Testing is designated to happen at stage five in the software development life cycle. This does not imply that this is the only place testing happens. This is just the stage where the product is checked against the software requirements specifications. Testing usually occurs in every stage of the development cycle. This is to ensure the work being completed is inline with the final product and developers are not getting off track.

During the testing stage, the testers will check the functional requirements and non-functional requirements. The functional testing generally includes the main function,accessibility, error messages and usability. When checking the main functions they will test to make sure the functions work correctly. While checking the accessibility the test will check to see if the different areas can be accessed by the groups that have access and deny access to the groups that do not have the correct authorization. Testing the error messages is to ensure that the correct errors messages are displayed. Testing the usability is to ensure the users can maneuver the software without any challenges. The non-functional testing looks at security, reliability, scalability, and efficiency.. During the security testing, the testers will be looking for the ability to defend against attacks for internal and external objects. Testing the reliability will ensure that the system can perform the requested functions without failing. When testing scalability the main objective is to see how the process and expand to meet increased demand. Here the testers will also look at the efficiency. This test is to make sure the software can handle the quantity of users and not bog down the program.

Testing is vital to a successful software development life cycle because it helps identify the problems sooner rather than later and can save money and time. If there were no tests on the software and the developers just trusted having no errors in their code there could be a mountain of problems when the general public is allowed to start using the software. Testing helps ensure the software released will perform the way it is expected to and give the users an experience that will keep bringing them back to use the software over and over.

Testing should always occur earlier than it does in the software development life cycle. The earlier the testing starts the quicker problems are discovered and fixes can begin. This may slow down the development phase but will speed up the items for release because the software will be of a higher quality and will not have extensive delays just before deployment.

In software development there needs to be a way to check the work that is being completed. This is done through testing. The first step is static testing. This test is done before the program is developed. The next step is dynamic testing. This test is done after the program is developed. Although static and dynamic testing are both ways of testing, they are exceedingly different. With static and dynamic testing being so different, it is very important to use both of them when developing software.

Static testing starts before any code has even been written.Static testing starts with the reviewing of the documentation that has the specific requirements that are needed. These documents will be compared to the design documents to ensure that everything coincides. After these documents are reviewed, a plan on how to test the code will be developed. The code will be written and reviewed to ensure that all of the requirements were addressed and will work properly. During the static testing the goal is to catch errors before the developers create code that will have to be fixed.

After the static testing is completed and the code is written to a point where the code can be tested, this is where dynamic testing begins. Dynamic testing is used to check the behavior of the software, the memory usage, and how the software performs on all of the systems. Dynamic testing is to make sure that the software does the job that it was designed to do, and does it the way that the documents wanted it to do. During the dynamic test the goal is to catch errors and fix them before the product is released.

Static and dynamic testing are very different. Static testing is done without running the code whereas dynamic testing only works if the code is ran. Static testing checks the requirements and designs for problems and dynamic testing checks the program for functionality and performance. Static testing is used to catch errors before the development starts and dynamic testing is used to catch errors before the software is delivered.

It is important to use both static and dynamic testing, because they are used at different times and for different reasons. Static testing is used to set the project up for success and help keep the cost down and the production levels up. Catching the errors and shedding light on problems before the development starts makes this possible. Dynamic testing is used to check the functional and non-functional aspects of the software. When being used with static testing, the developers will have a clear and concise document to build off of.

For most of my testing I used the assertTrue. I chose to use this method because it made sense to add a false statement and see if an error popped up. I also use the assertThrows to check for exceptions. I think that I used this incorrectly. I believe that I used it to test the original true inputs instead of the changed inputs.

Looking back over this project I think that I would have used assertEquals and assertNotNull. f I would have used assertEquals. I could have cut down on the amount of code that I wrote and simply just checked everything that needed to equal a specific integer or string. Since every object had to be checked to not be null, the assertNotNull would have really been helpful, however with my limited understanding I decided to stick with what I know would work.

While working on this project, to stay cautious, I made sure that I understood the code well enough that I had a plan on how to test it. I did not want to struggle with having to understand the array to test. Thinking about the test kept me from trying ideas that I did not fully understand. I felt that while learning how to write lest it was important to know what code was being tested and how the code was working so the test could check the changes. A simple example is checking for null. When developing the code it is simple to put in a comment that says this can not be null, but when developing the test it is imperative to actually check that a variable is not null.

As the world keeps evolving, and becoming more dependent on software to catch problems and run the correct counter measures, the importance of being disciplined to making quality code is ever increasing. We are putting more lives in the hands of software than ever before and cutting corners when testing will only result in more tragedy. I feel that this observation is the best thing that I learned in this class. I now know that writing clear code and ensuring that the test covers a high percentage will help reduce any problems that may occur due to syntax errors, or simple human error.