

Software Reengineering Project

Mitchel Pyl & Randy Paredis

1 Introduction

This document is meant as additional information on the reengineering and refactoring of the **JFreeChart** project, which was the assignment of the Software Reengineering course of 2019, at the University of Antwerp.

JFreeChart is a Java library that can be used to add/show professional-looking graphs and charts in your Java applications. This inherently implies that is it useful in a lot of different contexts and scenarios that require this kind of feature.

The ability for such a library for being flexible and expandable with a vast amount of new features would therefore be an incredible advantage for this.

1.1 Problem at Hand

At this point in time, **JFreeChart** has a wide range of possible graphs, charts and plots it can generate for any kind of data you'd like. Unfortunately, we don't have any experience with this project and/or its uses. When trying to *Read all the Code in One Hour* (3.2 from [1]) and to *Skim the Documentation* (3.3 from [1]), it is quite difficult to get a good understanding of what exactly the core of this library does.

If a user or a client would, for instance, like to create graphs in which every datapoint has a different, predetermined and userdetermined, symbol associated with it¹, this would be impossible with the current state of the code. A general fix for this kind of problems within the software-world is to refactor the code so it becomes more flexible and easier to understand.

When we take a closer look at the code, we can identify a few different symptoms why the code in general should be refactored (*Missing tests, Too much time for simple changes...*; 1.1 from [1]).

¹As was the assignment.

2 Project Management

2.1 Setting Direction

The most important aspect of managing a reengineering project is to find a strategy in which the reengineering will be the most useful and successful (Chapter 2 from [1]). This is why we first discussed a strategy to use in the actual reengineering, before jumping into the code like headless chickens.

Using some tools, we were able to *Agree on Maxims* (2.1 from [1]) and more specifically the *Most Valuable First* (2.4 from [1]). With these strategies in mind, we can give all refactoring targets a weight, so we can easily list the most important ones. As described in 2.4 from [1], such a weight technically has nothing to do with cyclic complexities, but with what's valuable to the customer. In our case, these luckily (or coincidentally) line up to a certain point.

Learning the most important rule in software reengineering, *If It Ain't Broke, Don't Fix It*. (2.6 from [1])², we know we'd best not touch any code that is "working" correctly and has nothing to do with any of the valuable targets. For instance, within the scope of the assignment, it is not useful to take a look at refactoring the `ImageMapUtils`.

While on the topic, although all strategies have their merit and are important in some way, we believe some of them are more important and/or practical to follow. *Keep It Simple* (2.7 from [1]) is one of them, which we will keep in mind during the refactoring process.

3 Project Analysis and Tool Usage

In order to solidly identify the issues with `JFreeChart` and find possible refactoring targets, we made use of a few helpful tools that allowed for clear identification of possible problem areas. This way we can clearly *Study the Exceptional Entities* (4.3 from [1]). Even though we know that what we find may be tedious or ambiguous to interpret, we will still make our conclusions based on what we know and expect.

3.1 Repository Visualization with Gource

The first tool we made use of was `Gource`. It is a clean and fancy piece of software that can turn the history of a git repository into a visual representation. This is useful for a few reasons. First, it allows us to see clearly

²Note: this is a rule, not a lifeline, nor an excuse. (as per [1])

who the main contributors are. There was no surprise that this was *David Gilbert*.

A second thing we could deduce from this simulation is that the code was not made using TDD³. We can clearly see that there are first adaptations to the codebase, before changing the tests.

Thirdly, we can identify the possible points in time when a refactoring stage happened in this project. These are moments when a lot of files are added, removed, or modified; which is highlighted in *Gource*. Granted, it is possible that some of these changes are due to merging multiple branches together.

We've identified that possible refactorings happened in November 2008 (increase in functionality), March 2013 (update to almost all files), December 2014 (update to almost all files and removal of a lot of files), July 2017 (file tree restructuring) and July 2018 (general changes).

Finally, we can use the resulting visualization to *Learn from the Past* (5.5 from [1]). We can see which classes were changed a lot and which ones remained untouched for the main bulk of the development.

Classes that changed a lot most likely indicate that they are coupled to other classes in the system, marking these classes as important in understanding the general feel of the system.

Classes that remained mainly untouched could indicate abandoned code, but let's assume (for academic purposes) another possibility. Let's say that these classes indicate features or functionality that are complete. Usually these features cannot give a good enough representation of what's important in the system and whatnot. In either case of untouched code, we can remove our focus from these classes.

3.2 Repository Mining with CodeScene

Another tool we made use of was CodeScene, the powerful visualization tool using *Predictive Analytics* to find hidden risks and social patterns in your code.

CodeScene allowed us to get a general feel of the current state of `JFreeChart`. It gave us a clear representation of possible refactoring targets (see *attachment 4*) and hotspots (see *attachment 1*) within the code⁴.

