Objects :

Objects are the basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle. They may also represent user-defined data such as vectors, time and lists. Programming problem is analyzed in terms of objects and the nature of communication between them. Program objects should be chosen such that they match closely with the real-world objects. Objects take up space in the memory and have an associated address like a record in Pascal, or a structure in C. When a program is executed, the objects interact by sending messages to one another. For example :

(1) If "customer" and "account" are two objects in a program, then the customer object may send a message to the account object requesting for the bank balance.

(2) Orange is an object, its characteristics are its orange colour, special shape and its behaviour that it is juicy and sweet/sour in taste.

Classes :

Objects contain data and code to manipulate that data. The entire set of data and code of an object can be made a user-defined data type with the help of a class. In fact, objects are variables of the type class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created. *A class is thus a collection of objects of similar type*.

For example, mango, apple and orange are members of the class fruit. Classes are user-defined data types and behave like the built-in types of a programming language.

The syntax used to create an object is no different than the syntax used to create an integer object in C. If fruit has been defined as a class, then the statement

fruit mango;

will create an object mango belonging to the class fruit.

Data Abstraction and Encapsulation :

The wrapping up of data and functions into a single unit (called class) is known as encapsulation. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. *This insulation of the data from direct access by the program is called data hiding or information hiding*.

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes. The attributes are sometimes called ***data members*** because they hold information. The functions that operate on these data are called ***member functions***.

Inheritance :

Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. For example, the bird robin is a part of the class 'flying bird' which is again a part of the class 'bird'. The principle behind this sort of division

is that each derived class shares common characteristics with the class from which it is derived.

In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one. The new class will have the combined features of both the classes.

Polymorphism :

Polymorphism is an another important OOP concept. Polymorphism, a Greek term, means the ability to take more than one form. An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation. For example, consider the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation. The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.

Dynamic Binding :

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.

Message Passing :

An object oriented program consists of a set of objects that communicate with each other. Objects communicate with each other by sending and receiving information. Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.

BENEFITS OF OOP :

OOP offers several benefits to both the program designer and the user. The principal advantages are:

- Through inheritance, we can eliminate redundant code and extend the use of existing classes.
- We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.
- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple instances of an object to co-exist without any interference.
- It is possible to map objects in the problem domain to those in the program.
- It is easy to partition the work in a project based on objects.
- The data-centered design approach enables us to capture more details of a model in implementable form.
- Object-oriented systems can be easily upgraded from small to large systems.
- Message passing techniques for communication between objects makes the interface descriptions with external systems much simpler.

- Software complexity can be easily managed.

APPLICATIONS OF OOP :

OOP has become one of the programming buzzwords today. There appears to be a great deal of excitement and interest among software engineers in using OOP. Applications of OOP are beginning to gain importance in many areas. The most popular application of object-oriented programming, up to now, has been in the area of user interface design such as windows. Hundreds of windowing systems have been developed, using the OOP techniques.

Real-business systems are often much more complex and contain many more objects with complicated attributes and methods. OOP is useful in these types of applications because it can simplify a complex problem.

The promising areas for application of OOP include:

- Real-time systems
- Simulation and modeling
- Object-oriented databases
- Hypertext, hypermedia and expertext
- Al and expert systems
- Neural networks and parallel programming
- Principles of Object-Oriented Programming
- Decision support and office automation systems
- CIM/CAM/CAD systems

The richness of OOP environment has enabled the software industry to improve not only the quality of software systems but also its productivity. Object-oriented technology is certainly changing the way the software engineers think, analyze, design and implement systems.