

**Object oriented programming lab file**

**Submitted by: Submitted to:**

Himanshu Bhaker Prof. Amit Panwar

Batch: 11 (CSF)

Sap Id: 500123230

# **Experiment -1**

TITLE: Introduction to Java Environment.

1. Explore and understand the role of JDK, JRE and JVM.

### **Role of JDK, JRE, and JVM**

The **Java Development Kit (JDK)**, **Java Runtime Environment (JRE)**, and **Java Virtual Machine (JVM)** are essential components in the Java programming ecosystem. Each has a distinct role in developing, compiling, and running Java applications.

## **1. Java Virtual Machine (JVM)**

The **JVM** is an abstract machine that provides the runtime environment required to execute Java bytecode. It is platform-dependent, meaning each operating system has its own implementation.

### **Key Functions of JVM:**

* **Loading**: Loads compiled .class files into memory.
* **Execution**: Executes Java bytecode instructions.
* **Memory Management**: Uses features like the **Garbage Collector (GC)** to automatically manage memory.
* **Security**: Provides a secure environment using features like the **ClassLoader** and **Security Manager**.
* **Just-In-Time (JIT) Compiler**: Enhances performance by converting bytecode into native machine code at runtime.

## **2. Java Runtime Environment (JRE)**

The **JRE** is a package that provides the libraries and environment needed to run Java applications. It includes the **JVM** and core libraries but does not contain development tools like compilers.

### **Key Components of JRE:**

* **JVM**: To run Java programs.
* **Core Libraries**: Provides essential Java classes (e.g., java.lang, java.util).
* **Runtime Libraries**: Includes essential APIs and supporting files for Java application execution.

**Role:** The JRE is required for running Java applications but not for developing them.

## **3. Java Development Kit (JDK)**

The **JDK** is a complete software development kit that includes the JRE, JVM, and essential development tools. It is necessary for writing, compiling, and debugging Java programs.

### **Key Components of JDK:**

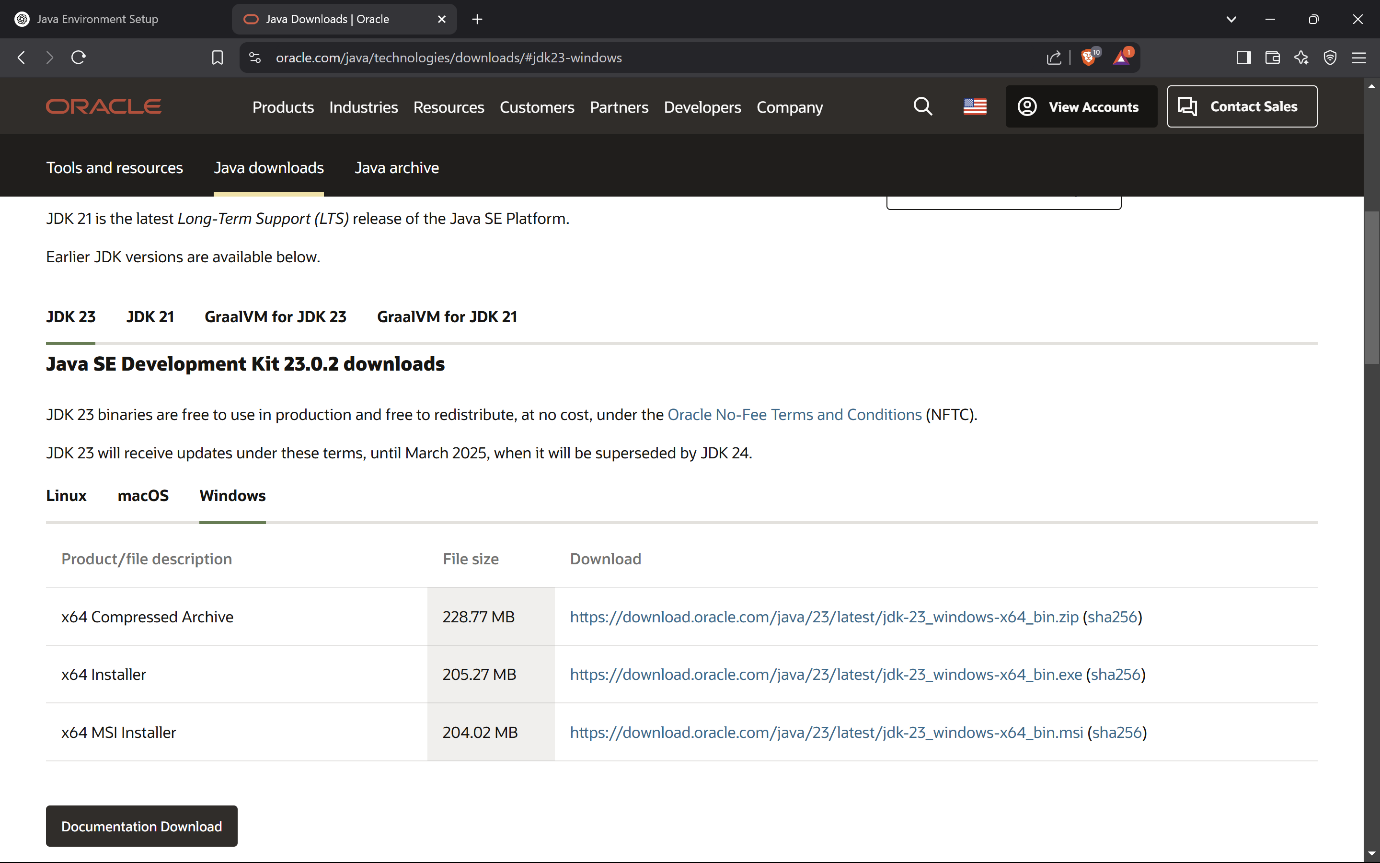
* **JRE**: Includes JVM and core libraries.
* **Compiler (javac)**: Translates Java source code into bytecode.
* **Debugger (jdb)**: For detecting and fixing code errors.
* **JavaDoc (javadoc)**: For generating documentation from comments in the source code.
* **Other Tools**: Includes utilities like jar, java, keytool, etc.

**Role:** The JDK is essential for developing and compiling Java programs.

2. Install latest available JDK and verify the Java Environment.

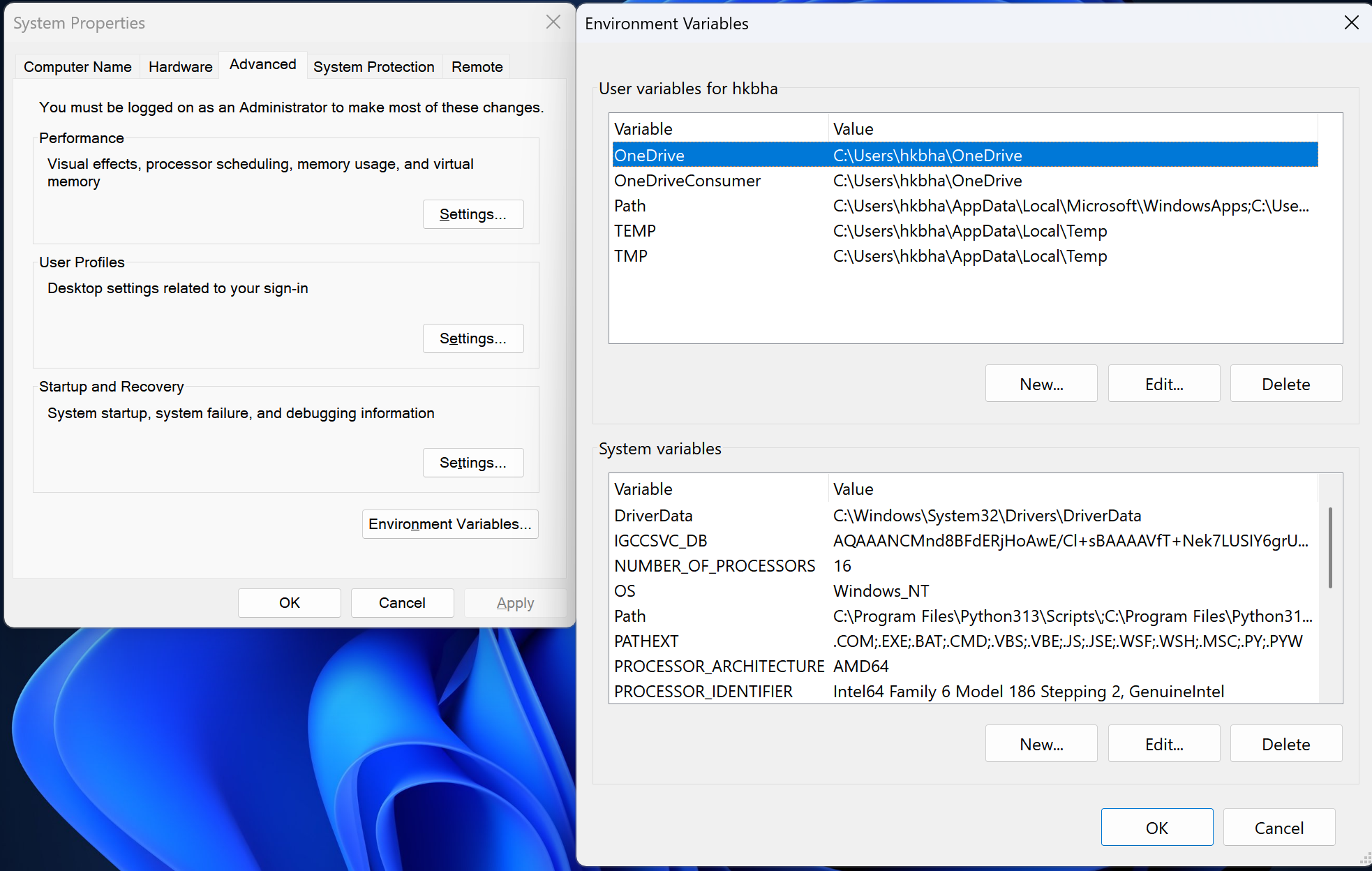
## Steps to install JDK:

1. Download the latest JDK from oracle website.



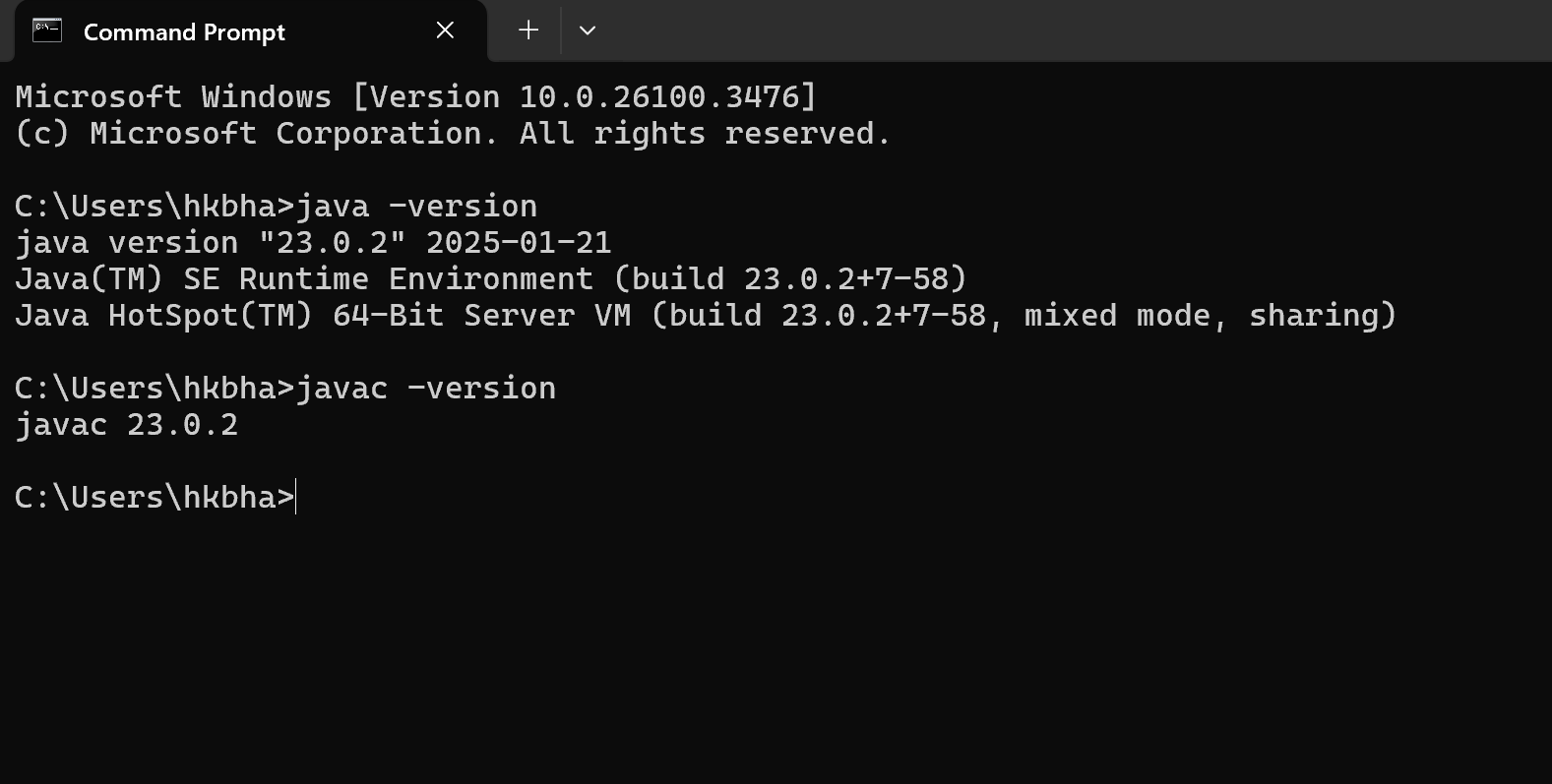
1. Run the installer and follow the instructions.
2. After installation, configure the environment variables.

Add C:\Program Files\Java\jdk-<version>\bin to the **Path** variable in System Properties.



### To verify installation:

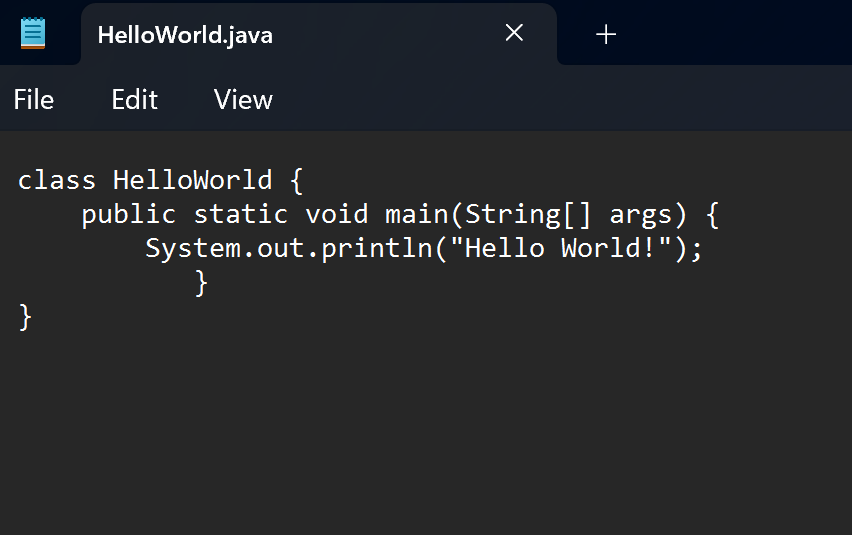
1. Open command prompt and type:



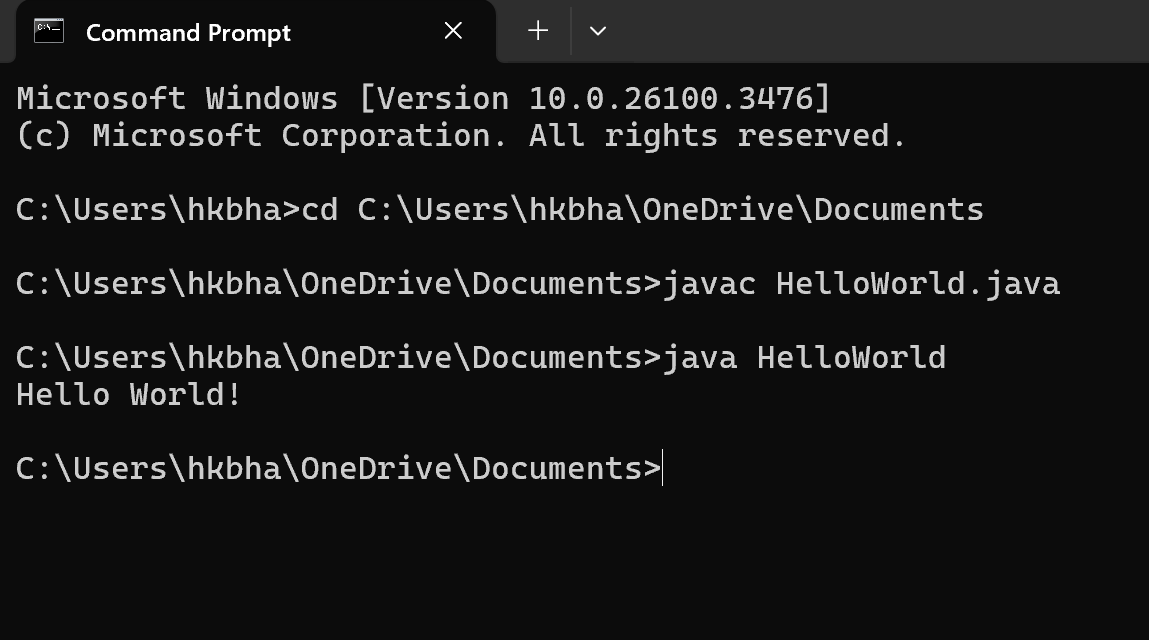
Both command returns version details, your environment is correctly set up.

3. Create a Sample Hello World Program using simple text editor (e.g. Notepad) and show the steps to compile and execute the program using command prompt.

* Step 1: Open notepad and write the code



* Step 2: Save the file as HelloWorld.java
* Step 3: Open Command Prompt
* Step 4: Navigate the folder where file is saved using cd command.
* Step 5: Compile and run the program



4. Display your name and complete address in different lines.

## Code:

public class Name {

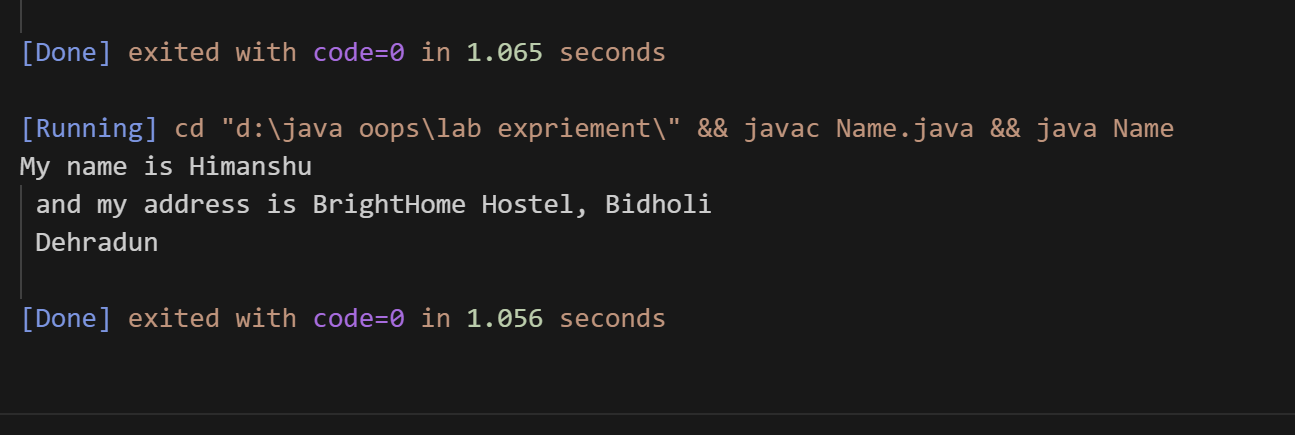
    public static void main(String[] args) {

        System.out.println("My name is Himanshu\nand my address is BrightHome Hostel, Bidholi \n Dehradun");

    }

}

## Output:



Additional Question: 5. Design a visually appealing gradesheet that displays the Name, Roll Number, SAP ID, and Result. Use escape sequences and special characters like '\*' to enhance its presentation. [No need to take any input from User].

## Code:

public class GradeSheet {

    public static void main(String[] args) {

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

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

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

        System.out.println(" Name       : Himanshu         ");

        System.out.println(" Roll No    : R2142230275      ");

        System.out.println(" SAP ID     : 500123230        ");

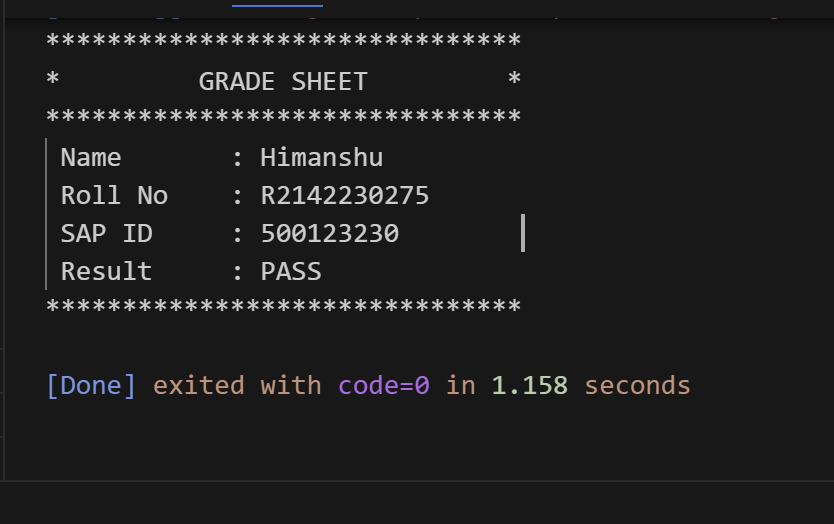
        System.out.println(" Result     : PASS             ");

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

    }

}

## Output:



# Experiment 2: - Basic Java Programming

1. Write a program to find area of triangle.

## Code:

import java.util.Scanner;

public class Area\_of\_Triangle {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the base of triangle:");

        int base = scanner.nextInt();

        System.out.println("Enter the height of triangle");

        int height = scanner.nextInt();

        double area = 0.5\*base\*height;

        System.out.println("Area of triange:" + area);

        scanner.close();

}

}

## Output:

## 

1. Write a program to find simple interest.

## Code:

import java.util.Scanner;

public class Simple\_interest {

    public static void main(String[] args) {

        Scanner scanner = new Scanner (System.in);

        System.out.println("Enter the principle amount:");

        double principle = scanner.nextDouble();

        System.out.println("Enter the time period:");

        double timeperiod = scanner.nextDouble();

        System.out.println("Enter the rate of interest:");

        double rate = scanner.nextDouble();

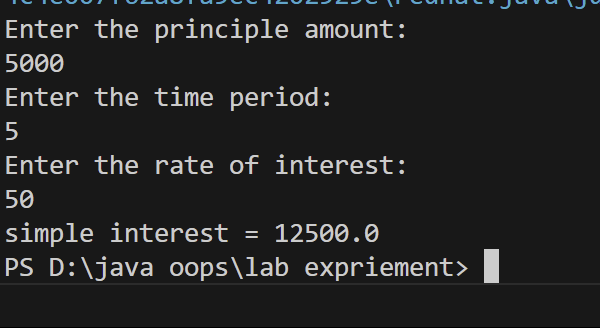
        double Interest = (principle\*rate\*timeperiod)/100;

        System.out.println("simple interest = " + Interest);

    }

    }

## Output:



1. Write a program to implement a command line calculator. (Try for Add sub Mul in same program for 2 digits.) (Hint: Integer.parseInt will be used) For e.g. java calc 20 + 30 Output should be Sum of 20 and 30 is 50 java calc 50 \* 30 Output should be Product of 50 and 30 is 1500.

