Program 1: WAP in Java to print Hello.

public class Q1{

    public static void main(String[] args) {

        System.out.println("Hello, It's Java");

    }

}

Output🡪



Program 2: WAP in java to understand the difference between print() and println().

public class Q2 {

    public static void main(String[] args) {

        // Using println() method

        System.out.println("Hello, World!"); // This will print "Hello, World!" followed by a newline

        // Using print() method

        System.out.print("Hello, "); // This will print "Hello, " without a newline

        System.out.println("World!"); // This will print "World!" followed by a newline

        // Printing multiple values using print()

        System.out.print("Java");

        System.out.print(" is");

        System.out.print(" awesome!");

        // This will print "Java is awesome!" without any newline

        // Printing multiple values using println()

        System.out.println("Java");

        System.out.println(" is");

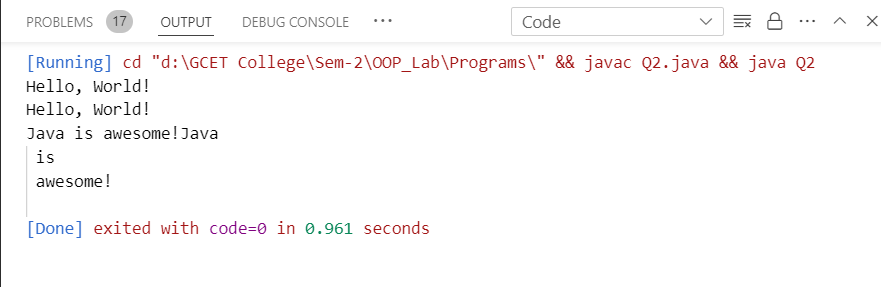
        System.out.println(" awesome!");

        // This will print each word on a new line

    }

}

Output🡪



Program 3: WAP in Java with two classes create a object of first class and call into another class (having main method).

class FirstClass {

    public void display(){

        System.out.println("Calling method of another class");

    }

}

public class SecondClass {

    public static void main(String[] args) {

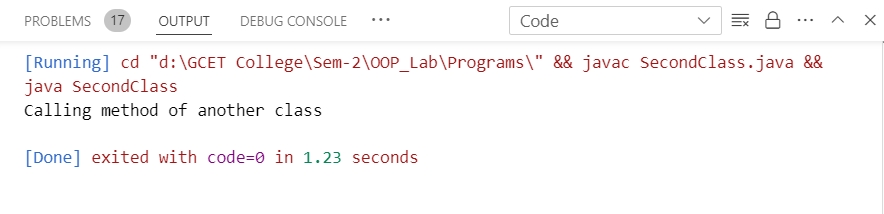
        FirstClass obj = new FirstClass();

        obj.display();

    }

}

Output🡪



Program 4: WAP in Java to product of two numbers.

public class Q4 {

    public static void main(String[] args) {

        int a = 25;

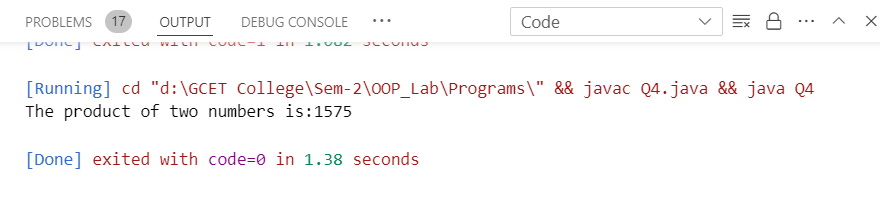
        int b = 63;

        System.out.println("The product of two numbers is:" + a\*b);

    }

}

Output🡪



Program 5: WAP in Java to product of two numbers (Input by the user).

import java.util.Scanner;

public class Q5 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int a,b;

        System.out.println("Enter first value");

        a = sc.nextInt();

        System.out.println("Enter second value");

        b = sc.nextInt();

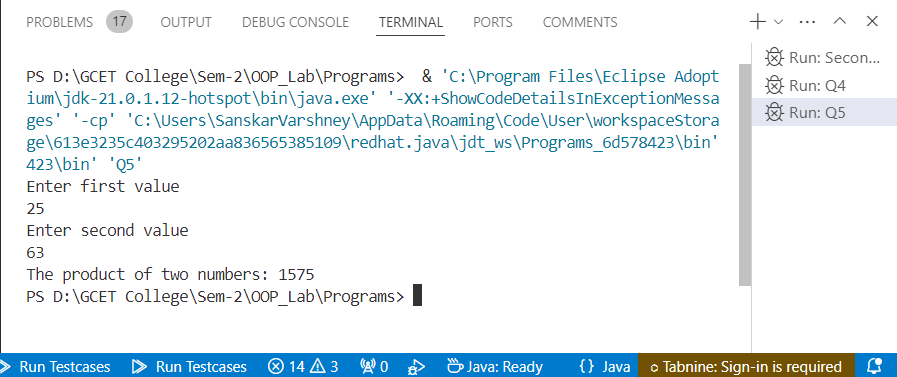
        System.out.println("The product of two numbers: " + a\*b);

        sc.close();

    }

}

Output🡪



Program 6: WAP in Java illustrate the concept of local instance and static variable.

public class Q6 {

    // Static variable

    static int staticVar = 50;

    // Instance variable

    int instanceVar = 10;

    public static void main(String[] args) {

        // Local variable

        int localVar = 5;

        System.out.println("Static variable: " + staticVar);

        // Creating an object of the class to access instance variable

        Q6 obj = new Q6();

        System.out.println("Instance variable: " + obj.instanceVar);

        System.out.println("Local variable: " + localVar);

        // Calling a method to demonstrate usage of local, instance, and static variables

        obj.methodExample();

    }

    void methodExample() {

        // Accessing local, instance, and static variables within a method

        int localVar = 20;

        System.out.println("Inside method - Local variable: " + localVar);

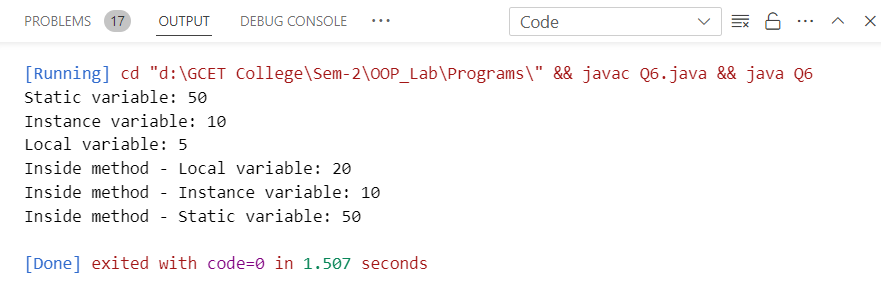
        System.out.println("Inside method - Instance variable: " + instanceVar);

        System.out.println("Inside method - Static variable: " + staticVar);

    }

}

Output🡪



Program 7: WAP in Java to implement implicit and explicit type casting.

public class Q7 {

    public static void main(String[] args) {

        // Implicit type casting (widening conversion)

        int numInt = 100;

        long numLong = numInt;  // int is automatically cast to long

        float numFloat = numLong;  // long is automatically cast to float

        System.out.println("Implicit Type Casting:");

        System.out.println("int to long: " + numLong);

        System.out.println("long to float: " + numFloat);

        // Explicit type casting (narrowing conversion)

        double numDouble = 123.456;

        int numIntExplicit = (int) numDouble;  // double is explicitly cast to int

        System.out.println("\nExplicit Type Casting:");

        System.out.println("double to int: " + numIntExplicit);

        // Explicit type casting with loss of data

        int numInt2 = 257;

        byte numByte = (byte) numInt2;  // int is explicitly cast to byte (loss of data for values > 127)

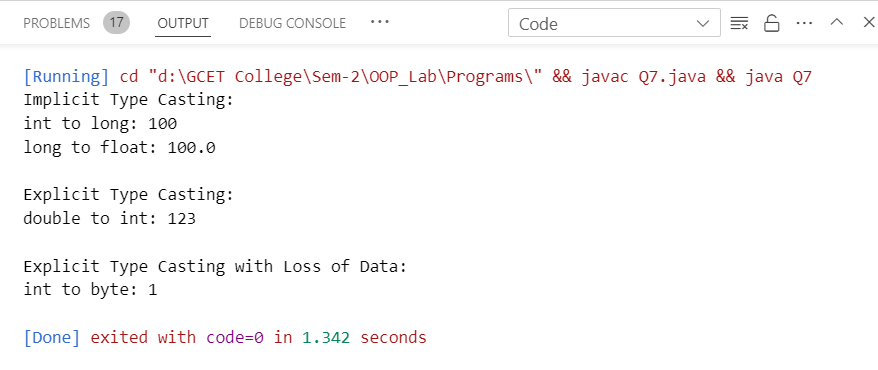
        System.out.println("\nExplicit Type Casting with Loss of Data:");

