JAVA Notes 4 – Conditional Statements

Java, like all other programming languages, is equipped with specific statements that allow us to check a condition and execute certain parts of code depending on whether the condition is true or false. Such statements are called conditional, and are a form of composite statement.

In Java, there are two forms of conditional statements:

- the if-else statement, to choose between two alternatives.
- the switch statement, to choose between multiple alternatives.

The if-else statement

Syntax:
```
if (condition) statement1;
else stetment2;
```

Example:
```
if (Mark >= 50) System.out.println("You PASSED!");
else System.out.println("You Failed!");
```

Compound statements

In the if we can use a single statement or more than one statement known as compound statement (block of statements). Compound statements are enclosed in curly brackets {}

Syntax:
```
if (condition) {
        Stament1;
        Statement2;
        …
}
```

Example:
```
if (Mark >= 50) {
        System.out.println("You PASSED!");
        pass = pass + 1;
}
```

Logical and Comparison Operators

Logical Operators	
Operator	**meaning**
&	AND
|	OR
^	XOR
||	Short circuit OR
&&	Short circuit AND
!	Unary NOT

Comparison Operators	
Operator	**meaning**
==	Equal to
>	Greater than
>=	Greater or Equal
<	Smaller than
<=	Smaller or Equal
!=	Not Equal to

Note: The operators in red are not included in the syllabus.

Nested ifs

This means you have an if in an if – the inner ifs are executed if the first if is true i.e. in 4 example below Distinction, Merit, Pass or Fail are only displayed if the mark is from 0 to 100.

Syntax:
```
if (condition1) {
        if (condition2) statement1;
        if (condition3) statement2;
        …
}
```

Example:
```
if (mark>=0 && mark<=100){
        if (mark>=90) System.out.println(“Distinction”);
        if (mark>=75 && mark<90) System.out.println(“Merit”);
        if (mark>=50 && mark<75) System.out.println(“Pass”);
        else if (mark<50)System.out.println("Fail");
}
else System.out.println(“Invalid mark”);
```

switch

This is a multiway branch statement i.e. easy way to send execution to different parts of the program. Used instead of using many if-else-if statements

Syntax:
```
switch (expression) {
        case value1: statement; break;
        case value2: statement; break;
        …
        case valueN: stamen; break;
        default: statement; break;
}
```

- When one of the cases is found true it will be executed and goes out of the switch.
- If none of the cases are found to be true then the default will be executed.
- If no default is present (it is optional) then no action is taken.
- The break statement is optional but if there's no break the program will continue to check the next cases.
- Sometimes it is useful to omit the break since you would need to continue checking the other cases – as seen in example 2 below.
- Can be used instead of if-else-if
- When you have many cases the if is better

Important features of the switch statement:

- The switch works only with: byte, char, int
- The switch can only be used for equality and NOT the other comparison operations as the if