UNIT 1 INTRODUCTION TO OBJECT ORIENTED MODELING

Structure **Page Nos.**

1.0 Introduction 7
1.1 Objectives 7
1.2 Object Oriented Modeling 8
1.3 Basic Philosophy of Object Orientation 10
1.4 Characteristics Object Oriented Modeling 11
1.4.1 Class and Objects
1.4.2 Links and Association
1.4.3 Generalization and Inheritance
1.5 An Object Model 16
1.6 Benefits of OO Modeling 17
1.7 Introduction to OOA& Design Tools 17
1.8 Summary 19
1.9 Solutions/Answers 19

1.0 INTRODUCTION

Object oriented design methods emerged in the 1980s, and object oriented analysis methods emerged during the 1990s. In the early stage, object orientation was largely associated with the development of graphical user interfaces (GUIs), and a few other applications became widely known. In the 1980s, Grady Booch published a paper on how to design for Ada, and gave it the title, *Object Oriented Design*. In 1991, Booch was able to extend his ideas to a genuinely object oriented design method in his book with the same title, revised in 1993 (Booch, 1994) [*sic*].

The Object Modeling Technique (OMT) covers aspects of object oriented analysis and design. OOT provides a very productive and practical way of Software development. As object oriented Technology (OOT) is not language dependent, there is no need for considering a final implementation language, during Object Oriented Modeling (OOM). OOT combine structural, control and functional aspects of the system. We will discuss the structural, control and functional aspects of the systems in great detail in block 3 of this course.

In this unit, we will discuss the basic notions of object orientation. This unit will cover discussion on objects, classes, links, association, generalization, and inheritance. We will discuss the basics of an object model with the help of an example. Towards the end of this unit we will cover the benefits of OOM. In this unit, you will also be introduced to some OOAD tools.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- explain basics of object oriented Modeling;
- define Objects and Classes;
- explain the concepts of links and Associations;
- explain the concept of Generalization and Inheritance;
- describe benefits of Object Oriented Modeling, and
- explain the use of some OOAD tools.

1.2 OBJECT ORIENTED MODELING

Object oriented modeling is entirely a new way of thinking about problems. This methodology is all about visualizing the things by using models organized around real world concepts. Object oriented models help in understanding problems, communicating with experts from a distance, modeling enterprises, and designing programs and database. We all can agree that developing a model for a software system, prior to its development or transformation, is as essential as having a blueprint for large building essential for its construction. Object oriented models are represented by diagrams. A good model always helps communication among project teams, and to assure architectural soundness.

It is important to note that with the increasing complexity of systems, importance of modeling techniques increases. Because of its characteristics Object Oriented Modeling is a suitable modeling technique for handling a complex system. OOM basically is building a model of an application, which includes implementation details of the system, during design of the system.

Brooks observes that the hard part of software development is the manipulation of its essence due to the inherent complexity of the problem, rather than the accidents of its mapping into a particular language, which are due to temporary imperfections in our tools that are rapidly being corrected (Brooks-87).

As you know, any system development refers to the initial portion of the software life cycle: analysis, design, and implementation. During object oriented modeling identification and organization of application with respect to its domain is done, rather than their final representation in any specific programming language. We can say that OOM is not language specific. Once modeling is done for an application, it can be implemented in any suitable programming language available.

OOM approach is a encouraging approach in which software developers have to think in terms of the application domain through most of the software engineering life cycle. In this process, the developer is forced to identify the inherent concepts of the application. First, developer organize, and understood the system properly and then finally the details of data structure and functions are addressed effectively.

In OOM the modeling passes through the following processes:

- System Analysis
- System Design
- Object Design, and
- Final Implementation.

System Analysis: In this stage a statement of the problem is formulated and a model is build by the analyst in encouraging real-world situation. This phase show the important properties associated with the situation. Actually, the analysis model is a concise, precise abstraction and agreement on how the desired system must be developed. You can say that, here the objective is to provide a model that can be understood and criticized by any application experts in the area whether the expert is a programmer or not.

