

Sara Fatma [Java Assignment-1]

Basic Java:

- 1) Write a program to calculate the area of a circle, rectangle, or triangle based on user input.

```
// Write a program to calculate the area of a circle, rectangle, or triangle based on user input.

import java.util.*;
class Question1{
    public static void main(String []args){

        Scanner sc = new Scanner(System.in);

        System.out.println("Choose the shape to calculate area:");
        System.out.println("1. Circle");
        System.out.println("2. Rectangle");
        System.out.println("3. Triangle");
        System.out.print("Enter your choice (1/2/3): ");

        int choice = sc.nextInt();

        switch (choice) {
            case 1:
                System.out.print("Enter the radius of the circle: ");
                float radius = sc.nextFloat();
                float circleArea = 3.14f * radius * radius;
                System.out.println("Area of the circle:"+ circleArea);
                break;

            case 2:
                System.out.print("Enter the length of the rectangle: ");
                float length = sc.nextFloat();
                System.out.print("Enter the width of the rectangle: ");
                float width = sc.nextFloat();
                float rectangleArea = length * width;
                System.out.println("Area of the rectangle: "+ rectangleArea);
                break;

            case 3:
                System.out.print("Enter the base of the triangle: ");
                float base = sc.nextFloat();
                System.out.print("Enter the height of the triangle: ");
                float height = sc.nextFloat();
                float triangleArea = 0.5f * base * height;
                System.out.println("Area of the triangle: "+ triangleArea);
                break;

            default:
                System.out.println("Invalid choice. Please select 1, 2, or 3.");
        }

        sc.close();
    }
}
```

```
✓ TERMINAL
● PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions> cd
Choose the shape to calculate area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1/2/3): 1
Enter the radius of the circle: 4
Area of the circle:50.24
```

2) Create a program to check if a number is even or odd.

```
J Question2.java
1 //Create a program to check if a number is even or odd.
2
3 import java.util.*;
4 class Question2 {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         System.out.println("Enter a number to check if its even or odd: ");
8         int number = sc.nextInt();
9
10        if (number % 2 == 0){
11            System.out.println("Number is even");
12        }else{
13            System.out.println("Number is odd");
14        }
15
16        sc.close();
17
18    }
19 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
✓ TERMINAL
● PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions> cd "d:\nucleusTeq\
● Enter a number to check if its even or odd:
3
Number is odd
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions>
```

3) Implement a program to find the factorial of a given number.

The screenshot shows a Java code editor with a dark theme. The code in `Question3.java` is:

```
1 // Implement a program to find the factorial of a given number.
2 import java.util.*;
3
4 public class Question3 {
5
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8
9         System.out.print("Enter a number whose factorial is to be printed: ");
10        int number = sc.nextInt();
11        long factorial = 1;
12
13        for(int i = 1; i <= number; i++){
14            factorial = factorial * i;
15        }
16
17        System.out.println("Factorial of " + number + " is " + factorial);
18        sc.close();
19    }
20
21 }
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
▼ TERMINAL
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions"
● Enter a number whose factorial is to be printed: 5
Factorial of 5 is 120
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions>
```

4) Write a program to print the Fibonacci sequence up to a specified number.

The screenshot shows a Java code editor with a dark theme. The code in `Question4.java` is:

```
1 //Write a program to print the Fibonacci sequence up to a specified number.
2
3 import java.util.Scanner;
4
5 public class Question4 {
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         System.out.print("Enter the number of terms: ");
9         int n = sc.nextInt();
10        sc.close();
11
12        System.out.print("Fibonacci Series: ");
13        for (int i = 0; i < n; i++) {
14            System.out.print(fibonacci(i) + " ");
15        }
16    }
17
18    public static int fibonacci(int n) {
19        if (n <= 1)
20            return n;
21        return fibonacci(n - 1) + fibonacci(n - 2);
22    }
23
24 }
```

Below the code editor is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
▼ TERMINAL
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions"
● Enter the number of terms: 10
Fibonacci Series: 0 1 1 2 3 5 8 13 21 34
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions>
```

- 5) Use loops to print patterns like a triangle or square.

The screenshot shows a Java code editor with the file named "Question5.java". The code prints a square pattern of asterisks based on user input. The terminal window below shows the execution of the program and its output for a size of 5.

