KODNEST ASSIGNMENT- 7

Submitted by

NAME : YADHUKRISHNA MK

EMAIL :yadhu8824@gmail.com

DIFFERENCE BETWEEN ELSE IF AND SWITCH CONDITIONAL CONTROL CONSTRUCT

SWITCH CONDITIONAL CONTROL CONSTRUCT

The Java *switch statement* executes one statement from multiple conditions.

ELSE IF LADDER

The if-else-if ladder statement executes one condition from multiple statements.

DIFFERENCES

1. **Syntax:**

- Switch case: It uses the '`switch`' keyword followed by an expression in parentheses. Cases are defined using the `case` keyword, and the block of code for each case is enclosed within curly braces.
- Else-if ladder: It uses multiple '`if`' statements followed by conditions. The conditions are checked one after the other, and the block of code associated with the first true condition is executed.

2.**Condition evaluation:**

- Switch case: The expression used in the switch statement is evaluated once, and the control jumps directly to the matching case. This means it is suitable for situations where you need to compare a single value against multiple constants.
- Else-if ladder: Each condition in the else-if ladder is evaluated one after the other until a true condition is found. This means all the conditions are checked sequentially, and the control passes through each condition, even if the earlier ones are true.

3.**Supported data types:**

- Switch case: It can only handle integral data types (e.g., int, char) and certain enumerated types.
- Else-if ladder: It can handle any expression that evaluates to a boolean value (true or false). This includes all data types that can be used in boolean expressions.

4.**Expression complexity:**

- Switch case: The expression inside the switch statement must result in a constant value, meaning it cannot be a complex expression or a range of values.
- Else-if ladder: The conditions in the else-if ladder can involve complex expressions, logical operations, and comparisons, allowing for more flexible conditions.

5.**Case matching:**

- Switch case: The case values must be constant and unique. The control will jump to the first matching case and execute the code within that case. If no match is found, an optional `default` case can be used.
- Else-if ladder: The conditions can be overlapping, meaning multiple conditions can be true for a given input, leading to multiple blocks of code being executed. The order of the conditions matters, and the first true condition is executed.

6.**Fall-through behavior:**

- Switch case: If a case block does not end with a `break` statement, the control will fall through to the next case and continue executing the code in that case and any subsequent ones until a `break` statement is encountered or the switch block ends.
- Else-if ladder: Each `if` statement in the ladder is independent of others, and there is no implicit fall-through behavior like in switch case.

7.**Readability and maintainability:**

- Switch case: It is more concise and readable when dealing with multiple constant values. It can be a good choice for simple, straightforward scenarios.
- Else-if ladder: For complex conditions or ranges of values, an else-if ladder may be more readable and maintainable as it allows expressing conditions more explicitly.

8. **Use cases**:

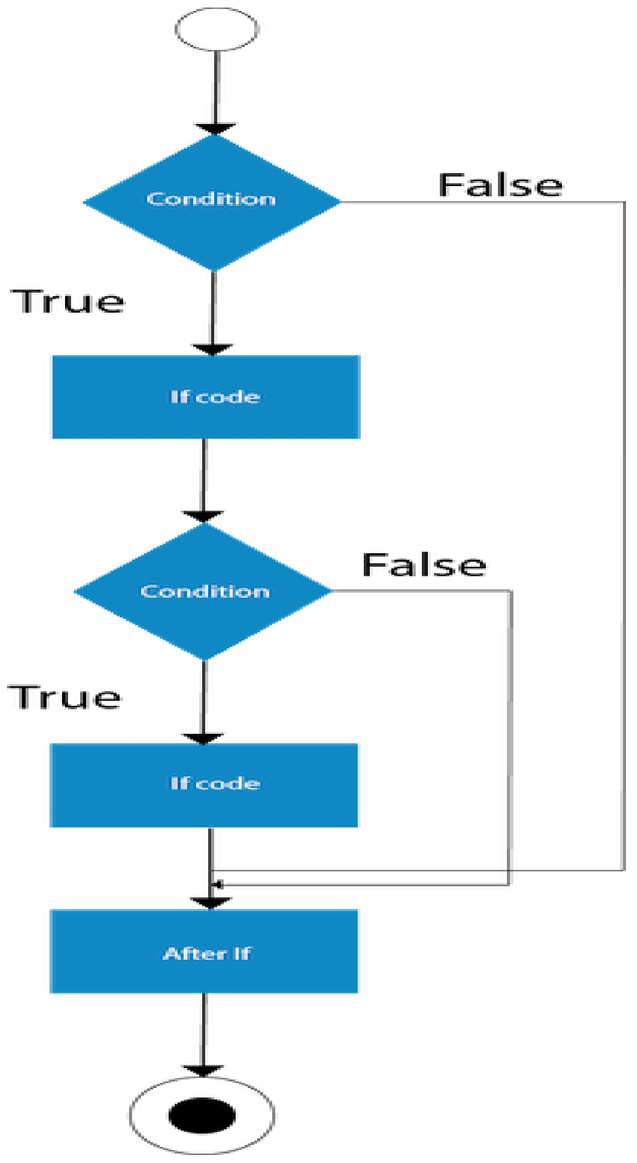
- Switch case: It is commonly used when you have a fixed set of values to compare against a single variable and when fall-through behavior is needed (e.g., menu options, handling weekdays).
- Else-if ladder: It is used when you have a series of different conditions, each leading to a separate block of code execution, or when you need to evaluate complex expressions.

Nested simple if

The nested if statement represents the *if block within another if block*. Here, the inner if block condition executes only when outer if block condition is true.

Syntax:

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



Example:

```
//Java Program to demonstrate the use of Nested If Statement.
public class JavaNestedIfExample {
public static void main(String[] args) {
    //Creating two variables for age and weight
    int age=20;
    int weight=80;
    //applying condition on age and weight
    if(age>=18){
        if(weight>50){
            System.out.println("You are eligible to donate blood");
        }
    }    }}
```

Nested else if

A nested if else statement is an if else statement inside another if or else block.

Syntax:

```
If(condition)
{
    If(condition)
       {
         //statements
        }
     else
        {
          // else block statement
        }
else
{
//statement
}
}
```

```
/Java Program to demonstrate the use of Nested If Statement.
public class JavaNestedIfExample2 {
public static void main(String[] args) {
    //Creating two variables for age and weight
    int age=25;
    int weight=48;
    //applying condition on age and weight
 if(age>=18){
        if(weight>50){
            System.out.println("You are eligible to donate blood");
        } else{
            System.out.println("You are not eligible to donate blood");

        }
    }
 else
{
      System.out.println("Age must be greater than 18");
    }
}  }
```