Lab 9 – 08227 Advanced Programming

This tutorial introduces the reader to more aspects of using classes in C++, and to Exceptions in C++.

1.0 Person Exercise 1

Download the **Lab9.zip** file and extract the files to **G:/08227/Lab9/**. Open the **Contacts** solution.

You are to write a simple program that will act as a Contact Address Book. Therefore, we will require a class that will hold the information about a Person, and a class that will act as the Address Book. Ignore the MyException class included in the provided code until exercise 5.

Add to your Contacts project a class called **Person** and a class called **AddressBook**; this is done in the same way as described in a previous lab tutorial.

2.0 Person Exercise 2

Open the **Person.h** header file and add the following declaration to the class:

```
private:
	string m_name;
	int m_age;
```

Also, remember to include the string class declaration into the Person.h preprocessor section:

```
#include <string>
using namespace std;
```

The **string** class represents a collection of text characters that can be manipulated, e.g. adding 2 string together, splitting a string, etc. Therefore, for simplicity we will use the string class to represent the name of our Person.

Due to the two private member data variables, we will require **public mutators** and **accessors** for both of them. Implement the mutators and accessors.

Place some checking code inside of the age mutator so that a negative age cannot be set.

Initialise the two data members to appropriate values in the constructor (i.e. = 0, or = “”). It is preferred to initialise data members in the initialisation list rather than inside of the constructor body.

Add a constructor to the Person class that accepts a name and an age and initializes the data members to the name and age given (remember the age checking; use the age mutator).

From the main.cpp file create an instance of the Person class (do not forget to include Person.h) with an appropriate name and age.

3.0 Person Exercise 3

Implement a method in the Person class that will **write** the **m_name** and **m_age** data members of the **Person** class to an **output stream**.

Call the write function from the main.cpp file and pass it the console output stream (**cout**). The m_name and m_age of the Person should now be displayed in the **console window**.

4.0 Person Exercise 4

Implement a method in the Person class that will **read** the **m_name** and **m_age** data members of the **Person** class from an **input stream**.

Create another instance of a Person and call the read function from the main.cpp file and pass it the console input stream (**cin**). Then call the write function of this instance to display the Person's m_name and m_age in the console window.

5.0 Person Exercise 5

Using the provided **MyException** class, throw a MyException exception in the **Person::setAge()** method if the age given is either below 0 or above 150. You should provide the message `"Invalid age in Person class"` and the line number `__LINE__` to MyException. Catch the MyException in the main.cpp file.

HINT: Make sure that you include the MyException.h in the Person.cpp file, and the main.cpp file.

6.0 Person Exercise 6 [Advanced]

Increase the functionality of the MyException class so that you can provide any or both of the following useful information about the line of code that has generated the exception, and return the information in the what() method.

Macro	Description
`__FUNCTION__`	This is the name of the method or function, in the form of a string constant
`__FILE__`	This is the name of the current input file, in the form of a string constant