```
1 // Use loops to print patterns like a triangle or square.
2 import java.util.*;
3 public class Question5
4 {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         System.out.print("Enter the size of the square: ");
8         int n = sc.nextInt();
9         sc.close();
10
11        for (int i = 0; i < n; i++) {
12            for (int j = 0; j < n; j++) {
13                System.out.print("* ");
14            }
15            System.out.println();
16        }
17    }
18 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

▼ TERMINAL

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\BasicJavaQuestions>
● Enter the size of the square: 5
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Data Types and Operators:

- 1) Explain the difference between primitive and reference data types with examples.

Feature	Primitive Data Type	Reference Data Type
Storage	Stores actual value	Stores memory address (reference)
Memory Location	Stored in stack	Object in heap , reference in stack
Default Value	0, false, '\u0000'	null
Methods Available?	No methods	Methods available

Feature	Primitive Data Type	Reference Data Type
Mutability	Immutable	Mutable (modifiable object properties)
Performance	Faster (direct access)	Slower (memory access via reference)

Example of Primitive Data Type

The screenshot shows a Java code editor with the file `Question1.java`. The code defines a `Question1` class with a `main` method. It prints a primitive `int` value and objects of `String` and `Student` classes. The `Student` class has a constructor and a `getName` method. The terminal below shows the execution of the program and its output.

```
J Question1.java ×
J Question1.java
1 import java.util.*;
2 public class Question1 {
3     public static void main(String[] args) {
4
5         //Primitive data type
6         int num = 10;
7
8         System.out.println("primitive data type :" + num);
9
10        //Non Primitive data type
11        String name = "Sara Fatma";
12        int[] numbers = {10, 20, 30, 40}; // Array reference
13        Student student = new Student("Sara", 21); // Object reference
14
15        System.out.println("Name: " + name);
16        System.out.println("First number in array: " + numbers[0]);
17        System.out.println("Student Name: " + student.getName());
18    }
19 }
20
21 class Student {
22     private String name;
23     private int age;
24
25     public Student(String name, int age) {
26         this.name = name;
27         this.age = age;
28     }
29
30     public String getName() {
31         return name;
32     }
33 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> ▾ TERMINAL

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\DataTypeAndOperators\Question1.java ; if ($?) { java Question1 }
primitive data type :10
Name: Sara Fatma
First number in array: 10
Student Name: Sara
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\DataTypeAndOperators\
```

2) Write a program to demonstrate the use of arithmetic, logical, and relational operators.

```
J Question2.java
1 import java.util.*;
2
3 public class Question2 {
4     public static void main(String[] args) {
5         int[] arr = {1, 2, 3, 4, 5,
6             // Arithmetic Operators
7             int a = 10, b = 5;
8             System.out.println("Arithmetic Operators:");
9             System.out.println("a + b = " + (a + b)); // Addition
10            System.out.println("a - b = " + (a - b)); // Subtraction
11            System.out.println("a * b = " + (a * b)); // Multiplication
12            System.out.println("a / b = " + (a / b)); // Division
13            System.out.println("a % b = " + (a % b)); // Modulus
14
15            // Relational Operators
16            System.out.println("\nRelational Operators:");
17            System.out.println("a == b: " + (a == b)); // Equal to
18            System.out.println("a != b: " + (a != b)); // Not equal to
19            System.out.println("a > b: " + (a > b)); // Greater than
20            System.out.println("a < b: " + (a < b)); // Less than
21            System.out.println("a >= b: " + (a >= b)); // Greater than or equal to
22            System.out.println("a <= b: " + (a <= b)); // Less than or equal to
23
24            // Logical Operators
25            boolean x = true, y = false;
26            System.out.println("\nLogical Operators:");
27            System.out.println("x && y: " + (x && y)); // Logical AND
28            System.out.println("x || y: " + (x || y)); // Logical OR
29            System.out.println("!x: " + (!x)); // Logical NOT
30        }
31    }
```

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\DataTypeAndOperatorsQuestions> cd "d:
Question2.java" ; if ($?) { java Question2 }
● Arithmetic Operators:
a + b = 15
a - b = 5
a * b = 50
a / b = 2
a % b = 0

Relational Operators:
a == b: false
a != b: true
a > b: true
a < b: false
a >= b: true
a <= b: false

