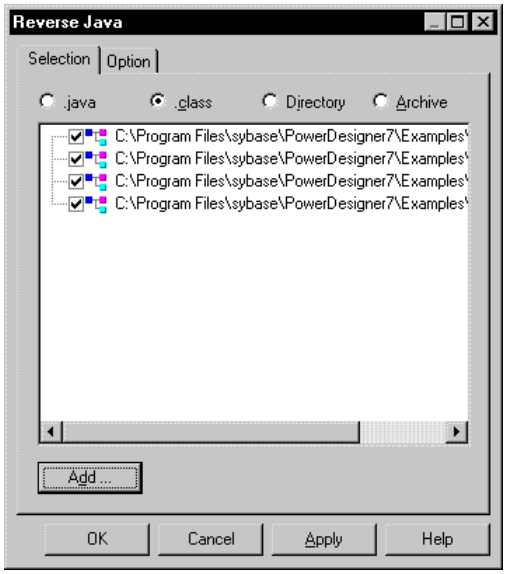
3 Click the Add button.

The Browse for Folder dialog box appears.



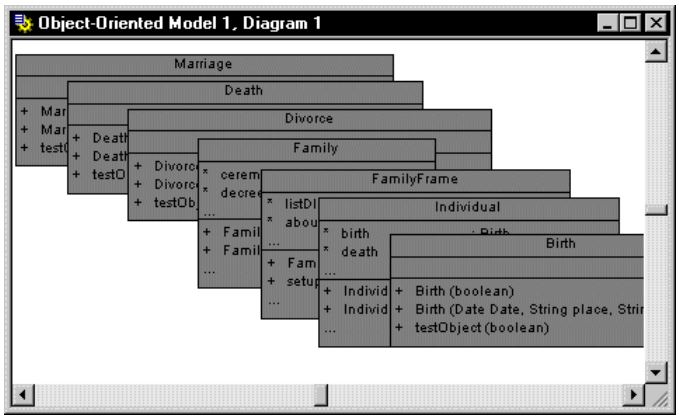
4 Select the directory that contains the Java files you want to reverse and click OK.

You return to the Reverse Java dialog box. It displays the files you selected.



5 Click OK.

A Progress box appears and the classes are added to your model. The classes are visible in the diagram and in the Browser.



The reversed classes are listed in the Reverse page of the Output window, situated in the bottom part of the PowerDesigner main window.

Reverse engineering archived .jar or .zip files

Each .jar or .zip file contains definitions of one or several classes. These files can often contain large numbers of class definitions.

A .jar file is a compressed file type that contains one or several Java class definitions. When you reverse engineer a .jar file, PowerDesigner creates a class for each class definition in the .jar file.

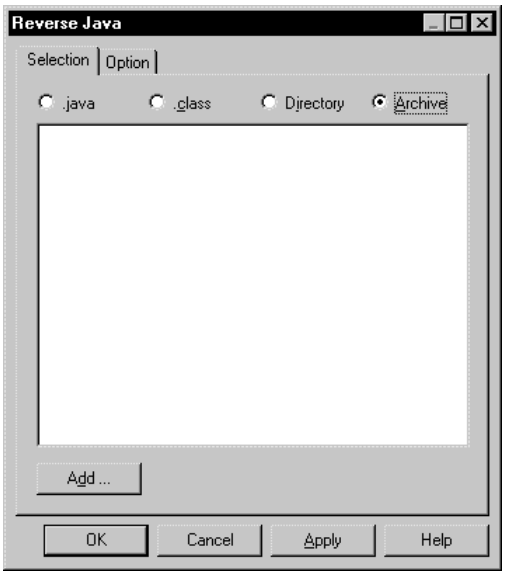
A .zip file can contain one or several Java class files, as well as other files. When you reverse engineer a .zip file, only the Java class files are reversed. Information stored in other files is not reversed and is totally discarded.

❖ **To reverse engineer .jar or .zip files to a diagram:**

1 Select Language➤Reverse Engineer Java.

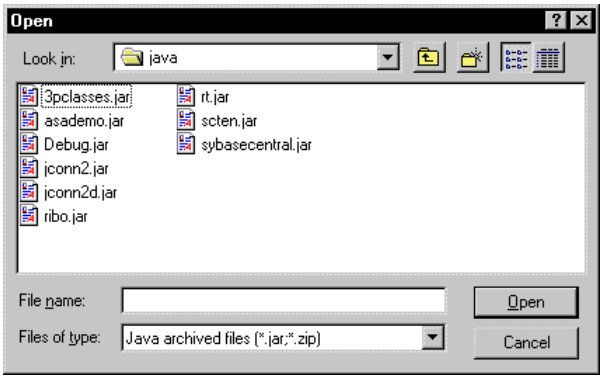
The Reverse Java dialog box appears.

2 Select the Archive radio button.



3 Click the Add button.

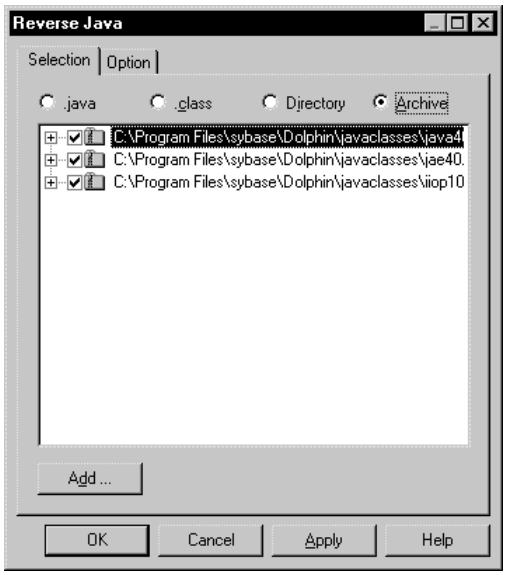
A standard Open dialog box appears.



4 Select the files that you want to reverse and click Open.

Multi-selection
You can select several files simultaneously by using the CTRL or SHIFT keys.

The Reverse Java dialog box displays the files you selected.



5 Click OK.

A Progress box appears and the classes are added to your model. The classes are visible in the diagram and in the Browser.

The reversed classes are listed in the Reverse page of the Output window, situated in the bottom part of the PowerDesigner main window.

Reverse engineering PowerBuilder

You can reverse engineer PowerBuilder NVO (non-visual objects) into an OOM from either of the following sources:

- PowerBuilder applications
- SRU files

For each reversed PowerBuilder object, a class is created in the model, with the same name and containing the same information. When you reverse engineer an object that has the same name as a class that already exists in a model, you can choose in the Merge Model window either to replace the existing class, or to keep the existing class definition in the model.

You can reverse only the following PowerBuilder User Objects:

- Custom Class
- Standard Class
- Custom Visual
- External Visual
- Standard Visual

Reverse engineering PowerBuilder options

You define PowerBuilder reverse engineering options from the Reverse PowerBuilder dialog box.

You can define the following PowerBuilder reverse engineering options:

Option	Result of selection
Ignore operation body	Reverses PowerBuilder objects without including the body of the code
Ignore Comments	Reverses PowerBuilder objects without including code comments
Create Symbols	Creates a symbol for each object in the diagram. Otherwise, reversed objects are visible in the browser
Mark Classifiers not to be generated	Reversed classifiers (classes and interfaces) cannot then be generated from the model. To be able to generate the classifier, you must select the Generate checkbox in its property sheet
Create Associations	Creates associations between classes
Libraries	Loads the corresponding PowerBuilder model in the workspace. The Setup program installs these models with PowerDesigner. They contain the class libraries of each version of PowerBuilder and are useful to you in that you can load them quickly into PowerDesigner and thus save time reversing them

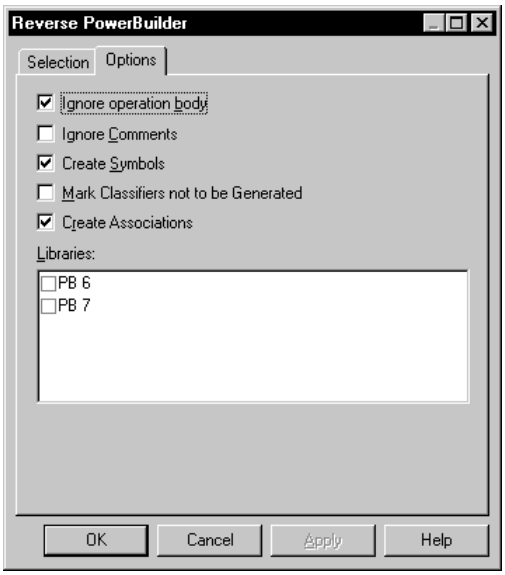
❖ **To define PowerBuilder reverse engineering options:**

1 Select Language➤Reverse Engineer PowerBuilder.

The Reverse PowerBuilder dialog box opens.

2 Click the Options tab.

The Options page appears.



3 Select PowerBuilder reverse options.

4 Click Apply.

5 Click Cancel.

Loading a PowerBuilder library model in the workspace

When you reverse engineer PowerBuilder files, you can, at the same time, load one of the PowerBuilder models that contains the class libraries of a particular version of PowerBuilder. You can select to reverse a PowerBuilder library model from the options page of the Reverse PowerBuilder dialog box. The Setup program installs these models in the PowerDesigner library folder.

You can open a PowerBuilder library model in the workspace from the PowerDesigner Library directory.

❖ **To load a PowerBuilder library model:**

1 Select File➤Open.

An open file dialog box appears.

2 Select or browse to the PowerDesigner Library directory.

The available library files are listed. Each PB file corresponds to a particular version of PowerBuilder.

3 Select the file PB7.OOM.

This file contains all the library class files of PowerBuilder version 7.

4 Click Open.

The OOM opens in the workspace.

Reverse engineering objects from a PowerBuilder application

When you reverse engineer objects from a PowerBuilder application, you can select only one PowerBuilder application from the PB application dropdown listbox. You can then add objects that belong to this application to the list of objects to reverse.

If PowerBuilder is not installed on your machine, you cannot generate objects for a PowerBuilder application, and you can reverse only SRU files.

Reverse engineered PowerBuilder objects always keep their original names.

❖ **To reverse PowerBuilder objects from a PowerBuilder application:**

1 Select Language➤Reverse Engineer PowerBuilder.

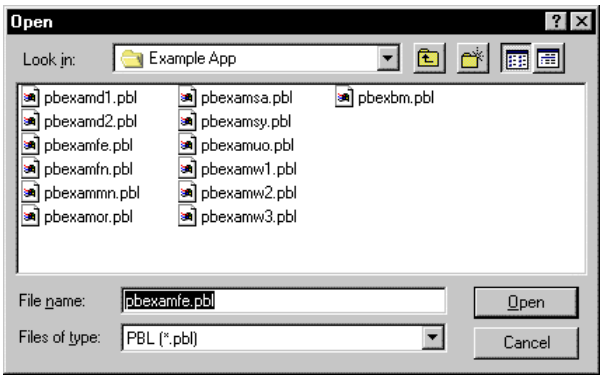
The Reverse PowerBuilder dialog box appears.

2 Select the PBL radio button.

3 Select a PowerBuilder application from the PB Application dropdown listbox.

4 Click the Add button.

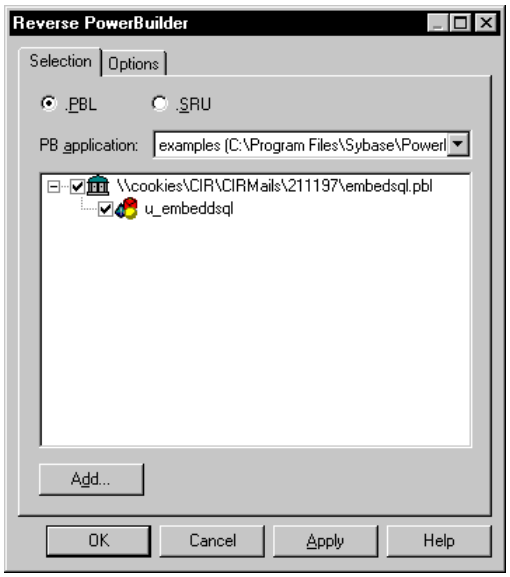
A standard Open dialog box appears.



5 Select the file that you want to reverse and click Open.

> **Multi-selection**
> You can select several files simultaneously by using the CTRL or SHIFT keys.

You return to the Reverse PowerBuilder dialog box. It displays the files you selected.



6 Click OK.

A Progress box appears and the classes are added to your model. The classes are visible in the diagram and in the Browser.

The reversed classes are listed in the Reverse page of the Output window, situated in the bottom part of the PowerDesigner main window.

Reverse engineering objects from SRU files

SRU files are text files containing the definition of PowerBuilder User Objects.

You do not have to have PowerBuilder installed on your machine to reverse engineer objects contained in SRU files.

Reverse engineered PowerBuilder objects always keep their original names.

❖ **To reverse engineer PowerBuilder objects from SRU files:**

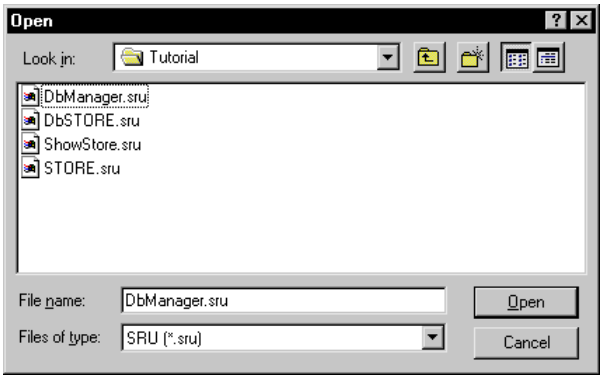
1 Select Language➤Reverse Engineer PowerBuilder.

The Reverse PowerBuilder dialog box appears.

