# LAB - 4

## Java Programming Lab

### Topics Covered

Classes & object Constructor

Syeda Reeha Quasar
14114802719
4C7

# EXPERIMENT – 4.1

## Aim:

WAP that creates a class circle with instance variables for the center and the radius. Initialize and display its variables.

## Theory:

**Class Definition in Java**

A Class is a template for an object. Every object has a class which defines the structure of an object (that means what are various component in it, termed as member elements) its functional inter feces (called methods) which decide what messages the object will respond to and how. The general form of class definition is shown below.

Class  Class Name

[extends SuperClassName ]

    [ implements Interface ] {

[declaration of member elements ]

[ declaration of methods ]}

Creating and Initializing Objects

Take another look at how we've been creating Circle objects:

Circle c = new Circle();

Constructor overloading in Java

In Java, we can overload constructors like methods. The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

Consider the following java program, in which we have used different constructors in the class.

Instantiating a Class

The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the object constructor.

Note: The phrase "instantiating a class" means the same thing as "creating an object." When you create an object, you are creating an "instance" of a class, therefore "instantiating" a class.

**Concepts:**

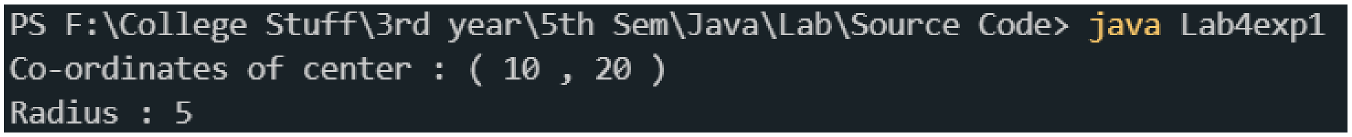
- **class** keyword is used to declare a class in Java.
- **public** keyword is an access modifier that represents visibility. It means it is visible to all.
- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.
- **void** is the return type of the method. It means it doesn't return any value.
- **main** represents the starting point of the program.
- **String[] args** or **String args[]** is used for command line argument. We will discuss it in coming section.
- **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class. We will discuss the internal working of System.out.println() statement in the coming section.
- **Java Utils:** Resources Job Search Discussion. Java. util package contains the **collections framework, legacy collection classes, event model, date and time facilities, internationalization, and miscellaneous utility classes**. This reference will take you through simple and practical methods available in java.
- util. Java util package contains **collection framework, collection classes**, classes related to date and time, event model, internationalization, and miscellaneous utility classes. ... On importing this package, you can access all these classes and methods.
- **Scanner** is a class in **java. util package used for obtaining the input of** the primitive types like int, double, etc. and strings. ... To create an object of Scanner class, we usually pass the predefined object System.in, which represents the standard input stream.
- A switch statement **allows a variable to be tested for equality against a list of values**. Each value is called a case, and the variable being switched on is checked for each case.

- The input is **the data that we give to the program**. The output is the data what we receive from the program in the form of result. Stream represents flow of data or the sequence of data.

## Source Code:

```
class CircleExp1 {
    public int x;
    public int y;
    public int radius;
}

public class Lab4exp1 {
    public static void main(String args[]) {
        CircleExp1 obj = new CircleExp1();
        obj.x = 10;
        obj.y = 20;
        obj.radius = 5;
System.out.println("Co-ordinates of center : ( " + obj.x + " , " + obj.y + " )");
System.out.println("Radius : " + obj.radius );
    }
}
```

## Output:



```
PS F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code> java Lab4exp1
Co-ordinates of center : ( 10 , 20 )
Radius : 5
```

# EXPERIMENT – 4.2

## Aim:

Modify experiment 4.1 to have a constructor in class circle to initialize its variables.