        System.out.println("int to byte: " + numByte);

    }

}

Output🡪



Program 8: WAP in Java to implement the various operators in java.

public class Q8 {

    public static void main(String[] args) {

        // Arithmetic operators

        int a = 10, b = 20;

        System.out.println("Arithmetic Operators:");

        System.out.println("a + b = " + (a + b));

        System.out.println("a - b = " + (a - b));

        System.out.println("a \* b = " + (a \* b));

        System.out.println("b / a = " + (b / a));

        System.out.println("b % a = " + (b % a));

        // Relational operators

        System.out.println("\nRelational Operators:");

        System.out.println("a == b: " + (a == b));

        System.out.println("a != b: " + (a != b));

        System.out.println("a > b: " + (a > b));

        System.out.println("a < b: " + (a < b));

        System.out.println("b >= a: " + (b >= a));

        System.out.println("b <= a: " + (b <= a));

        // Logical operators

        boolean x = true, y = false;

        System.out.println("\nLogical Operators:");

        System.out.println("x && y: " + (x && y));

        System.out.println("x || y: " + (x || y));

        System.out.println("!x: " + (!x));

        // Bitwise operators

        int m = 5, n = 7;

        System.out.println("\nBitwise Operators:");

        System.out.println("m & n: " + (m & n));

        System.out.println("m | n: " + (m | n));

        System.out.println("m ^ n: " + (m ^ n));

        System.out.println("~m: " + (~m));

        // Assignment operators

        int c = 10;

        System.out.println("\nAssignment Operators:");

        c += 5;  // Equivalent to c = c + 5

        System.out.println("c += 5: " + c);

        c -= 3;  // Equivalent to c = c - 3

        System.out.println("c -= 3: " + c);

        c \*= 2;  // Equivalent to c = c \* 2

        System.out.println("c \*= 2: " + c);

        c /= 2;  // Equivalent to c = c / 2

        System.out.println("c /= 2: " + c);

        c %= 3;  // Equivalent to c = c % 3

        System.out.println("c %= 3: " + c);

        c <<= 2;  // Equivalent to c = c << 2

        System.out.println("c <<= 2: " + c);

        c >>= 2;  // Equivalent to c = c >> 2

        System.out.println("c >>= 2: " + c);

        c &= 2;  // Equivalent to c = c & 2

        System.out.println("c &= 2: " + c);

        c |= 2;  // Equivalent to c = c | 2

        System.out.println("c |= 2: " + c);

        c ^= 2;  // Equivalent to c = c ^ 2

        System.out.println("c ^= 2: " + c);

        // Unary operators

        int d = 5;

        System.out.println("\nUnary Operators:");

        System.out.println("++d: " + (++d));

        System.out.println("--d: " + (--d));

        System.out.println("-d: " + (-d));

        System.out.println("+d: " + (+d));

        // Conditional (Ternary) Operator

        int max = (a > b) ? a : b;

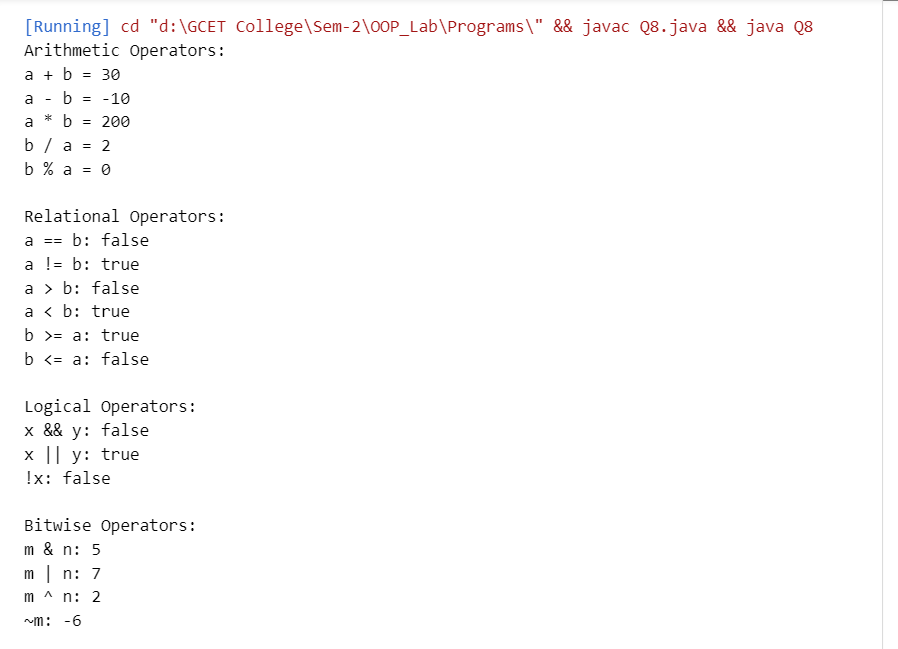
        System.out.println("\nConditional (Ternary) Operator:");

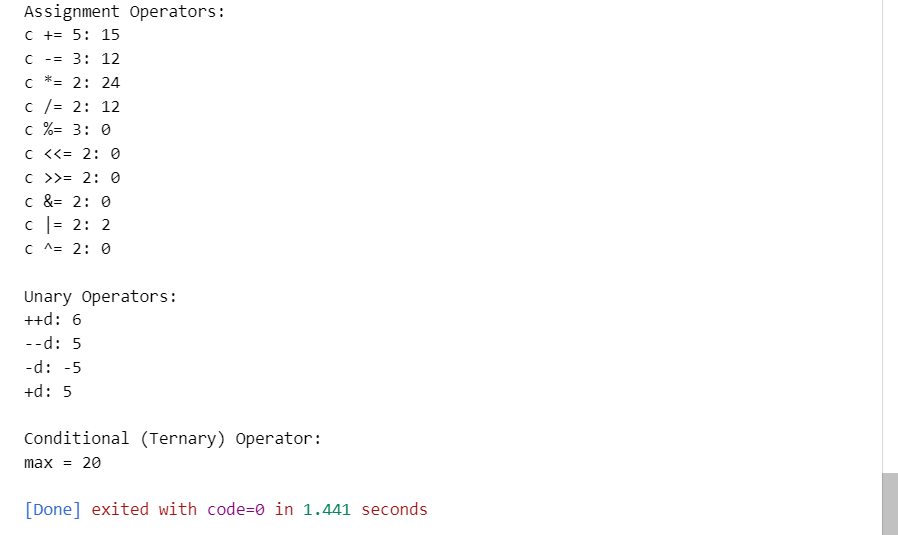
        System.out.println("max = " + max);

    }

}

Output🡪





Program 9: WAP in java for constructor overloading.

public class Q9 {

    private int id;

    private String name;

    private int age;

    // Constructor with no parameters

    public Q9() {

        this.id = 0;

        this.name = "Unknown";

        this.age = 0;

    }

    // Constructor with one parameter

    public Q9(int id) {

        this.id = id;

        this.name = "Unknown";

        this.age = 0;

    }

    // Constructor with two parameters

    public Q9(int id, String name) {

        this.id = id;

        this.name = name;

        this.age = 0;

    }

    // Constructor with three parameters

    public Q9(int id, String name, int age) {

        this.id = id;

        this.name = name;

        this.age = age;

    }

    public void display() {

        System.out.println("Student ID: " + id);

        System.out.println("Student Name: " + name);

        System.out.println("Student Age: " + age);

    }

    public static void main(String[] args) {

        // Using default constructor

        Q9 student1 = new Q9();

        student1.display();

        // Using constructor with one parameter

        Q9 student2 = new Q9(101);

        student2.display();

        // Using constructor with two parameters

        Q9 student3 = new Q9(102, "John");

        student3.display();

        // Using constructor with three parameters

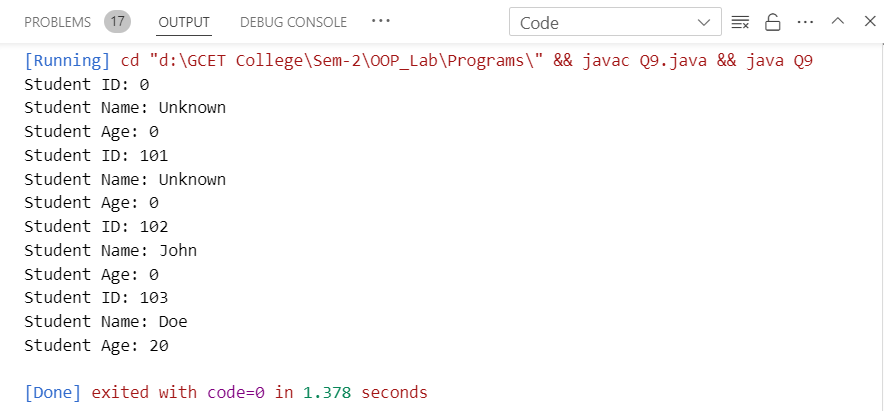
        Q9 student4 = new Q9(103, "Doe", 20);

        student4.display();

