Assignment 1 report

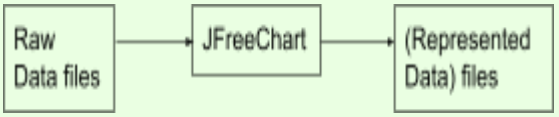
Course: Software Re-engineering.

Course code: COM3523

University username: acb20za

Section 1 : Initial Analysis.

1- Definition.
The target system is JFreeChart. "JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications." as described by David Gilbert (2005).



2- Running maven test succeeded.

3- Finding all *.Java files.

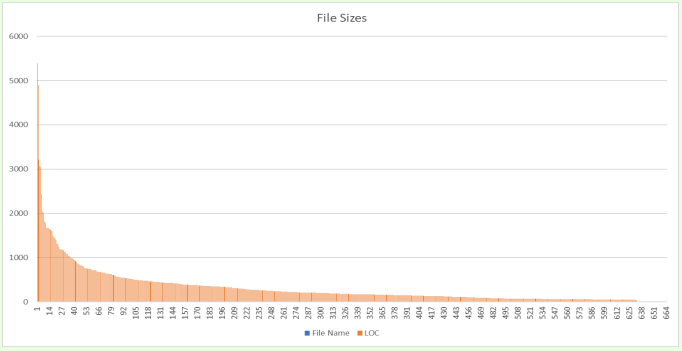
Just looking at the files, it can be speculated that the main java programs either deal with the chart or the data used in the chart.
For the chart, there are different programs that control the different aspects of the chart such as text, plot, legend, title, and rendering a chart.
For the data, there are different programs that are used to process different types of data such as json, time data, xml and xy pairs type of data.

4- Finding files sizes and storing them in a csv file.
An executable bash script lineCount.bh was created and *.java file sizes were stored in fileSizes.csv.

-After sorting file sizes by largest to smallest, it was found that few files contained +2000 LOC and two files in particular contained more than 4000 LOC (XYPlot.java , CategoryPlot.java). It is expected that these two classes are important and would have a lot of functionality inside the system.

-Looking at the source code, XYPlot.java is described as a general class that is used to plot data in the form of (x,y) pairs and can produce different types of charts. Similarly, CategoryPlot.java is a plotting class that renders each data item in the chart. It makes sense that these two files are the largest since the main functionality of the system is to plot and produce charts.

5- Visualizing files sizes data.



As expected, the pattern in file sizes supports the 80-20 law which states that 80% of the outcomes are derived from 20% of causes. In this case, the first 20% of files (largest) are expected to be responsible for 80% of the system functionality. Looking at the top 20% of files (133 files), it was found that 97% of these are concerned with the chart component of the system and the rest 3% deal with the data component of the system.

Section 2 : Reverse-engineer class diagrams.

1- Creating a 'fat jar' for JFreeChart package following lab instructions.
2- Adding the jar file path to the project Run/Debug configuration.
3- Adding ClassDiagram and ClassDiagramGenerator classes to the project.
4- Generating ClassDiagramGenerated.dot by running ClassDiagramGenerator.
5- Visualising the .dot file with GraphvizOnline.

Looking at the resulted class diagram, few observations about key classes were made: (inheritance)

- 4 classes inherit from AbstractRenderer. including:
  - AbstractXYItemRenderer around 28 classes inherit from this class including:
    - XYLineAndShape Renderer
      5 classes inherit from this class
  - AbstractCategoryClassRenderer.
    8 classes inherit from this class including:
    - BarRenderer
      5 classes inherit from this class.

- 2 Classes inherit from RenderState including:
  - XYItemRenderState
    6 classes inherit from this class.

Few observations about key classes were made (composition):

- BarPainter is composed of BarRenderer.
- MinMaxCategoryRenderer, AreaRendererEndType and PaintScale are all composed of two classes each.
- There is a recursive composition at class XYSplineRenderer$FillType.

Finally, it was found that 18 classes have zero relationships to any of the other classes.

