ES - Report 1

Design Patterns & Code Smells



Made by:

Bruno Melo - 60019

Duarte Cruz - 59765

João Pereira - 60180

Neel Badracim - 60492

Ricardo Bessa - 60485

Index

[**Objectives**](#_fn95p6vlhq7l)5

[**Constraints**](#_g6s418xlxiyb)5

[**Bruno Melo**](#_o7uh9jno1n19)6

[Design Patterns](#_8nch8c9n2gu9) 6

[Pattern 1: Memento Pattern](#_pvj97umc7yp9) 6

[Pattern 2: Decorator Pattern](#_b94ro4o59xd1) 7

[Pattern 3: Factory Method Pattern](#_kj6p7ibl5806) 8

[Code Smells](#_d8pi3uhem7ao) 9

[Code Smell 1: Dead Code](#_he88bemrvaiu) 9

[Code Smell 2: Data Class](#_5sbhu7rkt655) 10

[Code Smell 3: Dead Code](#_uf0u2kgccyxt) 11

[**Duarte Cruz**](#_lzn85lgxvky5)12

[Design Patterns](#_ifcnu2acghho) 12

[Pattern 1: Chain of Responsibility Pattern(Behavioral)](#_tqyqhpk7ekkd) 12

[Pattern 2: Builder Pattern(creational)](#_h5zm8e9x5qk3) 13

[Pattern 3: Iterator](#_wxlk8n2up8c0) 14

[Code Smells](#_dvyy9wpt5650) 15

[Code Smell 1: Comments and Dead Code](#_4b8mnbd8aanf) 15

[Code Smell 2: Shotgun surgery](#_fwhiys8n0coo) 16

[Code Smell 3: Dead code and Long parameter list](#_dcqq5aq5v67c) 17

[**João Pereira**](#_hkhgb759dydp)18

[Design Patterns](#_7ixhurpxzfg7) 18

[Pattern 1: Decorator Pattern](#_fc5ds3y33mme) 18

[Pattern 2: Façade Pattern](#_d11f3eq3faj7) 18

[Pattern 3: Factory Method Pattern](#_qk8m21mxay7e) 19

[Code Smells](#_uumkznmua5yh) 20

[Code Smell 1: Data Class](#_t4updv8kc108) 20

[Code Smell 2: Comments and Dead Code](#_8ywigdceq55h) 20

[Code Smell 3: Shotgun Surgery](#_h35bqqexpbl0) 21

[**Neel Badracim**](#_976g2kqt021h)22

[Design Patterns](#_fdvy7rw88ssi) 22

[Pattern 1: Proxy Pattern (Structural Design Pattern)](#_z8z8dtukgptz) 22

[Pattern 2: Iterator](#_vj8sivrh53v2) 22

[Pattern 3: Singleton Method (Creational)](#_gyco82qf38v0) 23

[Code Smells](#_fyr9zwybs3c9) 24

[Code Smell 1: Data Class](#_wr5gbatxhcph) 24

[Code Smell 2: Dead Code](#_wssr7p4oppa) 24

[Code Smell 3: Comments and Speculative Generality](#_1qmwvbkfroua) 25

[**Ricardo Bessa**](#_45z55vsw805d)26

[Design Patterns](#_ksisuza2mikh) 26

[Pattern 1: Facade Pattern (Structural Design Pattern)](#_5pipm5xn4d8s) 26

[Pattern 2: Singleton Pattern (Creational Design Pattern)](#_xhqv3y2y4u2d) 27

[Pattern 3: Abstract Factory Pattern (Creational)](#_4eqpyvfmkskx) 28

[Code Smells](#_xicon3ejgu6h) 29

[Code Smell 1: Comments](#_6kxalzes97f5) 29

[Code Smell 2: Dead Code](#_qm6ebix1j9zn) 30

[Code Smell 3: Comments and Speculative generality](#_gu5659kstl51) 31

# Objectives

The objective of this project is to identify design patterns discussed in class. For that, our identification will include:

○ Illustrating code snippet

○ The exact location on the codebase

○ An explanation of the rationale for identifying this as a pattern instantiation.

# Constraints

Each team member will be responsible for identifying three different design patterns. Design patterns may overlap from one team member to another, as long as they correspond to different locations in the code. Each team member should also review three other design patterns identified by the other team members.