2 Select the SRU radio button.

3 Click the Add button.

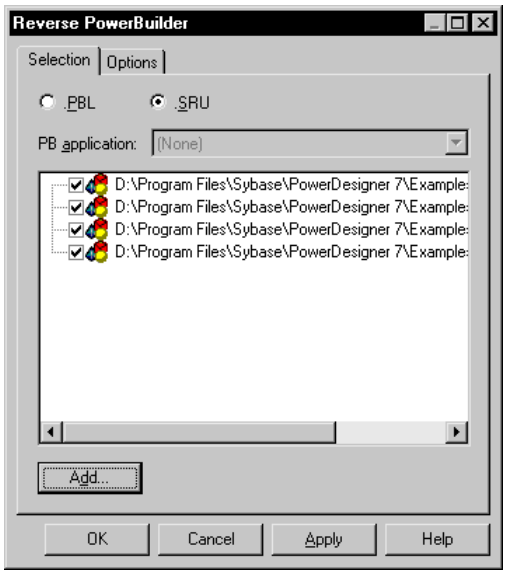
A standard Open dialog box appears.



4 Select the SRU files that you want to reverse and click Open.

> **Multi-selection**
> You can select several files simultaneously by using the CTRL or SHIFT keys.

You return to the Reverse PowerBuilder dialog box. It displays the files you selected.



5 Click OK.

A Progress box appears and the classes are added to your model. The classes are visible in the diagram and in the Browser.

The reversed classes are listed in the Reverse page of the Output window, situated in the bottom part of the PowerDesigner main window.

Reverse engineering XML

You can reverse engineer one of the following types of XML file to an OOM:

- XML - DTD Provides an overall structure for an XML file in DTD format.
- XML - Schema Provides an overall structure for an XML file in Schema format.
- XML - Data All other XML documents describing data or schemas.

When you reverse a DTD file into an OOM, you get more readable view of the DTD. This feature can be very helpful when you want to observe and understand a new DTD that you have not generated.

XML - DTD

When you reverse engineer a DTD file:

- Elements of type #PCDATA are reversed as attributes.
- An element that has both a parent and a child element is linked to its parent element by an aggregation link.
- If an empty element has no child object but has attributes, it is reversed as a class and its attributes become attributes of the class.
- Attributes of type ID and IDREF(S) with ID and IDREF(S) datatypes can be changed into associations.

XML - Schema

When you reverse engineer an XML - Schema file:

- <type> elements are reversed as classes.
- An <element> not declared as <type> is reversed as an attribute.

XML - Data

The XML Mapping in the XOL file defines which element becomes a class, an attribute or an association.

Reverse engineering XML options

You define XML reverse engineering options from the Reverse XML dialog box.

You can define the following XML reverse engineering options:

Option	Result of selection
Create symbols	Creates a symbol for each reversed XML object in the diagram. Otherwise, reversed objects are visible only in the browser

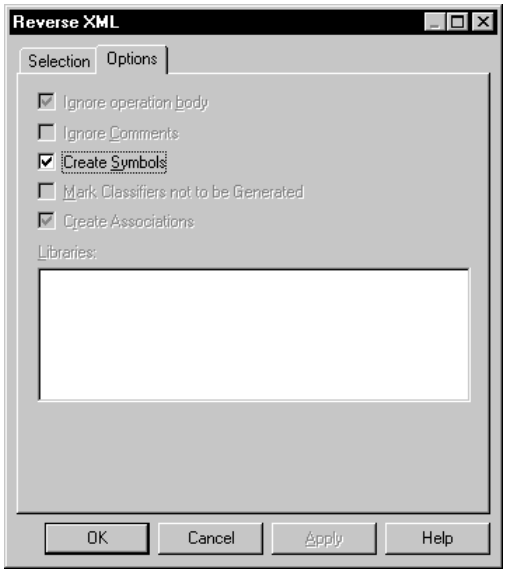
❖ **To define XML reverse engineering options:**

1 Select Language➤Reverse Engineer XML.

The Reverse XML dialog box opens.

2 Click the Options tab.

The Options page appears.



3 Select XML reverse options.

4 Click Apply.

5 Click Cancel.

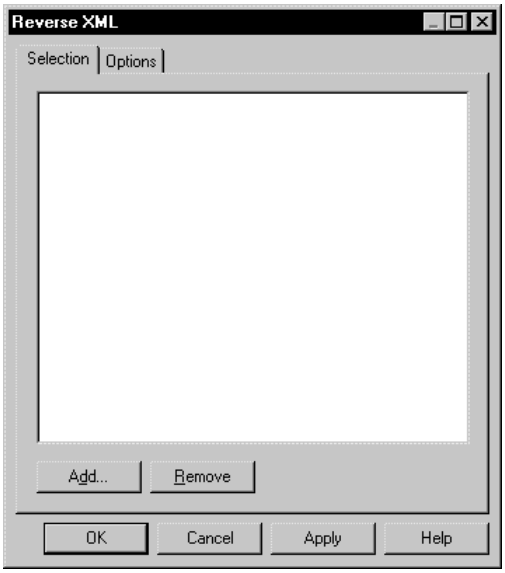
Reverse engineering XML files

XML files can be reversed into an OOM.

❖ **To reverse engineer an XML file:**

1 Select Language➤Reverse Engineer XML.

The Reverse XML dialog box opens to the Selection page.



2 Click the Add button.

A standard Open dialog box appears.

3 Select the files that you want to reverse and click Open.

> **Multi-selection**
> You can select several files simultaneously by using the CTRL or SHIFT keys.

You return to the Reverse XML dialog box. It displays the files you selected.

4 Click OK.

A Progress box appears and the objects are added to your model. The objects are visible in the diagram and in the Browser.

The reversed classes are listed in the Reverse page of the Output window, situated in the bottom part of the PowerDesigner main window.

Reverse engineering into a new OOM

You can reverse engineer object language files (Java, PowerBuilder, XML) into a new OOM.

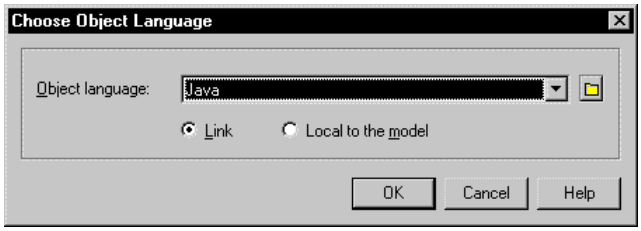
❖ **To reverse engineer object language files into a new OOM:**

1 Select File➤Reverse Engineering➤Object Language.

The Choose Object Language dialog box appears.

2 Click the Link radio button.

3 Select an object language in the dropdown list box.



4 Click OK.

Depending on the chosen object language the corresponding dialog box appears to let you select a file and reverse options.

5 Click OK to start reverse engineering.

A message in the Output window indicates that the specified file is fully reverse engineered.

CHAPTER 5

Generating Objects from an OOM

About this chapter

This chapter describes how to generate objects from an OOM.

Contents

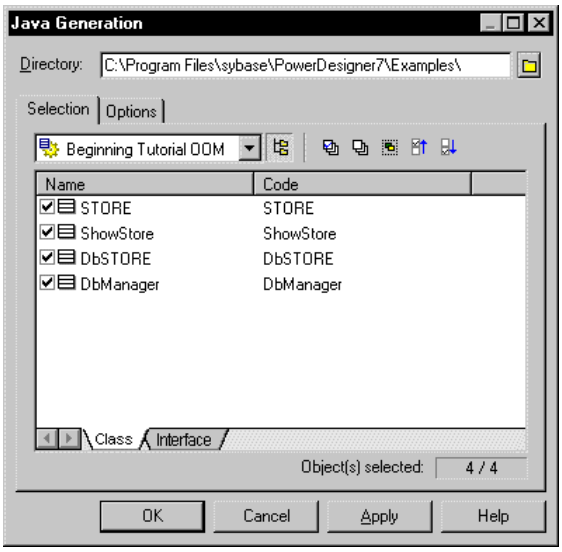
Generating objects

You can generate the following types of objects from an OOM:

Object type	What is generated
Java source files	.java files from the classes and interfaces of the model that you can then compile using a Java compiler
PowerBuilder objects	PowerBuilder NVO (non-visual objects) that you can use directly in PowerBuilder
Java BeanInfo classes	Java BeanInfo classes that you generate from the classes in a model
XML objects	XML definition files and enhanced definition files such as XML schema

Selecting objects to include in the generation

You select objects for generation from the Selection page.



Listing objects contained in a model or package

You can display in the list, objects in the current model, or objects in individual packages contained in the model.

If you select the Include Sub-Packages tool, you can display in the list either all objects in the current model, or all objects in a package.

You have the following selection options:

Parent object	Include Sub-Packages	Displays
Model	Selected	All objects in model including all objects contained in packages and sub-packages
Model	Not selected	All objects in model except objects contained in packages and sub-packages
Package	Selected	All objects contained in package including all objects contained in sub-packages
Package	Not selected	All objects in package except objects contained in sub-packages

Graphically selected objects
Graphically selected objects in your model can be automatically selected for generation by clicking the Use Graphical Selection tool in the Selection page tool bar.

Selecting objects

Then you can select the objects that you want to generate using the following select tools:

Tool	Action	When selected
	Include Sub-Packages	Displays objects contained in sub-packages
	Select All	Selects all objects in the model
	Unselect All	Deselects all objects in the model
	Use graphical selection	Selects graphically selected objects in the model diagram window
	Moves selection to top	Moves the selection to the top of the object list
	Moves selection to bottom	Moves the selection to the bottom of the object list

Selection tips
If you want to use a selection tool for all object type pages, press CTRL + the desired selection tool.

Generating Java source files

You generate Java source files from the classes and interfaces of a model. A separate file, with the file extension .java, is generated for each class or interface that you select from the model. You can only generate Java files from one model at a time.

You can compile the .java class files that you generate from an OOM in any Java compiler tool. You can also run Java in a database server such as Sybase Adaptive Server Anywhere. Using Sybase Adaptive Server Anywhere, you can call Java from SQL by calling Java functions (methods) from SQL statements. Java methods provide a more powerful language than SQL stored procedures for adding logic to the database.

You can use Java classes as data types. Every Java class installed in a database becomes available as a data type that can be used as the data type of a column in a table.

You can save Java objects in tables. An instance of a Java class (a Java object) can be saved as a value in a table. Java objects can be inserted into a table, SELECT statements can be executed against the fields and methods of objects stored in a table, and Java objects can be retrieved from a table.

Defining Java generation options

You can set Java generation options to check a model for errors before generating, or to sort the order in which the attributes and operations of the class in a Java class definition file are displayed according to visibility or type sort criteria.

Check model

You can check the model before generation. The generation stops if an error is found.

Visibility sort

You can sort the order in which attributes and operations are arranged in the code of the classes by the following criteria.

Option	Result of selection
Public - Private	After generation, public attributes and operations are placed before private attributes and operations in the class definition
Private - Public	After generation, private attributes and operations are placed before public attributes and operations in the class definition
None	Attributes and operations order remains unchanged after generation

Type sort

You can sort the order that attributes and operations are arranged in the code of the classes of the model.

Option	Result of selection
Attributes - Operations	Displays the class attributes before the operations in the generated class definition
Operations - Attributes	Displays the class operations before the attributes in the generated class definition

You define Java generation options from the Java Generation dialog box.

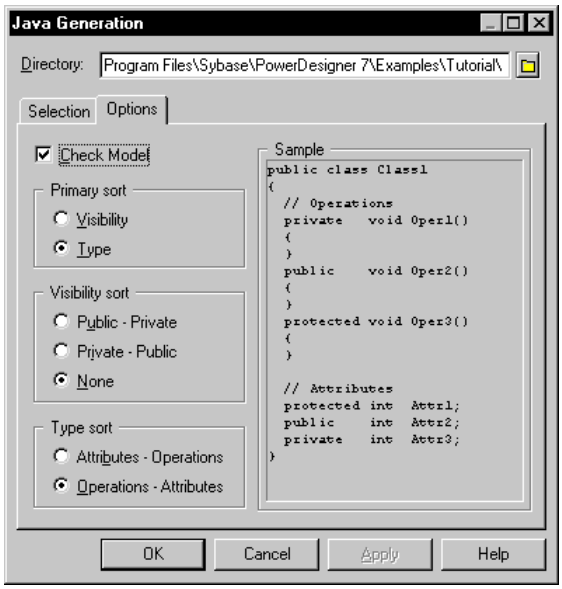
❖ **To define Java generation options:**

1 Select Language➤Generate Java Code.

The Java Generation dialog box opens.

2 Click the Options tab.

The Options page appears.



3 Select Java generation options.

4 Click Apply.

5 Click Cancel.

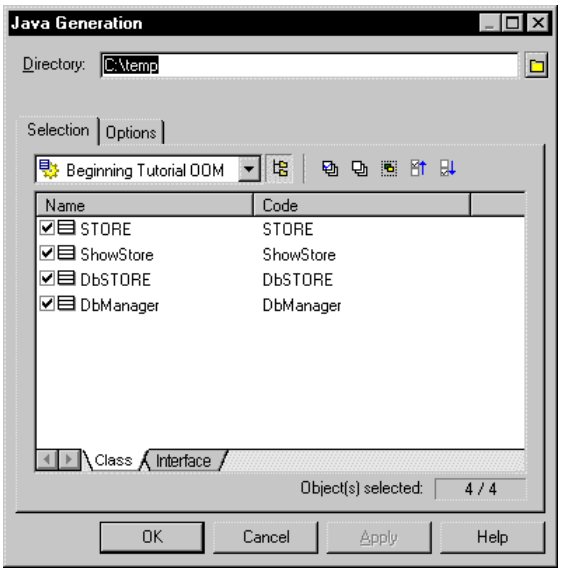
Generating Java class definition files

PowerDesigner generates a Java class definition file for each of the classes you select to generate in the Java Generation dialog box. You can select any of the classes from the model, including those that are contained within packages or sub-packages. The generated files contain the definition of each class and have the file extension .java.

