**Object-Oriented Programming Paradigm**

The major motivating factor in the invention of object-oriented approach is to remove some of the flaws encountered in the procedural approach. OOP treats data as a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the functions that operate on it, and protects it from accidentalmodification from outside functions. OOP allows decomposition of a problem into a numberof entities called ***objects***and then builds data and functions around these objects. Theorganization of data and functions in object-oriented programs is shown in Fig. 1.6. Thedata of an object can be accessed only by the functions associated with that object. However,functions of one object can access the functions of other objects.



**Some of the striking features of object-oriented programming are:**

* Emphasis is on data rather than procedure.
* Programs are divided into what are known as objects.
* Data structures are designed such that they characterize the objects.
* Functions that operate on the data of an object are tied together in the data structure.
* Data is hidden and cannot be accessed by external functions
* Objects may communicate *with each other* through functions.
* New data and functions can be easily added whenever necessary.
* *Follows bottom-up* approach in program design.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **Basic concepts of Object- Oriented Programming**
2. Objects
3. Classes
4. Inheritance
5. Data abstraction and encapsulation
6. Polymorphism
7. Dynamic binding
8. Message passing
9. **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. Objects take up space in the memory and have an associated address.

When a program is executed, the objects interact by sending messages to one another. For example, if "customer" and "account" arc two objects in a program, then the custom object may send a message to the account object requesting for the bank balance.

Each object contains **data and code to manipulate the data.** An object is a unit that contains data and the functions that operate on that data. An object is said to be an instance of a class. In C++ the class variables are known as objects.

****

1. **Classes:**

The entire set of data and code of an object can be made a user defined data types with the help of class. Objects are the variables of type class. Once a class has been defined, we can create any number of objects belonging to that class. A class is a collection of objects of similar type. Classes are user-defined data types and behave like the built-in types of a programming language.

Ex: mango, apple and orange are members of the class fruit.

If fruit has been defined as a class, then the statement--fruit mango;

Will create an object **mango** belonging to the class fruit.

1. **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'. Each subclass shares common characteristics with the class from which it is derived. In addition to the characteristics shared with other member of the class, each subclass has its own particular characteristics. In C++ original class known as base class, other classes that share the characteristics are known as derived classes.



**Reusability**: 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. The real appeal and power of the inheritance mechanism is that it allows the programmer to reuse a class that is almost, but not exactly, what he wants, and to tailor the class in such a way that it does not introduce any undesirable side-effects into the rest of the classes.

1. **Data abstraction and encapsulation**

**Encapsulation :**The wrapping up of data and functions into a single unit is known as encapsulation. Here data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. The 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 explanation. 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. They encapsulate all the essential properties of the objects that are to be created. The attributes are sometimes called ***data members***because they hold information. The functions that operate on these data are sometimes called ***methods or member functions.***

Since the classes use the concept of data abstraction, they are known as **Abstract Data Types (ADT).**

1. **Polymorphism**

Polymorphism, a Greek term, means the ability to take more than one form. An operation may exhibit different behaviours in differentinstances. The behaviour depends upon the types of data used in the operation.

Ex: 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 behaviours in different instances is known as **operator overloading*.***

Same function name can be used to handle different number and different types of arguments known as **function overloading.**



1. **Dynamic binding:**

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code associated with a given **procedure call is not known until the time of the call at run time.**

Consider the procedure “draw” in fig. 1.9. by inheritance, every object will have this procedure. Its algorithm is, however, unique to each object and so the draw procedure will be redefined in each class that defines the object. **At run-time, the code matching the object under current reference will be called**

