

SOFE 3980U Software Quality Winter 2020

**Final Project Report**

**Improving quality of JFreeChart - an open-source software**

**Authors:**

Durreshahwar Arif, Student ID: 100587401

Isaiah Iremiren, Student ID: 100651421

Ibaada Arif, Student ID: 100655270

Damola Jimoh, Student ID: 100654285

Matthew Brown, Student ID: 100670025

**Introduction**

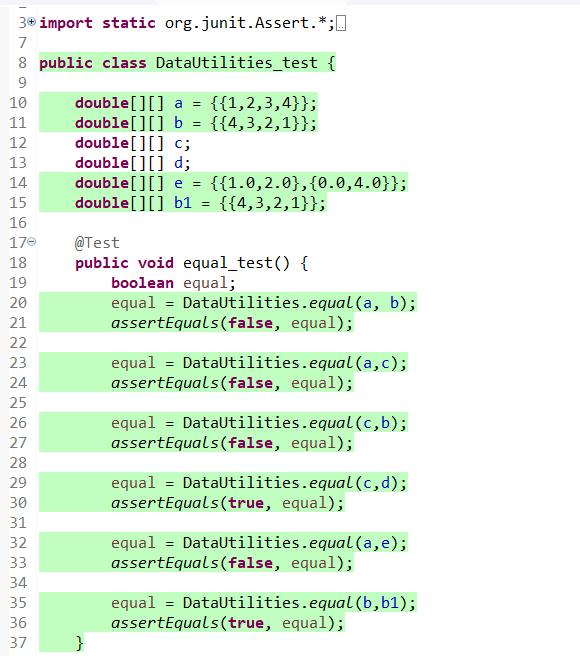
The creation of open-source software is a great way for developers to allow others to be able to fix bugs that are found on the software, it also allows users more flexibility and makes the software cheaper and more accessible. The open-source software we tested was JFreeChart. It is a Java-based framework that allows the creation of both interactive and non-interactive charts such as pie chart, bar chart and gantt chart. Its latest version, 1.5.0, was released in 2017. JFreeChart targets both server-side and client-side applications and supports many output types such as JavaFX components, imager files (PNG and JPEG) and vector graphic file formats. For the purpose of this project, the functionality of the open-source software will be tested.

**Objective**

The objective of this project is to improve the quality of an open-source software through the methods and practices of software testing in order to find and fix any potential bugs existing in the software. Junit test cases were created to identify the code coverage level and test the quality of the software. Test data was utilized to verify the software behaviour given the test cases. Lastly, a set of statistics of code bugs was gathered and analyzed for the purpose of improvement in the software.

