Individual Project

Analysing JFreeChart

Version 1.0.0 (3/3/2023)

1. Overview

JFreeChart is a long-standing, popular Java framework that enables developers to easily incorporate plots or charts into their applications. At 21 years old it is a genuine piece of legacy code.

In this project we will carry out a “fact-finding” mission. We will analyse its source code, monitor its executions, and will analyse its development history.

2. Setup

We will use GitHub Classroom, which will contain the template repository containing the software analysis code and scripts in it. Follow this link to access your individual GitHub classroom repository for this assignment:

https://classroom.github.com/a/cuGZAOXA

For the target system, we will be analysing JFreeChart. Use the version we have forked, which can be cloned from here:

https://github.com/UOS-Reengineering/jfreechart.git

You should clone the above repository to your PC for the sake of analysis. To avoid confusion, any commits you make as part of the submission need to be commits to your classroom repository (as cloned from the top link). You will only be using the JFreeChart repository to gather your data and as the subject of your analysis.

3. Instructions

For this assignment, the goal is to put all of the skills that we have covered so far into practice. The main assessed part of the work will be a write-up, but figures and committed code will be required as well. The report needs to adopt the following structure:

1. **Initial analysis** (200-400 words)

   Using a set of Bash commands, analyse all of the non-testing Java files in JFreeChart (i.e. files in the ‘src/main/java’ directory).

   Analyse the distribution of file sizes (measured as the number of lines of code). Store the data as a CSV file.

   In the report, include a visualisation of the data (e.g. a bar plot). You may use Excel to create this, or some other charting software if you prefer. Briefly summarise the results, and describe what (if

anything) these results indicate about the system. Are there any files that stand out? Are there any java packages that are of concern? You may choose to back up your thoughts with contextual evidence in the source code.

Commit: Make a commit that summarises your code in its own bash .sh file. Use the commit message to link the code to this part of the submission. E.g. “initial analysis shell file”.

2. **Reverse-engineer class diagrams** (200-400 words)

   Lots of the interesting chart-rendering appears to be happening in the org.jfree.chart.renderer package. Let’s take a closer look.

   Reverse-engineer a class diagram of the classes within this package.

   Create a second, enhanced class diagram, where you vary the size / shape / line thickness of the rendered classes in relation to some metric that is obtained via reflection, such as the number of methods in the class, or the number of data members.

   Present the resulting class diagrams. Include a discussion that highlights (1) your understanding of what the key classes are, based on the inheritance and association structure, and (2) what classes may be particularly interesting when one takes the additional information of the second class diagram into account.

   Commit: Make a commit that captures the extra code you needed to write for the enhanced class diagram. Use the commit message to highlight that it is for the enhanced class diagram.

3. **Call analysis** (200-400 words)

   Compute the call graph for all of the methods in JFreeChart. Write code that will create a CSV file that orders the classes in terms of their Fan-In and Fan-Out (note - we require fan-in/out for classes, not methods).

   Present two charts (e.g. bar-plots), created from these CSV files, one of which shows the Fan-In and one of which shows the Fan-Out.

   Provide an accompanying discussion of some notable classes. Compare and contrast to your analysis of the number of lines of code, created in step 1.

   Commit: Make a commit that captures the extra code you needed to write. Use the commit message to highlight that it is for the call analysis.

4. **Dynamic analysis** (200-400 words)

   JFreeChart can render lots of different types of charts. Pick one type of chart. Your objective is to use software reconnaissance to isolate the source code that is relevant to setting up and rendering that specific type of chart.

   For this you will need to identify at least one JUnit test case that exercises this function, as well as as 2-3 that do not.

   Collect the respective traces, and then apply Software Reconnaissance to them.

Discuss your findings. Are any classes involved that were particularly notable in any of the previous exercises? How localised are the files that are involved? Are they all in the same package? Or are they spread throughout the code base?

Commit: Make a commit that captures the extra code you needed to write for the software reconnaissance. Use the commit message to highlight that it is for the dynamic analysis component.

5. **Finding complex classes - COM6523 students only** (200-400 words)
   [Only for students on COM6523 - no extra credit will be available to COM3523 students who complete this question]

   Write an analysis that will create a CSV file that, for each class, calculates the sum of the cyclomatic complexities for each of its methods. The resulting metric is referred to as the "Weighted Methods per Class" metric or WMC.

   Recall that the cyclomatic complexity is calculated as E-N+2 (where E is the number of edges in the control flow graph and N is the number of nodes).

   Present the metrics in a visual way or as an ordered table. Discuss your findings. Relate these to your previous findings.

   Commit: Make a commit that captures the extra code you needed to write for the WMC computation. Use the commit message to highlight that it is for this component.

6. **Conclusions** (200-400 words)

   Write a small summary of the findings from the various analyses.

   This must draw upon the material covered in lectures - offering some useful justification on the basis of the various forms of information collected, as well as knowledge about good design principles.

Assessment

The assessment of your submission will be conducted according to the schema below. Pay attention to the different weightings for your respective degree programmes.