## Theory:

**Class :** 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. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator ,  e.g  c1.radius , and invoke other class methods  by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**New Operator :** The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor. Example of object initialisation : Circle c1 = new Circle();

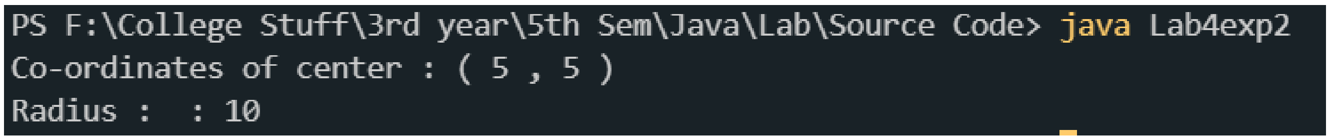
**Instance Variables :** Instance variables are declared in a class, but outside a method, constructor or any block. Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed. Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.

**Constructor :** In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory. It is a special type of method which is used to initialize the object. Every time an object is created using the new() keyword, at least one constructor is called. We can pass our values , when calling out the constructor using new operator, to initialize the object's values. If there is no constructor available in the class It calls a default constructor. In such case, Java compiler provides a default constructor by default.

## Source Code:

```
class Circle {
    int x;
    int y;
    int radius;

    //constructor
    Circle(int i, int j, int r) {
        x = i;
        y = j;
        radius = r;
    }
}

public class Lab4exp2 {
    public static void main(String args[]) {
        Circle obj = new Circle(5, 5, 10);
System.out.println("Co-ordinates of center : ( " +   obj.x + " , " + obj.y + 
" )");
            System.out.println("Radius :  : " + obj.radius);
    }
}
```

## Output:



```
PS F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code> java Lab4exp2
Co-ordinates of center : ( 5 , 5 )
Radius :  : 10
```

# EXPERIMENT – 4.3

## Aim:

Modify experiment 4.2 to show constructor overloading.

## Theory:

**Class :** 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. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator ,  e.g  c1.radius , and invoke other class methods  by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**New Operator :** The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor. Example of object initialisation : Circle c1 = new Circle();

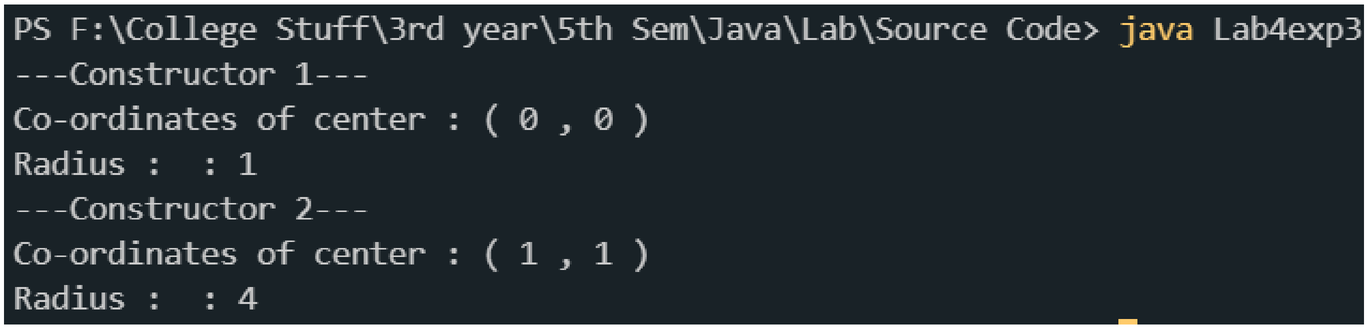
**Constructor :** In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory. It is a special type of method which is used to initialize the object. Every time an object is created using the new() keyword, at least one constructor is called. We can pass our values , when calling out the constructor using new operator, to initialize the object's values. If there is no constructor available in the class It calls a default constructor. In such case, Java compiler provides a default constructor by default.