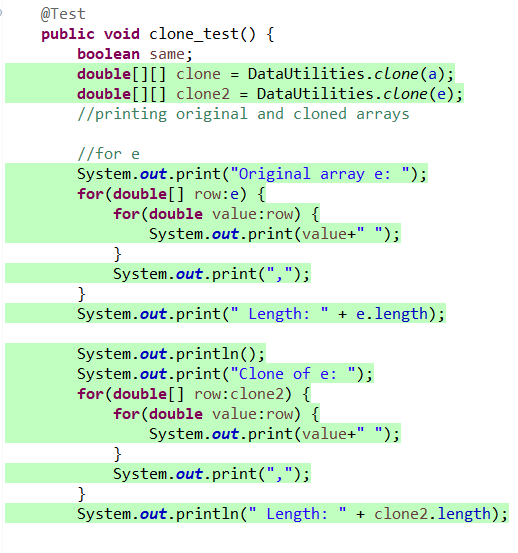
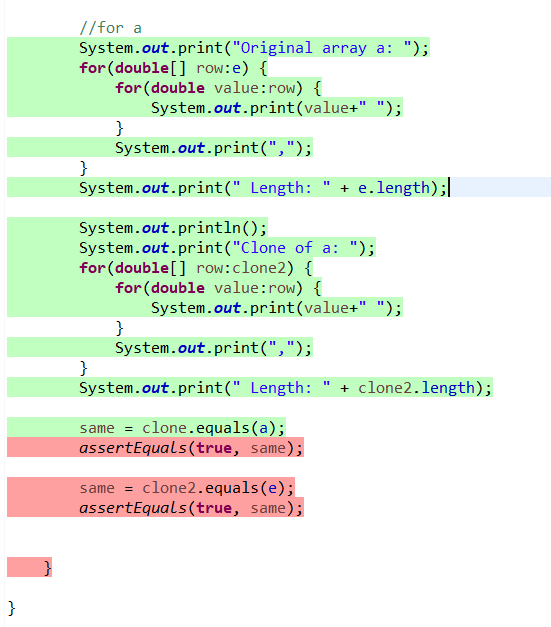
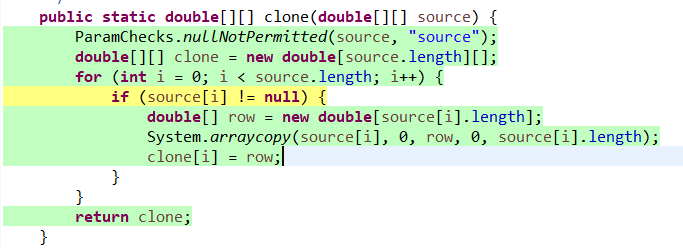
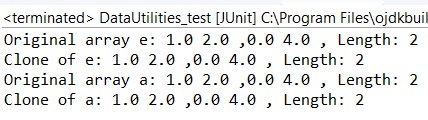
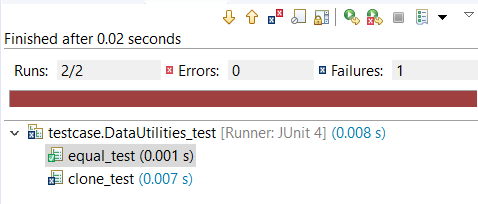
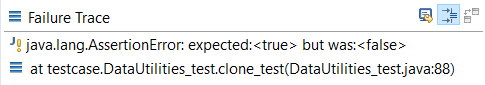
**Test cases**

A series of test cases were implemented given a set of conditions and variables to determine whether the software requirements were satisfied and function correctly. The first test case implemented was the: **Equal\_test** which was aimed to test the quality of the equal function. The job of the equal function is to test 2 arrays for equality. For 2 arrays to be considered equal, they must have the exact same dimensions and their values must match. In addition, Arrays are permitted to be null, therefore, null arrays were also tested. The test case created for equal function verifies the correct behaviour of this function and shows good code coverage.

* + 

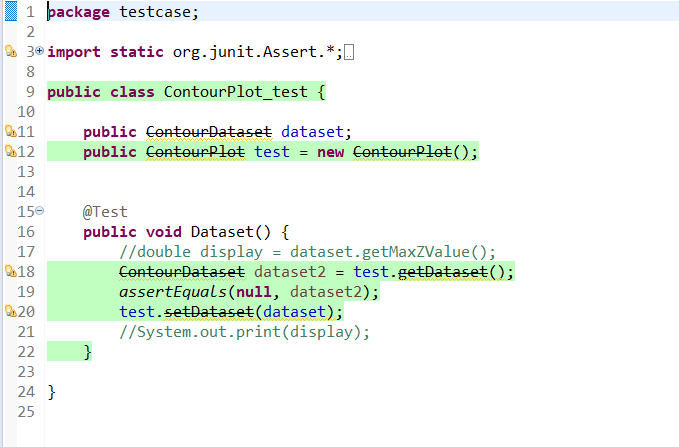
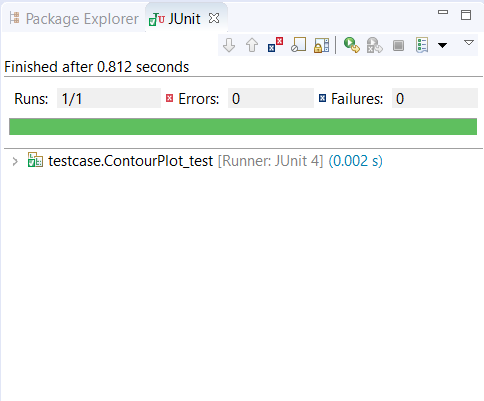
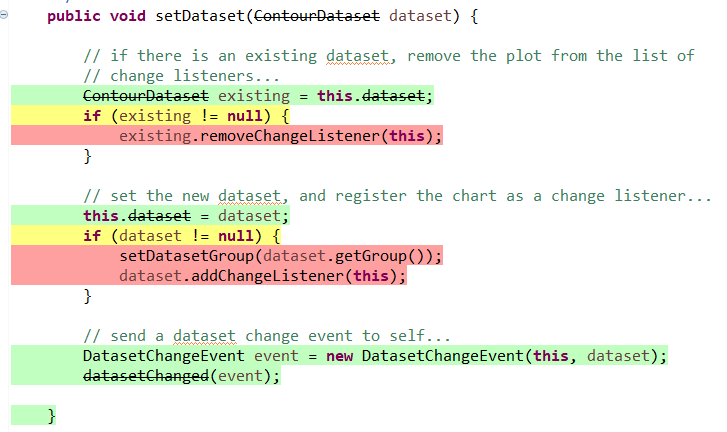
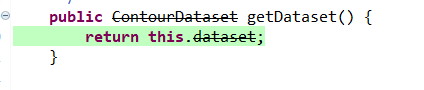
**Figure 1.** Display test case for Equal\_test, the output shows a positive result - test case proves arrays c and d which are both null arrays are found to be equal.

