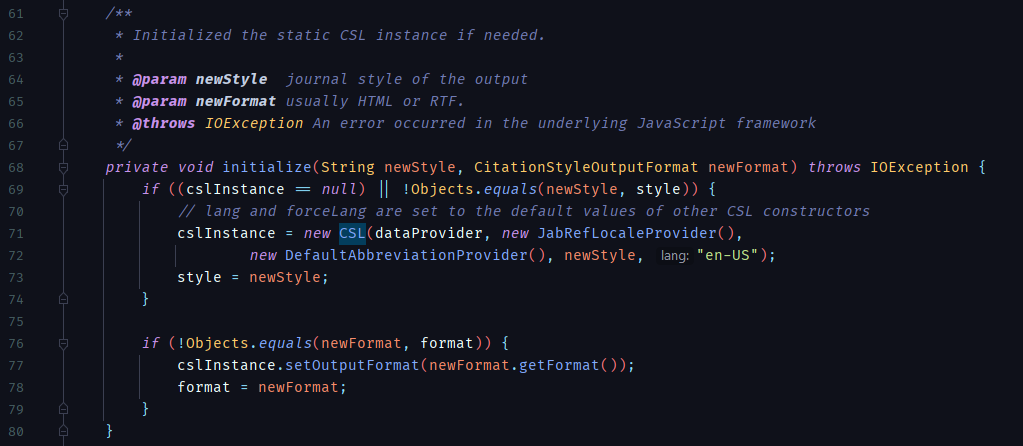
**Design Patterns**

# Adapter Pattern (Structural)

src/main/java/org/jabref/logic/citationstyle/CSLAdapter.java





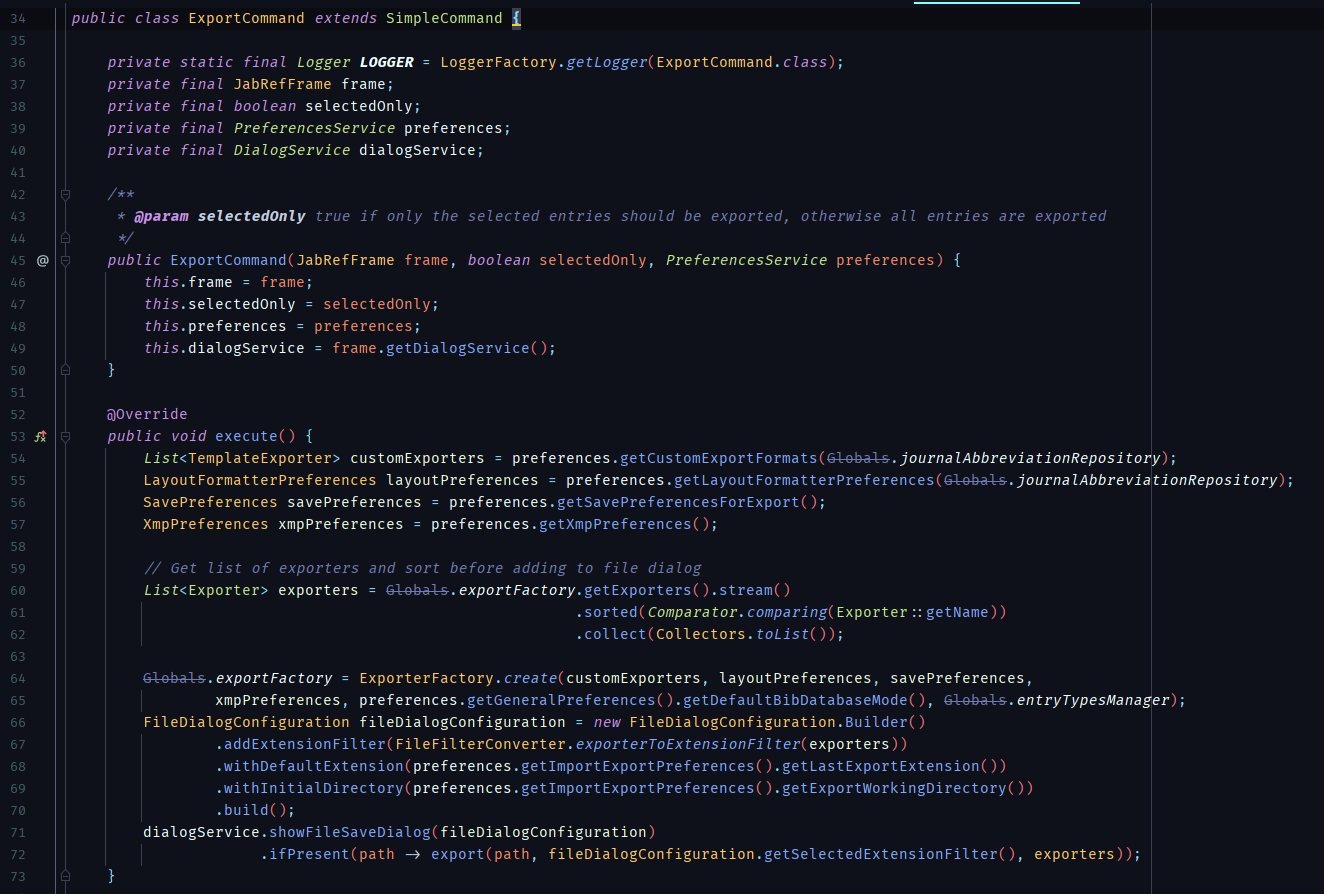
**Description:** The **CSLAdapter** class, as its name suggests, is a fairly standard implementation of the identified design pattern, as its only responsibility is to provide a conversion of the contents obtained from the instantiated **CSL** (Citation Style Language) object into, as it stands, a **List** of **Bibliography** Entries.

