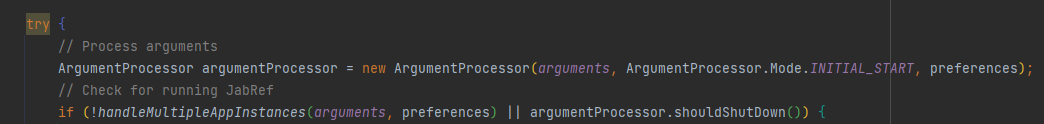
# Design Patterns

## Design Pattern #10- Facade Pattern

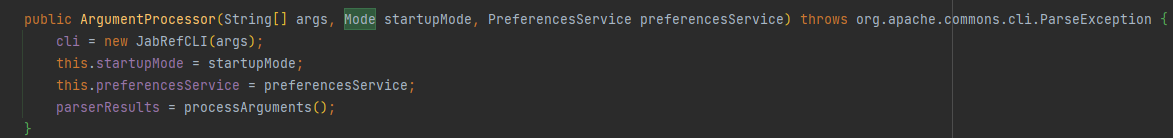
### Found:

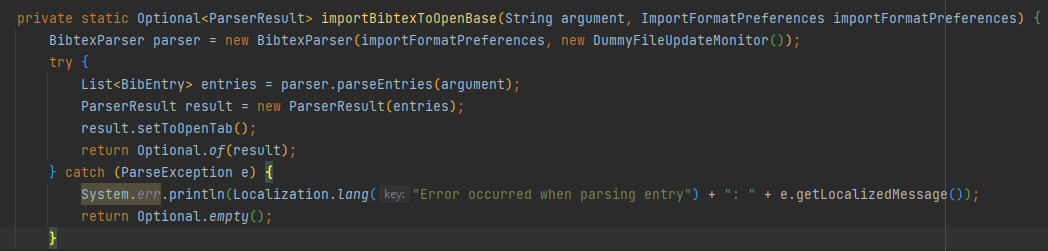
@ SE2122\_55677\_56773\_\_56971\_57066\_58655/src/main/java /org/jabref/cli/ArgumentProcessor.java

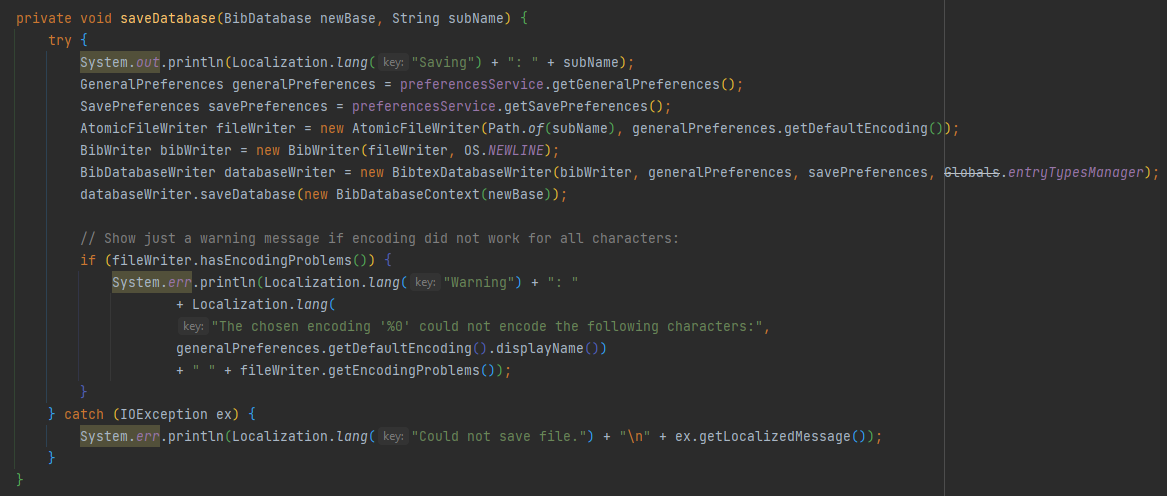
The class ArgumentProcessor is first instantiate in JabRefMain and I believe this class follows the Facade Design Pattern because it provides a convenient access for the client to interact with JabRef functionalities.



The ArgumentProcessor class is responsible for interacting with many different packages and classes (i.e.: gui, logic, model, preferences, PreferenceService.java, JabRefCLI.java, etc.). With its methods this class interacts with all the necessary objects to fulfill the client’s request, without forcing the client to interact with all the different objects. As we can see in the code below:







## Design Pattern #11- Builder Pattern

### Found:

@SE2122\_55677\_56773\_\_56971\_57066\_58655/src/main/java /org/jabref/gui/actions/ActionFactory.java

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

I believed this to be an implementation of The Build Pattern, as this class allows us to create of objects that implement the Action interface step by step, and customize it to suit our needs using the same construction code.

## Design Pattern #12- Singleton Pattern

### Found:

@@SE2122\_55677\_56773\_\_56971\_57066\_58655/src/main/java /org/jabref/gui/externalfiletype/ExternalFileTypes.java

Uma imagem com texto

Descrição gerada automaticamente

This is clear example o the Singleton design pattern, as it ensures that this class will only have one instance, while allowing us a public access point to this instance.