# Module 3: Basics of Object-Oriented Programming (OOP)

Pseudocode is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program.

Pseudocode summarizes a program’s flow but excludes underlying details. System designers write pseudocode to ensure that programmers understand a software project's requirements and align code accordingly.

Object-oriented design started right from the moment computers were invented. Programming was there, and programming approaches came into the picture. Programming is basically giving certain instructions to the computer. So, in this module, we are going to learn some basic concepts of OOPs along with Pseudo code.

## Module Objective

**At the end of the module, you will be able,**

- Illustrate about pseudo code.
- To learn the meaning of OOPs.
- To know basic concepts of OOPs.
- Dramatize the questions and solve the questions
- Dramatize the questions and solve the questions

## Pseudo Code Introduction

### What is Pseudo Code?

Pseudo code is a term which is often used in programming and algorithm-based fields. It is a methodology that allows the programmer to represent the implementation of an algorithm. Simply, we can say that it’s the cooked-up representation of an algorithm.

- Often at times, algorithms are represented with the help of pseudo codes as they can be interpreted by programmers no matter what their programming background or knowledge is.
- Pseudo code, as the name suggests, is a false code or a representation of code which can be understoodby even a layman with some school level programming knowledge.
- It’s simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can’t be compiled or interpreted by the computer.

## Advantages of Pseudocode

- Improves the readability of any approach. It’s one of the best approaches to start implementation of an algorithm.
- Acts as a bridge between the program and the algorithm or flowchart. Also works as a rough documentation, so the program of one developer can be understood easily when a pseudo code is written out. In industries, the approach of documentation is essential. And that’s where a pseudo-codeproves vital.
- The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer.

**How to write a Pseudo-code?**

- Arrange the sequence of tasks and write the pseudocode accordingly.
- Start with the statement of a pseudo code which establishes the main goal or the aim.

Example:

This program will allow the user to check the number whether it's even or odd.

- The way the if-else, for, while loops are indented in a program, indent the statements likewise, as it helps to comprehend the decision control and execution mechanism. They also improve the readability to a great extent.

Example:

```
if "1"
print response "I am case 1"

                    if "2"

print response "I am case 2"
```

- Use appropriate naming conventions. The human tendency follows the approach to follow what we see. If a programmer goes through a pseudo code, his approach will be the same as per it, so the naming mustbe simple and distinct.
- Use appropriate sentence casings, such as CamelCase for methods, upper case for constants and lower case for variables.

- Elaborate everything which is going to happen in the actual code. Don’t make the pseudo code abstract.
- Use standard programming structures such as ‘if-then’, ‘for’, ‘while’, ‘cases’ the way we use it in programming.
- Check whether all the sections of a pseudo code is complete, finite and clear to understand and comprehend.
- Don’t write the pseudo code in a complete programmatic manner. It is necessary to be simple to understand even for a layman or client, hence don’t incorporate too many technical terms.

## Introduction to OOPs

One must know OOPs Language is nothing but **Object-Oriented Programming Language**.

OOPs took the basic idea of structured programming and combined them with several new concepts. OOPs does not allow data to move freely between the function. It combines with both data and function that operate on that data into a single unit called an object.

It is defined as, a programming methodology that associates data structures with set of operators which act uponit.

In other words, it can be also defined as, “An approach resulting in modular programs by creating partitioned memory area for both data and function, which can be used as templates for creating copies of such modules.”

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

- Emphasis on data rather than procedure.
- Programs are divided among 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.

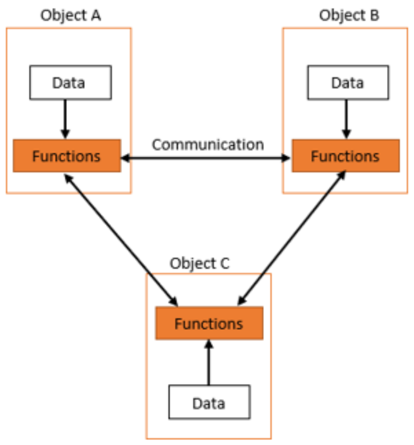


**Fig. Organization of Data and Functions in OOP**

Distinguish between Structured Programming and Object-oriented Programming:

**Object-oriented Programming: -**

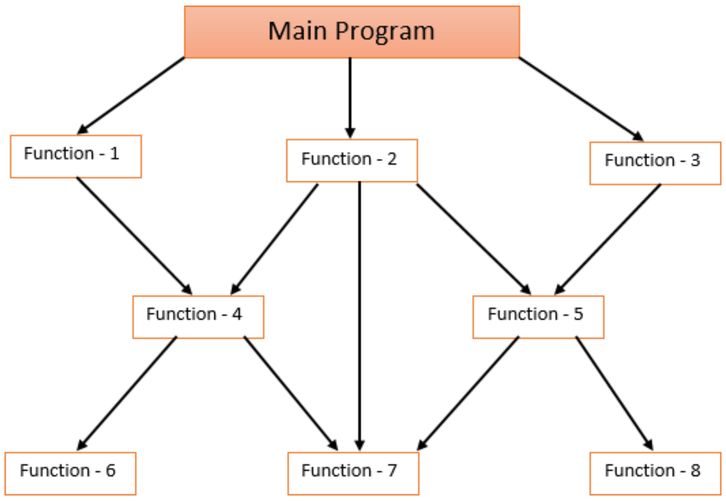
- The central concept of object-oriented programming is the object, which is a kind of module containing data and subroutines.
- An object is a kind of self-sufficient entity that has an internal state (the data it contains) and that can respond to messages (calls to its subroutines).
- A student-records object, for example, has a state consisting of the details of all registered students. If a message is sent to it telling it to add the details of a new student, it will respond by modifying its state to reflect the change.
- If a message is sent telling it to print itself, it will respond by printing out a list of details of all registered students.

## Structured Programming

A particular method or family of methods that a software engineer might use to solve a problem is known as a **methodology**. During the 1970s and into the 80s, the primary software engineering methodology was structured programming. The structured programming approach to program design was based on the following method:

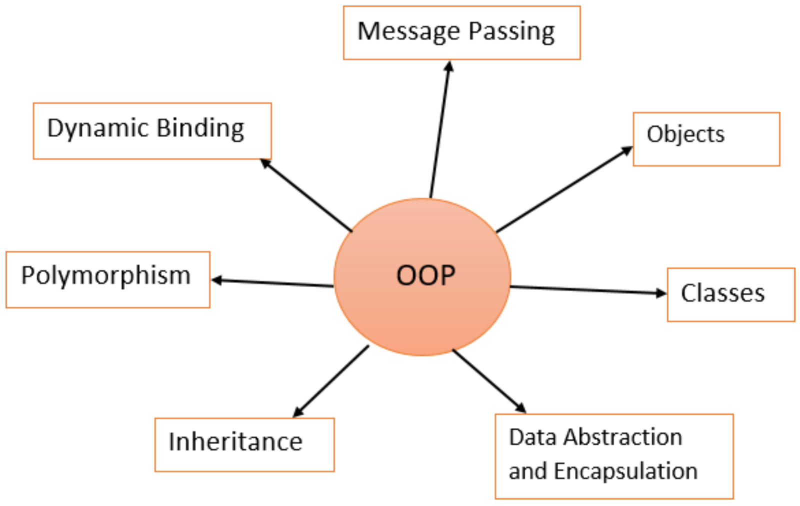
- To solve a large problem, break the problem into several pieces and work on each piece separately;
- To solve each piece, treat it as a new problem that can itself be broken down into smaller problems;
- Repeat the process with each new piece until each can be solved directly, without further decomposition.

- This approach is also called **top-down program design**.
- Large programs are divided into smaller programs known as functions.
- Data move openly around the system from function to function.
- Functions transform data from one form to another.



## Basic concepts of OOPs

It is necessary to understand some of the concepts used extensively in object-oriented programming. These include:



### Objects

- Object is collection of number of entities. 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 programhas to handle. They may also represent user-defined data such as vectors, time and lists.
- When a program is executed, the object interacts by sending messages to one another. Each object contains data and code to manipulate the data.
- Objects can interact without having known details of each other's data or code.

## Classes

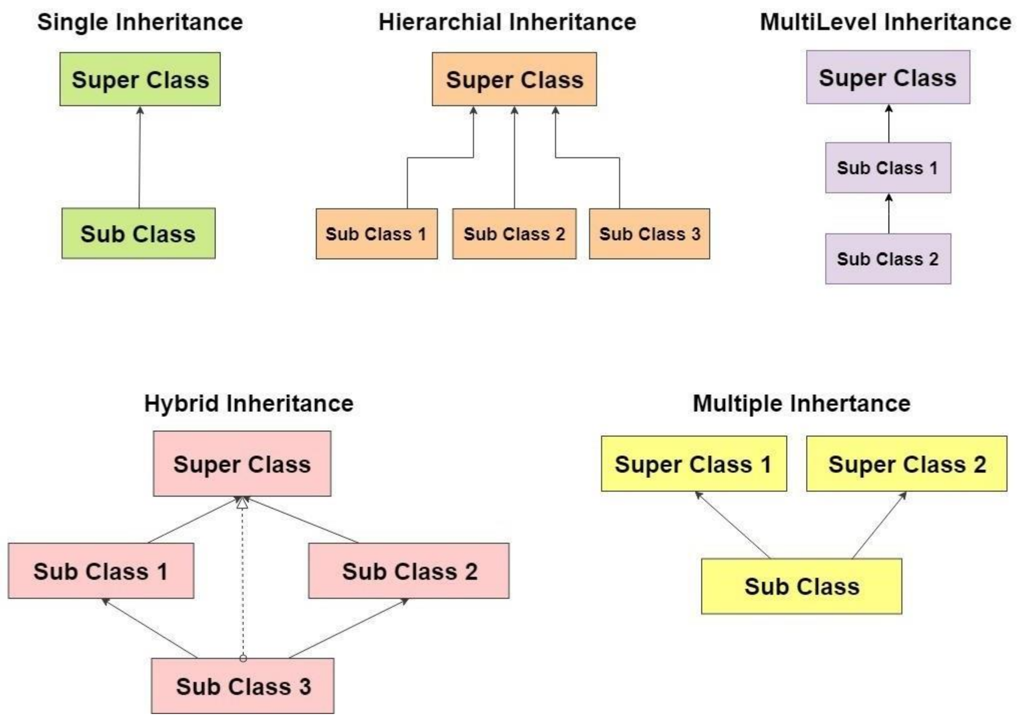
- Class is a collection of objects of similar type. Objects are variables of the type class.
- The entire set of data and code of an object can be made a user-defined data type with the help of class.
- Once a class has been defined, one can create any number of objects belonging to that class.
- Example: Apples, mangoes and oranges are the member of class fruit.
- `fruit mango`;
- If fruit has been defined as class, then the above statement will create an object **mango** belonging toclass **fruit**.
- Classes are user-defined data types.