    }

}

Output🡪



Program 10: WAP in java for method overloading.

public class Q10 {

    // Method to add two integers

    public int add(int a, int b) {

        return a + b;

    }

    // Method to add three integers

    public int add(int a, int b, int c) {

        return a + b + c;

    }

    // Method to add two double numbers

    public double add(double a, double b) {

        return a + b;

    }

    // Method to concatenate two strings

    public String add(String a, String b) {

        return a + " " + b;

    }

    public static void main(String[] args) {

        Q10 calculator = new Q10();

        // Calling add() method with two integers

        System.out.println("Sum of 5 and 3: " + calculator.add(5, 3));

        // Calling add() method with three integers

        System.out.println("Sum of 5, 3, and 2: " + calculator.add(5, 3, 2));

        // Calling add() method with two double numbers

        System.out.println("Sum of 5.5 and 3.5: " + calculator.add(5.5, 3.5));

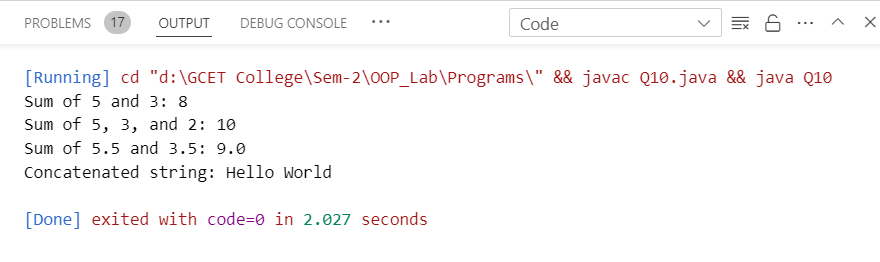
        // Calling add() method with two strings

        System.out.println("Concatenated string: " + calculator.add("Hello", "World"));

    }

}

Output🡪



Program 11: WAP in java for method overriding.

// Superclass

class Animal {

    void sound() {

        System.out.println("Animal makes a sound");

    }

}

// Subclass inheriting from Animal

class Dog extends Animal {

    // Overriding the sound method

    void sound() {

        System.out.println("Dog barks");

    }

}

// Subclass inheriting from Animal

class Cat extends Animal {

    // Overriding the sound method

    void sound() {

        System.out.println("Cat meows");

    }

}

public class Q11 {

    public static void main(String[] args) {

        Animal myAnimal = new Animal();  // Create a Animal object

        Animal myDog = new Dog();        // Create a Dog object

        Animal myCat = new Cat();        // Create a Cat object

        myAnimal.sound();  // Call the sound method on the Animal object

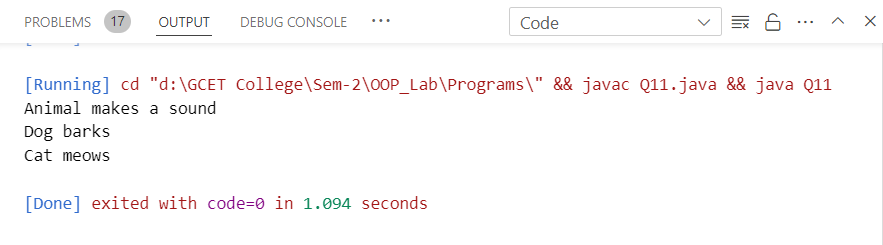
        myDog.sound();     // Call the sound method on the Dog object

        myCat.sound();     // Call the sound method on the Cat object

    }

}

Output🡪



Program 12: WAP in java to show run time polymorphism (up casting).

// Parent class

class Vehicle {

    void move() {

        System.out.println("Vehicle moves");

    }

}

// Child class extending Vehicle

class Car extends Vehicle {

    // Overriding the move method

    void move() {

        System.out.println("Car moves on the road");

    }

}

// Child class extending Vehicle

class Boat extends Vehicle {

    // Overriding the move method

    void move() {

        System.out.println("Boat moves on water");

    }

}

public class Q12 {

    public static void main(String[] args) {

        // Upcasting

        Vehicle myVehicle1 = new Car();  // Creating Car object and assigning to Vehicle reference

        Vehicle myVehicle2 = new Boat(); // Creating Boat object and assigning to Vehicle reference

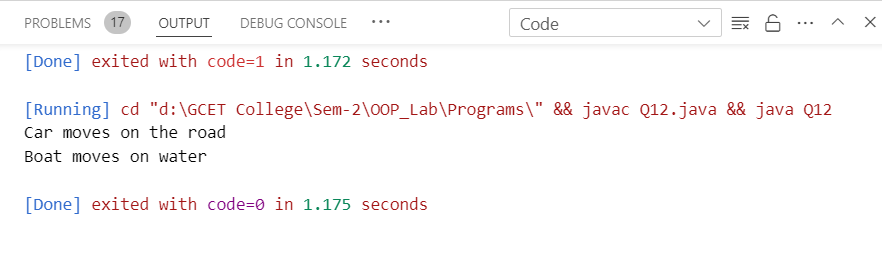
        myVehicle1.move();  // Calls Car's move method

        myVehicle2.move();  // Calls Boat's move method

    }

}

Output🡪



Program 13: WAP in java for access specifiers (all four).

public class Q13 {

    // Public access specifier

    public String publicVar = "Public variable";

    // Protected access specifier

    protected String protectedVar = "Protected variable";

    // Default access specifier

    String defaultVar = "Default variable";

    // Private access specifier

    private String privateVar = "Private variable";

    // Public method

    public void publicMethod() {

        System.out.println("This is a public method");

    }

    // Protected method

    protected void protectedMethod() {

        System.out.println("This is a protected method");

    }

    // Default method

    void defaultMethod() {

        System.out.println("This is a default method");

    }

    // Private method

    private void privateMethod() {

        System.out.println("This is a private method");

    }

    public static void main(String[] args) {

        Q13 obj = new Q13();

        // Accessing variables and methods using object

        System.out.println(obj.publicVar);

        System.out.println(obj.protectedVar);

        System.out.println(obj.defaultVar);

        // System.out.println(obj.privateVar);  // This will throw compilation error because privateVar is private

        obj.publicMethod();

        obj.protectedMethod();

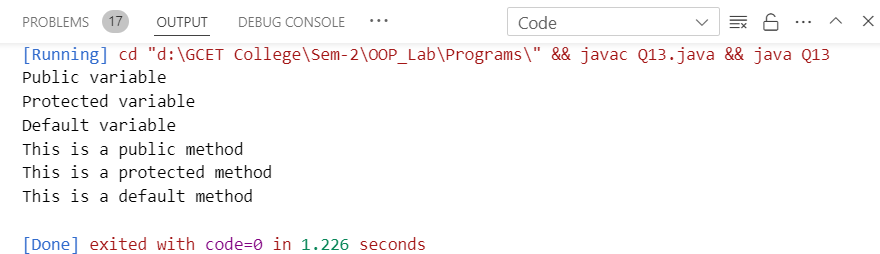
        obj.defaultMethod();

        // obj.privateMethod();  // This will throw compilation error because privateMethod is private

    }

}

Output🡪



Program 14: WAP in java to implement the single dimension array.

public class Q14 {

    public static void main(String[] args) {

        // Declare an array of integers

        int[] numbers = new int[5];

        // Assign values to the array

        numbers[0] = 1;

        numbers[1] = 2;

        numbers[2] = 3;

        numbers[3] = 4;

        numbers[4] = 5;

        // Print the array elements

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

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

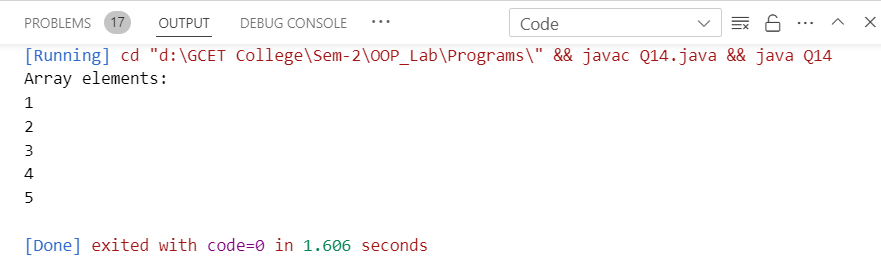
            System.out.println(numbers[i]);

