COMPS203F Quiz 02 [10 marks], time allowed: 30 minutes

(1) Write your **name** and **student ID** on your answer paper
(2) This is an Open Book quiz but you should do it using **your OWN efforts** only.
(3) Write clearly the **part letter** [e.g. (b)] and then your answers. Marks will be **deducted** if not done so.

Question 1

In each of the following parts, write **additional lines** to do the required task. We assume they will be in correct positions.

(a) Write a class `Account` with a **private** attribute `availableAmount` (type `double`). Also write the getter and setter methods of the attribute. The names of getter and setter methods should be `getWord()` and `setWord()` respectively if the attribute name is `word`. Write a constructor with a single parameter `availableAmount` and a suitable statement to initialize the attribute using the value of the parameter. Also write another constructor which is empty and without any parameter.

(b) After the end of the class `Account` add a non-public class `SavingsAccount`, which is a subclass of `Account` with a `double`-type attribute `interest`. In the non-public class, there is no "`public`" before the word "`class`". Also write the getter/setter method of the attribute and the constructor `SavingsAccount(double availableAmount, double interest)` which suitably initializes the attributes.

(c) In `SavingsAccount`, override the `getAvailableAmount()` method so that the sum of `availableAmount` and `interest` is returned.

(d) In `Account`, add a **class** method `averageAmount(Account[] accountArray)` which returns the average available amount of the accounts in the parameter. You can assume each element of the array refers to a properly initialized `Account` or `SavingsAccount` object.

(e) After the end of the class `SavingsAccount`, add another non-public class `FrozenAccount`, which is a subclass of `Account` without any attribute. Override the `getAvailableAmount()` method to return zero (ignoring the attribute `availableAmount` of its superclass). Notice the class method `averageAmount()` should return a correct value **without** any change after a new subclass is added.