Project 01: OOP Project

Nicholas Jones

Capitol Technology University

Prepared for Professor Mehri

CS-200

**Objectives:**

1. Re-write the code for Lab 4 so that the class Person is an abstract class that contains a pure virtual function.

2. Use polymorphism to print the content of objects from the derived classes.

**Inheritance Hierarchy**

Base Class:

Person

Derived Class:

Staff

Derived Class:

Faculty

Derived Class:

Student

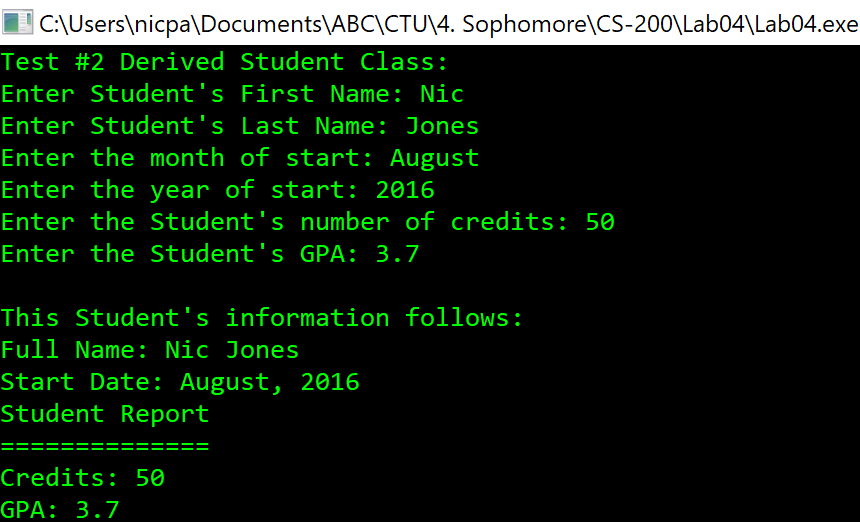
Derived Class:

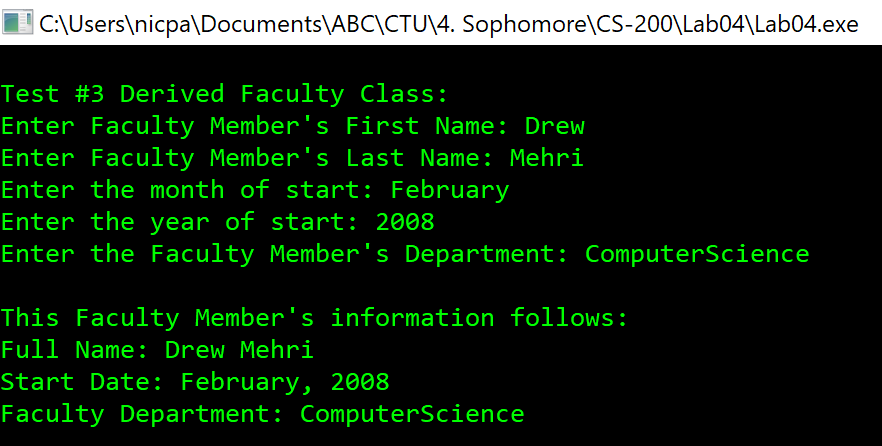
Administration

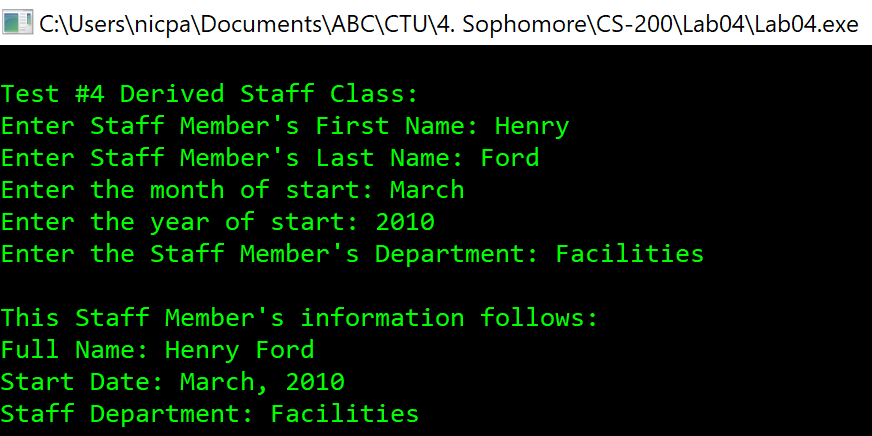
Derived Class:

