Java If ... Else

❮ Previous

Next ❯

Java Conditions and If Statements

Java supports the usual logical conditions from mathematics:

- 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`

You can use these conditions to perform different actions for different decisions.

Java has the following conditional statements:

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

The if Statement

Use the `if` statement to specify a block of Java code to be executed if a condition is `true`.

Syntax

```
if (condition) {
  block of code to be executed if the condition is true
}
```

Note that `if` is in lowercase letters. Uppercase letters (If or IF) will generate an error.

In the example below, we test two values to find out if 20 is greater than 18. If the condition is `true`, print some text:

Example

```
if (20 > 18) {
  System.out.println("20 is greater than 18");
}
```

Run example »

We can also test variables:

Example

```
int x = 20;
int y = 18;
if (x > y) {
  System.out.println("x is greater than y");
}
```

Run example »

Example explained

In the example above we use two variables, **x** and **y**, to test whether x is greater than y (using the `>` operator). As x is 20, and y is 18, and we know that 20 is greater than 18, we print to the screen that "x is greater than y".

The else Statement

Use the `else` statement to specify a block of code to be executed if the condition is `false`.

Syntax

```
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
}
```

Example

```
int time = 20;
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
// Outputs "Good evening."
```

Run example »

Example explained

In the example above, time (20) is greater than 18, so the condition is `false`, so we move on to the `else` condition and print to the screen "Good evening". If the time was less than 18, the program would print "Good day".

The else if Statement

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

Syntax

```
if (condition1) {
  block of code to be executed if condition1 is true
} else if (condition2) {
  block of code to be executed if the condition1 is false and condition2 is
true
} else {
  block of code to be executed if the condition1 is false and condition2 is
false
}
```

Example

```
int time = 22;
if (time < 10) {
  System.out.println("Good morning.");
} else if (time < 20) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
// Outputs "Good evening."
```

Run example »

Example explained

In the example above, time (22) is greater than 10, so the **first condition** is `false`. The next condition, in the `else if` statement, is also `false`, so we move on to the `else` condition since **condition1** and **condition2** is both `false` - and print to the screen "Good evening".

However, if the time was 14, our program would print "Good day."

Short Hand If...Else (Ternary Operator)

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

Syntax

```
variable = (condition) ? expressionTrue : expressionFalse;
```

Instead of writing:

Example

```java
int time = 20;
if (time < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
```

Run example »

You can simply write:

Example

```java
int time = 20;
String result = (time < 18) ? "Good day." : "Good evening.";
System.out.println(result);
```

Run example »

❮ Previous

Next ❯