Object Oriented Programming (OOPs) Concept in Java

Java - What is OOP?

OOP stands for **Object-Oriented Programming**.

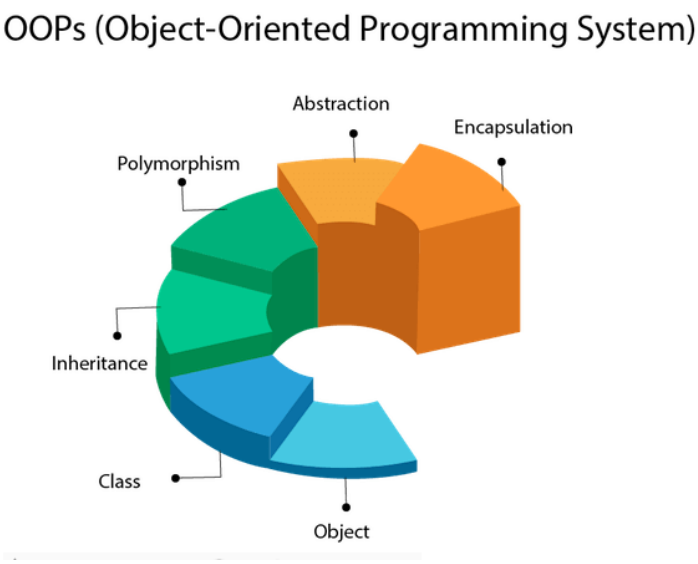
Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.

Object means a real-world entity such as a pen, chair, table, computer, watch, etc.

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects.

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.



Java - What are Classes and Objects?

A **class** is defined as a collection of objects. You can also think of a class as a blueprint from which you can create an individual object.

To create a class, we use the keyword class.

Syntax of a class in Java:

```
class ClassName {
  // fields
  // methods
}
```

An **object** is an entity in the real world that can be distinctly identified. Objects have states and behaviors. In other words, they consist of methods and properties to make a particular type of data useful.

An object consists of:

- **A unique identity:** Each object has a unique identity, even if the state is identical to that of another object.
- **State/Properties/Attributes:** State tells us how the object looks or what properties it has.
- **Behavior:** Behavior tells us what the object does.

examples of object states and behaviors in Java:

Let's look at some real-life examples of the states and behaviors that objects can have.

Example 1:

- Object: car.
- State: color, brand, weight, model.
- Behavior: break, accelerate, turn, change gears.

Example 2:

- Object: house.
- State: address, color, location.
- Behavior: open door, close door, open blinds

To create an object of Main, specify the class name, followed by the object name, and use the keyword new:

Syntax of an object in Java:

```
public class Number {

int y = 10;

public static void main(String[] args) {

Number myObj = new Number();

System.out.println(myObj.y);

}

}
```

Multiple Objects

You can create multiple objects of one class:

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj1 = new Main();  // Object 1
    Main myObj2 = new Main();  // Object 2
    System.out.println(myObj1.x);
    System.out.println(myObj2.x);
  }
}
```

Java Class Attributes

We used the term "variable" for x in the example (as shown below). It is actually an **attribute** of the class. Or you could say that class attributes are variables within a class:

Create a class called "Main" with two attributes: x and y:

```
public class Main {
  int x = 5;
  int y = 3;
}
```

Accessing Attributes

You can access attributes by creating an object of the class, and by using the dot syntax (.):

The following example will create an object of the Main class, with the name myObj. We use the x attribute on the object to print its value:

Create an object called "myObj" and print the value of x:

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Modify Attributes

You can also modify attribute values:

```
public class Main {
  int x = 10;

  public static void main(String[] args) {
    Main myObj = new Main();
    myObj.x = 25; // x is now 25
    System.out.println(myObj.x);
  }
}
```

Multiple Objects

If you create multiple objects of one class, you can change the attribute values in one object, without affecting the attribute values in the other:

```
public class Main {
  int x = 5;

  public static void main(String[] args) {
    Main myObj1 = new Main();  // Object 1
    Main myObj2 = new Main();  // Object 2
    myObj2.x = 25;
    System.out.println(myObj1.x);  // Outputs 5
    System.out.println(myObj2.x);  // Outputs 25
  }
}
```

Multiple Attributes

You can specify as many attributes as you want:

```
public class Main {
  String fname = "John";
  String lname = "Doe";
  int age = 24;

  public static void main(String[] args) {
    Main myObj = new Main();
    System.out.println("Name: " + myObj.fname + " " + myObj.lname);
    System.out.println("Age: " + myObj.age);
  }
}
```