❖ **To generate .java files:**

1 Select Language➤Generate Java Code.

The Java Generation dialog box opens.



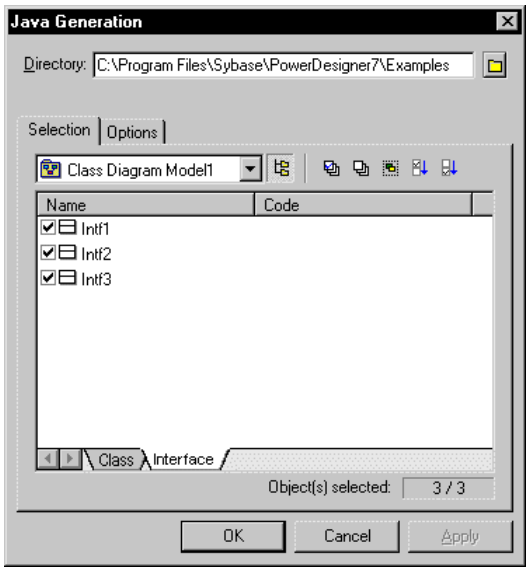
2 Type a destination directory for generated Java files in the Directory box.
or
Click the Browse to Folder button to the right of the Directory box and browse to select a directory path.

3 Select a model or package from the Folder Selection dropdown listbox.

4 Select the classes that you want to generate from the list.

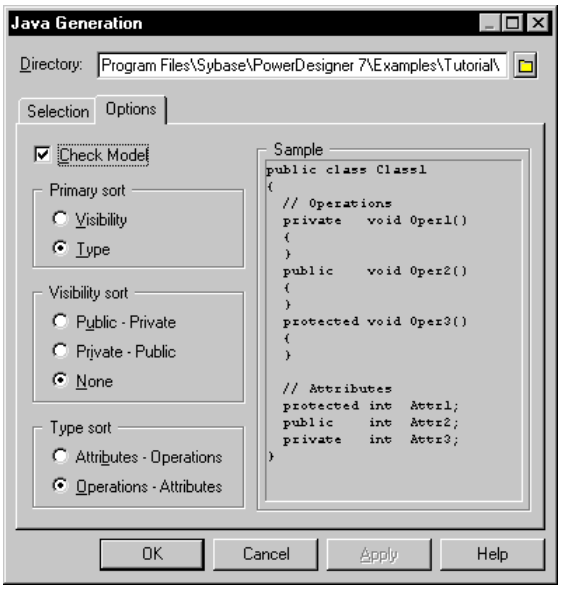
> **Select Tools**
> All the classes and interfaces of the model, including those that are grouped into packages, are selected and displayed by default. You can use the Select tools to the right of the Folder Selection dropdown listbox to modify the selection. The Include Sub-Packages tool, enables you to include in your selection all the classes and interfaces that are situated within packages.

5 Click the Interface tab and select the interfaces you want to generate.



6 Click the Options tab.

The Options page appears.



7 Select the Select Java generation options.

> **Navigating between pages**
> Use CTRL+PAGEDOWN or CTRL+PAGEUP to move to the next or to the previous tab and display the corresponding page.

8 Click OK.

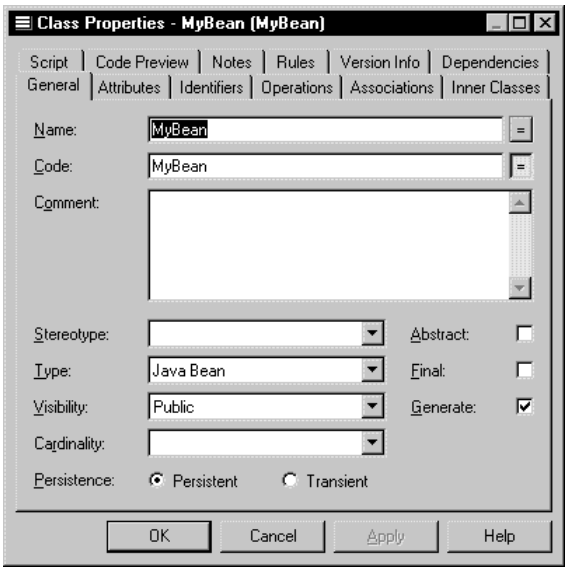
A Java class definition file is generated with the file extension .java for each of the classes that you selected.

Creating Java BeanInfo classes

A Java Bean is a reusable software component that can be visually manipulated in a software development tool.

You can create Java BeanInfo classes from the classes in an OOM. PowerDesigner generates a new BeanInfo class for each of the classes that you select in the model. You can select any of the classes from the model, including those that are contained within packages.

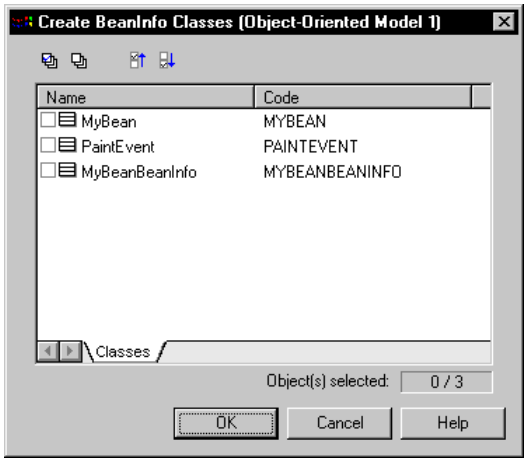
A BeanInfo class can only be created from a class if its type is Java Bean. You can define the type of a class from its property sheet:



❖ **To create Java BeanInfo classes:**

1 Select Language➤Create BeanInfo Classes.

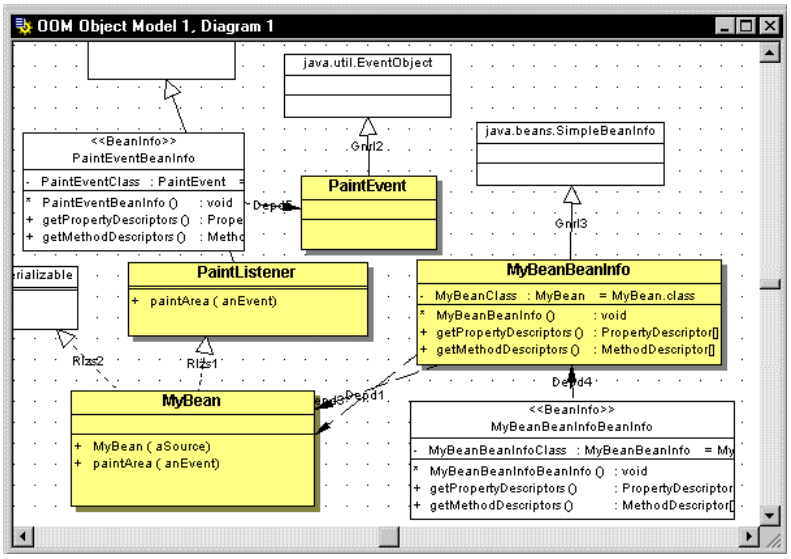
A selection window appears. It contains a list of all the classes in the model of type Java Bean.



2 Select the classes for which you want to generate Java BeanInfo classes.

3 Click OK.

A BeanInfo class is created in the model for each of the classes you selected.



Generating objects for PowerBuilder

You can generate PowerBuilder NVO (non-visual objects) from the classes of an OOM to either of the following:

- A PowerBuilder application
- SRU files

You can create PowerBuilder user objects only from the classes of the diagram and not from interfaces.

pbl application

You can generate PowerBuilder NVOs from an OOM that you can use directly in PowerBuilder. To generate to a PowerBuilder pbl application, you must have PowerBuilder already installed on your machine.

sru files

You can generate NVOs from the classes in an OOM. A separate file with the extension .sru is created for each of the classes that you select in the OOM. Each file contains a NVO corresponding to the definition of each class in the OOM.

Defining PowerBuilder generation options

You can set the following PowerBuilder generation options:

Option	Result of selection
Check model	Checks the model before generation and stops generation if an error is found
PBL	PowerBuilder library directory and application into which PowerDesigner generates
SRU	Directory in which you generate PowerBuilder non-visual object .sru files

PBL

When generating objects for a PowerBuilder application, you must make a selection in both the PB library and PB application fields. If PowerBuilder is not installed on your machine, you cannot generate objects for a PowerBuilder application.

Option	Result of selection
PB library	Directory into which PowerDesigner generates PowerBuilder library files
PB application	PowerBuilder application into which you generate PowerBuilder non-visual objects. If PowerBuilder is not installed on your machine, no application appears in the list

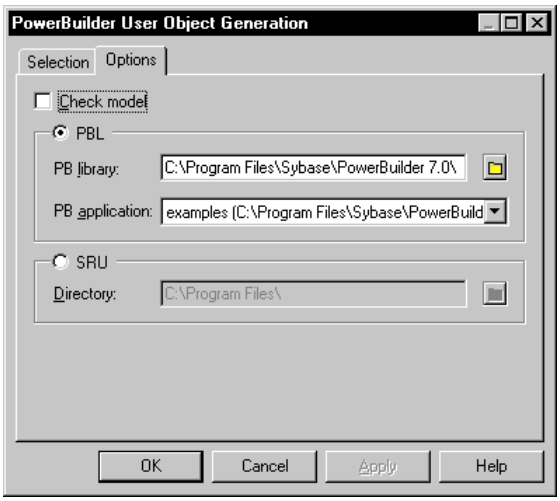
❖ **To define PowerBuilder generation options:**

1 Select Language➤Generate PowerBuilder.

The PowerBuilder User Object Generation dialog box opens.

2 Click the Options tab.

The Options page appears.



3 Select the PBL option, type or select a library directory in the PB library box, and select an application from the PB application Library listbox.
or
Click the SRU checkbox, and type or select a directory in which you want to generate the sru files.

4 Click Apply.

5 Click Cancel.

Generating objects for a PowerBuilder application

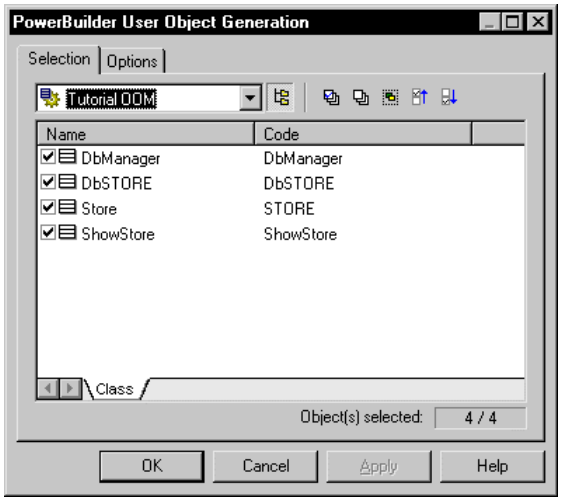
When you generate PowerBuilder objects, you must specify both the PowerBuilder library and the application that will use the objects, otherwise you will not be able to use them in PowerBuilder.

If PowerBuilder is not installed on your machine, you cannot generate objects for a PowerBuilder application.

❖ **To generate PowerBuilder user objects for a PowerBuilder application:**

1 Select Language➤Generate PowerBuilder.

The PowerBuilder User Object Generation dialog box opens.



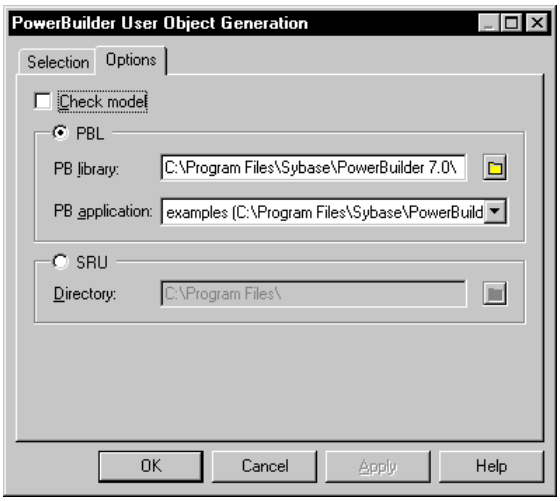
2 Select a model or package from the Folder Selection dropdown listbox.

3 Select the classes that you want to generate from the list.

Select Tools
All the classes of the model, including those that are grouped into packages, are selected and displayed by default. You can use the Select tools to the right of the Folder Selection dropdown listbox to modify the selection. The Include Sub-Packages tool, enables you to include in your selection all the classes that are situated within packages.

4 Click the Options tab.

The Options page appears.



5 Select the PBL option

6 Type a library directory for generated User Objects in the PB library box.
or
Click the Browse to Folder button to the right of the PB library box and browse to select a library.

7 Select a PowerBuilder application from the PB application dropdown listbox.

8 Click OK.

A PowerBuilder User Object is generated in the PowerBuilder application for each of the classes that you selected.

Generating PowerBuilder objects in sru files

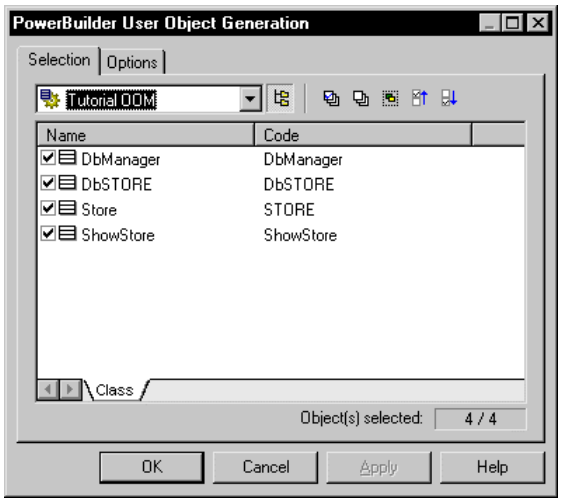
When you generate PowerBuilder objects in sru files, a separate file is created for each of the classes that you select in the OOM.

