Lab 7

Author: Nigel Nelson
Course: CS2040
Date: 4/24/21

- **What did you learn from doing this lab?**
  - I learned a lot over the course of this two-week lab. For starters, I learned about the nuances of inheritance in c++. For starters, I learned that in order for a child class to inherit from a parent class, you must declare "Child_Class_Name : Parent_Class_Name" in the child class declaration. In addition, I learned that methods are statically bound by default in c++. This means that if you create a child class under the declaration of a parent class, if you call a method defined in both the parent and child class, the parent class's implementation will be executed. This can be changed by declaring "virtual" before method declarations, which creates dynamic binding. This means that what ever class is calling a method at runtime, the compiler will execute that class's implementation. In addition, I learned about managing memory on the heap and preventing memory leaks. This was done by using the "delete" command to remove objects created with the "new" command from the heap. Also, I learned that destructors are created so that when a class is removed from memory, that class deallocates all of the data that it puts on the heap.
- **What did you find challenging about the lab?**
  - What I found most challenging about this lab was the communication between classes. Early in the design of my lab I struggled with correctly parsing in the specified configuration file and assigning values to the correct objects. Ultimately, this issue came down to a poor understanding of memory management, where I was creating new items, and assigning them by value, which led to their destruction as soon as the scope moved past that individual method. In addition, I struggled with removing items from the heap. This was largely due to my initial misunderstanding that all variables had to be deleted before the end of the scope. This led to me trying to delete local ints and strings inside methods, which resulted in the inability to compile. Lastly, I struggled with inheritance. This was due to the fact that with so many classes, you had to be very careful to include the "virtual" keyword to ensure dynamic binding, and at times I would forget to do this. This resulted in long debugging sessions trying to unearth why a method was returning an unexpected value.
- **What would you recommend changing if this lab is reused in future years?**
  - Overall, I really enjoyed this lab, this is because I do not believe there has been a single assignment to date that has taught me as much about a language as this lab has taught me about c++. The only advice that I would have is to change the instructions in the lab write up such that it is conveyed that the shown UML diagram is merely a possible way classes can interact, and that object-oriented design should be a students first priority when designing this lab.