Conditional Statements

1. If Statement

- Simple if
- if - else
- if - else ladder
- Nested if – else

2. Switch Statement

Simple if - The Java if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

```
if(condition)
{
  // Statements to execute if
  // condition is true
}
```

if – else - Use the else if statement to specify a new condition if the first condition is false.

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

if - else ladder – It is used to decide among multiple options. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

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

Nested if – else - Nested if-else refers to an if-else statement within an if-else statement. When we write an inner if-else condition within an outer if-else condition, then it is referred to as a nested if-else statement in java.

```
if(condition1)
{
          if(condition2)
        {
          // Stetement2.1
        } else {
          // Stetement2.2
        }

} else {
  // Stetement1
}
```

Switch Statement – It is a multi-way branch statement. In simple words, the Java switch statement executes one statement from multiple conditions. It is like an if-else-if ladder statement. It provides an easy way

to dispatch execution to different parts of code based on the value of the expression. Basically, the expression can be a byte, short, char, or int primitive data types. It basically tests the equality of variables against multiple values.

Some Important Rules for Switch Statements

1. **There can be any number of cases just imposing condition check but remember duplicate case/s values are not allowed.**
2. **The value for a case must be of the same data type as the variable in the switch.**
3. **The value for a case must be constant or literal. Variables are not allowed.**
4. **The break statement is used inside the switch to terminate a statement sequence.**
5. **The break statement is optional. If omitted, execution will continue on into the next case.**
6. **The default statement is optional and can appear anywhere inside the switch block. In case, if it is not at the end, then a break statement must be kept after the default statement to omit the execution of the next case statement.**

```
switch(expression)
{
  // case statements
  case value1 :
    // Statements
    break; // break is optional

  case value2 :
    // Statements
    break; // break is optional

  default :
    // Statements
}
```