6- First class diagram results summary:

Few classes have 5 or more classes inheriting from them as mentioned in step 5. These abstract classes are key classes to look at when analysing the chart.renderer package. In specific, AbstractXYItemRenderer is a very important abstract class as it has the most number of classes inheriting from it. Furthermore, BarPainter is a key class as it is composed of BarRenderer. Evenmore, 5 classes inherit from BarRenderer which means that they are also dependent on BarPainter as well as any classes inheriting from them.

7- Generating enhanced class diagram
After modifying the code to highlight the classes that contained more than

30 methods, another few key classes were identified including: CategoryItemRenderer and XYItemRenderer. Both have more than 130 methods which is significantly higher than all other classes except for AbstractRenderer which has the most number of methods (158).

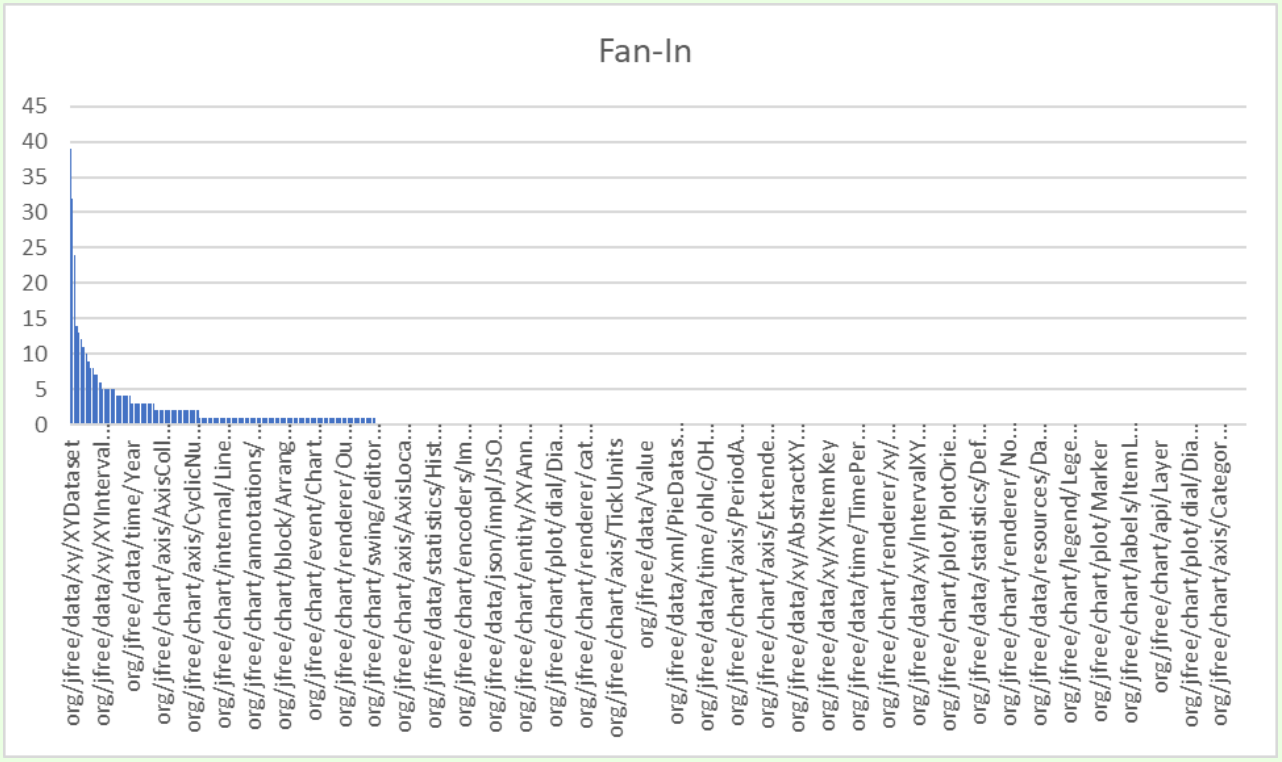
(note that the threshold of 30 methods was chosen as the appropriate threshold after scanning through the number of methods in each class using the class diagram).

