Elle Lohning
BlazerID = glohning

Tasked with creating an inventory management system, the final product successfully allows users to search the full inventory and browse all products in it. First, the system was coded in C++, as this language has an increased amount of syntax, variable requirements, and initialization requirements than Python.

Syntax requirements meant a lengthier program. Using constructors for variable value setting after intialization may provide a more secure usage of data long term, but when it comes to ease of implementation this complicates the code. Parentheses at the beginning and end of every method or multiline if/else ensure that programmers and the system do not lose track of which class, method, or control flow statement they are in. However, this can visually and syntactically overcomplicate code and cause the readability of the program to vary from programmer to programmer as they decide their own indentation and parenthesis usage. Since Python requires specific indentations to know whether a control flow sequence or method is at its last line of code, this can consolidate code and make it universally more comprehensible from a code legibility standpoint. It also makes writing code a little bit faster if the programmer is already familiar with the fewer requirements of the language.

As I am personally more comfortable with languages that have heavier syntax requirements like Java and C++, this meant the ease of implementation for the database was initially a little bit more difficult. Once I was familiar with the language, the ease of implementation of the database in Python was much smoother than C++, except for its lack of allowing overloaded constructors. C++ requires for loops to be implemented with an initialization, an ending factor, and an incrementation factor. This creates the opportunity for a

variety of errors that Python handles behind the programmer interface, like out-of-bounds array exceptions and infinite loops.

In order to use functions that the programmer has not created in C++, specific libraries must be imported into the file. This meant additional research into what functions specific libraries of C++ included. While this requirement of C++ can be tedious, it is much more efficient for memory and data management than Python's open library. Another similar case is Python's lack of variable type initialization requirements. Not requiring class types makes traversing through arrays, creating variables, and changing variables significantly easier. However, this creates opportunities for programmers to be absent-minded when focusing on intended data types for variables. A less methodical code can lead to unexpected runtime errors, and a less particular syntax can lead to unexpected type errors.

One aspect that made implementation in Python more difficult rather than easier was its lack of the ability to handle overloaded constructors. For the database created in C++, each type of ball (volleyballs, soccerballs, and ping pong balls for table tennis) was able to be initialized under one class type, Ball. Due to the differences in the information needed to specific which type of ball, the constructor created for volleyballs and soccerballs in the Ball class had more parameters. Since C++ allows for more than one overloaded constructor, all three ball types were able to be included in the Ball class. Since Python does not allow more than one overloaded constructor, intializing a ping pong ball object of the Ball class with more than one less parameter was impossible. An additional Table Tennis Ball class had to be created.

The physical notes used to create the individual classes and their parameters are shown in the image below. It is not quite a UML diagram, but along the lines of one to aid my

implementation of the database. Upon implementation, it was decided that a parent class, Product, would erase repetition in the classes of the types of products.