System Design: At this stage, the complete system architecture is designed. This is the stage where the whole system is divided into subsystems, based on both the system analysis model and the proposed architecture of the system.

Object Design: At this stage, a design model is developed based on the analysis model which is already developed in the earlier phase of development. The object design decides the data structures and algorithms needed to implement each of the classes in the system with the help of implementation details given in the analysis model.

Final Implementation: At this stage, the final implementation of classes and relationships developed during object design takes place a particular programming language, database, or hardware implementation (if needed). Actual implementation

should be done using software engineering practice. This helps to develop a flexible and extensible system.

Whole object oriented modeling is covered by using three kinds of models for a system description. These models are:

- object model,
- dynamic model, and
- functional model.

Object models are used for describing the objects in the system and their relationship among each other in the system. The dynamic model describes interaction among objects and information flow in the system. The data transformations in the system are described by a functional model. **All three models are applicable during all stages of development**. These models bear the responsibility of acquiring implementation details of the system development. It is important to note that you cannot describe a system completely until unless all three modes are described properly. In block 3 of this course, we will discuss these three models in detail.

Before we discuss the characteristics of object oriented modeling, let us see how object oriented development is different from structured development of the system. In the structured approach, the main emphasis is **on specifying and decomposing system functionality**. Structured approach is seen as the **most direct way of implementing a desired goal**. A structured approach has certain basic problems, such as, if the requirements of system change then a system based on decomposing functionality may require massive restructuring, and, the system gradually become unmanageable. In contrast to the structured approach, the basic focus of object-oriented approach is to **identify objects** from the application domain, and then to **associate procedures** (methods) around these identified objects.

You can say that object oriented development is an indirect way of system development because in this approach **a holistic view of application domain is considered, and objects are identified** in the related problem domain. A historic view of application helps in realizing the situations and characteristics of the system. Taking a **holistic view of the problem domain rather than considering functional requirements of a single problem give an edge to object oriented development**. Once the objects are created with the needed characteristics, they communicate with each other **by message passing** during problem solving.

☞ Check Your Progress 1

1) What is OOM?

……..

……..

……..

2) List different steps involved in OOM process.

……..

……..

……..

3) Differentiate OO development from structured development.

……..

……..

……..

1.3 BASIC PHILOSOPHY OF OBJECT ORIENTATION

There are several characteristics of object-oriented technology. Some of these characteristics have been discussed in course MCS-024. We have also implemented some of them in Java programming language, although these characteristics are not unique to object-oriented systems in the sense that they vary from object based systems to object oriented systems. However, most of the properties are particularly well supported in object oriented systems.

Now, let us discuss about the basic characteristics around which object oriented systems are developed.

Abstraction

Abstraction is one of the very important concepts of object oriented systems
Abstraction focues on the essential, inherent aspects of an object of the system. It does not represent the accidental properties of the system. In system development, abstraction helps to focus on what an object is supposed to do, before deciding how it should be implemented. The use of abstraction protects the freedom to make decisions for as long as possible, by avoiding intermediate commitments in problem solving. Most of the modern languages provide data abstraction. With the abstraction, ability to use inheritance and ability to apply polymorphism provides additional freedom and capability for system development. When you are using abstraction during analysis, you have to deal with application-domain concepts. You do not have to design and make implementation decisions at that point.

Encapsulation

Encapsulation, or information hiding, is the feature of separating the external aspects of an object, from the internal implementation details of that object. It helps in hiding the actual implementation of characteristics of objects. You can say that encapsulation is hiding part of implementation that do internal things, and these hidden parts are not concerned to outside world. Encapsulation enables you to combine data structure and behaviour in a single entity. Encapsulation helps in system enhancement. If there is a need to change the implementation of object without affecting its external nature, encapsulation is of great help.

Polymorphism

Class hierarchy is the deciding factor in the case of more than one implementation of characteristics. An object oriented program to calculate the area of different Figures would simply call the Find_ Area operation on each figure whether it is a circle, triangle, or something else. The decision of which procedure to use is made implicitly by each object, based on its class polymorphism makes maintenance easier because the calling code need not be modified when a new class is added.