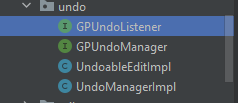
# Bruno Melo

## Design Patterns

### Pattern 1: Memento Pattern

**Location of the pattern:**

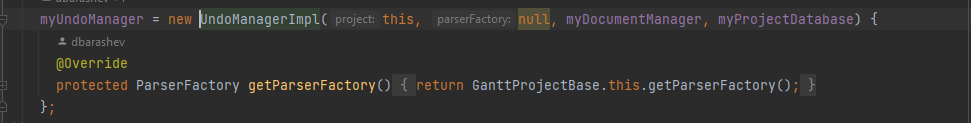
ganttproject/src/main/java/net/sourceforge/ganttproject/undo:



ganttproject/src/main/java/net/sourceforge/ganttproject/undo/UndoableEditImpl.java:



ganttproject/src/main/java/net/sourceforge/ganttproject/GanttProjectBase.java:



**Explanation:**

This set of classes and interfaces all work together in order to give the user the possibility of undoing edits in the Gantt Project. Undoing an edit is the same as returning the Gantt Chart object to one of its previous states. This is a characteristic of the Memento Pattern.

Review 1: This design pattern has been reviewed and approved by Ricardo Bessa.

### Pattern 2: Decorator Pattern

**Location of the pattern:**

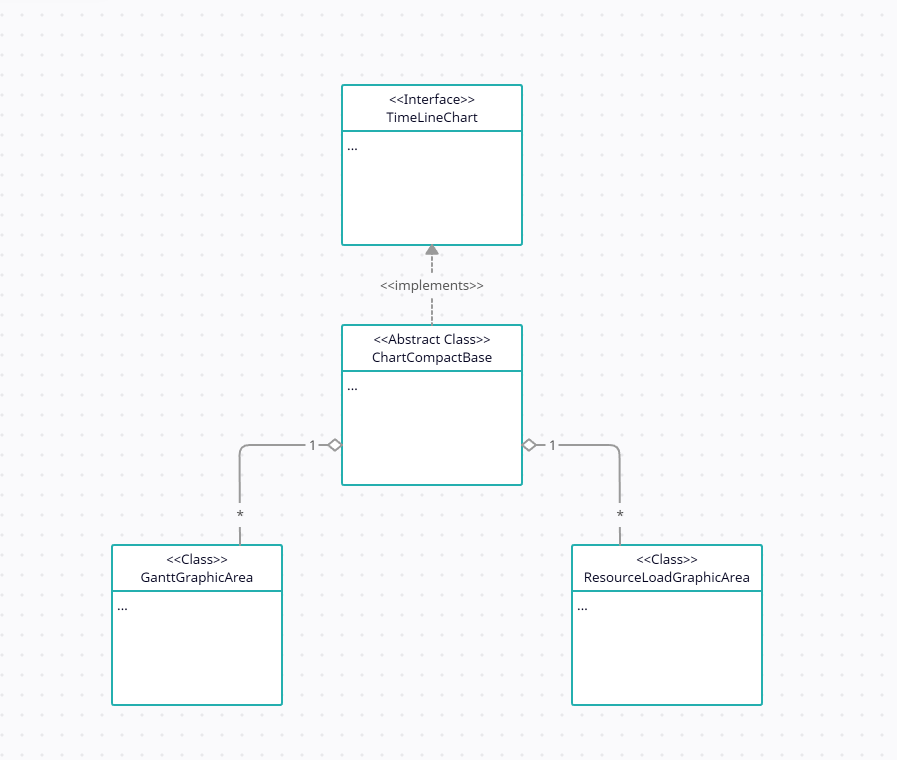
ganttproject/src/main/java/net/sourceforge/ganttproject/ChartComponentBase.java

ganttproject/src/main/java/net/sourceforge/ganttproject/chart/TimelineChart.java

ganttproject/src/main/java/net/sourceforge/ganttproject/GanttGraphicArea.java

ganttproject/src/main/java/net/sourceforge/ganttproject/ResourceLoadGraphicArea.java

**Explanation:**

****

Decorator pattern allows a user to add new functionality to an existing object without altering its structure. This type of design pattern comes under structural pattern as this pattern acts as a wrapper to existing class.

This pattern creates a decorator class which wraps the original class and provides additional functionality keeping class methods signature intact.