        }

    }

}

Output🡪



Program 15: WAP in java to copy the elements from one array to another array.

public class Q15 {

    public static void main(String[] args) {

        // Declare and initialize an array

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

        // Create a destination array with the same length as sourceArray

        int[] destinationArray = new int[sourceArray.length];

        // Copy elements from sourceArray to destinationArray

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

            destinationArray[i] = sourceArray[i];

        }

        // Print the source and destination arrays

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

        printArray(sourceArray);

        System.out.println("\nDestination Array:");

        printArray(destinationArray);

    }

    // Method to print an array

    public static void printArray(int[] arr) {

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

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

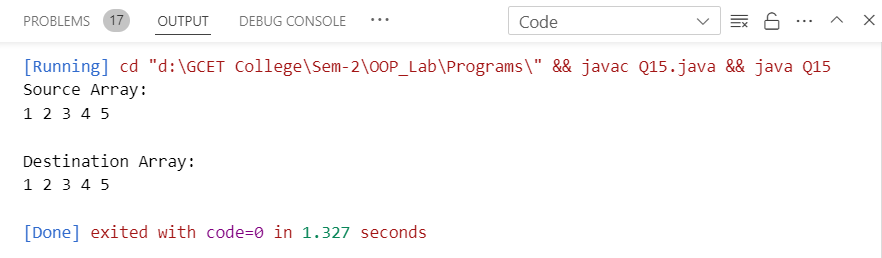
        }

        System.out.println(); // Move to the next line

    }

}

Output🡪



Program 16: WAP in java to perform the addition and multiplication in 2-D array.

public class Q16 {

    public static void main(String[] args) {

        int[][] arr1 = {

            {1, 2, 3},

            {4, 5, 6},

            {7, 8, 9}

        };

        int[][] arr2 = {

            {9, 8, 7},

            {6, 5, 4},

            {3, 2, 1}

        };

        // Perform addition

        int[][] sum = addArrays(arr1, arr2);

        System.out.println("Sum of the arrays:");

        printArray(sum);

        // Perform multiplication

        int[][] product = multiplyArrays(arr1, arr2);

        System.out.println("\nProduct of the arrays:");

        printArray(product);

    }

    // Method to add two 2-D arrays

    public static int[][] addArrays(int[][] arr1, int[][] arr2) {

        int[][] result = new int[arr1.length][arr1[0].length];

        for (int i = 0; i < arr1.length; i++) {

            for (int j = 0; j < arr1[0].length; j++) {

                result[i][j] = arr1[i][j] + arr2[i][j];

            }

        }

        return result;

    }

    // Method to multiply two 2-D arrays

    public static int[][] multiplyArrays(int[][] arr1, int[][] arr2)

{

        int[][] result = new int[arr1.length][arr2[0].length];

        for (int i = 0; i < arr1.length; i++) {

            for (int j = 0; j < arr2[0].length; j++) {

                for (int k = 0; k < arr2.length; k++) {

                    result[i][j] += arr1[i][k] \* arr2[k][j];

                }

            }

        }

        return result;

    }

    // Method to print a 2-D array

    public static void printArray(int[][] arr) {

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

            for (int j = 0; j < arr[i].length; j++) {

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

            }

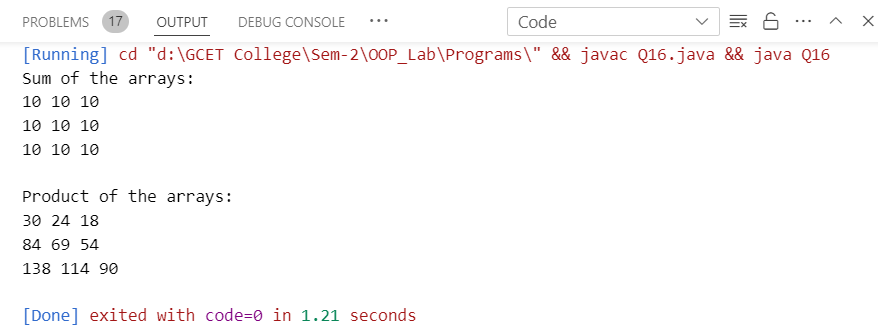
            System.out.println();

        }

    }

}

Output🡪



Program 17: WAP in java for simple inheritance.

// Parent class

class Animal {

    void eat() {

        System.out.println("This animal eats food.");

    }

}

// Child class inheriting from Animal

class Dog extends Animal {

    void bark() {

        System.out.println("The dog barks.");

    }

}

public class Q17 {

    public static void main(String[] args) {

        // Creating an object of the child class

        Dog dog = new Dog();

        // Calling methods from both parent and child class

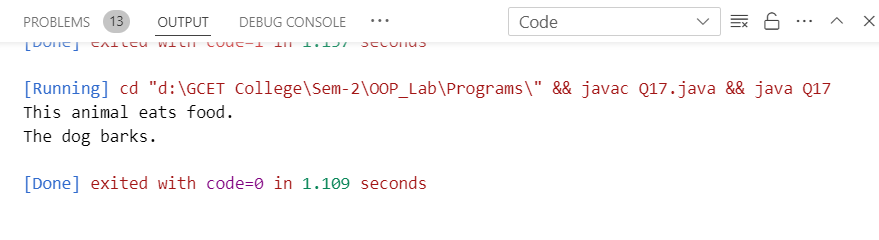
        dog.eat();  // inherited method from Animal class

        dog.bark(); // method of Dog class

    }

}

Output🡪



Program 18: WAP in java for Final Keyword.

public class Q18 {

    public static void main(String[] args) {

        final int x = 10;

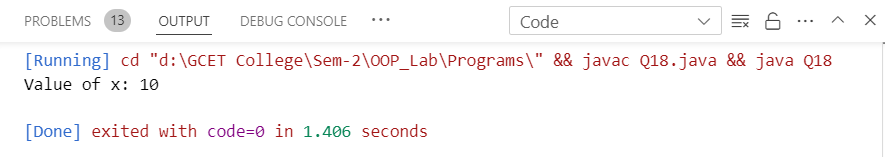
        // x = 20; // Error: Cannot assign a value to final variable x

        System.out.println("Value of x: " + x);

    }

}

Output🡪



Program 19: WAP in java for Super Keyword.

class Parent {

    Parent() {

        System.out.println("This is Parent class constructor.");

    }

}

class Child extends Parent {

    Child() {

        super(); // Calling Parent class constructor

        System.out.println("This is Child class constructor.");

    }

}

public class Q19 {

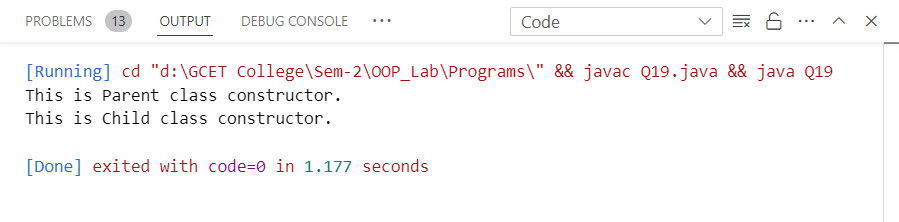
    public static void main(String[] args) {

        Child childObj = new Child();

    }

}

Output🡪



Program 20: WAP in java for chaining constructor.

public class Q20 {

    private String name;

    private int age;

    // Default constructor

    public Q20() {

        this("Unknown", 0);  // Calling parameterized constructor

    }

    // Parameterized constructor

    public Q20(String name, int age) {

        this.name = name;

        this.age = age;

    }

    public void display() {

        System.out.println("Name: " + name);

        System.out.println("Age: " + age);

    }

    public static void main(String[] args) {

        Q20 person1 = new Q20();  // Calls default constructor

        Q20 person2 = new Q20("John", 30);  // Calls parameterized constructor

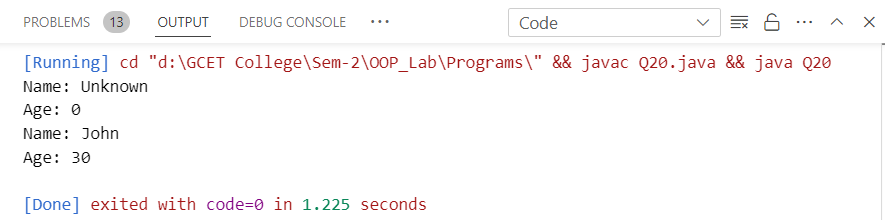
        person1.display();

        person2.display();

    }

}

Output🡪



Program 21: WAP in java for abstract method and abstract class.

