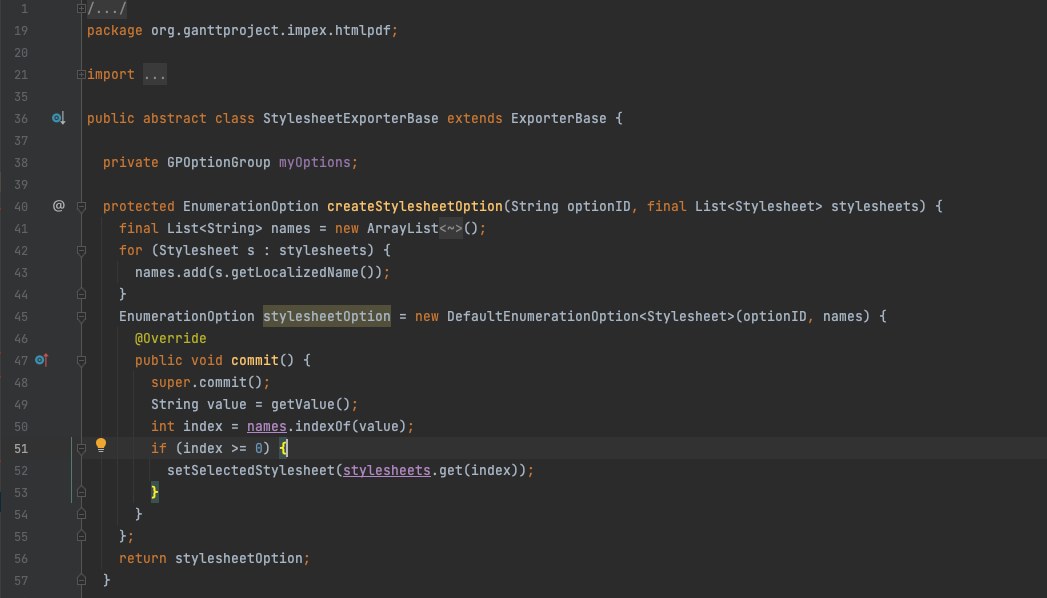
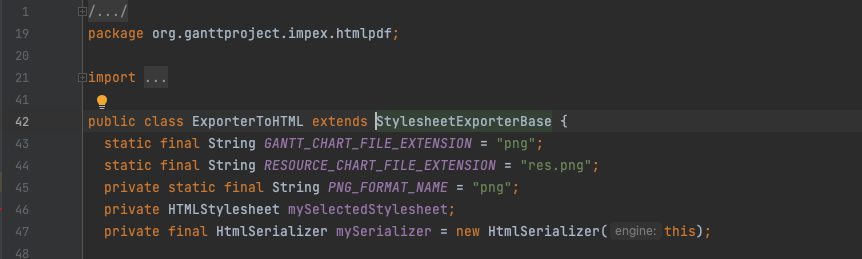
Design Pattern 1: Template Method Pattern

SuperClass:



SubClasses:





Locations:



Synopsis: Defines an algorithm’s steps generally, deferring the implementation of some steps to subclasses. In other words, it is concerned with the assignment of responsibilities and it makes sense to be used when we have two separated classes with really similar functionalities.

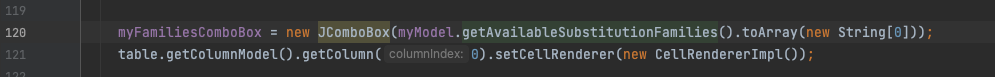
Rationale: We have StylesheetExporterBase as the SuperClass and ExporterToHTML and ExporterToPDF being the SubClasses. Both of the SubClasses implement the methods that are not implemented in the Abstract Class and since they are similar Classes it just made sense to have a SuperClass that centralises some of the similar functionalities.

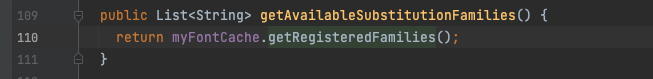
Design Pattern 2: Chain of Responsibility Design Pattern

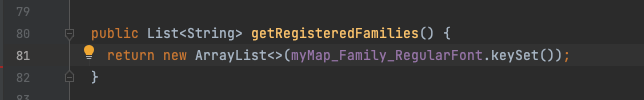
ITextEngine:



FontSubstitutionPanel:

(Inside getComponent method in line 42)

FontSubstitutionModel:

TTFontCache:

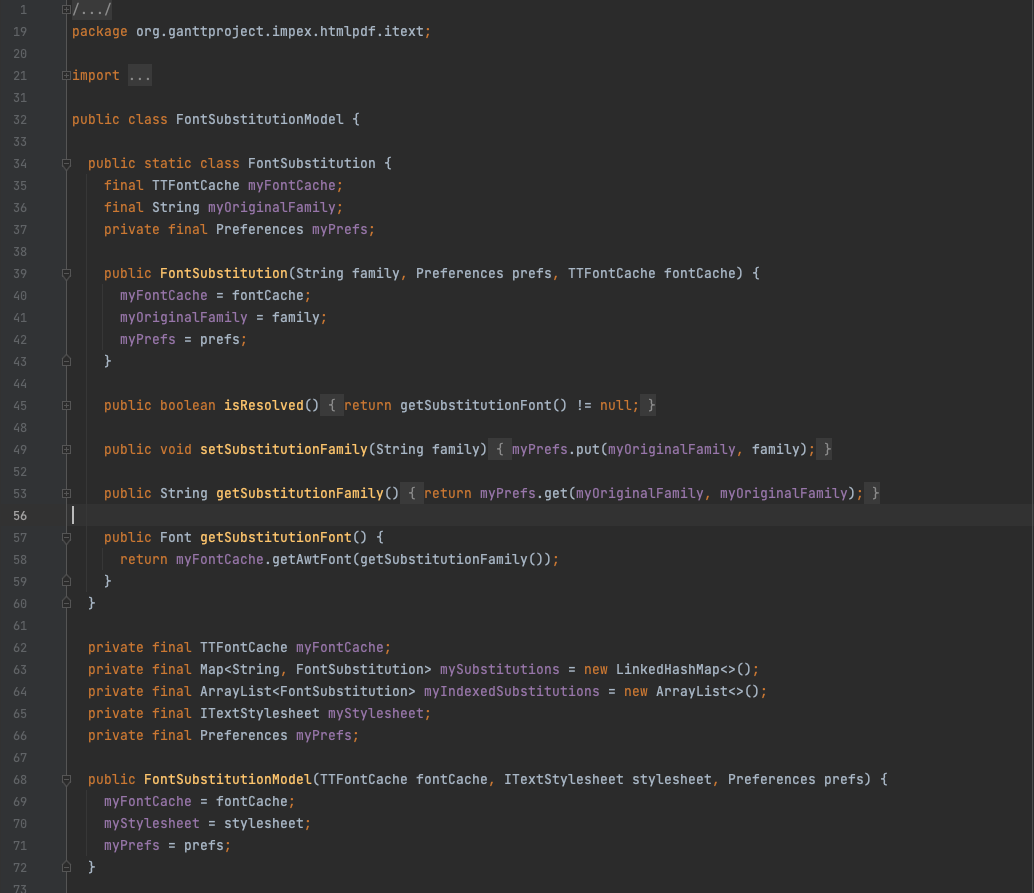
Location: 

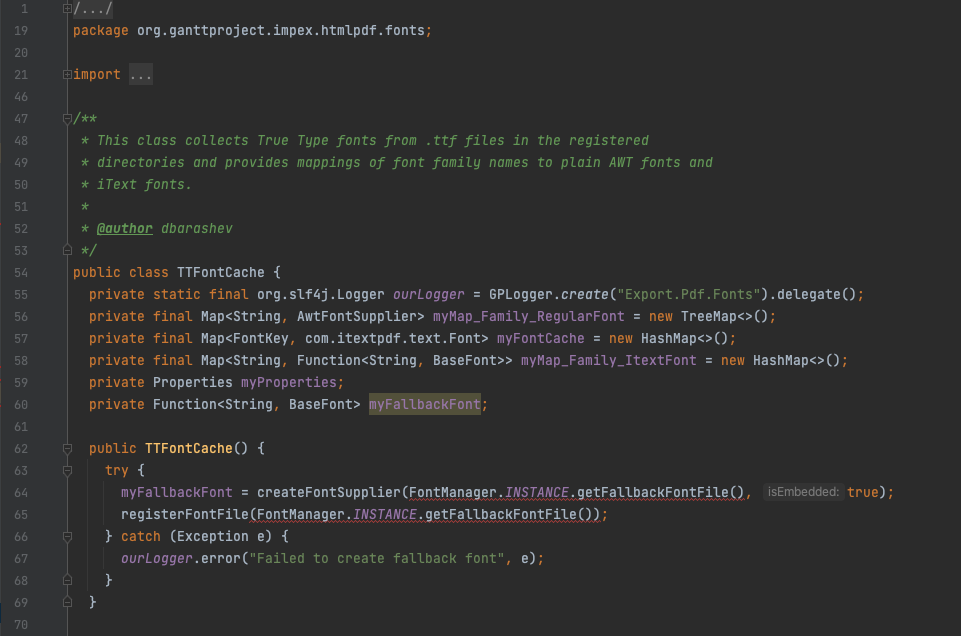
Synopsis: This pattern is used when a client sends a request and then this request is propagates until a class is able to handle it.

Rationale: As we can see in the images above, this classes propagate the first request which was the createFontPanel() in line 87. The request is being propagated until it reaches a class that can handle it, TTFontCache.

Design Pattern 3: Visitor Design Pattern

Visitor Class:



One of the Composite Class:

Location:



Synopsis: Allows you to add operations to a Composite structure without changing the structure itself, and so you can check the state of the composite without changing it.

Rationale: We have the Visitor class being FontSubstituitonModel, which has some methods that can check and change the state of the Composite in TTFontCache, such as 49, 53, 57.