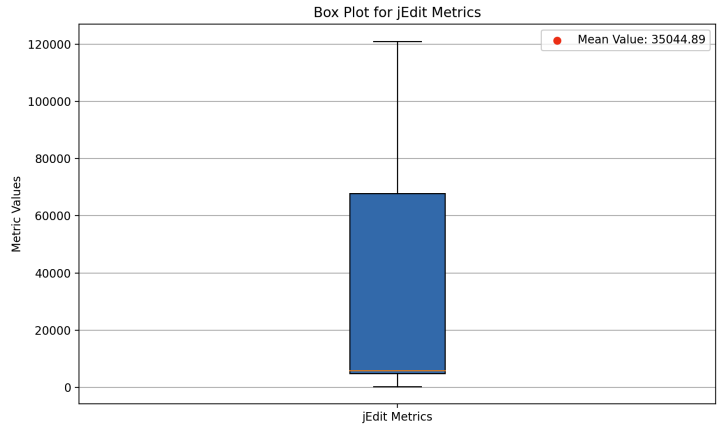
3.3

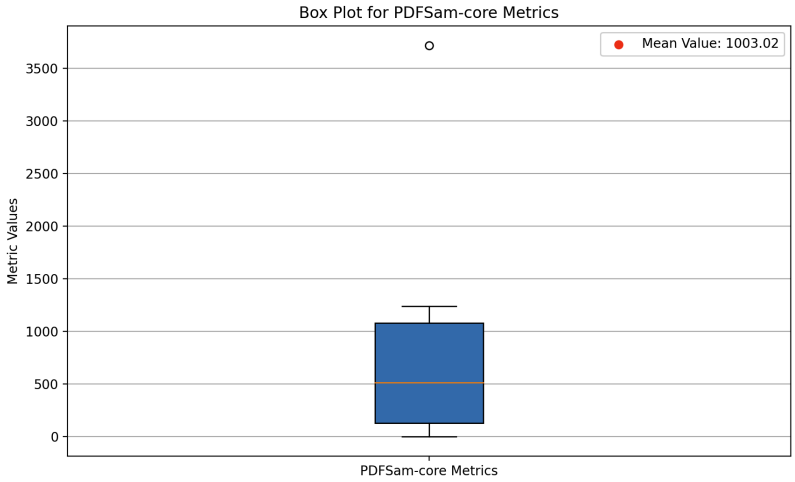


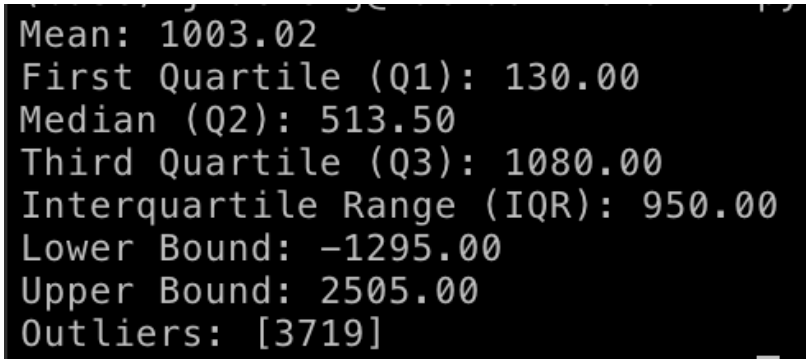

```
Mean: 35044.89
First Quartile (Q1): 4875.00
Median (Q2): 5854.00
Third Quartile (Q3): 67814.00
Interquartile Range (IQR): 62939.00
Lower Bound: -89533.50
Upper Bound: 162222.50
Outliers: []
```

The average value of the metrics (mean) came out to be approximately 35,044.89, which gives a general idea of the central tendency of the data. The first quartile (Q1) was calculated to be 4,875.00, indicating that about 25% of the data points lie below this value, helping to understand the lower end of the metric values. The median value (Q2), which divides the data into two equal halves, was found to be 5,854.00, meaning that half of the values are below this number and half are above. The third quartile (Q3) was about 67,814.00, showing that 75% of the metric values are below this number and helping to understand where the majority of the larger values lie. The interquartile range (IQR), which is the difference between Q3 and Q1, was calculated to be 62,939.00. This range helps measure the spread of the middle 50% of the data, indicating a significant amount of variability. To identify potential outliers, I calculated the lower and upper bounds. The lower bound was found to be -89,533.50 (which doesn't practically apply in this case since negative values aren't relevant here), and the upper bound was 162,222.50.

The distribution of metrics shows a wide spread, with the mean being quite a bit higher than the median. This suggests that there are some very high values skewing the average upwards, which can be seen in metrics like "Total Lines of Code" and "Method Lines of Code." The IQR further supports the idea that there's considerable variability in the dataset, especially towards the upper end.

PDF-core:





For jEdit, the mean value was 35,044.89. The first quartile (Q1) was 4,875.00, the median (Q2) was 5,854.00, and the third quartile (Q3) was 67,814.00, indicating significant variability. The interquartile range (IQR) was 62,939.00, with a lower bound of -89,533.50 and an upper bound of 162,222.50. No outliers were found, suggesting the data is within an expected range despite the variability. For PDFSam-core, the mean value was 1,003.02. Q1 was 130.00, Q2 was 513.50, and Q3 was 1,080.00, resulting in an IQR of 950.00. The lower bound was -1,295.00, and the upper bound was 2,505.00. One value (3,719) was identified as an outlier, indicating some components are significantly more complex than others.

