**WRITE AN OBJECT ORIENTED APPLICATION**

La **Banque Fortis Internationale** vous embauche pour développer une application bancaire qui permettrait de gérer les transactions des comptes clients.

**Fortis International Bank** hires you in order to implement a bank application to manage the banking transactions.  
  
First of all, you have to define the features and the behavior of the System’s objects

In addition, an account has *an identification number*, a *date opened*, a *pin* (personal identification number) and a *balance*.

* An account could be checking or saving.  
  A saving account has an *annual interest rate* and *savings balance.*
* *A* checking account has a *minimum balance*

A customer has one or many accounts and other features (define them).

A customer has to do different type of transactions (check\_balance, withdraw and/or deposit)

1 - Define all the System’s classes and draw the UML diagram

2 - Create the overloaded constructors in order to build customers’ accounts

3 - Create set and get methods to access to the objects properties

4 - Build all t**he services** that manipulate an account, such as:

* validate pin,
* make deposit (to credit the account),
* make withdrawal (to debit the account),
* display the accounts states,
* calculate monthly interest(this interest must be added to *savings balance)*
* modify the interest rate (modifier le taux d'intérêt)

5 – Build the **business classes**:

* Build the account class,
* Build the inherited classes : checking account and saving account
* Build the utility class : Date
* Build the customer class
* Build the transaction class

6 – Implement the **Comparable** interface to:

* order the customer account by account number (in ascending order)lass
* avoid duplicate account

7 – Encapsulate the mega-data by adding the data collection class

* define a data collection to store accounts (TreeSet)
* define three services in the data class as following:

• add an account,  
• remove an account,

• search for an account,  
• display accounts

8 - Create the driven application tester (the entry point of the java virtual machine)

* The customers objects and the accounts objects must be created during compile-time
* The customers transactions must be done in the real time during execution-time as following :
* Open the session ( input account number and pin)
* display balance,
* debit an account,
* credit an account,
* come back to the menu
* Log off the session
* validate input data
* create a data collection to store the accounts objects
* save all transactions **to a binary file** (accounts.ser),to do it you must have **all objects being serialized**
* at the end of the day, we need to read data from binary file(accounts.ser) and display them to the screen
* handle the exceptions than can occur during the runtime execution
* Create a *test data set* to test your application and to ensure that your software functions correctly (save the prints screens of the *test data set* in a word document file named firstname-lastname.docx

**Part 1** (20%): From 1- 2- 3-4-5 and, Test Driven Application (Tester Application)

**Due-date: June 30th, 2016**