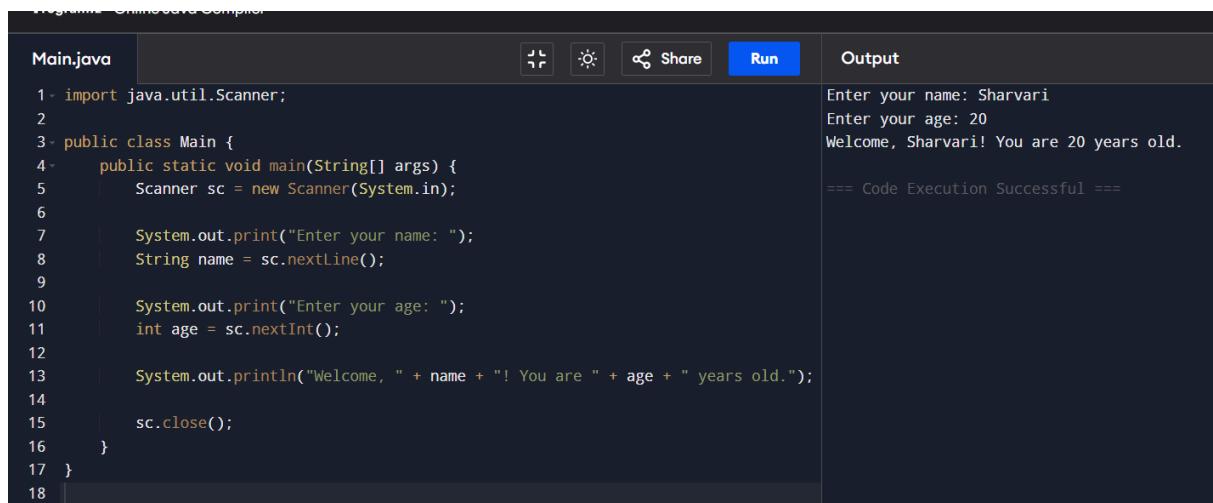


Name of the Student: Sharvari Muley

Date of Submission: 4-11-2025

Number of Programs Completed (out of 50): 34

1. Develop a Java program to take user input for name and age and display a welcome message.



Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         System.out.print("Enter your name: ");
8         String name = sc.nextLine();
9
10        System.out.print("Enter your age: ");
11        int age = sc.nextInt();
12
13        System.out.println("Welcome, " + name + "! You are " + age + " years old.");
14
15        sc.close();
16    }
17 }
18
```

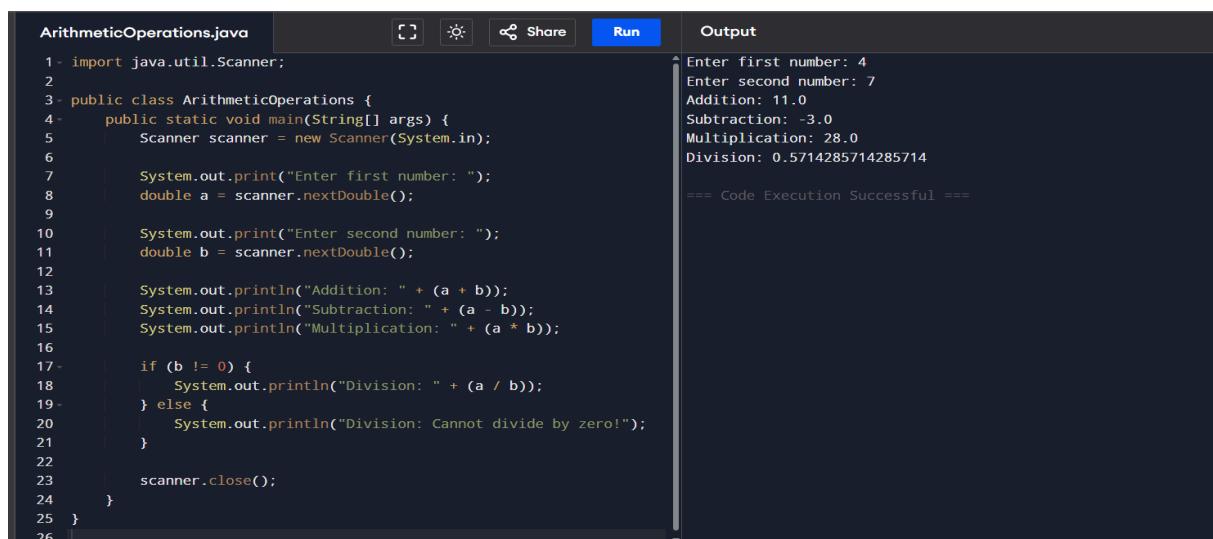
Run

Output

```
Enter your name: Sharvari
Enter your age: 20
Welcome, Sharvari! You are 20 years old.

