DD2480 Purpose and Architecture - JabRef

Group 28

March 2024

Purpose

JabRef is a reference and citation management tool. It can be used on Linux, Mac and Windows through a GUI. JabRef works across different platforms allowing the user to maintain and manage their references while allowing them to quickly reference them in the user's preferred LaTeXeditor. Regarding the management, among other features, it allows the user to group their reference materials in libraries.

Architecture

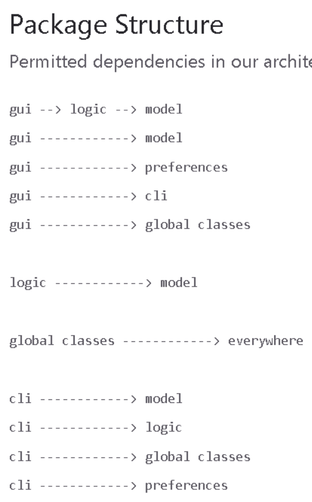
The project provides an in-depth look at its architecture through its documentation. Furthermore, they provide justification for some architectural decisions. Below, **bolded** words refer to packages.

At a high level, the project implements the Facade structural design pattern. The user's interface is completely detached from the underlying logic. This separation is made clear through the different packages contained within the project and is further explained in the documentation as such:

> "We have been successfully transitioning from a spaghetti to a more structured architecture with the **model** in the center, and the **logic** as an intermediate layer towards the **gui** which is the outer shell. "

In order to hold general or utility code, the project includes the **preferences** and the **cli** packages.

Regarding dependencies, the project enforces one-way dependencies towards the "middle", specifically, towards **model**. This is illustrated below in Figure 1. These dependencies are enforced using JUnit tests. Since this is a transitional time for the project, as evidenced in the quote, there are some classes that don't fit "neatly" into the current architecture. These classes are temporarily treated as if they were part of the **gui** package with regards to their dependencies.



Package Structure

Permitted dependencies in our architecture are:

```
gui --> logic --> model
gui ------------> model
gui ------------> preferences
gui ------------> cli
gui ------------> global classes

logic ------------> model

global classes ------------> everywhere

cli ------------> model
cli ------------> logic
cli ------------> global classes
cli ------------> preferences
```

Figure 1: Allowed dependencies - from the documentation