When we take a deeper look into the code (or at least the graphical representation thereof), we can identify that we most probably will need to take a look at the `org.jfree.chart.renderer` package (see *attachment 2*)

³Test Driven Development

⁴Please refer to the attachments at the end of this document.

and the `org.jfree.chart.plot` package (see *attachment 3*), as far as the hotspots are concerned.

On the topic of refactoring targets, it is clear that the `org.jfree.chart.plot` package (see *attachment 5*) really inquires our attention. More specifically the `XYPlot` (*attachment 7*), `CategoryPlot` (*attachment 8*), `PiePlot` (*attachment 9*), `AbstractXYItemRenderer` (*attachment 10*) and `AbstractCategoryItemRenderer` (*attachment 11*) classes. In the attachments, the most complex functions are listed (sorted from high to low complexity). These top functions⁵ are most likely to be refactoring targets.

3.3 Code Coverage with Cobertura

6.3 of [1] tells us to *Use a Testing Framework*. Not only is this a good idea in refactoring, but in all software projects in general. JUnit was already available in `JFreeChart`, so there is no need to change or alter this part of the project. Linked with JUnit was `Cobertura`, a maven plugin that allows us to check how much code was covered with the available tests.

The overview that is generated from this plugin gives us enough information in order to determine which classes and functions were not covered in the project, also yielding possible missing tests. These missing tests can be seen as a symptom for code requiring refactoring (1.1 from [1]). But as discussed above, we will *Fix Problems, Not Symptoms* (2.5 from [1]) and more specifically, we will mainly focus on the tests that concern our main refactoring targets.

In general, we can deduce that the code coverage of `JFreeChart` at this point in time is way below comfortable for us.

We also noticed that there is currently no mutation testing being done on this project. Even though we do realize this would give way too much situations and possibilities to cover, we currently have no idea of how good the tests currently are.

4 Bibliography


References

- [1] Serge Demeyer, Stéphane Ducasse, Oscar Nierstrasz. *Object-Oriented Reengineering Patterns*. 2009.

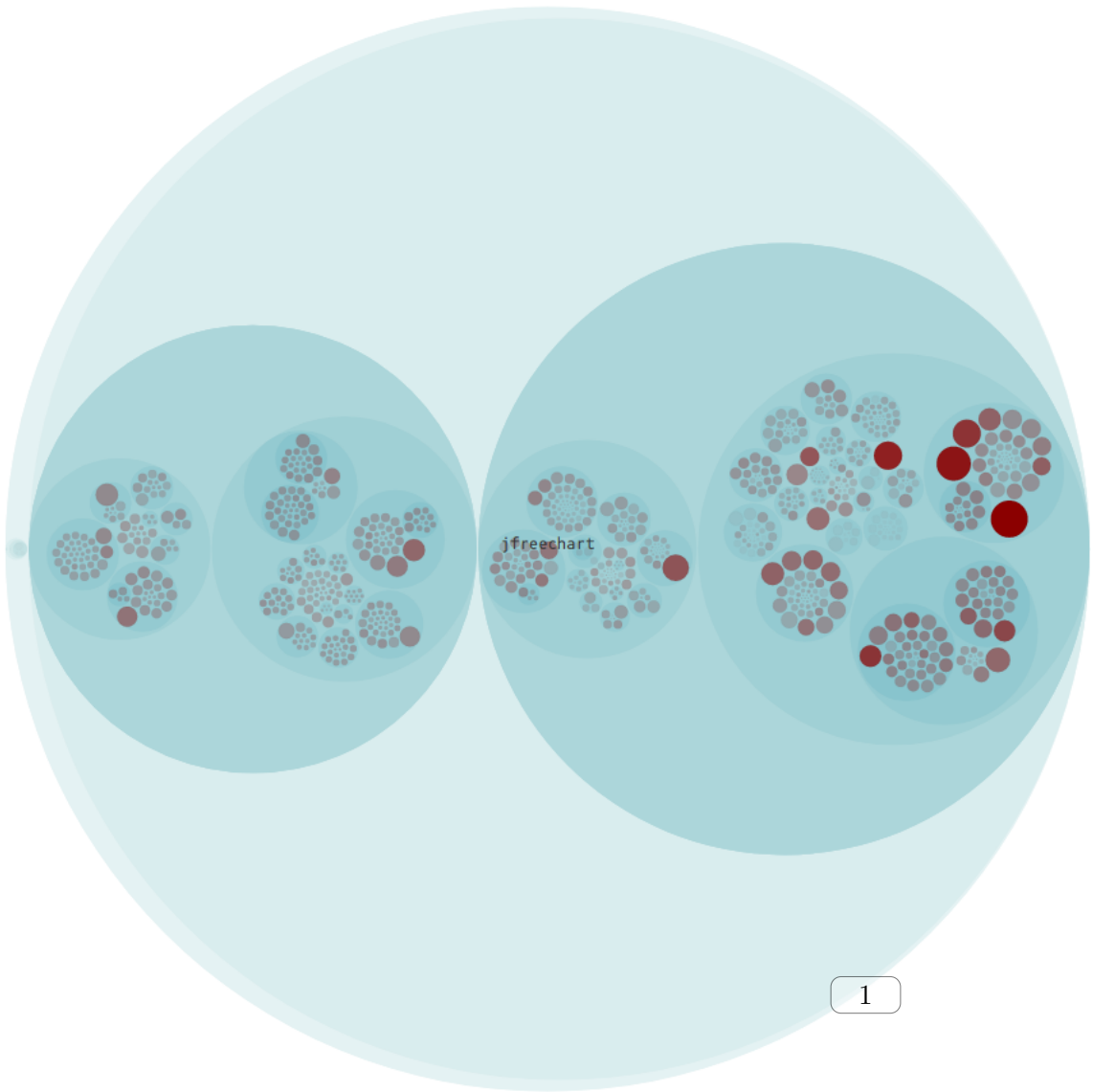
⁵The ones in red.

