# Assignment 7 for CPTN230

**Name:** Simple Class Inheritance

As always you should read this document before doing anything else. Most of this you have already seen but read it anyway for specific requirements.

**References:**

* Text book
* The source files from previous assignments
* The documentation you created for previous assignments
* A7 Sample files
* Internet Help Sites
* Anything else except other people, this an individual assignment

### Assignment Overview:

This assignment has you creating a series of classes that implement an inheritance scheme. That is the classes will work together in an inheritance tree to allow you to create different but related sets of objects.

The number of classes you decide to implement is up to you but you must have at least four classes and three levels to the inheritance tree. You may use more if you wish, but additional classes will only add additional work and not require any additional techniques to complete the assignment.

Requirements for the classes are:

* They use public inheritance
* All data members are private
* All member functions are public
* Do not create virtual or pure virtual member functions
* They are not “friends”
* Do not use multiple inheritance

The application is a test program. That is, it is to test the implementation of the classes and the code. It should be treated that way.

Requirements for the application are:

* The application will create named instances (objects) from each class
* The application will demonstrate the use of all available member functions from all objects
* The application may not use any global variables or objects
* The application must not use pointers or references to objects
* The application does not need any functions other than main, unless you want to create them. Your choice

The supplied example only uses three objects but demonstrates all requirements for the assignment. You will need more classes for the assignment than I used in the example.

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
* CPTN230A7application\_bettle.pdf
* CPTN230A7class\_vehicle\_bettle.pdf
* CPTN230A7classh\_vehicle\_bettle.pdf
* CPTN230A7class\_plane\_bettle.pdf
* CPTN230A7classh\_plane\_bettle.pdf
* CPTN230A7class\_jet\_bettle.pdf
* CPTN230A7classh\_jet\_bettle.pdf
* CPTN230A7output\_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 may 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.

See the next section for details.

### Assignment Deliverables:

There are at least eleven files to be delivered with this assignment. Maybe more if create more than the four required classes.

The first is the design document. As noted it must be a Microsoft Word document named **A7\_*lastname*.docx** where ***lastname*** is your actual last name.

The Word document must contain the following clearly separated sections.

* Document name, author and creation date
* A one paragraph description of what the application does
* A description of the program variables and objects
* The pseudo code or flow chart for each function and member function used in your application and classes
* Conclusions about the application

The next group of files contain the code and header information for your application and classes. They are to be named

* CPTN230A7application\_lastname.cpp – The application source file
* CPTN230A7class\_first\_lastname.h – Your first class header file
* CPTN230A7class\_first\_lastname.cpp – Your first class source file
* CPTN230A7class\_second\_lastname.h – Your second class header file
* CPTN230A7class\_second\_lastname.cpp – Your second class source file
* CPTN230A7class\_third\_lastname.h – Your third class header file
* CPTN230A7class\_third\_lastname.cpp – Your third class source file
* CPTN230A7class\_fourth\_lastname.h – Your fourth class header file
* CPTN230A7class\_fourth\_lastname.cpp – Your fourth class source file
* Additional files for additional classes as needed

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

The last file is a capture of the output when actually running the final version of the application. This name of this file is

* CPTN230A7output\_lastname.docx – Yes, this a Microsoft Word document

Again substitute lastname with your actual last name.

### Strenuously Recommended Approach:

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

1. Understand the problem.

Before you begin any software development task you must fully understand what is required to complete the task. How else can you know when you are done?

Completely read this document from beginning to end before doing anything else. We will also cover it in class. The various sections of this document interrelate. Questions raised in one section are often answered in another section. This will give you the big picture of what is going on.

1. Identify and gather resources.

Under the “References” section I listed some of the references you might use for this and future assignments. In reality, you may use any resource or reference you want except other people. I am the only person you may come to for help. I also recommend you use documents and code created in previous assignments. They can help in reminding you on the correct syntax for code and content format for the documents. Liberal use of cut/copy and paste is encouraged.

1. Choose which classes you want to implement and “draw” a simple inheritance tree to describe their relationship.

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.

Decide which classes you want to create but they need to be ones that lend themselves to inheritance. Use the in class examples but you will be working from the generic to the more specific.

