Control Flow in Java

Table of Contents

Conditional Statements

What are Conditional Statements?

Conditional statements allow you to execute a block of code based on a condition. This is useful when you want your program to make decisions and perform different actions based on different conditions.

if-else Statement

Syntax:

```
if (condition) {
    // code to be executed if condition is true
} else {
    // code to be executed if condition is false
}
```

Example:

```
class IfElseExample {
    public static void main(String[] args) {
        int number = 10;

        if (number > 0) {
            System.out.println("The number is positive.");
        } else {
            System.out.println("The number is not positive.");
        }
    }
}
```

Output:

```
The number is positive.
```

In this example:

- If the condition `number > 0` is true, the program prints "The number is positive."
- If the condition is false, it prints "The number is not positive."

Nested if-else

You can also nest `if-else` statements to check multiple conditions.

Example:

```
class NestedIfElseExample {
    public static void main(String[] args) {
        int number = 0;

        if (number > 0) {
            System.out.println("The number is positive.");
        } else if (number < 0) {
            System.out.println("The number is negative.");
        } else {
            System.out.println("The number is zero.");
        }
    }
}
```

Output:

```
The number is zero.
```

switch Statement

The `switch` statement is another way to handle conditional logic, especially when you have multiple possible values for a variable.

Syntax:

```
switch (expression) {
    case value1:
        // code to be executed if expression == value1
        break;
    case value2:
```

```
        // code to be executed if expression == value2
        break;
    // more cases...
    default:
        // code to be executed if none of the above cases are true
}
```

Example:

```
class SwitchExample {
    public static void main(String[] args) {
        int day = 3;
        String dayName;

        switch (day) {
            case 1:
                dayName = "Monday";
                break;
            case 2:
                dayName = "Tuesday";
                break;
            case 3:
                dayName = "Wednesday";
                break;
            case 4:
                dayName = "Thursday";
                break;
            case 5:
                dayName = "Friday";
                break;
            case 6:
                dayName = "Saturday";
                break;
            case 7:
                dayName = "Sunday";
                break;
            default:
                dayName = "Invalid day";
                break;
        }

        System.out.println("The day is: " + dayName);
    }
}
```

Output:

```
The day is: Wednesday
```

In this example:

- The `switch` statement checks the value of `day` and matches it with the appropriate case.
- The `break` statement prevents fall-through to subsequent cases.

Loops

What are Loops?

Loops are used to execute a block of code repeatedly until a certain condition is met. This is useful when you want to perform repetitive tasks.

for Loop

Syntax:

```
for (initialization; condition; update) {
    // code to be executed
}
```

Example:

```
class ForLoopExample {
    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Count: " + i);
        }
    }
}
```

Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

In this example:

- The loop starts with `i = 1` and runs as long as `i <= 5`.
- After each iteration, `i` is incremented by 1.

while Loop

Syntax:

```
while (condition) {
    // code to be executed
}
```

Example:

```
class WhileLoopExample {
    public static void main(String[] args) {
        int i = 1;

        while (i <= 5) {
            System.out.println("Count: " + i);
            i++;
        }
    }
}
```

Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

In this example:

- The loop runs as long as the condition `i <= 5` is true.
- The variable `i` is incremented within the loop.

do-while Loop

The `do-while` loop is similar to the `while` loop, but it guarantees that the loop body will execute at least once.

Syntax:

```
do {
    // code to be executed
} while (condition);
```

Example:

```java
class DoWhileLoopExample {
    public static void main(String[] args) {
        int i = 1;

        do {
            System.out.println("Count: " + i);
            i++;
        } while (i <= 5);
    }
}
```

Output:

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

In this example:

- The loop body is executed first, then the condition `i <= 5` is checked.
- The loop continues as long as the condition is true.

Practice Problems

1. Pattern Printing

Problem Statement:

Write a program to print the following pattern:

```
*
**
***
****
*****
```

Solution:

```java
class PatternPrinting {
    public static void main(String[] args) {
        int rows = 5;
```

```
        // Outer loop for each row
        for (int i = 1; i <= rows; i++) {
            // Inner loop to print stars
            for (int j = 1; j <= i; j++) {
                System.out.print("*");
            }
            // Move to the next line after each row
            System.out.println();
        }
    }
}
```

Output:

```
*
**
***
****
*****
```

Explanation:

- The outer loop runs from `1` to `5`, representing each row.
- The inner loop prints stars (`*`) equal to the current row number.

2. Simple Calculator

Problem Statement:

Write a program that takes two numbers and an operator as input and performs the corresponding arithmetic operation (addition, subtraction, multiplication, or division).

Solution:

```
import java.util.Scanner;

class SimpleCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking input from the user
        System.out.print("Enter the first number: ");
        double num1 = scanner.nextDouble();

        System.out.print("Enter the second number: ");
        double num2 = scanner.nextDouble();

        System.out.print("Enter the operator (+, -, *, /): ");
        char operator = scanner.next().charAt(0);
```

```
        double result;

        // Switch case to handle different operations
        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
            case '*':
                result = num1 * num2;
                break;
            case '/':
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    System.out.println("Division by zero is not allowed.");
                    return;
                }
                break;
            default:
                System.out.println("Invalid operator!");
                return;
        }

        // Displaying the result
        System.out.println("The result is: " + result);

        scanner.close();
    }
}
```

Output Example:

```
Enter the first number: 10
Enter the second number: 5
Enter the operator (+, -, *, /): *
The result is: 50.0
```

Explanation:

- The program first takes two numbers and an operator as input from the user.
- It then uses a `switch` statement to perform the corresponding arithmetic operation.
- The result is displayed based on the chosen operation.

Here are the practice questions with solutions that you can include in your `.md` file:

3. **Find the Maximum of Three Numbers**

Problem Statement:

Write a program that takes three integers as input and determines which one is the maximum.