5 Attachments

On the following pages, we've included a set of screenshots from **CodeScene** that are referred to in the previous sections. The attachment number is overlayed over the image.


Hotspots identify the modules with most development activity -- often technical debt. 

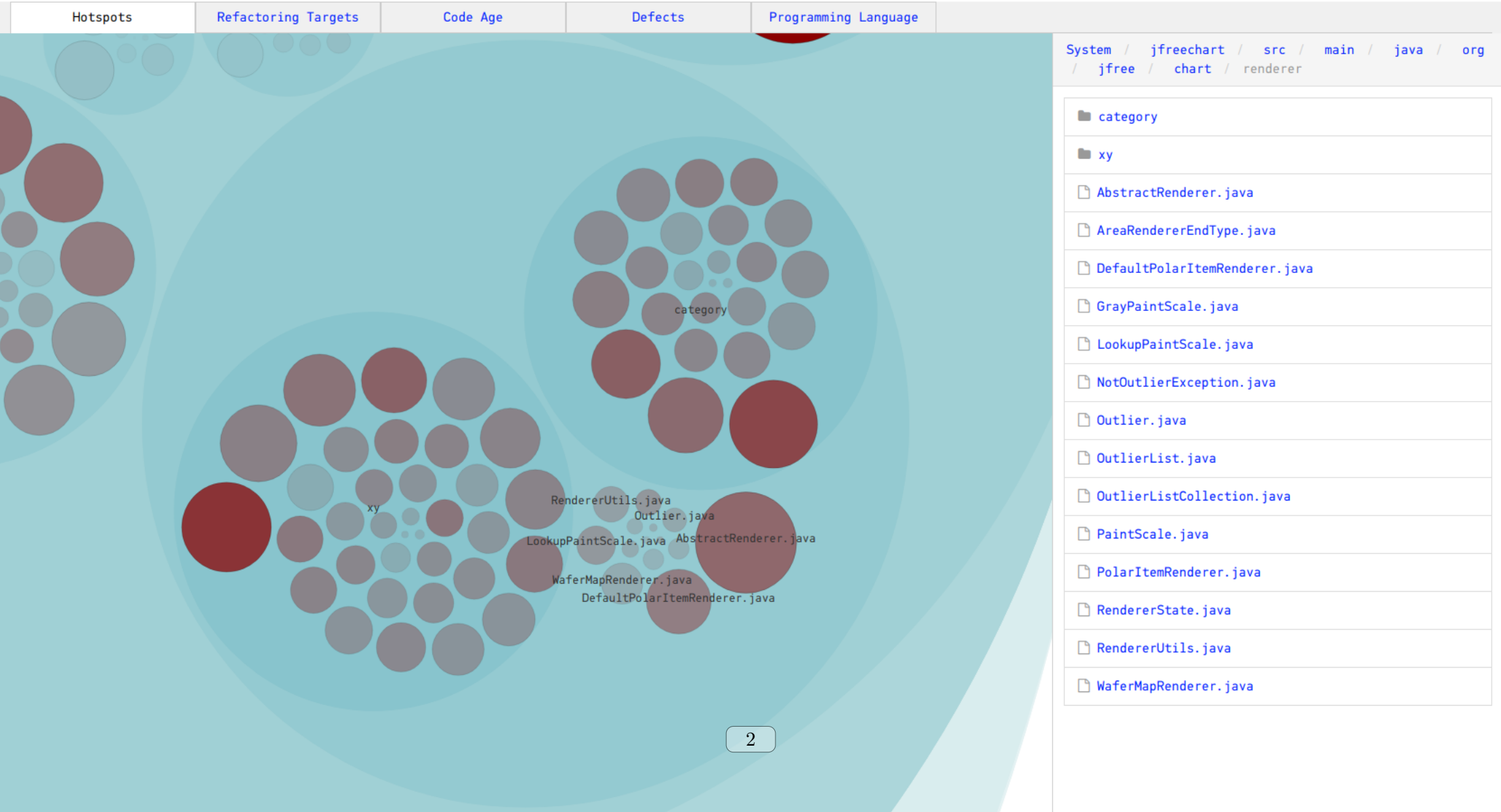
| | | | | | |
|----------|---------------------|----------|---------|----------------------|--|
| Hotspots | Refactoring Targets | Code Age | Defects | Programming Language | |
|----------|---------------------|----------|---------|----------------------|--|




