Java Control Flow Statements

Control flow statements are crucial for guiding the execution of programs based on certain conditions. In Java, these include `if-else`, `else-if`, and `switch` statements. Let's dive into each of these with examples and explanations.

1. If-Else Statements

If-Else Statements allow the program to execute different blocks of code depending on whether a condition is true or false. It's one of the most commonly used control flow statements.

Basic Syntax of `if-else`

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

Example:

```
int number = 10;

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

- **Output**: "The number is positive."

---

2. `else if` Statements

The `else if` statement is used when you have multiple conditions to check. It allows you to check several conditions in sequence.

Syntax of `if-else if-else`

```
if (condition1) {
    // Code to be executed if condition1 is true
```

```
} else if (condition2) {
    // Code to be executed if condition2 is true
} else {
    // Code to be executed if none of the conditions is true
}
```

Example:

```
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."

Here, if the first condition (`number > 0`) fails, it checks the second condition (`number < 0`) and if both fail, it executes the `else` block.

---

3. Nested If-Else Statements

You can place an `if-else` statement inside another `if` or `else` statement. This is called **nesting**.

Syntax of Nested `if-else` :

```
if (condition1) {
    if (condition2) {
        // Code to be executed if both conditions are true
    } else {
        // Code to be executed if condition1 is true but condition2 is false
    }
} else {
    // Code to be executed if condition1 is false
}
```

Example:

```
int age = 20;

if (age >= 18) {
```

```
    if (age >= 21) {
        System.out.println("You are an adult and eligible for drinking.");
    } else {
        System.out.println("You are an adult but not eligible for drinking.");
    }
} else {
    System.out.println("You are a minor.");
}
```

- **Output**: "You are an adult but not eligible for drinking."

---

4. Switch Statement

The **Switch Statement** allows a variable to be compared against a list of values (cases) and executes the matching code block. It is a cleaner way of handling multiple conditions compared to multiple `if-else` statements.

Basic Syntax of `switch`

```
switch (variable) {
    case value1:
        // Code to be executed if variable == value1
        break;
    case value2:
        // Code to be executed if variable == value2
        break;
    // Additional cases if needed
    default:
        // Code to be executed if none of the cases match
}
```

Example:

```
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";
}

System.out.println(dayName);
```

- **Output**: "Wednesday"

Key Points

- **If-Else Statements**:
  - **Simple** `if` : Executes the code block if the condition is true.
  - `else` **Block**: Executes when the condition is false.
  - `else if` : Provides additional conditions if the first `if` condition fails.
  - **Nested** `if-else` : Used for complex conditions with multiple layers.
- **Switch Statements**:
  - `case` : Defines different conditions to compare with the variable.
  - `break` : Exits the switch block after a case is matched.
  - `default` : Executes if no case matches.

Example of Nested If-Else and Switch

Example combining `if-else` , `else if` , and `switch` :

```
int age = 19;
String eligibility;

if (age >= 18) {
```

```
    if (age >= 21) {
        eligibility = "You are eligible to vote, drink, and drive!";
    } else if (age >= 19) {
        eligibility = "You are eligible to vote and drink!";
    } else {
        eligibility = "You are eligible to vote!";
    }
} else {
    eligibility = "You are not eligible for voting or drinking.";
}

System.out.println(eligibility);
```

- **Output**: "You are eligible to vote and drink!"

Conclusion

- **If-Else** is used for simple conditional checks and can be nested for more complex logic.
- **Else-If** allows multiple conditions to be tested in a cleaner way than multiple `if-else` statements.
- **Switch** is more efficient when dealing with multiple possible values for a single variable and is more readable compared to multiple `if-else` statements.