Sharing of Structure and Behaviour

One of the reasons for the popularity of object-oriented techniques is that they encourage sharing at different levels. Inheritance of both data structure and behaviour allows common structure (base class) to be used in designing many subclasses based on basic characteristics of base class, and develop new classes with less effort. Inheritance is one of the main advantages of any object oriented language, because it gives scope to share basic code.

In a broader way we can say that object oriented development not only allows information sharing and reuse within an application, but also, it gives a base for project enhancement in future. As and when there is a need for adding new characteristics in the system, they can be added as an extension of existing basic

features. This can be done by using inheritance, and that too, without major modification in the existing code. But be aware that just by using object orientation you do not get a license to ensure reusability and enhancement. For ensuring reusability and enhancement you have to have a more general design of the system. This type of design can be developed only if the system is properly studied and features of proposed system are explored.

Emphasis on Object Structure, not on Operation Implementation

In object orientation the major emphasis is on specifying the characteristics of the objects in a system, rather than implementing these characteristics. The uses of an object depend highly on the facts of the application and regular changes during development. As requirements extend, the features supplied by an object are much more stable than the ways in which they are used, hence software systems built on object structure are more secure.

While developing a system using the object oriented approach, main emphasis is on the essential properties of the objects involved in the system than on the procedure structure to be used for implementation. During this process what an object is, and its role in system is deeply thought about.

1.4 CHARACTERISTICS OF OBJECT ORIENTED MODELING

In object oriented modeling objects and their characteristics are described. In any system, objects come into existence for playing some role. In the process of defining the roles of objects, some features of object orientation are used. In this section we will discuss these features, which include:

- Class and Objects
- Links and Association
- Generalization and Inheritance

Let us start our discussion with Class and Objects.

1.4.1 Class and Objects

A *class* is a collection of things, or concepts that have the same characteristics. Each of these things, or concepts is called an *object.*

We will discuss, in the next unit of this block, that the class is the most fundamental construct within the UML.

Classes define the basic words of the **system being modeled**. Using a set of classes as the core vocabulary of a software project tends to greatly facilitate understanding and agreement about the meanings of terms, and other characteristics of the objects in the system.

Classes can serve as the foundation for data modeling. In OOM, the term classes is usually the base from which visual modeling tools–such as Rational Rose XDE, Visual Paradigm function and design the model of systems.

Now, let us see how the characteristics that classes share are captured as attributes and operations. These terms are defined as follows:

- *Attributes* are named slots for data values that belong to the class. As we have studied in MCS-024, different objects of a given class typically have at least some differences in the values of their attributes.
- *Operations* represent services that an object can request to affect the behaviour of the object or the system itself.

In unit 3 of this block, we will cover the standard UML notation for OOM in detail. Here, we will mention about standard notation of class. The notation for a class is **a box with three sections**. The top section contains the name of the class in boldface type, the middle section contains the attributes that belong to the class, and the bottom section contains the class's operations as you can see in *Figure 1***.**

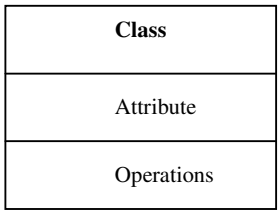

Figure 1: Class notation

You can, also show a class without its attributes or its operations, or the name of the class can appear by itself as shown in *Figure 2***.**

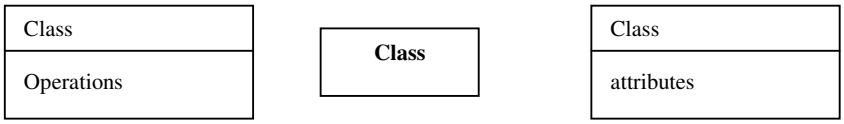

Figure 2: Alternate class notations

The naming convention for classes are as follow:

- Class names are simple nouns or noun phrases.
- Attribute names in a class are simple nouns or noun phrases. The first word is not capitalized, but subsequent words may be capital.
- Operation names are simple verbs. As with attributes, the first word is not capitalized and subsequent words may be capital.

