JAVA

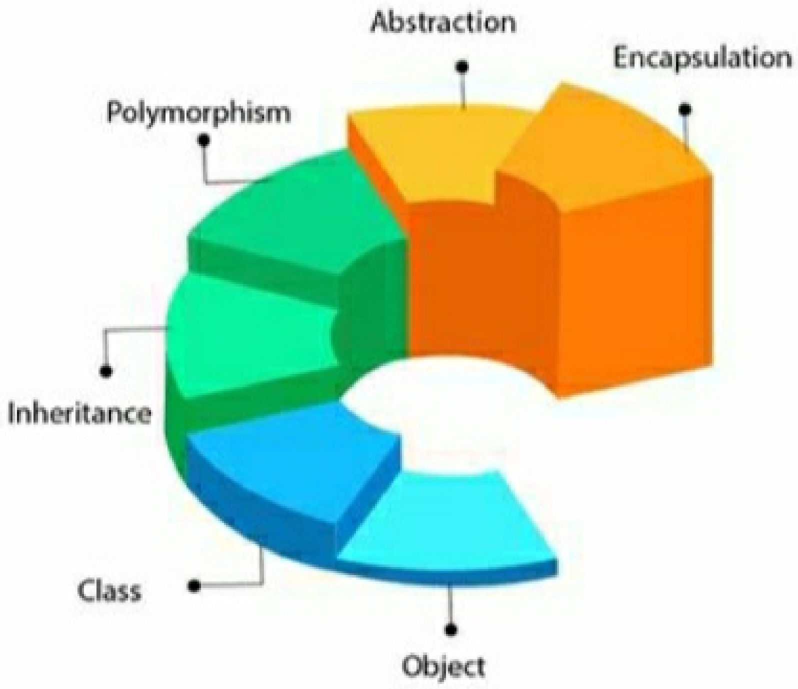
OOPS

OOPS is abbreviated as Object Oriented Programming system in which programs are considered as a collection of objects. Each objectis nothing but an instanceof a class.

Why do we use object-oriented programming?

- OOPs, make the development and maintenance of projects easier.
- OOPs provide the feature of data hiding that is good for security concern.
- We can provide the solution to real-world problems if we are using object-oriented programming.

OOPs Concepts :



Classes in Java

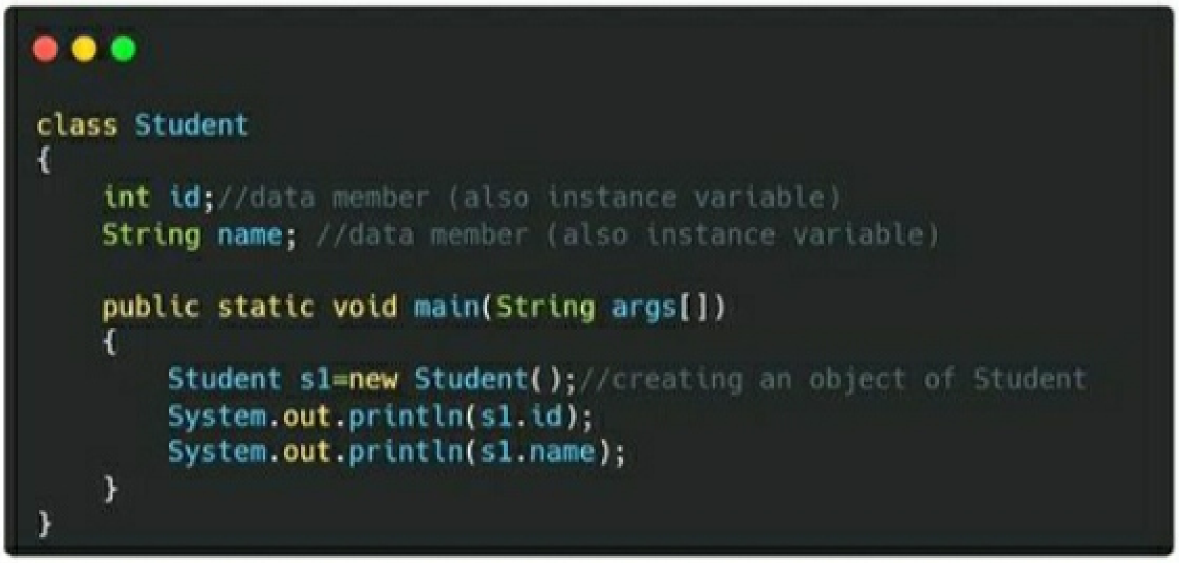
A class is simply a representation of a type of object.It is the blueprint/plan/template that describes the details of an object.

Syntax

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

Here, **fields** (variables) and **methods** represent the **state** and **behavior** of the object respectively.

