Group 28 : DD2480
Assignment 4: Issue resolution

Report for assignment 4

Project

Name: Jabref
URL: https://github.com/JabRef/jabref
Description:
Graphical Java application for managing BibTeX and biblatex (.bib) databases

Onboarding experience

Did you choose a new project or continue on the previous one?
We have selected a new project because the previous one because we didn't find any good issues in java that did not already have any assignees.

If you changed the project, how did your experience differ from before?
The previous project was harder to understand and get into but they had very good documentation for the onboarding. This new project also has good documentation for onboarding, but it was fairly simple to build even without looking too much at the documentation.

Effort spent

Link to spreadsheet with time for each member:
https://docs.google.com/spreadsheets/d/1L6MOPXgzwOv3DVEJGJEkStJ9qTWWLIL1uys3fZB_Ceg/edit#gid=0

Overview of issue(s) and work done.

Projet Plan:
1) Become familiar with the project
   - Understanding the project scope, goals, and constraints.
2) Identify requirements for the issue
   - Determine the specific needs and expectations related to the issue
3) Install required software to test its compatibility with the project
   - Set up necessary tools and environments to assess software integration
4) Develop solution for the issue - make program compatible with target software
   - Design and implement adjustments to ensure compatibility.
5) Develop tests for the generated/refactored code.
   - Create comprehensive test cases to validate the implemented changes.
6) Observe patch-creation procedures for the project.
   - Follow established protocols for generating and applying patches
7) Perform an 'in-house' code review before creating and sending a patch request to the project.
   - Conduct an internal assessment on code quality.
8) If needed, update the solution to include community feedback, re-review the solution and send an updated patch.
   - Refine the solution if needed and resubmit for review.

Architectural Context: (P+: point 2)
The changes belong mainly to the **gui** package. This respects the project's structural design pattern of Facade. We also respect the project's dependency direction, as there are no dependencies between the code added by us towards the code added in the **gui** package from the **preferences** package. With regards to the **gui**, we have added a way to select TeXworks as an option in the program's preferences. The code in **preferences** is there to provide further utility, in this case, by providing a default path towards the TeXworks executable.

Title: JR doesn't send citation commands to TeXworks
URL: https://github.com/JabRef/jabref/issues/3197
Summary: Fix so JabRef can send citation commands to TeXworks.
Scope (functionality and code affected).
Created a new class and had to add simple code to make it work in several other classes (PushToApplications.java, PushToTeXworks.java, JabPreferences.java, PushToTeXworksTest.java)

After successfully resolving the main issue and during the process of writing tests for it, our team discovered that other pushers (Lyx, SublimeText, Texmaker, TeXstudio, Vim, WinEdt, TexShop) lacked associated tests. Recognizing the importance of thorough testing for ensuring the reliability and functionality of these components, we proposed the creation of new tests for these pushers as a new issue within the project's repository. Upon discussing this suggestion with the project owner, we reached an agreement on the significance of this task and were provided with guidance on how to proceed.

Title: Tests for Push application (Lyx,SublimeText,Texmaker,TeXstudio,Vim,WinEdt,TexShop) #10941
URL: https://github.com/JabRef/jabref/issues/10941
Summary: Create tests for the pusher applications.

Requirements for the new feature or requirements affected by functionality being refactored

Fix so TexWorks is listed as an option in JabRef and so that it correctly sends citation commands. Since TexWorks did not previously exist as an option there were no tests for it. Just one other external program has an existing test.

Optional (point 3): trace tests to requirements.

- ***Identifying and describing requirement :***
  - **R1 - Application integration:** The software must correctly integrate with external applications, specifically TeXworks, to push citation keys.
  - **R2 - Command line generation:** The system must generate the correct command line instructions for pushing entries to TeXworks.
  - **R3 - Display name retrieval:** The system should accurately retrieve and provide the display name for TeXworks to the user interface.
  - **R4 - Tooltip information:** The software should provide meaningful tooltip information for the "PushToTeXworks" functionality.
- ***Tracing tests to requirements :***

Test function	**Requirement ID**	**Requirement description**	**Test purpose**
`testDisplayName()`	R3	Display name retrieval	Ensures the display name for TeXworks is correctly returned.
`testGetCommandLine()`	R2	Command line generation	Verifies that the command line generated for pushing entries to TeXworks is correct.
`pushEntries()`	R1	Application integration	Tests the integration with TeXworks by pushing citation keys (Disabled for automation).
`testGetTooltip()`	R4	Tooltip information	Checks if the tooltip for the 'PushToTeXworks' feature is accurate and informative.