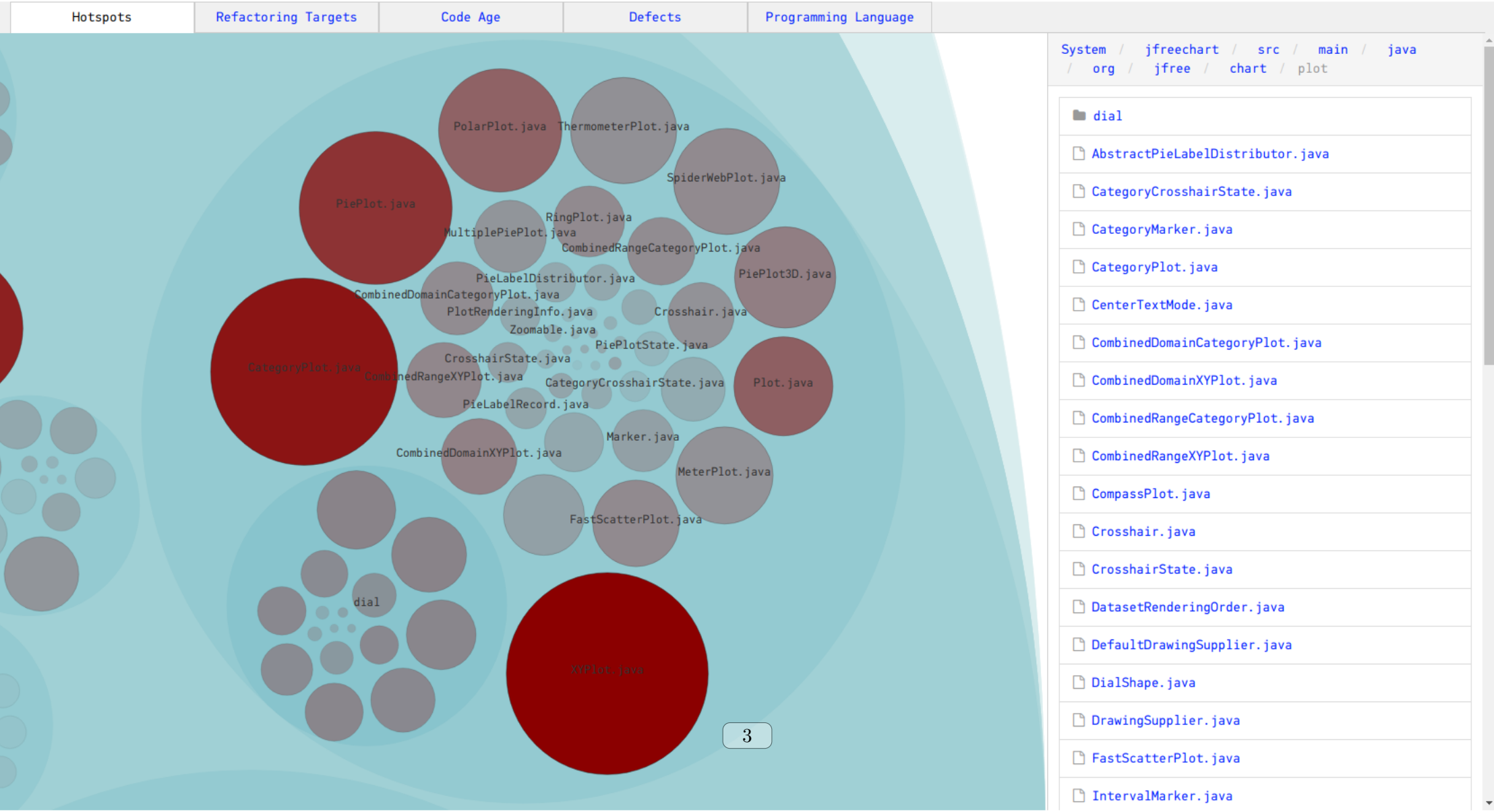
System

 [jfreechart](#)

Hotspots identify the modules with most development activity -- often technical debt. 



Hotspots identify the modules with most development activity -- often technical debt. 