Objects

The notation for an object is the same in basic form as that for a class. There are three differences between the notations, which are:

- Within the top section of the class box, the name of the class to which the object belongs appears after a **colon**. The object may have a name, which appears before the colon, or it may be anonymous, in which case nothing appears before the colon.
- The contents of the top compartment are underlined for an object.
- Each attribute defined for the given class has a specific value for each object that belongs to that class.

You can see the notion of an object you can see in *Figure 3***.**

Figure 3*:* Notation of object

If you look around you will find many examples of real world objects such as your books, your desk, your television, etc.

Everything that the software object knows (state) and can do (behaviour) is expressed by the variables and the methods within that object. In other words, all the objects share states and behaviour. Let us say that a software object that models your real-world bicycle would have variables that indicated the bicycle's current state: its speed is 20 mph, and its current gear is the 3rd gear, etc.

Communication by Message Passing

You will agree that a single object alone is generally not very useful. Objects usually appear as a components of a larger program or a system. Through the interaction of these objects, functionality of systems are achieved. Software objects interact and communicate with each other by *message passing* to each other. When object X wants object Y to perform one of methods of object Y, object X sends a message to object Y. Message passing provide two significant benefits:

- An object's characteristics are expressed through its methods, so message passing supports all possible interactions between objects.
- It closes the gap between objects. Objects do not need to be in the same process, or even on the same machine, to send and receive messages back and forth to each other.

1.4.2 Links and Association

Links and associations are the basic means used for establishing relationships among objects and classes of the system. In the next subsection we will discuss links and associations which are used for representing relationship.

General Concepts

A link is a physical or conceptual connection between objects for example, a student, **Ravi study in IGNOU**. Mathematically, you can define a link as a tuple that is an ordered list of objects. Further, a link is also defined as an instance of an association. In other words you can say that an association is a group of links with a common structure and common meanings, for example, a student study in a university. All the links in an association connects objects from the same classes. A link is used to show a relationship between two (or more) objects.

Association and classes are similar in the sense that classes describe objects, and association describe **links**. *Figure 4a* shows us how we can show the association between Student and University

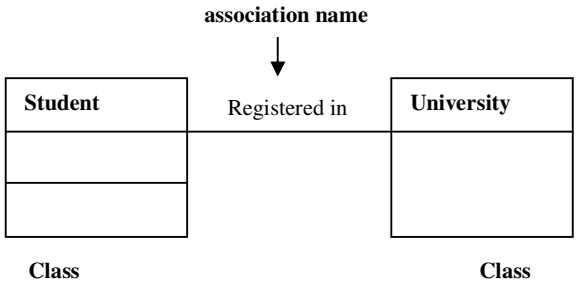

Figure 4a: Association

Note that every association has roles. For example, in *Figure 4b* you can see that two classes, Student and University, have their defined roles. Here you can also see that binary association has two roles, one from each class.

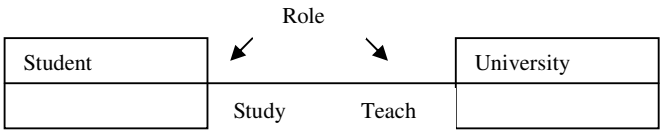

Figure 4b: Roles in association

Associations may be binary, ternary, or have higher order. In exercise, the vast majority of association are binary or ternary associations. But a ternary association is formed compulsion; they cannot be converted into binary association. If a ternary association is decomposed in some other association, some information will be lost. In *Figure 5* you can see a ternary association.

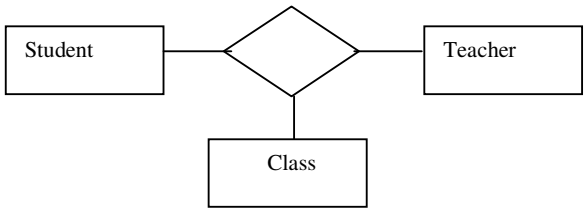

Figure 5: Ternary association

Multiplicity

Multiplicity in an association specifies how many objects participate in a relationship. Multiplicity decides the number of related objects. Multiplicity is generally explained as "one" or "many," but in general it is a subset of the non-negative integers.