Logical Operators:
x && y: false
x || y: true
!x: false
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\DataTypeAndOperatorsQuestions>
```

- 3) Create a program to convert a temperature from Celsius to Fahrenheit and vice versa.

The screenshot shows a Java code editor with the following code:

```
Question3.java
1 import java.util.*;
2
3 public class Question3 {
4     public static void main(String[] args) {
5
6         Scanner sc = new Scanner(System.in);
7
8         System.out.println("Temperature Converter");
9         System.out.println("1: Celsius to Fahrenheit");
10        System.out.println("2: Fahrenheit to Celsius");
11        System.out.print("Choose an option (1 or 2): ");
12        int choice = sc.nextInt();
13
14        if (choice == 1) {
15            System.out.print("Enter temperature in Celsius: ");
16            float celsius = sc.nextFloat();
17            float fahrenheit = (celsius * 9/5) + 32;
18            System.out.println("Temperature in Fahrenheit: " + fahrenheit);
19        } else if (choice == 2) {
20            System.out.print("Enter temperature in Fahrenheit: ");
21            float fahrenheit = sc.nextFloat();
22            float celsius = (fahrenheit - 32) * 5/9;
23            System.out.println("Temperature in Celsius: " + celsius);
24        } else {
25            System.out.println("Invalid choice!");
26        }
27
28        sc.close();
29    }
30 }
```

Below the code editor, there is a terminal window showing the execution of the program:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
▼ TERMINAL
Enter temperature in Celsius: 20
Temperature in Fahrenheit: 68.0
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\DataTypeAndOperatorsQuestions> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\DataTypeAndOperatorsQuestions"
question3.java ; if ($?) { java Question3 }
Temperature Converter
1: Celsius to Fahrenheit
2: Fahrenheit to Celsius
Choose an option (1 or 2): 2
Enter temperature in Fahrenheit: 68
Temperature in Celsius: 20.0
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\DataTypeAndOperatorsQuestions>
```

Control Flow Statements:

- 1) Write a program to check if a given number is prime using an if-else statement.

The screenshot shows a Java code editor with a file named `Question1.java`. The code implements a prime number checker using a for loop and an if-else statement. It prompts the user for input, checks if the number is prime, and prints the result. The terminal below shows the execution of the code and its output for the number 20.

```
J Question1.java X
J Question1.java
1 import java.util.*;
2
3 public class Question1 {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.print("Enter a number: ");
7         int num = sc.nextInt();
8
9
10        if (num <= 1) {
11            System.out.println(num + " is not a prime number.");
12        } else {
13            boolean isPrime = true;
14            for (int i = 2; i <= Math.sqrt(num); i++) {
15                if (num % i == 0) {
16                    isPrime = false;
17                    break;
18                }
19            }
20
21            if (isPrime) {
22                System.out.println(num + " is a prime number.");
23            } else {
24                System.out.println(num + " is not a prime number.");
25            }
26        }
27        sc.close();
28    }
29 }
30 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ..

> ▾ TERMINAL

PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ControlFlowStatement
ns> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ControlFlowStat
estions\" ; if (\$?) { javac Question1.java } ; if (\$?) { java Question1 }
Enter a number: 20
20 is not a prime number.

2) Implement a program to find the largest number among three given numbers using a conditional statement.

The screenshot shows a Java code editor with the file `Question2.java` open. The code reads three integers from the user and prints the largest one. Below the editor is a terminal window showing the execution of the program and its output.

```
1 Question2.java
2 import java.util.*;
3
4 public class Question2 {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7
8         System.out.print("Enter first number: ");
9         int num1 = sc.nextInt();
10
11        System.out.print("Enter second number: ");
12        int num2 = sc.nextInt();
13
14        System.out.print("Enter third number: ");
15        int num3 = sc.nextInt();
16
17        sc.close();
18
19        int largest;
20
21        if (num1 >= num2 && num1 >= num3) {
22            largest = num1;
23        } else if (num2 >= num1 && num2 >= num3) {
24            largest = num2;
25        } else {
26            largest = num3;
27        }
28
29        System.out.println("The largest number is: " + largest);
30    }
31 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> **TERMINAL**

```
Enter second number: 111
Enter third number: 4
The largest number is: 222
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ControlFlowStat
● ns> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ControlFlo
estions\" ; if ($?) { javac Question2.java } ; if ($?) { java Question2
Enter first number: -1
Enter second number: -3
Enter third number: 2
The largest number is: 2
```

3) Use a for loop to print a multiplication table.

