Task 21

public class task21 {

    public static void main(String[] args)

   {

       // initializing array

       int[] arr = { 1, 2, 3, 4, 5 };

       // size of array

       int n = arr.length;

       // traversing array

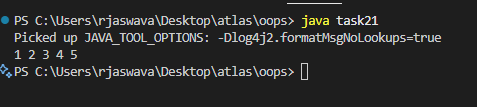
       for (int i = 0; i < n; i++)

           System.out.print(arr[i] + " ");

   }

}

Output



Task 22

public class task22 {

    public static void main(String[] args)

   {

       // declares an Array of integers.

       int[] arr;

       // allocating memory for 5 integers.

       arr = new int[5];

       // initialize the elements of the array

       // first to last(fifth) element

      arr[0] = 10;

      arr[1] = 20;

       arr[2] = 30;

       arr[3] = 40;

       arr[4] = 50;

       // accessing the elements of the specified array

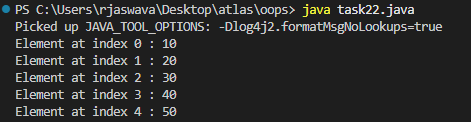
       for (int i = 0; i < arr.length; i++)

           System.out.println("Element at index " + i + " : " + arr[i]);

   }

}

Output



Task 23

class Student {

    public int roll\_no;

    public String name;

        Student(){

        this.roll\_no = 0;

        this.name = "";

    }

     Student(int Roll\_no, String Name){

        this.roll\_no = Roll\_no;

        this.name = Name;

    }

}

public class task23 {

    public static void main(String[] args){

        Student sobj1 = new Student();

        Student sobj2 = new Student();

        Student sobj3 = new Student();

        // declares an Array of Student

        Student[] arr;

        // allocating memory for 5 objects of type Student.

        arr = new Student[5];

        // initialize the elements of the array

        arr[0] = new Student(1, "aman");

        arr[1] = new Student(2, "vaibhav");

        arr[2] = new Student(3, "shikar");

        arr[3] = new Student(4, "dharmesh");

        arr[4] = new Student(5, "mohit");

        // accessing the elements of the specified array

        for (int i = 0; i < arr.length; i++)

            System.out.println("Element at " + i + " : { "

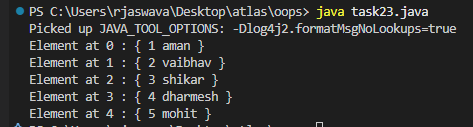
                             + arr[i].roll\_no + " "

                             + arr[i].name+" }");

    }

}

Output



Task 24

class Student{

   public String name;

    Student(String name){

       this.name = name;

   }

  @Override

   public String toString(){

       return name;

   }

}

public class task24 {

   public static void main (String[] args){

       // declares an Array and initializing the

      // elements of the array

       Student[] myStudents = new Student[]{

         new Student("Dharma"),new Student("sanvi"),

         new Student("Rupa"),new Student("Ajay")

       };

        // accessing the elements of the specified array

       for(Student m:myStudents){

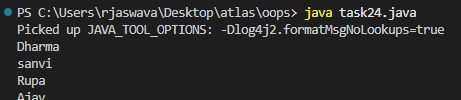
           System.out.println(m);

       }

   }

}

Output



Task 25

public class task25 {

    public static void main(String[] args)

   {

       int[] arr = new int[4];

       arr[0] = 10;

       arr[1] = 20;

       arr[2] = 30;

       arr[3] = 40;

       System.out.println(

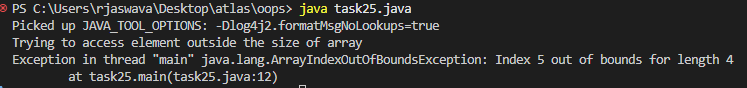
           "Trying to access element outside the size of array");

       System.out.println(arr[5]);