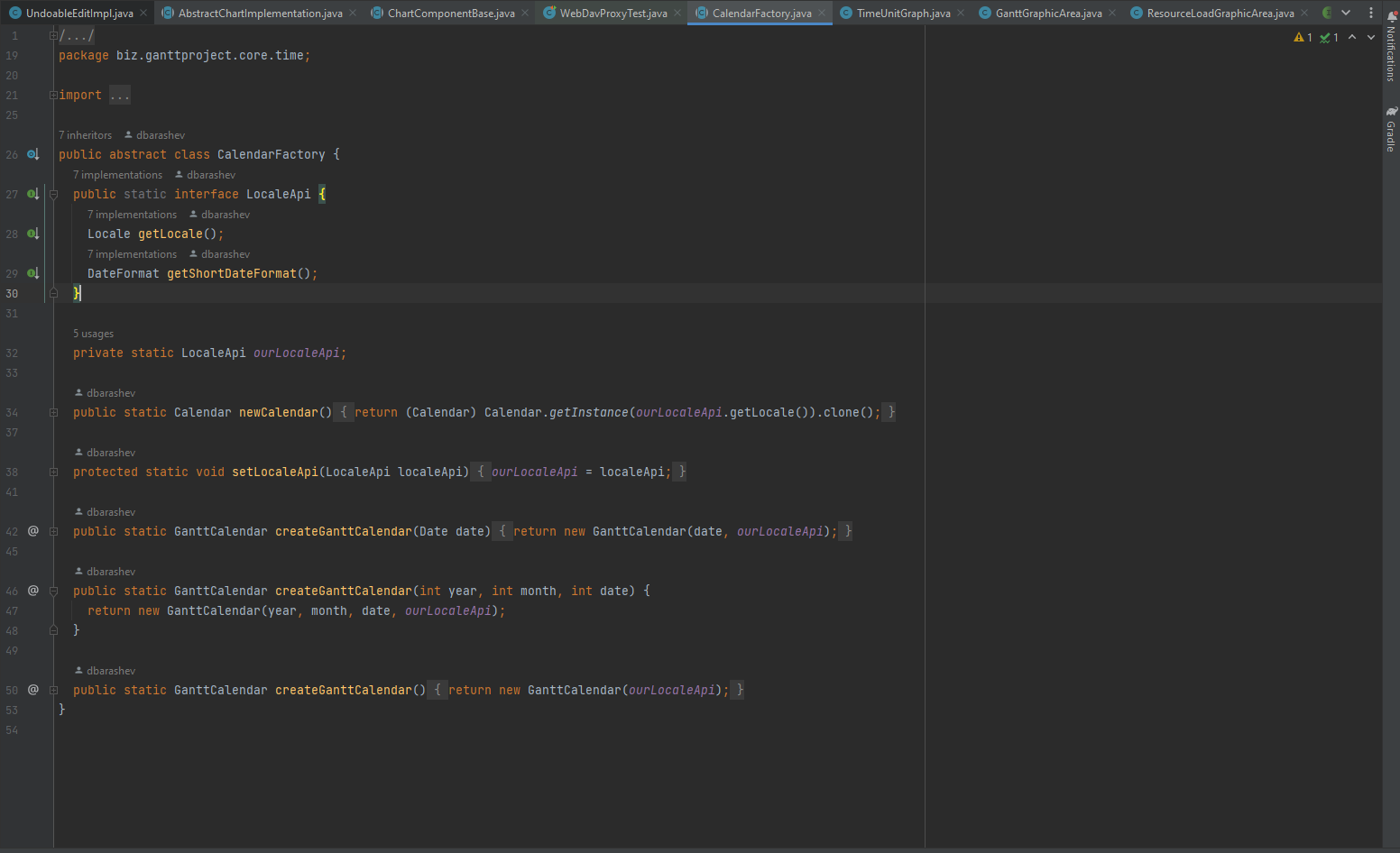
The ChartComponentBase class is decorated/wrapped by the GanttFraphicArea and ResourseLoadGraphicArea classes which both expand on the decorator class (ChartComponentBase) without altering its implementation.

