COMP1127 Project (25% of course work grade) – Semester 2 (2014/2015)

Due: Friday April 24, 2015 @ 11:55pm

---

1. Decide on a way to represent a *tagged* Abstract Data Type called Bank Account. The data stored on a Bank Account are:
    a) Account Number
    b) Account Owner
    c) Balance ***[1 mark]***

2. Define the following methods for your ADT:

	Name of function	**Description**
Constructors	makeAccount1	Accept as arguments: Account Number and Account Owner Return your representation of an Account ADT with balance set to 0. ***[1 mark]***
	makeAccount2	Accept as arguments: Account Number, Account Owner and starting balance. Return your representation of an Account ADT ***[1 mark]***
Selectors	getAccNumber	Returns the account number ***[1 mark]***
	getAccountName	Returns the name on the account ***[1 mark]***
	getAccountBalance	Returns the current balance in the account ***[1 mark]***
Mutator	changeName	Accept as an argument the new name of the account holder. Change the name ONLY if it is different from what is previously stored. ***[1 mark]***
Predicates	isAccount	Returns *true* if object accepted is an Account ***[1 mark]***
	isBalanceEnough	Accepts two arguments: an Account object and an amount. Returns *true if* the amount is less than the balance in the account. ***[2 marks]***

3. Write functions to do the following:
    a) Withdraw an amount from a given account, if sufficient money is available. ***[2 marks]***
    b) Deposit an amount into a given account. ***[1 mark]***
    c) Transfer money from a valid account to another valid account. Transference should ONLY be done if sufficient money is in the account. ***[2 marks]***

4. This question focuses on collections of Bank Accounts:
    a) Choose to represent your collection of bank Account as a Stack or Queue.
    b) Write the relevant methods needed to (*correct naming of your methods is extremely important*):
        i. Add an Account to the collection ***[1 mark]***
        ii. Remove an Account from the collection ***[1 mark]***
        iii. Check what account is at the front or top ***[1 mark]***
        iv. Check if the collection is empty ***[1 mark]***
        v. Output all account information (**Formatting of output is important)** ***[1 mark]***

5. Write a main method to do the following:
    a) Accept account information and add your collection. (**NB: Use a loop and ensure you allow ONLY valid accounts to the collection)** ***[2 marks]***
    b) Charge a balance deduction fee of $500, for accounts with less than $2000 balance. ***[1 mark]***
    c) Charge a general 1% bank charges fee to ALL accounts. ***[1 mark]***

Appropriate commenting of your program. ***[1 mark]***

BONUS MARKS [2%]
Bonus marks will be awarded for the following:

- *Use of object oriented programming*
- *Properly structuring your code to facilitate easy reading and appropriate imports.*