## Data Abstraction and Encapsulation

- Combining data and functions into a single unit called class and the process is known as **Encapsulation**. Data encapsulation is the most striking feature of a class. The data is not accessible tothe outside world, and only those functions which are wrapped in the class on access it.
- **Abstraction** refers to the act of representing essential features without including the background details or explanations. Classes uses 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.

## Inheritance

- It is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of **hierarchical classification**.
- The concept of inheritance provides the idea of reusability means we can add additional features toan existing class without **Modifying** it.
- This is possible by driving a new class from the existing one. The new class will have the combined features of both the classes.
- Note that each sub-class defines only those features that are unique to it. Without the use of classification, each class would have to explicitly include all of its features.
- There is various type of inheritance available in OOPs, as shown in figure below:



## Polymorphism

- Polymorphism is another important OOP concept. The name **Polymorphism** is derived from combination of two Greek words i.e. **Poly** means **Many** and **Morphe** means **Form**. So basically, Polymorphism means ability to take more than one form.
- An operation may exhibit different behaviours in different instances. The behaviour depends uponthe types of data used in the operation.
- The process of making an operator to exhibit different behaviours in different instances is known as **operator overloading**.
- Using a single function name to perform different types of tasks is known as **function overloading**.

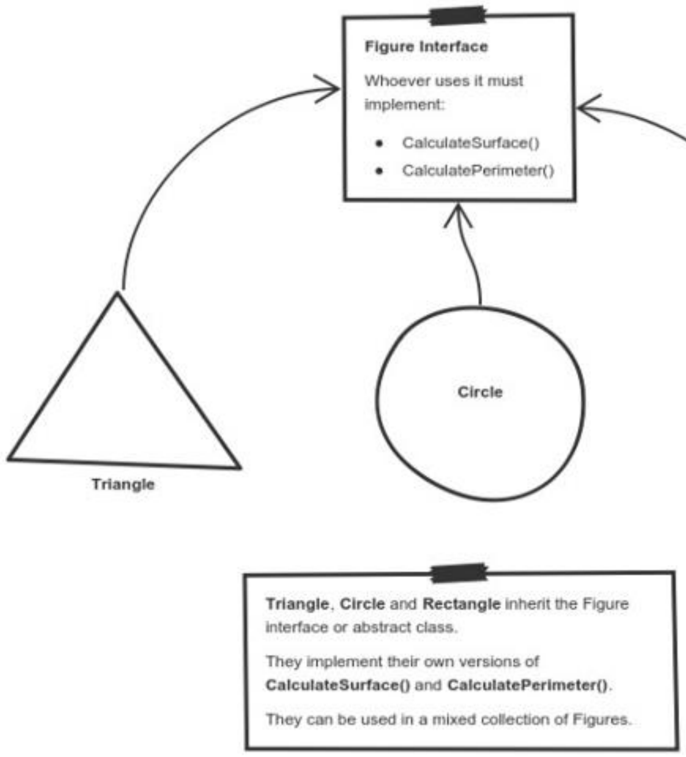


Figure Interface
Whoever uses it must implement:
CalculateSurface()
CalculatePerimeter()
Triangle
Circle
Rectangle
Triangle, Circle and Rectangle inherit the Figure interface or abstract class.
They implement their own versions of CalculateSurface() and CalculatePerimeter().
They can be used in a mixed collection of Figures.

**Fig. Polymorphism using different shapes**

**Dynamic Binding**

- Binding refers to the linking of a procedure call to code to be executed in response to the call. Dynamic Binding means the code associated with a given procedure call is not known until the time of the call at run-time.
- A function call associated with a polymorphic reference depends on the dynamic type of that reference.
- 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.