Trail: Learning the Java Language
Lesson: Language Basics
Section: Control Flow Statements

The if-then and if-then-else Statements

The `if-then` Statement

The `if-then` statement is the most basic of all the control flow statements. It tells your program to execute a certain section of code *only if* a particular test evaluates to `true`. For example, the `Bicycle` class could allow the brakes to decrease the bicycle's speed *only if* the bicycle is already in motion. One possible implementation of the `applyBrakes` method could be as follows:

```
void applyBrakes(){
     if (isMoving){  // the "if" clause: bicycle must be moving
          currentSpeed--; // the "then" clause: decrease current speed
     }
}
```

If this test evaluates to `false` (meaning that the bicycle is not in motion), control jumps to the end of the `if-then` statement.

In addition, the opening and closing braces are optional, provided that the "then" clause contains only one statement:

```
void applyBrakes(){
     if (isMoving) currentSpeed--; // same as above, but without braces
}
```

Deciding when to omit the braces is a matter of personal taste. Omitting them can make the code more brittle. If a second statement is later added to the "then" clause, a common mistake would be forgetting to add the newly required braces. The compiler cannot catch this sort of error; you'll just get the wrong results.

The `if-then-else` Statement

The `if-then-else` statement provides a secondary path of execution when an "if" clause evaluates to `false`. You could use an `if-then-else` statement in the `applyBrakes` method to take some action if the brakes are applied when the bicycle is not in motion. In this case, the action is to simply print an error message stating that the bicycle has already stopped.

```
void applyBrakes(){
     if (isMoving) {
          currentSpeed--;
     } else {
          System.err.println("The bicycle has already stopped!");
     }
}
```

The following program, `IfElseDemo`, assigns a grade based on the value of a test score: an A for a score of 90% or above, a B for a score of 80% or above, and so on.

```
class IfElseDemo {
```

```
    public static void main(String[] args) {

        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);
    }
}
```

The output from the program is:

```
Grade = C
```

You may have noticed that the value of `testscore` can satisfy more than one expression in the compound statement: `76 >= 70` and `76 >= 60`. However, once a condition is satisfied, the appropriate statements are executed `(grade = 'C';)` and the remaining conditions are not evaluated.

For Loop Extra Information

Multiple expressions in for loops

```
public class Main {
  public static void main(String[] args) {
    for (int i = 0, j = 0; i < 5; i++, j--)
      System.out.println("i = " + i + " j= " + j);
  }
}
/*
i = 0 j= 0
i = 1 j= -1
i = 2 j= -2
i = 3 j= -3
i = 4 j= -4
*/
```

Taken From: http://www.java2s.com/Tutorial/Java/0080__Statement-Control/0100__For-Loop.htm 1

For Loop Extra Information

Using the Floating-Point Values as the control value in a for loop

```
public class MainClass {

  public static void main(String[] arg) {
    for (double radius = 1.0; radius <= 2.0; radius += 0.2) {
      System.out.println("radius = " + radius + "area = " + Math.PI * radius * radius);
    }
  }

}
```

```
radius = 1.0area = 3.141592653589793
radius = 1.2area = 4.523093421169302
radius = 1.4area = 6.157521601035994
radius = 1.5999999999999999area = 8.04247719318987
radius = 1.7999999999999998area = 10.178760197630927
radius = 1.9999999999999998area = 12.566370614359169
```

Taken From: http://www.java2s.com/Tutorial/Java/0080__Statement-Control/0100__For-Loop.htm 2