System / jfreechart / src / main / java

/ org / jfree / chart / plot

 dial

 AbstractPieLabelDistributor.java

 CategoryCrosshairState.java

 CategoryMarker.java

 CategoryPlot.java

 CenterTextMode.java

 CombinedDomainCategoryPlot.java

 CombinedDomainXYPlot.java

 CombinedRangeCategoryPlot.java

 CombinedRangeXYPlot.java

 CompassPlot.java

 Crosshair.java

 CrosshairState.java

 DatasetRenderingOrder.java


 DefaultDrawingSupplier.java

 DialShape.java

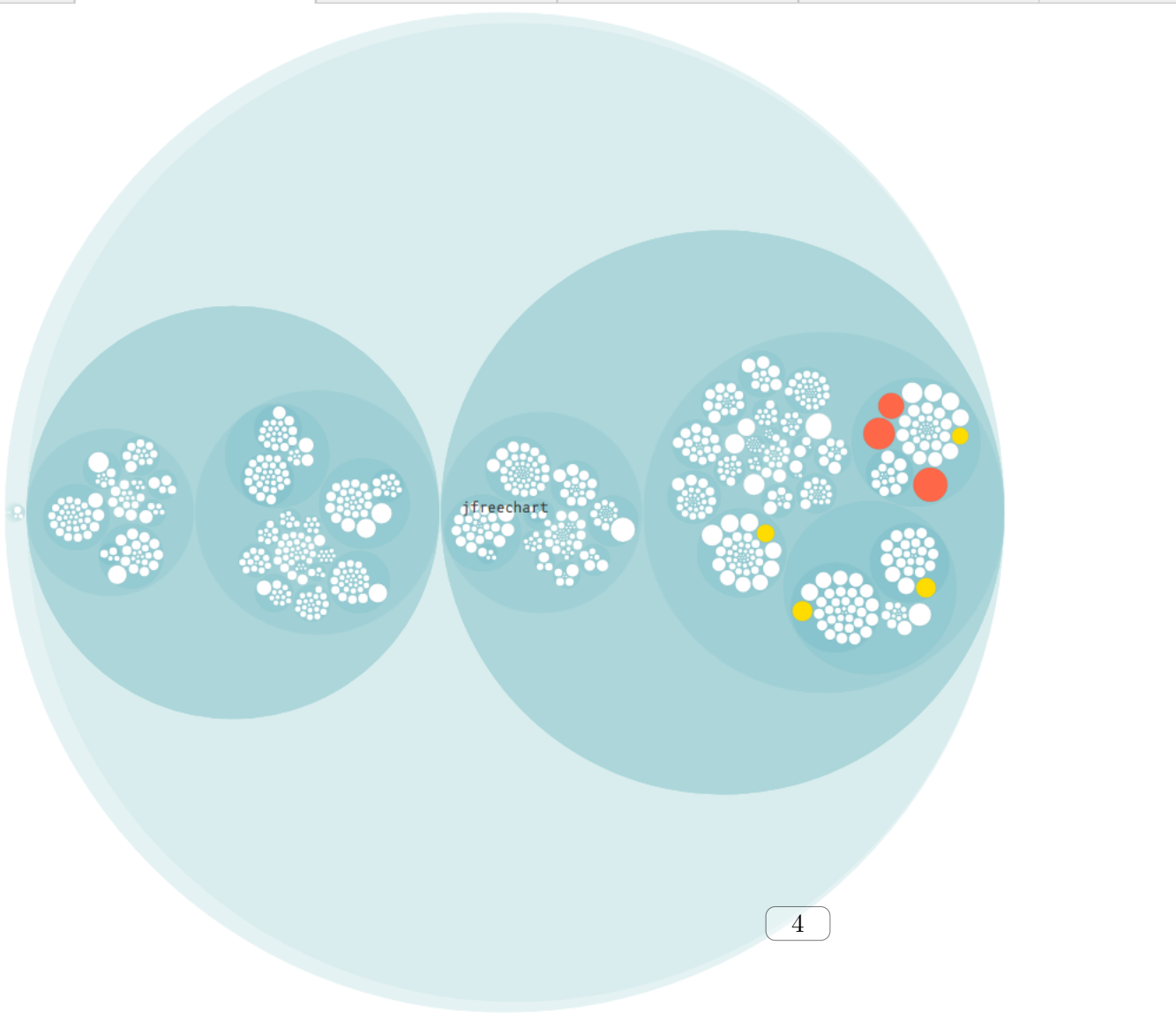
 DrawingSupplier.java

 FastScatterPlot.java

 IntervalMarker.java


Prioritize improvements to the highlighted files. Red is most serious. 

| | | | | |
|----------|---------------------|----------|---------|----------------------|
| Hotspots | Refactoring Targets | Code Age | Defects | Programming Language |
|----------|---------------------|----------|---------|----------------------|



System

 [jfreechart](#)

Prioritize improvements to the highlighted files. Red is most serious. 