The second test case implemented was the **Clone\_test** - which was aimed to test the quality of the clone function. The clone function takes an array, which is not allowed to be null, and returns a clone of the array. In this case, arrays of different dimensions and values were tested and the clones were the exact same as seen in the console output. However, the test case resulted in failure, when the equal function was used to test the equality of the respective clones. Thus, through this test case, we suspected a fault in clone function which is responsible for cloning of the arrays.

* + 
  + 
  + 
  + 
  + 
  + 

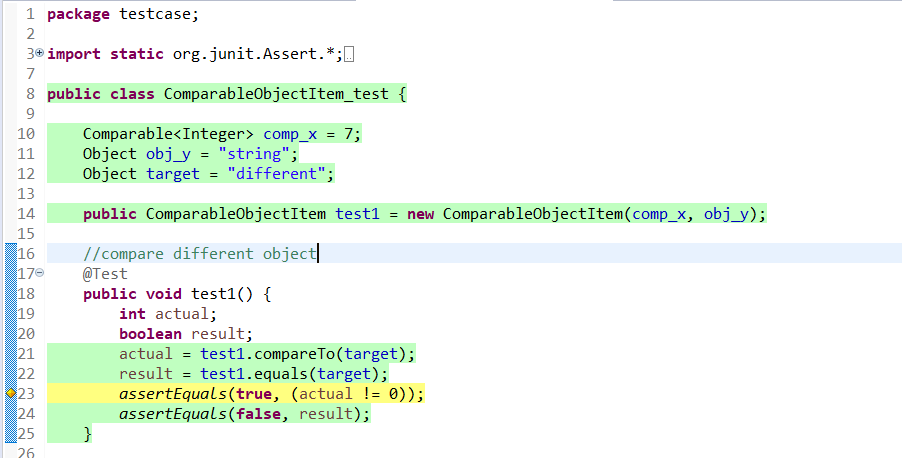
**Figure 2.** Displays a fault in the Clone\_test - the result of the clone was “false” contradicting the expected value “true” value of the clone.

The third test case implemented was the **Clone\_test** - which was aimed to test the quality of the getDataset function. The getDataset function is responsible for returning the primary dataset for the plot, in this case, the plot **test** of **ContourPlot** type. The **test** plot was initialized with a **null** dataset which was stored in the variable **dataset** of type **ContourDataset**. The getDataset function was used to retrieve the dataset within the **test** plot, which was then tested for its **null** value. Although the dataset works fine, a problem was encountered within the testing as the dataset of the plot was not able to initialize to a non-null value, which prevented further testing for the functionality of getDataset function.