Table 1: Multiplicity Indicators.

Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
n	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)

In associations, generally movement is in both the directions of the relationships but if you want to be specific in any particular direction, you have to mark it by an arrow as given in *Figure 6.*

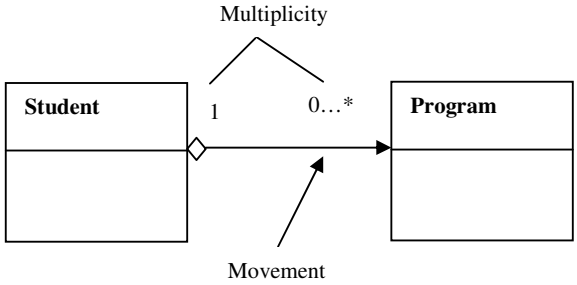

Figure 6: Association and movement

Aggregation

Aggregation is a special form of association, which models the "part-whole" or "a-part-of" relationship as an aggregate (the whole) and parts. The most considerable property of aggregation is transitivity, that is, if X is part of Y and Y is part of Z, then X is part of Z. Aggregation is seen as a relationship in which an assembly class is related to component class. In this component objects are not having separate existence, they depend on composite objects as you can see in *Figure 7* Exam Schedule is not having separate existence.

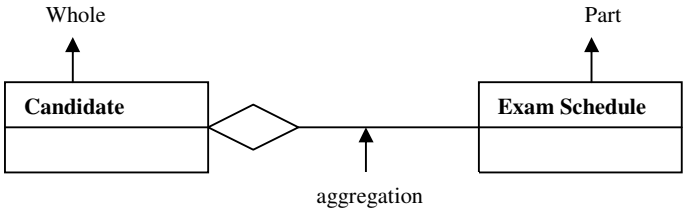

Figure 7: Association and whole-part relationship

☞ Check Your Progress 2

1) What is abstraction?

……

……

2) What is association? Give example of association.

……

……..

3) What is multiplicity in associations? Give example to explain multiplicity?

……

……

1.4.3 Generalization and Inheritance

In this section we will discuss the concepts of generalization, inheritance, and their uses in OOM.

Generalization

Generalization and inheritance are powerful abstractions for sharing the structure and/or behaviour of one or more classes.

Generalization is the relationship between a class, and it defines a hierarchy of abstraction in which subclasses (one or more) inherit from one or more superclasses.

Generalization and inheritance are transitive across a subjective number of levels in the hierarchy. Generalization is an "is-a-kind of" relationship, for example, Saving Account is a kind of Account, PG student is kind of Student, etc.

The notation for generalization **is a triangle connecting a super class to its subclasses**. The superclass is connected by a line to the top of the triangle. The subclasses are connected by lines to a horizontal bar attached to the base of the triangle. Generalization is a very useful construct for both abstract modeling and implementation. You can see in *Figure 8*, a generalization of Account class.

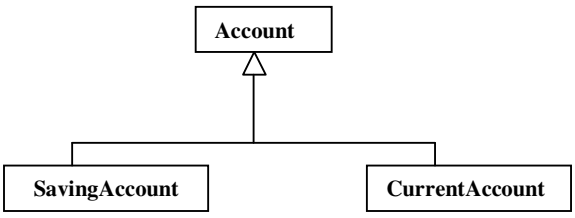

Figure 8: Generalization of account class

Inheritance

Inheritance is taken in the sense of code reuse within the object oriented development. During modeling, we look at the resulting classes, and try to group similar classes together so that code reuse can be enforced. Generalization, specialization, and inheritance have very close association. Generalization is used to refer to the relationship among classes, and inheritance is used for sharing attributes and operations using the generalization relationship. In respect of inheritance, generalization and specialization are two phases of a coin in the sense that if a

subclass is seen from a superclass the subclass is seen as a specialized version of superclass and in, reverse, a superclass looks like general form of subclass.

