Lab lecture exercises – 14 October 2016

1. 1l means that a bottle contains a minimum of 1 litre estimated. EU regulations (see `http://en.wikipedia.org/wiki/Estimated_sign`) specify the maximum negative tolerance for these estimates. For instance, an item marked 50ml must contain at least 45.5ml. The permitted tolerances are:

nominal quantity in ml	maximal negative error
5-50	9%
50-100	4.5 ml
100-200	4.5%
200-300	9 ml
300-500	3%
500-1000	15 ml
1000-10000	1.5%

   Write a class `Estimate` by expanding the method stub that can be found in the file `Estimate.java` in the `lab3.zip` handout on Canvas `https://canvas.bham.ac.uk/courses/21955/`.

   Test your program with the JUnit Test file provided:
   `junit EstimateTest`.

   Note in test driven development you write first the tests and then refine the program until all tests pass.

2. Use a `for` loop to write a static method `sumOdd` that sums up the first `n` odd numbers (odd numbers are $1, 3, 5, 7, 9, \ldots$). Write a main method in which you call the method with 100 and print out the value.

   Write JUnit tests for `sumOdd` and test the method.

   Extract the documentation.

3. Let a one-dimensional array of strings be given. Write a static method that concatenates all the strings, separated by the string `"***"`.

   Write JUnit tests and test your method.

   Extract the documentation.

4. Let a two-dimensional array of strings be given. Write a static method that concatenates all the strings (row by row), separating strings in the same row by `"***"` and separating rows by `"+++"`.

   Write JUnit tests and test your method.

   Extract the documentation.

5. Let a one-dimensional array `a` of `int` of size `n` be given. Write a static method `copyArray1` that creates a new one-dimensional array `b` of size `2*n` and copies the elements of `a` to `b` into their original position, that is, for `i<n` it should be `b[i]` is equal to `a[i]`.

6. Write a corresponding static method `copyArray2` for copying two-dimensional arrays.

7. In the `BankAccount` example we have considered the three field variables `accountNumber`, `accountName`, and `balance`.

   Add a fourth field variable `statement` of Type `ArrayList<Transaction>`. Extend the method `public void payin(Amount amount)` and `public void withdraw(Amount amount)` so that each time there is a `Transaction` (that is, the `payin` or the `withdraw` method is called), the `ArrayList` of `Transactions` is suitably extended. Write also a method `printStatement()` to print the statement out.

   Make use of the `Transaction` class provided.