BACHELOR OF TECHNOLOGY YEAR 3

OBJECT ORIENTED ANALYSIS AND DESIGN

WEEK 7

UNIFIED MODELING LANGUAGE (UML)

Definition: Unified Modeling Language *(UML) is a* graphical language *for* OOAD *that gives a standard way to write a software system's blueprint.*

It helps to *visualize*, *specify*, construct, and document the artifacts of an object-oriented system.

It is used to *depict* the *structures* and the *relationships* in a complex system.

BRIEF HISTORY

It was developed in 1990s as a combination of several techniques, prominently *OOAD* technique by Grady Booch, *OMT* (Object Modeling Technique) by James Rumbaugh, and *OOSE* (Object Oriented Software Engineering) by Ivar Jacobson. UML attempted to standardize semantic models, syntactic notations, and diagrams of OOAD.

SYSTEMS AND MODELS IN UML

a) *System*: A set of elements organized to achieve certain objectives of a system. Systems are often divided into subsystems and described by a set of models.

b) *Model:* Model is a simplified, complete, and consistent abstraction of a system, created for better understanding of the system.

c) *View:* A view is a projection of a system's model from a specific perspective.

CONCEPTUAL MODEL OF UML

The Conceptual Model of UML encompasses three major elements:

- Basic building blocks
- Rules
- Common mechanisms

A. Basic building blocks

The three building blocks of UML are:-

1. Items

There are four kinds of items in UML, namely:-

a) *Structural Items:* These are the nouns of the UML models representing the static elements that may be either physical or conceptual. The structural things are *class*, *interface*, *collaboration*, *use case*, *active class*, *components*, and *nodes*.

b) *Behavioral Items:* These are the verbs of the UML models representing the *dynamic behavior* over time and space. The two types of behavioral items are interaction and state machine.

c) *Grouping Items:* They comprise the *organizational* parts of the UML models. There is only one kind of grouping item which is the *package*.

d) *Annotational Items:* These are the explanations in the UML models representing the comments applied to describe elements.

2. Relationships

Relationships are the *connections* between things. The four types of relationships that can be represented in UML are:-

a) *Dependency:* This is a semantic relationship between two items such that a change in one thing brings a change in the other. The former is the independent items, while the latter is the dependent items.

b) *Association:* This is a structural relationship that represents a group of links having common structure and common behavior.

c) *Generalization:* This represents a generalization/specialization relationship in which subclasses inherit structure and behavior from super-classes.

d) *Realization:* This is a semantic relationship between two or more classifiers such that one classifier lays down a contract that the other classifiers abide by.

3. Diagrams

A diagram is a *graphical representation* of a system. It comprises of a *group of elements* generally in the form of a *graph*. UML includes nine diagrams in all, namely:-

- Class Diagram
- Object Diagram
- Use Case Diagram
- Sequence Diagram
- Collaboration Diagram
- State Chart Diagram

- Activity Diagram
- Component Diagram
- Deployment Diagram

B. Rules

UML has a number of *rules* so that the models are *semantically self-consistent* and related to other models in the system *harmoniously*. UML has semantic rules for the following:-

- Names
- Scope
- Visibility
- Integrity
- Execution

C. Common Mechanisms

UML has *four* common mechanisms:-

- Specifications
- Adornments
- Common Divisions
- Extensibility Mechanisms

1. Specifications

In UML, behind each *graphical notation*, there is a textual statement denoting the *syntax* and *semantics*. These are the specifications which provide a *semantic package* containing all the parts of a system and the *relationship* among the different *paths*.

2. Adornments

Each element in UML has a unique graphical notation. Besides, there are notations to represent the important aspects of an element like name, scope, visibility, etc.

3. Common Divisions

Object-oriented systems can be divided in many ways but the two common ways of division are:-

a) *Division of classes and objects:* A class is an abstraction of a group of similar objects. An object is the concrete instance that has actual existence in the system.

b) *Division of Interface and Implementation:* An interface defines the rules for interaction. Implementation is the concrete realization of the rules defined in the interface.

4. Extensibility Mechanisms

UML is an open-ended language. It is possible to extend the capabilities of UML in a controlled manner to suit the requirements of a system. The extensibility mechanisms are:

a) *Stereotypes:* It extends the *vocabulary* of the UML, through which new building blocks can be created out of existing ones.

b) *Tagged Values:* It extends the *properties* of UML building blocks.

c) *Constraints:* It extends the *semantics* of UML building blocks.

UNIFIED MODELING LANGUAGE (UML) NOTATIONS

UML defines specific notations for each of the building blocks.

1. Class

A class is represented by a rectangle having three sections:-

- The top section containing the name of the class
- The middle section containing class attributes
- The bottom section representing operations/methods of the class

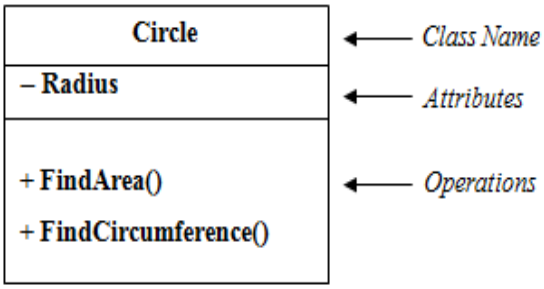
The visibility of the attributes and operations can be represented in the following ways:-

a) *Public:* A public member is visible from anywhere in the system. In class diagram, it is prefixed by the symbol '+'.

b) *Private:* A private member is visible only from within the class. It cannot be accessed from outside the class. In class diagram, a private member is prefixed by the symbol '–'.

c) *Protected:* A protected member is visible from within the class and from the subclasses inherited from this class, but not from outside. It is prefixed by the symbol '#'.