The screenshot shows a Java code editor with a file named Question3.java. The code prints the multiplication table of a user-specified number. The terminal window shows the output for the number 9, displaying the table from 9x1 to 9x10.

```
Question3.java
1 import java.util.*;
2
3 public class Question3 {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.print("Enter a number: ");
7         int num = sc.nextInt();
8
9         System.out.println("Multiplication Table of " + num);
10        for (int i = 1; i <= 10; i++) {
11            System.out.println(num + " x " + i + " = " + (num * i));
12        }
13
14        sc.close();
15    }
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ControlFlowStatementQuestions>
```

4) Create a program to calculate the sum of even numbers from 1 to 10 using a while loop.

The screenshot shows a Java code editor with a file named Question4.java. The code uses a while loop to sum even numbers from 2 to 10. The terminal window shows the output of the program.

```
Question4.java
1 import java.util.*;
2
3 public class Question4 {
4     public static void main(String[] args) {
5         int num = 2, sum = 0;
6
7         while (num <= 10) {
8             sum += num;
9             num += 2;
10        }
11
12        System.out.println("Sum of even numbers from 1 to 10: " + sum);
13    }
14 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL ...

TERMINAL

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ControlFlowStatementQuestions> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ControlFlowStatementQuestions\" ; if ($?) { javac Question4.java } ; if ($?) { java Question4 }
● Sum of even numbers from 1 to 10: 30
```

Arrays:

- 1) Write a program to find the average of elements in an array.

The screenshot shows a Java code editor with a dark theme. The code is named `Question1.java` and is located in a folder named `ArrayQuestions`. The code itself is as follows:

```
1 //Write a program to find the average of elements in an array.
2 import java.util.*;
3
4 public class Question1 {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7
8         System.out.println("Enter the number of elements in array");
9
10        int n = sc.nextInt();
11
12        int []arr = new int[n];
13
14        int sum = 0;
15        System.out.println("Enter the elements of array :");
16        for(int i= 0; i<n ; i++){
17            arr[i] = sc.nextInt();
18            sum += arr[i];
19        }
20
21        float avg = (float) sum / n;
22
23        System.out.println("Average of the elements: " + avg);
24
25        sc.close();
26    }
27 }
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, showing the following terminal session:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAs
Enter the number of elements in array
5
Enter the elements of array :
1
2
3
4
5
Average of the elements: 3.0
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ArrayQuestions>
```

- 2) Implement a function to sort an array in ascending order using bubble sort or selection sort.

```
J Question2.java X
ArrayQuestions > J Question2.java
1  //implement a function to sort an array in ascending order using bubble sort
2  import java.util.*;
3
4  public class Question2 {
5      public static void bubbleSort(int[] arr) {
6          int n = arr.length;
7          boolean swapped;
8
9          for (int i = 0; i < n - 1; i++) {
10              swapped = false;
11              for (int j = 0; j < n - 1 - i; j++) {
12                  if (arr[j] > arr[j + 1]) {
13                      // Swap arr[j] and arr[j+1]
14                      int temp = arr[j];
15                      arr[j] = arr[j + 1];
16                      arr[j + 1] = temp;
17                      swapped = true;
18                  }
19              }
20              // If no two elements were swapped in the inner loop, the array is sorted
21              if (!swapped) break;
22          }
23      }
24
25      public static void main(String[] args) {
26          Scanner sc = new Scanner(System.in);
27
28          System.out.print("Enter the number of elements: ");
29          int n = sc.nextInt();
30
31          int[] arr = new int[n];
32
33          System.out.println("Enter the elements:");
34          for (int i = 0; i < n; i++) {
35              arr[i] = sc.nextInt();
36          }
37
38          bubbleSort(arr);
39
40          System.out.println("Sorted array in ascending order:");
41          for (int num : arr) {
42              System.out.print(num + " ");
43          }
44
45          sc.close();
46      }
47  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

▼ TERMINAL

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma"
● Enter the number of elements: 10
Enter the elements:
12
3
111
765
233
0
-3
55
234
975
Sorted array in ascending order:
-3 0 3 12 55 111 233 234 765 975
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ArrayQuestions>
```

