**Software Modeling**

By: Ms. Enas Naffar

1/ 11/ 2016

Software Modeling

OO Modeling Is A Conceptual Process Independent Of Programming Language. It Is A Way Of Thinking, Not Programming.

**What is a MODEL?**

A model is an abstract, simplified or incomplete description/ representation of a part of the real word.

**UML**

The Unified Modeling Language (UML) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. See http://www.uml.org/

UML 2.0 defines 13 types of diagrams, divided into three categories:

* **Structure diagrams** (class, object, component, composite structure, package, deployment).
* **Behavior diagrams** (use case, activity, state machine).
* **Interaction diagrams** (sequence, communication, timing, interaction overview).

**CLASS DIAGRAMS**

An **Object** isan entity with a well-defined role in an application.

Each object has:

* ***State***: defined by using the attributes, their values, and associations with other objects
* ***Behavior***: represents how an object acts and reacts
* ***Identity***: uniqueness, no two objects are the same

A **Class** isa logical grouping of objects with similar attributes and behavior. By grouping objects into classes, we abstract the problem.

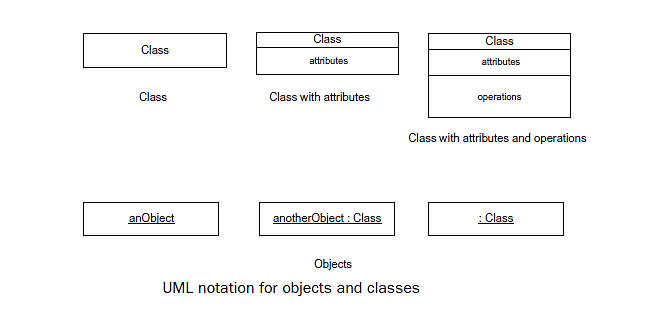
An **Operation** is a function or service provided by all instances of a class.

**Encapsulation**: the technique of hiding internal implementation details of an object from external view.

In a **class diagram**, classes and objects are depicted as boxes, and the static (i.e., permanent) relationships between them are depicted as lines connecting the boxes.

The following main types of relationships between classes are supported: ***associations***, ***whole/part*** relationships, and ***generalization/specialization*** relationships.

In UML, the class box always holds the class name. Optionally, the attributes and operations of a class may also be depicted. When all three are depicted, the top compartment of the box holds the class name, the middle compartment holds the attributes, and the bottom compartment holds the operations.



**Associations**

An **association** is a static, structural relationship between two or more classes. An association between two classes, which is referred to as a *binary association*, is depicted as a line joining the two class boxes.

An association has a ***name*** and optionally a small black arrowhead to depict the ***direction*** in which the association name should be read.

On each end of the association line joining the classes is the ***multiplicity*** of the association, which indicates ***how many instances of one class are related to an instance of the other class***. The multiplicity of an association can be exactly one (1), optional (0..1), zero or more (∗), one or more (1..∗), or numerically specified (m..n), where *m* and *n* have numeric values.

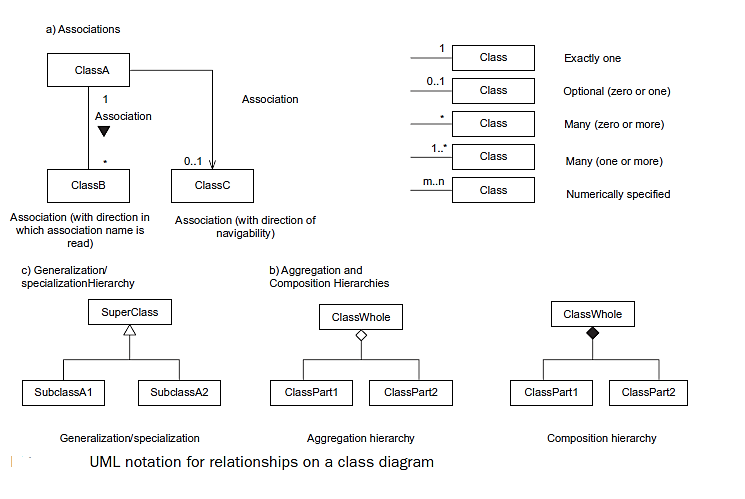
**Aggregation and Composition Hierarchies**

Aggregation and composition hierarchies are **whole/part** relationships. The **composition** relationship (shown by a black diamond) is a *stronger* form of whole/part relationship than the **aggregation** relationship (shown by a hollow diamond).