During inheritance, a subclass may override a superclass feature by defining that feature with the same name. The overriding features (the subclass feature with the same names of superclass features) refines and replaces the overridden feature (the superclass feature).

Now let us look at the diagram given in *Figure 9***.** In this diagram, Circle, Triangle, and Square classes are inherited from Shape class. This is a case of single inheritance because here, one class inherits from only one class.

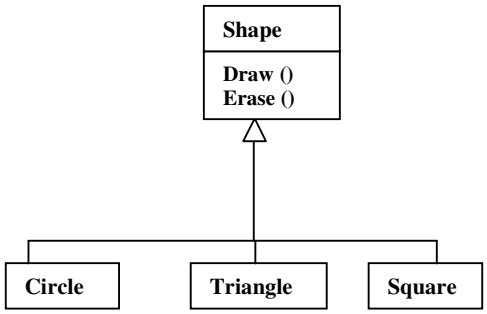

Figure 9: Single inheritance

Multiple inheritance is shown in *Figure 10*. Here, one class is inherited from more than one class.

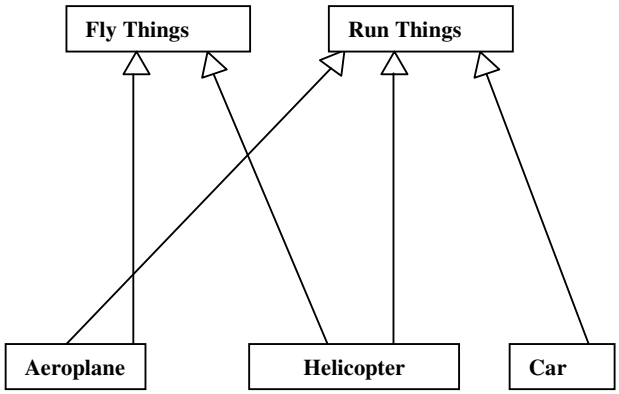

Figure 10: Multiple inheritance

1.5 AN OBJECT MODEL

In object oriented modeling system understanding and on the basis of that identification of classes. Establishing relationship among different classes in the system are the first and foremost activity. Here, we have a simple model of a University System with respect to different levels of courses offered by the University. As you can see in *Figure11*, we have given the basic classes of this system.

This diagram covers different levels of students in the hierarchy. Similarly, for other classes, such as Administration and Faculty, hierarchy level can be drawn to give a broader view of whole system.