## Code:

import java.util.Scanner;

public class Calculator {

    public static void main(String[] args) {

        Scanner scanner = new Scanner (System.in);

        System.out.println("Enter the first digit:");

        int a = scanner.nextInt();

        System.out.println("Enter the second digit:");

        int b = scanner.nextInt();

        System.out.println("sum of digit: ");

        int c = (a+b);

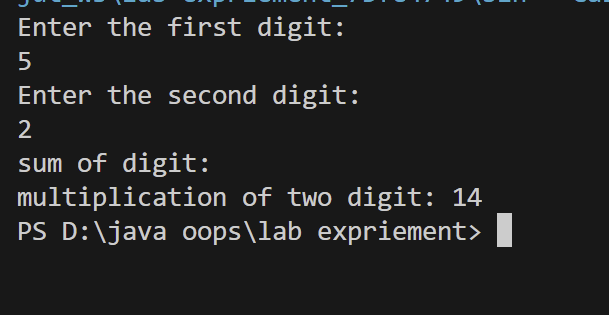
        int Product = c\*b;

        System.out.println("multiplication of two digit: " + Product);

    }

}

## Output:



1. Write a Java program to check whether a given number is positive, negative, or zero using an if-else statement.

## Code:

import java.util.Scanner;

public class Check\_number {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("enter the number:");

        int number = scanner.nextInt();

if (number > 0){

            System.out.println("Number is positive");

        }

        else if (number < 0){

            System.out.println("Number is negative ");

        }

        else {

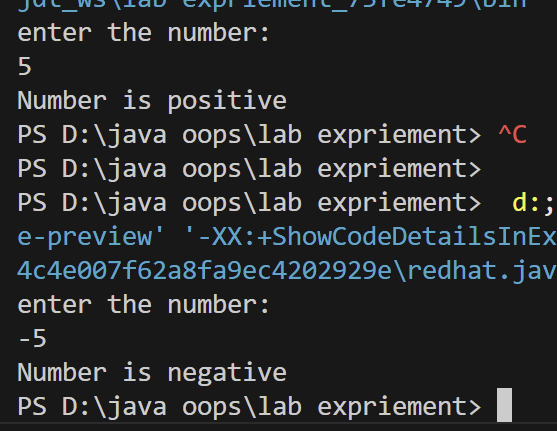
            System.out.println("Number is = 0");

        }

}

    }

## Output:



1. Create a program that accepts three integers and determines the greatest among them using nested if-else statements.

## Code:

import java.util.Scanner;

public class Find\_greatest\_no {

    public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter the first digit:");

    int a = scanner.nextInt();

    System.out.println("Enter the second digit:");

    int b = scanner.nextInt();

    System.out.println("Enter the third digit:");

    int c = scanner.nextInt();  
 int greatest;

    if (a >= b){

        if ( a >= c ){

            greatest = a;

        }

        else {

            greatest = c;

        }

    }

    else {

        if (b >= c) {

            greatest = b;

        }else {

            greatest = c;

        }

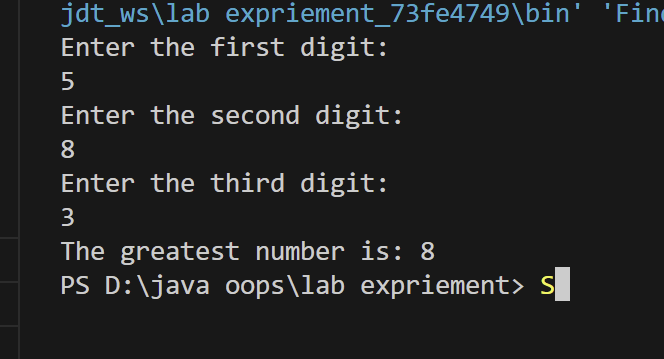
    }

    System.out.println("The greatest number is: " + greatest);

     scanner.close()  
}

}

## Output:



1. Create a program that accepts a number (1–7) and displays the corresponding day of the week using a switch statement.

## Code:

import java.util.Scanner;

public class Days {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the number between (1-7):");

        int day = scanner.nextInt();

        switch (day){

            case 1:

            System.out.println("Monday");

            break;

            case 2:

            System.out.println("Tuesday");

            break;

            case 3:

            System.out.println("Wednesday");

            break;

case 4:

            System.out.println("Thursday");

            break;

            case 5:

            System.out.println("Friday");

            break;

            case 6:

            System.out.println("Saturday");

            break;

            case 7:

            System.out.println("Sunday");

            break;

            default:

            System.out.println("Invalid input! Please enter a number between 1 and 7.");

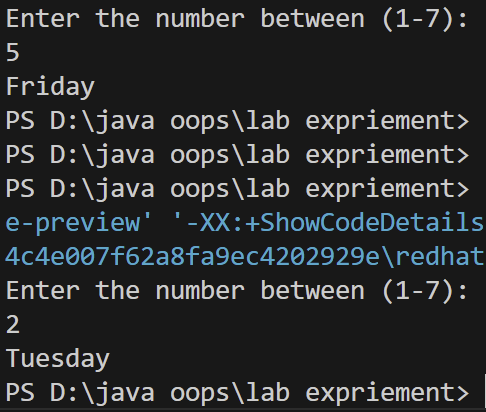
        }

        scanner.close();

        }

    }

## Output:



1. Write a program to calculate the final grade of a student based on the marks entered in three subjects. Use the following grading scale:

Average >= 90:

Grade A Average >= 75:

Grade B Average >= 50:

Grade C Otherwise: Grade F

## Code:

import java.util.Scanner;

public class FinalGradeCalculator {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        double total = 0;

        for (int i = 1; i <= 3; i++) {

            System.out.print("Enter the marks for subject " + i + ": "); // Prompting the user for input

            total += scanner.nextDouble(); // Collecting user input

        }

        double average = total / 3;

        String grade = (average >= 90) ? "Grade A" :

                       (average >= 75) ? "Grade B" :

                       (average >= 50) ? "Grade C" : "Grade F";

        System.out.printf("Average Marks: %.2f%n", average);

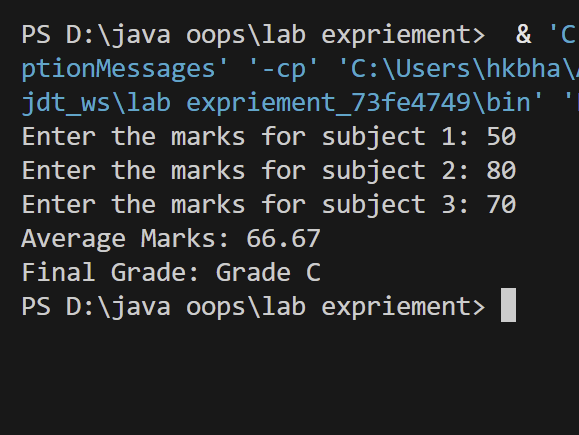
        System.out.println("Final Grade: " + grade);

        scanner.close();

    }

}

## Output:



# EXPERIMENT – 3

# TITLE: Basic Java Programming (Loops & Arrays)

1. Write a program to calculate the sum of all integers between 10 and 950 that are divisible by both 6 and 9.

## Code:

import java.util.Scanner;

public class SumDivisbleBy6And9{

    public static void main(String[] args) {

        int start = 10;

        int end = 950;

        int sum = 0;

        for (int i = start; i <= end; i++) {

            if (i % 6 == 0 && i % 9 == 0 ) {

                sum += i;

            }

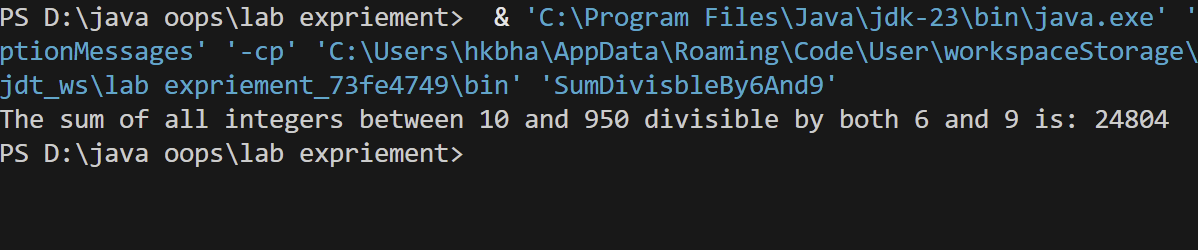
        }

        System.out.println("The sum of all integers between 10 and 950 divisible by both 6 and 9 is: " + sum);

    }

}

## Output:



1. Write a Java program that takes an integer as input and calculates the sum of its digits using a while loop.

## Code

import java.util.Scanner;

public class Sum\_of\_digit {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("enter the interger");

        int number = scanner.nextInt();

        int sum = 0;

        while (number != 0){

        int digit = number % 10;

        sum += digit;

        number /= 10;

