Java Conditions and If Statements

You already know that 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

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");
}
```

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

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

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 < 18) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}

// Outputs "Good evening."
```

Short Hand If...Else

There is also a short-hand if else, which is known as the ternary operator because it consists of three operands.

It can be used to replace multiple lines of code with a single line, and is most often used to replace simple if else statements:

Syntax

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

Example

```
int time = 20;

String result = (time < 18) ? "Good day." : "Good evening.";

System.out.println(result);
```

Java Switch Statements

Instead of writing many `if..else` statements, you can use the `switch` statement.

The `switch` statement selects one of many code blocks to be executed:

Syntax

```
switch(expression) {

  case x:

    // code block

    break;

  case y:

    // code block

    break;

  default:

    // code block

}
```

This is how it works:

- The `switch` expression is evaluated once.
- The value of the expression is compared with the values of each `case`.
- If there is a match, the associated block of code is executed.
- The `break` and `default` keywords are optional, and will be described later in this chapter

The example below uses the weekday number to calculate the weekday name:

Example

```
int day = 4;

switch (day) {

  case 1:

    System.out.println("Monday");

    break;

  case 2:

    System.out.println("Tuesday");

    break;

  case 3:

    System.out.println("Wednesday");

    break;

  case 4:

    System.out.println("Thursday");

    break;

  case 5:

    System.out.println("Friday");
```

```
      break;
    case 6:
      System.out.println("Saturday");
      break;
    case 7:
      System.out.println("Sunday");
      break;
  }
// Outputs "Thursday" (day 4)
```

The break Keyword

When Java reaches a `break` keyword, it breaks out of the switch block.

This will stop the execution of more code and case testing inside the block.

When a match is found, and the job is done, it's time for a break. There is no need for more testing.

A break can save a lot of execution time because it "ignores" the execution of all the rest of the code in the switch block.

The default Keyword

The `default` keyword specifies some code to run if there is no case match:

Example

```
  int day = 4;
  switch (day) {
    case 6:
      System.out.println("Today is Saturday");
      break;
    case 7:
      System.out.println("Today is Sunday");
      break;
    default:
      System.out.println("Looking forward to the Weekend");
  }

// Outputs "Looking forward to the Weekend"
```