#112

Get More Refcardz! Visit refcardz.com

www.dzone.com

DZone Refcardz

CONTENTS INCLUDE:

Getting Started with UML

By James Sugrue

ABOUT UML

The Unified Modeling Language is a set of rules and notations for the specification of a software system, managed and created by the Object Management Group. The notation provides a set of graphical elements to model the parts of the system.

This Refcard outlines the key elements of UML to provide you with a useful desktop reference when designing software.

Hot Tip

UML Tools

There are a number of UML tools available, both commercial and open source, to help you document your designs. Standalone tools, plug-ins and UML editors are available for most IDEs.

Diagram Types

UML 2 is composed of 13 different types of diagrams as defined by the specification in the following taxonomy.

STRUCTURAL DIAGRAMS

Class Diagrams

Class diagrams describe the static structure of the classes in your system and illustrate attributes, operations and relationships between the classes.

Modeling Classes

The representation of a class has three compartments.

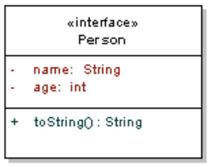


Figure 1: Class representation

From top to bottom this includes:

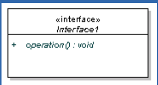
- **Name** which contains the class name as well as the stereotype, which provides information about this class. Examples of stereotypes include `<<interface>>`, `<<abstract>>` or `<<controller>>`.
- **Attributes** lists the class attributes in the format `name:type`, with the possibility to provide initial values using the format `name:type=value`
- **Operations** lists the methods for the class in the format `method( parameters):return type`.

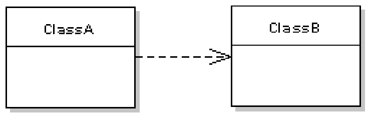
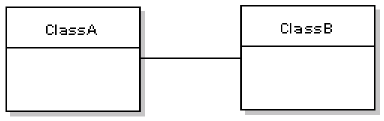
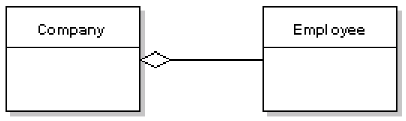
Operations and attributes can have their visibility annotated as follows: + public, # protected, - private, ~ package

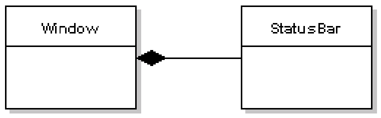
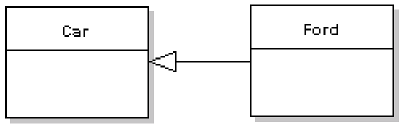
Hot Tip

Interfaces

Interface names and operations are usually represented in italics.



Relationship	Description
Dependency "...uses a..."	A weak, usually transient, relationship that illustrates that a class uses another class at some point.  **Figure 2:** ClassA has dependency on ClassB
Association "...has a..."	Stronger than dependency, the solid line relationship indicates that the class retains a reference to another class over time.  **Figure 3:** ClassA associated with ClassB
Aggregation "...owns a..."	More specific than association, this indicates that a class is a container or collection of other classes. The contained classes do not have a life cycle dependency on the container, so when the container is destroyed, the contents are not. This is depicted using a hollow diamond.  **Figure 4:** Company contains Employees

Composition "…is part of…"	More specific than aggregation, this indicates a strong life cycle dependency between classes, so when the container is destroyed, so are the contents. This is depicted using a filled diamond.  **Figure 5:** StatusBar is part of a Window
Generalization "…is a…"	Also known as inheritance, this indicates that the subtype is a more specific type of the super type. This is depicted using a hollow triangle at the general side of the relationship.  **Figure 6:** Ford is a more specific type of Car

Association Classes

Sometimes more complex relationships exist between classes, where a third class contains the association information.

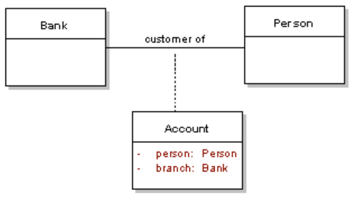


Figure 7: Account associates the Bank with a Person

Annotating relationships

For all the above relationships, direction and multiplicity can be expressed, as well as an annotation for the relationship. Direction is expressed using arrows, which may be bi-directional.

The following example shows a multiple association, between ClassA and ClassB, with an alias given to the link.

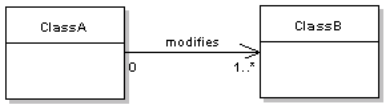


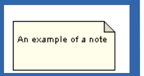
Figure 8: Annotating class relationships

Relationships can also be annotated with constraints to illustrate rules, using {} (e.g. {ordered}).

Hot Tip

Notes

Notes or comments are used across all UML diagrams. They used to hold useful information for the diagram, such as explanations or code samples, and can be linked to entities in the diagram.



Object Diagrams

Object diagrams provide information about the relationships between instances of classes at a particular point in time. As you would expect, this diagram uses some elements from class diagrams.

Typically, an object instance is modeled using a simple rectangle without compartments, and with underlined text of the format InstanceName:Class

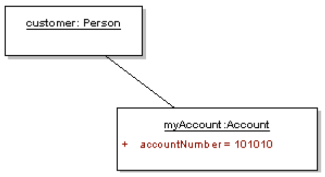


Figure 9: A simple object diagram

The object element may also have extra information to model the state of the attributes at a particular time, as in the case of myAccount in the above example.