Solution:

```
import java.util.Scanner;

class MaxOfThree {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking three integer inputs
        System.out.print("Enter the first number: ");
        int num1 = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int num2 = scanner.nextInt();
        System.out.print("Enter the third number: ");
        int num3 = scanner.nextInt();

        // Finding the maximum number
        int max = num1;

        if (num2 > max) {
            max = num2;
        }

        if (num3 > max) {
            max = num3;
        }

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

        scanner.close();
    }
}
```

Expected Output:

```
Enter the first number: 10
Enter the second number: 20
Enter the third number: 15
The maximum number is: 20
```

4. **Check if a Number is Prime**

Problem Statement:

Write a program that takes an integer as input and checks whether it is a prime number.

Solution:

```
import java.util.Scanner;

class PrimeCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking integer input
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        boolean isPrime = true;

        // Checking if the number is less than 2
        if (num < 2) {
            isPrime = false;
        } else {
            // Checking divisibility from 2 to num/2
            for (int i = 2; i <= num / 2; i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        // Printing the result
        if (isPrime) {
            System.out.println(num + " is a prime number.");
        } else {
            System.out.println(num + " is not a prime number.");
        }

        scanner.close();
    }
}
```

Expected Output:

```
Enter a number: 29
29 is a prime number.
```

5. **Sum of Digits of a Number**

Problem Statement:

Write a program that takes an integer as input and calculates the sum of its digits.

Solution:

```java
import java.util.Scanner;

class SumOfDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking integer input
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        int sum = 0;

        // Calculating the sum of digits
        while (num != 0) {
            sum += num % 10;
            num /= 10;
        }

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

        scanner.close();
    }
}
```

Expected Output:

```
Enter a number: 1234
The sum of digits is: 10
```

6. **Factorial of a Number**

Problem Statement:

Write a program that takes a positive integer as input and calculates its factorial.

Solution:

```java
import java.util.Scanner;

class Factorial {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking integer input
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        long factorial = 1;

        // Calculating the factorial
```

```
        for (int i = 1; i <= num; i++) {
            factorial *= i;
        }

        System.out.println("The factorial of " + num + " is: " + factorial);

        scanner.close();
    }
}
```

Expected Output:

```
Enter a number: 5
The factorial of 5 is: 120
```

7. **Reverse a Number**

Problem Statement:

Write a program that takes an integer as input and outputs its reverse.

Solution:

```
import java.util.Scanner;

class ReverseNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking integer input
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        int reverse = 0;

        // Reversing the number
        while (num != 0) {
            int digit = num % 10;
            reverse = reverse * 10 + digit;
            num /= 10;
        }

        System.out.println("The reverse of the number is: " + reverse);

        scanner.close();
    }
}
```

Expected Output:

```
Enter a number: 1234
The reverse of the number is: 4321
```

8. **Print a Fibonacci Series**

Problem Statement:

Write a program that prints the first `n` numbers of the Fibonacci series.

Solution:

```java
import java.util.Scanner;

class FibonacciSeries {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking integer input for number of terms
        System.out.print("Enter the number of terms: ");
        int n = scanner.nextInt();

        int firstTerm = 0, secondTerm = 1;

        System.out.print("Fibonacci Series: " + firstTerm + ", " + secondTerm);

        for (int i = 3; i <= n; i++) {
            int nextTerm = firstTerm + secondTerm;
            System.out.print(", " + nextTerm);
            firstTerm = secondTerm;
            secondTerm = nextTerm;
        }

        scanner.close();
    }
}
```

Expected Output:

```
Enter the number of terms: 5
Fibonacci Series: 0, 1, 1, 2, 3
```

9. **Number Guessing Game**

Problem Statement:

Write a program that generates a random number between 1 and 100, and asks the user to guess the number. The program should give hints whether the guess is too low, too high, or correct.

Solution:

```
import java.util.Scanner;

class NumberGuessingGame {
    public static void main(String[] args) {
        int randomNumber = (int) (Math.random() * 100) + 1;
        Scanner scanner = new Scanner(System.in);
        int guess = 0;

        System.out.println("Guess a number between 1 and 100:");

        while (guess != randomNumber) {
            System.out.print("Enter your guess: ");
            guess = scanner.nextInt();

            if (guess < randomNumber) {
                System.out.println("Too low. Try again.");
            } else if (guess > randomNumber) {
                System.out.println("Too high. Try again.");
            } else {
                System.out.println("Congratulations! You've guessed the correct
number.");
            }
        }

        scanner.close();
    }
}
```

Expected Output:

```
Guess a number between 1 and 100: 50
Too low. Try again.
Guess a number between 1 and 100: 75
Too high. Try again.
Guess a number between 1 and 100: 63
Congratulations! You've guessed the correct number.
```

10. **Palindrome Checker**

Problem Statement:

Write a program that takes a string as input and checks if it is a palindrome (reads the same forwards and backwards).

Solution:

```java
import java.util.Scanner;

class PalindromeChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking string input
        System.out.print("Enter a string: ");
        String str = scanner.nextLine();
        String originalStr = str.toLowerCase();
        String reversedStr = "";

        // Reversing the string
        for (int i = originalStr.length() - 1; i >= 0; i--) {
            reversedStr += originalStr.charAt(i);
        }

        // Checking if the string is a palindrome
        if (originalStr.equals(reversedStr)) {
            System.out.println("The string is a palindrome.");
        } else {
            System.out.println("The string is not a palindrome.");
        }

        scanner.close();
    }
}
```

Expected Output:

```
Enter a string: madam
The string is a palindrome.
```

These solutions, with comments and explanations, should help students better understand the concepts of control flow and how to apply them in Java programs.