JabRef's development strategy

We aim to keep up to high-quality code standards and use code quality tools wherever possible.

To ensure high code-quality,

- We follow the principles of Java by Comparison.
- We follow the principles of Effective Java.
- We use Design Patterns when applicable.
- We document our design decisions using the lightweight architectural decision records MADR.
- We review each external pull request by at least two JabRef Core Developers.

Read on about our automated quality checks at Code Quality.

Continuous integration

JabRef has automatic checks using GitHub actions in place. One of them is checking for the formatting of the code. Consistent formatting ensures more easy reading of the code. Thus, we pay attention that JabRef's code follows the same code style.

Binaries are created using gradle and are uploaded to https://builds.jabref.org. These binaries are created without any checks to have them available as quickly as possible, even if the localization or some fetchers are broken. Deep link to the action: https://github.com/JabRef/jabref/actions?workflow=Deployment.

Branches

The branch main is the main development line and is intended to incorporate fixes and improvements as soon as possible and to move JabRef forward to modern technologies such as the latest Java version.

Other branches are used for discussing improvements with the help of pull requests. One can see the binaries of each branch at https://builds.jabref.org/. Releases mark milestones and are based on the `main` branch at a point in time.

How JabRef acquires contributors

- We participate in Hacktoberfest.
- We participate in Google Summer of Code.

Historical notes

JabRef 4.x

The main roadmap for JabRef 4.x was to modernize the UI, make the installation easier and reduce the number of opened issues.

JabRef 3.x

JabRef at the beginning of 2016 had a few issues:

- Most of the code is untested, non-documented, and contains a lot of bugs and issues.
- During the lifetime of JabRef, a lot of features, UI elements and preferences have been added. All of them are loosely wired together in the UI, but the UI lacks consistency and structure.
- This makes working on JabRef interesting as in every part of the program, one can improve something. :smiley:

JabRef 3.x is the effort to try to fix a lot of these issues. Much has been achieved, but much is still open.

We currently use two approaches: a) rewrite and put under test to improve quality and fix bugs, b) increase code quality. This leads to pull requests being reviewed by two JabRef developers to ensure i) code quality, ii) fit within the JabRef architecture, iii) high test coverage.

Code quality includes using latest Java features, but also readability.