Class Diagrams

A class diagram is a UML diagram that models the static structure of a system, which contain classifiers and their relationships to each other. A class diagram can depict some or all of the classes in a model. Conversely, a single model can support many class diagrams and a single classifier can appear in more than one diagram.

Class diagrams typically contain classes that represent the abstraction of a concept. They also contain other classifiers such as interfaces, signals, and enumerations. Class diagrams also use multiple variations of association relationships to indicate which classifiers need to share data with other classifiers. You can also use other relationships on class diagrams such as generalizations, realizations, and dependencies.

Class diagrams are often the foundation for component and deployment diagrams, which contain components and nodes, respectively, instead of classes.

Using Class Diagrams

Class diagrams are helpful because they are very effective for visualizing, specifying, and documenting structural features in models. For example, during analysis and design you can create class diagrams to:

- Capture the structure of the classifiers that form the system's architecture.
- Model the structural aspects of a model using attributes, operations, and signals.
- Show the common classifier roles and responsibilities that define the system's behavior.
- Show the implementation classes in a package, the structure and behavior of one or more classes, or a view of inheritance hierarchy.
- Show the workers and entities in a business object model.

You can also use class diagrams during the implementation phase of the software development lifecycle to forward and reverse engineer your models and source code.

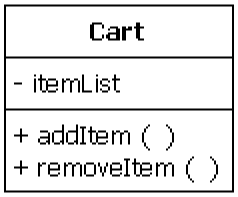
Classes

A class is a model element that represents an abstraction of a concept or thing. It identifies the attributes, operations, relationships, and semantics that objects created from it will have. While the class identifies these structural and behavioral features, each object created from it typically provides its own values for the attributes.

For example, an e-commerce application may include a "Cart" class. The class defines the attributes, such as "itemList," and operations, such as "addItem( )," that all objects created from it will have. At run time, multiple objects may be created from the "Cart" class, each using the attributes and operations that the class defines. However, each object typically has different values for its attributes. For instance, a "cart101" object may call the "addItem( )" operation to add videos to its "itemList" attribute; in contrast, a "cart202" object may call the "addItem( ) " operation to add books to its "itemList" attribute.

Shape

A class usually appears as a rectangle with three compartments. Its upper compartment contains the class name. Its middle compartment contains the attributes. Its lower compartment contains the operations.



You can show or hide the attribute and operation compartments as well as a signal reception compartment using the Show/Hide Compartment button on the Appearance Toolbar.

Features of Classes

If desired, you can add the following to classes.

- Attributes – Represent properties that objects created from the class define. Attributes are usually implemented as variables, but they can be implemented using other techniques. In various languages, attributes are also known as "member variables," "properties," and "variables."
- Operations – Represent behaviors that objects created from the class provide. In various languages, operations are also known as "functions," "member functions," "methods," and "procedures."
- Signals – Identify the signals that objects created from the class can receive. The reception of a signal is known as an "event."

Using Classes

You can add classes to your model to represent the following:

- In models depicting businesses, classes represent the workers and artifacts of the business.
- In models depicting software systems, classes represent the classes used to develop the system or their equivalents. Examples include Java classes for Java applications, database tables for databases, and Web pages for Web applications. When classes represent entities such as database tables or Web pages, they are typically assigned a stereotype.

The classes in a model usually appear in class diagrams. Classes or instances of classes (objects or classifier roles) also appear frequently in other types of diagrams, including activity, component, and sequence diagrams.

Naming Conventions

A class has a name that reflects the role it plays in the system. Usually, the name is taken from the vocabulary of the system for which the application is being created.

Adding and Modifying Model Elements in Class Diagrams

To illustrate the static structure of a model, you can create one or more class diagrams. You can add and modify shapes in class diagrams to represent classes, and to show their internal structure (attributes, operations, and literals) and their relationships to other classes.