- 3) Create a program to search for a specific element within an array using linear search.

```
J Question3.java X
ArrayQuestions > J Question3.java
1  //Create a program to search for a specific element within an array using linear search
2
3  import java.util.*;
4
5  public class Question3 {
6      public static int linearSearch(int[] arr, int key) {
7          for (int i = 0; i < arr.length; i++) {
8              if (arr[i] == key) {
9                  return i;
10             }
11         }
12     return -1;
13 }
14
15    public static void main(String[] args) {
16        Scanner sc = new Scanner(System.in);
17
18        System.out.print("Enter the number of elements: ");
19        int n = sc.nextInt();
20
21        int[] arr = new int[n];
22
23        System.out.println("Enter the elements:");
24        for (int i = 0; i < n; i++) {
25            arr[i] = sc.nextInt();
26        }
27
28        System.out.print("Enter the element to search: ");
29        int key = sc.nextInt();
30
31        int index = linearSearch(arr, key);
32
33        if (index != -1) {
34            System.out.println("Element found at index: " + index);
35        } else {
36            System.out.println("Element not found in the array.");
37        }
38
39        sc.close();
40    }
41 }
42
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma"
● Enter the number of elements: 5
Enter the elements:
12
75
7
90
23
Enter the element to search: 75
Element found at index: 1
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\ArrayQuestions:
```

Object Oriented Programming (OOP):

1. Create a class to represent a student with attributes like name, roll number, and marks.

The screenshot shows a Java code editor with four tabs at the top: Question1.java, Question2.java, Question3.java, and Question4.java. The Question1.java tab is active, displaying the following code:

```
1 //Create a class to represent a student with attributes like name, roll number, and marks.
2
3 import java.util.*;
4 class Student{
5
6     String name;
7     int rollNumber;
8     float marks;
9
10
11    public Student(String name, int rollNumber, float marks){
12        this.name = name;
13        this.rollNumber = rollNumber;
14        this.marks = marks;
15    }
16
17    public void displayStudentDetails(){
18        System.out.println("Name: " + name);
19        System.out.println("Roll Number: " + rollNumber);
20        System.out.println("Marks: " + marks);
21    }
22 }
23 public class Question1 {
24     public static void main(String[] args) {
25         Student studentObj = new Student("Sara Fatma",1,97.5f);
26         studentObj.displayStudentDetails();
27     }
28 }
```

Below the code editor is a terminal window showing the execution of the code. The terminal tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, with TERMINAL being the active tab. The terminal output is as follows:

```
> PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma"
● Name: Sara Fatma
  Roll Number: 1
  Marks: 97.5
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\OopQuestion>
```

2. Implement inheritance to create a "GraduateStudent" class that extends the "Student" class with additional features.

```
Question1.java Question2.java X Question3.java Question4.java
OopQuestion > Question2.java
1 import java.util.*;
2
3 class Student {
4     String name;
5     int rollNumber;
6     float marks;
7
8     Student(String name, int rollNumber, float marks) {
9         this.name = name;
10        this.rollNumber = rollNumber;
11        this.marks = marks;
12    }
13
14    public void displayStudentDetails() {
15        System.out.println("Name: " + name);
16        System.out.println("Roll Number: " + rollNumber);
17        System.out.println("Marks: " + marks);
18    }
19 }
20
21 class GraduateStudent extends Student {
22     String course;
23
24     GraduateStudent(String name, int rollNumber, float marks, String course) {
25         super(name, rollNumber, marks);
26         this.course = course;
27     }
28
29     @Override
30     public void displayStudentDetails() {
31         super.displayStudentDetails();
32         System.out.println("Course: " + course);
33     }
34
35 }
36
37 class Question2 {
38     public static void main(String[] args) {
39         GraduateStudent gradStudent = new GraduateStudent("Sara Fatma", 1, 97.5f, "AI & ML");
40         gradStudent.displayStudentDetails();
41     }
42 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
▼ TERMINAL
● PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\OopQuestion> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\OopQuestion"
Name: Sara Fatma
Roll Number: 1
Marks: 97.5
Course: AI & ML
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\OopQuestion>
```

