|  |
| --- |
| **Java Object Oriented Programming** |

**Object Oriented Programming**

Object Oriented Programming is a way to model real world objects as Software Objects which contain both data and code. Object Oriented Programming is also known as Class based programming as everything over here happens using classes and objects.

**Objects**

Object is a real world entity which has state and behaviour. For example, let us take an example of computer which is a real world object. Its states are as follows

* The amount of RAM it has
* The amount of Disk space it has
* The number of cores its CPU has

Behaviour for a computer can be

* Booting up
* Shutting down
* Beeping or outputting some sound
* Putting some output on the screen

**Classes**

A class can also be defined as a blueprint or template from which we can create an individual object. It contains the state (variables) and behaviour (methods ) of the instance objects we can create from it

|  |
| --- |
| class car {  */\* States \*/*      String brand;      String model;      String color;      float engineCapacity;      float fuelTankCapacity;      int seats;  */\* Behaviours \*/*  *public* void unLockDoors() {          System.out.println("Doors Are Unlocked!");      }  *public* void ignition() {          System.out.println("Cranking the self start!");      }  *public* void turnOnHeadLights() {          System.out.println("Headlights Are On!");      }  *public* void turnOnBlinkers() {          System.out.println("Blinkers Are On!");      }  }  *public* *class* classDemo {  } |

Classes can be seen as powerful user defined datatypes which can be extremely customized as per the needs of the user.

**Methods**

Methods in java is a collection of statements that is used for performing a specific task. Methods are used to avoid code redundancy.

A method is written as per the following convention

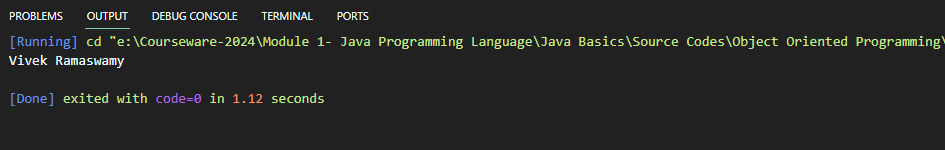
|  |
| --- |
| <access modifier> <static (optional)> <return type> <method name>(arg1, arg2,….. arg n); |

Eg:-

public static void main(String[ ] args)

|  |
| --- |
| class students {      String firstName = "Vivek";      String lastName = "Ramaswamy";      int roll = 24;      int std = 5;      char section = 'A';  */\* Methods \*/*  *public* String getStudentFullName() {  *return* firstName + " " + lastName;      }  }  *public* *class* classDemo1 {  *public* *static* void main(String[] *args*) {          students s1 = *new* students();          System.out.println(s1.getStudentFullName());      }  } |

Output



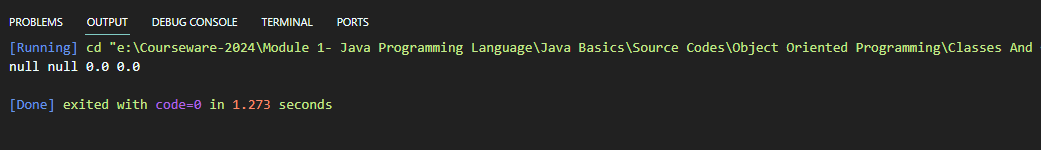
**Creating Objects From Class**

|  |
| --- |
| class car {  */\* States \*/*      String brand;      String model;      String color;      float engineCapacity;      float fuelTankCapacity;      int seats;  */\* Behaviours \*/*  *public* void unLockDoors() {          System.out.println("Doors Are Unlocked!");      }  *public* void ignition() {          System.out.println("Cranking the self start!");      }  *public* void turnOnHeadLights() {          System.out.println("Headlights Are On!");      }  *public* void turnOnBlinkers() {          System.out.println("Blinkers Are On!");      }  }  *public* *class* classDemo {  *public* *static* void main(String[] *args*) {          car c1 = *new* car();      }  } |