        }

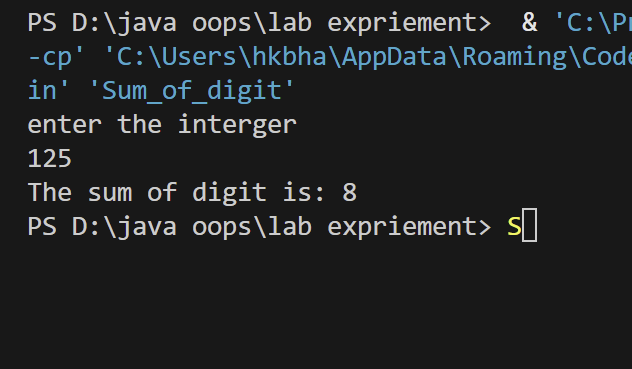
        System.out.println("The sum of digit is: " + sum);

        scanner.close();

    }

   }

## Output:



1. Write a Java program that prints the first N terms of the Fibonacci series using a loop.

## Code:

import java.util.Scanner;

public class Fibonacci\_series {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of terms");

        int Number = scanner.nextInt();

        int First\_Term = 0, Second\_Term = 1;

        if (Number <= 0 ){

            System.out.println("There is no series please enter positive integer");

        }

        else {

            System.out.println("The first " + Number + "terms of the Fibonacci series are:");

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

                System.out.println(First\_Term + "");

int nextTerm = First\_Term + Second\_Term;

                First\_Term = Second\_Term;

                Second\_Term = nextTerm;

}

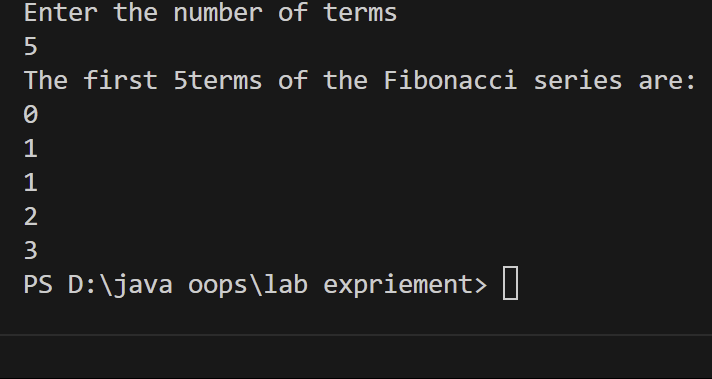
        }

        scanner.close();

    }

    }

## Output:



1. Write a Java program to count and display the total number of prime numbers between 1 and 1000.

## Code:

// PrimeNumbers.java

// Class definition

public class PrimeNumbers {

// Main method

public static void main(String[] args) {

int count = 0; // Variable to store count of primes

// Display prime numbers

System.out.println("Prime numbers between 1 and 1000:");

// Loop through numbers

for (int num = 2; num <= 1000; num++) {

if (isPrime(num)) {

System.out.print(num + " "); // Output prime number

count++; // Increment count

}

}

// Print total count

System.out.println("\nTotal number of prime numbers: " + count);

}

// Method to check prime status

public static boolean isPrime(int num) {

for (int i = 2; i <= Math.sqrt(num); i++) {

if (num % i == 0) {

return false; // Not prime

}

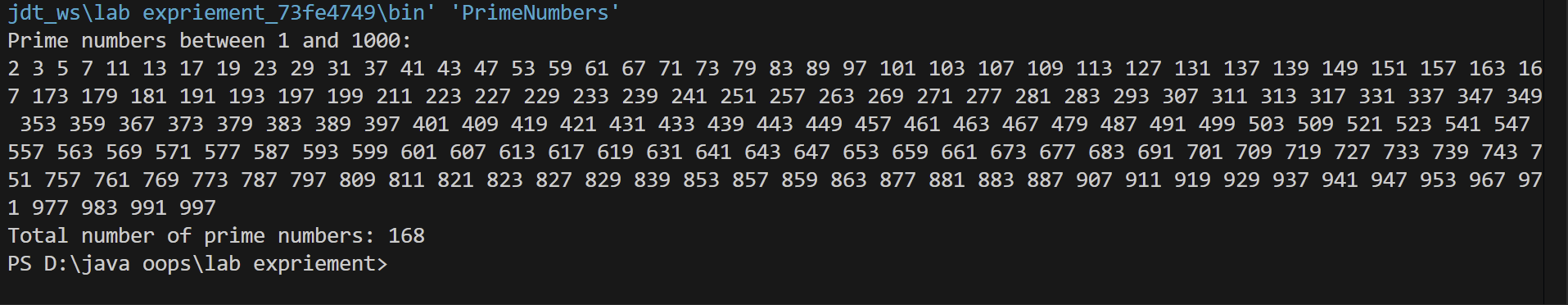
}

return true; // Prime

}

}

## Output:



1. Write a Java program that counts how many times a given number appears in an array. Input: arr = [2, 3, 2, 5, 2, 6], target = 2 Output: 3

Code:

import java.util.Arrays;

public class NumberFrequency {

    public static void main(String[] args) {

        int[] arr = {2, 3, 2, 5, 2, 6};

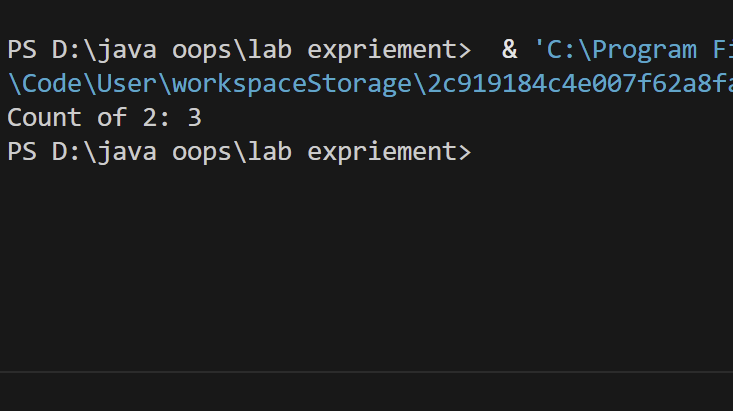
        int target = 2;

        long count = Arrays.stream(arr).filter(num -> num == target).count();

        System.out.println("Count of " + target + ": " + count);

    }

}

Output: 

1. Write a Java program to find the second largest element in an integer array without sorting the array.

Code:

public class SecondLargest {

    public static int findSecondLargest(int[] arr) {

        if (arr.length < 2) {

            throw new IllegalArgumentException("Array must contain at least two elements.");

        }

        int largest = Integer.MIN\_VALUE;

        int secondLargest = Integer.MIN\_VALUE;

        for (int num : arr) {

            if (num > largest) {

                secondLargest = largest;

                largest = num;

            } else if (num > secondLargest && num != largest) {

                secondLargest = num;

            }

        }  return secondLargest;

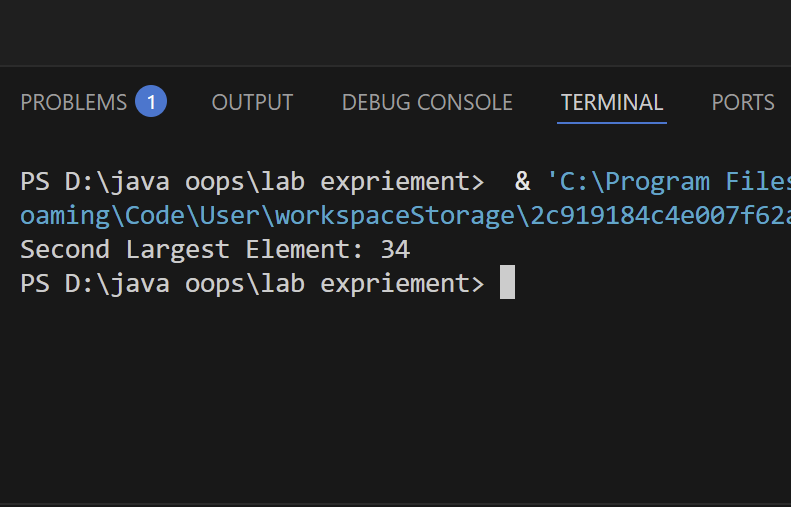
    }

public static void main(String[] args) {

        int[] arr = {12, 35, 1, 10, 34, 1};

        System.out.println("Second Largest Element: " + findSecondLargest(arr));

    }

}

1. WAP to print the following pattern using loop ? # # # ? ? ? ? ? # # # # # # # ? ? ? ? ? ? ? ? ?

Code:

public class Pattern {

    public static void main(String[] args) {

        int rows = 5; // Number of rows

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

            char symbol = (i % 2 != 0) ? '?' : '#'; // Alternating between '?' and '#'

            for (int j = 0; j < (2 \* i - 1); j++) {

                System.out.print(symbol + " ");