Code changes

Patch

Link to branch with changes for issue #3197:
https://github.com/vladdobre/jabref/tree/%233197-TeXworks

Link to branch with changes for issue #10941:
https://github.com/vladdobre/jabref/tree/%2310941-tests-push

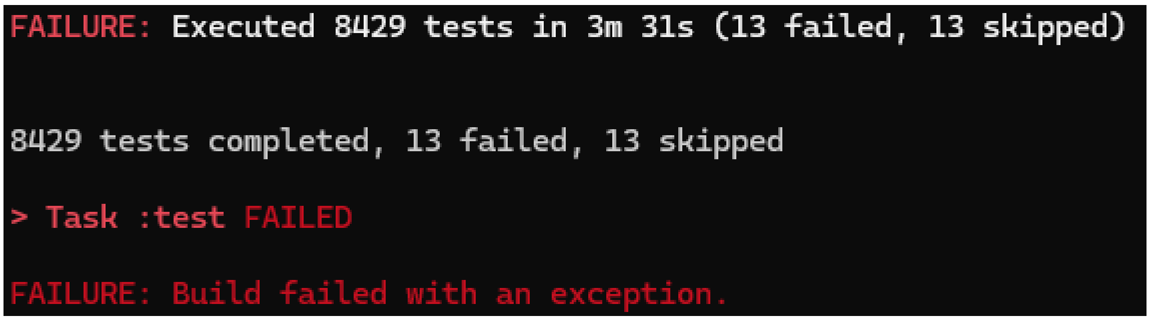
Optional (point 4): the patch is clean:
https://github.com/JabRef/jabref/compare/main...vladdobre:jabref:%233197-TeXworks

Optional (point 5): considered for acceptance (passes all automated checks):
https://github.com/JabRef/jabref/pull/10953 - Our patch for issue #3197 passes all automated checks see link.

Test results

Overall results with link to a copy or excerpt of the logs (before/after refactoring).

Test result before:



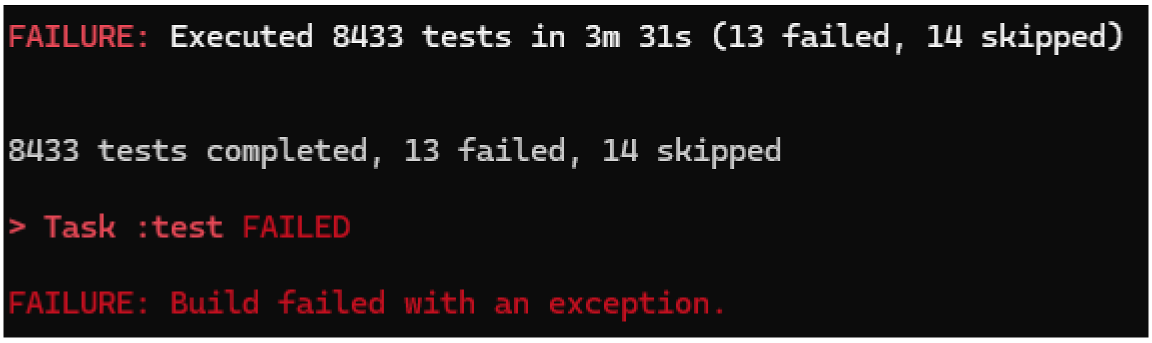
```
FAILURE: Executed 8429 tests in 3m 31s (13 failed, 13 skipped)


8429 tests completed, 13 failed, 13 skipped

> Task :test FAILED

FAILURE: Build failed with an exception.
```

Test result after #3197:



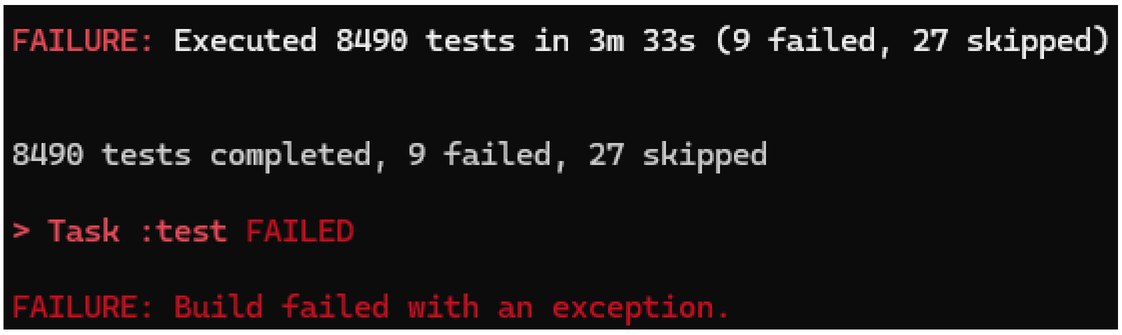
```
FAILURE: Executed 8433 tests in 3m 31s (13 failed, 14 skipped)


8433 tests completed, 13 failed, 14 skipped

> Task :test FAILED

FAILURE: Build failed with an exception.
```

Test result after #10941:



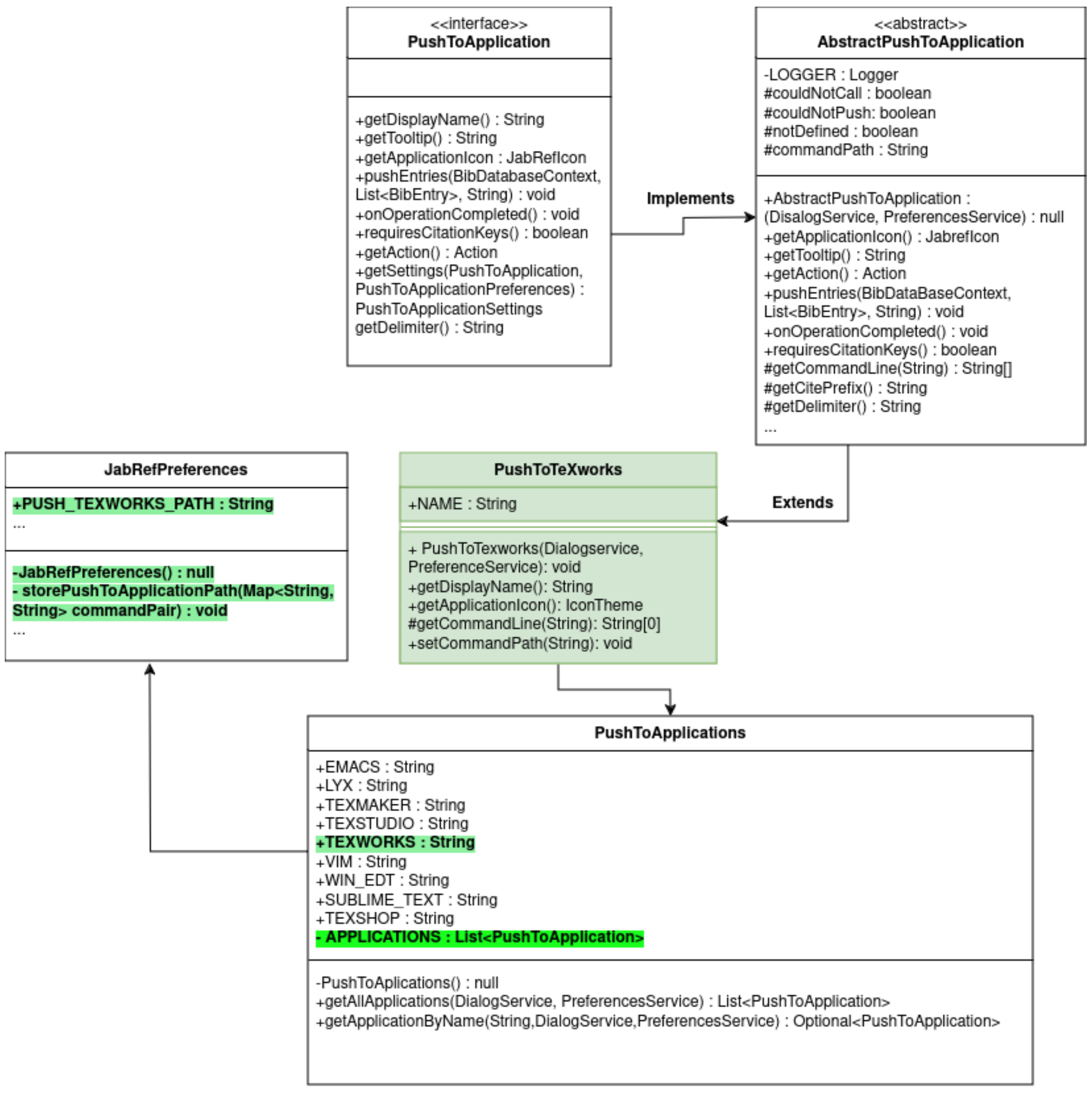
```
FAILURE: Executed 8490 tests in 3m 33s (9 failed, 27 skipped)


8490 tests completed, 9 failed, 27 skipped

> Task :test FAILED

FAILURE: Build failed with an exception.
```