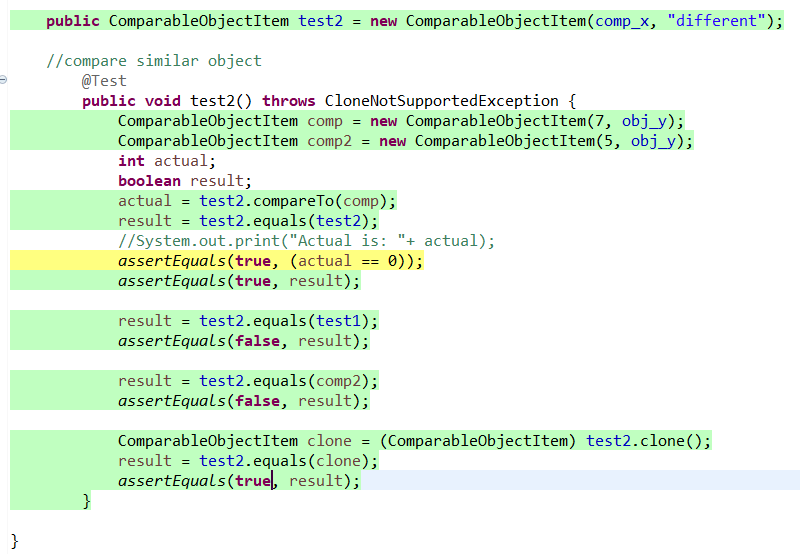
* + 
  + 
  + 
  + 

**Figure 3.** Display test case for ContourPlot\_test, the output shows a positive result - test plot is initialized with a null dataset and stored in the variable dataset of type ContourDataset. The test shows a minor fault as the dataset of the plot is unable to be initialized with a non-null value, preventing further testing of the functionality of getDataset.

The fourth test case implemented was the **ComparableObjectItem\_test** - which was aimed to test the quality of the compareTo and equals functions. The compareTo function is responsible for indicating the order of an object relative to another object. For the order, only the x-value, in this case **comp\_x** of the **Comparable** type, is considered. If the order i.e. the x-values are equal, 0 is returned, and if the order is less than that of the object being compared to, a negative value is returned, otherwise, a positive value is returned. In addition, the equals function tests if objects are equal, and return true only when testing the same objects or a clone and return false if they are not the same.

* + 

**Figure 4.** Displays a test case for ComparableObjectItemt\_test, the output shows a positive result. Displays a test for two objects and if they are equal/clone/same object a true value is returned and false is returned otherwise.

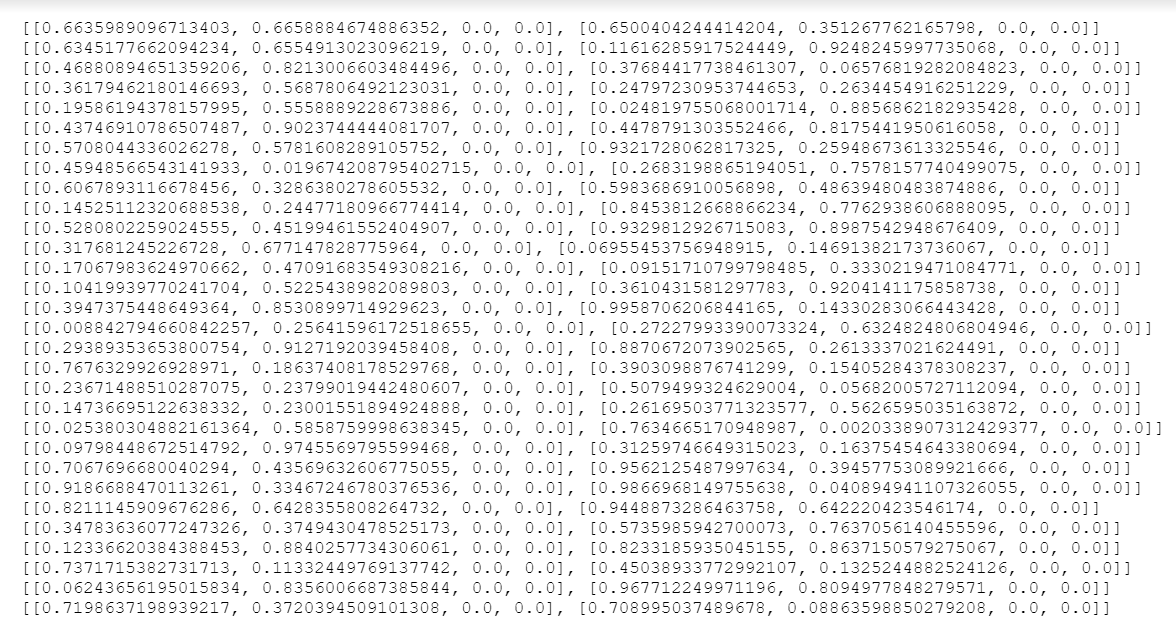
* + 

**Figure 5.** Displays a test case for ComparableObjectItemt\_test, the output shows a positive result.This tests examines the quality of the equals and clone functions. The clone function returns a clone of the object passed to it and throws an exception if the object is of a different subclass. This test examines the function by checking if the clone created returned true when passed to the equals function.

**Test data**

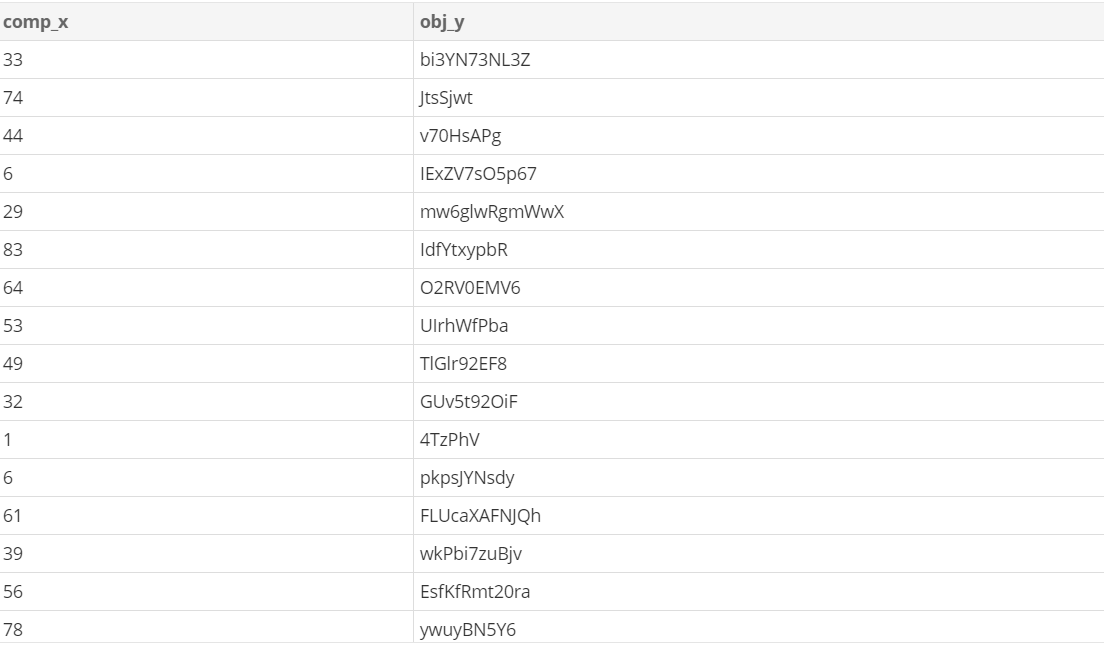
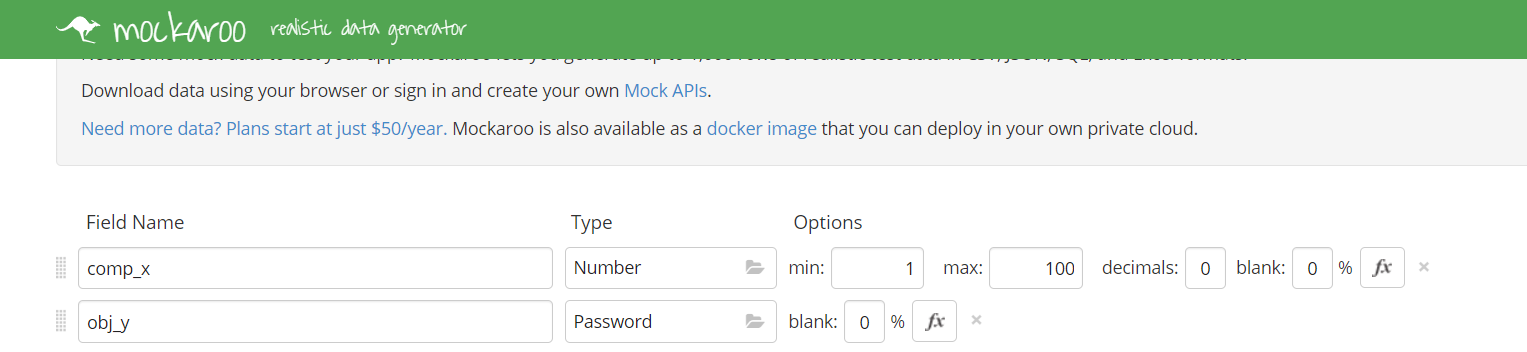
* **Equals Test Data**

