## **Leticia Silva 60221**

### Code Patterns

1. Proxy Pattern

public class ReadOnlyProxyDocument implements Document {  
  
 private final Document myDelegate;  
  
 public ReadOnlyProxyDocument(Document delegate) {  
 myDelegate = delegate;  
 }  
  
 @Override  
 public String getFileName() {  
 return myDelegate.getFileName();  
 }  
  
 @Override  
 public boolean canRead() {  
 return myDelegate.canRead();  
 }

Location:

ganttproject/src/main/java/net/sourceforge/ganttproject/document/ReadOnlyProxyDocument.java

This proxy class is something that acts as a simplified version of the original object, providing controlled access of a functionality.

2. Template Pattern

abstract class SearchServiceBase<SR extends SearchResult<SO>, SO> implements SearchService<SR, SO>

public class ResourceSearchService extends SearchServiceBase<ResourceSearchService.MySearchResult, HumanResource>

public class TaskSearchService extends SearchServiceBase<TaskSearchService.MySearchResult, Task>

Location:

ganttproject/src/main/java/net/sourceforge/ganttproject/search/ResourceSearchService.java

The two classes differ on the responsibilities but have very similar functionality and order of operations, so the abstract class SearchServiceBase makes the code understandable.

1. Factory Pattern

public interface ParserFactory {  
 GPParser newParser();  
  
 GPSaver newSaver();  
}

Location:

ganttproject/src/main/java/net/sourceforge/ganttproject/parser/ParserFactory.java

The interface ParserFactory hides the creation code of the objects GPParser and GPSaver, so the creation process does not depend on concrete classes.