Section	0-39%	40%-54%	55%-70%	70%-100%	Weighting COM3523	Weighting COM6523
1 (Initial analysis)	No successful attempt at either an analysis of file sizes.	There is a functional Bash script, along with data and charts. Very limited insights.	There is a functional Bash script, along with data and charts. Insightful analysis that draws attention to particular files and speculates about their role in the system.	There is a functional Bash script, along with data and charts. Insightful analysis that draws attention to particular files and speculates about their role in the system. Extensive references to evidence. Corroborated by evidence from the source code.	10%	10%
2 (Class diagrams)	No successful attempt to reverse-engineer a class diagram.	A basic class diagram is reverse-engineered. The accompanying analysis is limited.	You produce both class diagrams (basic and enhanced). The enhanced diagram captures at least one additional metric. There is an insightful accompanying analysis.	You produce both class diagrams (basic and enhanced). The enhanced diagram not only captures the additional metric, but this is presented in a way that is intuitive. The analysis delivers some genuinely useful insights into the architecture of the renderer package.	25%	20%
3 (Call analysis)	There is no successful attempt to construct the call graph.	The call graph is constructed, but no fan-in or fan-out is successfully computed. There is a limited accompanying description.	The call graph is constructed, along with a fan-in / fan-out analysis. There is a limited accompanying description.	The call graph is constructed, along with a fan-in / fan-out analysis. There is a comprehensive and insightful analysis.	20%	10%

Section	0-39%	40%-54%	55%-70%	70%-100%	Weighting COM3523	Weighting COM6523
4 (Dynamic analysis)	No trace has been collected.	Trace collection is successful, but the choice of traces is poor. There has been a partially successful effort to implement the software reconnaissance. Some reasonably insightful observations.	Trace collection is successful, the choice of traces is reasonable, but could be better. There has been a successful effort to implement the software reconnaissance. The write-up is limited, but rightly highlights a few relevant classes.	Trace collection is successful, the choice of traces is considered. There has been a successful effort to implement the software reconnaissance. The write-up is insightful, nicely relating to other relevant parts of the report.	25%	20%
5 (Finding complex classes)	No or negligible effort made to compute the WMC.	You have managed to compute the cyclomatic complexity for individual methods. The discussion of the results is sound, but incomplete.	You have managed to compute the full WMC. The results are not presented in a particularly accessible manner. A reasonably comprehension discussion is attached.	You have managed to compute the full WMC. The results are excellently presented, and the discussion is comprehensive.	0%	20%
6 (Conclusions and overall presentation)	Conclusions are not insightful. Important elements are missing, report is hard to read. Everything has been committed as a single commit. The report is not submitted as a PDF.	Conclusions lack insight. Charts and tables are hard to read.	Conclusions are insightful. Charts and tables are easy to read. Writeup for individual sections is sound.	Conclusions are insightful. Charts and tables are easy to read. Writeup for individual sections is sound. Consistently, different parts of the writeup are linked to each other to provide a coherent overall view of the system.	20%	20%

Submission

Your submission will be your GitLab repository (specifically, the final commit you make to it before the deadline).

In your personal repository created via GitHub Classroom, create a new directory called “Submission”. This should contain all of your submission documents.

This should include:

(1) A written report, submitted as a PDF (this is strict - do not submit other formats, such as Word documents). This should include the main figures, tables, and charts, with the written material for each of the 6 parts detailed above.

    Please ensure that the front page of your report contains your university user name (not your registration number, but ID that you use to log into lab computers, e.g. mine is 'ac1nw').

(2) Separate data files for the individual activities (e.g. csv files etc.). If there are lots of them, they can be put into a separate sub-directory or zipped into a zip file. If you compress files, *you must only use zip compression as a format, no other formats will be processed.*

    Data files for each of the activities should be committed separately. Do not commit and push everything in a single go. The point is that you make regular, atomic commits as you work on this. See the guidelines for each task as to what needs to be committed.

Aside from the materials added to the Submission folder, any code that was added or altered to the template repository should be highlighted through individual commits as discussed previously (the code can be located anywhere in the repository).

There will not be a submission-point on Blackboard, and submissions via email will not be accepted.

The deadline for the submission is 17:00 GMT, on the 17th of March.

Late submissions will be subject to standard late submission penalties, as stipulated by the university regulations. A summary of these can be found in the UG and PG student handbooks.

Support

To ensure fairness, we will only respond to queries on the Blackboard forum dedicated to this assignment. As usual, please read other queries before you post your own, to ensure that your query has not already been answered. To assist your colleagues when they are checking the forum, please give your query a descriptive title.

You must under no circumstances post your own solutions (or parts thereof) to the forum - so please be mindful when you are posing questions.

Unfair Means

It is important to bear in mind the departmental rules on unfair means. Activities such as plagiarism or collusion will be treated as a serious academic offence. This could lead to the award of a grade of zero for this assignment.

We will be closely scrutinising submissions to detect such practices, because it is important that this assessment is a genuine reflection of your own understanding of the module.

To avoid any potential accidental wrongdoings, it is especially important that you do not discuss your solutions with other students. If you have questions about this assignment, please use the discussion forum.