Lab lecture exercises – 11 November 2016

1. Assume the `BankAccount` class from last Wednesday, follow the Canvas pages `https://canvas.bham.ac.uk/courses/21955` at Teaching `wk07.zip`.

   Create a sub-class `IslamicMortgageAccount` which has three additional fields:

   - `private int months`, which specifies the number of months over which the mortgage has to be paid back,
   - `private long payout`, the amount paid out to the customer initially (to buy the house), and
   - `private long fee`, the upfront fees involved.

   Note that for an Islamic mortgage account no interest is paid. All fees for the mortgage are to be included in the initial `fee`. The initial balance on the account is the negative of the sum of the `payout` and the `fee`.

   Implement for the class a constructor as well as getters and setters for the new field variables. Furthermore write two methods `public int initialPayment()` and `public int followUpPayment()` such that the payments are evenly distributed over `month -1` followup months (given as `followUpPayment()`), the remainder has to be paid in the first month as `public int initialPayment()`. Also write an appropriate `toString()` method which specifies in addition to the information given for any `BankAccount` information about the `payout`, the `fee`, the `months`, the `initialPayment` and the `followUpPayment`.

   It should not be possible to make any withdrawals or transfers of money from an `IslamicMortgageAccount`. How can we guarantee this?

   For instance, assume a mortgage (payout plus fee) of pounds 100000 over 240 months, then this would be paid back as 239 equal payments of pounds 418 plus pounds 98 in the first month. That is, `initialPayment()` should return 98 and `followUpPayment()` should return 418.

p.t.o.

2. In this exercise you should extend the above mentioned `BankAccount` class to create a `SavingsAccount` which has two additional fields:

   - `private double interestRate`, which specifies the annual interest rate that is paid for the money in the account.
   - `private ArrayList<Transaction> transactions`, an ArrayList that contains all transactions (initial balance, deposit, or withdrawal).

   In addition to the usual constructor, getters and setters, and `toString()` method, you should write a method `public int annualInterest()` which computes the total interest accumulated over the year. Note that the interest is paid pro rata for each of the 365 days of the year (if the year has 366 days, no interest is paid for the 366th day).

   For example, if the interest rate is 1% and for the first 200 days there are £1000 in the account, for this period an interest of $£1000 * 0.01 * \frac{200}{365}$, i.e., £5.47, accrues.

   If on the first day after that period another £1000 are paid in, for the remaining 165 days, the interest is computed on the £2000 that are in the account, that is, $£2000 * 0.01 * \frac{165}{365}$. I.e., £9.04 are accrued for this period. This gives a total annual interest of £5.47 + £9.04, which is rounded to £15.

   You have also to implement a `Transaction` class, where a single transaction should store four pieces of information:

   - the day of the year (1–365), when the transaction took place (there is no interest paid for the 366th day in a leap year),
   - the type of transaction (initial balance, deposit, or withdrawal),
   - the amount of the transaction, and
   - the balance after the transaction took place.

Note that single days should NOT be broken down further to compute e.g., an hourly interest. For computing the daily interest, the last balance of that day should be taken.