// Abstract class

abstract class Shape {

    // Abstract method (no implementation)

    abstract void draw();

    // Concrete method

    void display() {

        System.out.println("Displaying shape...");

    }

}

// Concrete subclass of Shape

class Circle extends Shape {

    // Implementing abstract method

    void draw() {

        System.out.println("Drawing circle...");

    }

}

// Concrete subclass of Shape

class Rectangle extends Shape {

    // Implementing abstract method

    void draw() {

        System.out.println("Drawing rectangle...");

    }

}

public class Q21 {

    public static void main(String[] args) {

        Shape shape1 = new Circle();  // Upcasting

        Shape shape2 = new Rectangle();  // Upcasting

        shape1.draw();

        shape1.display();

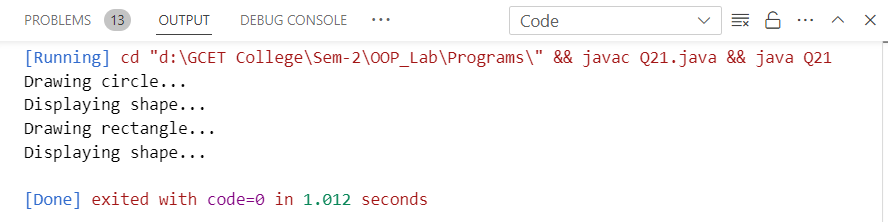
        shape2.draw();

        shape2.display();

    }

}

Output🡪



Program 22: WAP in java for interface.

// Define an interface

interface Animal {

    // Abstract method (does not have a body)

    void sound();

    // Default method

    default void eat() {

        System.out.println("This animal eats food.");

    }

    // Static method

    static void sleep() {

        System.out.println("This animal sleeps.");

    }

}

// Implementing the Animal interface

class Dog implements Animal {

    // Implementing sound method

    public void sound() {

        System.out.println("Dog barks");

    }

    // Overriding the default eat method

    public void eat() {

        System.out.println("Dog eats bones.");

    }

}

// Implementing the Animal interface

class Cat implements Animal {

    // Implementing sound method

    public void sound() {

        System.out.println("Cat meows");

    }

}

public class Q22 {

    public static void main(String[] args) {

        Animal myDog = new Dog();  // Upcasting

        Animal myCat = new Cat();  // Upcasting

        myDog.sound();

        myDog.eat();  // Overridden method

        Animal.sleep();  // Static method

        myCat.sound();

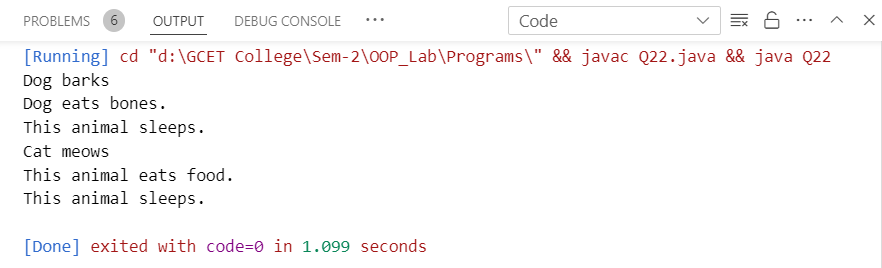
        myCat.eat();  // Default method

        Animal.sleep();  // Static method

    }

}

Output🡪



Program 23: WAP in java for multiple inheritance.

// Interface for Animal

interface Animal {

    void eat();

}

// Interface for Bird

interface Bird {

    void fly();

}

// Class implementing multiple interfaces

class Sparrow implements Animal, Bird {

    @Override

    public void eat() {

        System.out.println("Sparrow eats insects.");

    }

    @Override

    public void fly() {

        System.out.println("Sparrow flies high.");

    }

}

public class Q23 {

    public static void main(String[] args) {

        Sparrow sparrow = new Sparrow();

        // Calling methods

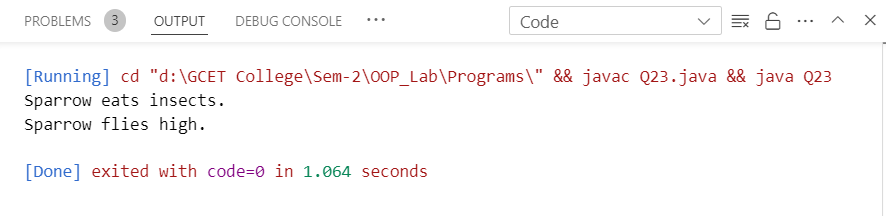
        sparrow.eat();

        sparrow.fly();

    }

}

Output🡪



Program 24: WAP in java for Object Cloning(shallow and deep copy).

// Class to be cloned

class Student implements Cloneable {

    String name;

    Address address;

    public Student(String name, Address address) {

        this.name = name;

        this.address = address;

    }

    // Shallow copy

    protected Object clone() throws CloneNotSupportedException {

        return super.clone();

    }

    // Deep copy

    public Student deepClone() {

        Address clonedAddress = new Address(this.address.street, this.address.city);

        return new Student(this.name, clonedAddress);

    }

    @Override

    public String toString() {

        return "Student [name=" + name + ", address=" + address + "]";

    }

}

// Address class for deep copy

class Address {

    String street;

    String city;

    public Address(String street, String city) {

        this.street = street;

        this.city = city;

    }

    @Override

    public String toString() {

        return "Address [street=" + street + ", city=" + city + "]";

    }

}

public class Q24 {

    public static void main(String[] args) throws CloneNotSupportedException {

        Address address = new Address("123 Main St", "City");

        Student originalStudent = new Student("John", address);

        // Shallow copy

        Student clonedStudent = (Student) originalStudent.clone();

        System.out.println("Shallow Copy: ");

        System.out.println("Original: " + originalStudent);

        System.out.println("Cloned: " + clonedStudent);

        // Deep copy

        Student deepClonedStudent = originalStudent.deepClone();

        System.out.println("\nDeep Copy: ");

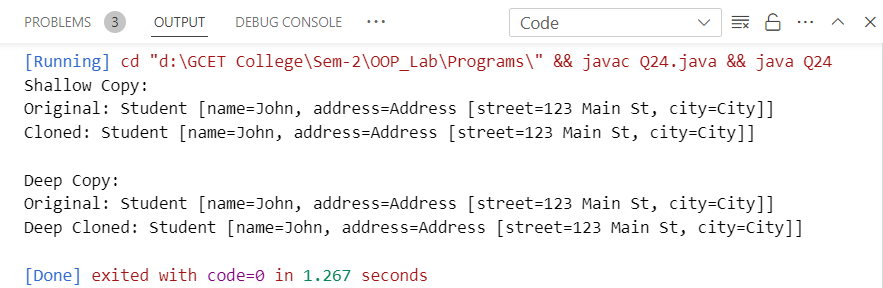
        System.out.println("Original: " + originalStudent);

        System.out.println("Deep Cloned: " + deepClonedStudent);

    }

}

Output🡪



Program 25: WAP in java for Inner Classes (all types).

public class Q25 {

    // Nested Inner Class (Static Inner Class)

    static class StaticInnerClass {

        void display() {

            System.out.println("Inside Static Inner Class");

        }

    }

    // Non-static Nested Inner Class (Inner Class)

    class InnerClass {

        void display() {

            System.out.println("Inside Non-static Inner Class");

        }

    }

    // Method with Local Inner Class

    void localInnerClassExample() {

        class LocalInner {

            void display() {

                System.out.println("Inside Local Inner Class");

            }

        }

        LocalInner localInner = new LocalInner();

        localInner.display();

    }

    // Method with Anonymous Inner Class

    void anonymousInnerClassExample() {

        Thread t = new Thread(new Runnable() {

            @Override

            public void run() {

                System.out.println("Inside Anonymous Inner Class");

            }

        });

        t.start();

    }

    public static void main(String[] args) {

        // Accessing Static Inner Class

        Q25.StaticInnerClass staticInnerObj = new Q25.StaticInnerClass();

        staticInnerObj.display();

        // Accessing Non-static Inner Class

        Q25 outerObj = new Q25();

        Q25.InnerClass innerObj = outerObj.new InnerClass();

        innerObj.display();

        // Accessing Local Inner Class

        outerObj.localInnerClassExample();

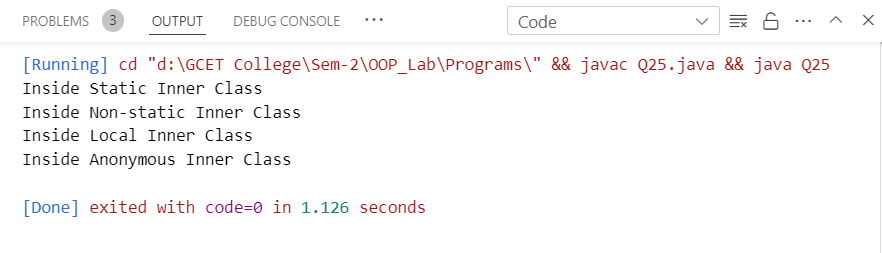
        // Accessing Anonymous Inner Class

        outerObj.anonymousInnerClassExample();

    }

}

Output🡪



Program 26: WAP in java to create the package (user defined package).