            }

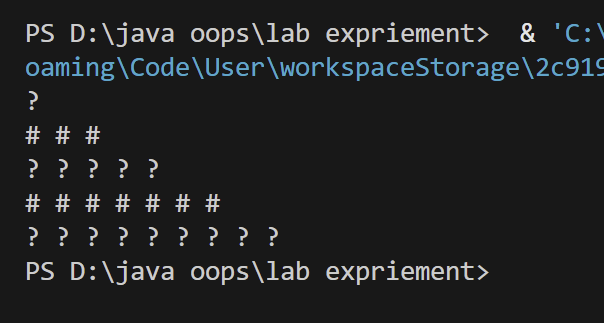
            System.out.println();

        }

    }

}

Output:



1. Write a Java program that copies all elements from one array to another using a loop.

Code: public class ArrayCopy {

    public static void main(String[] args) {

        // Original array

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

        // Create a new array of the same length

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

        // Copy elements using a loop

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

            targetArray[i] = sourceArray[i];

        }

        // Print the copied array

        System.out.println("Copied array:");

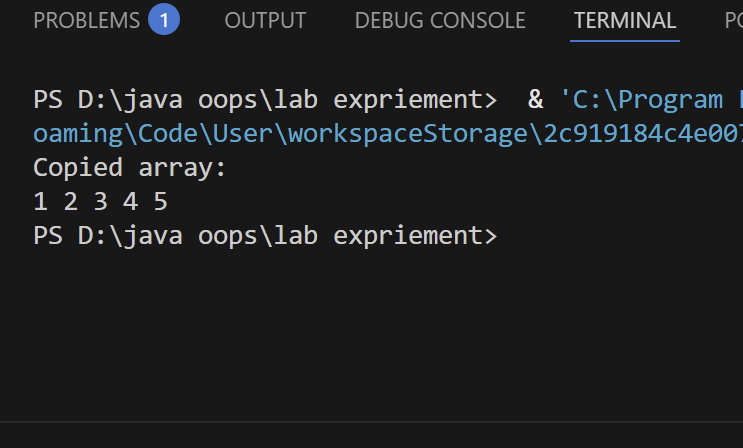
        for (int num : targetArray) {

            System.out.print(num + " ");

        }

    }

}

Output: 

1. Given an array containing N-1 unique numbers from 1 to N, write a Java program to find the missing number. Input: [1, 5, 6, 2, 4] Output: 3

Code: public class MissingNumber {

    public static void main(String[] args) {

        int[] arr = {1, 5, 6, 2, 4};

        int n = arr.length + 1; // The total numbers should be from 1 to N

        // Calculate the expected sum of numbers from 1 to N

        int expectedSum = n \* (n + 1) / 2;

        // Calculate the actual sum of the elements in the array

        int actualSum = 0;

        for (int num : arr) {

            actualSum += num;

        }

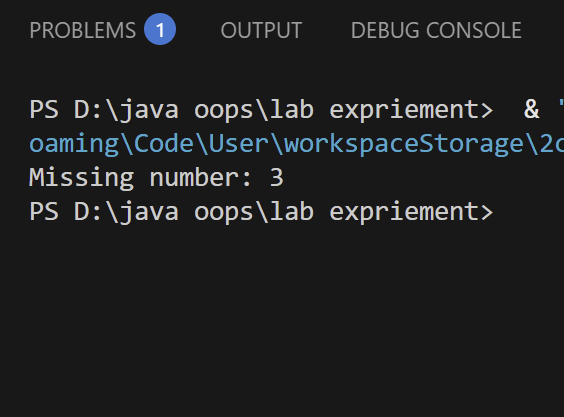
        // The missing number is the difference

        int missingNumber = expectedSum - actualSum;

        System.out.println("Missing number: " + missingNumber);

    }

}

Output: 

1. Write a Java program to rotate an array right by K positions. Input: arr = [1, 2, 3, 4, 5], K = 2 Output: [4, 5, 1, 2, 3]

Code: import java.util.Arrays;

public class ArrayRotation {

    public static void main(String[] args) {

        int[] arr = {1, 2, 3, 4, 5};

        int k = 2;

        // Ensure k does not exceed array length

        k = k % arr.length;

        // Rotate the array using reversal method

        reverse(arr, 0, arr.length - 1);   // Reverse entire array

        reverse(arr, 0, k - 1);            // Reverse first K elements

        reverse(arr, k, arr.length - 1);   // Reverse remaining elements

        System.out.println("Rotated Array: " + Arrays.toString(arr));

    }

    // Method to reverse a portion of the array

    private static void reverse(int[] arr, int start, int end) {

        while (start < end) {

            int temp = arr[start];

            arr[start] = arr[end];

            arr[end] = temp;

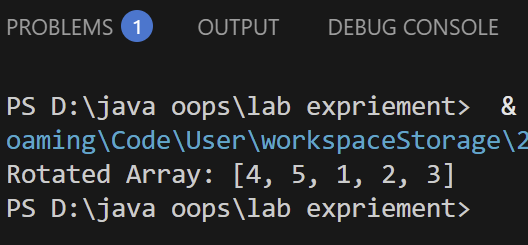
            start++;

            end--;

        }

    }

}

Output: 

# EXPERIMENT – 4

# TITLE: Classes (Constructors, Access modifiers, Method Overloading, static & non static data members, this)

1. Create a Student class with attributes for name and age. Implement a default constructor to assign default values and a parameterized constructor to initialize the attributes with user defined values. Create objects using both constructors and display their details.

Code: class Student {

    private String name;

    private int age;

    public Student() {

        this.name = "Default Name";

        this.age = 18;

    }

    public Student(String name, int age) {

        this.name = name;

        this.age = age;

    }

    public void display() {

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

    }

    public static void main(String[] args) {

        Student student1 = new Student();

        Student student2 = new Student("Himanshu", 20);

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

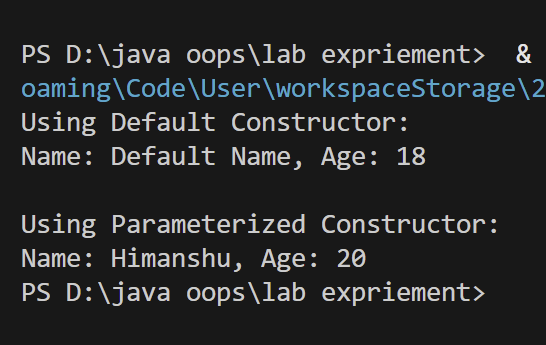
        student1.display();

        System.out.println("\nUsing Parameterized Constructor:");

        student2.display();

    }

}

Output: 

1. Create a BankAccount class with a private variable balance to store the account balance. Implement a public method deposit(double amount) to add funds, a protected method withdraw(double amount) to deduct funds, and a default-access method checkBalance() to display the current balance. Create an object of the class and demonstrate which methods and variables can be accessed both inside and outside the class.

Code: class BankAccount {

    private double balance;

    public BankAccount(double initialBalance) {

        this.balance = initialBalance;

    }

    public void deposit(double amount) {

        if (amount > 0) {

            balance += amount;

            System.out.println("Deposited: $" + amount);

        } else {

            System.out.println("Invalid deposit amount.");

        }

    }

    protected void withdraw(double amount) {

        if (amount > 0 && amount <= balance) {

            balance -= amount;

            System.out.println("Withdrawn: $" + amount);

        } else {

            System.out.println("Insufficient balance or invalid withdrawal amount.");

        }

    }

    void checkBalance() {

        System.out.println("Current balance: $" + balance);

    }

}

public class BankDemo {

    public static void main(String[] args) {

        BankAccount account = new BankAccount(1000);

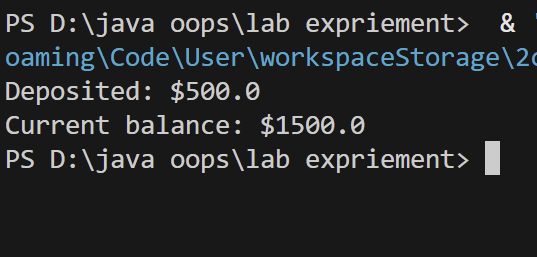
        account.deposit(500);

        account.checkBalance();

    }

}

Output:



1. Create a Calculator class that contains a method add() to perform addition. Overload the add() method to handle different types and numbers of parameters, such as adding two integers, two doubles, and three integers. Create an object of the class and demonstrate all method variations.