== Code Execution Successful ==
```

2. Write a Java program that takes two numbers and performs basic arithmetic operations (+, -, *, /).



ArithmeticOperations.java

```
1 import java.util.Scanner;
2
3 public class ArithmeticOperations {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6
7         System.out.print("Enter first number: ");
8         double a = scanner.nextDouble();
9
10        System.out.print("Enter second number: ");
11        double b = scanner.nextDouble();
12
13        System.out.println("Addition: " + (a + b));
14        System.out.println("Subtraction: " + (a - b));
15        System.out.println("Multiplication: " + (a * b));
16
17        if (b != 0) {
18            System.out.println("Division: " + (a / b));
19        } else {
20            System.out.println("Division: Cannot divide by zero!");
21        }
22
23        scanner.close();
24    }
25 }
26
```

Run

Output

```
Enter first number: 4
Enter second number: 7
Addition: 11.0
Subtraction: -3.0
Multiplication: 28.0
Division: 0.5714285714285714

== Code Execution Successful ==
```

3. Create a program to convert temperature from Fahrenheit to Celsius.

```

TempConverter.java
1 import java.util.Scanner;
2
3 public class TempConverter {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in); // Scanner for input
6
7         System.out.print("Enter temperature in Fahrenheit: ");
8         double fahrenheit = scanner.nextDouble(); // Read Fahrenheit
9
10        // Convert Fahrenheit to Celsius
11        double celsius = (fahrenheit - 32) * 5 / 9;
12
13        // Display result
14        System.out.println("Temperature in Celsius: " + celsius);
15    }
16 }
17

```

Output

```

Enter temperature in Fahrenheit: 34
Temperature in Celsius: 1.11111111111112

== Code Execution Successful ==

```

4.Design a Java application to calculate simple interest using the formula: $SI = (P \times R \times T) / 100$.

```

SimpleInterest.java
1 import java.util.Scanner;
2
3 public class SimpleInterest {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in); // Scanner for input
6
7         // Get Principal, Rate, Time
8         System.out.print("Enter Principal (P): ");
9         double P = scanner.nextDouble();
10
11        System.out.print("Enter Rate (R): ");
12        double R = scanner.nextDouble();
13
14        System.out.print("Enter Time (T): ");
15        double T = scanner.nextDouble();
16
17        // Calculate Simple Interest
18        double SI = (P * R * T) / 100;
19
20        // Display result
21        System.out.println("Simple Interest: " + SI);
22    }
23 }
24

```

Output

```

Enter Principal (P): 300
Enter Rate (R): 34444
Enter Time (T): 3
Simple Interest: 309996.0

== Code Execution Successful ==

```

5.Write a Java program to determine whether a given year is a leap year.

LeapYearCheck.java	Run	Output
<pre>1- import java.util.Scanner; 2 3- public class LeapYearCheck { 4- public static void main(String[] args) { 5- Scanner scanner = new Scanner(System.in); // Input scanner 6 7- System.out.print("Enter a year: "); 8- int year = scanner.nextInt(); // Read year 9 10 // Check leap year conditions 11 if ((year % 4 == 0 && year % 100 != 0) (year % 400 == 0)) { 12 System.out.println(year + " is a leap year."); 13 } else { 14 System.out.println(year + " is not a leap year."); 15 } 16 } 17 } 18</pre>	Enter a year: 2005 2005 is not a leap year. == Code Execution Successful ==	

6.Develop a program to check whether an input number is prime or not using for loop..

PrimeCheck.java	Run	Output
<pre>1- import java.util.Scanner; 2 3- public class PrimeCheck { 4- public static void main(String[] args) { 5- Scanner scanner = new Scanner(System.in); // Scanner for input 6 7- System.out.print("Enter a number: "); 8- int num = scanner.nextInt(); // Read number 9 10 boolean isPrime = true; 11 12 if (num <= 1) { 13 isPrime = false; // 0 and 1 are not prime 14 } else { 15 // Check for divisors 16 for (int i = 2; i <= num / 2; i++) { 17 if (num % i == 0) { 18 isPrime = false; 19 break; 20 } 21 } 22 } 23 24 // Output result 25 System.out.println(num + (isPrime ? " is a prime number." : " is not a prime number.")); </pre>	Enter a number: 132 132 is not a prime number. == Code Execution Successful ==	

7.Write a program to reverse a number using a while loop.

ReverseNumber.java		Run	Output
<pre>1- import java.util.Scanner; 2 3- public class ReverseNumber { 4- public static void main(String[] args) { 5- Scanner scanner = new Scanner(System.in); // Scanner for input 6 7- System.out.print("Enter a number: "); 8- int num = scanner.nextInt(); // Read number 9- int reversed = 0; 10 11 // Reverse the number using while loop 12 while (num != 0) { 13 int digit = num % 10; // Get last digit 14 reversed = reversed * 10 + digit; // Append to reversed 15 num /= 10; // Remove last digit 16 } 17 18 // Output reversed number 19 System.out.println("Reversed Number: " + reversed); 20 } 21 }</pre>	Enter a number: 45 Reversed Number: 54 == Code Execution Successful ==		

8.Create a Java application to generate Fibonacci series up to a given number using do-while loop..

FibonacciSeries.java		Run	Output
<pre>1- import java.util.Scanner; 2 3- public class FibonacciSeries { 4- public static void main(String[] args) { 5- Scanner scanner = new Scanner(System.in); // Scanner for input 6 7- System.out.print("Enter the maximum number for Fibonacci series: "); 8- int max = scanner.nextInt(); // Max limit 9 10 int a = 0, b = 1; 11 12 // Generate series using do-while 13 do { 14 System.out.print(a + " "); 15 int next = a + b; 16 a = b; 17 b = next; 18 } while (a <= max); 19 } 20 }</pre>	Enter the maximum number for Fibonacci series: 6 0 1 1 2 3 5 == Code Execution Successful ==		

9.Design a recursive program to compute the factorial of a number using function.

