Introduction to OOPs

Languages like Pascal, C, FORTRAN, and COBOL are called procedure oriented programming languages. Since in these languages, a programmer uses procedures or functions to perform a task. When the programmer wants to write a program, he will first divide the task into separate sub tasks, each of which is expressed as functions/ procedures. This approach is called procedure oriented approach.

The languages like C++ and Java use classes and object in their programs and are called Object Oriented Programming languages. The main task is divided into several modules and these are represented as classes. Each class can perform some tasks for which several methods are written in a class. This approach is called Object Oriented approach.

Difference between Procedure Oriented Programming and OOP:

Procedure Oriented Programming	Object Oriented Programming
1. Main program is divided into small parts depending on the functions.	1. Main program is divided into small object depending on the problem.
2. The Different parts of the program connect with each other by parameter passing & using operating system.	2. Functions of object linked with object using message passing.
3. Every function contains different data.	3. Data & functions of each individual object act like a single unit.
4. Functions get more importance than data in program.	4. Data gets more importance than functions in program.
5. Most of the functions use global data.	5. Each object controls its own data.
6. Same data may be transfer from one function to another	6. Data does not possible transfer from one object to another.
7. There is no perfect way for data hiding.	7. Data hiding possible in OOP which prevent illegal access of function from outside of it. This is one of the best advantages of OOP also.
8. Functions communicate with other functions maintaining as usual rules.	8. One object link with other using the message passing.
9. More data or functions can not be added with program if necessary. For this purpose full program need to be change.	9. More data or functions can be added with program if necessary. For this purpose full program need not to be change.
10. To add new data in program user should be ensure that function allows it.	10. Message passing ensure the permission of accessing member of an object from other object.
11. Top down process is followed for program design.	11. Bottom up process is followed for program design.
12. Example: Pascal, Fortran	12. Example: C++, Java.

Features of OOP:

- **Class:** In object-oriented programming, a class is a programming language construct that is used as a blueprint to create objects. This blueprint includes attributes and methods that the created objects all share. Usually, a class represents a person, place, or thing - it is an abstraction of a concept within a computer program. Fundamentally, it encapsulates the state and behavior of that which it conceptually represents. It encapsulates state through data placeholders called member variables; it encapsulates behavior through reusable code called methods

General form of a class:

```
class class_name
{
        Properties (variables);
        Actions (methods);
}
```

e.g.:

```
class Student
{       //properties -- variables
        int rollNo;
        String name;
        //methods -- actions
        void display ()
        {
                System.out.println ("Student Roll Number is: " + rollNo);
                System.out.println ("Student Name is: " + name);
        }
}
```

Variables inside a class are called as instance variables.

Variables inside a method are called as method variables.

- **Object:** An Object is a real time entity. An object is an instance of a class. Instance means physically happening. An object will have some properties and it can perform some actions.

  Object contains variables and methods. The objects which exhibit similar properties and actions are grouped under one class. “To give a real world analogy, a house is constructed according to a specification. Here, the specification is a blueprint that represents a class, and the constructed house represents the object”.

- To access the properties and methods of a class, we must declare a variable of that class type. This variable does not define an object. Instead, it is simply a variable that can refer to an object.
- We must acquire an actual, physical copy of the object and assign it to that variable. We can do this using **new** operator. The new operator dynamically allocates memory for an object and returns a reference to it. This reference is, more or less, the address in memory of the object allocated by new. This reference is then stored in the variable. Thus, in Java, all class objects must be dynamically allocated

General form of an Object:

```
Class_name variable_name;            // declare reference to object
variable_name = new Class_name ( );          // allocate an object
```

e.g.:

```
Student s;      // s is reference variable
s = new Student ();            // allocate an object to reference variable s
```

The above two steps can be combined and rewritten in a single statement as:

Student s = new Student ();

Now we can access the properties and methods of a class by using object with dot operator as:

s.rollNo, s.name, s.display ()

- **Encapsulation:** Wrapping up of data (variables) and methods into single unit is called Encapsulation. Class is an example for encapsulation. Encapsulation can be described as a protective barrier that prevents the code and data being randomly accessed by other code defined outside the class. Encapsulation is the technique of making the fields in a class private and providing access to the fields via methods. If a field is declared private, it cannot be accessed by anyone outside the class

```
class Student
{
        private int rollNo;
        private String name;
        //methods -- actions
        void display ()
        {
                System.out.println ("Student Roll Number is: " + rollNo);
                System.out.println ("Student Name is: " + name);
        }
}
```

- **Abstraction:** Providing the essential features without its inner details is called abstraction (or) hiding internal implementation is called Abstraction. We can enhance the internal implementation without effecting outside world. Abstraction provides security. A class contains lot of data and the user does not need the entire data. The advantage of abstraction is that every user will get his own view of the data according to his requirements and will not get confused with unnecessary data. A bank clerk should see the customer details like account number, name and balance amount in the account. He should not be entitled to see the sensitive data like the staff salaries, profit or loss of the bank etc. So such data can be abstracted from the clerks view.

e.g.:

```
class Bank
{       private int accno;
        private String name;
        private float balance;
        private float profit;
        private float loan;
        void display_to_clerk ()
        {
                System.out.println ("Accno = " + accno);
                System.out.println ("Name = " + name);
                System.out.println ("Balance = " + balance);
        }
}
```

