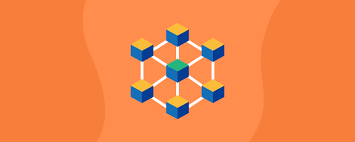
**Design Patterns**

****Tiago Fernandes 57677 (team\_member\_2)

# Factory method

There is na abstract class in org.jabref.logic.exporter package called Exporter which is a normal class with some data and functionalities.

This Exporter class works as the product where the ExporterFactory (in the same package) works as the product’s creator (factory). This factory class provides an interface which objective is to create Exporter objects, allowing its subclasses to change and deal with the object in a specific way.

# Composite

There is na interface in package org.jabref.logic.importer package called SearchBasedFetcher. This interface works as the component of the patterns and extends the WebFecther interface, located in the same package.



Working as the Leaf of this pattern we have the GrobidCitationFetcher in the package org.jabref.logic.importer.fetcher which implements the SearchBasedFecther interface, implementing both SearchBasedFecther and WebFetcher methods.



In the org.jabref.logic.importer.fetcher package we have a class called CompositeSearchBasedFetcher which implements the SearchBasedFetcher interface. Alike the Leaf class it also implements both SearchBasedFetcher and WebFetcher methods. This was identified as the composite class because it stores SearchBasedFetcher objects. This objects can be either leafs (GrobidCitationFecther) or even other composites (CompositeSearchBasedFetcher).



# Template method

There is an abstract class called BibDatabaseWriter () in the org.jabref.logic.exporter package.

This class has a method  that calls a template method 

This template method is responsible for calling the multiple methods in the class (steps of the algorithm) that are implemented in the same class or by another class if the methods are abstract. The class in the JabRef tool that implements these abstract methods is the BibtexDatabaseWriter (  ).

This class extends BibDatabaseWriter and belongs to the same package as its superclass.

This pattern is easily identified by looking at the template method in the abstract class, since it calls all steps that are implemented within the class or by its subclasses.