IT

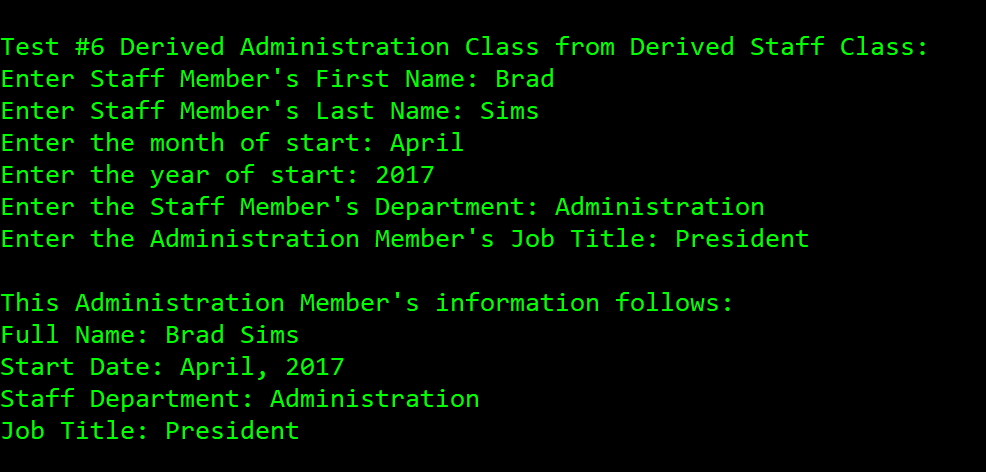
**Final Product**

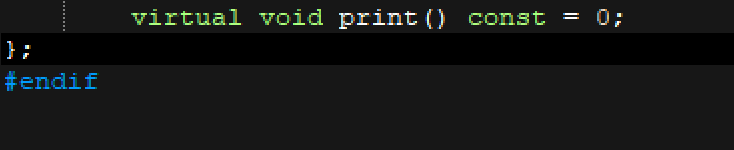
**Figure 1** *  
This image shows the Student class that is derived from Person and has the addition of GPA and Credits as well as what was inherited from person. Note: this information is printed using a pointer to the specific object type of Student from the print() pure virtual function.*

**Figure 2** *  
This image shows the Faculty class that is derived from Person and has the addition of department as well as what was inherited from person. Note: this information is printed using a pointer to the specific object type of Faculty from the print() pure virtual function.*