   }

}

Output



Task 26

import java.io.\*;

public class task26 {

    public static void main(String[] args){

       // Two Dimensional Array

      // Declared and Initialized

      int[][] arr = new int[3][3];

       // Number of Rows

       System.out.println("Rows : " + arr.length);

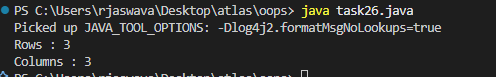
       // Number of Columns

       System.out.println("Columns : " + arr[0].length);

   }

}

Output



Task 27

public class task27 {

    // main function

   public static void main(String args[])

   {

       // declaring and initializing 2D array

       int arr[][] = { { 2, 7, 9 }, { 3, 6, 1 }, { 7, 4, 2 } };

       // printing 2D array

       for (int i = 0; i < 3; i++) { // rows

           for (int j = 0; j < 3; j++) // columns

               System.out.print(arr[i][j] + " ");

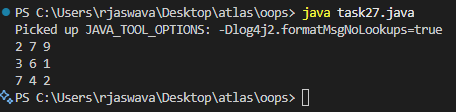
           System.out.println();

       }

   }

}

Output



Task 28

public class task28 {

     public static void main(String args[])

   {

       int arr[] = { 3, 1, 2, 5, 4 };

       // passing array to method m1

       sum(arr);

   }

   public static void sum(int[] arr)

   {

       // getting sum of array values

       int sum = 0;

       for (int i = 0; i < arr.length; i++)

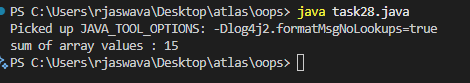
           sum += arr[i];

       System.out.println("sum of array values : " + sum);

   }

}

Output



Task 29

public class task29 {

    public static void main(String args[])

   {

       int arr[] = m1();

       for (int i = 0; i < arr.length; i++)

           System.out.print(arr[i] + " ");

   }

   public static int[] m1()

   {

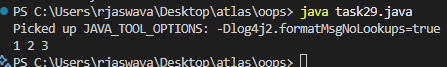
       // returning  array

       return new int[] { 1, 2, 3 };

   }

}

Output



Task 30

public class task30 {

    public static void main(String args[])

   {

       int intArray[] = { 1, 2, 3 };

       int cloneArray[] = intArray.clone();

       // will print false as shallow copy is created

       System.out.println(intArray == cloneArray);

       for (int i = 0; i < cloneArray.length; i++) {

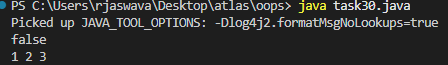
           System.out.print(cloneArray[i] + " ");

       }

   }

}

Output



Task 31

public class task31 {

public static void main(String args[])

   {

       int intArray[][] = { { 1, 2, 3 }, { 4, 5 } };

       int cloneArray[][] = intArray.clone();

       // will print false

       System.out.println(intArray == cloneArray);

       // will print true as shallow copy is created

       // i.e. sub-arrays are shared

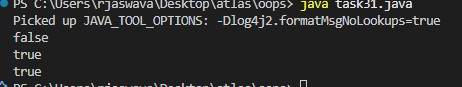
       System.out.println(intArray[0] == cloneArray[0]);

       System.out.println(intArray[1] == cloneArray[1]);

   }

}

Output



Task 32

// Java Program for Deep Copy of 2D Array

public class task32 {

    // A simple class for illustration purposes

    static class MyClass {

        int value;

        MyClass(int value) {

            this.value = value;

        }

    }

    public static void main(String[] args) {

        // Create the original 2D array with objects

        MyClass[][] originalArray = {

            {new MyClass(1), new MyClass(2)},

            {new MyClass(3), new MyClass(4)}

        };

        // Perform the deep copy

        MyClass[][] deepCopiedArray = deepCopy2DArray(originalArray);

        // Modify an element in the original array to demonstrate independence

        originalArray[0][0].value = 99;

