NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Software Engineering

Merge Document Phase 1

Grupo 08

Code smells

Even though our team found many code smells, which we will specify bellow, we found a common trend in the use of lengthy methods that result in confusing logic, most of them, without detailed comments for better understanding of the code.

Knowing this code is open source we find that it's especially important for, at very least, the code be easy to read weather it is by commenting of shortening big methods in auxiliary logic.

The code smells identified are just a few of the ones we suspect the code has, we will see in the Code Metrics section the effects that this sheer amount of code smells has on the overall logic.

Code smells found

- Inversion of Boolean comparison.
- Switch statements without default cases.
- Nesting of Try-Catch blocks.
- Else if chains instead of using switches.
- Similar methods in classes.
- Cases in switches that do nothing.
- Code (Strings) that should be made constants.
- Data clumps.
- Empty classes with no data.
- Unused parameters.
- If statement succession.
- Methods too big.
- Data classes.
- Long parameter lists.
- Extreme lack of comments that make the above problems even worse.

For a more detailed explanation of the above code smells see the documentation produced by the team in the folder of each member.

Design patterns

The JabRef code is designed used many design patterns from many gof groups. The following are the design patterns we found.

Design patterns found:

- Facade
- Composite
- Template method
- Prototype
- Factory method
- Proxy
- Singleton
- Observer

For a more detailed explanation of the above design patterns see the documentation produced by the team in the folder of each member.

Code metrics

After identifying code smells and design patterns we proceeded to analyse the code using the plugin code metrics and the results we found didn't surprise us.

Code metrics analysed:

- MOOD (METRICS FOR OBJECT ORIENTED DESIGN)
- Chidamber-Kemerer
- Dependency
- Complexity
- Martin packaging

We will present the definitions for each code metrics but to see a more in-depth explanation see the documents developed in each member's directory to see a more in-depth explanation.

- MOOD: These metrics are useful for projects with heavy use of object-oriented programming.
- Chidamber-Kemerer: These metrics analyse individual class efficacy and efficiency, from the amount of methods called by other classes to the coupling each class is subjected to.
- Dependency: The Dependency metrics measures the class/interface/package dependency on others to function properly.
- Complexity: The Complexity Metrics analyse the complexity of the code, mainly the number of paths that need to be tested.
- Martin Packaging: These metrics verify the dependencies between classes and their stability.

After analysis we have all concluded that the code isn't well structured, classes do too much on their own (there should be more inheritance and dependency according to the code metrics) and the code in general is poorly taken care of.

Use case diagrams

From designing the use case diagrams we were able to identify 3 of the main actors involved in JabRef, the User, the File System, and the Web. All three play a major role in the functionalities of the program with the user being the most important.

The user interacts directly with the application to execute the actions it wants to make, most of the actions interact directly with the file system (saving libraries locally ect…) or the internet.

To see the use case diagrams found see the documentation provided by each member.