3. Demonstrate polymorphism by creating methods with the same name but different parameters in a parent and child class.

```
J Question1.java J Question2.java J Question3.java X J Question4.java
DopQuestion > J Question3.java
1 // Demonstrate polymorphism by creating methods with the same name but different parameters in a parent and child class.
2 import java.util.*;
3
4 class PetOwner {
5     String name;
6     int age;
7
8     PetOwner(String name, int age){
9         this.name = name;
10        this.age= age;
11    }
12
13     public void display(){
14
15         System.out.println("Name of the owner:" + name);
16         System.out.println("age of the owner:" + age);
17         System.out.println(" ");
18     }
19
20     public void display(String message){
21         System.out.println(message + ":" );
22         System.out.println(" ");
23         System.out.println("Name of the owner:" + name);
24         System.out.println("age of the owner:" + age);
25         System.out.println(" ");
26     }
27 }
28
29 class Pet extends PetOwner {
30
31     String petName;
32     int petAge;
33
34     Pet(String ownerName, int ownerAge, String petName, int petAge) {
35         super(ownerName, ownerAge);
36         this.petName = petName;
37         this.petAge = petAge;
38     }
39 }
```

```
40
41     @Override
42     public void display(){
43
44         System.out.println("Name of the owner:" + name);
45         System.out.println("age of the owner:" + age);
46         System.out.println("Name of the pet:" + petName);
47         System.out.println("age of the pet:" + petAge);
48     }
49
50
51     public void display(String message){
52         System.out.println(message + ":" );
53         System.out.println(" ");
54         System.out.println("Name of the owner:" + name);
55         System.out.println("age of the owner:" + age);
56         System.out.println("Name of the pet:" + petName);
57         System.out.println("age of the pet:" + petAge);
58     }
59 }
60
61
62 public class Question3 {
63     public static void main(String[] args) {
64
65         PetOwner ownerObj1 = new PetOwner("John", 25);
66         ownerObj1.display()["Overloading display() in class PetOwner"];
67
68         System.out.println("After Overriding display() method in Pet class");
69         System.out.println(" ");
70
71         Pet petObj1 = new Pet("John", 25, "Buddy", 2);
72         petObj1.display("Overloading Display() method in Pet class");
73     }
74 }
```

The screenshot shows a terminal window with the following output:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma"
● Overloading display() in class PetOwner:

Name of the owner:John
age of the owner:25

After Overriding display() method in Pet class

Overloading Display() method in Pet class:

Name of the owner:John
age of the owner:25
Name of the pet:Buddy
age of the pet:2
```

4. Explain the concept of encapsulation with a suitable example.

The screenshot shows a Java code editor with two files open:

```
Question1.java Question2.java Question3.java Question4.java ×

OopQuestion > Question4.java
1 //Explain the concept of encapsulation with a suitable example.
2 import java.util.*;
3
4 class Person {
5     // (data hiding)
6     private String name;
7
8
9     public void setName(String name) {
10         this.name = name;
11     }
12
13
14     public String getName() {
15         return name;
16     }
17 }
18 public class Question4 {
19     public static void main(String[] args) {
20         Person p = new Person();
21         p.setName("Sara Fatma");
22         System.out.println("Name: " + p.getName());
23     }
24 }
```

Below the code editor is a terminal window with the following output:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma"
Name: Sara Fatma
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma>
```

String Manipulation:

1. Write a program to reverse a given string.

The screenshot shows a Java IDE interface with three tabs at the top: "Question1.java StringQuestions X", "Question2.java StringQuestions ●", and "Question3.java StringQuestions ▶". The "Question1.java" tab is active. The code in the editor is:

```
StringQuestions > J Question1.java
1 import java.util.*;
2 public class Question1 {
3     public static void main(String[] args) {
4         Scanner sc = new Scanner(System.in);
5         System.out.print("Enter a string: ");
6         String input = sc.nextLine();
7
8         String reversed = new StringBuilder(input).reverse().toString();
9         System.out.println("Reversed String: " + reversed);
10
11     sc.close();
12 }
13
14 }
```

Below the editor are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL". The "TERMINAL" tab is active, showing the following command-line session:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions\" ; if ($?) { javac Question1.java ; if ($?) { java Question1 } }
● Enter a string: Hello World!
Reversed String: !dlroW olleH
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions>
```

2. Implement a function to count the number of vowels in a string.

The screenshot shows a Java IDE interface with three tabs at the top: "Question1.java StringQuestions", "Question2.java StringQuestions X", and "Question3.java StringQuestions". The "Question2.java" tab is active. The code in the editor is:

```
StringQuestions > J Question2.java
1 import java.util.*;
2 public class Question2 {
3
4     public static int countVowels(String str) {
5         int count = 0;
6         str = str.toLowerCase();
7
8         for (char ch : str.toCharArray()) {
9             if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')
10                 count++;
11         }
12     return count;
13 }
14
15     public static void main(String[] args) {
16         Scanner sc = new Scanner(System.in);
17         System.out.print("Enter a string: ");
18         String input = sc.nextLine();
19
20         int vowelCount = countVowels(input);
21         System.out.println("Number of vowels: " + vowelCount);
22
23     sc.close();
24 }
25 }
```

Below the editor are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL". The "TERMINAL" tab is active, showing the following command-line session:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions"
Enter a string: Hi How are you?
Number of vowels: 6
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions>
```

3. Create a program to check if two strings are anagrams.

The screenshot shows a Java development environment with several tabs at the top: "Question1.java StringQuestions", "Question2.java StringQuestions", "Question3.java StringQuestions" (which is active), and "Question4.java StringQuestions". The main area displays the code for Question3.java:

```
StringQuestions > J Question3.java
1 import java.util.*;
2 public class Question3 {
3     public static boolean areAnagrams(String str1, String str2) {
4         str1 = str1.replaceAll("\\s", "").toLowerCase();
5         str2 = str2.replaceAll("\\s", "").toLowerCase();
6         if (str1.length() != str2.length()) {
7             return false;
8         }
9         char[] arr1 = str1.toCharArray();
10        char[] arr2 = str2.toCharArray();
11        Arrays.sort(arr1);
12        Arrays.sort(arr2);
13        return Arrays.equals(arr1, arr2);
14    }
15    public static void main(String[] args) {
16        Scanner sc = new Scanner(System.in);
17        System.out.print("Enter first string: ");
18        String str1 = sc.nextLine();
19        System.out.print("Enter second string: ");
20        String str2 = sc.nextLine();
21        if (areAnagrams(str1, str2)) {
22            System.out.println("The strings are anagrams.");
23        } else {
24            System.out.println("The strings are NOT anagrams.");
25        }
26    }
27    sc.close();
28 }
29 }
```

Below the code editor are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL". The "TERMINAL" tab is selected, showing the following command-line interaction:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions"
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions> java Question3
Enter first string: arc
Enter second string: car
The strings are anagrams.
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\StringQuestions> 
```

Advanced Topics:

1. Explain the concept of interfaces and abstract classes with examples.

Both **interfaces** and **abstract classes** are used for abstraction in Java, but they have key differences in how they work and are used.

1. Abstract Class (Partial Abstraction)

An **abstract class**:

- Can have **both abstract (unimplemented) and concrete (implemented) methods**.
- Can have **constructors, instance variables, and static methods**.
- Supports **inheritance (extends keyword)**.
- Cannot be instantiated directly.

2. Interface (Full Abstraction)

An **interface**:

- **Only contains abstract methods (before Java 8).**
- Can contain **default methods and static methods** (from Java 8).
- **Cannot have constructors or instance variables.**
- Supports **multiple inheritance** (unlike abstract classes).

Abstract Class vs Interface:

Feature	Abstract Class	Interface
Methods	Can have both abstract & concrete methods	Only abstract methods (before Java 8)
Variables	Can have instance variables	Only public, static, and final constants
Constructors	Can have constructors	Cannot have constructors
Multiple Inheritance	Not supported	Supported
Access Modifiers	Methods can be public, protected, or private	Methods are public by default
Use Case	Used when classes share common behavior	Used for defining a contract that multiple classes can follow

Abstraction example

```
AdvancedQuestions > J Abstraction.java
1  abstract class Animal {
2      protected String name;
3      public Animal(String name) {
4          this.name = name;
5      }
6      abstract void makeSound();
7      public void display() {
8          System.out.println("Animal Name: " + name);
9      }
10 }
11 class Dog extends Animal {
12     public Dog(String name) {
13         super(name);
14     }
15
16     @Override
17     void makeSound() {
18         System.out.println("Woof! Woof!");
19     }
20 }
21
22 public class Abstraction{
23     public static void main(String[] args) {
24         Dog myDog = new Dog("Buddy");
25         myDog.display();
26         myDog.makeSound();
27     }
28 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