The diamond touches the aggregate or composite (Class Whole) class box.

**Generalization/Specialization Hierarchy**

A generalization/specialization hierarchy is an **inheritance** relationship. A generalization is depicted as an arrow joining the subclass (child) to the superclass (parent), with the arrowhead touching the superclass box

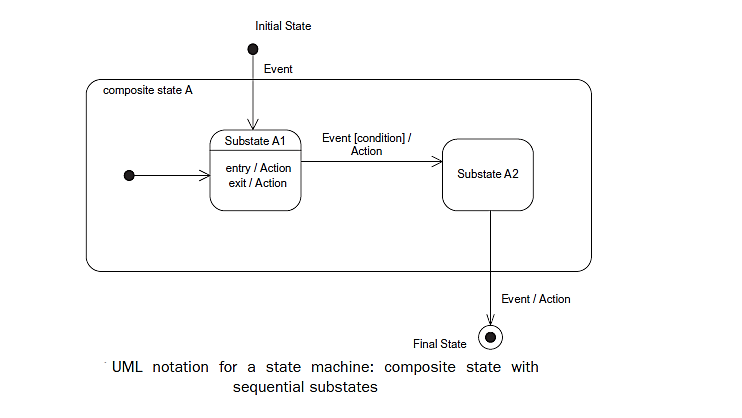


**STATE MACHINE DIAGRAMS**

In the UML notation, a state transition diagram is referred to as a **state machine diagram** (sometimes **statechart** is used). *S****tates*** are represented by rounded boxes, and ***transitions*** are representedby arcs that connect the rounded boxes. The initial state ofthe statechart is depicted by an arc originating from a small black circle. Optionally,a final state may be depicted by a small black circle inside a larger white circle,sometimes referred to as a *bull’s-eye*.

A statechart may be hierarchically decomposedsuch that a composite state is broken down into substates.

On the arc representing the state transition, the notation ***Event [condition]/ Action*** is used. The ***event*** causes the state transition.

****The optional Boolean ***condition*** (guard)must be true, when the event occurs, for the transition to take place. The optional ***action*** is performed as a result of the transition.

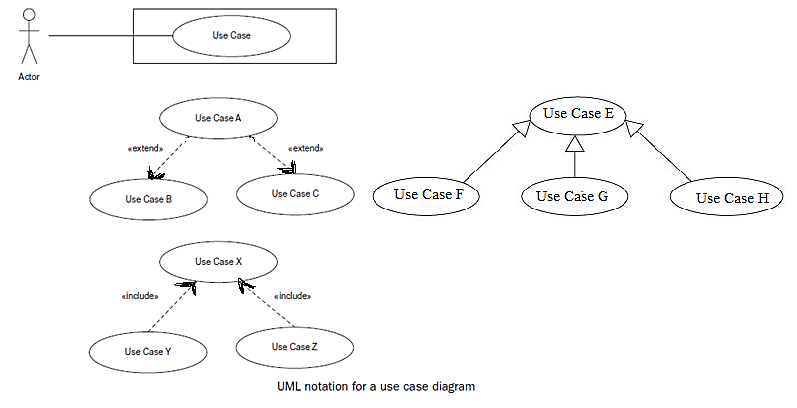
Optionally, a state may have any of the following:

■ An **entry action**, performed when the state is entered.

■ An **exit action**, performed on exit from the state.

**USE CASE DIAGRAMS**

An ***actor*** initiates a use case. A ***use case*** defines a sequence of interactions between the actor and the system. An actor is depicted as a stick figure on a use case diagram.

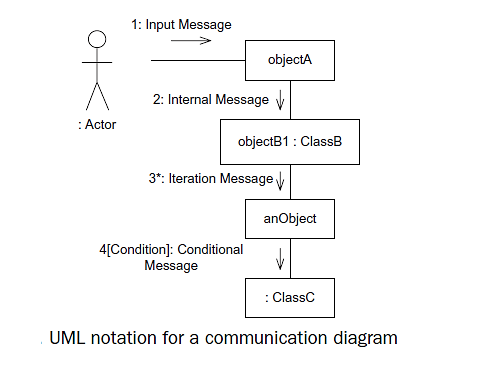


The system is depicted as a box. A use case is depicted as an ellipse inside the box.