        // Display the original and deep copied arrays

        System.out.println("Original Array:");

        print2DArray(originalArray);

        System.out.println("\nDeep Copied Array:");

        print2DArray(deepCopiedArray);

    }

    // Method to perform a deep copy of a 2D array with objects

    private static MyClass[][] deepCopy2DArray(MyClass[][] originalArray) {

        int rows = originalArray.length;

        int cols = originalArray[0].length;

        MyClass[][] copiedArray = new MyClass[rows][cols];

        for (int i = 0; i < rows; i++) {

            for (int j = 0; j < cols; j++) {

                // Create a new instance of the object for each element

                copiedArray[i][j] = new MyClass(originalArray[i][j].value);

            }

        }

        return copiedArray;

    }

    // Method to print a 2D array with objects

    private static void print2DArray(MyClass[][] array) {

        for (MyClass[] row : array) {

            for (MyClass element : row) {

                System.out.print(element.value + " ");

            }

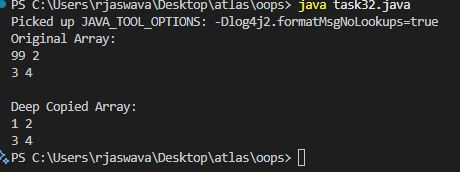
            System.out.println();

        }

    }

}

Output



Task 33

class Calculation {

   int z;

   public void addition(int x, int y) {

      z = x + y;

      System.out.println("The sum of the given numbers:"+z);

   }

   public void Subtraction(int x, int y) {

      z = x - y;

      System.out.println("The difference between the given numbers:"+z);

   }

}

public class task33 extends Calculation {

   public void multiplication(int x, int y) {

      z = x \* y;

      System.out.println("The product of the given numbers:"+z);

   }

   public static void main(String args[]) {

      int a = 20, b = 10;

      task33 demo = new task33();

      demo.addition(a, b);

      demo.Subtraction(a, b);

      demo.multiplication(a, b);

   }

}

class My\_Calculation2 extends Calculation {

   public void multiplication(int x, int y) {

      z = x \* y;

      System.out.println("The product of the given numbers:"+z);

   }

   public static void main(String args[]) {

      int a = 20, b = 10;

      My\_Calculation2 demo = new My\_Calculation2();

      demo.addition(a, b);

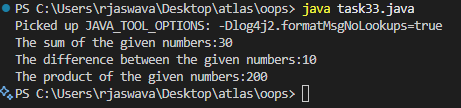
      demo.Subtraction(a, b);

      demo.multiplication(a, b);

   }

}

Output



Task 34

class Clock {

    int hours;

    int minutes;

    Clock() {

        hours = 0;

        minutes = 0;

    }

}

class Calculation {

    int z;

    public void addition(int x, int y) {

        z = x + y;

        System.out.println("The sum of the given numbers:"+z);

    }

}

public class task34 {

    public static void main(String args[]) {

        Clock c = new Clock();

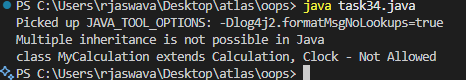
        System.out.println("Multiple inheritance is not possible in Java");

        System.out.println("class MyCalculation extends Calculation, Clock - Not Allowed");

    }

}

Output



Task 35

class Customer {

    int cost = 40;

    String items = "Tomatoes";

    Customer() {

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

    }

    void purchage\_list() {

        System.out.println("cost of tomatoes in Customer class is " + cost);

    }

}

public class task35 extends Customer {

    void billing() {

        String items = "onions";

        int cost = 30;

        super.items = "Potatoes";

        super.cost = 50;

        super.purchage\_list();

        System.out.println(items);

        System.out.println(cost);

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        System.out.println(super.items);

        System.out.println(super.cost);

    }

    public static void main(String[] args) {

        Customer cobj = new Customer();

        cobj.purchage\_list();

