# Assignment 2 for CPTN230

**Name:** Create and Document a Class Part 1 - Pointers

**References:**

* Text book
* The source files from Assignment 1
* The documentation you created for Assignment 1
* A2 Sample files
* Internet Help Sites
* Anything else except other people, this an individual assignment

### Assignment Overview:

The goal of this assignment is to have you create a class and demonstrate various techniques of accessing that class. You will only need to define and create one class and an application to use it. 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 Assignment 1.

You get to define what your class is and does. Have some fun with it but between the class and application you will need to demonstrate the following techniques and comply with any restrictions.

* The application will create multiple instances (objects) of your class
* The application will create a pointer to this class object type
* The application will use a function to access the objects via a pointer
* The application may not use any global variables or objects
* The class will include the use of a pointer within the object
* The class will demonstrate the use of the “this” pointer

Further, in Assignment 1 you were asked to analyze and create the documentation for existing code. This is called reverse engineering. In this assignment, we will do the work in the more traditional sequence. You will create the documentation first and then create and test the code. The documentation for this assignment will contain sections that are identical to those you used in Assignment 1.

As always, the requirements for this assignment are exact and no deviation from the requirements or substitution of requirements is allowed. They would not be allowed in a software development organization so we will not allow them here.

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 the Blackboard via “View/Complete Assignment” link.

### Assignment Description:

1. Download any needed files from Blackboard.

There are several files.

* This file
* CPTN230A2application\_bettle.pdf
* CPTN230A2class\_bettle.pdf
* CPTN230A2classh\_bettle.pdf
* CPTN230A2output\_bettle.pdf

The .pdf files are copies of demonstration code that will be 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.

This is, as always, a Microsoft Word 2007 document. It will eventually contain sections used in the same way the document for Assignment 1 was used. See “Assignment Deliverables” for the document outline. Hopefully you kept a copy of the outline document from Assignment 1 to get it started.

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. If you write code first and then try to reverse engineer the documentation I will know. There subtle differences that creep into documents that are reversed engineered from code versus documentation that is created first. Most people don’t recognize these subtle differences. I do and points will be lost for reverse engineering a design document.

I may even 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 is covered in detail in the “Strenuously Recommended Approach” section of this document. 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 the Blackboard via “View/Complete Assignment” link.

There are 4 files to be delivered for this assignment. See the next section for details. Don’t forget the warning. It is described at the end of this document.

### Assignment Deliverables:

There are four files to be delivered with this assignment.

1. A Microsoft Word 2007 Document containing your design and findings. The name of this file will be **CPTN230\_A2\_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 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
* Conclusions about the application
* References documented using APA format

1. **CPTN230\_A2\_application\_*lastname*.cpp** – The application source file
2. **CPTN230\_A2\_class\_*lastname*.h** – Your class header file
3. **CPTN230\_A2\_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 1 for details.

1. Identify and gather resources.

See Assignment 1 for details.

1. Decide what you want your class and application to do for this assignment.

I usually start this on paper and sketch out ideas until I know what I want to create. Later on this will become a more formal process but for now sketched out ideas will work. This is a repetitive process until the idea is fleshed out. Note it won’t be complete but it will be a good start and help develop the big picture.

Decide what object you want to create. I personally start with an airplane object. Why, because it is fun. I like planes. Decide on a few data members that you will want to use for the object. Data members are the attributes of the object. For my plane, I’ll use speed, altitude, direction and owner. The data members you choose will later help determine what member functions you need.

Next, you need to decide what your application will do with the object(s). For this assignment, the application only really needs to test object creation, usage and deletion. Such an application is often referred to as a test driver application or driver for short. Remember your driver needs to create multiple objects, at least one pointer to the object and a function that uses a pointer to reference the objects.

Please note there is nothing to document here. This is a decision step. You cannot design or code an application until you know what it will do.

1. Start your documentation.

See Assignment 1 for details. This should take about 1 minute if you saved your generic outline from Assignment 1. If not, you get to start over. You do not need to document the SDE again.

1. Add the document introduction.

See Assignment 1 for details.