Code: class Calculator {

    public int add(int a, int b) {

        return a + b;

    }

    public double add(double a, double b) {

        return a + b;

    }

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

        return a + b + c;

    }

    public static void main(String[] args) {

        Calculator calc = new Calculator();

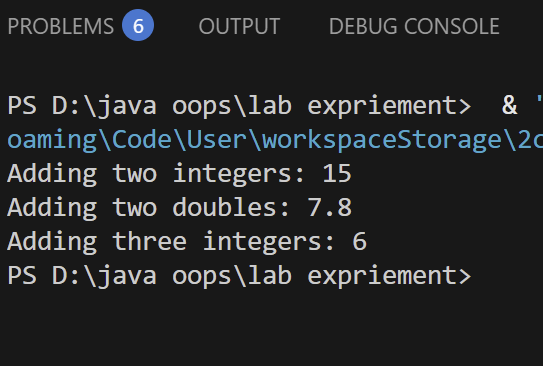
        System.out.println("Adding two integers: " + calc.add(5, 10));

        System.out.println("Adding two doubles: " + calc.add(5.5, 2.3));

        System.out.println("Adding three integers: " + calc.add(1, 2, 3));

    }

}

Output: 

1. Create a Student class that has a static variable universityName and a non-static variable studentName. Include a static method to display the university name. Then, create multiple student objects to demonstrate how the static variable is shared among all instances, while the non-static variable holds unique values for each object.

Code:

class Student1 {

    static String universityName = "Upes";

    String studentName;

    public Student1(String studentName) {

        this.studentName = studentName;

    }

    static void displayUniversityName() {

        System.out.println("University: " + universityName);

    }

    void displayStudent() {

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

    }

    public static void main(String[] args) {

        Student1.displayUniversityName();

        Student1 student1 = new Student1("Himanshu");

        Student1 student2 = new Student1("Hardik");

        Student1 student3 = new Student1("Ram");

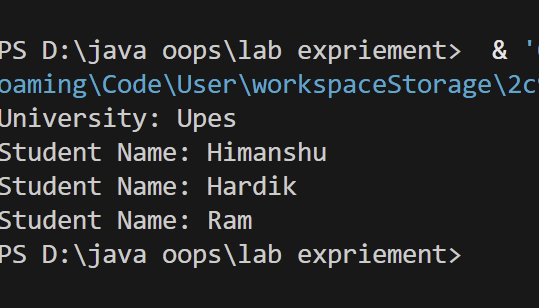
        student1.displayStudent();

        student2.displayStudent();

        student3.displayStudent();

    }

}

Output: 

1. A student is developing a course registration system that allows students to enroll in courses. Each course has a course name and a course code. Implement a Course class with appropriate attributes and use the “this” keyword to differentiate between class attributes and constructor parameters during initialization. Create an object of the Course class and display the course details. [Keep the variable name of formal arguments in constructor and instance variables same.]

Code: class Course {

    private String courseName;

    private String courseCode;

    public Course(String courseName, String courseCode) {

        this.courseName = courseName;

        this.courseCode = courseCode;

    }

    public void displayCourseDetails() {

        System.out.println("Course Name: " + courseName);

        System.out.println("Course Code: " + courseCode);

    }

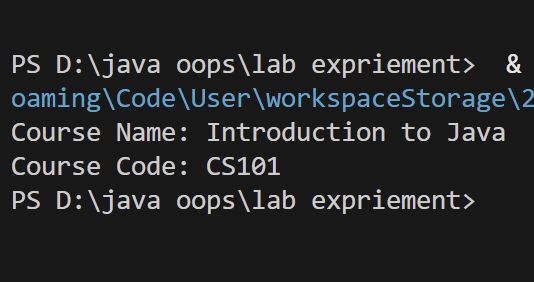
    public static void main(String[] args) {

        Course course = new Course("Introduction to Java", "CS101");

        course.displayCourseDetails();

    }

}

Output: 

1. Develop an Employee Management System in Java to track employee details such as ID, name, department, and salary, while calculating total salary expenditure and employee count. Create an Employee class with instance variables (employeeID, name, department, and private salary), a static variable totalEmployees to track the employee count, a default constructor for initializing default values, and a parameterized constructor for setting user provided details. Include a static method to display totalEmployees, a calculateSalary() method to return the salary, and a displayEmployeeInfo() method to show employee details. Use the this keyword in constructors to differentiate class variables from parameters. In the main method, create Employee objects using both constructors, display the total number of employees, and show salary details for each employee.

Code:

class Employee {

    // Instance variables

    private int employeeID;

    private String name;

    private String department;

    private double salary;

    // Static variable to track total employees

    private static int totalEmployees = 0;

    // Default constructor

    public Employee() {

        this(0, "Unknown", "None", 0.0);

    }

    // Parameterized constructor

    public Employee(int employeeID, String name, String department, double salary) {

        this.employeeID = employeeID;

        this.name = name;

        this.department = department;

        this.salary = salary;

        totalEmployees++;  // Increment total employees

    }

    // Static method to display total employees

    public static void displayTotalEmployees() {

        System.out.println("Total Employees: " + totalEmployees);

    }

    // Method to return salary

    public double calculateSalary() {

        return this.salary;

    }

    // Method to display employee details

    public void displayEmployeeInfo() {

        System.out.println("Employee ID: " + this.employeeID);

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

        System.out.println("Department: " + this.department);

        System.out.println("Salary: $" + this.salary);

        System.out.println("-----------------------------");

    }

}

public class EmployeeManagementSystem {

    public static void main(String[] args) {

        // Creating Employee objects using different constructors

        Employee emp1 = new Employee(101, "Himanshu", "Finance", 55000);

        Employee emp2 = new Employee(102, "Ram", "HR", 48000);

        Employee emp3 = new Employee(); // Using default constructor

        // Display Employee details

        emp1.displayEmployeeInfo();

        emp2.displayEmployeeInfo();

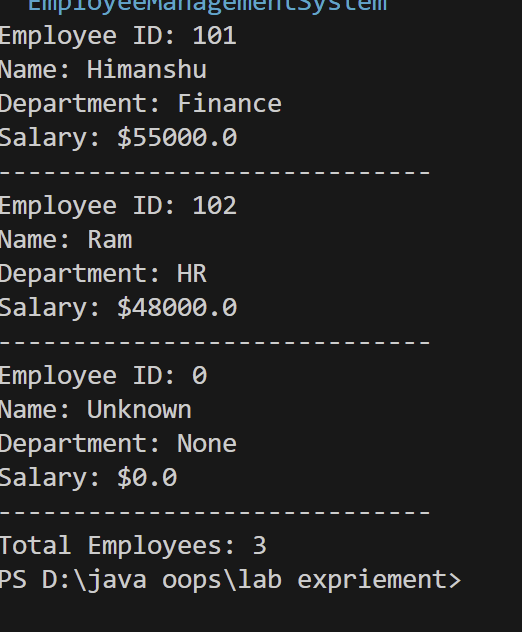
        emp3.displayEmployeeInfo();

        // Display total employees count

        Employee.displayTotalEmployees();

    }

}

Output: 

# EXPERIMENT – 5 (TITLE Inheritance)

1. Write a Java program to demonstrate that a private member of a superclass cannot be accessed directly from a derived class.

Code:

class SuperClass {

    private int privateVar = 42;

    int getPrivateVar() {

        return privateVar;

    }

}

class SubClass extends SuperClass {

    void display() {

        // System.out.println(privateVar); // This would cause a compilation error

        System.out.println(getPrivateVar()); // Access through a public method

    }

}

public class Main {

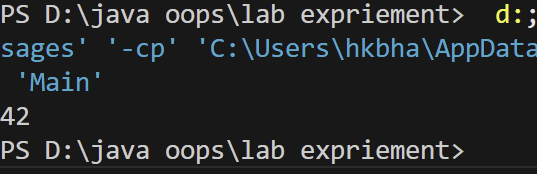
    public static void main(String[] args) {

        SubClass obj = new SubClass();

        obj.display();

    }

}

Output: 

1. Create a Java program with a Player class and derive three subclasses: Cricket\_Player, Football\_Player, and Hockey\_Player. Implement attributes such as name, age, and position, and methods like play() and train() to represent these players.