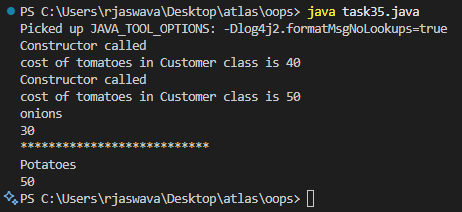
        task35 tobj = new task35();

        tobj.billing();

    }

}

Output



Task 36

class Superclass {

    int var;

    Superclass(int var) {

        this.var = var;

    }

    public void getVar() {

        System.out.println("var value in super class is " + var);

    }

}

public class task36 extends Superclass {

    task36(int var) {

        super(var);

    }

    public static void main(String[] args) {

        Superclass sobj = new Superclass(100);

        sobj.getVar();

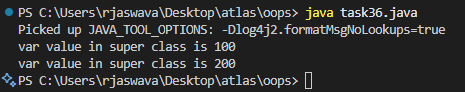
        task36 tobj = new task36(200);

        tobj.getVar();

    }

}

Output



Task 37

public class task37 {

    void add(int x, int y) {

        System.out.println("x = " + x + " y = " + y);

    }

    void add(int x, int y, int z) {

        System.out.println("x = " + x + " y = " + y + " z = " + z);

    }

    public static void main(String[] args) {

        task37 obj = new task37();

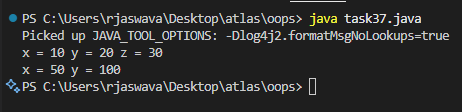
        obj.add(10, 20, 30);

        obj.add(50, 100);

    }

}

Output



Task 38

public class task38 {

    void add(char x, char y) {

        System.out.println("x = " + x + " y = " + y);

    }

    void add(int x, int y) {

        System.out.println("x = " + x + " y = " + y);

    }

    void add(int x, float y) {

        System.out.println("x = " + x + " y = " + y);

    }

    void add(float x, int y) {

        System.out.println("x = " + x + " y = " + y);

    }

    public static void main(String[] args) {

        task38 obj = new task38();

        obj.add('d', 'a');

        obj.add(100, 100);

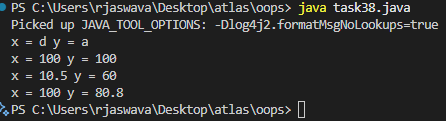
        obj.add(10.50f, 60);

        obj.add(100, 80.80f);

    }

}

Output



Task 39

class Employee {

    private int pwd;

    protected int Salary;

    public int empid;

    Employee() {

        pwd = 0;

        Salary = 0;

        empid = 0;

    }

}

class Hr extends Employee {

    Hr() {

        //super.pwd = 1254;    // Cannot access private member

        super.Salary = 50000;  // Can access protected member

        super.empid = 10001;   // Can access public member

    }

}

public class task39 {

    public static void main(String[] args) {

        Employee emp = new Employee();

        //emp.pwd = 1254;      // Cannot access private member

        emp.Salary = 50000;    // Can access protected member

        emp.empid = 10001;     // Can access public member

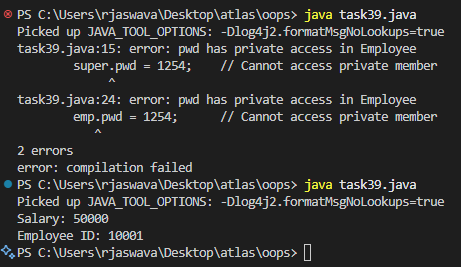
        System.out.println("Salary: " + emp.Salary);

        System.out.println("Employee ID: " + emp.empid);

    }

}

Output



Task 40

abstract class Employee {

   private String name;

   private String address;

   private int number;

   public Employee(String name, String address, int number) {

      System.out.println("Constructing an Employee");

      this.name = name;

      this.address = address;

      this.number = number;

   }

   public double computePay() {

     System.out.println("Inside Employee computePay");

     return 0.0;

