**Data Structures (2028C) -- Spring 2020 – Lab 5**

***Topics covered: Templates and Exceptions***

*Lab due:* ***Sunday, Feb 23 at 11:55PM for Monday Section***

***Tuesday, Feb 25 at 11:55 PM for Wednesday Section***

**Objective:**

The objective of this Lab is to examine C++ templates and exception handling.

**Task 1:** Create a base class that will be used as the basis for the remainder of the lab.

1. Create a new project. You can name this whatever you like.
2. Design a class to abstractly model a shelf of games. You may want to read the entire assignment before starting this task.
   1. Your shelf needs an array to hold a number of board games (use the class defined in lab 4 as the data type for the array). The number of board games should be set in a global constant (use 10 for the time being).
   2. Your shelf needs a variable to hold the number of board games currently being stored in your shelf.
   3. Create a default constructor that initializes any values, so the shelf starts empty.
   4. Your shelf needs a function that will add a board game and another to remove a board game. The add function should receive a board game as an input parameter and the remove should return a board game. For the sake of simplicity, assume board games are fungible.
   5. Your shelf needs a function that returns the number of board games currently being housed in the shelf.
   6. Create the implementation code for the above functions as required.
3. Include in the submission a description of how you designed your add and remove methods. This description should not be code or pseudo code. Complete this before moving on to task 2.

**Task 2:** Use the shelf class.

1. Create a program that tests the shelf class. Prompt the user for which class method to invoke. This may look like:

|  |
| --- |
| Press 1 to add a board game to the shelf.  Press 2 remove a board game from the shelf.  Press 3 see how many board games are currently on the shelf.  Press 4 to quit. |

* 1. When adding a board game to the shelf, dynamically create (using the new operator) the board game. You may want to set attribute values in the board game class.
  2. When removing a board game you may want to display the attribute values set when the board game was created.

1. Use your test program to test all member functions and ensure the class is working correctly. At this time, only test conditions that should not throw an error. You don’t need to test conditions such as adding a board game when the shelf is full or removing a board game when the shelf is empty as that will be part of the next task.
2. Include in the lab report a screen shot(s) board of the output of your tests.

**Task 3:** Throw and catch errors.

1. Modify the code in your shelf class to test if the shelf is full or empty. You will need to create the exception classes.
   1. If you are adding a board game to a full shelf, throw a FullShelf exception.
   2. If you are removing a board game from an empty shelf, throw an EmptyShelf exception.
2. Modify your main program so that it traps any errors and displays an error message to the user then continues operations normally.
3. Include in the lab report a screen shot(s) of the output of a test. Include a discussion of the advantages of trapping an error in the class versus the calling function.

**Task 4:** Templates.

1. Create a new project
2. Copy the code from your shelf class into a new class called EntertainmentCollection.
3. Modify your class to use a template rather than board game as the item being stored in the array.
4. Copy and modify the code from task 2 to implement the EntertainmentCollection using the video game class created in lab 4.
5. Include in the lab report a screen shot(s) of the output of a test. Include a discussion of the advantages of using a template over specific types.

**Lab Submission:**

1. Write a lab report including the following information:
   1. A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.
   2. The sections from each task indicated to be included in the lab report.
2. Include all source code from all tasks, input and output files (if any), and any special instructions to compile and run those programs.
3. Mention the contribution of each group member explicitly.
4. Package all files in a single zip folder and upload the file on canvas.
5. TA: [muppalsa@mail.uc.edu](mailto:muppalsa@mail.uc.edu)

**Lab Grading:**

1. 20% - Lab attendance
2. 20% - Task 1 has been correctly implemented and meets all requirements.
3. 10% - Task 2 has been correctly implemented and meets all requirements.
4. 15% - Task 3 has been correctly implemented and meets all requirements.
5. 15% - Task 4 has been correctly implemented and meets all requirements.
6. 20% - Lab report contains all required information and is well written.

If program fails to compile, 0% will be given for that Task.