1. Add the application overview to the design document.

This should be done from the ideas you sketched out earlier. You should have a good idea what the application will do by now. Just add a paragraph or two describing what the application will do. You will probably refine your description as you add design details.

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

As part of the sketching out or decision step you know what your object is and what it will do. Document what data members you will need. This will also help you decide what member functions you need. For each data member you typically need a getter and setter. You will also need a constructor. Other member functions will be identified as needed.

Note that the design document should not list real code here. This should just be a table of data members.

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

Critical note here. This is not code. It is a description of code. You will need to document what each member function does, what inputs, outputs and local variables are needed. This can be done via pseudo code or flow charts. I still suggest pseudo code because it makes it easier to create comments in your real code later. If I see real code in the design document then I know you reverse engineered it. Each member function will need to be designed separately.

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

Repeat the process for each function, including main, in the application design using the exact same methods as in the previous step.

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

Describe all global variables and objects as well as any local variables and objects in the main function. For any global or local main variables or objects list their type, purpose and initial values. For any objects list the data members and their type, purpose and initial values when the object is created. See your class header file description above. However, remember there are no global variables or objects for this application.

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

Now you are ready to code. If you have been writing code before this step you are doing the assignment incorrectly.

Using the SDE of choice, create a project/solution and create an “empty” application source file. This “empty” file just exists with a simple main function and just enough support items to write a simple message to the screen. This allows the creation of a working SDE for the application and a piece of code that “works.”

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

Remember class header files normally only contain data members and member function prototypes. Add an include statement to the application file. Compile, link and test until the header file is complete. A strong suggestion is to compile, link and test often. Only add a line or two to the source or header files between compilation events. It is a quick way to catch and isolate syntax failures.

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

An “empty” class source file is one that has a simple constructor that initializes the data members. Compile, link and test.

Then add an object creation to the application. Compile, link and test.

Consider adding a temporary member function to the class that allows the object to display its data members and call it from the application. If such a member function was part of your plan then this is a competed item. If not, it is a good bit of test code in your application and class that you can remove later.

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

Now going through the list of features you are to add to the application (driver) add only enough code to the application to demonstrate that feature. As you go add only the member functions needed to the class source to implement that feature.

The reverse is also true for any technique you are required to demonstrate within the class.

Always add the minimal amount of code needed between compilation events. Compile, link and test often.

Resist the urge to go back and fix thing that are not broken.

You will likely have to make some minor updates to the design document as you create the real code.

Final note. Ensure all code includes comments based on the design. All source files must include small set of comments at the top including module identification, author and creation date.

1. Document your application conclusions.

Now that the application is finished, developers are often asked to perform a “post mortem” on their own code. For large applications, this is a formal team meeting. In our case, you get to do your own.

The issues to address are

* What went well
* Where did you encounter problems and how did you fix them
* What does not work
* If there was time where could you have used a better technique to implement a similar feature
* If there was to be a Version 2 of this applications what features would you add

Add these conclusions to your document in the appropriate section.

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

See Assignment 1 for details.

1. Proofread your Word document.

See Assignment 1 for details.

1. Rebuild the TOC.

See Assignment 1 for details.

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

Use the above link for assignment submissions using the same method used in Assignment 1 but with a warning.

Using this method of file submission you only get to click the “Submit” button once. Once clicked, the posted files (if any) are delivered to the grade book, the grade book indicates the assignment is done and you are locked out from further submissions for this assignment.

To submit multiple files, you must post all files to the assignment link and then, and only then, click the “Submit” button. This will be demonstrated in class.

As always time management is important. Even if you have not completed the assignment you must submit what you have on the day the assignment is due. The due date will be announced in class. Partial credit is given for partially completed assignments as long as the completed parts are correct. No late submissions will be accepted. In industry when your manager assigns a task and wants something to be done, they expect it on time. It is part of the “salary continuation plan.” I love that phrase.

1. Down load and test the deliverable.

It is your responsibility to make sure the file was uploaded and can be read. The only way I know to test this is to down load a temporary copy and make sure you can read it. Submission of unreadable files is not a valid reason for missing an assignment.