**Constructor Overloading :** Java Constructor overloading is a technique in which a class can have any number of constructors that differ in parameter list. The compiler differentiates these constructors by taking into account the number of parameters in the list and their type. The constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

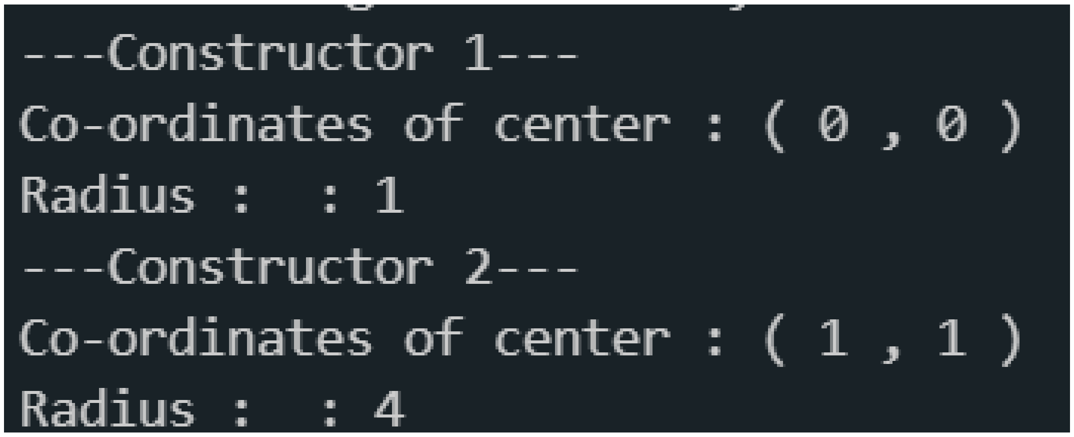
## Source Code:

```
class CircleExp3 {
    int x;
    int y;
    int radius;
    // constructor overloading
    // default constructor
    CircleExp3() {
        x = 0;
        y = 0;
        radius = 1;
    }
    // user input constructor
    CircleExp3(int i, int j, int r) {
        x = i;
        y = j;
        radius = r;
    }
}

public class Lab4exp3 {
    public static void main(String args[]) {
        CircleExp3 obj = new CircleExp3(1, 1, 4);
        CircleExp3 obj1 = new CircleExp3();
        System.out.println("---Constructor 1---");
        System.out.println("Co-ordinates of center : ( " + obj1.x + " , " +
obj1.y + " )");
        System.out.println("Radius :  : " + obj1.radius);
        System.out.println("---Constructor 2---");
        System.out.println("Co-ordinates of center : ( " + obj.x + " , " +
obj.y + " )");
        System.out.println("Radius :  : " + obj.radius);
    }
}
```

## Output:



```
PS F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code> java Lab4exp3
---Constructor 1---
Co-ordinates of center : ( 0 , 0 )
Radius :   : 1
---Constructor 2---
Co-ordinates of center : ( 1 , 1 )
Radius :   : 4
```



```
---Constructor 1---
Co-ordinates of center : ( 0 , 0 )
Radius :   : 1
---Constructor 2---
Co-ordinates of center : ( 1 , 1 )
Radius :   : 4
```

# EXPERIMENT – 4.4

## Aim:

WAP to display the use of this keyword.

## Theory:

**Class :** 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. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities. A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g c1.radius , and invoke other class methods by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**New Operator :** The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor. Example of object initialisation : Circle c1 = new Circle();

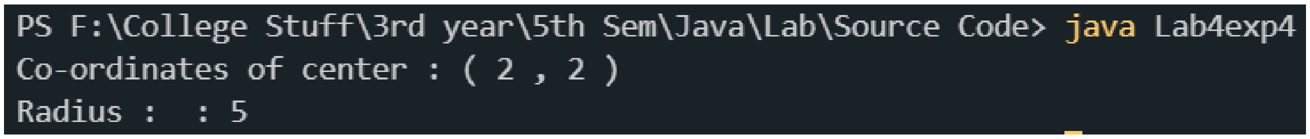
**Constructor :** In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory. It is a special type of method which is used to initialize the object. Every time an object is created using the new() keyword, at least one constructor is called. We can pass our values , when calling out the constructor using new operator, to initialize the object's values. If there is no constructor available in the class It calls a default constructor. In such case, Java compiler provides a default constructor by default.

