OBJECT ORIENTED PROGRAMMING (OOP'S CONCEPT)

- Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts.
    - Inheritance
    - Abstraction
    - Polymorphism
    - Encapsulation
- Object-oriented programming has several advantages over procedural programming:
    - OOP is faster and easier to execute.OOP provides a clear structure for the programs.
    - OOP helps to keep the Java code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug.
    - OOP makes it possible to create full reusable applications with less code and shorter development time.

INHERITANCE:-

When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability and extensibility. It is used to achieve runtime polymorphism.

SYNTAX:

```
Subclass-name extends Superclass-name
{
        //methods and fields
}
```

EXAMPLE:

```
class Employee{
        float salary=40000;
}
class Programmer extends Employee
{
    int bonus=10000;
   public static void main(String args[])
    {
        Programmer p=new Programmer();
```

```
        System.out.println("Programmer salary is:"+p.salary);
        System.out.println("Bonus of Programmer is:"+p.bonus);
    }
}
```

POLYMORPHISM:-

- If one task is performed in different ways, it is known as polymorphism.
- For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.
- In Java, we use method overloading and method overriding to achieve polymorphism.
- Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

EXAMPLE:

```
class Bike
{
   void run()
    {
       System.out.println("running");}
       }
  class Splendor extends Bike
  {
  void run(){
      System.out.println("running safely with 60km");
   }
}
 public static void main(String args[])
{
    Bike b = new Splendor();//upcasting
     b.run();
 }
}
```

ABSTRACTION:-

- Abstraction is hiding the irrelevant information to end user only necessary details should be provided.
- Hiding internal details and showing functionality is known as abstraction.
- For example phone call, we don't know the internal processing.
- In Java, we use abstract class and interface to achieve abstraction.

EXAMPLE:

```
abstract class Bike

{

  abstract void run();

}

class Honda4 extends Bike

{

void run(){

System.out.println("running safely");

}

public static void main(String args[])

{

 Bike obj = new Honda4();

                    obj.run();

}

}
```

ENCAPSULATION:-

- Encapsulation is any variable/function that is available inside a class should not be accessible without that specific class.
- Binding (or wrapping) code and data together into a single unit are known as encapsulation.
- For example, a capsule, it is wrapped with different medicines.
- A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

EXAMPLE:

```
public class Student
{
  private String name;
  public String getName()
  {
     return name;
   }
 public void setName(String name)
 {
     this.name=name
  }
}
```

ABSTRACT METHOD:

- An abstract method is a method that is declared without implementation.
- An abstract class may or maynot havr all abstract methods.some of them can be concrete methods
- A method defined abstract must always be redefined in the subclass,thus making overriding compulsory or either make the subclass itself abstract.

Example:

```
Public abstract int myMethod(int n1,int n2);
```

ABSTRACT CLASS:

- A class that is declared using “abstract” keyword is known as abstract class.
- It can have abstract methods(methods without body) as well as concrete methods(regular methods with body).
- A normal class(non-abstract class) cannot have abstract methods
- Abstract class is a restricted class that cannot be used to create objects[to access it, it must be inherited from another class].
- An abstract class can have both abstract and regular methods.

Example:

```
    abstract class Animal
{
Public abstract void animalSound();
Public void sleep()
{
     System.out.println(“Tiger”);
  }
}
```