```
FactorialRecursive.java
1- import java.util.Scanner;
2
3- public class FactorialRecursive {
4
5-     public static int factorial(int n) {
6-         if (n <= 1)
7-             return 1;
8-         else
9-             return n * factorial(n - 1);
10    }
11
12-    public static void main(String[] args) {
13-        Scanner scanner = new Scanner(System.in);
14-        System.out.print("Enter a number: ");
15-        int num = scanner.nextInt();
16-        System.out.println("Factorial of " + num + " is " + factorial(num));
17-        scanner.close();
18    }
19 }
```

Output:

```
Enter a number: 24
Factorial of 24 is -775946240
== Code Execution Successful ==
```

10.Implement a program to check whether a given number is an Armstrong number.

```
ArmstrongCheck.java
1- import java.util.Scanner;
2
3- public class ArmstrongCheck {
4-     public static void main(String[] args) {
5-         Scanner scanner = new Scanner(System.in); // Scanner for input
6
7-         System.out.print("Enter a number: ");
8-         int num = scanner.nextInt(); // Read number
9-         int original = num;
10        int sum = 0;
11
12        // Count digits
13        int digits = String.valueOf(num).length();
14
15        // Calculate Armstrong sum
16        while (num != 0) {
17            int digit = num % 10;
18            sum += Math.pow(digit, digits);
19            num /= 10;
20        }
21
22        // Compare and print result
23        if (sum == original) {
24            System.out.println(original + " is an Armstrong number.");
25        } else {
26            System.out.println(original + " is not an Armstrong number.");
27        }
28    }
29 }
```

Output:

```
Enter a number: 544
544 is not an Armstrong number.
== Code Execution Successful ==
```

11. Write a Java program to find the largest and smallest number in an array.

LargestSmallest.java				Run	Output
1- import java.util.Scanner; 2 3- public class LargestSmallest { 4- public static void main(String[] args) { 5- Scanner scanner = new Scanner(System.in); 6- System.out.print("Enter array size: "); 7- int n = scanner.nextInt(); 8- int[] arr = new int[n]; 9- System.out.println("Enter elements:"); 10- for (int i = 0; i < n; i++) arr[i] = scanner.nextInt(); 11- 12- int largest = arr[0], smallest = arr[0]; 13- for (int i = 1; i < n; i++) { 14- if (arr[i] > largest) largest = arr[i]; 15- if (arr[i] < smallest) smallest = arr[i]; 16- } 17- 18- System.out.println("Largest: " + largest); 19- System.out.println("Smallest: " + smallest); 20- scanner.close(); 21- } 22- }	Enter array size: 8 Enter elements: 45 76 98 55 23 18 04 14 Largest: 98 Smallest: 4 ==== Code Execution Successful ===				

12. Develop a program to sort an array using bubble sort algorithm.

BubbleSort.java				Run	Output
1- import java.util.Scanner; 2 3- public class BubbleSort { 4- public static void main(String[] args) { 5- Scanner scanner = new Scanner(System.in); 6- System.out.print("Enter array size: "); 7- int n = scanner.nextInt(); 8- int[] arr = new int[n]; 9- System.out.println("Enter elements:"); 10- for (int i = 0; i < n; i++) arr[i] = scanner.nextInt(); 11- 12- for (int i = 0; i < n - 1; i++) 13- for (int j = 0; j < n - i - 1; j++) { 14- if (arr[j] > arr[j + 1]) { 15- int temp = arr[j]; 16- arr[j] = arr[j + 1]; 17- arr[j + 1] = temp; 18- } 19- 20- System.out.println("Sorted array:"); 21- for (int num : arr) System.out.print(num + " "); 22- scanner.close(); 23- } 24- } 25- }	Enter array size: 5 Enter elements: 23 14 18 08 20 Sorted array: 8 14 18 20 23 ==== Code Execution Successful ===				

13.Implement linear search to find an element in an array.

The screenshot shows a Java code editor with the following code in `LinearSearch.java`:

```
1 import java.util.Scanner;
2
3 public class LinearSearch {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter array size: ");
7         int n = scanner.nextInt();
8         int[] arr = new int[n];
9         System.out.println("Enter elements:");
10        for (int i = 0; i < n; i++) arr[i] = scanner.nextInt();
11        System.out.print("Enter element to search: ");
12        int key = scanner.nextInt();
13
14        boolean found = false;
15        for (int i = 0; i < n; i++) {
16            if (arr[i] == key) {
17                System.out.println("Element found at index " + i);
18                found = true;
19                break;
20            }
21        }
22        if (!found) System.out.println("Element not found");
23        scanner.close();
24    }
25 }
```

The output window shows the execution of the program:

```
Enter array size: 4
Enter elements:
09
56
34
78
Enter element to search: 55
Element not found

== Code Execution Successful ==
```

14.Implement binary search to find an element in an array.

The screenshot shows a Java code editor with the following code in `BinarySearch.java`:

```
1 import java.util.Scanner;
2
3 public class BinarySearch {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.print("Enter array size: ");
7         int n = scanner.nextInt();
8         int[] arr = new int[n];
9         System.out.println("Enter sorted elements:");
10        for (int i = 0; i < n; i++) arr[i] = scanner.nextInt();
11        System.out.print("Enter element to search: ");
12        int key = scanner.nextInt();
13
14        int low = 0, high = n - 1, mid;
15        boolean found = false;
16        while (low <= high) {
17            mid = (low + high) / 2;
18            if (arr[mid] == key) {
19                System.out.println("Element found at index " + mid);
20                found = true;
21                break;
22            } else if (arr[mid] < key)
23                low = mid + 1;
24            else
25                high = mid - 1;
26        }
27    }
28 }
```

The output window shows the execution of the program:

```
Enter array size: 5
Enter sorted elements:
23
43
55
30
9
Enter element to search: 23
Element found at index 0

== Code Execution Successful ==
```

15. Write a Java program to perform matrix addition using for loop.

MatrixAddition.java		Run	Output
<pre>1- import java.util.Scanner; 2 3- public class MatrixAddition { 4- public static void main(String[] args) { 5- Scanner scanner = new Scanner(System.in); 6- System.out.print("Enter rows and columns: "); 7- int rows = scanner.nextInt(), cols = scanner.nextInt(); 8- 9- int[][] a = new int[rows][cols]; 10- int[][] b = new int[rows][cols]; 11- int[][] sum = new int[rows][cols]; 12- 13- System.out.println("Enter first matrix:"); 14- for (int i = 0; i < rows; i++) 15- for (int j = 0; j < cols; j++) 16- a[i][j] = scanner.nextInt(); 17- 18- System.out.println("Enter second matrix:"); 19- for (int i = 0; i < rows; i++) 20- for (int j = 0; j < cols; j++) 21- b[i][j] = scanner.nextInt(); 22- 23- for (int i = 0; i < rows; i++) 24- for (int j = 0; j < cols; j++) 25- sum[i][j] = a[i][j] + b[i][j];</pre>	Enter rows and columns: 2 2 Enter first matrix: 4 5 3 2 Enter second matrix: 6 7 8 4 Sum of matrices: 10 12 11 6 ==== Code Execution Successful ===		

16. Write a java program to find the sum of diagonal elements in an array.

DiagonalSum.java		Run	Output
<pre>1- import java.util.Scanner; 2 3- public class DiagonalSum { 4- public static void main(String[] args) { 5- Scanner sc = new Scanner(System.in); 6- System.out.print("Enter n*n: "); 7- int n = sc.nextInt(); 8- int[][] m = new int[n][n]; 9- System.out.println("Enter matrix:"); 10- for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) m[i][j] = sc.nextInt(); 11- int sum = 0; 12- for (int i = 0; i < n; i++) sum += m[i][i]; 13- System.out.println("Sum of Diagonal: " + sum); 14- sc.close(); 15- } 16- }</pre>	Enter n*n: 4 3 Enter matrix: 1 8 0 8 0 8 0 6 20 10 2 5 1 4 7 9 Sum of Diagonal: 20 ==== Code Execution Successful ===		

17.Check whether a given string is a palindrome.

Programiz Online Java Compiler

PalindromeCheck.java	Run	Output
<pre>1- import java.util.Scanner; 2 3- public class PalindromeCheck { 4- public static void main(String[] args) { 5- Scanner sc = new Scanner(System.in); 6- System.out.print("Enter a number: "); 7- String str = sc.nextLine(); 8- boolean isPalindrome = true; 9- int len = str.length(); 10- for (int i = 0; i < len / 2; i++) { 11- if (str.charAt(i) != str.charAt(len - 1 - i)) { 12- isPalindrome = false; 13- break; 14- } 15- } 16- System.out.println(isPalindrome ? "Palindrome" : "Not a palindrome"); 17- sc.close(); 18- } 19- }</pre>	  	Enter a number: 112 Not a palindrome ==== Code Execution Successful ===

18.Count the number of vowels, consonants, digits, and special characters in a string.

CharacterCounter.java

CharacterCounter.java	Run	Output
<pre>1- import java.util.Scanner; 2 3- public class CharacterCounter { 4- public static void main(String[] args) { 5- Scanner sc = new Scanner(System.in); 6- System.out.print("Enter string: "); 7- String s = sc.nextLine(); 8- int v = 0, c = 0, d = 0, sp = 0; 9- for (char ch : s.toCharArray()) { 10- if (Character.isLetter(ch)) { 11- ch = Character.toLowerCase(ch); 12- if ("aeiou".indexOf(ch) != -1) v++; else c++; 13- } else if (Character.isDigit(ch)) d++; 14- else sp++; 15- } 16- System.out.println("Vowels: " + v + "\nConsonants: " + c + "\nDigits: " + d 17- + "\nSpecial: " + sp); 18- sc.close(); 19- } 20- }</pre>	  	Enter string: Ham@burger*2 Vowels: 3 Consonants: 6 Digits: 1 Special: 2 ==== Code Execution Successful ===

19.Program to reverse the string using predefined methods in String class.

Programiz Online Java Compiler

ReverseString.java	Run	Output
<pre>1- import java.util.Scanner; 2 3- public class ReverseString { 4- public static void main(String[] args) { 5- Scanner sc = new Scanner(System.in); 6- System.out.print("Enter string: "); 7- String s = sc.nextLine(); 8- String rev = ""; 9- for (int i = s.length() - 1; i >= 0; i--) { 10- rev += s.charAt(i); 11- } 12- System.out.println("Reversed: " + rev); 13- sc.close(); 14- } 15- }</pre>	Enter string: 862477537291 Reversed: 192735774268 == Code Execution Successful ==	

20.Write a program to remove duplicate characters from a string.

Main.java

Main.java	Run	Output
<pre>1- import java.util.*; 2 3- public class Main { 4- public static void main(String[] args) { 5- Scanner sc = new Scanner(System.in); 6- 7- System.out.print("Enter a string: "); 8- String input = sc.nextLine(); 9- 10- String result = ""; 11- 12- for (int i = 0; i < input.length(); i++) { 13- char ch = input.charAt(i); 14- 15- // If character is not already added in result 16- if (result.indexOf(ch) == -1) { 17- result += ch; 18- } 19- } 20- 21- System.out.println("String after removing duplicates: " + 22- result); 23- } 24- }</pre>	Enter a string: SUCCEED String after removing duplicates: SUCED == Code Execution Successful ==	Clear

21. Develop a Java program to count the frequency of each word in a sentence.

The screenshot shows a Java code editor with a dark theme. The code in Main.java reads a sentence from standard input, converts it to lowercase, and splits it into words. It then counts the frequency of each word using a HashMap and prints the results. The output window shows the input sentence, the word frequency map, and a success message.

