

# Conditionals and Control Flow

## else Statement

The `else` statement executes a block of code when the condition inside the `if` statement is `false`. The `else` statement is always the last condition.

```
boolean condition1 = false;

if (condition1){
    System.out.println("condition1 is
true");
}
else{
    System.out.println("condition1 is
not true");
}
// Prints: condition1 is not true
```

## else if Statements

`else - if` statements can be chained together to check multiple conditions. Once a condition is `true`, a code block will be executed and the conditional statement will be exited.

There can be multiple `else - if` statements in a single conditional statement.

```
int testScore = 76;
char grade;

if (testScore >= 90) {
    grade = 'A';
} else if (testScore >= 80) {
    grade = 'B';
} else if (testScore >= 70) {
    grade = 'C';
} else if (testScore >= 60) {
    grade = 'D';
} else {
    grade = 'F';
}

System.out.println("Grade: "
+ grade); // Prints: C
```

## if Statement

An `if` statement executes a block of code when a specified boolean expression is evaluated as `true`.

```
if (true) {
    System.out.println("This code
executes");
}
// Prints: This code executes

if (false) {
    System.out.println("This code
does not execute");
}
// There is no output for the above
statement
```

## Nested Conditional Statements

A nested conditional statement is a conditional statement nested inside of another conditional statement. The outer conditional statement is evaluated first; if the condition is `true`, then the nested conditional statement will be evaluated.

```
boolean studied = true;
boolean wellRested = true;

if (wellRested) {
    System.out.println("Best of luck
today!");
    if (studied) {
        System.out.println("You are
prepared for your exam!");
    } else {
        System.out.println("Study before
your exam!");
    }
}

// Prints: Best of luck today!
// Prints: You are prepared for your
exam!
```

## AND Operator

The AND logical operator is represented by `&&` .

This operator returns `true` if the `boolean` expressions on both sides of the operator are `true` ; otherwise, it returns `false` .

```
System.out.println(true && true); //
Prints: true
System.out.println(true && false); //
Prints: false
System.out.println(false && true); //
Prints: false
System.out.println(false && false);
// Prints: false
```

## The OR Operator

The logical OR operator is represented by `||` .

This operator will return `true` if at least one of the `boolean` expressions being compared has a `true` value; otherwise, it will return `false` .

```
System.out.println(true || true); //
Prints: true
System.out.println(true || false); //
Prints: true
System.out.println(false || true); //
Prints: true
System.out.println(false || false);
// Prints: false
```

## NOT Operator

The NOT logical operator is represented by `!` .

This operator negates the value of a `boolean` expression.

```
boolean a = true;
System.out.println(!a); // Prints:
false

System.out.println(!true) // Prints:
true
```

## Conditional Operators - Order of Evaluation

If an expression contains multiple conditional operators, the order of evaluation is as follows:  
Expressions in parentheses -> NOT -> AND -> OR.

```
boolean foo = true && (!false ||  
true); // true  
/*  
(!false || true) is evaluated first  
because it is contained within  
parentheses.
```

Then !false is evaluated as true  
because it uses the NOT operator.

Next, (true || true) is evaluation as  
true.

Finally, true && true is evaluated as  
true meaning foo is true. \*/