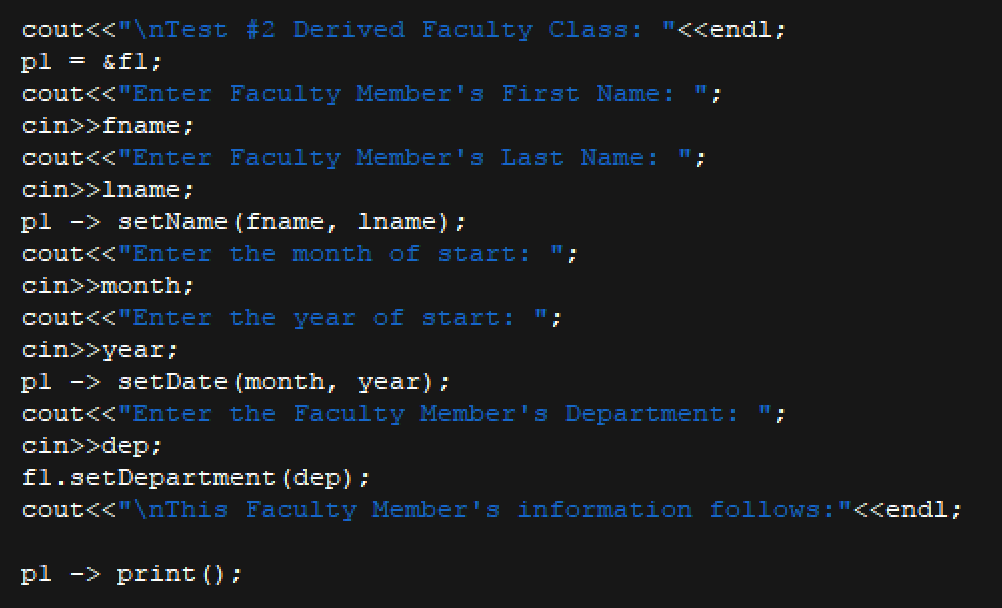
**Figure 3** *  
This image shows the Staff class that is derived from Person and has the addition of department as well as what was inherited from person. This class will serve as the base class to IT and Administration. Note: this information is printed using a pointer to the specific object type of Staff from the print() pure virtual function.*

**Figure 4** *  
This image shows the IT class that is derived from its base class of Staff. This class inherits all properties of Staff which also inherits the properties of Person, in addition to the IT members title and certification. Note: this information is printed using a pointer to the specific object type of IT from the print() pure virtual function.*

**Figure 5** *  
This screenshot shows the IT class that is derived from its base class of Staff. This class inherits all properties of Staff which also inherits the properties of Person, in addition to the Administrator’s title. Note: this information is printed using a pointer to the specific object type of Administation from the print() pure virtual function.*

**Figure 6  
***This image shows the pure virtual function that will be overridden for each specific class and their use of the print function.*

**Figure 7  
***This image shows the line of code where the p1 Person pointer was initialized to support dynamic binding for functions.*

**Figure 8  
***This image shows the main code for the test of the Faculty class. It is important to notice where the pointer is set to point to the reference of the faculty object. It is also important to notice the way the pointer object is used with the functions for dynamic binding, yet static binding is also used at some points for specific functions. Finally, the pure virtual function is used at the end to print the information in a specific way to the object type, although only a pointer is used.*

**Final Analysis**

After completing my first fully OOP project, I can truly see the effectiveness of object oriented programming when it uses the three aspects of encapsulation, inheritance, and polymorphism. When going back to redo Lab#4 to add polymorphism and make it truly OOP, I quickly began to realize how polymorphism would make the entire program much easier and efficient to code in two ways, and these two ways also aligned with the projects objectives. The first objective of this lab and advantage of polymorphism was the use of the pure virtual function. Using the pure virtual function allowed me to make my base Person class completely abstract and modify the use of my print function within each derived class type. This means that I could use the same print function for both Student and Staff, yet when I printed Student it would provide a GPA, and Staff would provide a job position title. The use of the pure virtual function proved to be very effective, especially for my print function that would be used within all derived classes in my application. The second objective of the project was to use polymorphism in the idea of dynamic binding on the functions that I was using. To do this I had to create a pointer of my base class Person and set it equal to the reference of whichever object I was going to be using the functions for. I was able to use the pointer to set data into my objects such as the name and date, although I also used static binding for some other functions. The two objectives of the project converged when I ended up using the pointer on the pure virtual print function as seen at the end of *Figure 8.*

References

Pitts, R. I. (n.d.). Introduction to Polymorphism. Retrieved November 18, 2017, from   
 https://www.cs.bu.edu/teaching/cpp/polymorphism/intro/

Jones, N. (2017). *Lab 04:Inheritence Part 1.* Capitol Technology University.

Inheritance and Polymorphism. (2001). Retrieved October 26, 2017, from

http://courses.cs.vt.edu/~cs2704/fall01/Projects/3/HokieHookSpec.pdf