UML class diagram and its description



<<interface>>
PushToApplication
+getDisplayName() : String
+getTooltip() : String
+getApplicationIcon : JabRefIcon
+pushEntries(BibDatabaseContext, List<BibEntry>, String) : void
+onOperationCompleted() : void
+requiresCitationKeys() : boolean
+getAction() : Action
+getSettings(PushToApplication, PushToApplicationPreferences) : PushToApplicationSettings
getDelimiter() : String
Implements
<<abstract>>
AbstractPushToApplication
-LOGGER : Logger
#couldNotCall : boolean
#couldNotPush: boolean
#notDefined : boolean
#commandPath : String
+AbstractPushToApplication : (DisalogService, PreferencesService) : null
+getApplicationIcon() : JabrefIcon
+getTooltip() : String
+getAction() : Action
+pushEntries(BibDataBaseContext, List<BibEntry>, String) : void
+onOperationCompleted() : void
+requiresCitationKeys() : boolean
#getCommandLine(String) : String[]
#getCitePrefix() : String
#getDelimiter() : String
...
JabRefPreferences
+PUSH_TEXWORKS_PATH : String
...
-JabRefPreferences() : null
- storePushToApplicationPath(Map<String, String> commandPair) : void
...
PushToTeXworks
+NAME : String
+ PushToTexworks(Dialogservice, PreferenceService): void
+getDisplayName(): String
+getApplicationIcon(): IconTheme
#getCommandLine(String): String[0]
+setCommandPath(String): void
Extends
PushToApplications
+EMACS : String
+LYX : String
+TEXMAKER : String
+TEXSTUDIO : String
+TEXWORKS : String
+VIM : String
+WIN_EDT : String
+SUBLIME_TEXT : String
+TEXSHOP : String
- APPLICATIONS : List<PushToApplication>
-PushToAplications() : null
+getAllApplications(DialogService, PreferencesService) : List<PushToApplication>
+getApplicationByName(String,DialogService,PreferencesService) : Optional<PushToApplication>

<Interface> PushToApplication:
Interface that is called PushToApplication. It has methods to get the name and tooltip of the application, retrieve its icon, push entries to it, and perform actions after pushing entries. It also includes methods to check if citation keys are needed, get an action associated with the push, retrieve settings, and get a delimiter for the entries.

<abstract> AbstractPushToApplication:
AbstractPushToApplication is an abstract class designed to facilitate the pushing of bibliography entries to various editors. It provides functionalities to retrieve the application's icon, tooltip, and action. The main method, pushEntries(), orchestrates the process of sending entries to the external application. Additionally, it manages different states during the operation through flags such as couldNotCall, couldNotPush, and notDefined. Furthermore, it inherits and implements methods from the PushToApplication interface.

PushToTeXworks:
Function entirely written by us, it contains methods that are designed to make TeXworks appear as an option in jabref as well as methods that give it the correct icon and methods for the importing of the citations. This class also extends AbstractPushToApplication.

PushToApplications:
PushToApplications is a class that helps us manage different applications for pushing bibliography entries. It stores names of various supported applications like Emacs, LyX, TeXmaker, TeXworks, etc. We can use its methods to get a list of all available applications or to find a specific application by its name.

JabRefPreferences:
JabRefPreferences (singleton) is a class that manages preferences for JabRef. It stores various preferences related to the application’s behavior, user interface, citation key patterns, external applications, and more.