The main noteworthy difference between this class and the typical adapter implementation, is that the present class doesn’t incorporate a constructor which receives a **CSL** instance, instead, the **makeBibliography** method (which is the only public method in the class, seen in the first code snippet) uses a private **initialize** method with 2 arguments received from the aforementioned public method (as seen in the second code snippet).

Additionally, although not particularly connected to the design pattern itself, the class also implements its own private **JabRefItemDataProvider** class, which implements the **ItemDataProvider** interface required as an argument for the instantiation of the adapted **CSL** object.

# Command Pattern (Behavioral)

src/main/java/org/jabref/gui/exporter/ExportCommand.java



src/main/java/org/jabref/gui/importer/ImportCommand.java



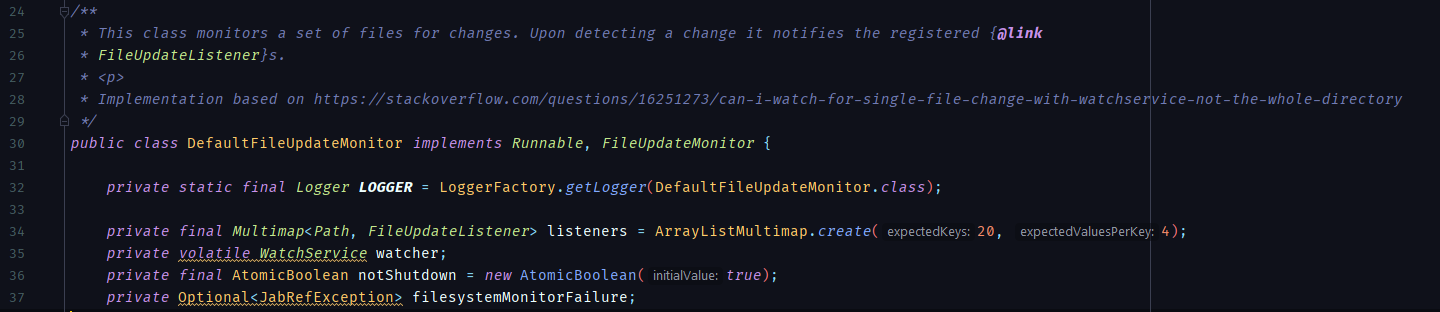
**Description:** The 2 classes specified above (**ExportCommand** and **ImportCommand**) are virtually perfect examples of the Command Pattern, since they are both extensions of the same **SimpleCommand** abstract class, although, this abstract class itself is an extension of the **CommandBase** abstract class that implements the base **Command** interface, the last 2 aforementioned being from a completely separate external library.

They both implement a standard **execute** method, which is called after the desired commands’ instantiation, which will contain all the information about the request in a stand-alone object.

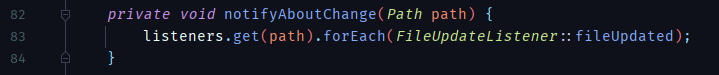
This pattern is especially useful for GUI elements (which happens to be the exact use here), since multiple ways to access similar actions don’t have to be tied down with their own implementation of the same operation. This enables extensibility by reusing or creating a new command and facilitates the implementation of other related functionalities, such as a command queue and undoing operations with the information stored inside each command.

# Observer Pattern (Behavioral)

src/main/java/org/jabref/gui/util/DefaultFileUpdateMonitor.java







**Description:** Standard implementation of the observer pattern. In this case, the **DefaultFileUpdateMonitor** class contains a map of **FileUpdateListener**(s), whose keys are their corresponding **Path**, which is used as an argument for the **notifyAboutChange** method (visible in the third code snippet and used in the **run** method in the second snippet), thus, only a handful of listeners with the specified **Path** will get updated.