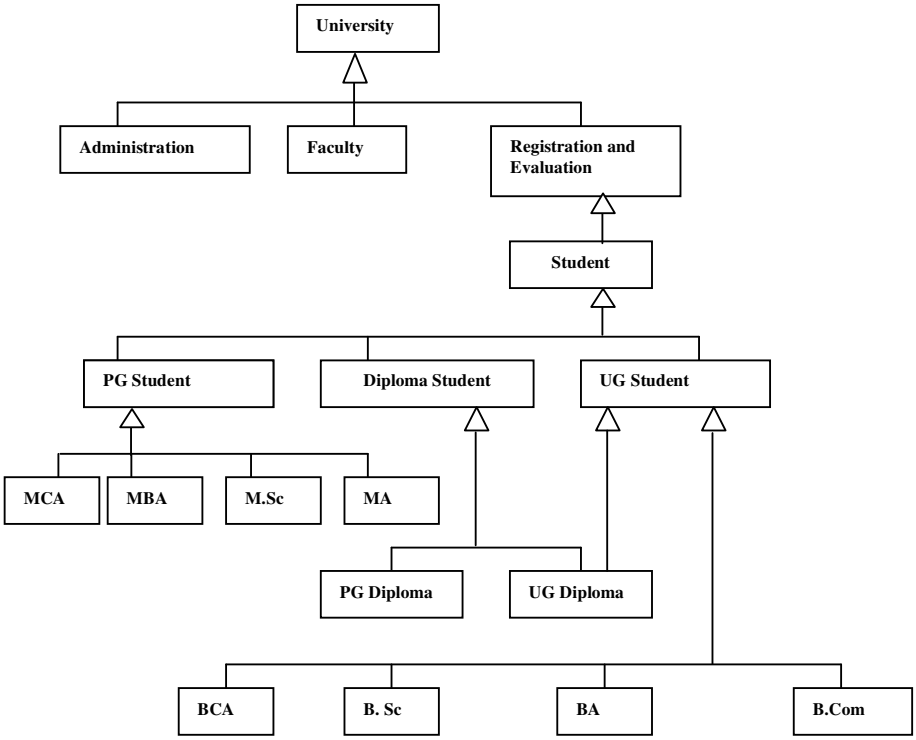

Figure 11: Object model for university system

1.6 BENEFITS OF OBJECT ORIENTED MODELING

There are several advantages and benefits of object oriented modeling. Reuse and emphasis on quality are the major highlights of OOM. OOM provides resistance to change, encapsulation and abstraction, etc. Due to its very nature, all these features add to the systems development:

- Faster development
- Increased Quality
- Easier maintenance
- Reuse of software and designs, frameworks
- Reduced development risks for complex systems integration.

The conceptual structure of object orientation helps in providing an abstraction mechanisms for modeling, which includes:

- Classes
- Objects
- Inheritance
- Association etc.

1.7 INTRODUCTION TO OBJECT ORIENTED ANALYSIS & DESIGN: TOOLS

Unified Modeling Language (UML) is a well accepted language for OOAD. It is used for visualizing, specifying, constructing, and in final documentation. The basic building blocks of UML used for OOAD are things, relationships, and diagrams. Basically, the *Unified Modeling Language* focuses on the concepts of Booch, OMT, and Object Oriented Software Engineering (OOSE). The result of these concepts is a single, common, and widely usable modeling language for users of these and other

methods. The *Unified Modeling Language also* promotes the concept of what can be done with existing methods.

Many modern applications are being developed based on object oriented principles such as classes, methods, and inheritance. To fulfil the needs of such developments, CASE tools offer many benefits for developers building large-scale systems. CASE tools enable us to abstract away from the mess of source code, to a level where design and propose become more clear and easier to understand and modify.

For making and representing these features of the systems to be developed, some tools are used. These tools support UML features and building blocks. Object modeling CASE tools provide support for object oriented modeling notations and methodologies, and they also generate parts of object oriented applications. New versions of many object oriented CASE tools are beginning to address new languages such as Java. Many of these object modeling CASE tools also support relational databases by performing arts of logic, and in some cases, physical database modeling and design, including schema generation and reverse engineering of RDBMS tables, and other elements.

Here, we have tried to give a list collected from many sites and individual searches, for UML modeling tools. Since UML is a fast growing engineering this list may keep on changing all the time.

Action Semantics: This is a group of firms that have responded to the OMG's RFP to define the action semantics for UML.

ArgoUML: This is a domain-oriented design environment that provides cognitive support of object oriented design. ArgoUML provides some of the same automation features of a commercial CASE tool.

ARTiSAN Software: This provides a variety of UML based CASE tools, including a real time modeling tool.
BridgePoint : This provide features of a real time UML modeling tool.

GDPro : this is a full suite of UML and code management tools.

MagicDraw UML: This has Full support for all UML diagrams: MagicDraw RConverter allows you to convert these UML diagrams into MagicDraw: Activity, Class, Collaboration, Component, Deployment, Sequence,State chart, Three-tiered, and Use Case diagrams.

Rational Rose: IBM Rational RequisitePro is a powerful and easy-to-use tool for requirements and use case management.

Visio 2000 Enterprise: It contains a UML suite that can build diagrams within Visio.

Visual Paradigm: Visual Paradigm for the Unified Modeling Language (VP-UML) is a UML CASE suite. The suite of tools is designed for a wide range of users, including Software Engineers, System Analysts, for building large scale software systems reliably through the use of the object oriented approach.

☞ Check Your Progress 3

1) What is inheritance?

……

……

2) Give an example of multiple inheritance.

……

……

3) Explain the benefit of OOM.

……

……

……..

1.8 SUMMARY

In this unit we have discussed the basic notions of object orientation, and the need for object oriented modeling. It is the basic characteristic of object orientation which makes it possible to develop systems in such a way that the system is open for reusability. In this unit, concepts of abstraction, encapsulation, polymorphism and sharing of structure and behaviour are discussed.

Further, in this unit, we have discussed notions of class and object. We saw that how inheritance, generalization/specialization, and associations are represented. In this unit, a hierarchy of classes representing different levels of students in a University system is representated, reuse and quality are mentioned as benefits of OOM, and in the last section, some tools, which support UML designs are mentioned.

1.9 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Object oriented modeling is a approach through which modeling of the systems are done by visualizing the system based on the real world concepts. Object oriented modeling is language independent and generally used for complex system development.

2) Steps involve in OOM are:

   System Analysis

   System Design

   Object Design and

   Final implementation.

3) Structured approach of problem solving is based on the concept of decomposition of system in to subsystem. In this approach of system development readjustment of some new changes in the system is very difficult. On the other hand in object oriented approach holistic view of application domain is considered and related object are identified. Further classification of objects are done. Object oriented approach give space for further enhancement of the system without too much increase in systems complexity.

Check Your Progress 2

1) Abstraction in object orientation is a concept which provide opportunity to express essential properties of the object without providing much details of implementation of these properties.

2) Association is used for establishing relationships between classes. Association describe links between (among) classes. For example, if a professor works in a university then it can be represented as

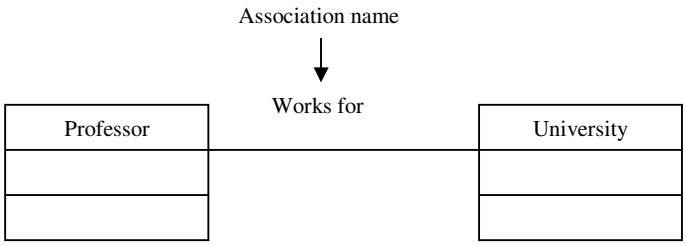

Figure 12: Association of professor and university

3) Multiplicity in an association indicate the number of objects participate in a relationship. For example in the association given in *Figure 13* you can see that one player can play for one team at a time so here multiplicity is 1.

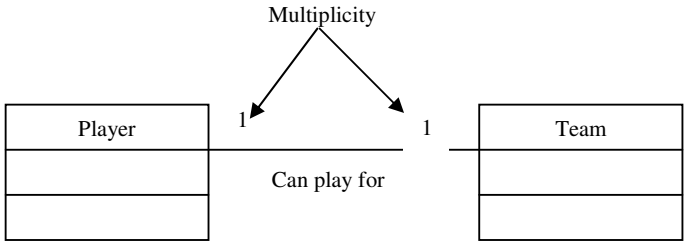

Figure 13: Multiplicity in association

Check Your Progress 3

1) Inheritance is an object orientation concept which allow reusability of design/code. Basic meaning of inheritance is that if one class is already defined than another class which also passes the property of existing class can be defined and inherit the property of existing class. For example, if a class named student is defined and another class for Post Graduate students is to be defined then PG Student class can inherit student class.

2) One example of multiple inheritance is a committee for students affair which consist of faculty and administrative staff member.

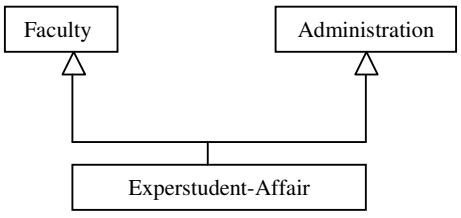

Figure 14: Multiple inheritance

3) Major benefits of object oriented modeling is that development of the system become fast, quality of the system is increase. It give freedom of use of existing design and code. Help in development of complex system with less risk due to the basic properties of object orientation which include class, objects and inheritance.