You do not have to have PowerBuilder installed on your machine to generate sru files.

❖ To generate PowerBuilder user objects in sru files:

1 Select Language➤Generate PowerBuilder.

The PowerBuilder User Object Generation dialog box opens.



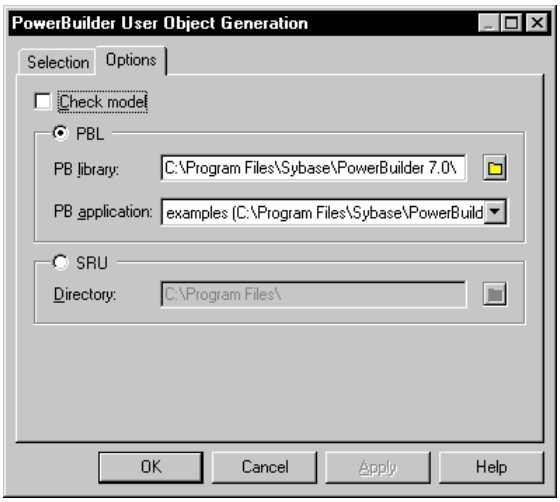
2 Select a model or package from the Folder Selection dropdown listbox.

3 Select the classes that you want to generate from the list.

> **Select Tools**
> All the classes of the model, including those that are grouped into packages, are selected and displayed by default. You can use the Select tools to the right of the Folder Selection dropdown listbox to modify the selection. The Include Sub-Packages tool, enables you to include in your selection all the classes that are situated within packages.

4 Click the Options tab.

The Options page appears.



5 Select the SRU option.

6 Select a directory in which you want to generate the .sru files.

7 Click OK.

A PowerBuilder User Object is generated with the file extension .sru for each of the classes that you selected.

Generating for XML

You can generate an XML DTD file from an OOM.

A DTD file provides an overall structure for an XML file. The DTD file can be used as a standard for validating data in XML files or for exchanging data in XML format.

You can generate an XML DTD in one of the following format types:

XML file format	Description
XML - DTD	Used for standard DTD specification. Each class is generated as an ELEMENT, with its attributes as sub-elements. Each Attribute is generated as a PCDATA ELEMENT
XML - Schema	Used for XML Schema specifications: Each class is generated as a <type>. Each attribute is generated as an <element>
XML - Data	Used for XML Data specification. Mapping is defined by the XOL specification

Navigable associations are migrated and generated as attributes, although they do have their own definition in the XOL file. You can specify a separate definition for a composition association.

Other objects such a interfaces, operations, and inheritance links are not included in the generated file.

Defining XML generation options

You can check a model before generation or simply generate directly. This generation option can be selected from the XML Generation dialog box.

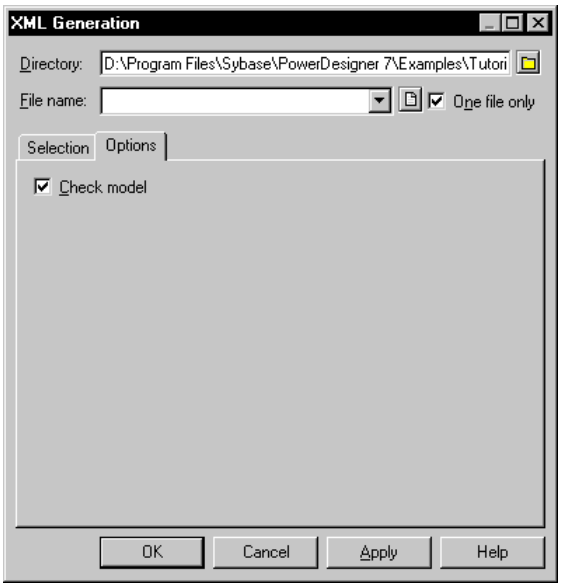
❖ **To define XML generation options:**

1 Select Language➤Generate XML.

The XML Generation dialog box opens.

2 Click the Options tab.

The Options page appears.



3 Select XML generation options.

4 Click Apply.

5 Click Cancel.

Generating XML objects

When you generate XML from an OOM, PowerDesigner creates an XML file containing the definition of each of the classes you select to generate in the XML Generation dialog box. You can select any of the classes from the model, including those that are contained within packages or sub-packages.

The generated file has the extension XML, however, its format depends on the current object language of the model. To change the XML format type, you must change the object language for the model.

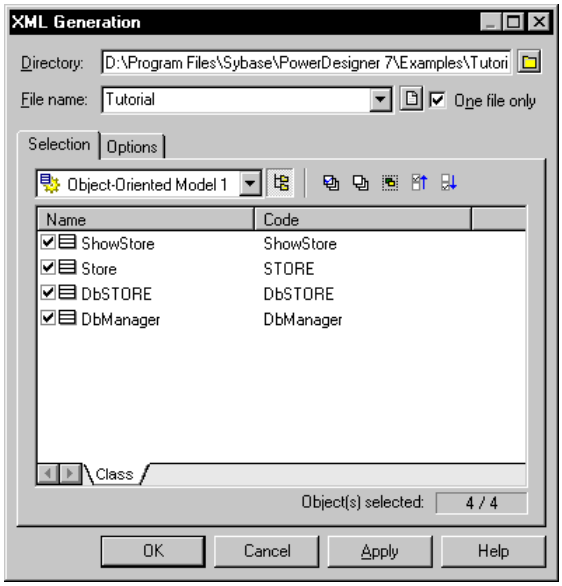
For more information on changing the current object language, see the chapter Object Language Properties.

You can create a new XML object language based on an existing one if you want to generate in another type of XML format that is different to those that are available with PowerDesigner.

❖ **To generate XML files:**

1 Select Language➤Generate XML.

The XML Generation dialog box opens.



2 Type a destination directory for generated XML file in the Directory box.
or
Click the Browse to Folder button to the right of the Directory box and browse to select a directory path.

3 Type a name for generated XML file in the File name box.

4 Select a model or package from the Folder Selection dropdown listbox.

5 Select the classes that you want to include in the generated file from the list.

> **Select Tools**
> All the classes of the model, including those that are grouped into packages, are selected and displayed by default. You can use the Select tools to the right of the Folder Selection dropdown listbox to modify the selection. The Include Sub-Packages tool, enables you to include in your selection all the classes that are situated within packages.

6 Click OK.

An XML file is generated with the file extension .xml.

Customizing scripts

You can customize scripts as follows:

- Insert scripts at the beginning and end of a script
- Insert scripts before and after a class or interface creation command

Customizing a creation script allows you to add descriptive information about a generated script, or manipulate the script in such a way that is not provided by PowerDesigner.

You can use the following variables in these scripts:

Variable	Description
%PACKAGE%	Name of the current package

CHAPTER 6

Generating a Conceptual Data Model from an Object-Oriented Model

About this chapter

This chapter describes how to generate a Conceptual Data Model (CDM) from an Object-Oriented Model (OOM).

Contents

Generating OOM objects to a CDM

When you generate a Conceptual Data Model (CDM) from an Object-Oriented Model (OOM), PowerDesigner translates OOM objects and data types to CDM objects and data types.

The current object language of an OOM has no effect on the generation to a CDM.

Translating OOM objects into CDM objects

CDM generation translates OOM objects into conceptual objects.

OOM object	CDM object after generation
Domain	Domain
Class	Entity (only if the Persistent and Generate checkboxes are selected in the class property sheet)
Interface	Not translated
Attribute	Attribute
Identifier	Identifier
Operation	Not translated
Association	Relationship or association
Dependency	Not translated
Realization	Not translated
Generalization	Inheritance

Translating OOM data types for a CDM

PowerDesigner supports both Java and conceptual data types. When you generate objects from an OOM to a CDM, Java data types are translated by PowerDesigner into conceptual data types. PowerDesigner conceptual data types cannot be modified.

Translating Java data types for a CDM

The following table lists the Java data types to which the object language file assigns translations:

Java data type	Code in CDM	What it stores
char	A	Character
boolean	BL	Two opposing values (true/false; yes/no; 1/0)
byte	BT	256 values
short	SI	16-bit integer
integer	I	32-bit integer
long	LI	32-bit integer
float	F	32-bit floating decimal numbers
double	N	Numbers with a fixed decimal point
String	TXT	Character strings

Generating a CDM from an OOM

You can generate a CDM from a global OOM or from a package within the model. Limiting CDM generation to a single package is useful when different designers own packages of the same OOM. Designers can generate their packages independently from others. Generating a package results in an independent CDM.

You generate a CDM from a diagram in the model.

You can generate a CDM in two ways:

Generate	Description
New CDM	Creates a new (default) CDM containing the objects translated from the OOM
Updated CDM	Creates a default CDM containing the objects translated from the OOM that is then merged with an existing CDM. You can choose to update, delete; or add objects in the existing CDM (target model) based on modifications made in the default CDM (source model)

For more information on merging two CDM, see the chapter Comparing and Merging Models in the PowerDesigner General Features Guide.

Generating and updating a CDM

To generate a CDM, you must indicate to generate one of the following:

- Generate new Conceptual Data Model
- Update an existing Conceptual Data Model

Generate new Conceptual Data Model

You must indicate the following parameters when you generate a new CDM:

Parameter	Description
Name	File name for the resulting CDM
Code	Reference code for the resulting CDM

Update existing Conceptual Data Model

You must indicate the following parameters when you update an existing CDM:

Parameter	Description
Select Model	Target Conceptual Data Model. This is the existing CDM that the newly generated CDM (source model) is merged with to create an updated CDM
Preserve Modifications	When selected, allows a comparison and merge of the newly generated CDM (default CDM) with the existing CDM

Clearing the Preserve Modifications checkbox
When Preserve modifications is not selected, PowerDesigner automatically replaces the selected target model (existing CDM) with the newly generated CDM. If you want to choose which objects to add or delete from the target model, you must select Preserve Modifications to compare and merge the two CDM.

CDM generation options

You can set the following general generation options:

Option	Description
Check model	Checks the model before generating the CDM, and stops generation if an error is found
Save generation dependencies	When selected, PowerDesigner keeps track of the identity of each generated object. This is useful when merging two CDM which have been generated from the same OOM. Objects can be compared and recognized as the same object, even if the object has been modified in the target CDM
Model Notation	Indicates the modeling methodology used in the generated CDM. You can choose Entity/Relationship, Merise, or Mixed. If you select Mixed, the two methodologies are available in the same model

Check model before generation
If you select the Check Model option, the procedure to generate a CDM starts by checking the validity of the OOM or package. A CDM results when no errors are found. You can set check options by selecting Tools➤Check Model.

Object selection parameters

You select objects for CDM generation from the Selection page.

Listing objects contained in a model or package

You can display in the list, objects in the current model, or objects in individual packages contained in the model.

If you select the Include Sub-packages tool, you can display in the list either all objects in the current model, or all objects in a package.

You have the following selection options:

Parent object	Include Sub-packages	Displays
Model	Selected	All objects in model including all objects contained in packages and sub-packages
Model	Not selected	All objects in model except objects contained in packages and sub-packages
Package	Selected	All objects contained in package including all objects contained in sub-packages
Package	Not selected	All objects in package except objects contained in sub-packages

Objects selected in the model
Objects selected in your diagram can be automatically selected for generation by clicking the Use Graphical Selection tool in the Selection page tool bar.

Generating a new CDM

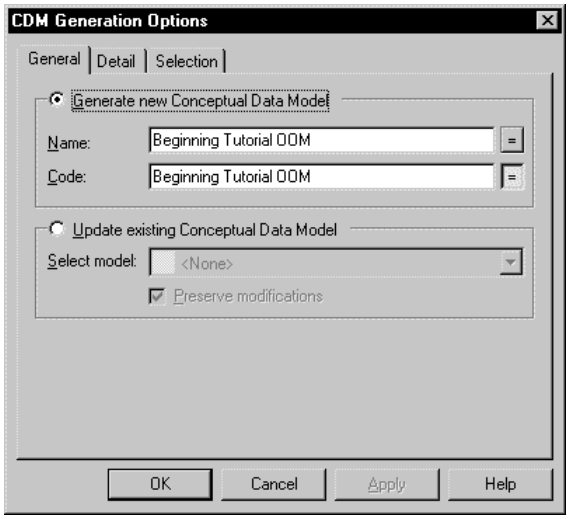
When you generate from an OOM to a new CDM, PowerDesigner creates a new CDM containing all the objects that you selected to generate in the OOM. The newly created CDM appears in the browser and the corresponding diagram opens in the main diagram window.

You can only generate a CDM from the active OOM diagram window.

❖ **To generate to a new CDM from an OOM:**

1 Select Tools➤Generate Conceptual Data Model.

The CDM Generation Options dialog box appears.