```
Main.java
1 public class Main {
2     public static void main(String[] args) {
3         Scanner sc = new Scanner(System.in);
4
5         System.out.println("Enter a sentence:");
6         String sentence = sc.nextLine();
7
8         // Convert to lowercase and split into words
9         String[] words = sentence.toLowerCase().split("\\s+");
10
11        HashMap<String, Integer> wordCount = new HashMap<>();
12
13        // Count frequency of each word
14        for (String word : words) {
15            if (wordCount.containsKey(word)) {
16                wordCount.put(word, wordCount.get(word) + 1);
17            } else {
18                wordCount.put(word, 1);
19            }
20        }
21
22    }
23
24    // Display result
25    System.out.println("\nWord Frequency:");
26    for (String key : wordCount.keySet()) {
27        System.out.println(key + " : " + wordCount.get(key));
28    }
29 }
30
31 }
```

Output:

```
Enter a sentence:
Raj is silent but Raj is powerful and wealthy

Word Frequency:
but : 1
silent : 1
powerful : 1
wealthy : 1
and : 1
raj : 2
is : 2

--- Code Execution Successful ---
```

22. Design a class BankAccount with methods for deposit, withdraw, and balance inquiry.

The screenshot shows a Java code editor with a dark theme. The code implements a bank account menu with options for deposit, withdrawal, and balance inquiry. It uses a switch statement to handle user input. The output window shows the menu options, user choices, and the resulting account balance.

```
Main.java
13
14    case 1:
15        System.out.print("Enter deposit amount: ");
16        double dep = sc.nextDouble();
17        account.deposit(dep);
18        break;
19
20    case 2:
21        System.out.print("Enter withdrawal amount: ");
22        double wd = sc.nextDouble();
23        account.withdraw(wd);
24        break;
25
26    case 3:
27        account.checkBalance();
28        break;
29
30    case 4:
31        System.out.println("Thank you!");
32        sc.close();
33        return;
34
35    default:
36        System.out.println("Invalid choice!");
37    }
38
39 }
40
41 }
```

Output:

```
--- Bank Menu ---
1. Deposit
2. Withdraw
3. Balance Inquiry
4. Exit

Enter your choice: 1
Enter deposit amount: 1800000
Deposited: 1800000.0

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Balance Inquiry
4. Exit

Enter your choice: 3
Current Balance: 1800000.0

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Balance Inquiry
4. Exit

Enter your choice: |
```

23.Design a Java class Employee with the following:A method **empDetails()** to accept and display employee details.A method **salary()** to compute basic salary components.A method **total()** to calculate the total salary (including allowances/deductions).

