Welcome
Notes
Workshops
Translation
Move Copy
Templates
Containers
Lambda Expressions
STL Containers
STL Algorithms
Smart Pointers
Multi-Threading
Assignments
Handouts
Practice
Resources

Part C - Class Relationships

Containers

Workshop 4

In this workshop, you code a container that holds notifications and a class that holds each message in a notifications object.

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 a simple set of 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 a pre-defined delimiting character.

The test data file 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:

- **`w4`** - the client application that collects and displays notifications
- **`Notifications`** - the module that holds and manages the messages
- **`Message`** - the module that manages the retrieval of a single message from a file and displays the message

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

Application

The source file that uses your two classes is:

```
// Workshop 4 - Containers
// w4.cpp
```

```
// Chris Szalwinski
// 2018-05-21

#include <iostream>
#include <fstream>
#include "Notifications.h"

char recordDelimiter{ '\n' };

int main(int argc, char* argv[]) {
    std::cout << "Command Line : ";
    for (int i = 0; i < argc; i++) {
        std::cout << argv[i] << ' ';
    }
    std::cout << std::endl;

    if (argc == 1) {
        std::cerr << "\n*** Insufficient number of arguments ***\n";
        std::cerr << "Usage: " << argv[0] << " fileName \n";
        return 1;
    }
    else if (argc != 2) {
        std::cerr << "\n*** Too many arguments ***\n";
        std::cerr << "Usage: " << argv[0] << " fileName \n";
        return 2;
    }
    std::ifstream input(argv[1]);
    if (!input) {
        std::cerr << "*** Failed to open file " << argv[1] << " successfully ***\n";
        return 3;
    }

    std::cout << "\nNotifications\n=============\n\n";
    sict::Notifications notifications = std::move(sict::Notifications(input));

    notifications.display(std::cout);
}
```

Notifications Module

A **Notifications** object can access a set of up to 10 **Message** objects. The **Notifications** object upon construction collects the addresses of **Message** objects from a file and destroys the objects once the **Notifications** object goes out of scope.

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

- **Notifications(std::ifstream&)** - constructor
- **Notifications(Notifications&&)** - move constructor
- **Notifications&& operator=(Notifications&&)** - move assignment operator
- **~Notifications()** - destructor
- **void display(std::ostream& os) const** - inserts the **Message** objects to the **os** output stream
- both the copy constructor and copy assingment operator are deleted

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 either nothing or a single message. The object retireves the message from an **std::ifstream** object. A **Message** 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
- **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 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.

Unless otherwise stated by your instructor, your submission should include the following components:

1. source code for your **`Notifications`** module
2. source code for your **`Messages`** modules
3. a text file named **`reflect.txt`** that includes:
    - identification of the Notifications class as a composition, aggregation or association
    - the corrected answers to the latest quiz that you received
    - a description of what you learned in completing this workshop