Exercise 7

Conditional constructs

By the end of this exercise you will be able to

- Use Java's conditional constructs: `if` and `switch`.
- Use braces {} when they are needed.

Introduction

Conditional constructs are very useful. Indeed the earlier exercises have used them, but it is here that we will cover them in detail. You will be familiar with the `if` statement from other languages but here is how it is written in Java:

```
if ( boolean expression ) {
      statements...
}
else {
      statements...
}
```

The `else` part of the conditional is optional. One important difference from C is how the expression that is tested must be boolean-valued. Therefore to test if a variable is non-zero you must write `if (n != 0) ...` whereas in C you could write the shorter statement `if (n) ...`

Java allows several `if` statements to effectively merge into one combined statement that is a multi-way switch:

```
if (n == 0) {
     System.out.println("n is zero");
} else if (n == 1) {
     System.out.println("n is one");
} else if (n == 2) {
     System.out.println("n is two");
} else {
     System.out.println("n is unknown");
}
```

If you are testing an expression against a large number of constants, as is the case with the previous example, then the alternative `switch` statement should be used:

```
switch (n) {
case 0:
     System.out.println("n is zero");
     break;
case 1:
     System.out.println("n is one");
     break;
case 2:
     System.out.println("n is two");
     break;
default:
     System.out.println("n is unknown");
     break;
}
```

For more information on these `if` and `switch` constructs, consult one of the many excellent books in this subject area.

Questions

1. Fetch the file `Iffy.java` which contains a simple application with a `main` method which simply calls the `callSomeMethods` method which in turn exercises the other methods in the class. The `warnIfNegative`, `resetIfNegative` and `isInRange` methods are each intended to perform simple data validation tasks as described by the header comments. However, none of these three methods work as intended.

2. Add to the body of `callSomeMethods` sufficient calls to `warnIfNegative` to determine what the problem is. Summarise your findings in a comment.

3. Now modify `warnIfNegative` so that it behaves as intended.

4. Similarly, add to the body of `callSomeMethods` sufficient calls to `resetIfNegative` to determine what the problem is. Summarise your findings in a comment.

5. Now modify `resetIfNegative` so that it behaves as intended.

6. Now uncomment the method `isInRange`. The code was commented out because it causes compilation errors. Fix the code so it performs as intended and compiles correctly.

7. Add to the body of `callSomeMethods` sufficient calls to `isInRange` to illustrate how your correct version works.

8. Is an `if` statement really necessary in this example? Write a `isInRangeIfLess` method which gives the same results as `isInRange` but which does not use an `if` statement.

9. Finally, if you are feeling keen, extend `isInRange` so that it prints a warning message if the upper bound is less than or equal to the lower bound.