```

Main.java
1  import java.util.Scanner;
2
3  public class Employee {
4      double basicSalary; // Basic Salary
5      double da; // DA
6      double pf; // PF Deduction
7      double grossSalary; // Gross Salary
8
9      void empDetails() {
10         Scanner sc = new Scanner(System.in);
11         System.out.print("Enter Employee ID: ");
12         int id = sc.nextInt();
13         System.out.print("Enter Employee Name: ");
14         String name = sc.nextLine();
15         System.out.print("Enter Basic Salary: ");
16         basicSalary = sc.nextDouble();
17
18         calculate();
19     }
20
21     void calculate() {
22         double hra = basicSalary * 0.10; // 10% of basic salary
23         double da = basicSalary * 0.05; // 5% deduction
24         grossSalary = basicSalary + hra + da;
25     }
26
27     // Method to calculate total salary
28     void total() {
29         double totalSalary = grossSalary - pf;
30
31         System.out.println("\n--- Salary Breakdown ---");
32         System.out.println("HRA: " + hra);
33         System.out.println("DA: " + da);
34         System.out.println("PF Deduction: " + pf);
35         System.out.println("Gross Salary: " + grossSalary);
36         System.out.println("Net Salary (Total): " + totalSalary);
37     }
38
39 }
40
41 public class Main {
42     public static void main(String[] args) {
43         Employee emp = new Employee();
44
45         emp.empDetails();
46         emp.salary();
47         emp.total();
48     }
49 }
50
51
52
53
54
55
56
57
58 }
59

```

Output

```

Enter Employee ID: 84466
Enter Employee Name: SSK
Enter Basic Salary: 60000

--- Employee Details ---
ID: 84466
Name: SSK
Basic Salary: 60000.0

--- Salary Breakdown ---
HRA: 12000.0
DA: 6000.0
PF Deduction: 3000.0
Gross Salary: 78000.0
Net Salary (Total): 75000.0

== Code Execution Successful ==

```

24.Create a Student class with marks in 3 subjects and compute the result with percentage.

```

Main.java
1  import java.util.Scanner;
2
3  public class Student {
4      int m1, m2, m3; // Marks of Subject 1, 2, 3
5      double percentage; // Percentage
6
7      void calculatePercentage() {
8          percentage = (m1 + m2 + m3) / 3;
9      }
10
11     void display() {
12         System.out.println("Student Name: " + name);
13         System.out.println("Marks: " + m1 + ", " + m2 + ", " + m3);
14         System.out.println("Percentage: " + percentage + "%");
15     }
16
17 }
18
19 public class Main {
20     public static void main(String[] args) {
21         Scanner sc = new Scanner(System.in);
22
23         System.out.print("Enter Student Name: ");
24         String name = sc.nextLine();
25
26         Student s = new Student();
27
28         System.out.print("Enter Marks of Subject 1: ");
29         s.m1 = sc.nextInt();
30         System.out.print("Enter Marks of Subject 2: ");
31         s.m2 = sc.nextInt();
32         System.out.print("Enter Marks of Subject 3: ");
33         s.m3 = sc.nextInt();
34
35         s.calculatePercentage();
36         s.display();
37
38         sc.close();
39     }
40 }
41
42
43
44

```

Output

```

Enter Student Name: SSK
Enter Marks of Subject 1: 87
Enter Marks of Subject 2: 82
Enter Marks of Subject 3: 93
Student Name: SSK
Marks: 87, 82, 93
Percentage: 87.33333333333333%

== Code Execution Successful ==

```

25.Create a class Volume and create three constructor with one arg,two arg and three arg with the help of constructor overloading concept.

Main.java

```
1 // Constructor with 2 arguments (Cylinder: pi*r^2*h)
2 Volume(double radius, double height) {
3     volume = 3.14 * radius * radius * height;
4 }
5
6 // Constructor with 3 arguments (Rectangular Box: l*b*h)
7 Volume(double length, double breadth, double height) {
8     volume = length * breadth * height;
9 }
10
11 void display() {
12     System.out.println("Volume = " + volume);
13 }
14
15 public class Main {
16     public static void main(String[] args) {
17         Volume cube = new Volume(5);
18         cube.display();
19
20         Volume cylinder = new Volume(3, 7);
21         cylinder.display();
22
23         Volume box = new Volume(4, 6, 8);
24         box.display();
25     }
26 }
27
```

Output

```
Volume = 125.0
Volume = 197.82
Volume = 192.0
== Code Execution Successful ==
```

26.Write a program to count number of object/instances created in a class.

Main.java