Key changes/classes affected

PushToApplications.java,
AbstractPushToApplication.java,
PushToTeXworks.java,
JabPreferences.java,
PushToTeXworksTest.java
PushToLyxTest.java
PushToSublimeTest.java
PushToTexmarkerTest.java
PushToTeXstudioTest.java
PushToVimTest.java
PushToWinEdtTest.java
PushToTexShopTest.java

Optional (point 1): Architectural overview.
Check the pdf: "DD2480_Architecture.pdf".

Optional (point 2): relation to design pattern(s).
Check the Architectural Context in the 'Overview of issue'

Overall experience

What are your main take-aways from this project? What did you learn?

- A problem that looks quite intimidating is sometimes very manageable once you actually start to work on it.

How did you grow as a team, using the Essence standard to evaluate yourself?

Using the Essence standard, we can say that with the end of this assignment our team will be in the **Adjourned** state. Our team was **Seeded** when the assignments were created and we found which group we were part of. We moved onto **Formed** after our first meeting where we set up communication mechanisms and started dividing up the work. Through the next assignments we moved through the **Collaborating** and **Performing** states as we got better at communicating and working together. We were able to consistently meet our commitments. Furthermore, our team proved resilient and adaptable in a changing context. With the completion of all assignments, we can say that our responsibilities have been fulfilled, and as such, this team moved from **Performing** to **Adjourned**.

Optional (point 6): *You can argue critically about the benefits, drawbacks, and limitations of your work carried out, in the context of current software engineering practice, such as the SEMAT kernel (covering alphas other than Team/Way of Working).*

Benefits:
Increased collaboration among team members where each member identified the need for a software based solution for the relevant issue making the work carried out later have a strong foundation (**Opportunity** alpha). In terms of the **Work** alpha, the different states

relating to the **Work** alpha helped the team with organizing how we should work to tackle the issue/s.

Drawbacks:
When several members are working together on the same codebase, they might encounter merge conflicts (**Software System** alpha). Resolving these conflicts can take up a lot of time and effort since we worked on someone else's repository. Some team members encountered challenges with building the project (**Software System** alpha) whether it is a dependency issue or Java version issue. Another drawback occurred at the beginning when choosing a project and a valid issue. The first project we chose had some unassigned issues which suggested that the project is limited in terms of development (**Requirements** alpha), this led to some poor time management in our work in the starting phases of the project.

Limitations:
During our work we noticed that there was an absence of associated tests for contributions from other pushers, because of this lack, the risk of defects in the concerning codebase could occur and potentially lead to instability in the system (a combination of **Work** and **Requirements** alpha**)**

Optional (point 7): Is there something special you want to mention here?

Optional (point 8): In the context of Jonas Öberg's lecture last week, where do you put the project that you have chosen in an ecosystem of open-source and closed-source software? Is your project (as it is now) something that has replaced or can replace similar proprietary software? Why (not)?

To say that our chosen project can replace a similar proprietary software would have to depend on certain aspects:
Features & Usability:
- After a thorough review on the whole project we would say that the project offers a comprehensive set of features for managing BibTeX such as moderately easy integration with LaTeX editors, advanced search capabilities, etc.
- The ease of use and error rates are something to consider for a project and especially how smooth it is to use essential features like database management. We would say that it was neither easy nor hard to use however, concatenating the help of their document and also the previous experiments, we would edge our project more to the easier side than the difficult side.

Community:
- A strong community around the project can be a major factor to a future replacement. An active and large community can provide ongoing support. In this particular case, we thought their documentation was done in a very clear manner making it a bit less confusing tackling the magnitude of the project. A documentation to rely on as support is very crucial when a project is considered as the proprietary software for its area.
- Another thing that a strong community can provide is ensuring that the software remains relevant and sort of "up to date" with the constant changes. An outdated

project does not bode well for the future for obvious reasons. In our project, there were several commits that were done during our time working on the project which indicated that there are ongoing updates and that there is a will to remain working on the project.

In conclusion:
To declare our chosen project as the proprietary software for managing BibTeX and their databases could not be based on just their features, usability and strong community. We had to check other similar projects to make an accurate conclusion. After exploring other similar projects such as Zotero, Mendeley and BibDesk, we could in our limited timeframe draw the conclusion that our project, JabRef, is the proprietary software in its respective area. This is because the other projects lacked or were in our personal opinion weaker than JabRef in majority of the aspects.