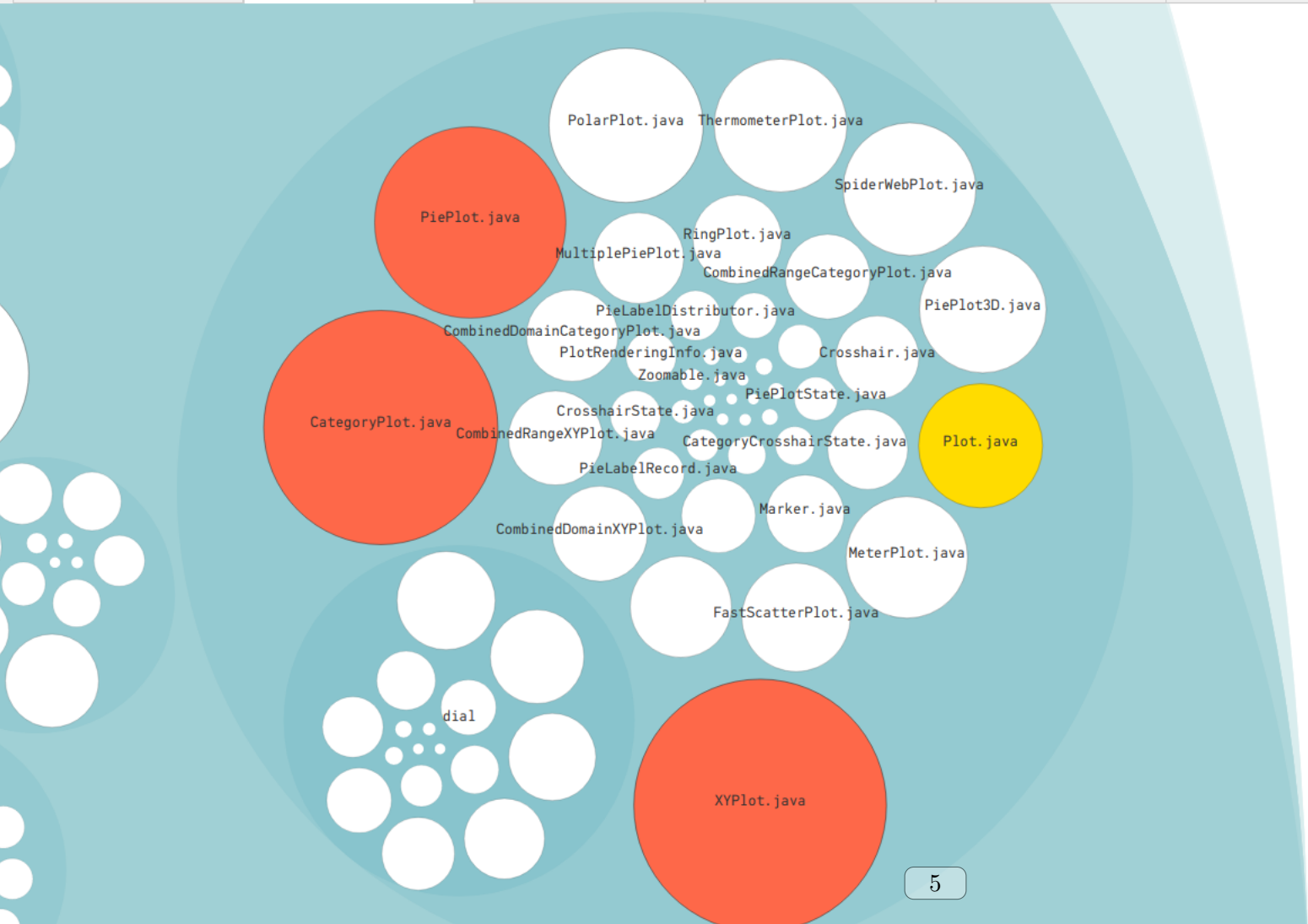
Hotspots

Refactoring Targets

Code Age

Defects

Programming Language



5

System / jfreechart / src / main / java / org / jfree / chart / plot

dial

AbstractPieLabelDistributor.java

CategoryCrosshairState.java

CategoryMarker.java

CategoryPlot.java

CenterTextMode.java

CombinedDomainCategoryPlot.java

CombinedDomainXYPlot.java

CombinedRangeCategoryPlot.java

CombinedRangeXYPlot.java

CompassPlot.java

Crosshair.java

CrosshairState.java

DatasetRenderingOrder.java

DefaultDrawingSupplier.java

DialShape.java

DrawingSupplier.java

FastScatterPlot.java

IntervalMarker.java












































Refactoring Targets













































































Prioritize improvements to these files since they have the highest technical debt interest rate.

