

Government College of Engineering, Jalgaon
Department of Computer Engineering
Experiment No: 03

Name:
Subject:CO310U (Application programming Lab)
Class:T.Y. B.Tech
Date of Performance:

PRN:
Sem:V(Odd)
Academic Year:2024-25
Date of Completion:

Aim: Write a program for the following

- a. Write a java program to implement a class mechanism. – Create a class, methods and invoke them inside the main method.
- b. Write a java program to implement constructor overloading
- c. Write a java program implement method overloading

Required Software: OpenJDK version "1.8.0_131"

OpenJDK Runtime Environment (build 1.8.0_131-8u131-b11-2ubuntu1.16.04.3-b11)

OpenJDK 64-Bit Server VM (build 25.131-b11, mixed mode)

Java Compiler Version - JAVAC 1.8.0_131

Theory:

Object:An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

An object has three characteristics:

- **State:** represents the data (value) of an object.
- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.or Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

An object is an instance of a class. 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

What is a class 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 in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface
- *is an entity which has state and behavior.*
- The object is *an instance of a class.*

Instance variable in Java

A variable which is created inside the class but outside the method is known as an instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when an object or instance is created. That is why it is known as an instance variable.

Method in Java

In Java, a method is like a function which is used to expose the behavior of an object.

Advantage of Method

- Code Reusability
- Code Optimization

new keyword in Java

The new keyword is used to allocate memory at runtime. All objects get memory in the Heap memory area.

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. 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. It calls a default constructor if

there is no constructor available in the class. In such case, Java compiler provides a default constructor by default. There are two types of constructors in Java: no-arg constructor, and parameterized constructor.

Note: It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

Rules for creating Java constructor

There are two rules defined for the constructor.

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

Types of Java constructors

There are two types of constructors in Java:

1. Default constructor (no-arg constructor)
2. Parameterized constructor

A constructor is called "Default Constructor" when it doesn't have any parameter.

Syntax of default constructor:

1. `<class_name>(){ }`

Java Parameterized Constructor

A constructor which has a specific number of parameters is called a parameterized constructor.

Why use the parameterized constructor?

The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.

Constructor Overloading in Java

In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods. Constructor overloading in Java is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

Difference between constructor and method in Java

There are many differences between constructors and methods. They are given below.

| Java Constructor | Java Method |
|--|---|
| A constructor is used to initialize the state of an object. | A method is used to expose the behavior of an object. |
| A constructor must not have a return type. | A method must have a return type. |
| The constructor is invoked implicitly. | The method is invoked explicitly. |
| The Java compiler provides a default constructor if you don't have any constructor in a class. | The method is not provided by the compiler in any case. |
| The constructor name must be the same as the class name. | The method name may or may not be the same as the class name. |

Method Overloading:

When a class has two or more methods by the same name but different parameters, at the time of calling based on the parameters passed, the respective method is called (or the respective method body will be bonded with the calling line dynamically). This mechanism is known as method overloading. When a class has two or more methods by the same name but different parameters, at the time of calling based on the parameters passed respective method is called (or respective method body will be bonded with the calling line dynamically). This mechanism is known as method overloading.

Conclusion:

Name & Sign of Course Teacher

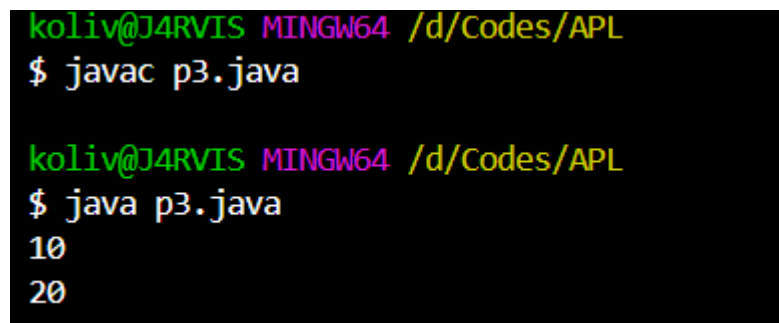
Program:A.

```

class A {
    int l = 10, b = 20;
    void display() {
        System.out.println(l);
        System.out.println(b);
    }
}

class p3 {
    public static void main(String args[]) {
        A a1 = new A();
        a1.display();
    }
}

```

Output:


```

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p3.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p3.java
10
20

```

Program:B.

```

class p3 {
    int id;
    String name;
    int age;
    // creating two-arg constructor
    p3(int i, String n) {
        id = i;
        name = n;
    }
    // creating three-arg constructor
    p3(int i, String n, int a) {
        id = i;

```

```

    name = n;
    age = a;
}

// method to display p3ent details

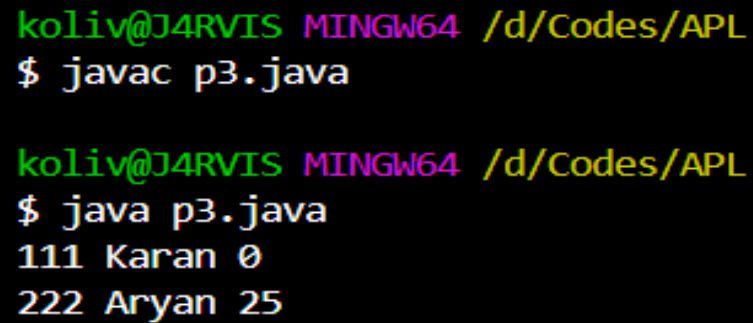
void display() {
    System.out.println(id + " " + name + " " + age);
}

public static void main(String args[]) {

    // creating p3ent objects
    p3 s1 = new p3(111, "Karan");
    p3 s2 = new p3(222, "Aryan", 25);
    // displaying p3ent information
    s1.display();
    s2.display();
}
}

```

Output:



```

koliv@J4RVIS MINGW64 /d/Codes/APL
$ javac p3.java

koliv@J4RVIS MINGW64 /d/Codes/APL
$ java p3.java
111 Karan 0
222 Aryan 25

```

Program:C

```

class A {
    int l = 10, b = 20;

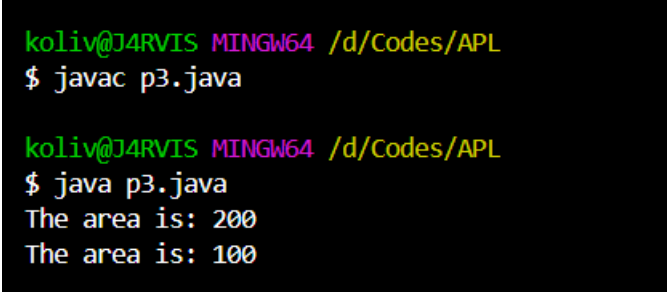
    int area() {
        return l * b;
    }

    int area(int l, int b) {
        return l * b;
    }
}

```

```
class p3 {  
    public static void main(String[] args) {  
        A a1 = new A();  
        int r1 = a1.area();  
        System.out.println("The area is: " + r1);  
  
        int r2 = a1.area(5, 20);  
        System.out.println("The area is: " + r2);  
    }  
}
```

Output:



```
koliv@J4RVIS MINGW64 /d/Codes/APL  
$ javac p3.java  
  
koliv@J4RVIS MINGW64 /d/Codes/APL  
$ java p3.java  
The area is: 200  
The area is: 100
```