Both jEdit and PDFSam-core show considerable variability, with the mean higher than the median, suggesting high values skew the average. Metrics like "Total Lines of Code" and "Method Lines of Code" indicate complexity that may need refactoring to improve maintainability.

3.4

JEidt:

Cohesion Analysis of Selected Classes

For jEdit, we identified two classes with the lowest cohesion based on their high Lack of Cohesion of Methods (LCOM) values. The first is **`org.jedit.options.CombinedOptions`**, with an LCOM value of 1.12. This class contains four methods, but only one of them interacts with the class attributes, indicating a lack of cohesiveness. The second is **`org.gjt.sp.jedit.gui.DockableWindowManagerImpl`**, with an LCOM value of 1. This class has 28 methods and 11 attributes, yet not all methods utilize these attributes. Many methods instead rely on attributes from five external classes, further demonstrating low cohesion.

On the other hand, two classes in jEdit exhibited the highest cohesion, as reflected by their low LCOM values. The class **`org.gjt.sp.jedit.gui.KeyEventWorkaround`** has an LCOM value of 0, with three methods and three attributes, all of which are used consistently across all methods. The second highly cohesive class is **`org.gjt.sp.jedit.textarea.ElasticTabstopsTabExpander`**, also with an LCOM value of 0. This class has three methods and one attribute, and all methods utilize this attribute, resulting in a high level of cohesion.

Differences in Cohesion

The classes with the highest cohesion are tightly focused, with all methods contributing to a single purpose and utilizing the class attributes consistently. In contrast, the classes with the lowest cohesion have methods that are only loosely related, often relying on external data, which leads to fragmented and less maintainable code. High cohesion contributes to better modularity, making the class easier to maintain, test, and understand, while low cohesion can make a class more difficult to work with and prone to errors.

PDFSAM:

In PDFSam-core, we identified two classes with the lowest cohesion, indicated by high LCOM values. The first class, **`org.pdfsam.ui.selection.single.SingleSelectionPane`**, has an LCOM value of 0.972. This class contains 18 methods and 14 attributes, but only 3 methods use 8 of these attributes, leading to low cohesion. The second class, **`org.pdfsam.merge.MergeParametersBuilder`**, has an LCOM value of 0.715. It has 10 methods and 8 attributes, but only 3 attributes are used by the methods, showing similarly low cohesion.

Conversely, two classes in PDFSam-core exhibit the highest cohesion with low LCOM values. The **`org.pdfsam.task.PdfRotationInput`** class has an LCOM value of 0.151, where 4 out of 5 methods access all 3 attributes, indicating high cohesion. Another highly cohesive class is **`org.pdfsam.ui.io.DestinationPane`**, with an LCOM value of 0.42. In this class, all four methods interact with the two available attributes, demonstrating a tight cohesion.

Differences in Cohesion

The classes with the highest cohesion are tightly focused, with all methods contributing to a single purpose and utilizing the class attributes consistently. In contrast, the classes with the lowest cohesion have methods that are only loosely related, often relying on external data, which leads to fragmented and less maintainable code. High cohesion contributes to better modularity, making the class easier to maintain, test, and understand, while low cohesion can make a class more difficult to work with and prone to errors.

3.5

Coupling Analysis of Selected Classes

For jEdit, we identified two classes with the highest coupling based on their Direct Class Coupling values. The first is **`org.gjt.sp.jedit.textarea.TextArea`**, with a coupling value of 20. This class's methods frequently access 24 attributes from 12 different external classes, either directly or through getter/setter methods. The second class, **`org.gjt.sp.jedit.View`**, has a coupling value of 19, as many of its methods interact with 78 attributes from 33 external classes, demonstrating substantial coupling.

On the other end, the classes with the lowest coupling in jEdit are **`org.gjt.sp.jedit.textarea.TextAreaMouseHandler`** and **`org.gjt.sp.jedit.JEditRegisterSaver`**, both with a coupling value of 1. The **`TextAreaMouseHandler`** class has 14 methods, but only one accesses an attribute from an external class. Similarly, the **`JEditRegisterSaver`** class has three methods, with only one interacting with an external attribute, indicating minimal coupling.

In PDFSam-core, the classes with the highest coupling are **`org.pdfsam.ui.selection.multiple.SelectionTable`** and **`org.pdfsam.ui.notification.NotificationsController`**, both with a coupling value of 9. And 8 of the 22 methods access attributes from external classes, while in **`NotificationsController`**, 5 of the 10 methods interact with attributes from external classes, indicating relatively high coupling for the system.

The classes with the lowest coupling in PDFSam-core are **`org.pdfsam.ui.selection.multiple.ReverseColumn`** and **`org.pdfsam.alternatemix.AlternateMixParametersBuilder`**, both with a coupling value of 0. None of the methods in these classes interact with attributes from external classes, indicating a complete absence of coupling.

Difference：

Classes with the highest coupling tend to be larger, with numerous methods and attributes, often implementing complex or varied functionality. This complexity necessitates frequent interactions with elements from other classes or packages, which increases their coupling. For example, these classes might need to call methods or access data in other classes to fulfill their responsibilities, leading to high dependency levels. In contrast, classes with the lowest coupling are generally smaller in size, with fewer methods and attributes, and typically focus on simple or single-purpose tasks. Their limited scope means they don't need to interact heavily with external classes, resulting in minimal or no coupling. These classes are more self-contained, making them easier to maintain and reuse, as they do not depend on other parts of the system.