If we try to print out the variables of the newly instantiated car object, we will see the following

|  |
| --- |
| class car {  */\* States \*/*      String brand;      String model;      String color;      float engineCapacity;      float fuelTankCapacity;      int seats;  */\* Behaviours \*/*  *public* void unLockDoors() {          System.out.println("Doors Are Unlocked!");      }  *public* void ignition() {          System.out.println("Cranking the self start!");      }  *public* void turnOnHeadLights() {          System.out.println("Headlights Are On!");      }  *public* void turnOnBlinkers() {          System.out.println("Blinkers Are On!");      }  }  *public* *class* classDemo {  *public* *static* void main(String[] *args*) {          car c1 = *new* car();          System.out.println(c1.brand + " " + c1.color + " " + c1.engineCapacity + " " + c1.fuelTankCapacity);      }  } |

Output



This happens as because Java puts default values in these attributes depending on their datatypes as we have not instantiated them during the creation of the objects

To initialize object attributes we have two ways,

Initializing Object Attributes

Using Getters And Setters utes

Using Constructors

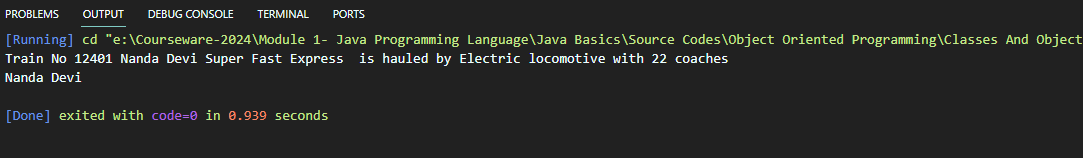
**Using Getters And Setters**

Object attributes should always be made private, so that they cannot be accessed directly from outside. They should be accessed using getter and setter methods only

Setter methods are used to set the attribute values, while Getter methods are used to retrieve the attribute values

|  |
| --- |
| class train {  *private* String trainName;  *private* int trainNumber;  *private* String trainType;  *private* int noOfCoaches;  *private* String locomotiveType;  */\* Setters \*/*  *public* void setTrainNumber(int *train\_number*) {          trainNumber = *train\_number*;      }  *public* void setTrainName(String *train\_name*) {          trainName = *train\_name*;      }  *public* void setTrainType(String *train\_type*) {          trainType = *train\_type*;      }  *public* void setNoOfCoaches(int *coaches*) {          noOfCoaches = *coaches*;      }  *public* void setLocomotiveType(String *locomotive\_type*) {          locomotiveType = *locomotive\_type*;      }  */\* Getters \*/*  *public* String getTrainInformation() {  *return* "Train No " + trainNumber + " " + trainName + " " + trainType + " " + " is hauled by " + locomotiveType                  + " locomotive with " + noOfCoaches + " coaches";      }  *public* String getTrainName() {  *return* trainName;      }  }  *public* *class* classDemo2 {  *public* *static* void main(String[] *args*) {          train t1 = *new* train();          t1.setTrainType("Super Fast Express");          t1.setTrainNumber(12401);          t1.setTrainName("Nanda Devi");          t1.setLocomotiveType("Electric");          t1.setNoOfCoaches(22);          System.out.println(t1.getTrainInformation());          System.out.println(t1.getTrainName());      }  } |

Output



**Using Constructors**

In Java, a Constructor is a block of codes similar to the method. It is called when an instance of the class is created. It is a special type of method that is used to initialize the object. Every time an object is created using the new() keyword, at least one constructor is called.