| | | |
|--|--|-------|
| jfreechart/src/main/java/org/jfree/chart/plot/XYPlot.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/plot/CategoryPlot.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/plot/PiePlot.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/renderer/xy/AbstractXYItemRenderer.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/renderer/category/AbstractCategoryItemRenderer.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/plot/Plot.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/plot/PolarPlot.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/renderer/xy/XYLineAndShapeRenderer.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/axis/ValueAxis.java | | X-Ray |
| jfreechart/src/test/java/org/jfree/chart/plot/XYPlotTest.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/axis/CategoryAxis.java | | X-Ray |
| jfreechart/src/main/java/org/jfree/chart/renderer/category/BarRenderer.java | | X-Ray |











































X-Ray File Results

| ⚡ Function | ⚡ Change Frequency | ⚡ Complexity/Size | ▼ Cyclomatic Complexity | | |
|-------------------------------|--------------------|-------------------|-------------------------|---|-----|
| equals | 7 | 206 | 112 |  | </> |
| draw | 6 | 245 | 45 |  | </> |
| drawQuadrants | 2 | 106 | 24 |  | </> |
| getDataRange | 3 | 95 | 20 |  | </> |
| render | 5 | 83 | 18 |  | </> |
| drawRangeGridlines | 3 | 39 | 13 |  | </> |
| clone | 6 | 68 | 12 |  | </> |
| drawDomainGridlines | 2 | 37 | 11 |  | </> |
| getLegendItems | 1 | 30 | 11 |  | </> |
| clearDomainMarkers | 2 | 45 | 10 |  | </> |
| clearRangeMarkers | 2 | 45 | 10 |  | </> |
| removeRangeMarker | 5 | 31 | 9 |  | </> |
| readObject | 3 | 55 | 9 |  | </> |
| removeDomainMarker | 3 | 29 | 9 |  | </> |
| drawAxes | 1 | 73 | 9 |  | </> |
| addDomainMarker | 4 | 38 | 8 |  | </> |
| setRangeAxis | 2 | 36 | 8 |  | </> |
| addRangeMarker | 2 | 35 | 8 |  | </> |
| zoomDomainAxes | 3 | 35 | 7 |  | </> |
| zoomRangeAxes | 3 | 35 | 7 |  | </> |
| setRangeAxisLocation | 2 | 23 | 7 |  | </> |
| setDomainAxisLocation | 2 | 23 | 7 |  | </> |
| calculateDomainAxisSpace | 1 | 36 | 7 |  | </> |
| calculateRangeAxisSpace | 1 | 35 | 7 |  | </> |
| checkAxisIndices | 1 | 19 | 7 |  | </> |
| drawDomainMarkers | 0 | 23 | 7 |  | </> |
| drawRangeMarkers | 0 | 22 | 7 |  | </> |
| setDomainAxis | 2 | 23 | 6 |  | </> |
| XYPlot | 7 | 96 | 5 |  | </> |
| getDatasetsMappedToDomainAxis | 4 | 18 | 5 |  | </> |
| getDatasetsMappedToRangeAxis | 4 | 18 | 5 |  | </> |
| panDomainAxes | 3 | 15 | 5 |  | </> |
| setRenderer | 2 | 23 | 5 |  | </> |
| panRangeAxes | 2 | 15 | 5 |  | </> |
| getDomainMarkers | 0 | 17 | 5 |  | </> |
| getRangeMarkers | 0 | 17 | 5 |  | </> |
| removeAnnotation | 5 | 12 | 4 |  | </> |
| getDomainAxis | 1 | 14 | 4 |  | </> |
| getRangeAxis | 1 | 14 | 4 |  | </> |
| getRendererForDataset | 1 | 11 | 4 |  | </> |
| getIndexOf | 1 | 9 | 4 |  | </> |
| indexOf | 1 | 87 | 4 | | </> |
| getQuadrantPaint | 1 | 7 | 4 | | </> |

X-Ray File Results

| Function | Change Frequency | Complexity/Size | Cyclomatic Complexity | |
|----------------------------|------------------|-----------------|-----------------------|---|
| equals | 6 | 180 | 101 |   |
| draw | 9 | 211 | 34 |   |
| drawRangeGridlines | 4 | 43 | 15 |   |
| render | 2 | 60 | 13 |   |
| drawAxes | 1 | 82 | 13 |   |
| clone | 5 | 65 | 12 |   |
| clearDomainMarkers | 1 | 49 | 10 |   |
| clearRangeMarkers | 1 | 49 | 10 |   |
| removeRangeMarker | 5 | 29 | 9 |   |
| removeDomainMarker | 3 | 28 | 9 |   |
| checkAxisIndices | 0 | 24 | 9 |   |
| readObject | 4 | 41 | 8 |   |
| addDomainMarker | 4 | 36 | 8 |   |
| addRangeMarker | 2 | 34 | 8 |   |
| calculateDomainAxisSpace | 1 | 43 | 8 |   |
| zoomRangeAxes | 3 | 35 | 7 |   |
| setDomainAxisLocation | 2 | 23 | 7 |   |
| setRangeAxisLocation | 2 | 21 | 7 |   |
| calculateRangeAxisSpace | 1 | 33 | 7 |   |
| drawDomainGridlines | 1 | 24 | 7 |   |
| datasetsMappedToDomainAxis | 4 | 22 | 6 |   |
| setRangeAxis | 3 | 23 | 6 |   |
| setRenderer | 2 | 26 | 6 |   |
| setDomainAxis | 2 | 23 | 6 |   |
| getLegendItems | 2 | 17 | 6 |   |
| CategoryPlot | 9 | 93 | 5 |   |
| datasetsMappedToRangeAxis | 3 | 19 | 5 |   |
| getDataRange | 1 | 21 | 5 |   |
| panRangeAxes | 1 | 18 | 5 |   |
| getRendererForDataset | 1 | 11 | 5 |   |
| drawRangeMarkers | 0 | 19 | 5 |   |
| drawDomainMarkers | 0 | 19 | 5 |   |
| getDomainMarkers | 0 | 17 | 5 |   |
| getRangeMarkers | 0 | 17 | 5 |   |
| removeAnnotation | 5 | 13 | 4 |   |
| getDomainAxisIndex | 4 | 9 | 4 |   |
| getDomainAxis | 1 | 14 | 4 |   |
| getRangeAxis | 1 | 148 | 4 |   |
| getCategoriesForAxis | 1 | 14 | 4 | |