**This keyword :** this() reference can be used during constructor overloading to call default constructor implicitly from parameterized constructor.

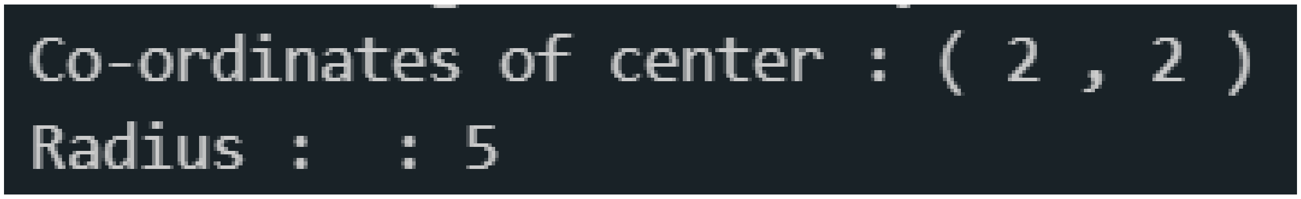
## Source Code:

```
class CircleExp4 {
    int x;
    int y;
    int radius;

    //constructor
    CircleExp4(int i, int j, int r) {
        this.x = i;
        this.y = j;
        this.radius = r;
    }
}

public class Lab4exp4 {
    public static void main(String args[]) {
        CircleExp4 obj = new CircleExp4(2, 2, 5);
        System.out.println("Co-ordinates of center : ( " + obj.x + " , " + obj.y
+ " )");
        System.out.println("Radius :  : " + obj.radius);
    }
}
```

## Output:



```
PS F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code> java Lab4exp4
Co-ordinates of center : ( 2 , 2 )
Radius :  : 5
```



```
Co-ordinates of center : ( 2 , 2 )
Radius :  : 5
```

# EXPERIMENT – 4.5

## Aim:

Write a program that can count the number of instances created for the class.

## Theory:

**Class :** 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. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities. A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g c1.radius , and invoke other class methods by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

**New Operator :** The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor. Example of object initialisation : Circle c1 = new Circle();

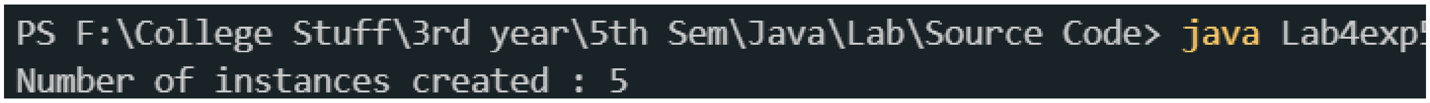
**Constructor :** In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory. It is a special type of method which is used to initialize the object. Every time an object is created using the new() keyword, at least one constructor is called. We can pass our values , when calling out the constructor using new operator, to initialize the object's values. If there is no constructor available in the class It calls a default constructor. In such case, Java compiler provides a default constructor by default.

**Static variable :** When a variable is declared as static, then a single copy of variable is created and shared among all objects at class level. Static variables are, essentially, global variables. All instances of the class share the same static variable.

## Source Code:

```
class countInstance {
    public static int count = 0;

    countInstance() {
        count++;
    }
}

public class Lab4exp5 {
    public static void main(String args[]) {
        countInstance obj1 = new countInstance();
        countInstance obj2 = new countInstance();
        countInstance obj3 = new countInstance();
        countInstance obj4 = new countInstance();
        countInstance obj5 = new countInstance();
        System.out.println("Number of instances created : " +
countInstance.count);
    }
}
```

## Output:



```
PS F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code> java Lab4exp
Number of instances created : 5
```

```
Number of instances created : 5
```

# EXPERIMENT – 4.6

## Aim:

Write a Java Program to get the cube of a given number using the static method.

