ESS 201: Programming II (Java)
Term 1 - 2019-20
Lab 1

This lab exercise has 3 parts. Part 1 is to help you get started and to test out your development environment. Part 2 is the actual assignment that needs to be submitted and will be graded. Part 3 is for you to play around with and understand memory management in Java.

1. Hello World

Create a file Hello.java and type in the following:

```
public class HelloWorld{
     public static void main(String[] args) {
          System.out.println("Greetings!");
     }
}
```

Using a command prompt, compile and run this program.
Check how the compiler reacts if you have syntax errors in your code (e.g. missing double quotes/semi-colons/brackets).
Do **not** submit this program to LMS or Domjudge!

2. Classes and Encapsulation

A savings bank account allows a customer to maintain a balance, and to deposit and withdraw money from it. The bank pays interest based on the balance in the account.

Define a class SavingsAccount that the bank class might use to manage this. It should contain the following methods:

1. getBalance: returns the current balance
2. deposit: adds the amount being deposited to the balance. This amount should be the argument to the method
3. withdraw: similarly, takes an argument which is the amount to be withdrawn. Returns the requested amount as return value if successful, and returns zero if not (e.g. when the balance is below the amount requested). Updates balance appropriately
4. addInterest: computes the interest for the account and adds this to the balance. For simplicity, we can assume that all deposit/withdraw transactions are done on the first of each month. So, addInterest (which is done on the 1st of the following month) can assume that the balance has been there for 1 month
5. getName: returns the name of the account owner

6. Constructor: takes in the name of the account owner, **annual** interest rate, and initial balance.

For simplicity, you can assume that all amounts are in round numbers (integers) and fractional parts can be ignored.

To test this, create a main method that does the following:

1. create an account with the following arguments (name, annual interest rate, initial balance) : “Newton”, 15, 10000
2. deposit 2000
3. withdraw 5000
4. withdraw 10000
5. add interest (assume this is done at the beginning of next month)
6. print out the account owner name and balance as:

```
Account owned by <name> has a balance of <balance>
```

Now design a Bank class. This maintains an array of SavingsAccount objects. (Assume there are not more than 10 accounts in the bank). It has the following methods (plus others you might need to implement the functionality):

1. addAccount: takes in the name, interest rate and initial balance. Creates a SavingsAccount with this information and adds it to the array of accounts it maintains
2. printAccounts: prints out the information about each account as below, one account per line:

```
Owner: <name> Balance: <balance>
```

The main of the Bank class reads standard input to create accounts and run transactions. When the input data is complete, it computes the interest for each account (here we assume that the deposit/withdraw transactions were done one month back) and then it prints out the information for each account using printAccounts.

The input data has multiple lines in the following format:
Each line starts with one of the following characters: N, D, W
If is starts with N, it implies it is a new account, and contains the name (string), interest rate (int) and initial balance (int) as the next 3 fields. Example:
N Newton 15 10000
If it starts with D, it implies a deposit to the most recently opened account, and contains the amount (int). Example:
D 2000
If it starts with W, it means money to be withdrawn from the most recently opened account, and contains the amount (int). Example:
W 1000

Sample Input
N Einstein 12 5000
D 2000
W 3000
N Planck 18 20000
W 5000
W 5000
W 5000
W 6000
W 5000

Expected output
Owner: Einstein Balance: 4040
Owner: Planck Balance: 0

Submit this to DomJudge and check if it runs correctly

3. Garbage Collection

Run the following piece of code:

```
public class Test1 {
     public static void main () {
          int N = 10;
          int M = 100000;
          for(int i =0; i< N; i++) {
               // create an int array of M elements
               int[] box = new int[M];
          }
     }
}
```

Are there large values of N (within the range of “int”) for which this program does not work?

If you modify the program as follows:

```
public class Test1 {
     public static void main () {
          int N = 10;
          int M = 100000;
          // creates an array containing int arrays
```

```
                int[][] boxes = new int[N][];
                for(int i =0; i< N; i++) {
                        int[] box = new int[M];
                        boxes[i] = box;
                }
        }
}
```

How large can you make N and still be able to run on your machine? Why is this different from the first version?

Can you re-configure Java on your machine to allow larger amounts of memory (heap)?
Do **not** submit this program to LMS or Domjudge!