**Containers**   
  
Workshop 4

In this workshop, you code a container class that holds notifications and a class that holds separate messages.

**LEARNING OUTCOMES**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* design and code a composition of objects
* read records from a file into a **string** object
* parse a string object into components based on simple rules
* reflect on the material learned in this workshop

**SPECIFICATIONS**

**Overview**

This workshop retrieves messages from a data file and collects them in a notification.  Each record in the data file contains a single message and ends with the same delimiting character.

The [test data file](https://scs.senecac.on.ca/~oop345/pages/workshops/w5_test.dat) contains:

|  |
| --- |
| **jim Workshop 5 is cool**  **harry @jim working on workshop 5 now**  **chris**  **dave what the ^#$%!**  **john @harry I'm done** |

The first message consists of a user name followed by a tweet.  The second message consists of a user name, a reply name prefaced by an **@**, and followed by a tweet.  Your solution ignores incomplete messages, such as the third message here.

**Solution**

Your complete solution to this workshop consists of three modules:

* **w5** - the application that collects and displays notifications
* **Notifications** - the module that holds and manages messages
* **Message** - the module that manages the retrieval and display of a single message

The classes for this workshop are defined in the **w5** namespace.

**Application**

The source file that uses your two classes is:

|  |
| --- |
| **// Workshop 5 - Containers**  **// w5.cpp**  **#include <iostream>**  **#include <fstream>**  **#include "Message.h"**  **#include "Notifications.h"**  **const char REC\_DELIMITER = '\n';**  **w5::Notifications collect(std::ifstream& in, char recDelim) {**  **w5::Notifications notifications;**  **do {**  **w5::Message message(in, recDelim);**  **if (!message.empty())**  **notifications += std::move(message);**  **} while(in);**  **return notifications;**  **}**  **int main(int argc, char\* argv[]) {**  **if (argc == 1) {**  **std::cerr << argv[0] << ": missing file operand\n";**  **return 1;**  **}**  **else if (argc != 2) {**  **std::cerr << argv[0] << ": too many arguments\n";**  **return 2;**  **}**  **std::ifstream file(argv[1]);**  **if (!file) {**  **std::cerr << argv[0] << "\n: Failed to open " << argv[1] << "\n";**  **return 3;**  **}**  **std::cout << "\nNotifications\n=============\n\n";**  **w5::Notifications notifications = collect(file, REC\_DELIMITER);**  **notifications.display(std::cout);**  **std::cout << "Press any key to continue ... ";**  **std::cin.get();**  **}** |

**Notifications Module**

A **Notifications** object manages access to a set of up to 10 **Message** objects.  The **Notifications** object collects copies of the **Message** objects, owns those copies and destroys them once they are no longer needed.

Your design of the **Notifications** class includes the following member functions:

* **Notifications()** - default constructor - empty
* **Notifications(const Notifications&)** - copy constructor
* **Notifications& operator=(const Notifications&)** - copy assignment operator
* **Notifications(Notifications&&)** - move constructor
* **Notifications&& operator=(Notifications&&)** - move assignment operator
* **~Notifications()** - destructor
* **void operator+=(const Message& msg)** - adds **msg** to the set
* **void display(std::ostream& os) const** - inserts the **Message** objects to the **os** output stream

Store the code for your **Notifications** module in two source files:

* **Notifications.h** - defines the **Notifications** class
* **Notifications.cpp** - implements the member functions for the **Notifications** class

**Message Module**

A **Message** object holds nothing or a single message that has been successfully retrieved from an **std::ifstream**file object.  An object that holds nothing is in a safe empty state.

Your design of the **Message** class includes the following member functions:

* **Message(std::ifstream& in, char c)** - constructor retrieves a record from the **in** file object, parses the record (as described above) and stores its components in the **Message** object.  **c** is the character that delimits each record
* **bool empty() const** - returns true if the object is in a safe empty state
* **void display(std::ostream&) const** - displays the **Message** objects within the container

Store the code for your **Message** module in two source files:

* **Message.h** - defines the **Message** class
* **Message.cpp** - implements the member functions for the class.

**Results**

The results generated by the application using your solution and the [test data file](https://scs.senecac.on.ca/~oop345/pages/workshops/w5_test.dat) are listed below:

|  |
| --- |
| **Notifications**  **=============**  **Message**  **User : jim**  **Tweet : Workshop 5 is cool**  **Message**  **User : harry**  **Reply : jim**  **Tweet : working on workshop 5 now**  **Message**  **User : dave**  **Tweet : what the ^#$%!**  **Message**  **User : john**  **Reply : harry**  **Tweet : I'm done**  **Press any key to continue ...** |

**SUBMISSION**

Follow your professor's submission instructions