Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Object-Oriented Programming/5b18b5337becc0459de9bf50

TUTORIAL

Object-Oriented Programming

Topics

Object Oriented Programming (OOP) is the most recent concept among programming paradigms and still means different things to different people. OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions in object-oriented programs. The data 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. Objects, classes, data abstraction and encapsulation, Inheritance, polymorphism, message passing are some basic concepts of OOP Systems. To know about object oriented programming one must think out of box from procedural programming. Object oriented programming is programming around only one thing called object.

Object-Oriented Programming

Object Oriented Programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand. It attempts to eliminate some flaws encountered in the procedural approach. Let us take a distinguish between Object Oriented Programming & Procedure Oriented Programming.

```
Object Oriented Programming
                        Procedure Oriented Programming
```

```
Emphasis is on data as well as on procedures.
                        Emphasis is on procedures.
Programs are divided into what are known as classes.
                        Large programs are divided into smaller
programs known as functions.
Data is hidden and cannot be accessed by external functions.
                        Data move openly around the system from
function to function.
New data and functions can be added whenever necessary.
                        Function transforms data from one form to
another.
Follows Bottom-Up approach in program design.
                        Follows Top-Up approach in program design.
```

Structured design methods evolved to guide developers who were trying to build complex systems using algorithms as their fundamental building blocks. Similarly, object-oriented design methods have evolved to help developers who exploit the expressive power of object-based and object-oriented programming languages, using the class and object as basic building blocks. Object-oriented analysis and design represents an evolutionary development, not a revolutionary one; it does not break with advances from the past, but builds upon proven ones. Unfortunately, most programmers today are formally and informally trained only in the principles of structured design. Certainly, many good engineers have developed and deployed countless useful software systems using these techniques. However, there are limits to the amount of complexity we can handle using only algorithmic decomposition; thus we must turn to object-oriented decomposition.

Jargon OOP

Object: Objects are basic run time entities in an object oriented system. They may represent a person, a place, a table of data or any item that the program has to handle. It is an instance of a class. It can be uniquely identified by its name and it defines a state which is represented by the values of its attributes at a particular time.

Class: A class is a definition of objects of the same kind. In other words, a class is a blueprint, template, or prototype that defines and

describes the static attributes and dynamic behaviors common to all objects of the same kind. Once a class has been defined, we can create any number of objects belonging to that class. For example, we can define a class called "Student" and create three instances of the class "Student" for "Girdhar", "Gopal" and "Amit".

Information Hiding and Encapsulation: The wrapping up of data and functions into a single unit is known as encapsulation. The data is not accessible to the outside world, and only functions which are wrapped in the class can access it. This insulation of data from direct access by the program is called information hiding.

Abstraction: It refers to the act of representing essential features without including the background details. Classes use the concept of abstraction and therefore known as Abstract Data Types (ADT).

Inheritance: The concept of inheritance recognizes that often one defined class is really just a special case of another class, sharing a lot of common data fields and methods. It is the process by which objects of one class acquire the properties of the objects of another class. For example, Car, Motorcycle, and Airplane are all types of Vehicle. So if a Vehicle class were created that maintained all the state information common to all vehicle types, then a Car class

could be developed that is a specialization of Vehicle, dealing with only the special properties that make Cars different from other Vehicle types. In OOP, inheritance provides the idea of reusability i.e. we can add additional features to an existing class without modifying it.

Polymorphism: It means the ability to take more than one form. An operation may exhibit different behaviours in different instances. The process of making an operator to exhibit different behaviours in different instances is known as operator overloading. Similarly we can use a single function name to perform different types of tasks known as function overloading.

Message Passing: In object-orientation, there is a view of autonomous objects which communicate with each other by exchanging messages. Objects react when they receive messages by applying methods on themselves. A message for an object is a

request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired result.

Defining your own classes

To map a real life model to OOP one needs to create classes and objects. Different programming languages have different syntax for creating them but semantics are same behind it. In some languages classes had just a single main method. Whereas classes are needed for more sophisticated applications also. These classes typically do not have a main method. Instead, they have their own instance fields and methods. To build a complete program, you combine several classes, one of which has a main method. The simplest form for a class definition in C++ is

```
class ClassName
{
    public/private/protected:   // (One of the available Access
Specifiers)
        constructor1
        constructor2
        . . .
        method1
        method2
        . . .
        field1
        field2
        . . .
};
```

The simplest form for a class definition in Java is

```
class ClassName
{
        constructor1
        constructor2
        . . .
        method1
        method2
        . . .
        field1
        field2
        . . .
}
```

Different languages may define different syntactic rules to specify OOP elements.

Java

```
class ForDemo
{
  private int val1;
  void fillData()       // Function to set the value of a object
  {
    val1 = 18;
  }
  void printInfo()      // Function to access the value.
  {
    System.out.println("The value is : "+ val1);
  }
}

class Main{
  public static void main(String[] args)
  {
    ForDemo ob1 = new ForDemo();        // Create an object of class ForDemo
    ob1.fillData();
// Call functions defined for a object.
    ob1.printInfo();
  }
}
```

C++

```
#include<iostream>
using namespace std;
class ForDemo
{
  private:
  int val1;

  public:
  void fillData()       // Function to set the value of a object
  {
    val1 = 18;
  }
  void printInfo()      // Function to access the value.
```

```
14    {
15      cout<<"The value is : "<<val1;
16    }
17  };
18
19  int main()
20  {
21    ForDemo ob1;  // Create an object of class ForDemo
22    ob1.fillData();
    // Call functions defined for a object.
23    ob1.printInfo();
24    return 0;
25  }
```