My implementation of a Queue, both in C++ and in Java, uses several variables and methods together to enable the user to create a queue of a set maximum size, add and remove elements, and print the queue back out to the user. I used an array in both programming languages, though through two different methods. I had the array set to a maximum size initially, which allowed me to be able to do checks to determine if adding or removing elements from the Queue would cause data loss or other potential errors. After numerous tests that had provided the correct output, I had determined that my java code had an error due to the maximum sizes. I created the Queue inside of a class in c++ and in Java to abstract the data, and confine the Queue to a single data type, which the user can create and use.

The primary difference between a Queue and Stack is the method for data removal from their lists. A Queue functions similar to a standard “lunch-line” where the first element in the list (line) is taken out, whereas a Stack removes the element that was most recently added. These programs can be easily be modified to be used as Generics for a Queue in each language. In Java, removing the main method would leave Queue usable as a generic template. Removing the main function in the C++ file would leave the class, which is a Generic Queue itself, likely similar to the Queue template provided as a library in C++.

Creating a Queue in C++ and Java is a very similar process, and even at times identical due to the similar syntax. Despite the similar syntax of these two languages, however, the programs are exceptionally different in other regards. While Java requires the user to define all code as elements of a class, C++ allows the user more flexibility in the program’s structure. One key note is that in C++ I was able to implement the dequeue method with a for loop that set the current element to the next element, as long as the iterator, i, had not reached the rear (end of the Queue). While this works in most cases, when the size of the array is the same as the size of the Queue, rear, this should cause an issue, because the element rear does not exist. This is because of the idea that an array starts at element 0, and goes to size -1.

In summary an error that occurred during implementation was that in Java when I had the size of the array and the Queue size the same, this was no problem in C++, because I used a pointer to reference a data type that was declared after the class instance. In Java, because pointers are not used, this created an issue when I was trying to replace elements that were outside of the array. In C++, the rear element is defined with a default value of zero, but never used. To resolve this issue in Java, there are several solutions, such as creating an if statement to catch this case. However, I decided to instead have the array function the same as it does in the c++ example because, despite the extra memory usage, this causes the dequeue’s for loop, on its last iteration, to set the last element of the Queue (array[rear])to the last element of the array (array[size]) when they have the “same” size. I did this by having the array size be 1 element larger than the user defines, so that if they define the size and rear to be same, they will actually differ by 1, which will set the new rear, when doing a dequeue, to 0, which is what we need.

Over all, the programs run very similar, after I had refined one particular error. To see how the java file preformed before the modification, see “example.java” as well as the pictures of comparisons between the two programs (javaExampleERROR vs javaQueueRefined/cPlusPlusRefined). My C++ file creates an int pointer when the class instance is created, and the pointer is defined when the size of the Queue is put in by the user. In my Java program, the Array is defined when the class is created, by having the user create a new class with the array size as a parameter. This causes me to need to increase the Java array’s size to be the given size + 1 to match the same functionality I defined in my C++ program. Both languages allow the user to define the array during the class’s creation, but creating the C++ class with a pointer helps to show the differences in language choices between the two.