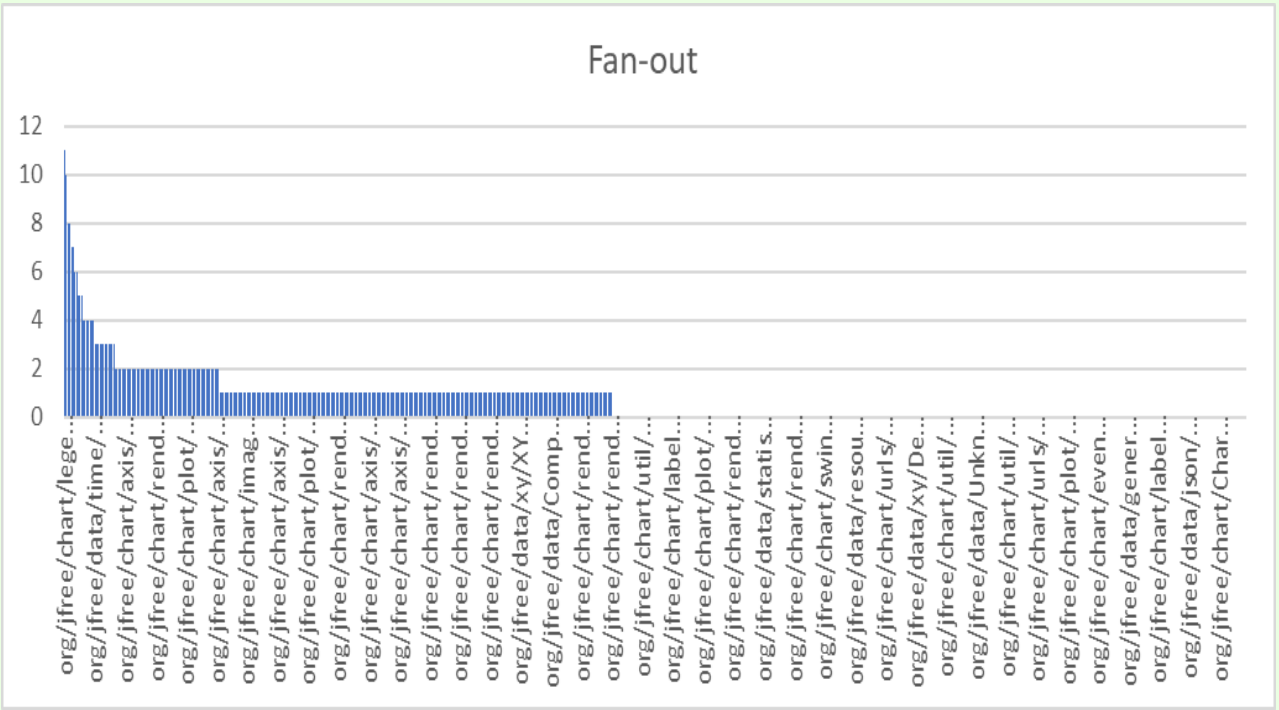
Section 3 : Call Analysis.

1- Added additional classes to calculate the CallGraph (e.g. CallGraph.java).
2- Generate CallGraph for U:/jfreechart/target/classes
3- Generate Fan-In and Fan-Out .csv files.

4- Analysis:
Assuming that the code generated the correct numbers of Fan-In and Fan-Out on a Class level and not Methods level. Few observations were made:

- Majority of the classes have 0 fan-in and fan-out, which was not particularly expected (makes me question my code lol)
- High Fan-In classes indicate that they are frequently reused by other higher level classes.
- High Fan-out classes indicate that they frequently use other lower level classes.
- Keeping those two principles in mind, in addition to the analysis of the files sizes, it would have been expected to see that the classes XYPlot and CategoryPlot show either high fan-in or fan-out as these are expected to be used heavily inside the system. The results were zero for both! (mistake?)
- Moreover, the class that scored the highest fan-in is an interface class called XYDataset. After inspecting XYPlot, it was found that XYPlot uses data from any class that implements XYDataset, which explains why XYDataset has a high fan-in.


