# CSC 232: Data Structures and Algorithms

**Lab 3: Getting to Know C++ Classes**

**Due: Immediately at the end of lab**

*We will do the first part of the lab together then you will work on the rest during the remaining lab time. Submit all the work you have completed before you leave the lab today regardless of whether or not you finished. Just do your best and make sure to ask for help if you are stuck for more than a few minutes!*

The purpose of this assignment is to:

* Learn how to create and use a class, its methods, and its objects
* Code methods for the Point class presented in lecture
* Enhance functionality of the class by adding methods and fields

**Requirements**

* Include the comment template at the top of your program. Use the coding style described in the *C++ Coding Style and Best Practices* document discussed in class and posted in BB.
* NOTE: you should have some method available for backing up your lab work – a USB drive, google drive, Microsoft onedrive, dropbox, whatever. Before you leave the lab, make a backup of your work somewhere external to your lab machine!!!

**Synchronous Steps** (all together now…)

1. Basic orientation to the structure of a C++ class
   1. Create a new, empty C++ project and add file lab3.cpp to it.
   2. Now let’s will walk through the code together.
   3. When I tell you to do so, run the code.
   4. Notice it is not complete – the only part of the point printed is the x-coordinate.
   5. Modify the code as follows
      1. Add a **getY()** and **setY()** method using **getX()** and **setX()** as examples
         1. Or **get\_y()** and **set\_y()** if you are using underscores
      2. Choose two real numbers and set the y-coordinate for points **a** and **b** to those numbers
      3. Add print statements to print the y-coordinate for points **a** and **b**
         1. Mimic (copy/paste/modify) the x-coordinate print statements
   6. Run the code and verify your y-coordinate getter and setter works
2. The copy constructor
   1. After the code for printing points **a** and **b**, create point **c** using the copy constructor
   2. Print point **c** and verify it has the same coordinate values as **b**
   3. Comment out the copy constructor declaration and method
   4. Re-run the code
   5. Why does it still work?
3. Overriding the << operator for printing a class
   1. Uncomment the **friend ostream& operator** declaration and definition
   2. Comment out all prior print statements and replace them with code to print using this overridden operator
   3. This is the standard technique for printing a class

**On Your Own**

1. Add code to create a vector of 3 points after the above code.
   1. Print all the points with a for loop (an iterator won’t work) using the overridden << operator and separate each point with three spaces
   2. Before continuing, make sure you can answer this question: How can you prove that the default constructor was used to create these points?
2. Use the translate() method to modify the *first* point in the vector to (3.0, 4.0) and print it.
3. Add the revert() method
   1. Add two fields **xOld** and **yOld** to your class (in the private section!)
   2. Modify your default constructor to initialize these new fields
   3. Modify your copy constructor to assign to these fields
   4. Add the declaration for **revert()** inside the class in the public section
   5. Add the definition outside the class; use translate() as a model
   6. Modify the *second* point in the Point vector with translate() and print it
   7. Revert it then print it again – verify it is the original value
4. If you are feeling adventurous, add a method that calculates the distance between two points
   1. You must **#include <math.h>** for the sqrt() function.