Review 2

### Pattern 3: Factory Method Pattern

**Location of the pattern:**

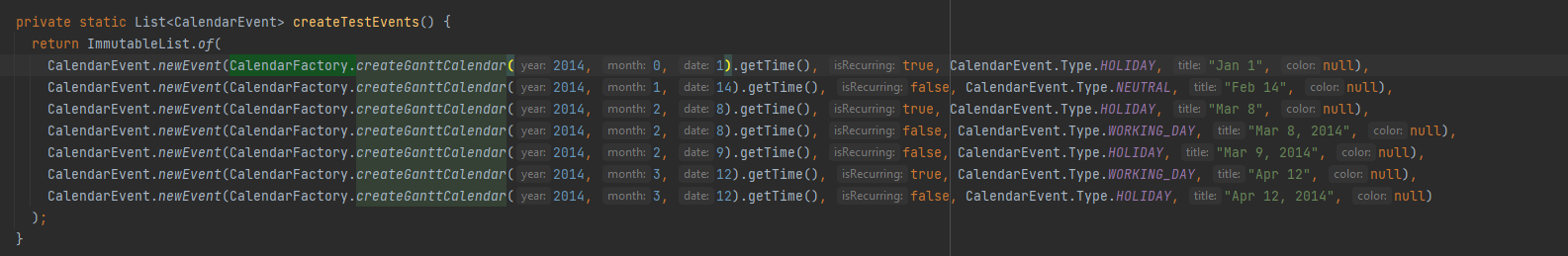
biz.ganttproject.core/src/main/java/biz/ganttproject/core/time/CalendarFactory.java



**Explanation:**

In the Factory pattern, we create objects without exposing the creation logic to the client and refer to newly created objects using a common interface.

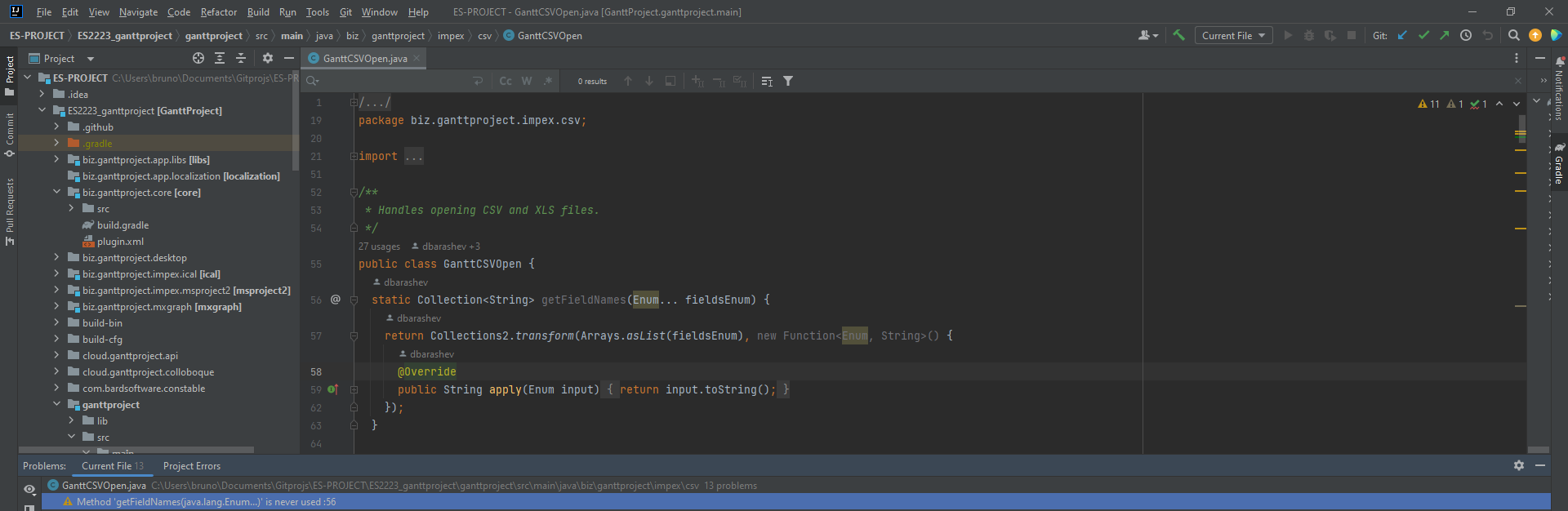
This is exactly what the CalendarFactory class does. It creates calendar objects without exposing its creation logic to the user.



Review 3

## Code Smells

### Code Smell 1: Dead Code



**Location of the code smell from repository root:**

ganttproject/src/main/java/biz/ganttproject/impex/csv/GanttCSVOpen.java, line 56.