1. Add a list of data members and member functions to the inheritance tree diagram.

Decide what data members are needed to support your classes. Remember data members in the base classes are inherited by the derived classes so data members in base classes should be applicable to all derived classes.

Decide what member functions are needed in each of the classes. For each data member add a “getter” and “setter” function. That is a pair of functions that will set or retrieve the value of a data member. Error checking is not required for this assignment. Further add an explicit default constructor and destructor. The constructor and destructor should “announce” they are running when they are activated.

When this step is done you have an “annotated” inheritance tree that will go into the design document.

1. Decide what objects need to be created and what tests need to be performed.

Next you need to decide what objects need to be in the application. You will need to create at least one named object for each class in the inheritance tree. This means at least four objects of four different types. Determine from the annotated tree what member functions are available for each object. Remember a class can use any member functions it defines or inherits from its base class(es). Write this information down as a table

1. Create the base design document outline.

This should take about 1 minute to complete.

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. This is where you will put the diagram you created earlier.

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

1. Add all class header descriptions to the design document, but do them one at a time.

As part of the sketching out step you know what your classes are and what they will do. Document what data members you will need and what member functions you are creating.

It is worth noting that all class headers will look very similar except for some minor differences. Use copy and paste.

1. Add the source code design for each class 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. Each member function will need to be designed separately. Note that each class source code design will be very similar with a few minor changes. Lot’s of copy and paste helps.

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

Repeat the process for the application. Use your table of objects and tests to determine exactly what tests you need to list.

1. Add the 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 object list the data members and their type, purpose and initial values when the object is created. See your class header file descriptions above. But remember there are no global variables or objects for this application.

Further, since you should have carefully planned what objects you needed for the tests and their initial values, just copy and paste the information from the list of objects table.

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

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 files and add them 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 for each class header. Compile, link and test until the header files are complete. A strong suggestion is to compile, link and test often. Only add a line or two to source or header files between compilation events. It is a quick way to catch and isolate syntax failures.

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

Create an “empty” class source file for each class. 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 for each object type to the application. Compile, link and test.

Consider adding a temporary member function to each class that allows the object to display its data members and call them 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 classes that you can remove later. For this assignment you will actually want some “easy” method to display the current value of each object.

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

Now going through the list of tests, call each member function from each object that uses that member function. Add the member functions to the classes as you go. I suggest starting from the base classes and working down the tree to the derived classes. This will allow a lot of copy and paste to speed up code development and reduce the likely hood of syntax mistakes.

I suggest fully testing the same type object operator first in one class. Then repeat the process for the other class testing the same operator.

A very serious note. If you find yourself writing a lot of original code here you did this assignment wrong and did not learn the lesson from Assignments 4 and 5. This application won’t get big, but over 90% of the code should be created with copy and paste. Please get the point!

Always add the minimal amount of code needed between compilation events. Compile 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.

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. Proofread your Word document.

Do this just like you should do for all the documents you create and for all the same reasons.

1. Post the documents and code to Blackboard via the “View/Complete Assignment” link.

Use the above link for assignment submissions using the same method used in previous assignments.

You will be submitting at least eleven files. They are:

* A7\_lastname.docx – Your design document
* CPTN230A7application\_lastname.cpp – The application source file
* CPTN230A7class\_first\_lastname.h – Your first class header file
* CPTN230A7class\_first\_lastname.cpp – Your first class source file
* CPTN230A7class\_second\_lastname.h – Your second class header file
* CPTN230A7class\_second\_lastname.cpp – Your second class source file
* CPTN230A7class\_third\_lastname.h – Your third class header file
* CPTN230A7class\_third\_lastname.cpp – Your third class source file
* CPTN230A7class\_fourth\_lastname.h – Your fourth class header file
* CPTN230A7class\_fourth\_lastname.cpp – Your fourth class source file
* CPTN230A7output\_lastname.docx – Results of running the final application

I request you zip them into one file for posting and remember you only get to use the “submit” button one time.

The assignment will not be accepted late. If it is not complete on the day it is due submit what you have completed.