   }

   public void mailCheck() {

      System.out.println("Mailing a check to " + this.name + " " + this.address);

   }

   public String toString() {

      return name + " " + address + " " + number;

   }

   public String getName() {

      return name;

   }

   public String getAddress() {

      return address;

   }

   public void setAddress(String newAddress) {

      address = newAddress;

   }

   public int getNumber() {

      return number;

   }

}

public class task40 {

   public static void main(String [] args) {

      /\* Following is not allowed and would raise error \*/

      //Employee e = new Employee("George W.", "Houston, TX", 43);

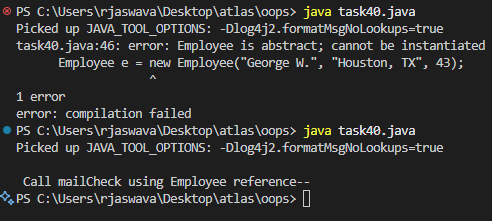
      System.out.println("\n Call mailCheck using Employee reference--");

      //e.mailCheck();

   }

}

Output



Task 41

abstract class Employee {

   private String name;

   private String address;

   private int number;

   public Employee(String name, String address, int number) {

      System.out.println("Constructing an Employee");

      this.name = name;

      this.address = address;

      this.number = number;

   }

   public double computePay() {

     System.out.println("Inside Employee computePay");

     return 0.0;

   }

   public void mailCheck() {

      System.out.println("Mailing a check to " + this.name + " " + this.address);

   }

   public String toString() {

      return name + " " + address + " " + number;

   }

   public String getName() {

      return name;

   }

   public String getAddress() {

      return address;

   }

   public void setAddress(String newAddress) {

      address = newAddress;

   }

   public int getNumber() {

      return number;

   }

}

class SalaryEmployee extends Employee {

   public SalaryEmployee(String name, String address, int number) {

      super(name, address, number);

   }

}

public class task41 {

   public static void main(String [] args) {

      SalaryEmployee e = new SalaryEmployee("George W.", "Houston, TX", 43);

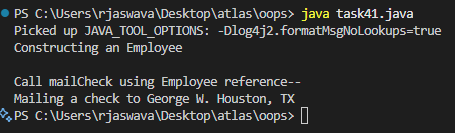
      System.out.println("\nCall mailCheck using Employee reference--");

      e.mailCheck();

   }

}

Output



Task 42

abstract class Gadgets {

    abstract void turnOn();

    abstract void turnOff();

}

class TVRemote extends Gadgets {

    @Override

    void turnOn() {

        System.out.println("TV is turned ON.");

    }

    @Override

    void turnOff() {

        System.out.println("TV is turned OFF.");

    }

}

class ACRemote extends Gadgets {

    @Override

    void turnOn() {

        System.out.println("AC is turned ON.");

    }

    @Override

    void turnOff() {

        System.out.println("AC is turned OFF.");

    }

}

public class task42 {

    public static void main(String[] args) {

        Gadgets tvRemote = new TVRemote();

        Gadgets acRemote = new ACRemote();

        tvRemote.turnOn();

        tvRemote.turnOff();

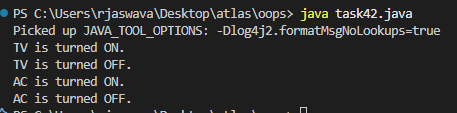
        acRemote.turnOn();

        acRemote.turnOff();

    }

}

Output



Task 43

import java.io.\*;

// Interface Declared

//Driver Code Ends

interface testInterface {

    // public, static and final

    final int tax = 10;

    // public and abstract

    void display();

}

// Class implementing interface

class TestClass implements testInterface {

    // Implementing the capabilities of

    // Interface

    public void display(){

      System.out.println("Myclass");

    }

}

class task43 {

//Driver Code Starts

    public static void main(String[] args)

    {

        TestClass t = new TestClass();

        t.display();

        System.out.println(t.tax);

    }

}

Output