Test datas are specifically used for computer programs and their ultimate goal is to verify a given set of input to a given function which then produces some expected result. The first test data used in this project was for the equals test. The test was a randomly generated 2x4 array, which tested the quality of the “equal” function and if the “equal” function worked. As a result, the test passed and worked fine, as the two arrays tested were found to be equal.



**Figure 5:** Equal\_test Data - data randomly generated by 2x4 array to test functionality of the “equal” function. The test data is generated into an a.txt file to be used for the test cases.

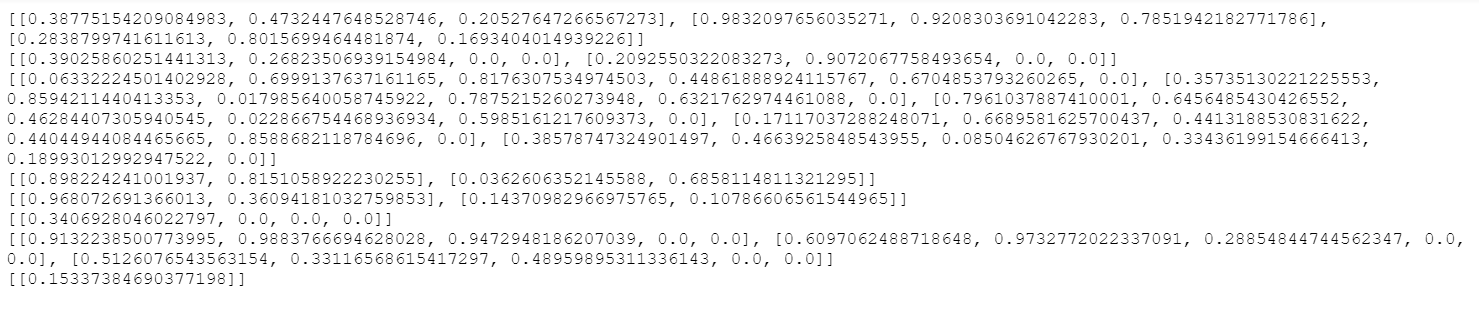
Another test data set for the equals function was created for the “comparable object item” test. An automatic test generator was used to randomly generate a string and a value that were compared to one another and made sure not to be zero. As a result, the test passed and worked fine, as the objects being compared returned true and they were equal.



**Figure 6:** Automatic test generator Mackaroo used to randomly generate a string and a value to be compared to one another and to not be zero.

* **Clone Test Data**

The second test data was used for the clone test. This test used a random generator which generated arrays of different dimensions and were specified by the user input and then tested the function. In the case of the clone test, the result was a failure, as the equal function was used to test the equality of the respective clones. This then concluded with a fault in the clone function, which was responsible for cloning the arrays. In sum, the result of the test was false as opposed to the expected true value.



**Figure 7:** Clone\_test Test Data - randomly generated array with different dimensions used to test the clone function. The function clones the array into another variable and then the test data is generated into an a.txt file and used for test cases.