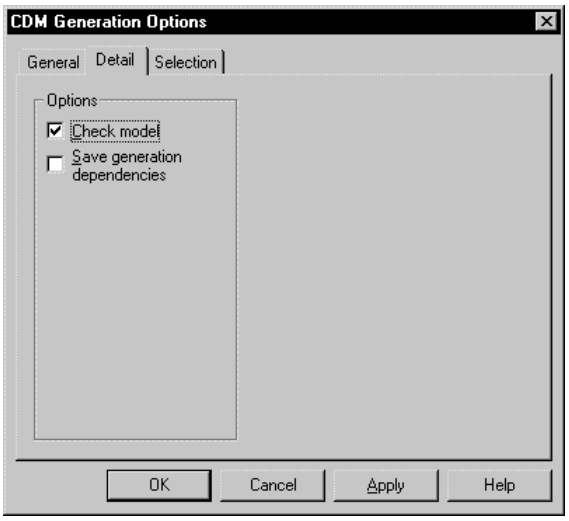


2 Click the Generate new Conceptual Data Model radio button.

3 Type a new name and code, otherwise, the CDM will have the same name and code as the OOM.

4 Click the Detail tab.

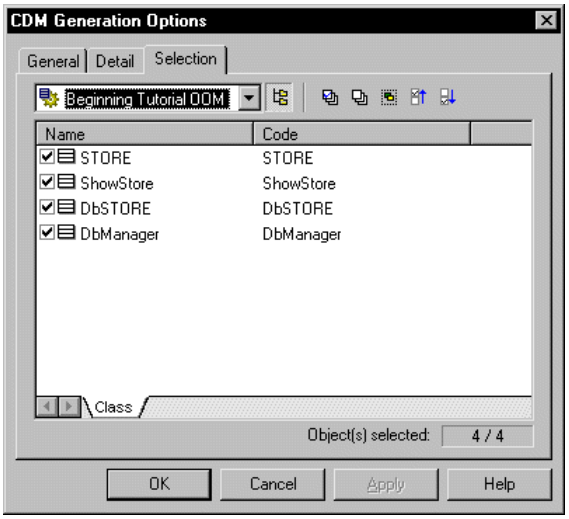
The Detail page appears.



5 Select or clear CDM generation options.

6 Click the Selection tab.

The Selection page appears.



7 Select the name of an OOM from the Select Location dropdown list.

Generating a CDM from a package
To generate a CDM from a package, select the package name from the Select Location dropdown listbox at the top of the page.

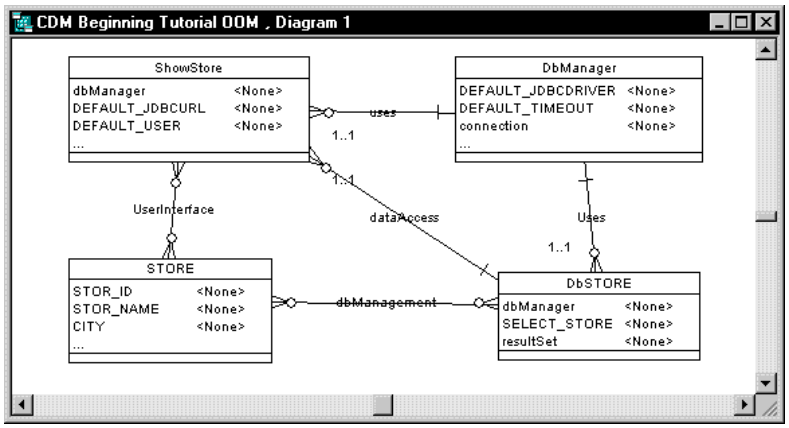
To generate CDM from a sub-package, select the Sub-Packages Included tool next to the Selection Location dropdown listbox, and then select a sub-package from the dropdown listbox.

8 Select checkboxes corresponding to each entity that you want to generate.

9 Clear checkboxes corresponding to each entity that you do not want to generate.

10 Click OK.

The Output window shows the progress of the generation process. The new CDM appears in the diagram window.



Updating an existing CDM

There are two ways to update an existing CDM depending on whether the Preserve Modifications options is selected or not selected:

Preserve Modifications	Result
Selected	You can manually compare and merge existing CDM (target model) with the newly generated CDM (source model)
Not selected	The existing CDM is automatically replaced by the newly generated CDM

When Preserve Modifications is selected, the Merge Models window appears after the new CDM has been successfully generated. You can use the Merge window to select objects to be updated, deleted, or added to the target model.

The target model must be open in the workspace to be merged with a source model.

You can only generate a CDM from the active OOM diagram window.

The existing CDM, into which you want to generate objects from the OOM, must be open in the workspace.

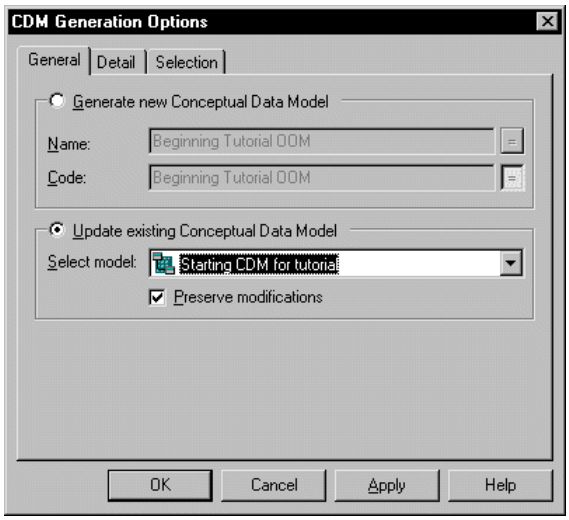
❖ **To update an existing CDM by generating from an OOM:**

1 Select Tools➤Generate Conceptual Data Model.

The CDM Generation Options dialog box appears. If you do not have a CDM in the current Workspace, the Update existing Conceptual Data Model option is not available.

2 Select the Update existing Conceptual Data Model radio button.

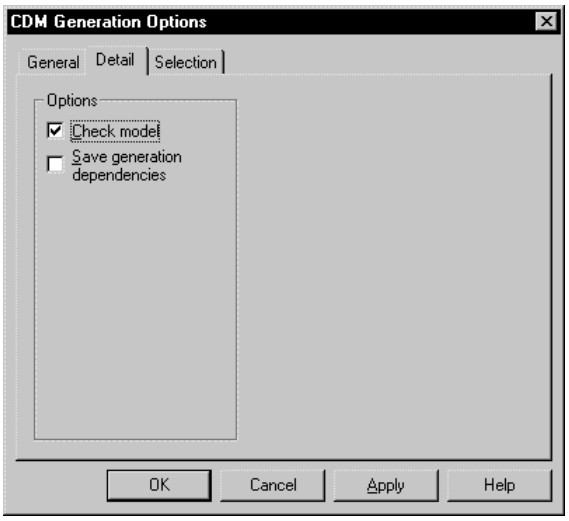
3 Select a target model from the Select Model dropdown listbox. This is the existing model that you want to update.



Preserve modifications
If you want to preserve the existing objects in the CDM, then the Preserve modifications checkbox must be selected. If you clear this checkbox, all existing objects in the CDM will be removed from the model, leaving only the objects generated from the OOM.

4 Click the Detail tab.

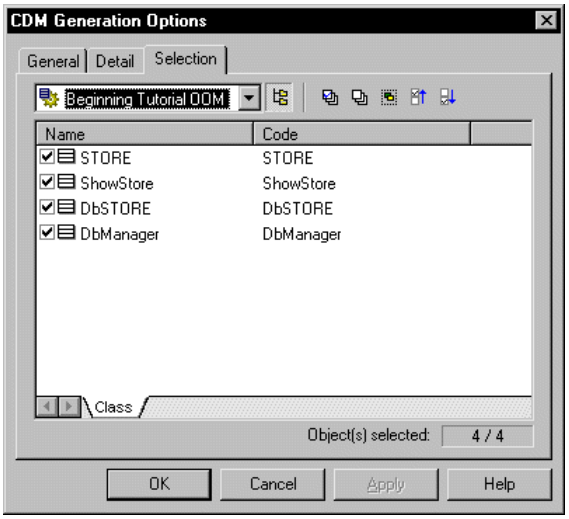
The Detail page appears.



5 Select or clear CDM generation options.

6 Click the Selection tab.

The Selection page appears.



7 Select the name of an OOM from the Select Location dropdown list. The default CDM is generated from this OOM.

Generating a CDM from a package
To generate a CDM from a package, select the package name from the Select Location dropdown listbox at the top of the page.

To generate CDM from a sub-package, select the Sub-Packages Included icon next to the Selection Location dropdown listbox, and then select a sub-package from the dropdown listbox.

8 Select class checkboxes for each entity that you want to generate.
or
Clear class checkboxes for each entity that you do not want to generate

9 Click OK.

If you selected the Preserve Modifications checkbox, the Merge Models window appears.

If you cleared the Preserve Modifications checkbox, the updated CDM appears in the diagram window.

Merging models
The Merge Models dialog box shows the newly generated CDM in the Source Model pane, and the existing CDM in the Target Model pane. You can select or clear object check boxes in the Source Model pane for CDM objects that you want to be included or deleted in the target model.

For more information on merging models, see the chapter Comparing and Merging Models in the PowerDesigner General Features Guide.

CHAPTER 7

Generating a Physical Data Model from an Object-Oriented Model

About this chapter

This chapter describes how to generate a Physical Data Model (PDM) from an Object-Oriented Model (OOM).

Contents

Generating OOM objects to a PDM

When you generate a Physical Data Model (PDM) from an Object-Oriented Model (OOM), PowerDesigner translates OOM objects and data types to PDM objects and data types supported by the current DBMS.

The current object language of an OOM has no effect on the generation to a PDM.

Translating OOM objects into PDM objects

PDM generation translates OOM objects into physical objects.

OOM object	PDM object after generation
Domain	Domain
Class	Table (only if the Persistent and Generate checkboxes are selected in the class property sheet). The cardinality of a class becomes the number of records of a table
Interface	Not translated
Attribute	Column
Identifier	Identifier
Operation	Stored-Procedure
Association	Reference or table
Dependency	Not translated
Realization	Not translated
Generalization	Reference

Generating from classes

For a class to become an table the Persistent and Generate checkboxes must be selected in the property sheet of the class.

The cardinality of a class becomes the number of records of a table.

Generating from associations

If the association has a many-to-many cardinality, that is, where both roles of the association have the * sign selected in their multiplicity dropdown listboxes, then the association is translated into a table in the generated PDM. If it has any other cardinality, that is, where one of the roles of the association does not have an * selected in its multiplicity dropdown listbox, then the association becomes a reference.

A role name becomes a migrated foreign key after PDM generation.

Translating OOM data types for a PDM

PowerDesigner supports both Java and physical data types. Data types that you select in the OOM are not always supported by the current DBMS. In this case, the data type is translated to a data type supported by the DBMS when you generate the PDM.

Translating Java data types for a PDM

The following table lists the Java data types to which the object language file assigns translations:

Java data type	Code in CDM	What it stores	Translation example for SQL Anywhere
char	A	Character	char
boolean	BL	Two opposing values (true/false; yes/no; 1/0)	numeric(1)
byte	BT	256 values	smallint
short	SI	16-bit integer	integer
integer	I	32-bit integer	integer
long	LI	32-bit integer	integer
float	F	32-bit floating decimal numbers	float
double	N	Numbers with a fixed decimal point	numeric
String	TXT	Character strings	long varchar

Generating a PDM from an OOM

You can generate a PDM from a global OOM or from a package within the model. Limiting PDM generation to a single package is useful when different designers own packages of the same OOM. Designers can generate their packages independently from others. Generating a package results in an independent PDM .

You generate a PDM from a diagram in the model.

You can generate a PDM in two ways:

Generate	Description
New PDM	Creates a new (default) PDM containing the objects translated from the OOM
Updated PDM	Creates a default PDM containing the objects translated from the OOM that is then merged with an existing PDM. You can choose to update, delete; or add objects in the existing PDM (target model) based on modifications made in the default PDM (source model)

For more information on merging two PDM, see the chapter Comparing and Merging Models in the PowerDesigner General Features Guide.

Generating and updating a PDM

To generate a PDM, you must indicate to generate one of the following:

- Generate new Physical Data Model
- Update existing Physical data Model

Generate new Physical Data Model

You must indicate the following parameters when you generate a new PDM:

Parameter	Description
DBMS	Database Management System definition (DBMS) for the resulting PDM
Link	DBMS for the resulting PDM refers to the DBMS definition file stored in the DBMS library
Local to the Model	DBMS for the resulting PDM is a copy of the DBMS definition file stored in the DBMS library
Name	File name for the resulting PDM
Code	Reference code for the resulting PDM

Update existing Physical Data Model

You must indicate the following parameters when you update an existing PDM:

Parameter	Description
Select Model	Target Physical Data Model. This is the existing PDM that the newly generated PDM (source model) is merged with to create an updated PDM
DBMS	Current Database Management System definition (DBMS) for the existing PDM
Preserve Modifications	When selected, allows a comparison and merge of the newly generated PDM (default PDM) with the existing PDM

Clearing the Preserve Modifications checkbox
When Preserve modifications is not selected, PowerDesigner automatically replaces the selected target model (existing PDM) with the newly generated PDM. If you want to choose which objects to add or delete from the target model, you must select Preserve Modifications to compare and merge the two PDM.

Defining PDM generation options

You can set the following general generation options:

Option	Description
Check model	Checks the model before generating the PDM, and stops generation if an error is found
Save generation dependencies	When selected, PowerDesigner keeps a record of which model was generated from
Table prefix	Helps you identify a table more easily in the model
Update Rule	Update referential integrity defined for references
Delete Rule	Delete referential integrity defined for references
PK index names	Primary key index name
Key index names	Alternate key index name
FK index names	Foreign key index name
FK threshold	Minimum number of estimated records in a table that are necessary before a foreign key index can be created

Check model before generation
If you select the Check Model option, the procedure to generate a PDM starts by checking the validity of the OOM or package. A PDM results when no errors are found. You can set check options by selecting Tools➤Check Model.

Object selection parameters

You select objects for PDM generation from the Selection page.

Listing objects contained in a model or package

You can display in the list, objects in the current model, or objects in individual packages contained in the model.

If you select the Include Sub-packages tool, you can display in the list either all objects in the current model, or all objects in a package.

You have the following selection options:

Parent object	Include Sub-packages	Displays
Model	Selected	All objects in model including all objects contained in packages and sub-packages
Model	Not selected	All objects in model except objects contained in packages and sub-packages
Package	Selected	All objects contained in package including all objects contained in sub-packages
Package	Not selected	All objects in package except objects contained in sub-packages

Objects selected in the model
Objects selected in your diagram can be automatically selected for generation by clicking the Use Graphical Selection tool in the Selection page tool bar.

