# Assignment 5 for CPTN230

**Name:** Multiple Class Operator and Constructor Overloading, Friends and Default Class Members

**References:**

* Text book
* The source files from previous assignments
* The documentation you created for previous assignments
* A5 Sample files
* Internet Help Sites
* Anything else except other people, this an individual assignment

### Assignment Overview:

There are two parts to this assignment.

The first part has you take an existing “nothing” class and application and fill it in. The goal is to have you understand the existence and purpose of default class member functions and when they need to be explicitly defined.

The second part has you expand on what was done in Assignment 4. The goal and purpose of this part of the assignment is to have you create multiple classes, allow them to become “friends” and demonstrate operator overloading. For this assignment you will use two classes. You will be asked to separate the application source file from the class’s source files. For each class you will be asked to create a separate header file from the class source (body) file. This is the exact same packing of files that was used in the previous assignments only now we are using more than one class.

The supplied example code uses a “foot” and “meter” class and overloads some of the arithmetic operators, assignment operator, casting operator and the constructor. You will need to define your own classes and which operators you plan to overload. Some commonly used categories from past assignments are “pound/kilo” or “Dollar/Euro” classes. The operators commonly chosen to overload are “=, +, -, \*, /”. Of course you may suggest your own classes and operators but clear it with me first.

* The application will create multiple instances (objects) from each class
* The application will demonstrate the use of all overloaded operators with same class operations and mixed class operations
* The application may not use any global variables or objects

You will create the design documentation first and then create and test the code. The design document format is the same as previous assignments.

The basic steps needed to complete this assignment are listed below. The “Assignment Description” takes each of these steps and expands on them as does the “Strenuously Recommended Approach” section.

1. Download any needed files from Blackboard.
2. Create the base documentation for the assignment.
3. In stages, create and test the code for the assignment.
4. Post the results to Blackboard via the “View/Complete Assignment” link.

### Assignment Description:

1. Download any needed files from Blackboard.

There are a bunch of them.

* This file
* CPTN230A5nothing\_bettle.pdf
* CPTN230A5application\_bettle.pdf
* CPTN230A5class\_foot\_bettle.pdf
* CPTN230A5classh\_foot\_bettle.pdf
* CPTN230A5class\_meter\_bettle.pdf
* CPTN230A5classh\_meter\_bettle.pdf
* CPTN230A5output\_bettle.pdf

The .pdf files are copies of demonstration code used in class.

You might want to print out a hard copy for ease of use but note that these .pdf files are larger the previous ones. Your choice.

1. Create the base documentation for the assignment.

The goal here is to get most of the document written so it will serve as a map for your application. You will back fill it as you complete actual code to add or fix details missed on the initial pass.

Do not start to write code until the initial documentation pass is completed. I will ask to see your documentation before the assignment is due to provide design comments and help.

1. In stages, create and test the code for the assignment.

Suggestions on how to do this are covered in detail in the “Strenuously Recommended Approach” section of this document. But I do want to mention that as you start coding there is a good chance you will have to make minor modifications to the design document. This is expected.

1. Post the results to Blackboard via the “View/Complete Assignment” link.

There are 7 files to be delivered for this assignment and yes zipping them into a single folder is allowed. See the next section for details.

### Assignment Deliverables:

There are seven files to be delivered with this assignment.

1. A Microsoft Word 2007 Document containing your write up and findings. The name of this file will be **CPTN230\_A5\_Design\_*lastname*.docx** where *lastname* is your actual last name. The format of the document is as follows. Do not deviate.

The Word document must contain the following clearly separated sections.

* Title page
  + Document name
  + Author
  + Creation Date
  + Course Number
* Table of Contents
* Introduction
* A one paragraph description of what the nothing class/application does
* Conclusions about the enhanced nothing class/application
* Captured Screen Output from nothing
* A one paragraph description of what the application does
* A description of the program variables and objects
  + Pseudo Code
    - A detailed description of all source code and header files including
    - The purpose of each non blank line
* Captured Screen Output from Application
* References documented using APA format

1. **CPTN230\_A5\_nothing\_*lastname*.cpp** – The enhanced nothing application source file
2. **CPTN230\_A5\_application\_*lastname*.cpp** – The application source file
3. **CPTN230\_A5\_class\_first\_*lastname*.h** – Your class header file
4. **CPTN230\_A5\_class\_first\_*lastname*.cpp** – Your class source file
5. **CPTN230\_A5\_class\_second\_*lastname*.h** – Your class header file
6. **CPTN230\_A5\_class\_second\_*lastname*.cpp** – Your class source file

The words “first” and “second” are to be replaced with the tag name used for each class.

### Strenuously Recommended Approach:

Even though the word “recommended” is in this section header, consider it as mandatory.

1. Understand the problem.

See previous assignments for details.

1. Identify and gather resources.

See previous assignments for details.

1. Decide what classes you want to implement and which operators you are going to overload.

Decide what classes you want to create but they need to be ones that lend themselves to mixed class operator overloading. Then choose what operators you need to overload. I do this by writing equations and expressions. Samples look like:

* obja = -obja
* obja = -objb
* objb = -obja
* obja = obja + obja
* obja = obja – obja
* obja = obja + objb
* obja = obja – objb
* objb = obja + objb
* objb= obja – objb
* obja == obja
* obja != obja
* obja < obja
* obja == objb
* obja != objb
* obja < objb

Decide what data members are needed to support your classes. From the data members and the list of operators you will overload, you can determine what member functions you need in which class.

Next you need to decide what your application will do with the instantiated objects from the classes. The application for this assignment is mainly a test driver to prove all the overloaded operators and constructors work correctly, so some careful planning here saves an incredible amount of time. For example, using generic forms of list…

* obj = -obj will likely be tested as a standalone equation testing the unary negation operator
* obj = obj + obj will likely be tested as a standalone equation testing the binary addition operator
* obj != obj will likely be tested as an expression used in an “if construct”

Create a table of all objects from both classes and their starting values. You will need this table to verify the constructors are working correctly. Further, this table will help in determining the results of the tests of the operators. Then create a table of tests the uses these objects. This second table should list the expected results of the test (pass, fail), the results of the test and any side effects of the tests. The table needs to include same class and mixed class tests. An example will be provided in class. These tables go into my design document and also as comments into the actual test driver application.

1. Create the base design document outline.

See previous assignments for details.

1. Add the document introduction.

See previous assignments for details.

1. Enhance the nothing class/application source file to explicitly create member functions for each default member function.

Note that the packaging of this single file breaks the proper and traditional grouping of code for a class. That is the source file contains both the class header and application. You will be adding a class body to this file when you explicitly define the default member functions. For this assignment only, this is allowed.

To prove the enhanced class is actually working, add cout statements to the member functions and application. This will produce output that can be captured later. This is also a valid learning and troubleshooting technique to use for other classes.

1. Document your enhanced nothing class conclusions.

See previous assignments for details.

1. Add the nothing class Captured Screen Output to the Document.

See previous assignments for details.

1. Add the application overview to the design document.

See previous assignments for details.

In addition, this is when you add the tables you created earlier.

1. Add the class header descriptions to the design document.

See previous assignments for details.

1. Add the class source code designs to the design document.

See previous assignments for details.

1. Add the application source code design to the design document.

See previous assignments for details.

1. Add the application variables and objects descriptions to the design document.

See previous assignments for details.

1. Create an “empty” application file and test it.

See previous assignments for details.

1. Create the class header files and add them to the project/solution.

See previous assignments for details.

1. Create “empty” class source files and add them to the project/solution.

See previous assignments for details.

1. In a repetitive procedure add one small feature to the application and test it.

See previous assignments for details.

1. Document your application conclusions.

See previous assignments for details.

1. Add the Captured Screen Output to the Document.

See previous assignments for details.

1. Build the references page from the in-line citations.

See previous assignments for details.

1. Proofread your Word document.

See previous assignments for details.

1. Rebuild the TOC.

See previous assignments for details.

1. Post the Word document to Blackboard via the “View/Complete Assignment” link.

See previous assignments for details.

If you wish you may zip them into one file for posting or submit multiple files. But regardless, remember you only get to use the “submit” button one time.

1. Down load and test the deliverable.

See previous assignments for details.