```
1 public class Main {
2
3     static int count = 0; // static variable to count objects
4
5     Main() {
6         count++; // increment whenever constructor runs
7     }
8
9     public static void main(String[] args) {
10        Main obj1 = new Main();
11        Main obj2 = new Main();
12        Main obj3 = new Main();
13
14        System.out.println("Number of objects created: " + count);
15    }
16 }
17
```

Output

```
Number of objects created: 3
== Code Execution Successful ==
```

27.Design a class hierarchy using inheritance: Person → Employee → Manager.

Main.java

```
50 class Manager extends Employee {
51     private String department;
52     private double bonus;
53
54     public Manager(String name, int age, String employeeId, double
55         salary,
56         String department, double bonus) {
57         super(name, age, employeeId, salary); // call Employee's
58         constructor
59         this.department = department;
60         this.bonus = bonus;
61     }
62
63     public void displayManager() {
64         displayEmployee(); // show employee info first
65         System.out.println("Department: " + department);
66         System.out.println("Bonus: Rs" + bonus);
67     }
68 }
69
70 public class Main {
71     public static void main(String[] args) {
72         // create a Manager (inherits all fields)
73         Manager m = new Manager("SSK", 21, "7517882753", 75000,
74             "Analytics", 15000);
75         m.displayManager();
76     }
77 }
```

Output

```
Name: SSK
Age: 21
Employee ID: 7517882753
Salary: Rs75000.0
Department: Analytics
Bonus: Rs15000.0
== Code Execution Successful ==
```

29.Use abstract classes to design a Vehicle class with car and bike subclasses.

Main.java

```
25
26 }
27
28 class Bike extends Vehicle {
29     Bike(String brand, int speed) {
30         super(brand, speed);
31     }
32
33     void start() {
34         System.out.println("Bike starts with a kick or self-start."
35             );
36     }
37
38 public class Main {
39     public static void main(String[] args) {
40         Vehicle car = new Car("Toyota", 180);
41         Vehicle bike = new Bike("Yamaha", 120);
42
43         System.out.println("---- Car Details ---");
44         car.showDetails();
45         car.start();
46
47         System.out.println("\n---- Bike Details ---");
48         bike.showDetails();
49         bike.start();
50     }
51 }
```

Output

```
--- Car Details ---
Brand: Toyota
Top Speed: 180 km/h
Car starts with a key or push button.

--- Bike Details ---
Brand: Yamaha
Top Speed: 120 km/h
Bike starts with a kick or self-start.

== Code Execution Successful ==
```

30.Implement a stack using an array with push, pop, and display operations.

The screenshot shows a Java code editor with a dark theme. The code is named Main.java and implements a stack using an array. It includes methods for pushing elements onto the stack, popping elements from the stack, and displaying the current elements. The output window shows the execution results: three elements (10, 20, 30) are pushed onto the stack, then one element (30) is popped, and finally the remaining elements (20, 10) are displayed. A message at the end indicates successful code execution.

```
Main.java
1  class Main {
2      int[] stack;
3      int top;
4
5      // Push element onto stack
6      void push(int data) {
7          if (top == stack.length - 1) {
8              System.out.println("Stack is Full!");
9          } else {
10             stack[++top] = data;
11             System.out.println(data + " pushed");
12         }
13     }
14
15     // Pop element from stack
16     int pop() {
17         if (top == -1) {
18             System.out.println("Stack is Empty!");
19         } else {
20             int data = stack[top];
21             stack[top] = null;
22             top--;
23             return data;
24         }
25     }
26
27     // Display stack elements
28     void display() {
29         if (top == -1) {
30             System.out.println("Stack is Empty!");
31         } else {
32             System.out.print("Stack elements: ");
33             for (int i = top; i >= 0; i--) {
34                 System.out.print(stack[i] + " ");
35             }
36             System.out.println();
37         }
38     }
39
40     public static void main(String[] args) {
41         Main s = new Main();
42
43         s.push(10);
44         s.push(20);
45         s.push(30);
46         s.display();
47         s.pop();
48         s.display();
49     }
50 }
```

Output:

```
10 pushed
20 pushed
30 pushed
Stack elements: 30 20 10
30 popped
Stack elements: 20 10
== Code Execution Successful ==
```

31.Create a queue using an array with enqueue and dequeue operations.

The screenshot shows a Java code editor with a dark theme. The code is named SimpleQueue.java and implements a queue using a fixed-size array. It includes methods for enqueuing elements into the queue and dequeuing elements from the queue. The output window shows the execution results: three elements (10, 20, 30) are added to the queue, then one element (10) is removed from the queue, and finally the remaining elements (20, 30) are displayed. A message at the end indicates successful code execution.

