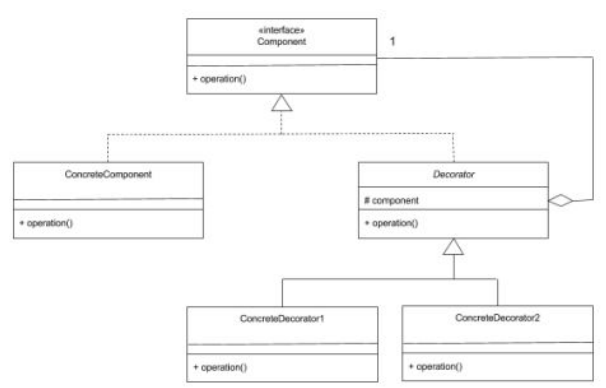
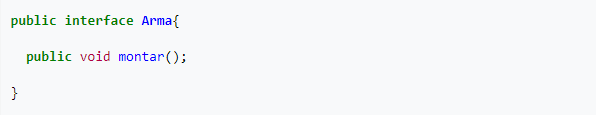
ES - Sprint 2 – Design Pattern (GOF) 3

**Decorator**

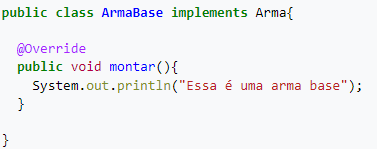
Allows additional behaviours or responsibilities to be dynamically attached to an object, using aggregation/composition to combine behaviours at run time. This design pattern makes use of interfaces and inheritance, so that the classes conform to a common type whose instances can be stacked in a compatible way. This builds a coherent combination of behaviour overall.

* **Structure**
* **Decorator java example (Code Snippet)**

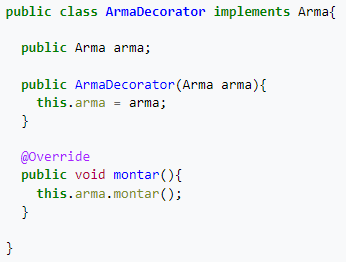
1. Component Interface



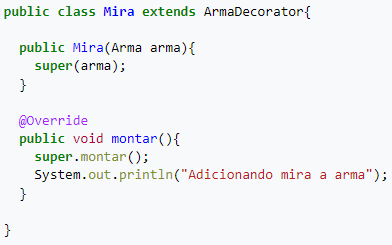
1. Concrete Component



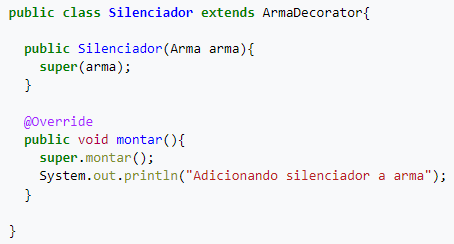
1. Decorator



1. Concrete Decorator 1



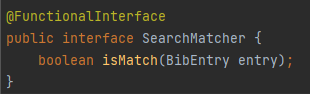
1. Concrete Decorator 2



**Code location:**

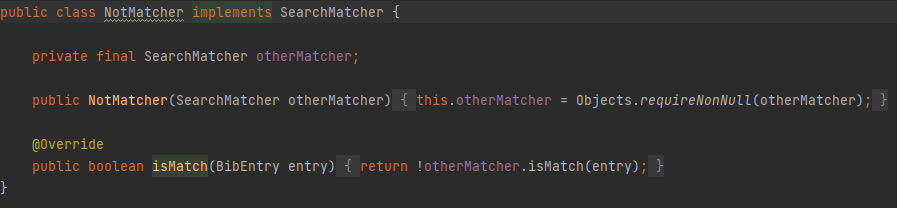
* **Component Interface**

“*src\main\java\org\jabref\model\search\SearchMatcher.java*”

****

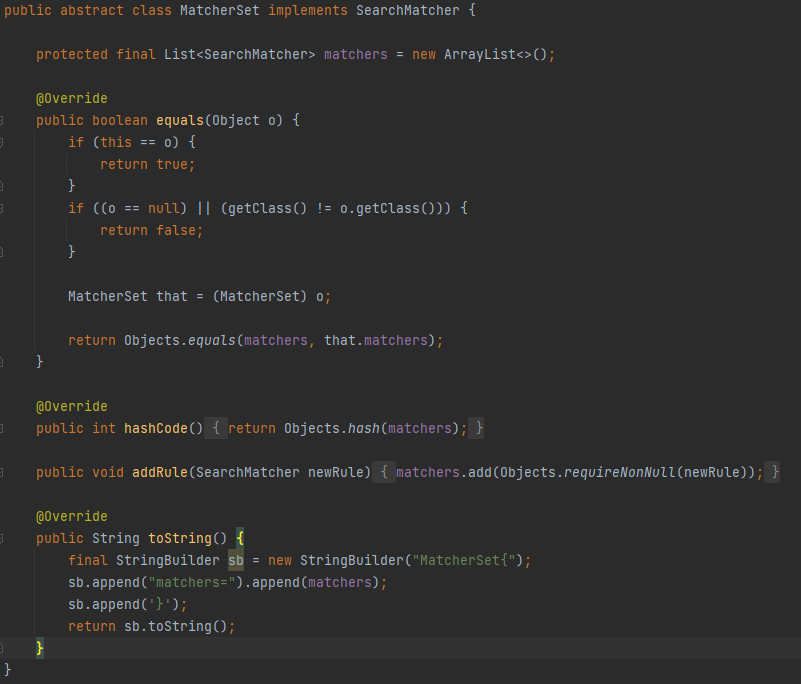
* **Concrete Decorator**

“*src\main\java\org\jabref\model\search\matchers\NotMatcher.java*”

**

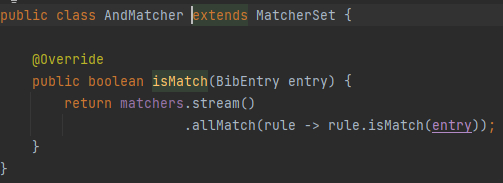
* **Decorator**

“*src\main\java\org\jabref\model\search\matchers\MatcherSet.java*”

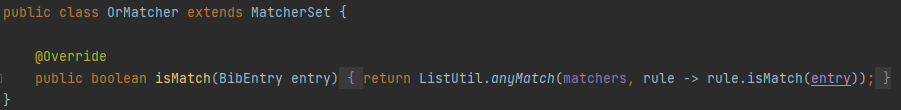
****

(matchers is the protected variable that stores a collection of type SearchMatcher (Component Interface) and the method isMatch (BibEntry entry) is implemented on the concrete decorator classes)

* **Concrete Decorator 1**

“*src\main\java\org\jabref\model\search\matchers\AndMatcher.java*”

* **Concrete Decorator 2**

****“*src\main\java\org\jabref\model\search\matchers\OrMatcher.java”*

**Justification:**

Like we saw above exist a protected variable, which type is the interface. The class Decorator is abstract, and all the classes that extends this one (Concrete Decorators) implement the method isMatcher (BibEntry entry).