Conditions

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

When using multiple conditions, we use logical AND (**&&**) and logical OR (**||**) operators. For Example:

If either of the condition is met the if block is executed below

```
if(month == 'December' || month == 'January') {
System.out.println("Winter Season");
}
```

If both of the condition are met then the if block is executed below

```
if(month == 'December' && day == '25') {
    System.out.println("Christmas Day!");
}
```

Java has the following conditional statements:

- **if** to specify a block of code to be executed, if a specified condition is true
- **else** to specify a block of code to be executed, if the same condition is false
- **else if** to specify a new condition to test, if the first condition is false
- **switch** to specify many alternative blocks of code to be executed

Syntax for if - else

```
if(condition){

//code if condition is true

}else{

//code if condition is false

}
```

Ternary Operator

We can also use ternary operator (? :) to perform the task of if...else statement. It is a shorthand way to check the condition.

condition ? true : false

5 > 6 ? true : false // this will give false
6 > 2 ? true : false // this will give true

Syntax for if - else if ladder

```
if(condition1){

//code to be executed if condition1 is true

}else if(condition2){

//code to be executed if condition2 is true

}

else if(condition3){

//code to be executed if condition3 is true

}

.....

else{

//code to be executed if all the conditions are false

}
```

Syntax for Nested if

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

Switch Case Statement

Another way to control the flow of the program is via a switch statement.
The switch statement is used when we have a number of options and in each case we execute different code. It acts similar to multiple if...else statements.

Syntax

```
switch(expression) {
    case a:
        //execute some code
        break;
    case b:
        //execute some other code
        break;
    default:
        //execute the default code
}
```

- First an expression is evaluated. The outcome of the expression is compared against each case. if the outcome of the expression matches any of the case conditions, the associated block of code is executed.
- The break keyword is used to exit the switch block. This is important because once a match is found, we don't want to continue to evaluate other case conditions.
- The default keyword is executed if no case match the value of the switch expression.
- Both break and default are optional, but is recommended for good coding practice.

PS: There is a newer syntax of switch case . Go ahead and try.