```
SimpleQueue.java
1  class SimpleQueue {
2      int front = -1;
3      int rear = -1;
4      int[] queue = new int[5]; // fixed size array
5
6      // Enqueue
7      void enqueue(int data) {
8          if (rear == queue.length - 1) {
9              System.out.println("Queue is Full!");
10         } else {
11             if (front == -1) {
12                 front = 0;
13             }
14             queue[++rear] = data;
15             System.out.println(data + " added to queue");
16         }
17     }
18
19     // Dequeue
20     void dequeue() {
21         if (front == -1 || front > rear) {
22             System.out.println("Queue is Empty!");
23         } else {
24             System.out.println(queue[front] + " removed from queue");
25             front++;
26         }
27     }
28 }
```

Output:

```
10 added to queue
20 added to queue
30 added to queue
Queue elements: 10 20 30
10 removed from queue
Queue elements: 20 30
== Code Execution Successful ==
```

32.Design a singly linked list with insert, delete, and traverse methods.

```
SinglyLinkedList.java | Run | Output | Clear
1  public class SinglyLinkedList {
2
3     Node head;
4
5     void insert(int data) {
6         Node newNode = new Node(data);
7
8         if (head == null) {
9             head = newNode;
10        } else {
11            Node temp = head;
12            while (temp != null) {
13                if (temp.data == data) {
14                    System.out.println("Element already found!");
15                    return;
16                }
17                temp = temp.next;
18            }
19            temp.next = newNode;
20        }
21    }
22
23    void traverse() {
24        Node temp = head;
25        while (temp != null) {
26            System.out.print(temp.data + " ");
27            temp = temp.next;
28        }
29    }
30
31    void delete(int data) {
32        Node temp = head;
33        Node previous = null;
34
35        if (temp == null) {
36            System.out.println("List is Empty!");
37            return;
38        }
39
40        while (temp != null) {
41            if (temp.data == data) {
42                if (previous == null) {
43                    head = temp.next;
44                } else {
45                    previous.next = temp.next;
46                }
47                System.out.println(data + " deleted");
48                return;
49            }
50            previous = temp;
51            temp = temp.next;
52        }
53    }
54
55    public static void main(String[] args) {
56        SinglyLinkedList list = new SinglyLinkedList();
57
58        list.insert(10);
59        list.insert(20);
60        list.insert(30);
61        list.traverse();
62
63        list.delete(20);
64        list.traverse();
65
66        list.delete(50);
67    }
68 }
```

Output:

```
10 inserted
20 inserted
30 inserted
Linked List: 10 20 30
20 deleted
Linked List: 10 30
Element not found!
== Code Execution Successful ==
```

33.Create an interface **Shape** with a method **area()** and implement it in **Circle**, **Square**.

```
Main.java | Run | Output | Clear
1  import java.util.Scanner;
2
3  public class Main {
4
5      interface Shape {
6          void area();
7      }
8
9      static class Circle implements Shape {
10         double radius;
11         Circle(double radius) {
12             this.radius = radius;
13         }
14         public void area() {
15             double area = Math.PI * radius * radius;
16             System.out.println("Area of Circle: " + area);
17         }
18     }
19
20     static class Square implements Shape {
21         double side;
22         Square(double side) {
23             this.side = side;
24         }
25         public void area() {
26             double area = side * side;
27             System.out.println("Area of Square: " + area);
28         }
29     }
30 }
```

Output:

```
Enter radius of Circle: 5
Enter side of Square: 6
Area of Circle: 78.53981633974483
Area of Square: 36.0
== Code Execution Successful ==
```

34. Demonstrate multiple inheritance in Java using two interfaces and one implementing class.

The screenshot shows a Java code editor interface. The code in the editor is:

```
Main.java [Java] Run Output Clear
1- public class Main {
2
3-     interface A {
4-         void methodA();
5-     }
6
7-     interface B {
8-         void methodB();
9-     }
10
11    public static class Test implements A, B {
12        public void methodA() {
13            System.out.println("Method A from Interface A");
14        }
15
16        public void methodB() {
17            System.out.println("Method B from Interface B");
18        }
19    }
20
21    public static void main(String[] args) {
22        Test obj = new Test();
23        obj.methodA();
24        obj.methodB();
25    }
26 }
27
```

The output window displays the results of running the code:

```
Method A from Interface A
Method B from Interface B
--- Code Execution Successful ---
```

36. Develop a program to count the number of words in a text file.

The screenshot shows a Java code editor interface. The code in the editor is:

```
Main.java [Java] Run Output Clear
1- import java.util.Scanner;
2
3- public class Main {
4-     public static void main(String[] args) {
5-         Scanner sc = new Scanner(System.in);
6-         // As we cannot add file on compiler
7-
8-         System.out.println("Enter your text:");
9-         String text = sc.nextLine();
10
11        // Remove leading/trailing spaces and split by any space or tab
12        String[] words = text.trim().split("\\s+");
13
14        // Count words (handle empty input safely)
15        int wordCount = text.trim().isEmpty() ? 0 : words.length;
16
17        System.out.println("Total number of words: " + wordCount);
18
19        sc.close();
20    }
21 }
```

The output window displays the results of running the code after entering text:

```
Enter your text:
Hi my name is sharvari muley , i'm pursuing my btech degree from St.vincent pallotti college in cse(ds)
Total number of words: 18
--- Code Execution Successful ---
```