Generating a new PDM

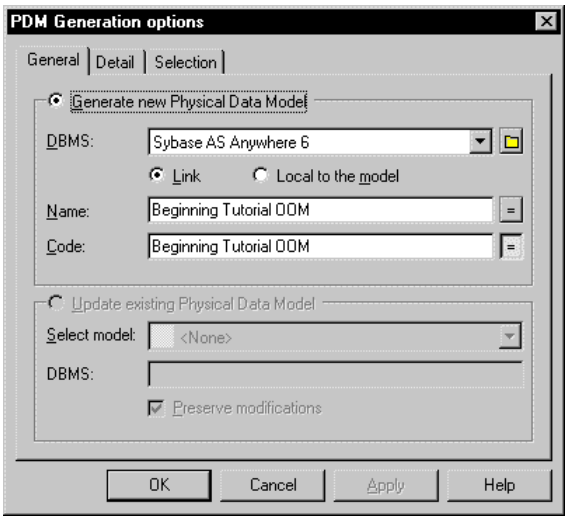
When you generate from an OOM to a new PDM, PowerDesigner creates a new PDM containing all the objects that you selected to generate in the OOM. The newly created PDM appears in the browser and the corresponding diagram opens in the main diagram window.

You can only generate a PDM from the active OOM diagram window.

❖ **To generate to a new PDM from an OOM:**

1 Select Tools➤Generate Physical Data Model.

The PDM Generation Options dialog box appears.



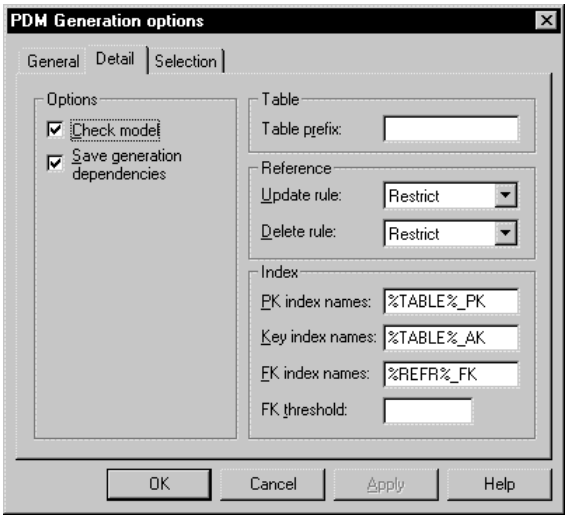
2 Click the Generate new Physical Data Model radio button.

3 Select the DBMS you want to be associated to your model from the DBMS dropdown listbox.

4 Type a new name and code, otherwise, the PDM will have the same name and code as the OOM.

5 Click the Detail tab.

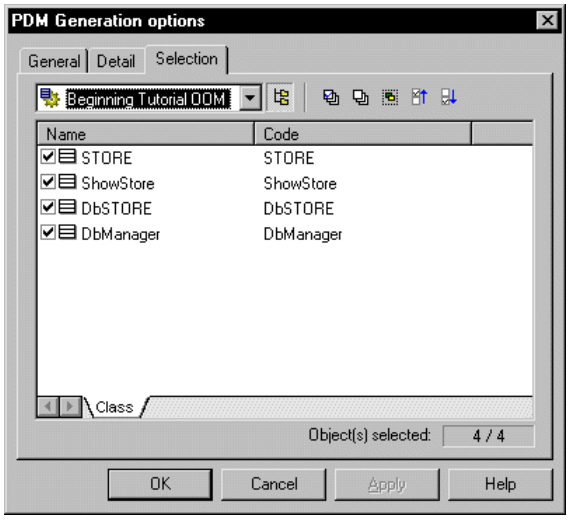
The Detail page appears.



6 Select or clear PDM generation options.

7 Click the Selection tab.

The Selection page appears.



8 Select the name of an OOM from the Select Location dropdown list.

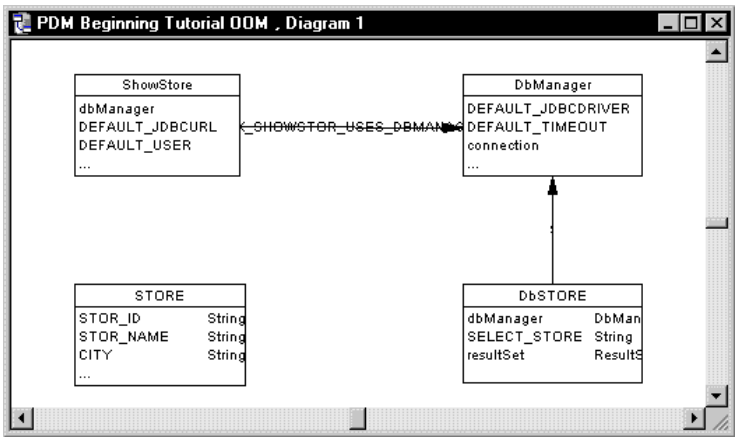
> **Generating a PDM from a package**
> To generate a PDM from a package, select the package name from the Select Location dropdown listbox at the top of the page.
>
> To generate PDM from a sub-package, select the Sub-Packages Included tool next to the Selection Location dropdown listbox, and then select a sub-package from the dropdown listbox.

9 Select checkboxes corresponding to each table that you want to generate.

10 Clear checkboxes corresponding to each table that you do not want to generate.

11 Click OK.

The Output window shows the progress of the generation process. The new PDM appears in the diagram window.



Updating an existing PDM

There are two ways to update an existing PDM depending on whether the Preserve Modifications options is selected or not selected:

Preserve Modifications	Result
Selected	You can manually compare and merge existing PDM (target model) with the newly generated PDM (source model)
Not selected	The existing PDM is automatically replaced by the newly generated PDM

When Preserve Modifications is selected, the Merge Models window appears after the new PDM has been successfully generated. You can use the Merge window to select objects to be updated, deleted, or added to the target model.

The target model must be open in the workspace to be merged with a source model.

You can only generate a PDM from the active OOM diagram window.

The existing PDM, into which you want to generate objects from the OOM, must be open in the workspace.

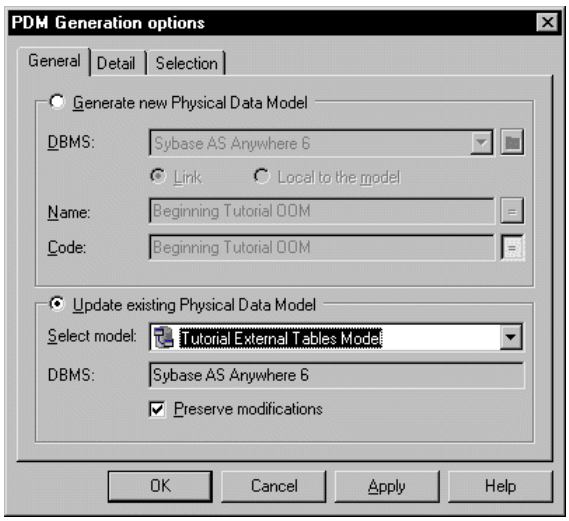
❖ **To update an existing PDM by generating from an OOM:**

1 Select Tools➤Generate Physical Data Model.

The PDM Generation Options dialog box appears. If you do not have a PDM in the current Workspace, the Update existing Physical Data Model option is not available.

2 Select the Update existing Physical Data Model radio button.

3 Select a target model from the Select Model dropdown listbox. This is the existing model that you want to update.

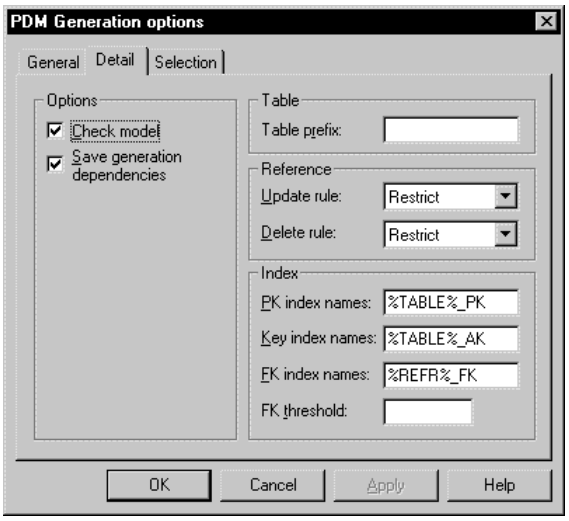


The DBMS that is attached to the model appears in the DBMS box.

Preserve modifications
If you want to preserve the existing objects in the PDM, then the Preserve modifications checkbox must be selected. If you clear this checkbox, all existing objects in the PDM will be removed from the model, leaving only the objects generated from the OOM.

4 Click the Detail tab.

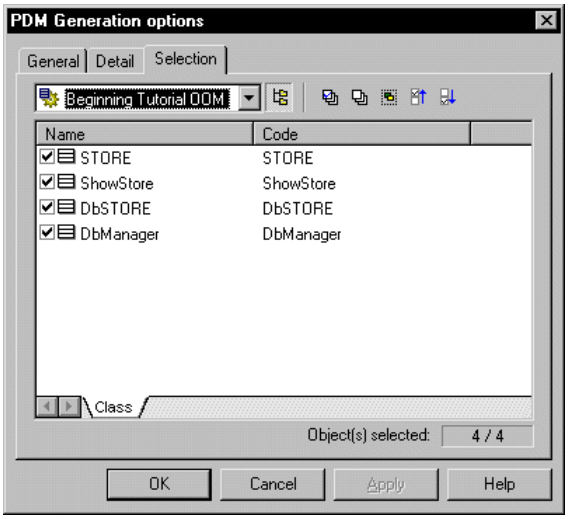
The Detail page appears.



5 Select or clear PDM generation options.

6 Click the Selection tab.

The Selection page appears.



7 Select the name of an OOM from the Select Location dropdown list. The default PDM is generated from this OOM.

Generating a CDM from a package
To generate a CDM from a package, select the package name from the Select Location dropdown listbox at the top of the page.

To generate CDM from a sub-package, select the Sub-Packages Included icon next to the Selection Location dropdown listbox, and then select a sub-package from the dropdown listbox.

8 Select class checkboxes for each table that you want to generate.
or
Clear class checkboxes for each table that you do not want to generate

9 Click OK.

If you selected the Preserve Modifications checkbox, the Merge Models window appears.

If you cleared the Preserve Modifications checkbox, the updated CDM appears in the diagram window.

Merging models
The Merge Models dialog box shows the newly generated PDM in the Source pane, and the existing PDM in the Target Model pane. You can select or clear object check boxes in the Source Model pane for PDM objects that you want to be included or deleted in the target model.

For more information on merging models, see the chapter Comparing and Merging Models in the PowerDesigner General Features Guide.

CHAPTER 8

Using Object Languages

About this chapter

This chapter explains how to use an object language in an OOM.

Contents

Object languages

An object language contains specifications for a particular language. It provides PowerDesigner with the syntax and guidelines for implementing stereotypes, data types, scripts and constants for an object language.

Every OOM is attached by default to an object language. When you create a new OOM, you choose an object language.

The definition for an object language can be edited from its property sheet, in which you can select and configure parameters that are used when defining objects or generating from an OOM.

You can attach only one particular object language to an OOM.

Types of object language

You can associate the following standard object languages to an OOM:

Object language type	Description
Analysis	General language in which you define parameters for models from which you want to generate a CDM or a PDM, or that you build simply for modeling purposes only
Standard Java	Standard Java language in which you can define parameters relating to Java code and generation
PowerBuilder	Standard PowerBuilder language in which you can define parameters relating to generating objects for PowerBuilder
XML – DTD	Standard XML language in which you can define parameters relating to generating objects in XML format
XML - Schema	The same as standard XML language properties, but also includes XML schema specifications used by certain XML
XML - Data	Used for XML Data specification. Mapping is defined by the XOL specification

Accessing object language properties

There are two different ways of accessing and modifying an object language properties:

Object language	Menu item	Description
Linked (to all models)	Tools➤Resources➤Object Languages	Used to define object languages for all models that are linked to this object language
Local to the model	Language➤Edit Current Object Language	Used to define the object language that is local to the current OOM

Modifying the current object language

You can modify the properties of the object language that is associated to the current model.

If the object language of the current model is local to the model, then any changes you make to the object language apply only to the current model.

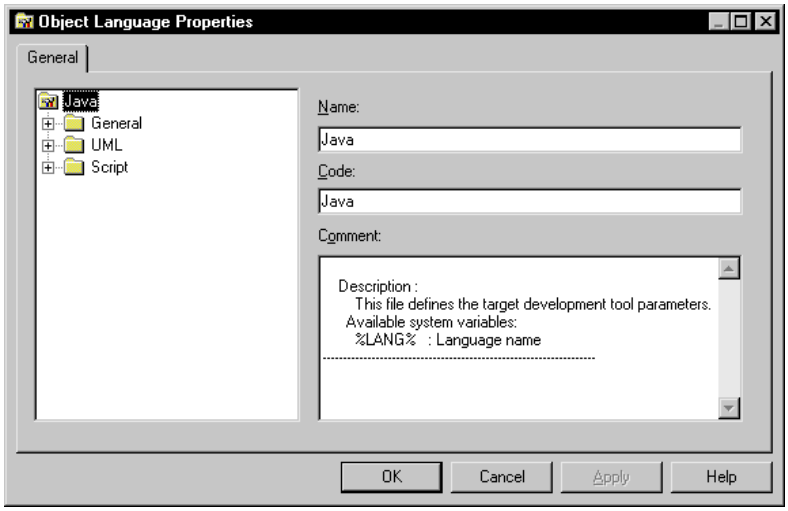
If the current model is linked to an object language, then any changes you make to the object language properties apply to all models that are linked to the object language.