Fan-In
45
40
35
30
25
20
15
10
5
0
org/jfree/data/xy/XYDataset
org/jfree/data/xy/XYInterval...
org/jfree/data/time/Year
org/jfree/chart/axis/AxisColl...
org/jfree/chart/axis/CyclicNu...
org/jfree/chart/internal/Line...
org/jfree/chart/annotations/...
org/jfree/chart/block/Arrang...
org/jfree/chart/event/Chart...
org/jfree/chart/renderer/Ou...
org/jfree/chart/swing/editor...
org/jfree/chart/axis/AxisLoca...
org/jfree/data/statistics/Hist...
org/jfree/chart/encoders/Im...
org/jfree/data/json/impl/JSO...
org/jfree/chart/entity/XYAnn...
org/jfree/chart/plot/dial/Dia...
org/jfree/chart/renderer/cat...
org/jfree/chart/axis/TickUnits
org/jfree/data/Value
org/jfree/data/xml/PieDatas...
org/jfree/data/time/ohlc/OH...
org/jfree/chart/axis/PeriodA...
org/jfree/chart/axis/Extende...
org/jfree/data/xy/AbstractXY...
org/jfree/data/xy/XYItemKey
org/jfree/data/time/TimePer...
org/jfree/chart/renderer/xy/...
org/jfree/data/xy/IntervalXY...
org/jfree/chart/plot/PlotOrie...
org/jfree/data/statistics/Def...
org/jfree/chart/renderer/No...
org/jfree/data/resources/Da...
org/jfree/chart/legend/Lege...
org/jfree/chart/plot/Marker
org/jfree/chart/labels/ItemL...
org/jfree/chart/api/Layer
org/jfree/chart/plot/dial/Dia...
org/jfree/chart/axis/Categor...


Fan-out
12
10
8
6
4
2
0
org/jfree/chart/lege...
org/jfree/data/time/...
org/jfree/chart/axis/...
org/jfree/chart/rend...
org/jfree/chart/plot/...
org/jfree/chart/axis/...
org/jfree/chart/imag...
org/jfree/chart/axis/...
org/jfree/chart/plot/...
org/jfree/chart/rend...
org/jfree/chart/axis/...
org/jfree/chart/axis/...
org/jfree/chart/rend...
org/jfree/chart/rend...
org/jfree/chart/rend...
org/jfree/data/xy/XY...
org/jfree/data/Comp...
org/jfree/chart/rend...
org/jfree/chart/rend...
org/jfree/chart/util/...
org/jfree/chart/label...
org/jfree/chart/plot/...
org/jfree/chart/rend...
org/jfree/data/statis...
org/jfree/chart/rend...
org/jfree/chart/swin...
org/jfree/data/resou...
org/jfree/chart/urls/...
org/jfree/data/xy/De...
org/jfree/chart/util/...
org/jfree/data/Unkn...
org/jfree/chart/util/...
org/jfree/chart/urls/...
org/jfree/chart/plot/...
org/jfree/chart/even...
org/jfree/data/gener...
org/jfree/chart/label...
org/jfree/data/json/...
org/jfree/chart/Char...

Section 4 : Dynamic Analysis.

COLLAPSE

```
java
-javaagent:"U:\cfg-and-dot-lab-sevsev73\target\CodeAnalysisToolkit-1.0-SNAPSHOT-jar-with-dependencies.jar"=org.jfree.chart
-Djava.util.logging.config.file="U:\cfg-and-dot-lab-sevsev73\logging.prop" -cp
"U:\jfreechart\target\test-classes":"U:\jfreechart\target\jfreechart-2.0.0-SNAPSHOT-jar-with-dependencies.jar;U:\junit-platform-console-standalone-1.9.2.jar"
org.junit.platform.console.ConsoleLauncher --select-class
org.jfree.chart.XYBarChartTest
```

Generated Logged files but ran out of time to process them. Just by scanning, I have found that even tho the classes have different functions they defo share some method calls. E.g
org.jfree.chart.internal.Args
nullNotPermitted

Which was noticed as the most used method in previous steps! Makes sense.