Name – Harshit

MIS – 112316018

Java Lab

Assignment 4

1. Write a Java program to demonstrate a no-argument constructor

Code:

// Source code is decompiled from a .class file using FernFlower decompiler.

class NoArgConstructorDemo {

   NoArgConstructorDemo() {

      System.out.println("No-Argument Constructor Called.");

   }

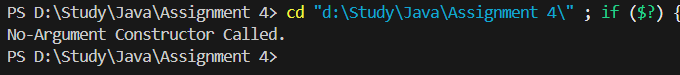
   public static void main(String[] var0) {

      new NoArgConstructorDemo();

   }

}

Output:

  
  
  
**2.**Create a program that shows constructor overloading in Java

Code:

// Source code is decompiled from a .class file using FernFlower decompiler.

class ConstructorOverloadingDemo {

   ConstructorOverloadingDemo() {

      System.out.println("Default Constructor.");

   }

   ConstructorOverloadingDemo(int var1) {

      System.out.println("Parameterized Constructor: " + var1);

   }

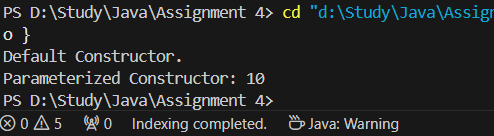
   public static void main(String[] var0) {

      new ConstructorOverloadingDemo();

      new ConstructorOverloadingDemo(10);

   }

}

Output:  
  
  
  
**3.**Implement a parameterized constructor to initialize an object

Code:  
class ParameterizedConstructorDemo {

    int value;

    ParameterizedConstructorDemo(int value) {

        this.value = value;

    }

    void display() {

        System.out.println("Value: " + value);

    }

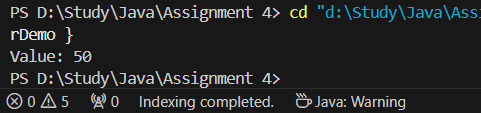
    public static void main(String[] args) {

        ParameterizedConstructorDemo obj = new ParameterizedConstructorDemo(50);

        obj.display();

    }

}

Output:  


**4.**Write a program that uses the this keyword to call one constructor from another

Code

class ThisConstructorDemo {

    ThisConstructorDemo() {

        this(100);

        System.out.println("No-Argument Constructor.");

    }

    ThisConstructorDemo(int value) {

        System.out.println("Value Constructor: " + value);

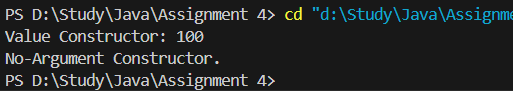
    }

    public static void main(String[] args) {

        new ThisConstructorDemo();

    }

}

Output:  
  
  
**5.**Implement a copy constructor to create a new object by copying another object's state

Code:

class CopyConstructorDemo {

    int data;

    CopyConstructorDemo(int data) {

        this.data = data;

    }

    CopyConstructorDemo(CopyConstructorDemo obj) {

        this.data = obj.data;

    }

    void display() {

        System.out.println("Data: " + data);

    }

    public static void main(String[] args) {

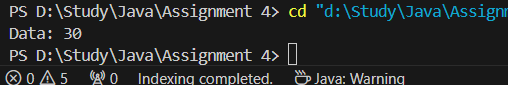
        CopyConstructorDemo original = new CopyConstructorDemo(30);

        CopyConstructorDemo copy = new CopyConstructorDemo(original);

        copy.display();

    }

}

Output:  
  
  
**6.**Write a program, having class ‘Air’ having data members: per\_of\_oxigen, per\_of\_nitrogen, and humidity.  Define overloaded constructors to initialize and method to display the values of these data members. Make the use of this () to call among constructors. Input values for two objects and compare them according to value of humidity.