X-Ray File Results





























| Function | Change Frequency | Complexity/Size | Cyclomatic Complexity | |
|----------------------------|------------------|-----------------|-----------------------|---|
| equals | 7 | 172 | 99 |   |
| drawItem | 5 | 83 | 16 |   |
| drawPie | 5 | 125 | 14 |   |
| drawSimpleLabels | 5 | 82 | 11 |   |
| getLegendItems | 4 | 62 | 10 |   |
| lookupSectionPaint | 5 | 42 | 8 |   |
| clone | 3 | 35 | 7 |   |
| drawLabels | 4 | 61 | 6 |   |
| draw | 4 | 57 | 6 |   |
| lookupSectionOutlineStroke | 3 | 29 | 6 |   |
| lookupSectionOutlinePaint | 3 | 29 | 6 |   |
| getArcCenter | 0 | 42 | 6 |   |
| getMaximumExplodePercent | 1 | 15 | 5 |   |
| getSectionKey | 0 | 12 | 5 |   |
| drawLeftLabels | 7 | 47 | 4 |   |
| drawRightLabels | 6 | 49 | 4 |   |
| drawLeftLabel | 3 | 43 | 4 |   |
| drawRightLabel | 1 | 44 | 4 |   |
| setInteriorGap | 1 | 13 | 4 |   |
| getArcBounds | 1 | 20 | 3 |   |
| getExplodePercent | 1 | 910 | 3 |   |
| PiePlot | 8 | 59 | 2 | |

X-Ray File Results





























Hotspots

Internal Temporal Coupling

Structural Recommendations

| Function | Change Frequency | Complexity/Size | Cyclomatic Complexity | |
|-------------------------------------|------------------|-----------------|-----------------------|---|
| equals | 5 | 45 | 25 |   |
| drawRangeMarker | 3 | 157 | 22 |   |
| drawDomainMarker | 3 | 157 | 22 |   |
| findRangeBounds | 4 | 39 | 12 |   |
| getLegendItem | 2 | 51 | 9 |   |
| findDomainBounds | 3 | 21 | 8 |   |
| getLegendItems | 0 | 21 | 7 |   |
| addEntity | 3 | 28 | 5 |   |
| updateCrosshairValues | 4 | 27 | 4 |   |
| addAnnotation | 4 | 21 | 4 |   |
| drawDomainLine | 3 | 28 | 4 |   |
| drawAnnotations | 2 | 22 | 4 |   |
| drawRangeLine | 1 | 10 27 | 4 |   |
| calculateRangeMarkerTextAnchorPoint | 1 | 17 | 3 |   |

X-Ray File Results

| Function | Change Frequency | Complexity/Size | Cyclomatic Complexity | |
|--------------------------------------|------------------|-----------------|-----------------------|---|
| equals | 4 | 47 | 23 |   |
| drawRangeMarker | 4 | 163 | 22 |   |
| clone | 4 | 71 | 16 |   |
| getLegendItems | 2 | 33 | 12 |   |
| drawDomainMarker | 3 | 72 | 9 |   |
| getLegendItem | 1 | 44 | 9 |   |
| findRangeBounds | 3 | 23 | 8 |   |
| addEntity | 1 | 31 | 6 |   |
| addItemEntity | 3 | 20 | 4 |   |
| initialise | 2 | 29 | 4 |   |
| drawDomainGridline | 1 | 32 | 4 |   |
| drawRangeLine | 1 | 27 | 4 |   |
| calculateRangeMarkerTextAnchorPoint | 1 | 11 16 | 3 |   |
| calculateDomainMarkerTextAnchorPoint | 1 | 15 | 3 |   |