To	See
Specify visibility of a model element	Specifying Visibility
Indicate that a classifier cannot be directly instantiated	Specifying Abstract for Classifiers or Operations
Make a class or object active	Making Classes and Objects Active
Indicate whether an attribute or association is computed from one or more other model elements	Specifying Derived for Attributes and Associations
Assign a default value when an instance of the classifier is created	Specifying Default Values for Association Ends, Attributes, and Parameters
Indicate that a classifier or operation should not have child classifiers when it participates in a generalization relationship	Specifying Leaf for Classifiers or Operations
Specify multiplicity for classifiers	Specifying Multiplicity for Classifiers
Indicate whether association ends or attributes (with multiplicity greater than one) are set in a specific order	Specifying Ordering for Association Ends and Attributes
Indicate whether an association end, attribute, or operation appears in each instance of the classifier or only once for the classifier	Specifying Owner Scope
Indicate whether an association, attribute, or classifier is permanent	Specifying Persistence
Indicate that a classifier is a root and that it should not have any parent classifiers when it participates in a generalization relationship	Specifying Root for Classifiers or Operations
Indicate a type for an association end, attribute, or parameter	Specifying Types for Association Ends, Attributes, and Parameters

Specifying Visibility

You can specify visibility for a feature (for example, attribute or operation) contained in a classifier, for a classifier contained in a package, or for an association end contained in an association relationship that connects two classifiers. The visibility indicates what other model elements can see and use the model element to which the visibility refers. For details about visibility, including the icons and text that represent it, see Visibility.

These are the levels of visibility:

- **PRIVATE** – Only model elements within the same container can see and use the private model element.
- **PROTECTED** – Only model elements within the same container or a descendent of the container can see and use the protected model element.
- **PUBLIC** – Any model elements that can see the container can also see and use the public model element.
- **PACKAGE** – Any model element that is in the same package as the container can see and use the model element with package visibility.

By default, the **Visibility** property for classifiers and their features is set to **PUBLIC**, and the **Visibility** property for association ends is set to **PRIVATE**.

To specify visibility for a classifier or classifier feature

1. In the Model Explorer, select the classifier or the classifier feature for which you want to specify visibility.
2. In the Properties window, expand **UML**, click **Visibility**, and then select the desired value.

The value of the property is updated. If the visibility is for a classifier, the appearance of the classifier in the Model Explorer and diagram window does not change. If the visibility is for an attribute or operation, its icon in the Model Explorer and its icon or text symbol in the diagram window are updated to reflect the specified value.

To specify visibility for an association end

1. In the diagram window or Model Explorer, select the association for which you want to specify visibility.
2. In the Properties window, expand **End1** or **End 2**, click **End1Visibility** or **End2Visibility**, and then select the desired value.

The value of the property is updated. If the association end has a name and the association appears in a diagram, the text symbol for the specified visibility level prefixes the name of the association end.

Specifying Abstract for Classifiers or Operations

When you specify that a classifier (for example, class, interface, or use case) is abstract, you indicate that it cannot be directly instantiated. For example, you can specify that a class is abstract if it has one or more abstract operations or is designed only to be a parent classifier for deriving child classifiers. As another example, you specify that a use case is abstract when the use case is an inclusion use case, meaning it is incorporated in other use cases but not used directly by an actor.

You can also specify that an operation is abstract to indicate that it does not have an implementation. An abstract operation indicates that child classes that inherit it must provide its implementation. When you have an abstract operation, you should make the class that contains it abstract.

By default, the **IsAbstract** property is set to **False**.

To specify that a classifier is abstract

1. In the Model Explorer or diagram window, select the classifier or the shape representing it.
2. In the Properties window, expand **UML**, and click **IsAbstract > True**.

The value of the property is updated. When a shape representing the abstract classifier appears in a diagram, its name is italicized. Names of concrete classifiers are in regular text.

To specify that an operation is abstract

1. In the Model Explorer, select the operation.
2. In the Properties window, expand **UML**, and click **IsAbstract > True**.

The value of the property is updated. When an abstract operation appears in a diagram, its name is italicized. Names of concrete operations are in regular text.

Making Classes or Objects Active

You can make a class or object active to indicate that it owns a thread of control and may initiate control activities. Processes and tasks are two kinds of active objects.

To make a class or object active

1. In the Model Explorer or diagram window, select the class or object.
2. In the Properties window, expand **UML**, and click **IsActive > True**.

The border of the active object or class in a diagram changes from a thin border to a thick border. In sequence diagrams, the lifeline of an active object changes from a dashed line to a thin rectangle.

References:

IBM Rational Software – Rational XDE Resources. Copyright © 2002-2003