Introduction to OOPS

The purpose of the *programming language* is to express the solution to a problem with the help of an *algorithm*. There are basically two ways of modelling a solution-

- Structured Programming model (Procedural programming model)
- Object Oriented Programming Model.

High Level Programming Languages (HLPL):

Using HLPL programmers could write a series of English-Like instructions that a *compiler* could translate into the binary language of computers. In its early stage, it was used to write solutions for the simple task such as calculations. But, its ability to write complex problems grows with the increase of the capacity and capability of the computer system.

Limitations of the other programming languages:

- No Re-usability of the existing code.
- The control of the execution within a program was transferred via *goto* statement. (Mismanagement of the flow of control in the program)
- Only global variables declaration.
- The programs were too long to understand and maintain.
- They were unstructured programming languages.

Structured Programming (1960):

- In this paradigm the long programs can be broken down into smaller units of few hundred statements.
- Functions/subroutines/procedures are introduced in this paradigm.
- Programs are collection of functions.
- Information can be passed between the functions using parameters.
- Another important concept in structured programming is – "*Abstraction*". Abstraction permits developers or programmers to look at something without being concerned with its internal details.

Advantages:

- Reusability of the existing piece of code.
- Instead of *goto* statement- powerful control instructions.
- Concept of local variables. (These variables cannot be accessed outside the function's scope)
- More simple and manageable coding.

Limitations:

- Can not model real world problems in natural ways.
- Limited reusability.

- Very difficult to impose for large programs

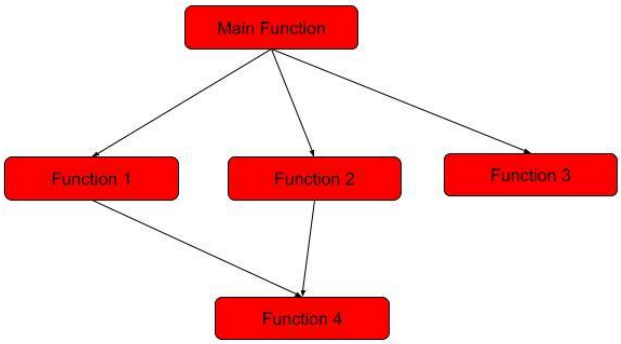

Fig 1. Calling of Functions.

Object-Oriented Programming (OOPS):

In this paradigm the basic emphasise is put on objects rather than functions. Here the characteristics of the objects decides the procedures/ functions to apply to them.

Example: To build a house, to grow a rose plant or to repair a vehicle, first we think about the object and its purpose and behaviour. Then we select the tools and procedures.

In OOPS, two kinds of abstraction can be found- "*functional abstraction"* and "*data abstraction".*

Characteristics of OOPS:

- **Object (Data hiding):** Objects are real world Entities. The basic idea of OOP is to combine both the data and functions that operate on data into a single unit which is nothing but an object. The Object's data are called as *data members* and the functions are called as *member functions*.
  Examples:
  - Employees in a Payroll processing system.
  - Data structures (linked lists, stacks, queues).
  - Computers.
  - Books.

- **Class (Encapsulation):** Classes are used to create user defined data types which are not built-in data types (int, float, long etc.). Class can also be called as a blueprint or a plan or a template.

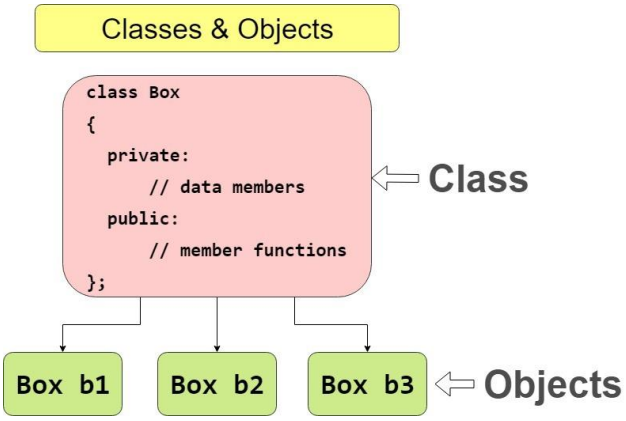

Fig 2. Objects and classes.

- **Inheritance:** Through inheritance, in oops we achieve reusability. Using this concept new classes can be built on top of the old ones. The new class (derived class), can inherit the data and functions of the original class (base class). The derive class can add its own data elements and functions in addition to those it inherits from its base class.

  Example

- **Polymorphism:** Polymorphism is the ability of any data to be processed in more than one form. The word itself indicates the meaning as poly means many and morphism means types. Polymorphism is one of the most important concepts of object-oriented programming language. The most common use of polymorphism in object-oriented programming occurs when a parent class reference is used to refer to a child class object. The concept of polymorphism (One thing with several distinct forms) can be extended even to operators.
- **Containership:** The objects can be categorised into multiple categories. Containership relationship can be observed in many real-world problems.

- **Re-usability:** Once a class is completed and tested, it can be distributed to other programmers for use in their own programs.