File 1:

// Here Package same name as directory is created with class HelloWorld same name as file name

// This file is located in directory 'MyPackage'

// This directory is created where the jdk is located, inside the 'bin' directory

package MyPackage;

public class HelloWorld {

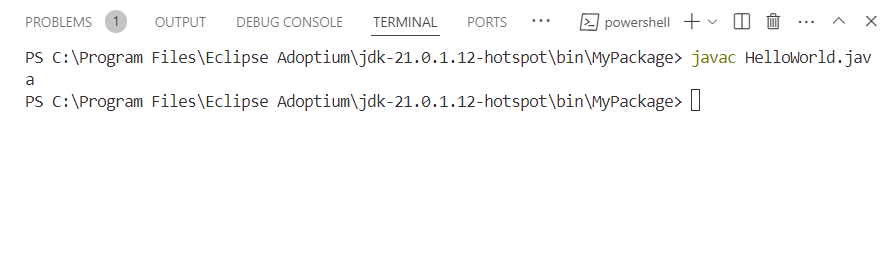
    public void sayHello() {

        System.out.println("Hello, World!");

    }

}

Output🡪



File 2:

// This file is located in bin directory

// Here package is imported

import MyPackage.HelloWorld;

public class Main {

    public static void main(String[] args) {

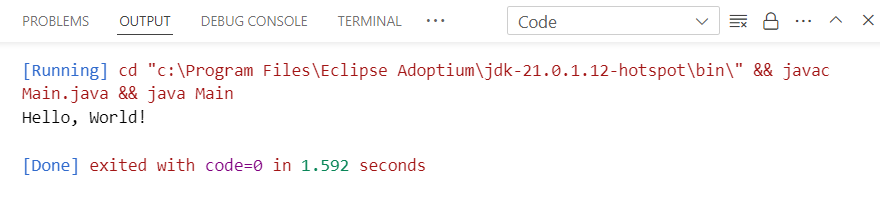
        HelloWorld hello = new HelloWorld();

        hello.sayHello();

    }

}

Output🡪



Program 27: WAP in java for exception handling by using try, catch and finally.

public class Q27 {

    public static void main(String[] args) {

        try {

            // Code that may throw an exception

            int result = divide(10, 0);

            System.out.println("Result: " + result);

        } catch (ArithmeticException e) {

            // Catching specific exception

            System.out.println("Exception caught: Division by zero");

        } finally {

            // Code that will always execute, regardless of whether an exception occurred or not

            System.out.println("Finally block executed");

        }

    }

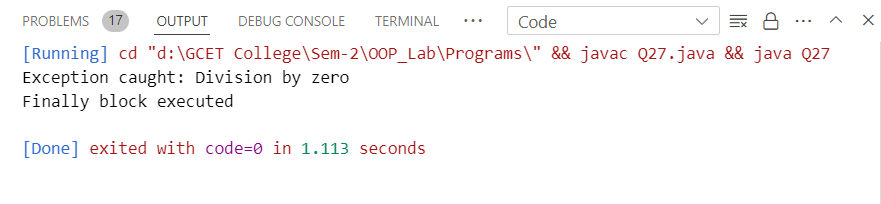
    public static int divide(int dividend, int divisor) {

        return dividend / divisor;

    }

}

Output🡪



Program 28: WAP in java for throw and throws Exception.

public class Q28 {

    public static void main(String[] args) {

        try {

            validateAge(15); // Calling a method that throws an exception

        } catch (InvalidAgeException e) {

            System.out.println("Caught exception: " + e.getMessage());

        }

    }

    // Method that throws a custom exception

    public static void validateAge(int age) throws InvalidAgeException {

        if (age < 18) {

            throw new InvalidAgeException("Age must be 18 or above.");

        } else {

            System.out.println("Age is valid: " + age);

        }

    }

}

// Custom exception class

class InvalidAgeException extends Exception {

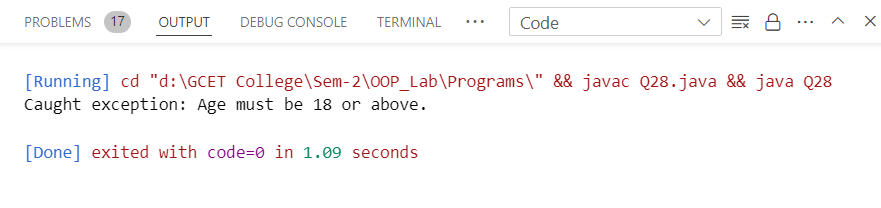
    public InvalidAgeException(String message) {

        super(message);

    }

}

Output🡪



Program 29: WAP in java to throw your own Exceptions.

public class Q29 {

    public static void main(String[] args) {

        try {

            // Simulating a condition where a custom exception needs to be thrown

            int balance = 100;

            int amountToWithdraw = 200;

            withdrawMoney(balance, amountToWithdraw);

        } catch (InsufficientBalanceException e) {

            System.out.println("Caught exception: " + e.getMessage());

        }

    }

    // Method that throws a custom exception if the balance is insufficient

    public static void withdrawMoney(int balance, int amount) throws InsufficientBalanceException {

        if (balance < amount) {

            throw new InsufficientBalanceException("Insufficient balance in the account.");

        } else {

            System.out.println("Withdrawal successful. Remaining balance: " + (balance - amount));

        }

    }

}

// Custom exception class for insufficient balance

class InsufficientBalanceException extends Exception {

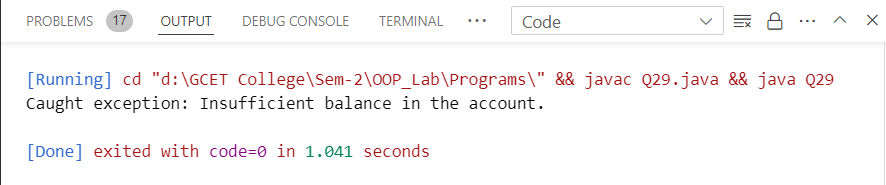
    public InsufficientBalanceException(String message) {

        super(message);

    }

}

Output🡪



Program 30: WAP in java to reading and writing in file using byte stream.

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class Q30 {

    public static void main(String[] args) {

        String fileName = "example.txt";

        String content = "Hello, this is a sample text.";

        // Writing to a file using FileOutputStream

        try (FileOutputStream outputStream = new FileOutputStream(fileName)) {

            byte[] bytes = content.getBytes();

            outputStream.write(bytes);

            System.out.println("Content written to file successfully.");

        } catch (IOException e) {

            System.out.println("Error writing to file: " + e.getMessage());

        }

        // Reading from a file using FileInputStream

        try (FileInputStream inputStream = new FileInputStream(fileName)) {

            int ch;

            System.out.println("Content read from file:");

            while ((ch = inputStream.read()) != -1) {

                System.out.print((char) ch);

            }

        } catch (IOException e) {

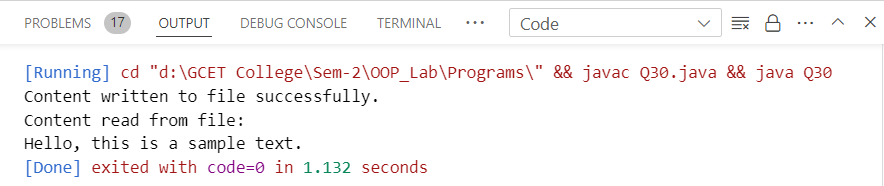
            System.out.println("Error reading from file: " + e.getMessage());

        }

    }

}

Output🡪



Program 31: WAP in java to reading and writing in file using character stream.

