Western University

CS 1037A

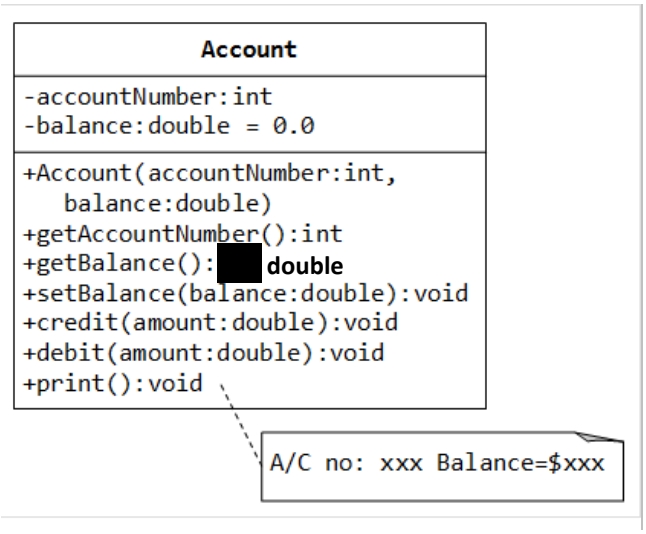
Assignment 2

## Instructions: Read carefully before you start. Also, read the attached file "Check_List_Assignment2"

This assignment should be done individually. The deadline is Friday November 7, 2014 at 11:55 pm. Please refer to the course outline (**Assignments Submissions and Late Assignments Policy Section)** for the late penalty. Each student must submit a zip file (zip and NOT rar) that contains the main folder of his/her assignments. The name of the main folder must be the student username. The main folder should contain two folders only; question1 and question2 (all lower case). The question1 folder should contain the source files (.cpp and .h) that are related to question1 (do not submit exe, solution or project files). The question2 folder should contain the source files (.cpp and .h) that are related to question2 (do not submit exe, solution or project files). The Readme file should be very simple. If you used visual studio for example, the Readme file should contain one or two lines like "project files were successfully compiled using visual studio xxx (say what year / version)". All files should work under Visual Studio C++. It is very important that you comment your code thoroughly. Marks will be deducted if the code is not commented properly.

Note: Read the whole questions before you start programming.

**Question 1:** Consider the following UML class notation for the class ***Account*** (25 points):




The description of the class Account is as follows:

1. ***accountNumber*** represents the account number of a bank customer
2. ***balance*** represents the balance of an account
3. Constructor that takes two arguments; ***accountNumber*** and ***balance***

4. Two accessors (getters) to return the values of ***accountNumber*** and ***balance***
5. ***setBalance*** member function to set the ***balance*** of the account
6. ***credit*** member function to deposit a certain amount to the ***balance***
7. ***debit*** member function to withdraw a certain amount from the ***balance***. If the amount is greater than the ***balance***, the system should print the message "amount withdrawn exceeds the current ***balance***". In this case (amount > ***balance***), the original ***balance*** should remain the same
8. ***print*** member function which shall print "Account no: xxx Balance=xxx" (e.g., Account no: 991234 Balance=$88.88), with ***balance*** rounded to two decimal places.

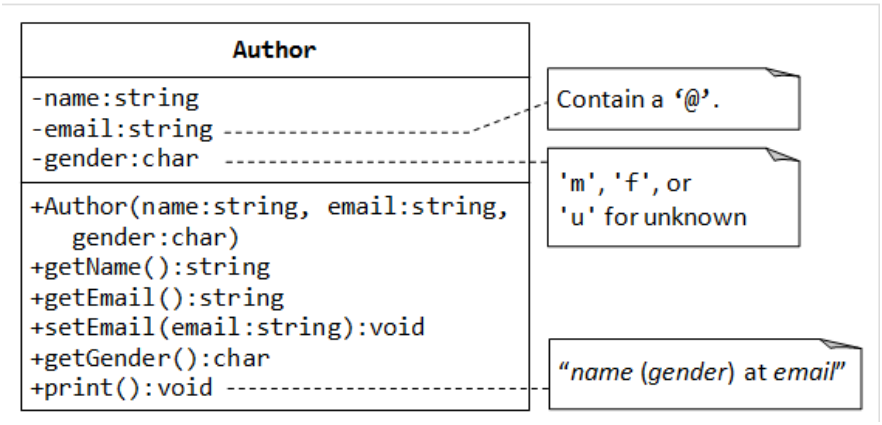
Write a program in C++ to implement the above requirements:

1. Your program should be composed into 3 files
    a. A header file for class definition
    b. An implementation file (the constructor should also be implemented here as well)
    c. The client program (also known as Test Driver)
2. All member functions that are not required to change any values should be declared as ***const***
3. Create an object Account1 (you should know in which file) with ***accountNumber*** 111 and ***balance*** $1,000
    a. Return the ***accountNumber*** of Account1
    b. Return the ***balance*** of Account1
    c. Set the ***balance*** of Account1 to $2,000
    d. Deposit $500 into Account1
    e. Withdraw $1,500 from Account1
    f. Print the details (Account number and balance) of Account1
4. Create an object Account2 with ***accountNumber*** 112 (***balance*** is default)
    a. Return the ***accountNumber*** of Account2
    b. Return the ***balance*** of Account2
    c. Set the ***balance*** of Account2 to $1,000
    d. Withdraw $1,200 from Account2
    e. Deposit $800 into Account2
    f. Print the details (Account number and balance) of Account2

Note:

1- You do not have to ask the user to input such values and store them in ***cin***. All the validation can be done when objects are created or by calling mutators
2- All Objects must be passed by value

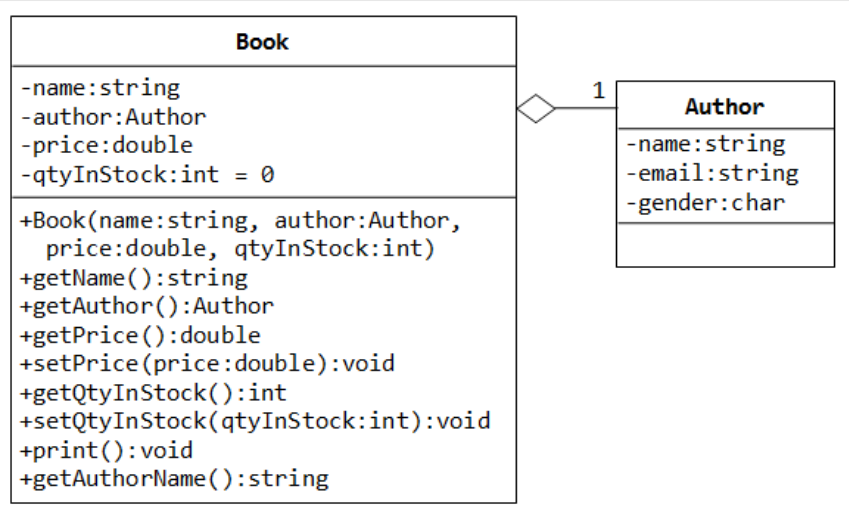
**Question 2: Consider the following classes (75 points):**




The description of the class Author is as follows:

1. ***name***: Name of the author
2. ***email***: Email of the author. It must contain the character '@'. Also the character '@' should not be at the beginning or at the end of the string ***email***.
3. ***gender***: The gender of the author. It should be either 'm' or 'f' or 'u' for unknown
4. ***constructor*** that takes 3 arguments. The constructor will initialize the ***name*** of the author, then will call the function ***setEmail***() to set and verify the ***email***, then to initialize and verify the ***gender***. If the ***gender*** is passed to the constructor as 'm' or 'f' then the constructor will set the value of ***gender*** with its corresponding value, otherwise the ***gender*** will be set as 'u'.
5. Three accessors to return the ***name***, ***email*** and ***gender*** of the author
6. ***setEmail***() member function to set the ***email*** of the author. This function should also validate the ***email*** to make sure it contains the '@' symbol. Moreover, the '@' symbol should not be at the beginning nor at the end of the ***email***. Hint: you can use the function ***string::find*** to locate its index in the string. If the ***email*** entered is invalid, print a message "invalid email" and set the ***email*** to empty string.
7. A ***print***() member function that prints "*name* (*gender*) at *email*", e.g., "Jen Smith (f) at jsmith@email.com"

Consider the class Book and its relationship (aggregation) with Author. Assume that a book is written by one and only one author.




The description of the book is as follows:

1. ***name***: Name of the book
2. ***author***: this is an object (instance of class Author)
3. ***price***: Price of the book. It must be positive
4. ***qtyInStock***: Quantity in stock. Must be positive or zero
5. constructor with four arguments
6. ***getName***(): returns the name of the book
7. ***getAuthor***(): returns the ***name*** of the author of the book. Note that author is an object (instance of class author)
8. ***getPrice***(): returns the price of the book
9. ***setPrice***(): sets and validates the price (>0)
10. ***getQtyInStock***(): returns the quantity in stock
11. ***setQtyInStock***(): sets and validates the quantity (>= 0). If the quantity is negative, print "quantity cannot be negative" then set the quantity to zero
12. ***print***(): prints "'*book-name*' by *author-name* (*gender*) @ *email*". Hint: you should call ***author***.***print***() here to get *author-name* (*gender*) @ *email*
13. ***getAuthorName***(): returns the name of the author of this Book instance

Write a program in C++ to implement the above requirements:

1. The Author program should be composed into 2 files; the header file and the implementation file. Note, you do not need Author Test Driver (client program) here since we can create Author objects in class Book
2. The Book program should be composed into 3 files; the header file, the implementation file and the client program. Note, you have to ***include*** Author.h inside Book.h
3. In the Book client program:
    a. Create an Author object with valid ***email*** and invalid ***gender*** (e.g. 'j')
    b. Print Author's details (call the ***print***() function in Author's class)
    c. Validate the ***email*** (call the ***setEmail***() and enter an invalid ***email***; e.g. "@john.com"

d. Call the three accessors in the Author's class
e. Create a Book object. Note: check the constructor of the Book. The second argument is the Author's object that you just created
f. Validate the ***qtyInStock*** (call the function and pass a negative number)
g. Call all other functions (accessors, as well as, ***print***()).

**Important Notes for Question 2:**

1- All objects must be passed by reference, via the reference (`&`), to avoid the overhead due of creating clone copies (when passing by value). If you do not intend to modify the object inside the function (with side effect to the original copy), you should set it as ***const***.
   a. In the Author class, data members ***name*** and ***email*** are strings (objects)
   b. In the ***Book*** class, data members of ***string*** and ***Author*** are objects. ***Author*** class was defined earlier; Remember that ***string*** is a class provided in C++ header `<string>`, belonging to the namespace `std`.
   c. E.g. the prototype of the Author's constructor should look like **Author(const string & name, const string & email, char gender);** Same applies to ***setEmail***() function since it takes ***email*** as parameter.

2- In the constructor, the names of the arguments must be the same as the names of member variables
3- You do not have to ask the user to input such values and store them in ***cin***. All the validation can be done when objects are created or by calling mutators