JabRef

(Notes by Leo Pruijt, HU University of Applied Science, Utrecht, The Netherlands)

JabRef is an open source bibliography reference manager. The native file format used by JabRef is BibTeX, the standard LaTeX bibliography format. JabRef runs on the [Java VM](http://www.java.com/) (version 1.8 or newer).  
User documentation and downloads: <http://www.jabref.org/>  
Source code, etc: <https://github.com/JabRef/jabref/releases>  
Source code analysis from: src\main\java\net  
Architecture documentation: <https://github.com/JabRef/jabref/wiki/High-Level-Documentation>

Documentation and source of JabRef 3.7 downloaded at 2016-11-17.

1. Introduction to JabRef by Tobias Diez (2016-11-28)
   1. Slides: JabRef and its Architecture - Presentation Tobias Diez - 2016-11-28.pdf
   2. Functionality of JabRef
      1. About 15.000 downloads a month: new users + new version for existing users
      2. Actual number of users not known
   3. History of development
      1. PhD students work on it for 3-4 year
      2. Tool is 13 years old in November 2016
      3. Feature driven development --> spaghetti structure
      4. Thereafter the presented architecture was introduced iteratively: Three layers + CLI + Preferences (last position of windows, database)
         1. Iteratively extended since version 2.10
   4. Intended architecture of JabRef (afterwards extended, based on interview 2016-11-29)
      1. Study sheet in presentation + Architecture documentation
      2. Rationale: Old structure: code organized per feature and related dialogue.
         1. Consequence: No reuse; Redundancy in code
         2. High interdependencies
         3. Development slower, since reuse was not possible.
      3. Objectives new architecture:
         1. More reuse of functionality in (new) logic layer
         2. Clear data objects (which are used everywhere) in the model layer
      4. Way of improving:
         1. A feature needs to be developed or improved.
         2. First a refactoring takes place in line with the intended architecture.
         3. Thereafter, the feature is implemented. Sometimes there is no time to implement the feature in the new release, while the refactoring is implemented.
      5. Future improvements
         1. Put more in model
         2. Aggregate and integrate packages within logic layer which cover the same functional area, like importer and exporter
   5. Test classes test on the main architectural rules (by means of reflection):
      1. Model is not allowed to use logic or gui
      2. Logic is not allowed to use gui
      3. Only gui is allowed to use java.awt and javax.swing
2. Intended architecture registered with HUSACCT
   1. The final intended architecture (modules, assignment of software units to code, rules) is documented in: JabRef 3.7 Intended Architecture - Final.xls (report generated by HUSACCT).
   2. Tobias Diez registered the architecture himself in HUSACCT (after a short introduction).
      1. Based on architecture documentation/presentation.  
         However, the documentation was not complete. Tobias used quite some (undocumented) knowledge to assign source code to modules and to define all the rules.
      2. The assignment of source packages and classes took some time, since the source code organization (still) is diverging from the intended architecture.
   3. Correction to the documented high level architecture
      1. GUI is not allowed to use CLI
      2. CommandLineInterface (CLI) is allowed to use GUI
   4. Additions to the intended architecture based on Externals
      1. GUI is the only module allowed to use AWT and Swing
      2. Logic (collab classes) is the only module to use Oracle and java.sql   
         Collab refers to collaboration with the database.  
         Some of the collab classes should be split to match the layered model (e.g. contain GUI related to DB setting/access).
   5. Encapsulated components are not present in the documented intended architecture.  
      Tobias mentioned several packages within Logic where the Facade patterns was used: cleanup, importer, exporter, integrity. However, dependency analysis showed that only package net.sf.jabref.logic.integrity seems to qualify for a component with a facade. If a module of type component is created (with HUSACCT) for this package, with IntegrityCheck and IntegrityMessage assigned to the interface of the component, then no violations to the façade convention rule are reported.

1. Violations in the source code against the rules in the intended architecture
   1. The violations reported bij HUSACCT\_5.3 are documented in: JabRef 3.7 Violations - Final.xls (report generated by HUSACCT).
   2. An overview of the violated rules is shown in the table below.



* 1. Relevance of the architecture compliance check results as explained by Tobias Diez (in an interview by Leo Pruijt after the second session (2016-11-29), where the violations were studied and discussed in detail):
     1. Rule 1: The violations have to be resolved, because it is not clean now. "The command line needs to be extracted."
     2. Rule 2: No big problem, but in the future the preferences have to be made language independent. Especially important, since the GUI will be ported from Swing to Java-FX.
     3. Rule 3: The dependencies of GUI on SQLException are no problem. But model/DBMSConnection should not use Connection and DriverManager.
     4. Rule 4: Violations should be removed. Violations are surprising: logic.exporter should not back call to JabRefMain.
     5. Rule 5: In the future, these violations should be resolved.  
        Solution (Tobias): Reverse control;
     6. Rule 6: Related to SQL violations. Should be solved. Move this part of the class to logic.
     7. Rule 7: As discussed under rule 5.  
        Tobias mentioned here a mapping mistake. Move DBMSConnectionProperties from logic to model.
     8. Consequence: reduces the violations of rule 5 and enlarges the number of violations of rule 7. So, more dependencies of model on preferences; to be dealt with as described in Rule 5.
     9. Rule 8: Violations should be removed. LP Much work? TD Yes, at least some work.
     10. Rule 9: Violations should be removed. Plan: remove globals completely.