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class Q31 {

    public static void main(String[] args) {

        String fileName = "example.txt";

        String content = "Hello, this is a sample text.";

        // Writing to a file using FileWriter

        try (FileWriter writer = new FileWriter(fileName)) {

            writer.write(content);

            System.out.println("Content written to file successfully.");

        } catch (IOException e) {

            System.out.println("Error writing to file: " + e.getMessage());

        }

        // Reading from a file using FileReader

        try (FileReader reader = new FileReader(fileName)) {

            int ch;

            System.out.println("Content read from file:");

            while ((ch = reader.read()) != -1) {

                System.out.print((char) ch);

            }

        } catch (IOException e) {

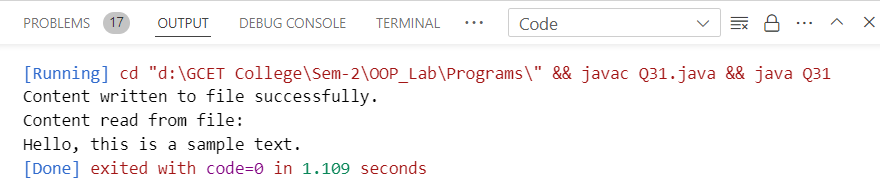
            System.out.println("Error reading from file: " + e.getMessage());

        }

    }

}

Output🡪



Program 32: WAP in java to reading and writing through console class.

import java.util.Scanner;

public class Q32 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Reading input from the console

        System.out.print("Enter your name: ");

        String name = scanner.nextLine();

        // Writing output to the console

        System.out.println("Hello, " + name + "! Welcome to the console I/O example.");

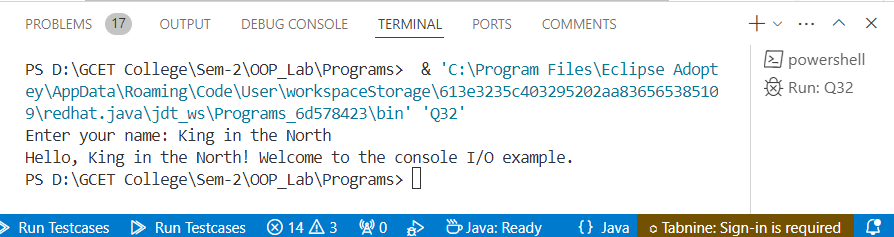
        // Closing the scanner

        scanner.close();

    }

}

Output🡪



Program 33: WAP in java how to create thread using Thread Class.

public class Q33 extends Thread {

    public void run() {

        // Code to be executed by the thread

        for (int i = 0; i < 5; i++) {

            System.out.println("Thread running: " + i);

            try {

                // Sleep for 1 second

                Thread.sleep(1000);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

    public static void main(String[] args) {

        // Create an instance of Q33

        Q33 thread = new Q33();

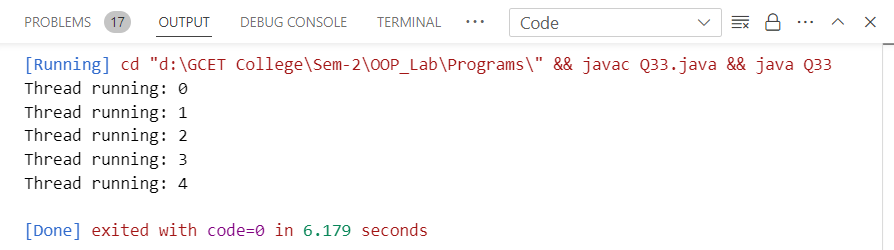
        // Start the thread

        thread.start();

    }

}

Output🡪



Program 34: WAP in java how to create thread using runnable interface.

public class Q34 implements Runnable {

    public void run() {

        // Code to be executed by the thread

        for (int i = 0; i < 5; i++) {

            System.out.println("Thread running: " + i);

            try {

                // Sleep for 1 second

                Thread.sleep(1000);

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

    public static void main(String[] args) {

        // Create an instance of Q34

        Q34 myRunnable = new Q34();

        // Create a Thread object with myRunnable as the target

        Thread thread = new Thread(myRunnable);

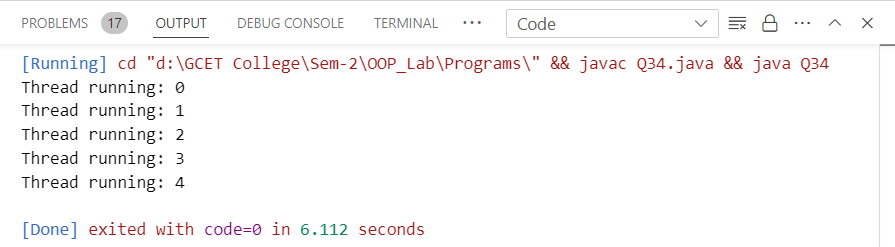
        // Start the thread

        thread.start();

    }

}

Output🡪



Program 35: WAP in java to implement multithreading.

// Runnable implementation

class MyRunnable implements Runnable {

    public void run() {

        for (int i = 1; i <= 5; i++) {

            System.out.println("Thread " + Thread.currentThread().getId() + ": " + i);

            try {

                Thread.sleep(1000); // Sleep for 1 second

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

    }

}

public class Q35 {

    public static void main(String[] args) {

        // Creating and starting threads using Runnable interface

        Thread thread1 = new Thread(new MyRunnable());

        Thread thread2 = new Thread(new MyRunnable());

        thread1.start();

        thread2.start();

        // Creating and starting threads using Thread class

        MyThread myThread1 = new MyThread();

        MyThread myThread2 = new MyThread();

        myThread1.start();

        myThread2.start();

    }

}

// Thread class extension

class MyThread extends Thread {

    public void run() {

        for (int i = 1; i <= 5; i++) {

            System.out.println("Thread " + Thread.currentThread().getId() + ": " + i);

            try {

                Thread.sleep(1000); // Sleep for 1 second

            } catch (InterruptedException e) {

                e.printStackTrace();

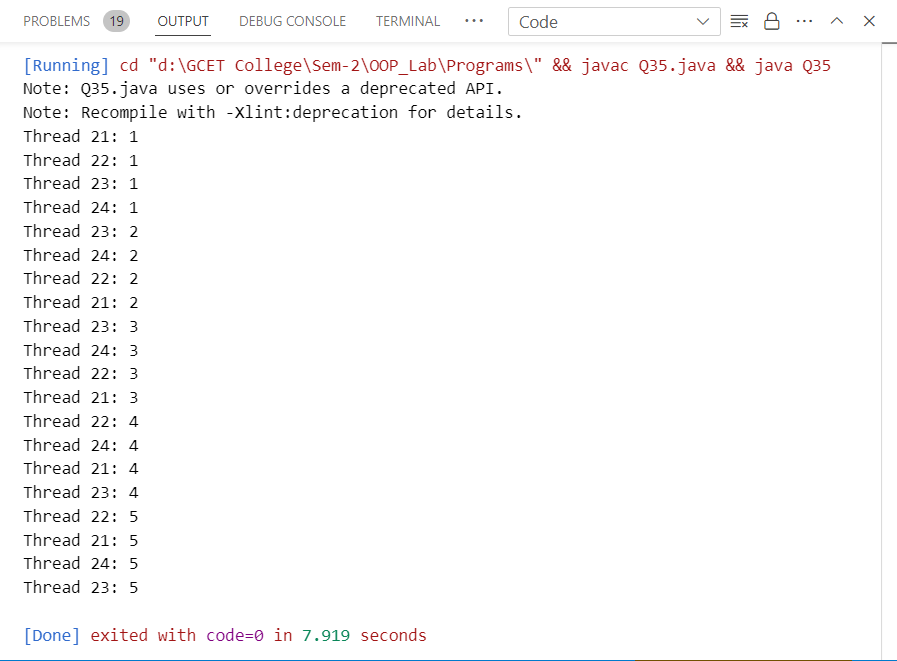
            }

        }

    }

}

Output🡪



Program 36: WAP in java to achieve synchronization in threads.

class Counter {

    private int count = 0;

    // Synchronized method to increment the count

    public synchronized void increment() {

        count++;

    }

    // Method to get the current count

    public int getCount() {

        return count;

    }

}

class MyThread extends Thread {

    private Counter counter;

    public MyThread(Counter counter) {

        this.counter = counter;

    }

    public void run() {

        for (int i = 0; i < 1000; i++) {

            counter.increment();

        }

    }

}

public class Q36 {

    public static void main(String[] args) {

        Counter counter = new Counter();

        MyThread thread1 = new MyThread(counter);

        MyThread thread2 = new MyThread(counter);

        thread1.start();

        thread2.start();

        try {

            thread1.join();

            thread2.join();

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

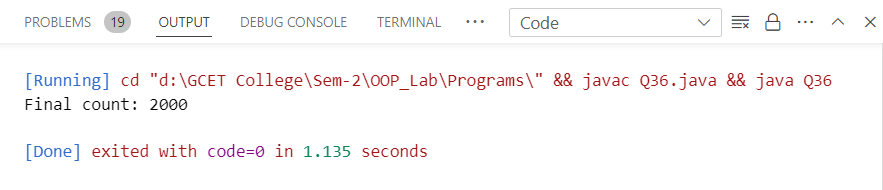
        // The expected value of count should be 2000

        System.out.println("Final count: " + counter.getCount());

    }

}

Output🡪



Program 37: WAP in java to implement the concept of Priorities of threads.

class MyThread extends Thread {

    public MyThread(String name) {

        super(name);

    }

    public void run() {

        System.out.println("Thread: " + getName() + " Priority: " + getPriority());

    }

}

public class Q37 {

    public static void main(String[] args) {

        // Create three threads with different priorities

        MyThread thread1 = new MyThread("Thread 1");

        MyThread thread2 = new MyThread("Thread 2");

        MyThread thread3 = new MyThread("Thread 3");

        // Set priorities for threads

        thread1.setPriority(Thread.MIN\_PRIORITY);

        thread2.setPriority(Thread.NORM\_PRIORITY);

        thread3.setPriority(Thread.MAX\_PRIORITY);

        // Start the threads

        thread1.start();

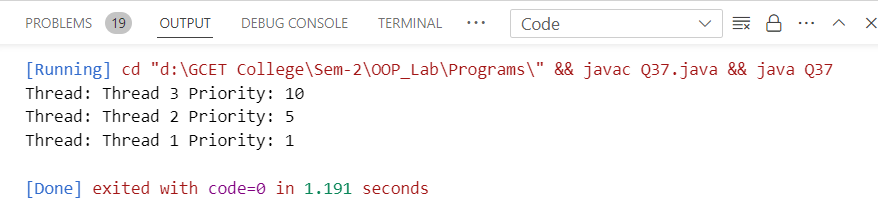
        thread2.start();

        thread3.start();

    }

}

Output🡪



Program 38: WAP in java to illustrate the concept of Generic Programming.