## Theory:

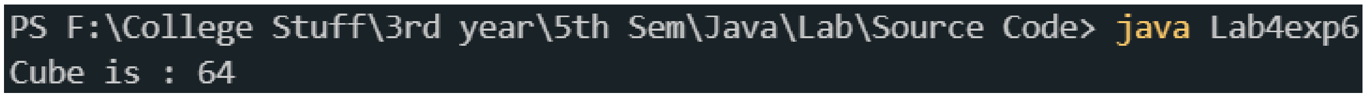
**Functions Used :**

- **Class :** 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. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.
- **Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g  c1.radius , and invoke other class methods  by using the method name and passing the parameters(if required), e.g c1.getArea(radius);
- **New Operator :** The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor. Example of object initialisation : Circle c1 = new Circle();
- **Static method :** The static keyword is used to create methods that will exist independently of any instances created for the class. Static methods do not use any instance variables of any object of the class they are defined in. Static methods take all the data from parameters and compute something from those parameters, with no reference to variables. Class variables and methods can be accessed using the class name followed by a dot and the name of the variable or method.

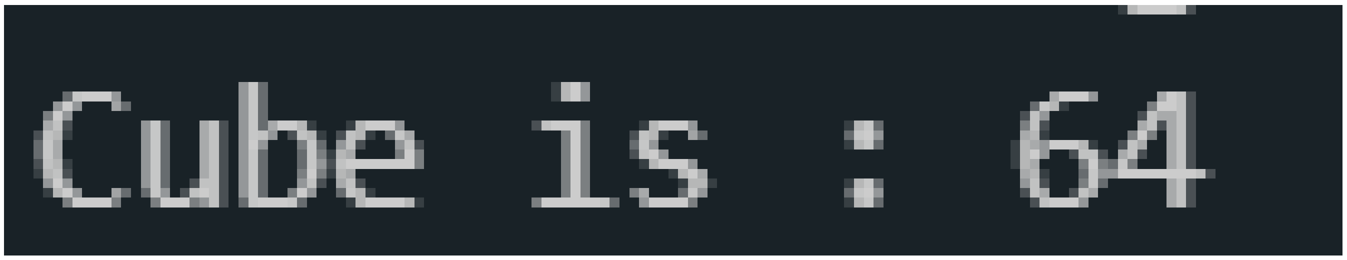
## Source Code:

```
class getCube {
    public static int Cube(int num) {
        return (num * num * num);
    }
}

public class Lab4exp6 {
    public static void main(String args[]) {
        int num = 4;
        System.out.println("Cube is : " + getCube.Cube(num));
    }
}
```

## Output:



```
PS F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code> java Lab4exp6
Cube is : 64
```



```
Cube is : 64
```

# EXPERIMENT – 4.7

## Aim:

WAP that describes a class person. It should have instance variables to record name, age and salary. Create a person object. Set and display its instance variables.

## Theory:

- **Class :** 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. It represents the set of properties or methods that are common to all objects of one type. A class in java has its methods, variables.

- **Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities. A typical Java program creates many objects, interact by invoking methods. We create object of a class in another class to access its methods and variables. We can declare an object by using the Class Name of the class of which we want to create an object, followed by object name, e.g Circle c1; . An object can access variables of other class by using the . operator , e.g c1.radius , and invoke other class methods by using the method name and passing the parameters(if required), e.g c1.getArea(radius);

- **New Operator :** The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor. Example of object initialisation : Circle c1 = new Circle();

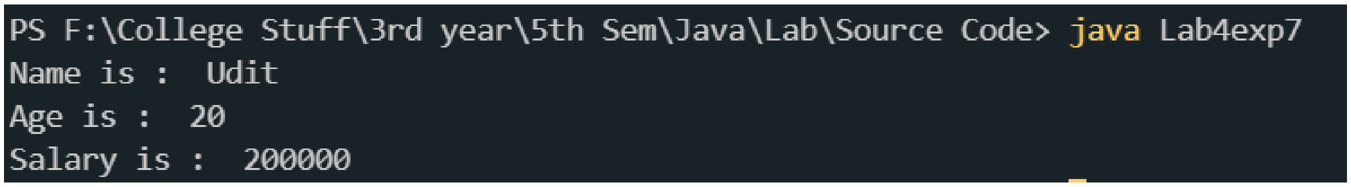
- **Instance Variables :** Instance variables are declared in a class, but outside a method, constructor or any block. Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed. Instance variables hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.