Communication associations connect actors with the use cases in which they participate. ***Relationships*** among use cases are defined by means of *include, extend* and *generalization* relationships.

**COMMUNICATION DIAGRAMS**

A **communication diagram** is an interaction diagram, which was called a *collaboration diagram* in UML 1.x, shows how cooperating ***objects*** dynamically interact with each other by sending and receiving ***messages***. The diagram depicts the structural organization of the objects that interact. Objects are shown as boxes, and lines joining boxes represent object interconnection. Labeled arrows adjacent to the arcs indicate the name and direction of message transmission between objects. The ***sequence of messages*** passed between the objects is numbered. An optional iteration is indicated by an asterisk (∗), which means that a message is sent more than once. An optional condition means that the message is sent only if the condition is true.



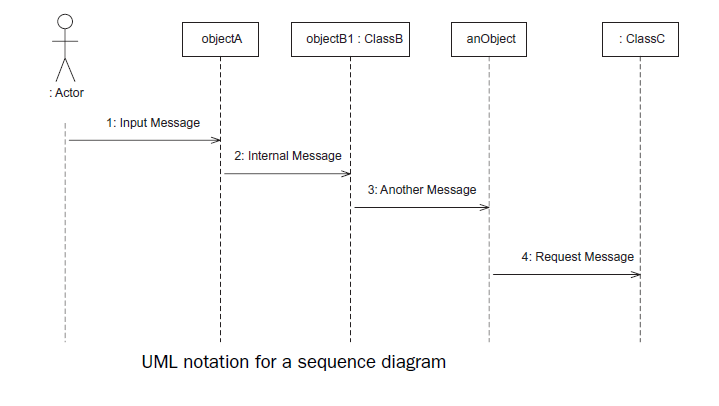
**SEQUENCE DIAGRAMS**

A **sequence diagram** is an interaction diagram, which depicts object interaction arranged in time sequence, as shown. A **sequence diagram** is a two-dimensional diagram in which the objects participating in the interaction are depicted horizontally and the vertical dimension represents time.

Starting at each ***object*** box is a vertical dashed line, referred to as a ***lifeline***. Optionally, each lifeline has an activation bar (not shown), depicted as a double solid line, which shows when the object is executing.

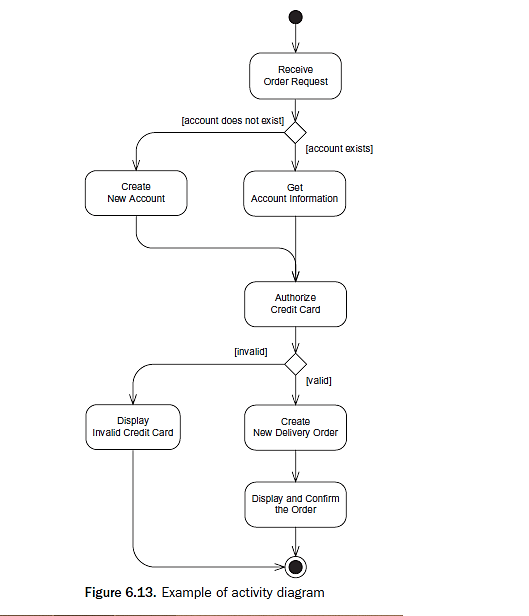
The ***actor*** is usually shown at the extreme left of the page. Labeled horizontal arrows represent ***messages***. Only the source and destination of the arrow are relevant.

The message is sent from the source object to the destination object. Time increases from the top of the page to the bottom. UML 2 has extended the notation for sequence diagrams to allow for **loops** and **conditions**.



**ACTIVITY DIAGRAMS**

An **activity diagram** is a UML diagram depicting the flow of control and sequencing among activities. An activity diagram shows the sequence of ***activities***, ***decision nodes***, ***loops***, and even ***concurrent activities***. Activity diagrams are widely used in **workflow** modeling – for example, for service-oriented applications.



**References:**

1. [**http://www.uml.org/**](http://www.uml.org/)
2. **Software modeling and design,** UML, Use Cases, Patterns, andSoftware Architectures**, Hassan Gomaa, 2011**
3. **Modeling Concepts,** Karlstad University. Adapted for a textbook by Blaha M. and Rumbaugh J., Object Oriented Modeling and Design, 2005