Exercise 5

Public versus private

By the end of this exercise you will be able to

- Understand when to use the `private` modifier and when to use the `public` modifier.
- Use `javadoc` to complete the definition of a class so that it works as intended.
- Understand how public and private relate to the concept of *encapsulation*.
- Understand how javadoc relates to the concept of encapsulation.

Introduction

Methods and properties can be labelled as `private` which restricts their use to within the class. The opposite idea to this is `public`, which allows methods and properties to be accessed from any class.

This idea of classes having private information that is not available to other classes is known as *encapsulation*. Programmers of C will have met the idea of having certain functions and variables private to the source file and this is the same idea but Java takes it further. In C, there is no way of making some members of a struct private and some public but in Java, the members of a class can each individually be labelled as `private` or `public`.

Privacy is a way of hiding information within a class in order to simplify the relationships between classes. It is a good policy to design classes with only a small number of public methods and properties, because then we are able to understand a program more easily by considering for each class separately:

1. What its purpose is: What job the class does internally, inside the body of each method.
2. How it relates to the other classes using the public methods and properties.

The `javadoc` program supplied with the Java system takes this information and publishes it into a web format for easy access. In this exercise, you should imagine that you are employed as a Java programmer and that your boss has given you the javadocs of a `SavingsAccount` class that she wants you to write for the purposes of representing customer's bank accounts.

Questions

1. Fetch the file `SavingsAccount.html` and load it into your web browser. The questions of this exercise will guide you through the process of writing the methods and properties described in this javadoc file.
2. With your text editor, open a new empty file called `SavingsAccount.java` which will contain all of the code that you will write for this exercise. Enter the outer shell of the class definition which consists of the following two lines:

```
public class SavingsAccount {

}
```

3. Javadoc normally provides you with a list of methods and properties, but in the case of this class there are only methods. This does not mean to say that your class will have no properties however, since javadoc only shows you how the class appears to other classes: that is to say, it only shows the public methods and properties.

   Since your class will be representing a savings account, you will most certainly need a property in which to store the customer's current account balance. You should now add such a property inside the braces of the class that you just created. The property will be called `balance`, should be of type `int` and should be labelled as `private`.

   In general when you are writing a class to meet the requirements that somebody has given to you, you are free to add as many private properties and methods as you like: All that matters as far as the users of the class is concerned is the public methods and properties.

4. Copy the following constructor which sets the initial balance to zero, into the `SavingsAccount` class:

   ```
   public SavingsAccount() {
      balance = 0;
   }
   ```

   Notice that the javadocs mention a second constructor which allows an initial value to be specified. Add this constructor to the class. This constructor differs from the first only in that it has one `int` parameter called `initialBalance` and in what it does with this parameter.

5. Read the javadocs for the `greet` method and then add this method to your class. The javadocs contain all the information you need to write this method, including what it should do, what is the return type and parameters of the method, and whether it should be private or public.

6. Read the javadocs for the `showBalance` method and then add this method to your class. Your method should simply return the current value of `balance`.

7. Here is the `deposit` method which you should copy into your class:

   ```
   public void deposit(int howMuch) {
      balance = balance + howMuch;
   }
   ```

   Write the `withdraw` method which is identical to this one except that the balance goes down by the value `howMuch`, rather than up.

8. Notice that the javadocs mention that the `deposit` and `withdraw` methods should check for a negative value of `howMuch`. Make some changes to these methods so that they print an error message if the amount given is negative.

9. Fetch the file `SavingsAccountManager.java` which contains some code to test out the `SavingsAccount` class that you have just written. Run this program to verify that that your `SavingsAccount` class works as intended.

10. **HARDER:** Complete the `transfer` method as stated in the javadocs.