❖ **To modify a value of a current object language:**

1 Open an OOM.

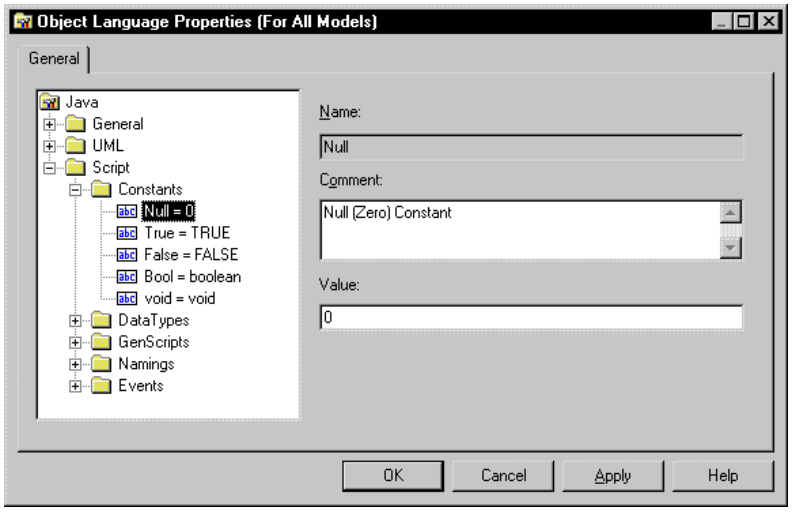
2 Select Language➤Edit Current Object Language.

The Object Languages Properties dialog box appears. In the left pane is a list of categories and sub-categories in which are contained the values that you can modify.



3 Expand a category node (and its sub-category if it has one), and select a value.

The name, associated comment, and value of the field appear in the zone to the right of the explorer window.



4 Modify the comment or value as required.

5 Click OK.

The next time you open an OOM, the modifications that you made in the object language editor will remain the same for the newly opened model.

Modifying linked object language properties

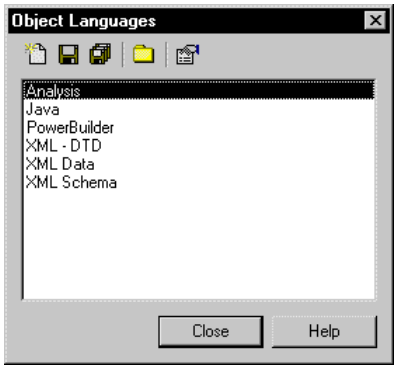
You can change any of the parameters of existing linked object languages. The changes you make apply to all models that are linked to the object language.

❖ **To modify the parameters of a linked object language:**

1 Open an OOM.

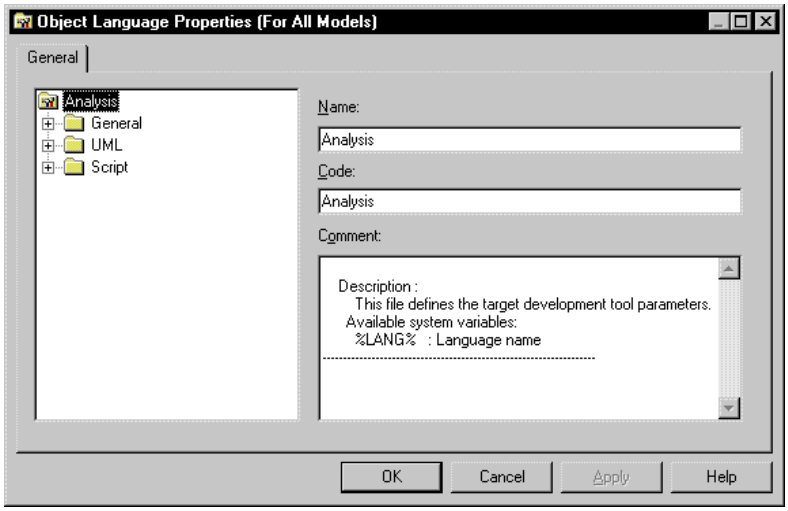
2 Select Tools➤Resources➤Object Languages.

The Object Languages window appears.



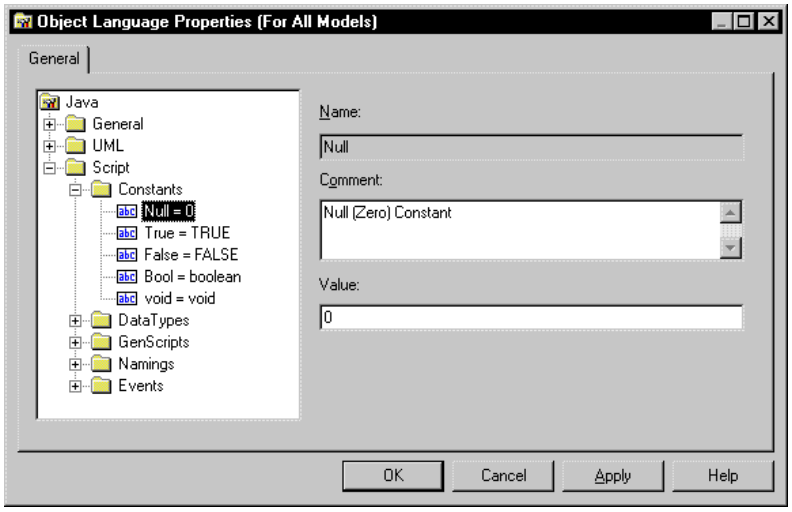
3 Select an object language and click Properties.
or
Double-click an object language.

The Object Languages Properties dialog box appears. In the left pane is a list of categories and sub-categories in which are contained the values that you can modify.



4 Expand a category node (and its sub-category if it has one), and select a value.

The name, associated comment, and value of the field appear in the zone to the right of the explorer window.



5 Modify the comment or value as required.

6 Click OK.

Changing the object language of an OOM

You can change the object language for an OOM, defining the new object language as being local to the model or as being linked to the model.

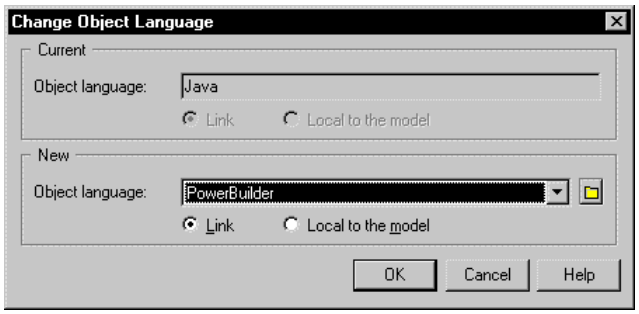
❖ **To change the object language of an OOM:**

1 Open an OOM.

2 Select Language➤Change Current Object Language.

The Change Object Language window appears.

3 Select a new object language from the Object Language dropdown listbox in the New groupbox.



4 Select the Link radio button if you want the new object language to be the general object language available for all models.
or
Select the Local to the model radio button if you want the new object language properties to apply only to the current model.

The chosen object language becomes the new one for the current model.

Creating a new object language

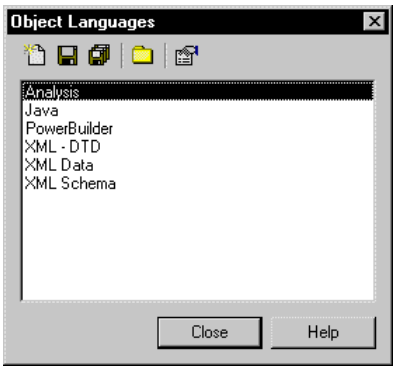
You can create a new set of object language properties and associate them to an OOM.

❖ **To create a new object language:**

1 Open an OOM.

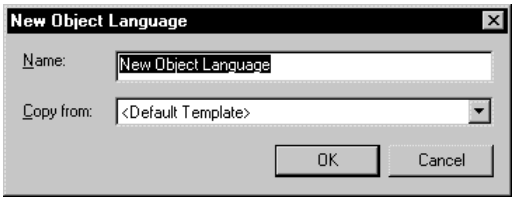
2 Select Tools➤Resources➤Object Languages.

The Object Languages window appears.



3 Click the New tool.

The New Object Language window appears.

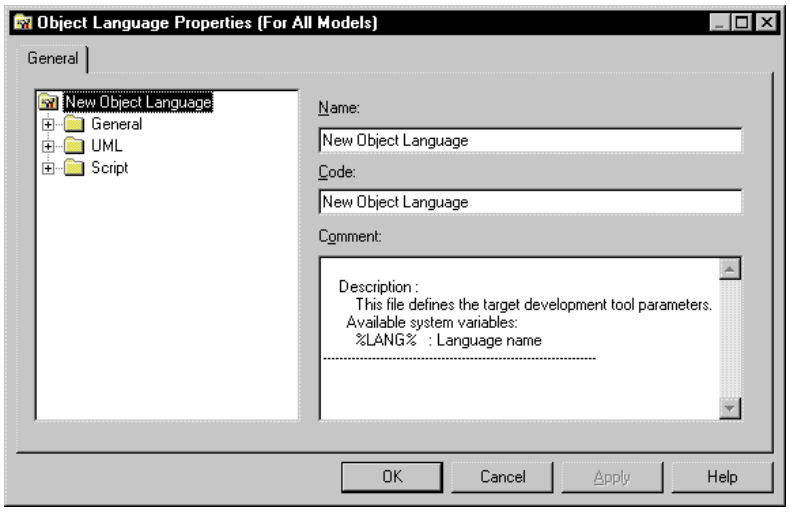


4 Type a name for the new object language in the Name box.

5 Select an existing object language from the Copy from dropdown listbox if you want the new object language to be based on an existing one.

6 Click OK.

The Object Languages Properties dialog box appears.



7 Expand the category nodes as appropriate and modify comments or values as required.

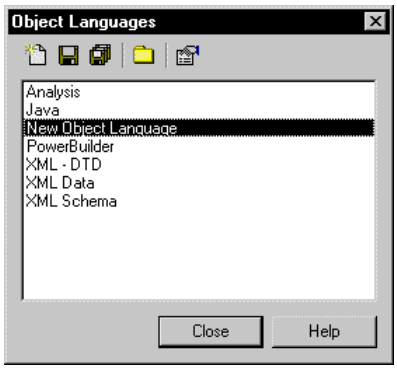
8 Click OK.

A standard Windows Save As box appears.

9 Type the filename and click Save.

The object language is saved in a file with the XOL extension.

You return to the Object Languages window with the new object language selected.



10 Click Close.

☞ For information on how to associate an object language to an OOM, see the section Changing the object language of an OOM.

Using the object language editor

You use the object language editor to consult or modify parameters that appear in categories or sub-categories of an Object-Oriented Model.

Categories

The object language editor is made up of a number of categories, sub-categories, that contain parameters. When you select a category or a parameter, its comment and values are displayed in fields in the right-hand side of the of the dialog box. You define object language editor parameters by modifying the values contained in these fields.

Each category and sub-category in the object language editor has the following properties:

Property	Description
Name	Name of category or sub-category
Comment	Description of selected category or sub-category

Fields

Each field in the object language editor has the following properties:

Property	Description
Name	Field name
Comment	Description of selected field
Value	Field value

Values

The values that you define as parameters are used to define object stereotypes, data types, scripts, and constants.

You can add new values, modify or delete existing ones. Any changes you make to parameters in the object language editor apply to all new objects that you create in existing or new models.

Modifying values in the object language editor

You have to use the object language editor from an Object-Oriented Model. The modifications that you make to values in the object language editor will apply to the current model, as well as to all new Object-Oriented Models.

Object language editor edit menu

When you right click a category or a field in the object language editor, the following editing options appear:

Edit option	Description
Add items…	Allows you to add a renamed copy of a selected field, to the list of fields in a category. When you select Add Items, a selection window appears. It contains a list of fields for the category. To add a field to the category, click the fieldname to select it, and click OK. The new field is added at the bottom of the listed fields for that category
Remove	Deletes the selected category or field
Restore Comment	Restores the default comment for a selected category or field which has been modified
Restore value	Restores the default value for a selected field which has been modified

Object language editor categories

The values you define as parameters fall into three categories:

Category	Description
General	Object language identification
UML	Object stereotypes as defined in UML
Script	Generation characteristics, command definition, and data type translations
Extended Attributes	Extended attributes for the OOM objects that will be used in the generation process

General category

The values that you define in the General category are used when you generate from an Object-Oriented Model. For example, when you generate a Java script file, the values you define in this category appear at the beginning of the file.

The following parameters are defined by default in an OOM:

Parameter	Description	Default value
Product	Name of the model	PowerDesigner
Version	Version of the model	7.0
Family	Default language for current model	Java

UML category

In the UML category, you can define the stereotypes of all objects that can have stereotypes. You can modify existing default stereotypes, or define new stereotypes for any object in the model.

When you modify the values of a stereotype for an object in an OOM, the changes apply to all existing objects and all new objects, of the same type, that you create in the model.

Stereotypes

You can create new stereotypes for all objects in an OOM, or you can modify the values of existing ones.