Example: Let us consider the Circle class introduced earlier. One attribute of a Circle is radius. The operations are findArea(), findCircumference(). The radius is a private data member. The following figure below gives the diagrammatic representation of the class.



2. *Object*

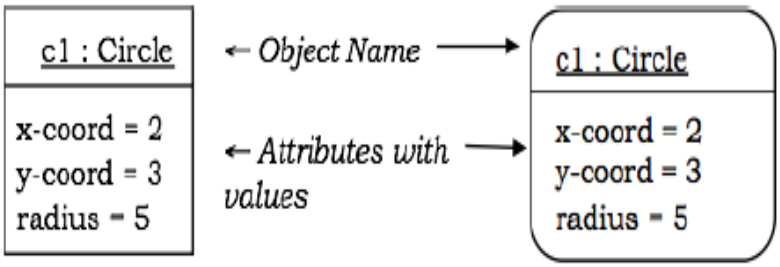
An object is represented as a rectangle with two sections:-

The top section contains the name of the object with the name of the class or package of which it is an instance of. The name takes the following forms: -

- *object-name:* class-name
- *object-name:* class-name :: package-name

The bottom section represents the attributes and the values. It takes the form attribute-name = value. Sometimes objects are represented using rounded rectangles.

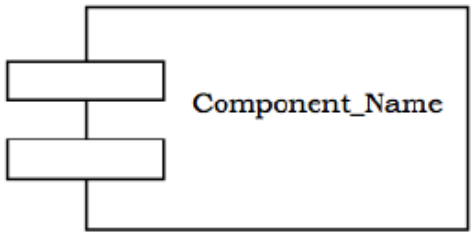
Example: Let us consider an object of the class Circle named c1. We assume that the center of c1 is at (2, 3) and the radius of c1 is 5. The figure below illustrates the object diagrammatically.



3. *Component*

A component is a *physical* and *replaceable* part of the system that conforms to and provides the *realization of a set of interfaces*. It represents the physical packaging of elements like classes and interfaces.

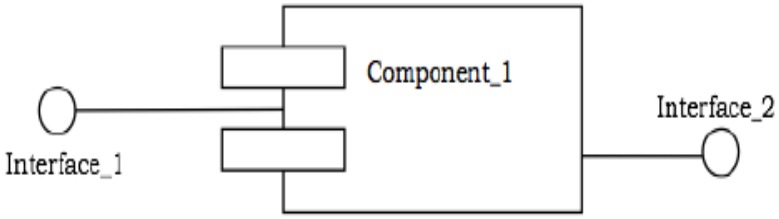
Notation: In UML diagrams, a component is represented by a rectangle with tabs as shown in the figure below.



4. *Interface*

Interface is a *collection* of *methods* of a *class* or *component*. It specifies the *set of services* that may be provided by the class or component.

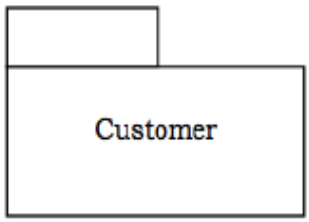
Notation: Generally, an interface is drawn as a circle together with its name. An interface is almost always attached to the class or component that realizes it. The following figure gives the notation of an interface.



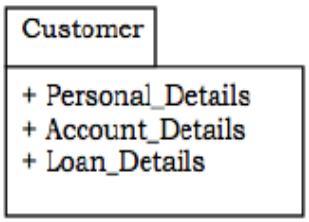
5. *Package*

A package is an organized group of elements. A package may contain structural things like classes, components, and other packages in it.

Notation: Graphically, a package is represented by a tabbed folder. A package is generally drawn with only its name. However it may have additional details about the contents of the package as shown below



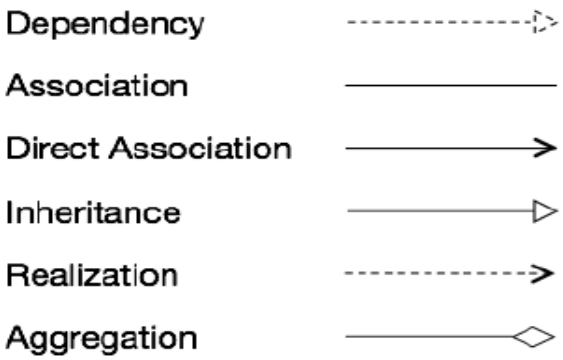
(a)



(b)

6. Relationship

The notations for the different types of relationships are as follows:-



Usually, *elements* in a relationship play *specific roles* in the relationship. A *role name* signifies the *behavior* of an element participating in a certain context.

Example: The following figures show examples of different relationships between classes. The first figure shows an association between two classes, Department and Employee, wherein a department may have a number of employees working in it. *Worker* is the *role name*. The '**1**' alongside Department and '*****' alongside Employee depict that the cardinality ratio is one–to–many. The second figure portrays the aggregation relationship; a University is the umbrella with many Departments.