In addition, the automated test generator Mackaroo was also used for the clone function. This test was to check the quality of the clone functions. The clone function returns a clone of the object passed to it and then throws an exception if the object is of a different subclass. In the test, the function is tested by checking if the clone created returned true when passed to the equals function, and as a result, it passed and worked fine.

* **Clone data test- Test the Contour plot**



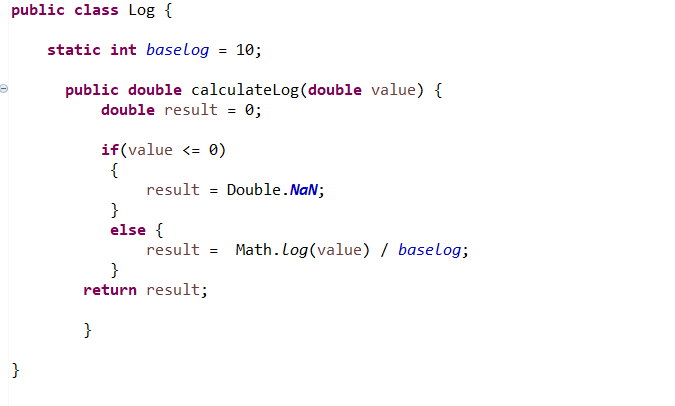
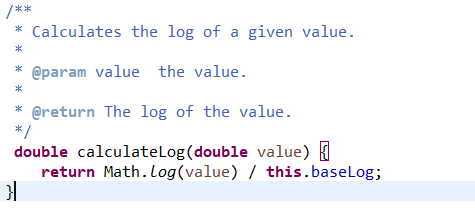
**Figure 8:** CounterPlot test data: the test data for counter plot was randomly generated using a tool called mackaroo. It generates numbers for the x-axis and y-axis dataset for the plotting. The test data was generated into a csv file

**Statistics of Bugs and Improvements**

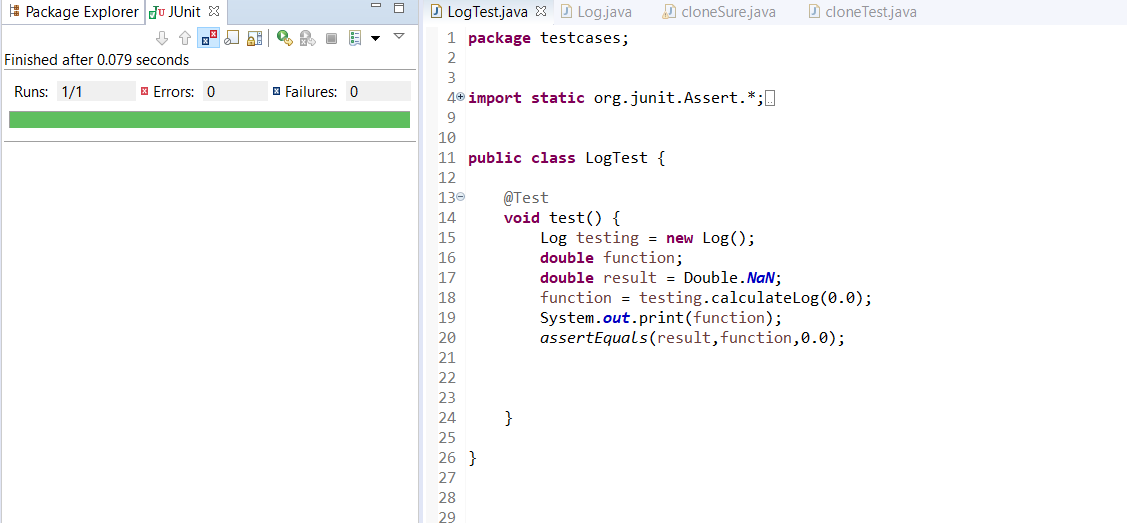
The JFreeChart open source software is an efficient software for producing graphs and data. In our case, when testing various functions, we came across some bugs going through some of the libraries and files.

