Lab 6: User & Message classes

Goals

By the end of this lab you should:

- be able to identify some common errors when using pointers and dynamic memory allocation
- have a good head start on the BBoard assignment
- have fully unit-tested the User and Message classes

Collaboration policy:

For this lab, collaboration **IS ALLOWED**. You will be working on the first stages of this week's programming assignment. You should place these 2 classes (User and Message), AND ONLY THESE 2 CLASSES, in your public workspace. NEVER place any part the BBoard class in your public workspace

Copy these User and Message classes to your private workspace once you have completed the lab and want to begin working on the BBoard class.

Problem Definition

In this week's assignment you will create an extremely rudimentary bulletin board with a simple command-line interface (like in the old days!).

This will require three collaborating classes - BBoard, User and Message.

For this lab, you will just unit test the User and Message classes. DO NOT implement any part of the BBoard class for this lab. That means your main function should just declare and test Users and Messages, no BBoards.

File Naming Specifications

The files that you submit should be named **main.cpp**, **Message.h**, **Message.cpp**, **User.h**, and **User.cpp**.
The names are case-sensitive.

Class Specifications

User and Message Classes

The User and Message classes are two simple classes we will later use in the BBoard class.

Note that - by design - the User class will ***never*** "report" a User's password - you can only ask it to check if a given username/password matches. This is one place where encapsulation is the only way to go!
Changing the public interface will result in a 0 for the assignment.

User.h

```
//inclusion guards

//includes

class User
{
  private:
    string username;
    string password;

  public:
    //creates a user with empty name and password.
    User();

    // creates a user with given username and password.
    User(const string& uname, const string& pass);

    //returns the username
    string get_username() const;

    // returns true if the stored username/password matches with the
    // parameters. Otherwise returns false.
    // Note that, even though a User with empty name and password is
    // actually a valid User object (it is the default User), this
    // function must still return false if given a empty uname string.
    bool check(const string &uname, const string &pass) const;

    // sets a new password.
    // This function should only set the new password if the current
    // (old) password is passed in. Also, a default User cannot have its
    // password changed.
    // Returns true if password changed, false if not.

    bool set_password(const string &oldpass, const string &newpass);
```

```
};

//end inc guards
```

Message.h

```
//inclusion guards
//includes

class Message
{
  private:
    string author;
    string subject;
    string body;

  public:
    //default constructor
    Message();

    //Constructor with parameters
    Message(const string &athr,
            const string &sbjct,
            const string &body);

    //displays the message in the given format. See output specs.
    //Note: R'Sub will require an endl as the last output of this
    //function.
    void display() const;
};
//end inc guards
```

Main Function

In the main function, you should unit test each of the User and Message functions. That means you should create User and Message objects and then test each of the constructors as well as the User's get_username, set_password, and check functions and Message's display function.

Compiling and Testing
You can compile the entire program, generating an executable named a.out, with the following command line:

```
g++ -W -Wall -Werror -ansi -pedantic main.cpp Message.cpp User.cpp
```

But you should compile the classes separately, without generating an executable, with a command line

similar to:

```
g++ -W -Wall -Werror -ansi -pedantic -c Message.cpp
```

replacing Message.cpp with whichever class you are trying to compile of course. And then to compile the result

```
g++ -W -Wall -Werror -ansi -pedantic main.cpp Message.o User.o
```

What to Submit

For this assignment you will turn in the following files to R'Sub (galah.cs.ucr.edu):

- main.cpp (Your test harness.)
- User.h (with exact interface given in specs.)
- User.cpp
- Message.h (with exact interface given in specs.)
- Message.cpp