1. **Message Passing**

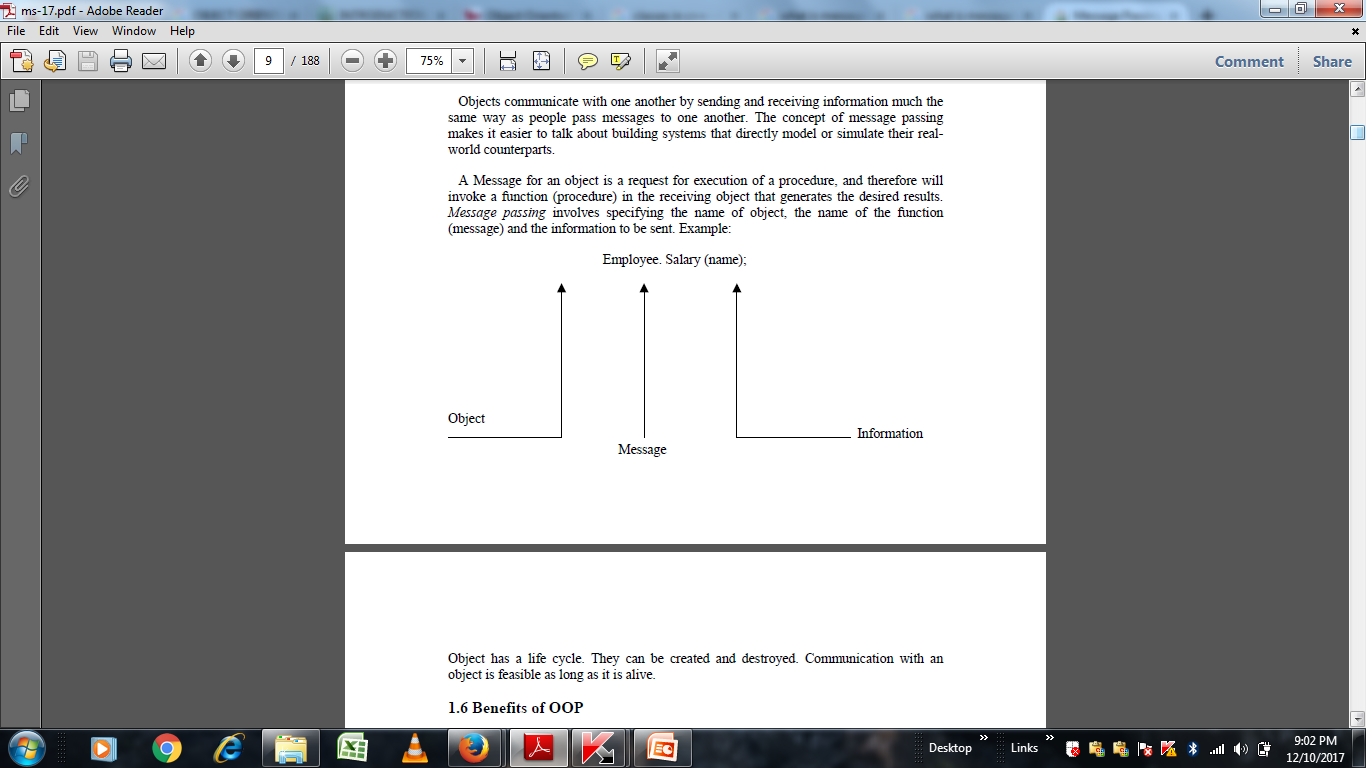
An object-oriented program consists of a set of objects that communicate with each other. The process of programming in an object-oriented language, therefore, involves the following basic steps:

1. Creating classes that define objects and their behaviour,

2. Creating objects from class definitions, and

3. Establishing communication among objects.

Objects communicate with one another by sending and receiving information much the same way as people pass messages to one another. The concept of message passing makes it easier to talk about building systems that directly model or simulate their real-world counterparts.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**2.) Benefits of OOP**

OOP offers several benefits to both the program designer and the user. Object-orientation contributes to the solution of many problems associated with the development and quality of software products. The new technology promises greater programmer productivity, better quality of software and lesser maintenance cost. 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.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3.) Applications of OOP:**

Applications of oop are 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-time systems
* Simulation and modeling
* Object-oriented databases
* Hypertext, hypermedia and expertext
* AI and expert systems
* Neural networks and parallel programming
* Decision support and office automation systems
* CIM/CAM/CAD systems

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*