Constructors are mainly of three types

Types Of Constructors

Default Constructors

User Defined Constructors

Parameterized Constructors

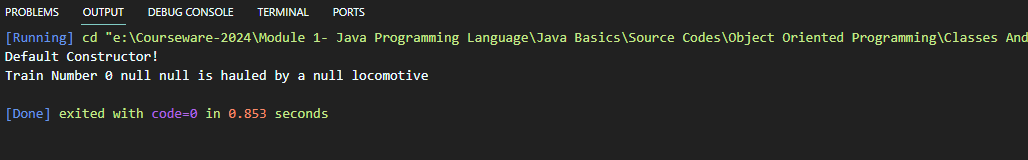
Non Parameterized Constructors

Copy Constructors

Default Constructors are invoked whenever an object is created. It initializes the Object attributes with their respective default values with respect to their datatypes

|  |
| --- |
| class train {  *private* int trainNumber;  *private* String trainName;  *private* String trainType;  *private* String locomotiveType;  *public* train() {          System.out.println("Default Constructor!");      }  *public* String getDescription() {  *return* "Train Number " + trainNumber + " " + trainName + " " + trainType + " is hauled by a " + locomotiveType                  + " locomotive";      }  }  *public* *class* classDemo3 {  *public* *static* void main(String[] *args*) {          train t2 = *new* train();          System.out.println(t2.getDescription());      }  } |

Output

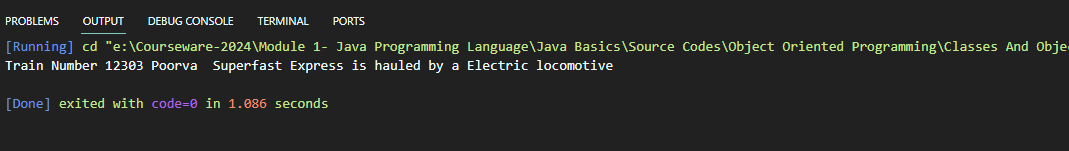


User defined constructors are those which are defined by the users. If the constructor takes in a value an argument, then it is a parameterized constructor. If it doesn’t take any value then it is a non parameterized constructor

**Non Parameterized Constructor**

|  |
| --- |
| class train {  *private* int trainNumber;  *private* String trainName;  *private* String trainType;  *private* String locomotiveType;  *public* train() {          trainNumber = 12303;          trainName = "Poorva ";          trainType = "Superfast Express";          locomotiveType = "Electric";      }  *public* String getDescription() {  *return* "Train Number " + trainNumber + " " + trainName + " " + trainType + " is hauled by a " + locomotiveType                  + " locomotive";      }  }  *public* *class* classDemo3 {  *public* *static* void main(String[] *args*) {          train t2 = *new* train();          System.out.println(t2.getDescription());      }  } |

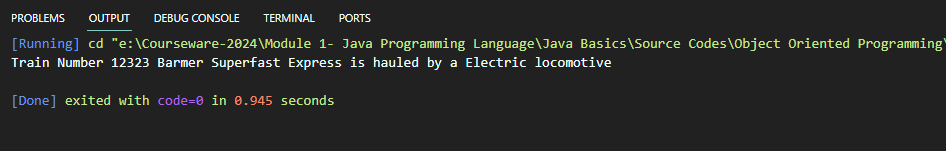
Output



**Parameterized Constructor**

|  |
| --- |
| class train {  *private* int trainNumber;  *private* String trainName;  *private* String trainType;  *private* String locomotiveType;  *// public train() {*  *// // System.out.println("Default Constructor!");*  *// trainNumber = 12303;*  *// trainName = "Poorva ";*  *// trainType = "Superfast Express";*  *// locomotiveType = "Electric";*  *// }*  *public* train(String *train\_name*, int *train\_number*, String *train\_type*, String *locomotive\_type*) {          trainName = *train\_name*;          trainNumber = *train\_number*;          trainType = *train\_type*;          locomotiveType = *locomotive\_type*;      }  *public* String getDescription() {  *return* "Train Number " + trainNumber + " " + trainName + " " + trainType + " is hauled by a " + locomotiveType                  + " locomotive";      }  }  *public* *class* classDemo3 {  *public* *static* void main(String[] *args*) {          train t2 = *new* train("Barmer", 12323, "Superfast Express", "Electric");          System.out.println(t2.getDescription());      }  } |

Output

****