Code:

class Air {

    double per\_of\_oxygen, per\_of\_nitrogen, humidity;

    Air() {

        this(21, 78, 50);

    }

    Air(double per\_of\_oxygen, double per\_of\_nitrogen, double humidity) {

        this.per\_of\_oxygen = per\_of\_oxygen;

        this.per\_of\_nitrogen = per\_of\_nitrogen;

        this.humidity = humidity;

    }

    void display() {

        System.out.println("Oxygen: " + per\_of\_oxygen + "%, Nitrogen: " + per\_of\_nitrogen + "%, Humidity: " + humidity + "%");

    }

    public static void main(String[] args) {

        Air air1 = new Air(21, 78, 70);

        Air air2 = new Air(19, 80, 65);

        air1.display();

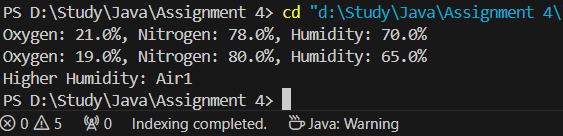
        air2.display();

        System.out.println("Higher Humidity: " + (air1.humidity > air2.humidity ? "Air1" : "Air2"));

    }

}

Output:

  
  
  
**7.**Write a Java Program which defines a class ‘Department’ having data members: name\_of\_hod, total\_students and per\_of\_result. Define overloaded constructors to initialize and method to display the values of these data members. Input values for five objects and determine whose average result per student is worse.

Code:  
class Department {

    String name\_of\_hod;

    int total\_students;

    double per\_of\_result;

    Department(String name\_of\_hod, int total\_students, double per\_of\_result) {

        this.name\_of\_hod = name\_of\_hod;

        this.total\_students = total\_students;

        this.per\_of\_result = per\_of\_result;

    }

    void display() {

        System.out.println("HOD: " + name\_of\_hod + ", Students: " + total\_students + ", Result: " + per\_of\_result + "%");

    }

    public static void main(String[] args) {

        Department[] departments = new Department[5];

        departments[0] = new Department("Dr. Smith", 50, 75.5);

        departments[1] = new Department("Dr. John", 40, 60.0);

        departments[2] = new Department("Dr. Alice", 45, 80.0);

        departments[3] = new Department("Dr. Emma", 55, 90.0);

        departments[4] = new Department("Dr. Brown", 35, 55.0);

        Department worst = departments[0];

        for (Department dept : departments) {

            if (dept.per\_of\_result < worst.per\_of\_result) {

                worst = dept;

            }

        }

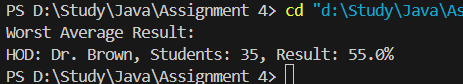
        System.out.println("Worst Average Result:");

        worst.display();

    }

}

Output:

  
  
  
**8.**Write a program which defines a class ‘College’ having data members: total students, branches and percentage of total result. Input the data using static method. Class will contain the constructor. Accept this data for two objects. Create a static method containing two object parameters and display the data of college whose result is better.

Code:  
class College {

    int total\_students;

    int branches;

    double per\_of\_result;

    College(int total\_students, int branches, double per\_of\_result) {

        this.total\_students = total\_students;

        this.branches = branches;

        this.per\_of\_result = per\_of\_result;

    }

    static College betterResult(College c1, College c2) {

        return c1.per\_of\_result > c2.per\_of\_result ? c1 : c2;

    }

    void display() {

        System.out.println("Students: " + total\_students + ", Branches: " + branches + ", Result: " + per\_of\_result + "%");

    }

    public static void main(String[] args) {

        College c1 = new College(500, 5, 85.0);

        College c2 = new College(600, 6, 90.0);

        College better = betterResult(c1, c2);

        better.display();

    }

}

Output:

  
  
  
**9.**Define a class ‘Forest’ having data members: area\_in\_sq\_cm, count\_of\_animals. Define overloaded constructors to initialize and method to display the values of these data members. Input values for two objects and determine which forest is having less space per animal.

Code:

class Forest {

    double area\_in\_sq\_cm;

    int count\_of\_animals;

    Forest(double area\_in\_sq\_cm, int count\_of\_animals) {

        this.area\_in\_sq\_cm = area\_in\_sq\_cm;

        this.count\_of\_animals = count\_of\_animals;

    }

    double spacePerAnimal() {

        return area\_in\_sq\_cm / count\_of\_animals;

    }

    void display() {

        System.out.println("Area: " + area\_in\_sq\_cm + " sq cm, Animals: " + count\_of\_animals);

    }

    public static void main(String[] args) {

        Forest f1 = new Forest(1000000, 200);

        Forest f2 = new Forest(500000, 150);

        f1.display();

        f2.display();

        Forest lessSpace = f1.spacePerAnimal() < f2.spacePerAnimal() ? f1 : f2;

        System.out.println("Forest with Less Space per Animal:");

        lessSpace.display();

    }

}

Output:

