# Assignment 4 for CPTN230

**Name:** Single Class Operator and Constructor Overloading

**References:**

* Text book
* The source files from previous assignments
* The documentation you created for previous assignments
* A4 Sample files
* Internet Help Sites
* Anything else except other people, this an individual assignment

### Assignment Overview:

The goal and purpose of this assignment is to have you create a class and demonstrate operator overloading. Operator overloading is using the existing C++ operators that work with built in data types and defining how they will work with user defined (and instantiated) objects. For this assignment you will only use one class. You will be asked to separate the application source file from the class source file. For the 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.

The supplied example code uses a “date” class and overloads the comparison operators. You will need to define your own class and which operators you plan to overload. Some commonly used examples in the past assignments are a fraction class or a complex number class. The operators commonly chosen to overload are “-, +, -, \*, /”. Of course you may suggest your own class and operators but clear it with me first.

* The application will create multiple instances (objects) of your class
* The application will demonstrate the use of all overloaded operators
* 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 several files.

* This file
* CPTN230A4application\_bettle.pdf
* CPTN230A4class\_bettle.pdf
* CPTN230A4classh\_bettle.pdf
* CPTN230A4output\_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. 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 4 files to be delivered for this assignment. See the next section for details.

### Assignment Deliverables:

There are four 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\_A4\_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.

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 application does
* A description of the program variables and objects
  + Pseudo Code
    - A description of all source code and header files including
    - The purpose of each non blank line
* Conclusions about the application
* Captured Screen Output
* References documented using APA format

1. **CPTN230\_A4\_application\_*lastname*.cpp** – The application source file
2. **CPTN230\_A4\_class\_*lastname*.h** – Your class header file
3. **CPTN230\_A4\_class\_*lastname*.cpp** – Your class source file

### Strenuously Recommended Approach:

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

1. Understand the problem.

See Assignment 2 for details.

1. Identify and gather resources.

See Assignment 2 for details.

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

Decide what class you want to create but it needs to be one that lends itself to operator overloading. Then choose what operators you need to overload. I do this by writing equations and expressions. Samples look like:

* obj = -obj
* obj = obj + obj
* obj = obj – obj
* obj == obj
* obj != obj
* obj < obj

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

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

* 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 object starting values. Use 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. 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 Assignment 2 for details.

1. Add the document introduction.

See Assignment 2 for details.

1. Add the application overview to the design document.

See Assignment 2 for details.

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

Big Hint!! I actually create the base document and write my ideas and eventual tables all in one big step. It saves time.

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

See Assignment 2 for details.

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

See Assignment 2 for details.

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

See Assignment 2 for details.

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

See Assignment 2 for details.

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

See Assignment 2 for details.

1. Create the class header file and add it to the project/solution.

See Assignment 2 for details.

1. Create an “empty” class source file and add it to the project/solution.

See Assignment 2 for details.

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

See Assignment 2 for details.

1. Document your application conclusions.

See Assignment 2 for details.

1. Add the Captured Screen Output to the Document.

Do this with copy and paste from the command window to the actual document. Ensure you capture the entire program output and not just part of it. Clean up any formatting so the captured output looks identical to what the program actually produced.

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

See Assignment 2 for details.

1. Proofread your Word document.

See Assignment 2 for details.

1. Rebuild the TOC.

See Assignment 2 for details.

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

See Assignment 2 for details.

1. Down load and test the deliverable.

See Assignment 2 for details.