**1Data Structures (2028C) -- Spring 2017 – Lab 5**

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

*Lab due:* ***Sunday, Feb 18 at 11:55PM for Monday Section***

***Tuesday, Feb 20 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.
2. Design a class to abstractly model a stable. You may want to read the entire assignment before starting this task.
   1. Your stable needs an array to hold a number of horses (use the class defined in lab 4 as the data type for the array). The number of horses should be set in a global constant (use 10 for the time being).
   2. Your stable needs a variable to hold the number of horses currently being cared for in your stable.
   3. Create a default constructor that initializes any values so the stable starts empty.
   4. Your stable needs a function that will add a horse and another to remove a horse. The add function should receive a horse as an input parameter and the remove should return a horse. For the sake of simplicity, assume horses are fungible.
   5. Your stable needs a function that returns the number of horses currently being housed in the stable.
   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 stable class.

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

|  |
| --- |
| Press 1 to add a horse to the stable.  Press 2 remove a horse from the stable.  Press 3 see how many horses are currently in the stable.  Press 4 to quit. |

* 1. When adding a horse to the stable, dynamically create (using the new operator) the horse. You may want to set attribute values in the horse class.
  2. When removing a horse you may want to display the attribute values set when the horse 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 horse when the stable is full or removing a horse when the stable is empty as that will be part of the next task.
2. Include in the lab report screen shots of the output of all your tests.

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

1. Modify the code in your stable class to test if the stable is full or empty. You will need to create the exception classes.
   1. If you are adding a horse to a full stable, throw a FullStable exception.
   2. If you are removing a horse from an empty stable, throw an EmptyStable 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 screen shots of the output of all your tests. Include a discussion of the advantages of exception handling over traditional error handling.

**Task 4:** Templates.

1. Create a new project
2. Copy the code from your stable class into a new class called AnimalCollection.
3. Modify your class to use a template rather than horse as the item being stored in the array.
4. Copy and modify the code from task 2 to implement the AnimalCollection using the fish class created in lab 4.
5. Include in the lab report screen shots of the output of all your tests. 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. The sections from each task indicated to be included in the lab report.
2. Include any special instructions to compile and run those programs.
3. In a group project, submissions should include what each group member has contributed.
4. Package all files in a single zip folder and upload the file to canopy website within the specified due date.

**Lab Grading:**

1. 10% - Lab attendance
2. 20% - Task 1 has been correctly implemented and meets all requirements.
3. 20% - Task 2 has been correctly implemented and meets all requirements.
4. 20% - 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.