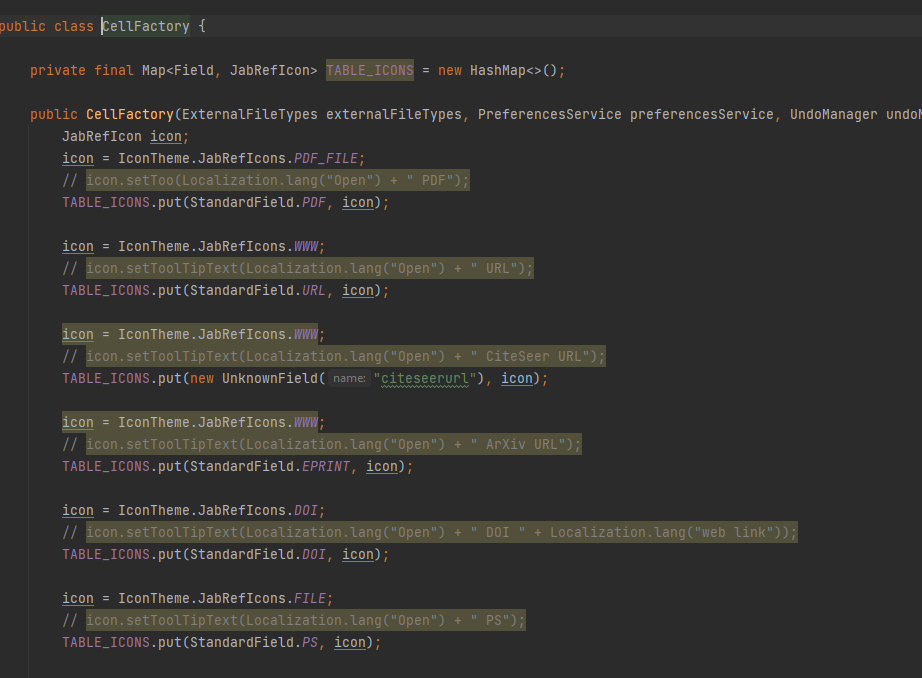
**DESIGN PATTERNS**

**The 3 Design Patterns I identified were the following:**

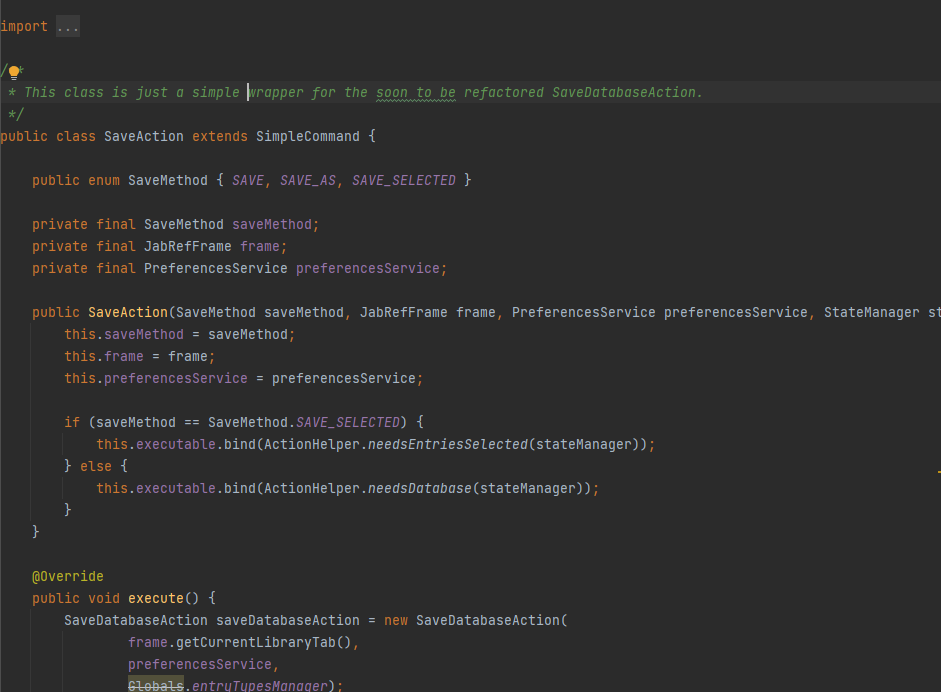
**FACTORY METHOD PATTERN**

****

**This “Factory Method” pattern can be found in jabref -> src -> main ->java -> org.jabref -> gui -> maintainable -> CellFactory.java**

**We can say that CellFactory.java uses a “Factory Method” pattern since it was designed for hiding the whole process of creating certain types of instances, such as icons, external file types or special fields, for example.**

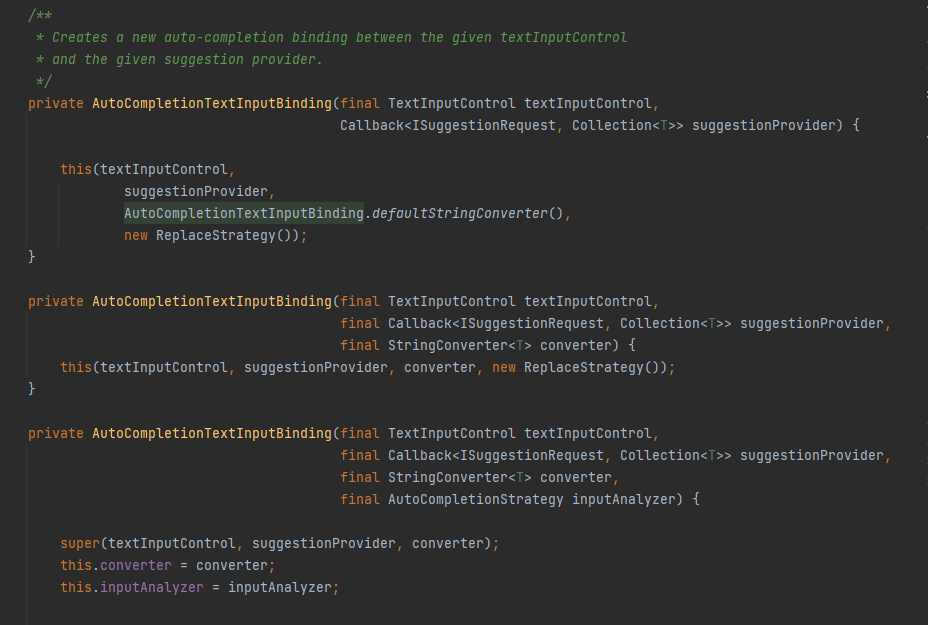
**FAÇADE PATTERN**



**This “Façade” pattern can be found in jabref -> src -> main -> java -> org.jabref -> gui -> exporter -> SaveAction.java**

**We can say that this class uses a “Façade” pattern since its whole purpose is to hide the complexity of a class (in this case, SaveDatabaseAction.java) and its communication with other classes (in this case, JabRefFrame.java or SaveMethod.java, for example) by providing a much simpler interface.**

**SINGLETON PATTERN**



**This “Singleton” pattern can be found in jabref -> src -> main -> java -> org.jabref->gui->autocompleter-> AutoCompletionTextInputBinding.java**

**By analysing this snippet of code, we can see that this class uses a “Singleton” pattern. We can conclude this from the use of the “private” keyword on these 3 constructors. Its purpose is to create 3 different and unique objects which all together will contribute to the auto-completion binding mechanism.**

**Pedro Simões, Nº 58674**