ELSE IF LADDER AND SWITCH CASE

DIFFERNCE BETWEEN THE ELSE IF LADDER ANS SWITCH CONTROL CONSTRUCT

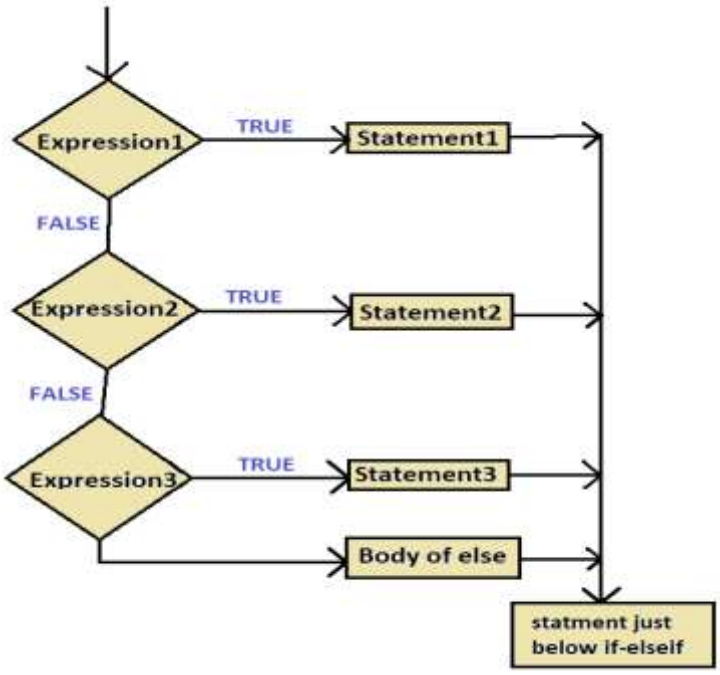
In Java, both the "else if" ladder and the "switch" control statement are used for making decisions and controlling the flow of the program based on different conditions. However, there are some key differences between the two:

1. Else If Ladder: Else-if ladder statement controls the statements to be executed on the basis of some conditions. Whenever statement is used, the compiler initially checks the condition whether it is true or false and if the condition is found to be true then the corresponding statements are executed. If the condition is false, it continues checking the next else if statement until the condition comes to be true or the control comes to the end of the else if ladder.

Syntax:

```
if( condition 1)
   statement 1;
else if (condition 2)
   statement 2;
.
.
.
else if(condition n)
    statement n;
else
   default statement;
```

Flowchart:



Description:

1. **if (condition1)**: The initial "if" statement checks the condition1. If the condition1 evaluates to true, the code block 1 associated with it is executed.
2. **else if (condition2)**: If the condition1 is false, the next "else if" statement checks condition2. If the condition2 evaluates to true, the code block 2 associated with it is executed.

3. **else if (condition3)**: If the condition2 is false, the subsequent "else if" statement checks condition3. If the condition3 evaluates to true, the code block 3 associated with it is executed.

4. **else**: If none of the conditions (condition1, condition2, condition3) are true, the optional "else" statement is executed. It provides a fallback block of code to be executed when none of the preceding conditions are satisfied.

Example:

```
import java.util.Scanner;
public class ElseIf {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the marks to get grade:");
		int marks= sc.nextInt();
		if(marks<50) {
			System.out.println("D Grade");
			System.out.println("Congratulations");
		}
		else if(marks>=50 && marks<60) {
			System.out.println("C Grade");
			System.out.println("Congratulations");
		}
		else if(marks>=60 && marks<70) {
			System.out.println("B Grade");
			System.out.println("Congratulations");
		}
		else if(marks>=70 && marks<80) {
			System.out.println("A Grade");
			System.out.println("Congratulations");
		}

		else {
			System.out.println("A+ Grade");
			System.out.println("Congratulations");
		}
	}
}
```

Output:

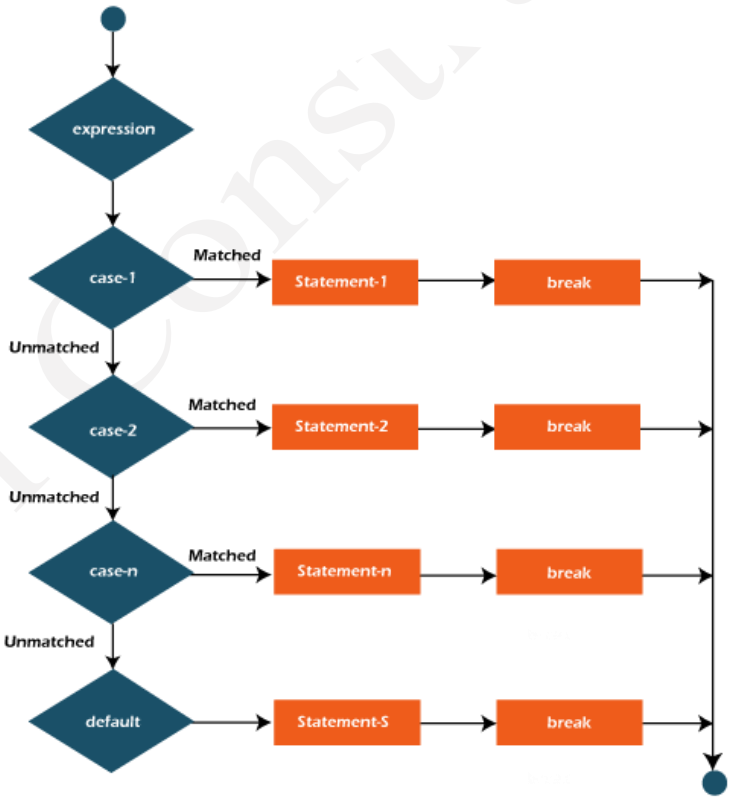
Enter the marks to get grade:
85
A+ Grade
Congratulations

2. Switch Control construct: The switch statement is similar to else-if ladder statement as it provides multiple conditions. It tests the value of variable or expression against a series of different cases or values. If a match is found then the block of code is executed otherwise the default case is executed.

Syntax:

```
switch (expression) {
   case value1:
      // Code block 1
      break;
   case value2:
      // Code block 2
      break;
   case value3:
      // Code block 3
      break;
   default:
      // Code block 4 (optional)
}
```

Flowchart:



Description:

1. **switch (expression)**: The switch statement starts with the keyword "switch" followed by the expression in parentheses. The expression is evaluated once, and its value is compared against the case values.
2. **case value1:**: Each "case" label represents a specific value that the expression will be compared against. If the expression matches a case value, the corresponding code block is executed.

3. // **Code block 1**: The code block associated with a particular case label is enclosed within curly braces. It contains the statements that will be executed if the corresponding case value matches the expression.
4. **break;**: After executing the code block of a matched case, the "break" statement is used to exit the switch statement. It prevents the execution of subsequent case blocks. Without the "break" statement, the control would fall through to the next case and execute its code block.
5. **default:**: The "default" label is optional and serves as the fallback option. If none of the case values match the expression, the code block associated with the "default" label is executed.

It's important to note that the expression in the switch statement must be of an integral type (byte, short, char, or int) or an enumeration type. Starting from Java 7, it also supports the use of strings.

Example:

```
import java.util.Scanner;
public class SwitchCase {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc= new Scanner(System.in);
		System.out.println("enter 1 to 7:");
		int n=sc.nextInt();
		switch(n) {
		case 1:
			System.out.println("Super Sunday");
			break;
		case 2:
			System.out.println("Lazy Monday");
			break;
		case 3:
			System.out.println("Energetic Tuesday");
			break;
		case 4:
			System.out.println("Angry Wednesday");
			break;
		case 5:
			System.out.println("Tired Thursday");
			break;
		case 6:
```

```
                System.out.println("Furious Friday");
                break;
            case 7:
                System.out.println("Happy Saturday");
                break;
            default:
                System.out.println("Hey!! please check the message and try again!!");
            }
        }
}
```

Output:

```
enter 1 to 7:
3
Energetic Tuesday
```

Difference between Else if and Switch case:

Features	Else if ladder	Switch Statement
Syntax	Series of "if" statements followed by "else if" statements and an optional "else" statement.	Starts with the "switch" keyword followed by the expression to be evaluated.
Expression Type	Conditions can be any boolean expressions, allowing for complex conditions and comparisons.	The expression must be of an integral type (byte, short, char, or int) or an enumeration type. Starting from Java 7, it also supports strings.
Multiple Conditions	Each condition is checked sequentially until a true condition is found. If multiple conditions are true, only the code block corresponding to the first true condition is executed.	Matches the expression value with a specific case value and executes the corresponding block of code.

Fall through Behavior	Once a true condition is found and its corresponding code block is executed, the program exits the entire ladder and continues with the rest of the program.	By default, after executing a matching case block, the control falls through to the next case block. This behavior can be explicitly controlled using the break statement.
Handling default case	Can have an optional "else" statement to handle the case where none of the conditions are true.	Has an optional "default" case that is executed when none of the case values match the expression.
Readability and maintainbality	Provides flexibility for complex conditions and comparisons. Suitable for scenarios where conditions involve ranges, logical operators, or other complex expressions.	Provides a concise and structured approach for handling multiple cases with simple equality checks. Enhances code readability when dealing with a large number of cases