In the preceding class, inspite of several data items, the display_to_clerk () method is able to access and display only the accno, name and balance values. It cannot access profit and loan of the customer. This means the profit and loan data is hidden from the view of the bank clerk

- **Inheritance:** Acquiring the properties from one class to another class is called inheritance (or) producing new class from already existing class is called inheritance. Reusability of code is main advantage of inheritance. In Java inheritance is achieved by using extends keyword. The properties with access specifier private cannot be inherited.

```
e.g.:   class Parent
        {
            String parentName;
            String familyName;
        }
        class Child extends Parent
        {
                String childName;
                int childAge;
                void printMyName()
                {
                        System.out.println ("My name is"+childName+" "+familyName);
                }
        }
```

In the above example, the child has inherited its family name from the parent class just by inheriting the class

- **Polymorphism:** The word polymorphism came from two Greek words 'poly' means 'many' and 'morphos' means 'forms'. Thus, polymorphism represents the ability to assume several different forms. The ability to define more than one function with the same name is called Polymorphism

```
e.g.:   int add (int a, int b)
        float add (float a, int b)
        float add (int a , float b)
        void add (float a)
        int add (int a)
```

- **Message Passing:** Calling a method in a class is called message passing. We can call methods of a class by using object with dot operator as:

```
object_name.method_name ();
e.g.:  s.display ();   ob.add (2, 5);   ob.printMyName ();
```

Program 1: Write a program to display details of student using class and object.

//Program to display the details of a student using class and object

```
class Student
{       int rollNo;             //properties -- variables
        String name;
        void display ()         //method -- action
        {       System.out.println ("Student Roll Number is: " + rollNo);
                System.out.println ("Student Name is: " + name);
        }
}
class StudentDemo
{       public static void main(String args[])
        {
```

```
                //create an object to Student class
                Student s = new Student ();
                //call display () method inside Student class using object s
                s.display ();
        }
}
//Program to display the details of a student using class and object
class Student
{       int rollNo;
        String name;
        void display ()
        {
                System.out.println ("Student Roll Number is: " + rollNo);
                System.out.print ("Student Name is: " + name);
        }
}
class StudentDemo
{       public static void main(String args[])
        {       Student s1 = new Student ();
                System.out.println ("First Student Details : " );
                s1.rollNo = 101;
                s1.name = "Suresh";
                s1.display ();
                Student s2 = new Student ();
                System.out.println ("Second Student Details : " );
                s2.rollNo = 102;
                s2.name = "Ramesh";
                s2.display ();
        }
}
```

- **Access Specifiers:** An access specifier is a key word that represents how to access a member of a class. There are four access specifiers in java.
  - **private:** private members of a class are not available outside the class.
  - **public:** public members of a class are available anywhere outside the class.
  - **protected:** protected members are available outside the class.
  - **default:** if no access specifier is used then default specifier is used by java compiler. Default members are available outside the class.
- **Constructor:**
  - A constructor is similar to a method that initializes the instance variables of a class.
  - A constructor name and classname must be same.

- A constructor may have or may not have parameters. Parameters are local variables to receive data.
- A constructor without any parameters is called default constructor.

e.g.
```
class Student
{       int rollNo;
        String name;
        Student ()
        {       rollNo = 101;
                name = "Kiran";
        }
}
```

- A constructor with one or more parameters is called parameterized constructor.

e.g.
```
class Student
{       int rollNo;
        String name;
        Student (int r, String n)
        {       rollNo = r;
                name = n;
        }
}
```

- A constructor does not return any value, not even void.

- A constructor is called and executed at the time of creating an object.
- A constructor is called only once per object.
- Default constructor is used to initialize every object with same data where as parameterized constructor is used to initialize each object with different data.
- If no constructor is written in a class then java compiler will provide default values

Write a program to initialize student details using default constructor and display the same.

```
//Program to initialize student details using default constructor and displaying the same.
class Student
{       int rollNo;
        String name;
        Student ()
        {       rollNo = 101;
                name = "Suresh";
        }
        void display ()
        {       System.out.println ("Student Roll Number is: " + rollNo);
                System.out.println ("Student Name is: " + name);
        }
}
class StudentDemo
{       public static void main(String args[])
        {       Student s1 = new Student ();
                System.out.println ("s1 object contains: ");
                s1.display ();
                Student s2 = new Student ();
                System.out.println ("s2 object contains: ");
                s2.display ();
        }
}
```

The keyword 'this': There will be situations where a method wants to refer to the object which invoked it. To perform this we use 'this' keyword. There are no restrictions to use 'this' keyword we can use this inside any method for referring the current object. This keyword is always a reference to the object on which the method was invoked. We can use 'this' keyword wherever a reference to an object of the current class type is permitted. 'this' is a key word that refers to present class object. It refers to

- Present class instance variables
- Present class methods.
- Present class constructor.

- **Garbage Collection:** Generally memory is allocated to objects by using 'new' operator and deleting an allocated memory is uncommon. This deletion of memory is supported by delete operator in C++ but this deletion of allocated memory works automatically in Java. This automatic deletion of already allocated but unused memory is called as garbage collection. This operation of garbage collection is accomplished by a method named "gc ()". This method is used for garbage collection.