Implementing Constructors

The classes we have designed so far do not have any ***user-defined*** constructors. Instances of a class with no constructor are always constructed with all instance variables set to zero (or null if they are object references). It is always a good idea to provide an explicit constructor.

Copy the contents of the Practical 6B folder from Blackboard to your computer. This folder contains the class and tester for the following questions.

1. Adapt the `BankAccount` class so that it has two constructors, one that sets the balance to 0 and one that sets balance to a specified amount. Adapt the tester program so that it constructs 2 `BankAccount` objects and tests the constructors.

2. A `BankAccount` should have an account number. What changes would we have to make to your class? Implement these changes

3. Now adapt your constructors so that the first sets the value of the `accountNumber` instance variable to a value passed in as a parameter and initialises balance to 0, while the second sets the `accountNumber` to a value passed in as a parameter and initialises the balance to another amount. It should not be possible to create a `BankAccount` object without specifying an `accountNumber` and it should not be possible to change the `accountNumber` once the object has been created. Test your class.

4. Open `Oblong.java`. Implement two constructors in the class,
   - One that will set instance variables to 0
   - One that will set instance variables to other values to be specified on construction.
   - Using the tester class provided test both constructors.

5. We want to prevent the creation of an `Oblong` with a negative width or height. We can do this within the constructor. If the values passed in are negative, we set them to 0. Implement and test the appropriate changes.

PrintQuota class

All students have a `PrintQuota` which allows them to print. The cost of printing a page is 5 cent, and `PrintQuota`s should have a balance which reduces every time a page is printed. The print quota can be topped up at any time. A student may also need to know the amount left in both euro and pages (i.e. the number of pages that they can print).

Remember that you should never declare an instance variable to hold something that you can calculate.

1. Draw a UML class diagram for your class.

2. Implement the `PrintQuota` class.

3. Write a simple program to test your class. Does the tester give expected results? Can it be improved?

4. Using the `PrintQuota` class, write a program that will allow the user to continually view balance in both pages and euro, top up, and print pages until the user decides to quit.

5. Adapt the `PrintQuota` class so that it has three constructors. One that sets the balance to 0 and the cost per page to a default value of €0.05. Another that sets the balance to 0 and the cost per page to a user determined amount. The third constructor should set both the balance and the cost per page to amounts determined by the user. Test your changes.

6. If it is possible to choose to print a page in either black and white or colour, what changes would you have to make to your class? (A colour page costs 12 cent per page).