// A generic class representing a generic box that can hold any type of object

class Box<T> {

    private T item;

    public void setItem(T item) {

        this.item = item;

    }

    public T getItem() {

        return item;

    }

    public void displayItemType() {

        System.out.println("Type of item in the box: " + item.getClass().getName());

    }

}

public class Q38 {

    public static void main(String[] args) {

        // Create a Box to hold an integer

        Box<Integer> intBox = new Box<>();

        intBox.setItem(123);

        System.out.println("Item in the integer box: " + intBox.getItem());

        intBox.displayItemType();

        // Create a Box to hold a string

        Box<String> stringBox = new Box<>();

        stringBox.setItem("Hello, Generics!");

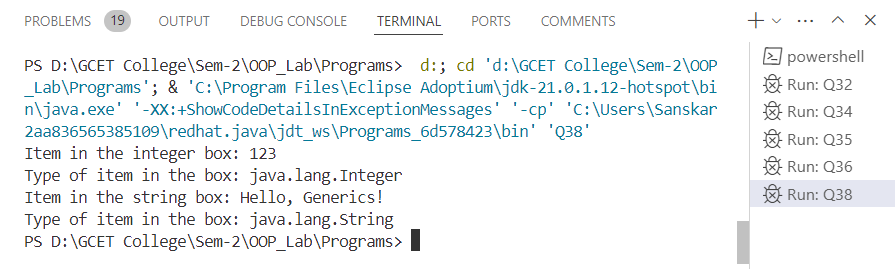
        System.out.println("Item in the string box: " + stringBox.getItem());

        stringBox.displayItemType();

    }

}

Output🡪



Program 39: WAP in java to illustrate the concept of event handling (using various event handlers).

import java.awt.\*;

import java.awt.event.\*;

public class Q39 extends Frame implements ActionListener, MouseListener, KeyListener {

    private TextField textField;

    private Label label;

    public Q39() {

        // Create components

        label = new Label("Click inside the window and press any key:");

        textField = new TextField(20);

        // Add components to the frame

        add(label, BorderLayout.NORTH);

        add(textField, BorderLayout.SOUTH);

        // Add event listeners

        addMouseListener(this);

        addKeyListener(this);

        // Set frame properties

        setTitle("Event Handling Example");

        setSize(300, 200);

        setVisible(true);

    }

    // ActionListener event handler

    public void actionPerformed(ActionEvent e) {

        textField.setText("Button clicked!");

    }

    // MouseListener event handlers

    public void mouseClicked(MouseEvent e) {

        textField.setText("Mouse clicked at (" + e.getX() + ", " + e.getY() + ")");

    }

    public void mouseEntered(MouseEvent e) {}

    public void mouseExited(MouseEvent e) {}

    public void mousePressed(MouseEvent e) {}

    public void mouseReleased(MouseEvent e) {}

    // KeyListener event handlers

    public void keyPressed(KeyEvent e) {

        textField.setText("Key pressed: " + e.getKeyChar());

    }

    public void keyReleased(KeyEvent e) {}

    public void keyTyped(KeyEvent e) {}

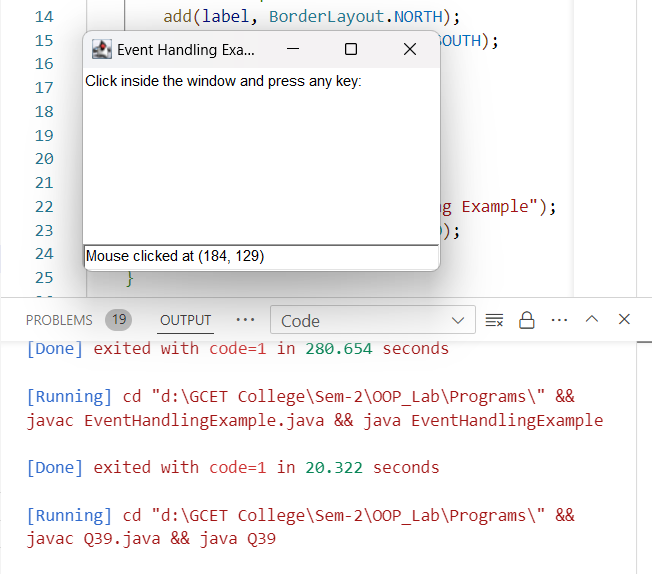
    public static void main(String[] args) {

        new Q39();

    }

}

Output🡪



Program 39: Create a simple registration application using various swing components (like: JFrame, JButton, JLabel, text fields, text areas, check box and ratio buttons).

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class RegistrationApp extends JFrame implements ActionListener {

    private JTextField firstNameField, lastNameField;

    private JRadioButton maleRadioButton, femaleRadioButton;

    private JTextArea addressArea;

    private JCheckBox termsCheckBox;

    private JButton registerButton;

    public RegistrationApp() {

        // Set frame properties

        setTitle("Registration Application");

        setSize(400, 300);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLayout(new BorderLayout());

        // Create components

        JPanel formPanel = new JPanel(new GridLayout(6, 2));

        JLabel firstNameLabel = new JLabel("First Name:");

        firstNameField = new JTextField();

        JLabel lastNameLabel = new JLabel("Last Name:");

        lastNameField = new JTextField();

        JLabel genderLabel = new JLabel("Gender:");

        maleRadioButton = new JRadioButton("Male");

        femaleRadioButton = new JRadioButton("Female");

        ButtonGroup genderGroup = new ButtonGroup();

        genderGroup.add(maleRadioButton);

        genderGroup.add(femaleRadioButton);

        JLabel addressLabel = new JLabel("Address:");

        addressArea = new JTextArea(4, 20);

        JScrollPane scrollPane = new JScrollPane(addressArea);

        JLabel termsLabel = new JLabel("Accept Terms and Conditions:");

        termsCheckBox = new JCheckBox();

        registerButton = new JButton("Register");

        // Add components to the form panel

        formPanel.add(firstNameLabel);

        formPanel.add(firstNameField);

        formPanel.add(lastNameLabel);

        formPanel.add(lastNameField);

        formPanel.add(genderLabel);

        formPanel.add(maleRadioButton);

        formPanel.add(new JLabel()); // Empty space

        formPanel.add(femaleRadioButton);

        formPanel.add(addressLabel);

        formPanel.add(scrollPane);

        formPanel.add(termsLabel);

        formPanel.add(termsCheckBox);

        // Add register button action listener

        registerButton.addActionListener(this);

        // Add components to the frame

        add(formPanel, BorderLayout.CENTER);

        add(registerButton, BorderLayout.SOUTH);

        // Set frame visible

        setVisible(true);

    }

    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == registerButton) {

            // Perform registration process

            String firstName = firstNameField.getText();

            String lastName = lastNameField.getText();

            String gender = maleRadioButton.isSelected() ? "Male" : "Female";

            String address = addressArea.getText();

            boolean acceptedTerms = termsCheckBox.isSelected();

            // Display registration information

            String message = "Registration Successful!\n\n";

            message += "First Name: " + firstName + "\n";

            message += "Last Name: " + lastName + "\n";

            message += "Gender: " + gender + "\n";

            message += "Address: " + address + "\n";

            message += "Terms and Conditions Accepted: " + (acceptedTerms ? "Yes" : "No");

            JOptionPane.showMessageDialog(this, message, "Registration Info", JOptionPane.INFORMATION\_MESSAGE);

        }

    }

    public static void main(String[] args) {

        new RegistrationApp();

    } }

Output🡪