Default stereotypes

The following objects have existing default stereotypes that you can modify:

- Class
- Operation
- Generalization
- Dependency
- Realization
- Package

Class stereotypes

A class has the following default stereotypes:

Stereotype	Description
actor	Coherent set of roles
enumeration	List of named values used as the range of a particular attribute type
exception	Exception class. Used mostly in relation to error messages
implementationClass	Class whose instances are statically typed, and that defines the physical data structure and methods of a class as implemented in traditional programming languages
process	Heavyweight flow that can execute concurrently with other processes
signal	Specification of an asynchronous stimulus communicated between instances
thread	Lightweight flow that can execute concurrently with other threads within the same process. Usually executes inside the address space of an enclosing process
type	Abstract class used only to specify the structure and behavior of a set of objects, not the implementation
utility	Class that has no instances

Operation stereotypes

An operation has the following default stereotype:

Stereotype	Description
constructor	Operation that creates and initializes an instance of a class

Generalization stereotypes

A generalization has the following default stereotype:

Stereotype	Description
implementation	Specifies that the child object inherits the implementation of the parent object but that it does not make public its interfaces, nor support them, thus violating its substitutability

Dependency stereotypes

A dependency has the following default stereotypes:

Stereotype	Description
access	Public contents of the target package that can by accessed by the source package
bind	Source object that instantiates the target template using the given actual parameters
call	Source operation that invokes the target operation
derive	Source object that can be computed from the target
friend	Source object that has special visibility towards the target
import	Specifies that everything that is declared as public in the target object becomes visible to the source object, as if it were part of the source object definition
include	Use case incorporates the behavior of another use case at a location that is specified by the source
instantiate	Operations on the source class create instances of the target class
refine	Degree of abstraction of the source object is finer than that of the target object
trace	Specifies that there is an historical link between the source object and the target object
use	Semantics of the source object are dependent on the semantics of the public part of the target object

Package stereotypes

A package has the following default stereotypes:

Stereotype	Description
Facade	Package that is a view of another package
Framework	Package that consists mostly of patterns
Model	Specifies a semantically closed abstraction of a system
Stub	Package that serves as a proxy for the public contents of another package
Subsystem	Grouping of elements, some of which constitute a specification of the behavior offered by the other contained elements
System	Package that represents the entire system being modeled

Script category

The Script category contains parameters that influence what will be included in the script files that you generate from an OOM.

The Script category contains the following sub-categories:

Sub-category	Description
Constants	Constant values
Data Types	Basic data type values
GenScripts	Constructor value
Namings	Getter and Setter operation default name values
Events	Standard event values

Default constants

The following constant values are defined by default:

Constant	Default value
Null	0
True	TRUE
False	FALSE
Void	void
Bool	boolean

Object scripts

Depending on the object language family (Java, XML, …) some pieces of generated code can be parameterized in this section.

Each object concerned by the generation process has a sub-category where its definition and other topics can be defined.

The following table lists all the objects that can be customized, with an example for each object specification.

Object	Specification	Example
Class	Definition: generated code for a class	<!ELEMENT %CSFRNAME% EMPTY> <!ATTRLIST %CSFRNAME %ATTRDEFINITIONS% >

Interface	Definition: generated code for an interface	Public interface %CSFRNAME% {
		%ATTRDEFINITIONS%
		%OPERDECLARATIONS%
		}

Attribute	Definition: generated code for an attribute	%ATTRNAME% (CDATA)
	Reference: generated code for a referenced attribute (migrated by a navigable association)	%ATTRNAME% %IDREF%
	ListItem: definition inside a list of attributes	%ATTRNAME%

Operation	Definition: generated code for an operation	%OPERDTTP% %OPERNAME% {
		%OPERBODY%
		}

	Declaration: declaration or prototype of the operation	%OPERDTTP% %OPERNAME%;
Parameter	Definition: generated code for a parameter	%PARMDTTP% %PARMNAME%

XML Mapping

The reverse engineering of XML Data documents needs a mapping table to identify which element or attribute becomes a class, attribute or association in the reversed OOM.

The XML Mapping sub-category, defined under Object scripts, contains three maps: ClassMapping, AttributeMapping and AssociationMapping.

In each map, the 'ID' item specifies the name of the element that will become an object. The name item specifies the attribute or sub-element that will be used as a name for the reversed object. For each kind of object other items may be specified.

Default data types

You can modify the following basic data types:

Data type	Default value
char	TXT
boolean	BL
byte	BT
short	SI
int	I
long	LI
float	F
double	N
*	TXT

Namings

You can modify the following default names for Getter and Setter operations:

Getter operation

Name	Default value
*	set%Code%

Setter operation

Name	Default value
boolean	is%Code%
*	get%Code%

Event

You can use this sub-category to define events on operations. The default exisiting events are constructor and destructor.

Extended Attributes category

The Extended Attributes category allows he user to define extended attributes for the OOM objects.

The extended attributes can be used in the generation process. Each extended attribute becomes a variable that can be referenced in the scripts defined in the Script category.

CHAPTER 9

Using Business Rules

About this chapter

This chapter describes how business rules help you model information.

Contents

What is a business rule?

A business rule is a written statement specifying what the information system must do or how it must be structured to support business needs.

A business rule is a rule that your business follows. A business rule could be a government-imposed law, a customer requirement, or an internal guideline.

Starts as an observation

Business rules often start as simple observations, for example "customers call toll-free numbers to place orders." During the design process they develop into more detailed expressions, for example what information a customer supplies when placing an order or how much a customer can spend based on a credit limit.

Guides modeling

Business rules guide and document the creation of a model. For example, the rule "an employee belongs to only one division" can help you graphically build the link between an employee and a division.

Complements graphics

Business rules complement model graphics with information that is not easily represented graphically. For example, some rules specify physical concerns in the form of formulas and validation rules. These technical expressions do not have a graphical representation.

Check parameters

You can attach business rules to objects in an OOM. You can generate business validation rules as check parameters if the validation rules are attached to domains.

For more information on defining and using check parameters, see the chapter Building an Object-Oriented Model.

Defining business rules in an OOM

You can define a business rule which can be attached to the following objects in an OOM:

Domains
Classes
Interfaces
Attributes
Identifiers
Operations
Associations
Generalizations
Realizations
Dependencies

Types of business rule

In PowerDesigner, you can define several different types of business rules.

Rule type	Describes	Example
Definition	Characteristics or properties of an object in the information system	A customer is a person identified by a name and an address
Fact	Certainty or existence in the information system	A client may place one or more orders
Formula	Calculation employed in the information system	The total order is the sum of all the order line costs
Validation	Constraint on a value in the information system	The sum of the order totals for a given client must not be greater than that client's allowance

Business rule properties

A business rule definition includes the following properties:

Property	Description	Maximum length
Name	Name for the rule	254
Code	Reference name for the rule	254
Comment	Descriptive label for the rule	—
Type	Indicates whether the rule is a definition, a fact, a formula, or a validation	—
Expression	Presence of associated expression	—
Notes	Presence of associated notes	—

Creating a business rule

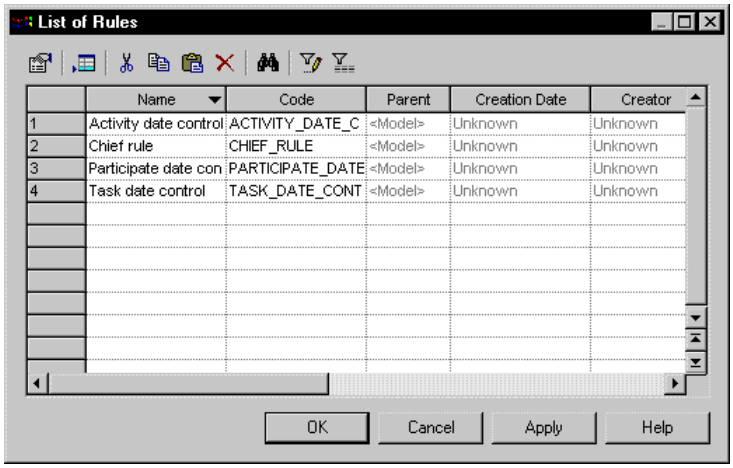
Before you create business rules, formulate your rules by asking yourself the following questions:

- What business problems do I want to address?
- Are there any procedures that my system must respect?
- Do any specifications dictate the scope of my project?
- Do any constraints limit my options?
- How do I describe each of these procedures, specifications, and constraints?
- How do I classify these descriptions: as definitions, facts, formulas, or validation rules?

❖ **To create a business rule:**

1 Select Model➤Business Rules.

The List of Rules appears. It displays the business rules defined for the model.



2 Click a blank line in the list.
or
Click the Add a Row tool.

An arrow appears at the beginning of the line.

3 Type a name and a code for the business rule.

4 Click Apply.

The creation of the new business rule is committed.

5 Click the new business rule line.

An arrow appears at the beginning of the line.

6 Click the Properties tool.
or
Double click the arrow at the beginning of the line.

The property sheet for the new business rule appears.

7 Select a business rule type from the Type dropdown listbox

8 Click OK.

Applying business rules to objects

From the list of business rules, you can apply a business rule to existing objects. You can also apply a business rule to objects from their property sheets or lists.

Applying a business rule to an object

You can add business rules that already exist in the model, and which belong to other objects.

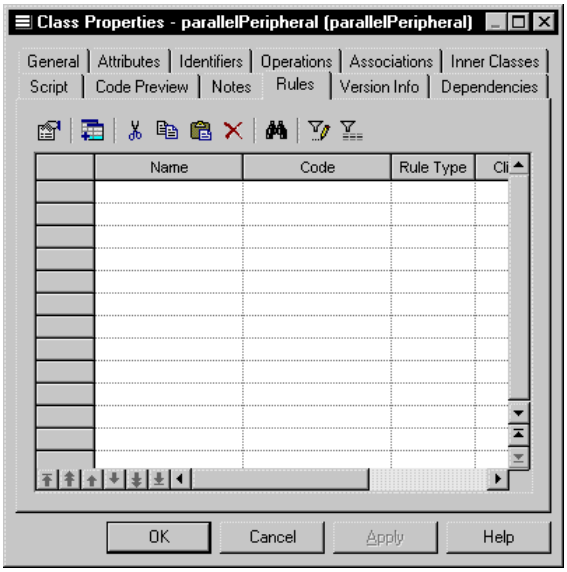
❖ **To apply a business rule to an object:**

1 Double-click an object in the model.

The object property sheet appears.

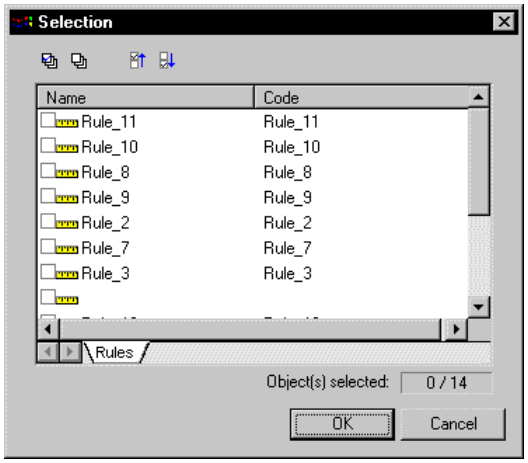
2 Click the Rules tab.

The Rules page appears.



3 Click the Add Rules tool.

The Selection window appears. It contains a list of all the business rules that exist in the model, with the exception of those that already belong to the object.



4 Select the business rules that you want to add to the object.

5 Click OK.

The business rules are added to the object and appear in the list of business rules for the object.

6 Click OK.

Attaching an expression to a business rule

A business rule typically starts out as a description. As you develop your model and analyze your business problem, you can complete a rule by adding a technical expression.

Each business rule can include two types of expression:

- Server
- Client

Expressions are used essentially in a CDM or a PDM.

For more information on expressions, see the chapter Using Business Rules in the PowerDesigner PDM User's Guide.

Glossary

abstract class	A class that cannot have any direct instances
aggregation	An form of association that specifies a part-whole relationship between a component class and an aggregate class
association	A structural relationship that describes a set of links between objects
association role	The endpoint of an association, a role specifies the multiplicity and visibility between the association and the class to which it is connected
attribute	A named property of an object that defines the characteristics of the object
Beaninfo class	Reusable software component that can be visually manipulated in a software development tool
business rule	A written statement specifying what the information system must do or how it must be structured to support business needs
cardinality	The number of elements in a set. The number has to be specific and cannot be a range, as is the case with multiplicity
class	A description of a set of objects that share the same attributes, operations, relationships, and semantics

class diagram	A class diagram is a view of a model that shows a set of packages, classes, interfaces, and their relationships that together represent the logical static design view of a system. A class diagram may contain all or part of the class structure of a system
classifier	A classifier is a mechanism that has structural (attributes) and behavioral (operations) features. All objects that can have instances are classifiers
composition	A form of aggregation of an association in which the class is attached to the association role is may be a part of only one composite at a time
constructor	An operation that creates and initializes an instance of a class
data type	A type whose values have no identity. Data types include primitive types and enumeration types
dependency	A semantic relationship between two modeling elements, in which a change to one modeling element (the independent element) may affect the semantics of the other modeling element (the dependent element)
domain	Set of values for which a data item is valid
generalization	A relationship between a more general element (the parent) and a more specific element (the child). The more specific element is fully consistent with the more general element and contains additional information
identifier	An identifier is a class attribute, or a combination of class attributes, whose values uniquely identify each occurrence of the class
inner class	A class definition within another class definition
interface	A collection of operations used to specify the externally visible behavior of a class, object, or other entity. In the case of a class or object, the interface includes the signatures of the operations

multiplicity	A specification of the range of allowable cardinalities that a set may assume
Object-oriented model (OOM)	Class structure that is the logical design view of a software system. An OOM is essentially a static conceptual model of a software system
object language	Contains the specifications for a particular language. It provides PowerDesigner with the syntax and guidelines for implementing stereotypes, data types, scripts and constants for a object language
operation	The implementation of a service that can be requested from an object in order to affect behavior. An operation has a signature, a name, and a list of parameters
package	A general purpose mechanism for organizing elements into groups
parameter	Specification of a variable that can be changed, passed, or returned. Parameters are used only for operations
persistence	Lifetime of the instances of a class
persistent object	An object that continues to exist after the process that created it has ceased to exist
realization	A semantic relationship between classifiers, in which one classifier specifies a contract that another classifier guarantees to carry out
return type	A list of values returned by a call of the operation
role	The named specific behavior of an object participating in a particular context
signature	The name and parameters of an operation

stereotype	An extension of the vocabulary of the UML, which allows you to create new kinds of building blocks that are derived from existing ones but that are specific to your problem
transient object	An object that ceases to exist when the process that created it ceases to exist
visibility	Denotes how an object can be seen and used by other objects

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z