- fields are used to store data
- methods are used to perform some operations



```
class Student
{
    int id;//data member (also instance variable)
    String name; //data member (also instance variable)

    public static void main(String args[])
    {
        Student s1=new Student();//creating an object of Student
        System.out.println(s1.id);
        System.out.println(s1.name);
    }
}
```

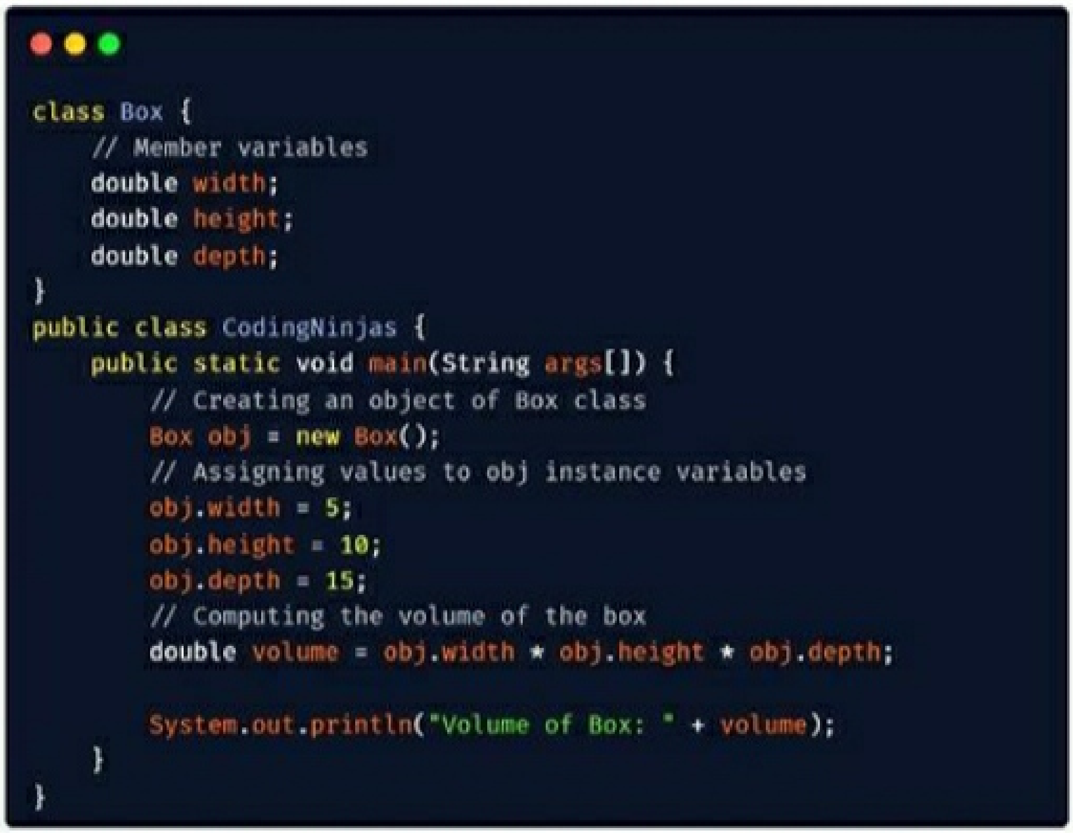
Object in Java

An object is an instanceof a class. It has its own state, behavior, and identity.

- **State:** It is represented by attributes of an object.
- **Behavior:** It is represented by methods of an object.
- **Identity:** It gives a unique name to an object

```
Dog obj = new Dog( );   // Creating an Object
```

we can create an object by using the new keyword. The new keyword is used to allocate memory for an object dynamically and return a reference to it.



```java
class Box {
    // Member variables
    double width;
    double height;
    double depth;
}
public class CodingNinjas {
    public static void main(String args[]) {
        // Creating an object of Box class
        Box obj = new Box();
        // Assigning values to obj instance variables
        obj.width = 5;
        obj.height = 10;
        obj.depth = 15;
        // Computing the volume of the box
        double volume = obj.width * obj.height * obj.depth;

        System.out.println("Volume of Box: " + volume);
    }
}
```

Access Modifiers

Access modifiers define the accessibility of a method, variable, constructor, or class. There are **four types** of access modifiers

- **Default:** declarations are visible only within the package (package private).
- **Private:** declarations are visible within the class only.
- **Protected:** declarations are visible within the package or all subclasses.
- **Public:** declarations are visible everywhere.

Access Modifier	Within class	Within package	Outside the package	Outside package by subclass
Private	**YES**	**NO**	**NO**	**NO**
Default	**YES**	**YES**	**NO**	**NO**
Protected	**YES**	**YES**	**NO**	**YES**
Public	**YES**	**YES**	**YES**	**YES**

Default Access Modifier

```
package defaultPackage;
class Logger {
   void message( ){
      System.out.println("This is a message");
   }
}
```

Private Access Modifier

```
class Data {
   private String name;

   // getter method
   public String getName() {
      return this.name;
   }
   // setter method
   public void setName(String name) {
      this.name= name;
   }
}
public class Main {
   public static void main(String[] main){
      Data d = new Data();

      // access the private variable using the getter and
          setter
      d.setName("iamrupnath");
      System.out.println(d.getName());
   }
}
```

Protected Access Modifier

```
class Animal {

   // protected method
   protected void display() {
      System.out.println("I am an animal");
   }
}

class Dog extends Animal {

   public static void main(String[] args) {

      // create an object of Dog class
      Dog dog = new Dog();
       // access protected method
      dog.display();
   }
}
```

Output: I am an animal

Public Access Modifier

```java
// Animal.java file
// public class
public class Animal {
   // public variable
   public int legCount;
       // public method
   public void display() {
      System.out.println("I am an animal.");
      System.out.println("I have " + legCount + " legs.");
   }
}
// Main.java
public class Main {
     public static void main( String[] args ) {
        // accessing the public class
      Animal animal = new Animal();

      // accessing the public variable
      animal.legCount = 4;
      // accessing the public method
      animal.display();
   }
}
```

Output:

```
I am an animal.
I have 4 legs.
```

Jayesh Deshmukh

Follow

Like Comment Share Save