One bug that was detected was within the clone function. The clone function takes an array and simply returns a duplicate array with the same size. However, when tested, we found that the original array and the duplicate array were not equal, suggesting that there was a bug located within the clone function.

For future improvements, we also came across another class within the JFreeChart framework, which was a logarithmic formater. Since this framework can be used to display data in various formats such as graphs, plots and even polynomial functions, this also includes logarithmic relationships. TheLogFormat class converts information into a logarithmic line, as well as computes the natural logarithm of a number. The bug that exists in one of the functions within this class is the “double calculateLog,” which takes a double value as an argument and finds the natural logarithm. Unfortunately, the only functionality of this class is that it returns the logarithm of the passed argument, but does not consider if the argument is zero. Hence, in order to improve the quality of this function, we should include if-else conditionals to check if the argument passed to this function is not zero, and then return the natural logarithm if it passes. The figures below show the calculateLog function and tests it to calculate the log of a given value with included if-else statements.

**Figure 9 :** To improve the quality of the calculateLog method and prevent the failure of this software, some if-else statements were added in order to deal with a case in which the number passed was less than or equal to 0. There is no logarithmic value negative numbers or 0 so this this updated code on the left shows the if-else logic



**Figure 10:** This was a JUnit test case that was conducted to make sure this function can appropriately handle values that are less than or equal to 0. In order to emulate a situation in which an error would occur, the result variable was initialized to Double.NaN, since logarithms of 0 or less do not actually have a value. The test runs successfully.

**Implementation Challenges**

* Test Case Implementation Challenges

There were a couple of test case implementation difficulties when we arrived at the testing phase of JFreeChart. Some of the class variables were unable to be assigned and simply had to be initialized to NULL. Unfortunately this did reduce coverage because we were not all of the possibilities could be explored. Unfortunately, the inability to identify the cause of this resulted in the increased complexity of code and class coupling.

* Test Case Data Sets Implementation Challenges

One of the challenges faced during the implementation of test case data sets was that we were unable to find an automatic test generator for the arrays. In comparison, an automatic test generator was found to randomly generate a string and a value when compared to one another, however, an automatic generator could not be found to do the same for arrays. In addition, it was difficult to implement a multi-dimensional array in a random test generator tool for the clone function test case and equal function test case.

Another challenge faced was utilizing symbolic test data for all the test cases. It was proved to be difficult to apply as it became very complex to use variables to represent all the data sets that we had. Symbolic test data would have been beneficial to implement because it allows for increased coverage of code especially when there branch conditions incorporated in source code. Symbolic execution cannot be used with arrays either. The clone method involved a two-dimensional array when testing. As efficient as symbolic execution testing can be, the limitations of this technique left us unable to use it.

**Conclusion**

In essence, several test cases were implemented to identify faults within the software - JFreeCharts. Many test data sets were also implemented to verify the expected software behaviour of the respective test cases and statistics on the bugs found within the software were analyzed for software quality improvements.

The test cases that were implemented covered the equal function which tested 2 arrays for equality, the clone function which takes a non-null array and returns a clone of that array, ContourPlot\_test which tests the quality of the getDataset function, and ComparableObject test which tests the quality of the compareTo and equals functions responsible for indicating the order of an object relative to another object.

There were several challenges we encountered when creating the test cases such as not being able to assign variables and as a result resting with null values. It was also difficult to pinpoint the faults within the test cases due to the complexity of the code and class coupling. There were also some test data implementation challenges such as implementing a multi-dimensional array in a random test generator tool for the clone function test case and equal function test case.

Lastly, the statistics of our test cases suggests that there is 1 fault found within the clone method as it fails to successfully duplicate the size and elements of an array. To test this function, randomly generated arrays were used. The statistics of our test cases also suggest that there is a bug within the calculatelog method. The method fails to consider the condition if an argument with the value of 0 is passed. To improve the quality of the JFreeCharts software, some changes were made to certain methods and functions. To improve the clone function, some lines of code were removed to reduce LOC and increase readability. Dynamic testing in the form of unit testing was done to check the functions. To improve the calculatelog method, an if-else statement was added to ensure that when an argument with the value 0 is passed to the function, the program will not fail during execution.