- **Constructor :** In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory. It is a special type of method which is used to initialize the object. Every time an object is created using the new() keyword, at least one constructor is called. We can pass our values , when calling out the constructor using new operator, to initialize the object's values. If there is no constructor available in the class It calls a default constructor. In such case, Java compiler provides a default constructor by default.

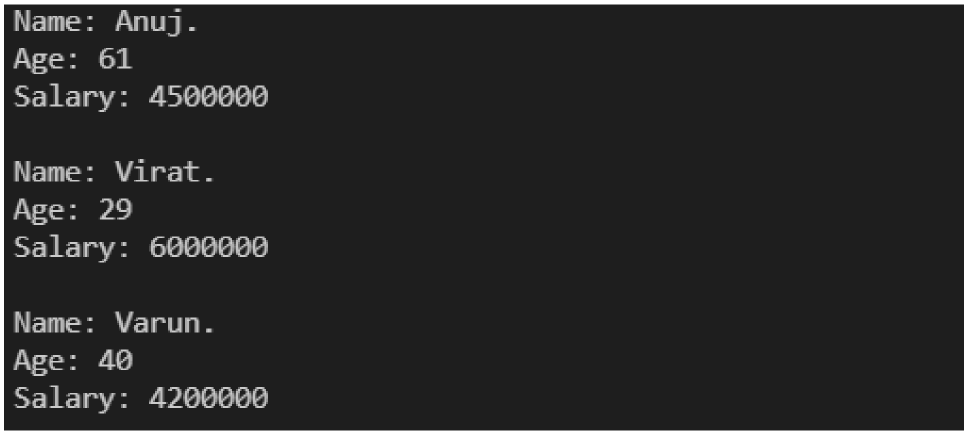
## Source Code:

```
class Person{
    String name;
    int age;
    int salary;
    Person(String n, int a, int s){
        name = n;
        age = a;
        salary = s;
    }
}

public class Lab4exp7 {
    public static void main(String args[]) {
        Person p1 = new Person("Udit",20,200000);

System.out.println("Name is :   " + p1.name + "\nAge is :   " + p1.age + "\nSalary
is :   " + p1.salary  );
    }
}
```

## Output:



```
PS F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code> java Lab4exp7
Name is :  Udit
Age is :  20
Salary is :  200000
```



```
Name: Anuj.
Age: 61
Salary: 4500000

Name: Virat.
Age: 29
Salary: 6000000

Name: Varun.
Age: 40
Salary: 4200000
```

## Viva Questions

### 1. Is Java a pure object oriented language?

Ans.

Java uses primitive data types and hence is not a pure object oriented language.

### 2. What are local variables?

Ans.

Local variables are those which are declared within a block of code like methods. Local variables should be initialized before accessing them.

### 3. Can a class declared as private be accessed outside its package?

Ans.

Not possible.

### 4. Can a class be declared as protected?

Ans.

A class can't be declared as protected. only methods can be declared as protected.

### 5. What is the access scope of a protected method?

Ans.

A protected method can be accessed by the classes within the same package or by the subclasses of the class in any package.

**6. What is the purpose of declaring a variable as final?**

Ans.

A final variable's value can't be changed. final variables should be initialized before using them.

**7. Can a class be declared as static?**

Ans.

We can not declare top level class as static, but only inner class can be declared static.

```
public class Test
{
static class InnerClass
{
public static void InnerMethod()
{ System.out.println("Static Inner Class!");
 }
}
public static void main(String args[])
{
Test.InnerClass.InnerMethod();
}
}
//output: Static Inner Class!
```

**8. When will you define a method as static?**

Ans.

When a method needs to be accessed even before the creation of the object of the class then we should declare the method as static.

**9. What are the restriction imposed on a static method or a static block of code?**

Ans.

A static method should not refer to instance variables without creating an instance and cannot use "this" operator to refer the instance.