Code: class Player {

    String name;

    int age;

    String position;

    Player(String name, int age, String position) {

        this.name = name;

        this.age = age;

        this.position = position;

    }

    void play() {

        System.out.println(name + " is playing as " + position);

    }

    void train() {

        System.out.println(name + " is training hard.");

    }

}

class Cricket\_Player extends Player {

    Cricket\_Player(String name, int age, String position) {

        super(name, age, position);

    }

}

class Football\_Player extends Player {

    Football\_Player(String name, int age, String position) {

        super(name, age, position);

    }

}

class Hockey\_Player extends Player {

    Hockey\_Player(String name, int age, String position) {

        super(name, age, position);

    }

}

public class PlayerDemo {

    public static void main(String[] args) {

        Cricket\_Player cp = new Cricket\_Player("Virat Kohli", 35, "Batsman");

        Football\_Player fp = new Football\_Player("Cristiano Ronaldo", 39, "Forward");

        Hockey\_Player hp = new Hockey\_Player("Manpreet Singh", 32, "Midfielder");

        cp.play();

        cp.train();

        fp.play();

        fp.train();

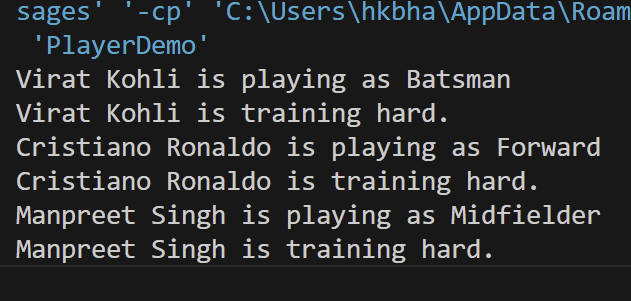
        hp.play();

        hp.train();

    }

}

Output:



1. Define a Worker class with DailyWorker and SalariedWorker as its subclasses. Each worker has a name and salary rate. Implement a method computePay(int hours) to compute weekly pay. DailyWorker is paid based on the number of days worked (assuming 8 hours per day), whereas SalariedWorker receives a fixed wage for 40 hours per week, regardless of actual hours worked. Use polymorphism to implement this program and test worker salary calculations.

Code: class Worker {

    protected String name;

    protected double salaryRate;

    public Worker(String name, double salaryRate) {

        this.name = name;

        this.salaryRate = salaryRate;

    }

    public double computePay(int hours) {

        return 0;

    }

}

class DailyWorker extends Worker {

    public DailyWorker(String name, double salaryRate) {

        super(name, salaryRate);

    }

    @Override

    public double computePay(int hours) {

        int daysWorked = hours / 8;

        return daysWorked \* salaryRate;

    }

}

class SalariedWorker extends Worker {

    public SalariedWorker(String name, double salaryRate) {

        super(name, salaryRate);

    }

    @Override

    public double computePay(int hours) {

        return salaryRate \* 40;

    }

}

public class WorkerTest {

    public static void main(String[] args) {

        Worker dailyWorker = new DailyWorker("Alice", 500);

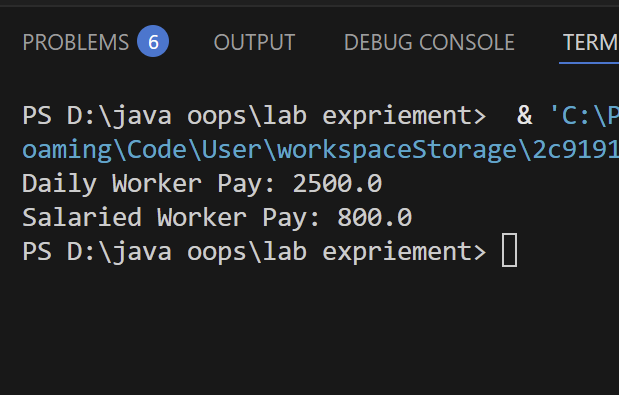
        Worker salariedWorker = new SalariedWorker("Bob", 20);

        System.out.println("Daily Worker Pay: " + dailyWorker.computePay(40));

        System.out.println("Salaried Worker Pay: " + salariedWorker.computePay(50));

    }

}

Output: 

1. Implement a Java program to calculate trunk call charges based on duration and type (Ordinary, Urgent, or Lightning). Use polymorphism to manage different charge rates for each type. Implement a Java program to calculate trunk call charges based on duration (in minutes) and type (Ordinary, Urgent, or Lightning). Use polymorphism to manage different charge rates for each type. The program should take user input for duration and type and display the total charge.

Code: import java.util.Scanner;

abstract class TrunkCall {

    protected int duration;

    public TrunkCall(int duration) {

        this.duration = duration;

    }

    abstract double calculateCharge();

}

class OrdinaryCall extends TrunkCall {

    private static final double RATE = 2.0;

    public OrdinaryCall(int duration) {

        super(duration);

    }

    @Override

    double calculateCharge() {

        return duration \* RATE;

    }

}

class UrgentCall extends TrunkCall {

    private static final double RATE = 3.5;

    public UrgentCall(int duration) {

        super(duration);

    }

    @Override

    double calculateCharge() {

        return duration \* RATE;

    }

}

class LightningCall extends TrunkCall {

    private static final double RATE = 5.0;

    public LightningCall(int duration) {

        super(duration);

    }

    @Override

    double calculateCharge() {

        return duration \* RATE;

    }

}

public class TrunkCallCalculator {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter duration (minutes): ");

        int duration = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter type of call (Ordinary/Urgent/Lightning): ");

        String type = scanner.nextLine().trim().toLowerCase();

        TrunkCall call = null;

        switch (type) {

            case "ordinary":

                call = new OrdinaryCall(duration);

                break;

            case "urgent":

                call = new UrgentCall(duration);

                break;

            case "lightning":

                call = new LightningCall(duration);

                break;

            default:

                System.out.println("Invalid call type!");

                scanner.close();

                return;

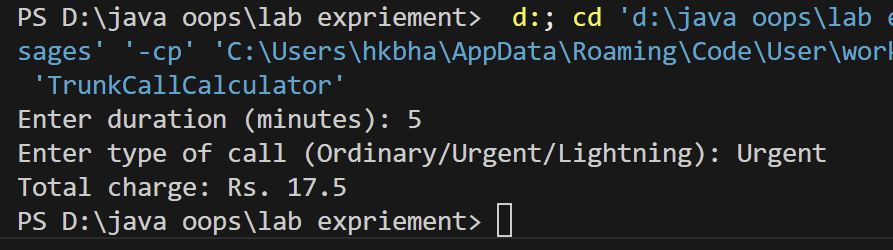
        }

        System.out.println("Total charge: Rs. " + call.calculateCharge());

        scanner.close();

    }

}

Output: 

1. Design a Java class Employee with attributes name, empid, and salary. Implement a default constructor, a parameterized constructor, and methods to return the employee’s name and salary. Add a method increaseSalary(double percentage) to raise the salary by a user specified percentage. Create a subclass Manager with an additional instance variable department. Develop a test program to validate these functionalities.

Code: class Employee1 {

    private String name;

    private int empid;

    private double salary;

    public Employee1() {

        this.name = "Unknown";

        this.empid = 0;

        this.salary = 0.0;

    }

    public Employee1(String name, int empid, double salary) {

        this.name = name;

        this.empid = empid;

        this.salary = salary;

    }

    public String getName() {

        return name;

    }

    public double getSalary() {

        return salary;

    }

    public void increaseSalary(double percentage) {

        if (percentage > 0) {

            this.salary += (this.salary \* percentage / 100);

        }

    }

    public void displayDetails() {

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

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

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

    }

}

class Manager extends Employee1 {

    private String department;

    public Manager(String name, int empid, double salary, String department) {

        super(name, empid, salary);

        this.department = department;

    }

    public void displayDetails() {

        super.displayDetails();

        System.out.println("Department: " + department);

    }

}

public class EmployeeTest {

    public static void main(String[] args) {

        Employee1 emp1 = new Employee1("Himanshu", 101, 50000);

        emp1.displayDetails();

        System.out.println("Increasing salary by 10%...");

        emp1.increaseSalary(10);

        emp1.displayDetails();

        System.out.println("\n----------------------\n");

        Manager mgr1 = new Manager("Sumit", 201, 70000, "HR");

        mgr1.displayDetails();

        System.out.println("Increasing salary by 15%...");

        mgr1.increaseSalary(15);

        mgr1.displayDetails();

    }

}

Output:

