Chapter - 2

About OOP

About OOP

What is OOP?

An OOP is a set of methodologies or principles using which large complex programs can be developed easily which are reusable, reliable and maintainable.

Features of OOP

The features of OOP are the following.

- Encapsulation
- Data Abstraction
- Inheritance
- Polymorphism
- Message Passing
- Extensibility
- Persistence
- Delegation
- Genericity

Encapsulation: It is a mechanism that associates or binds the code(methods) and the data(instance variables) it manipulates into a single unit at one place and keeps them safe from external interference and misuse. Advantage of encapsulation is security for data. The code that is binded with data can only access the data and no other code can access the data. Therefore this mechanism of binding data and code into a single unit provides security for data. This is one of the main advantage of OOP. In JAVA, encapsulation can be achieved using classes and objects.

Note : Assume **structure** is **class** and **structure variable** as **object**.

Data Abstraction: The technique of creating new data types using encapsulated-items that are well suited to an application to be programmed is known as data abstraction. It provides the ability to create user-defined data types, for modeling a real world object, having the properties of built-in data types and a set of permitted operators. The class is the construct in JAVA for creating user-defined data types called **A**bstract **D**ata **T**ypes (**ADT**s)

Abstraction refers to the act of representing essential features without including the background details or explanations.

Explanation: The user of the class is presented with a description of the class. The internal details of the class, which are not essential to the user, are not presented to him. This is the concept of information hiding or abstraction or data encapsulation. As far as the user is concerned, the knowledge of accessible data and methods of a class is enough. These interfaces, usually called the user interface methods, specify their abstracted functionality. Thus to the user, a class is like as black box with a characterized behaviour.

Inheritance: Inheritance allows the extension and reuse of existing code without having to rewrite the code from scratch. Inheritance involves the creation of new classes(sub classes) from the existing ones(super classes), thus enabling the creation of a hierarchy of classes that simulate the class and subclass concept of the real world. The new sub class inherits the members of the super class and also adds its own members. There are six forms of inheritances in OOP, they are

1) Single Inheritance
2) Multiple Inheritance
3) Hierarchical Inheritance
4) Multi-level Inheritance
5) Hybrid Inheritance
6) Multipath Inheritance

Java support only three types of inheritances such as **Single** Inheritance, **Hierarchical** Inheritance, **Multi-level** Inheritance.

Java does not support **multiple**, **hybrid** and **multipath** inheritances.

Advantages of inheritance

1) Re-usability of code.
2) Elimination of duplication of code.
3) Extending the features of a class.
4) In less time new classes or software components can be developed.
5) Code sharing can occur at several levels.

Polymorphism: The word polymorphism is derived from the Greek term, **poly** means **many** and **morphism** means **forms**. Polymorphism means one thing exhibiting many forms. An operation (method) may exhibit different behaviours in different instances. The behaviour depends upon the types of data used in the operation. Polymorphism allows a single name to be used for more than one related purpose. The following are other different ways of achieving polymorphism in a JAVA program:

- Method Overloading
- Dynamic Binding

Method overloading: It is the concept of two or more functions having same name that are different in their signature.

Dynamic Binding: It is the concept of binding of method call with the address of called method. This binding can be of two types. They are **compile time binding** and **dynamic binding**. Dynamic binding (also called as *late binding*) is a mechanism that allows a method call to bind with called method only at run-time. It is associated with polymorphism(overriding) and inheritance.

Dynamic binding is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time. Dynamic binding is important because this is how JAVA implements **run-time polymorphism**.

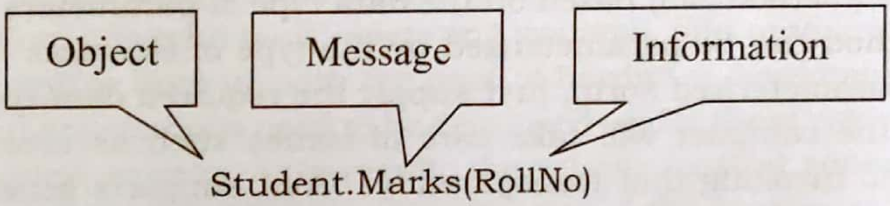
Message Passing or Message Communication

In conventional programming languages, a method is invoked on a piece of data (method-driven communication), where as in an object-oriented language, a message is sent to an object (message-driven communication). Hence, conventional programming is based on functional abstraction where as, object oriented programming is based on data abstraction. This is illustrated by a simple example of evaluating the square root of a number. In conventional functional programming, the function sqrt(x) for different data types(x's type), will be defined with different names, which takes a number as an input and returns its square root. For each data type of x, there will be a different version of the method sqrt( ). In contrast, in OOPL (Object-Oriented Programming language), the expression for evaluating the square root of x takes the

form x.sqrt( ), implying that the object x has sent a message to perform the square root operation on itself. Different data types of x, invoke a different function code for sqrt, but the expression (code) for evaluating the square root will remain the same. In object-oriented programming, the process of programming involves the following steps:

1. Create classes for defining objects and their behaviours.
2. Define the required objects.
3. Establish communication among objects through message passing.

Communication among the objects occurs in the same way as people exchange messages among themselves. The concept of programming, with the message passing model, is an easy way of modeling real-world problems on computers. A message for an object is interpreted as a request for the execution of a method. A suitable method is invoked soon after receiving the message and the desired results are generated within an object. A message comprises the name of the object, name of the method and the information to be sent to the object as shown below.



Object-oriented message communication

Like in the real world, objects also have a life cycle! They can be created and destroyed automatically, whenever necessary. Communication between the objects can take place as long as they are alive! Student is treated as an object sending the message Marks to find the marks secured by the student with the specified RollNo. In this case, a method call Marks( ) is treated as a message and a parameter RollNo is treated as information passed to the object.

Extensibility: It is a feature, which allows the extension of the functionality of the existing software components. In JAVA, this is achieved through abstract classes and inheritance.

Persistence: The phenomenon where the object (data) outlives the program execution time and exists between executions of a program is known as persistence. All database systems support persistence. In JAVA, this is not sup

ported. However, the user can build it explicitly using file streams in a program.

Delegation: It is an alternative to class inheritance. Delegation is a way of making object composition as powerful as inheritance. In delegation, two objects are involved in handling a request: a receiving object delegates operations to its delegate. This is analogous to the child classes sending requests to the parent classes. In JAVA, delegation is realized by using object composition. Here, new functionality is obtained by assembling or composing objects. This approach takes a view that an object can be a collection of many objects and the relationship is called the has-a relationship or containership. object composition is an alternative of multiple inheritance. In java, multiple inheritance can be achieved by object composition.

Genericity: It is a technique for defining software components that have more than one interpretation depending on the data type of parameters. Thus, it allows the declaration of data items without specifying their exact data type. Such unknown data types (generic data type) are resolved at the time of their usage (method call) based on the data type of parameters. For example, a sort method can be parameterized by the type of elements it sorts. To invoke the parameterized sort(), just supply the required data type parameters to it and the compiler will take care of issues such as creation of actual method and invoking that transparently. JAVA supports generics from java 1.5 version.

Why OOP?

Object-Oriented Programming popularly called OOP's is one of the buzzwords in the software. On one hand, OOP is a Programming paradigm in its own right and on the other it is a set of software engineering tools which can be used to build more reliable and reusable systems. Another kind of Programming methodology which has already revealed its power in the software field, is structured Programming. At present, Object-Oriented Programming is emerging from research laboratories and invading the field of industrial applications. The software industry has always been in search of a methodology or philosophy, which would eliminate the problems endemic to software in one shot. The latest candidate for this role is **Object Oriented methodology**.

Structured programming and Object-Oriented Programming fundamentally differ in the following way: Structured programming views the two core elements of any program-data and function as two separate entities where

as, OOP views them as a single entity. The benefit of uniting both data and functions into a single unit is security for data.

Object-Oriented programming as a paradigm is playing an increasingly significant role in the analysis, design, and implementation of software systems. Object-Oriented analysis, design, and Programming appear to be the Programming of the 1990's. Proponents assert that OOP is the solution to the software problem. Software developed using Object-Oriented techniques are proclaimed as more reliable, easier to maintain, easier to reuse and enhance, and so on. The Object-Oriented Paradigm is effective in solving many of the outstanding problems in software engineering.

Why New Programming paradigms?

With the continuous decline of hardware cost, high speed computing systems are becoming economically feasible. Innovations in the field of computer architecture supporting complex instructions is in turn leading to the development of better programming environments, which suit the hardware architecture. More powerful tools, operating systems, and programming languages are evolving to keep up with the pace of hardware development. Software for different applications need to be developed under these environments, which is a complex process. As a result, the relative cost of software is increasing substantially when compared to the cost of the hard ware of a computing system. Over several years the cost of software development and maintenance is increasing where as hardware cost is declining. Software maintenance is the process of modifying or extending the capabilities of the existing software. It requires mastery over the understanding and modifying the existing software and finally revalidating the modified software.

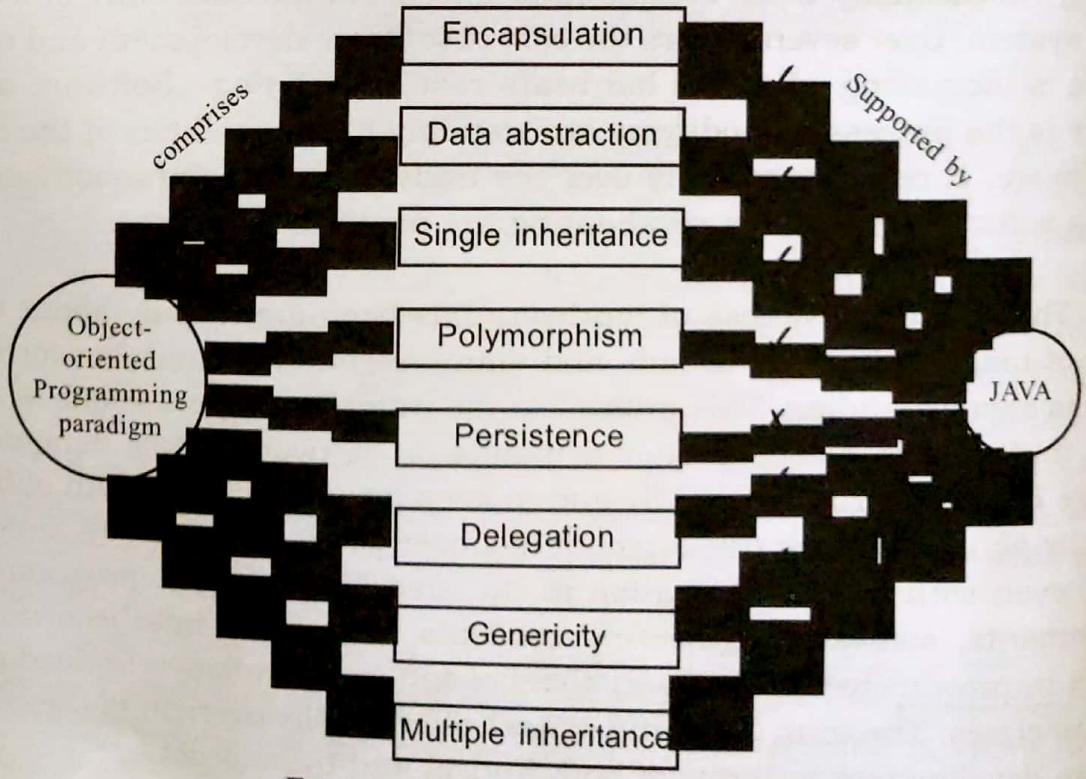
The cost effectiveness of hardware has been growing by about three orders of magnitude every decade and simultaneously the market for computers is also expanding. This multiplies the number of applications of computers and in turn places greater demands on software. While demand for software has been growing rapidly and to keep pace with the growth of hardware, the actual software development has been progressing slowly. Unfortunately, even with all the innovation in the area of languages, programming environments, software engineering concepts, etc., there has been no significant improvement in the productivity of software development, leading to software crises. The term "software crises" refers to the overrun of the cost of software development in terms of both budget and time-target.

The software crisis from the beginning, forcing for the development of software engineering principles, tools, and better programming paradigms to build more reliable and reusable systems. The solution to overcome software crisis is the Object-Oriented Paradigm.

OOP's! a New Paradigm

Object-Oriented Programming is a new way of solving problems with computers; instead of trying to mould the problem into something familiar to the computer, the computer is adapted to the problem. Object-Oriented Programming is designed around the data being operated upon as opposed to the operations themselves.

OOP languages provide the programmer the ability to create class hierarchies, instantiate co-operative objects collectively working on a problem to produce the solution and send messages between objects to process themselves. The power of object-oriented languages is that the programmer can create modular, reusable code and as a result, formulate a program by composition and modification of the existing modules. Flexibility is gained by being able to change or replace modules without disturbing other parts of the code. Software development speed is gained, on one hand, by reusing and enhancing the existing code.



Fetures of object-oriented paradigm

Object Oriented Programming

The easy way to master the management of complexity in the development of a software system is through the use of data abstraction (hiding). Procedure abstraction is suitable for the description of abstract operation, but it is not suitable for the description of abstract objects. This is a serious drawback in many applications since, the complexity of the data objects to be manipulated contribute substantially to the overall complexity of the problem.

The emergence of data-drive methods provides a disciplined approach to the problems of data abstraction in algorithmic oriented languages. It has resulted in the development of object-based language support only data abstraction. Object-based languages do not support features such as **inheritance** and **polymorphism**. Depending on the object features supported, the languages are classified into two categories.

1. Object-Based Programming Languages
2. Object-Oriented Programming Languages

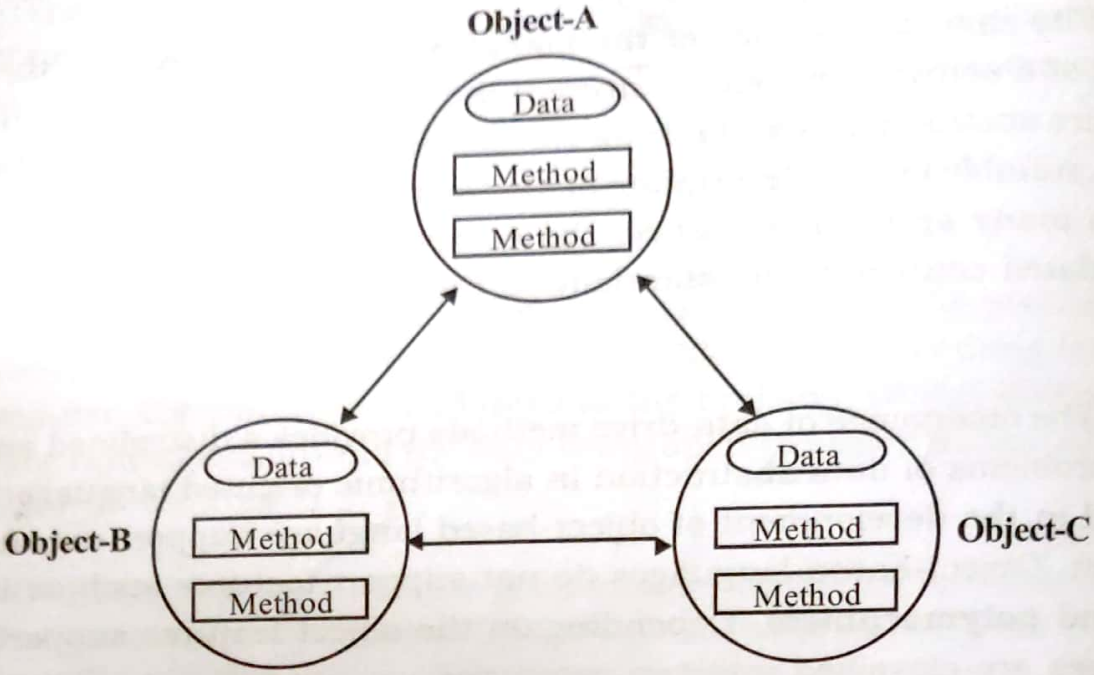
Object-based programming languages support **encapsulation** and **object identity** without supporting important features of OOP languages such as **polymorphism**, **inheritance**, and **message based communication**. Ada is one of the typical object-based programming languages:

Object-based languages=*encapsulation + object Identity*

Object-oriented languages incorporate all the features of objects-based programming languages along with **inheritance** and **polymorphism**. Therefore, an Object-oriented programming languages is defined by the following statement:

Object-oriented languages = *Object based features + Inheritance + polymorphism*

In OOP languages a module is a logical collection of classes and objects, instead of subprograms as in the earlier languages. Thus making classes and objects as the fundamental building blocks of OOPs.



Object oriented programming

The following are the important features of object-oriented programming:

- Improvement over the structured programming paradigm
- Emphasis on data rather than algorithm
- Data abstraction is introduced in addition to procedural abstraction
- Data and associated operation are unified into a single unit, the objects are grouped with common attributes, operations and semantics
- Programs are designed around the data being operated, rather than operation themselves (data decomposition rather than algorithmic decomposition)
- Relationships can be created between similar, yet distinct data types

Examples: C++, Smalltalk, Eiffel, Java, etc.