▼ TERMINAL

- PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions"
- Animal Name: Buddy
- Woof! Woof!
- PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions>

Interface Example

```
1 import java.util.*;
2 interface Vehicle {
3     int MAX_SPEED = 120;
4
5     void drive();
6     default void showSpeedLimit() {
7         System.out.println("Max speed is " + MAX_SPEED + " km/h");
8     }
9 }
10 class Car implements Vehicle {
11     @Override
12     public void drive() {
13         System.out.println("Car is driving...");
14     }
15 }
16 public class InterfaceExample {
17     public static void main(String[] args) {
18         Car myCar = new Car();
19         myCar.drive();
20         myCar.showSpeedLimit();
21     }
22 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> ▾ TERMINAL

- PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions"
Example
Car is driving...
Max speed is 120 km/h
- PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions>

2. Create a program to handle exceptions using try-catch blocks.

```
2
3 public class Question2 {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6
7         try {
8             System.out.print("Enter numerator: ");
9             int numerator = sc.nextInt();
10
11            System.out.print("Enter denominator: ");
12            int denominator = sc.nextInt();
13            int result = numerator / denominator;
14            System.out.println("Result: " + result);
15
16        } catch (ArithmaticException e) {
17            System.out.println("Error: Cannot divide by zero!");
18        } catch (Exception e) {
19            System.out.println("Error: Invalid input. Please enter numbers only.");
20        } finally {
21            sc.close();
22            System.out.println("Program execution completed.");
23        }
24    }
25 }
26
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

> ▾ TERMINAL

- PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions"
Enter numerator: 10
Enter denominator: 0
Error: Cannot divide by zero!
Program execution completed.
- PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions>

3. Implement a simple file I/O operation to read data from a text file.

The screenshot shows a Java IDE interface with several tabs at the top: Question3.java, FileReadExample.java (active), sample.txt, and Question4.java. The FileReadExample.java tab contains the following code:

```
1 import java.io.*;
2 public class FileReadExample {
3     public static void main(String[] args) {
4         String fileName = "sample.txt";
5         try {
6             FileReader fileReader = new FileReader(fileName);
7             BufferedReader bufferedReader = new BufferedReader(fileReader);
8             String line;
9             System.out.println("Contents of the file:");
10            while ((line = bufferedReader.readLine()) != null) {
11                System.out.println(line);
12            }
13            bufferedReader.close();
14            fileReader.close();
15        } catch (FileNotFoundException e) {
16            System.out.println("Error: File not found!");
17        } catch (IOException e) {
18            System.out.println("Error: An I/O error occurred.");
19        }
20    }
21 }
```

Below the code editor is a navigation bar with PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, showing the following terminal output:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions"
● Error: File not found!
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions"
{ java FileReadExample }
Contents of the file:
Inside sample.txt
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions>
```

4. Explore multithreading in Java to perform multiple tasks concurrently.

The screenshot shows a Java IDE interface with several tabs at the top: Question3.java, Question4.java (active), sample.txt, and Question4.java. The Question4.java tab contains the following code:

```
1 import java.util.*;
2
3 class MyThread extends Thread {
4     public void run() {
5         for (int i = 1; i <= 5; i++) {
6             System.out.println(Thread.currentThread().getName() + " - Count: " + i);
7             try {
8                 Thread.sleep(500); // Pause for 500ms
9             } catch (InterruptedException e) {
10                 System.out.println("Thread interrupted.");
11             }
12         }
13     }
14 }
15
16 public class Question4 {
17     public static void main(String[] args) {
18         MyThread thread1 = new MyThread();
19         MyThread thread2 = new MyThread();
20         thread1.start();
21         thread2.start();
22     }
23 }
```

Below the code editor is a navigation bar with PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected, showing the following terminal output:

```
PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma> cd "d:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions"
● Thread-1 - Count: 1
Thread-0 - Count: 1
Thread-1 - Count: 2
Thread-0 - Count: 2
Thread-1 - Count: 3
Thread-0 - Count: 3
Thread-1 - Count: 4
Thread-0 - Count: 4
Thread-1 - Count: 5
Thread-0 - Count: 5
○ PS D:\nucleusTeq\Training\java\JavaAssignment1-SaraFatma\AdvancedQuestions>
```