BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on Defining a class, adding variables and methods, creating objects, accessing data members in Java.

Classes in Java:-

A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A Class is a user defined data-type which has data members and member functions.

Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class.

A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

Class declaration in Java:-

To create a class, use the keyword **class.**

Syntax to declare a class:

```
class class_name
{
    field;
    method;
}
```

Example of class declarations:-

```
public class Dog
{
   String breed;
   int age;
   String color;
   void barking()
   {
   }
   void hungry()
   {
   }
```

```
  void sleeping()
 {
  }
}
```

Variables within a class:-

A class can contain any of the following variable types. Variables within a class is also known as class attributes.

Local variables:–

Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.

Instance variables:–

Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

Class variables:–

Class variables are variables declared within a class, outside any method, with the **static** keyword.

Method within class:-

Methods within a class is used to expose the behavior of

an object.

Object in Java:-

An object is an instance of a class. As we know that a class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Object Definitions:

- ➤ An object is a real-world entity.
- ➤ An object is a runtime entity.
- ➤ The object is an entity which has state and behavior.
- ➤ The object is an instance of a class.

Creating an Object

As we know that class provides the blueprints for objects. So basically, an object is created from a class. In Java, the **new** keyword is used to create new objects. The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

Syntax:-

```
class_name object_name=new class_name();
```

Example-1: main within the class:-

```
class Student
{
 int id;  //field or data member or instance variable
```

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

Output:-

0

Null

Example-2: main outside the class.

```
class Student
{
 int id;
 String name;
}
class TestStudent1
{
```

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

Output:-

0

Null

Object initialization:-

There are 3 ways to initialize object in Java.

- By reference variable
- By method
- By constructor

Initialization through reference:-

Initializing an object means storing data into the object. In this example we are going to initialize the object through a reference variable.

Example-1:-

```
class Student
{
 int id;
 String name;
}
class TestStudent2
{
 public static void main(String args[])
{
  Student s1=new Student();
  s1.id=101;
  s1.name="Ram";
 //printing members with a white space
 System.out.println(s1.id+" "+s1.name);
 }
}
```

Output:-

101 Ram

Example-2: We can also create multiple objects and store information in it through reference variable.

```
class Student
{
 int id;
 String name;
}
class TestStudent3
{
 public static void main(String args[])
{
  //Creating objects
  Student s1=new Student();
  Student s2=new Student();
  //Initializing objects
  s1.id=101;
  s1.name="Ram";
  s2.id=102;
  s2.name="Shyam";
  //Printing data
  System.out.println(s1.id+" "+s1.name);
```

```
  System.out.println(s2.id+" "+s2.name);
 }
}
```

Output:-

101 Ram

102 Shyam

Initialization through method:-

In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

Example:-

```
class Student
{
 int rollno;
 String name;
 void insertRecord(int r, String n)
{
  rollno=r;
  name=n;
 }
```

```
 void displayInformation()
{
System.out.println(rollno+" "+name);
}
}
class TestStudent4
{
 public static void main(String args[])
{
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

Output:-

```
111 Karan
222 Aryan
```

Initialization through a constructor:-

We will learn about constructors in Java later.

Some more Object and Class Example:-

Example-1

In this example we are maintaining records of employees.

```
class Employee
{
    int id;
    String name;
    float salary;
    void insert(int i, String n, float s)
    {
        id=i;
        name=n;
        salary=s;
    }
    void display(){System.out.println(id+" "+name+" "+salary);
    }
}
```

```
public class TestEmployee
{
public static void main(String[] args)
{
    Employee e1=new Employee();
    Employee e2=new Employee();
    Employee e3=new Employee();
    e1.insert(101,"ajeet",45000);
    e2.insert(102,"irfan",25000);
    e3.insert(103,"nakul",55000);
    e1.display();
    e2.display();
    e3.display();
}
}
```

Output:-

```
101 ajeet 45000.0
102 irfan 25000.0
103 nakul 55000.0
```

Example-2

There is given another example that maintains the records of Rectangle class.

```
class Rectangle
{
 int length;
 int width;
 void insert(int l, int w)
{
  length=l;
  width=w;
 }
 void calculateArea()
{
System.out.println(length*width);
}
}
class TestRectangle1
{
 public static void main(String args[])
{
```

```
  Rectangle r1=new Rectangle();
  Rectangle r2=new Rectangle();
  r1.insert(11,5);
  r2.insert(3,15);
  r1.calculateArea();
  r2.calculateArea();
}
}
```

Output:-

```
55
45
```