**Explanation:**

The method getFieldNames(Enum... fieldsEnum) is never used, also known as dead code.

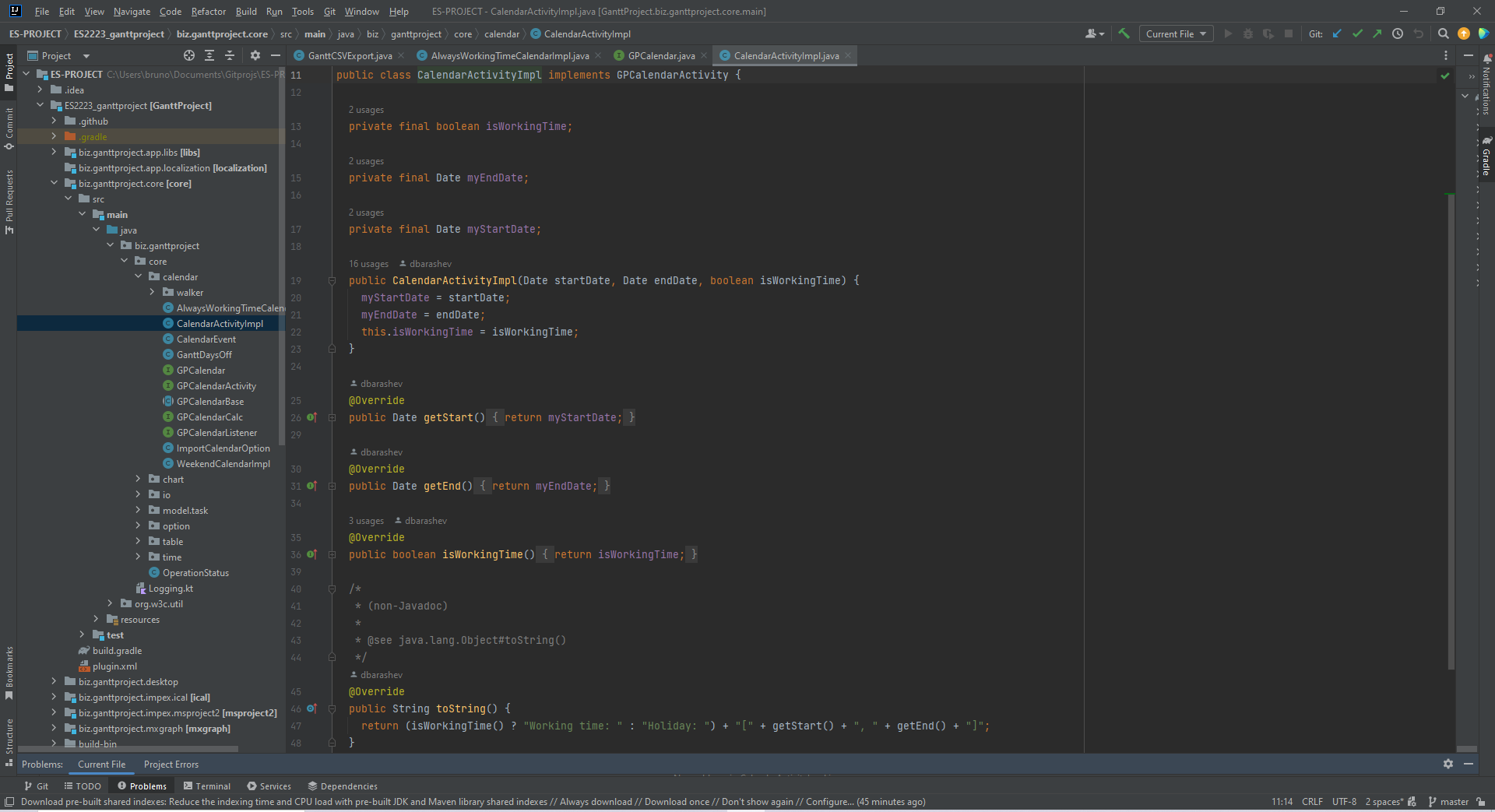
**Refactoring proposal:**

This method should be deleted.

Review 1-

This code smell was viewed and approved by Duarte

### Code Smell 2: Data Class



**Location of the code smell from repository root:**

biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/CalendarActivityImpl.java

**Explanation:**

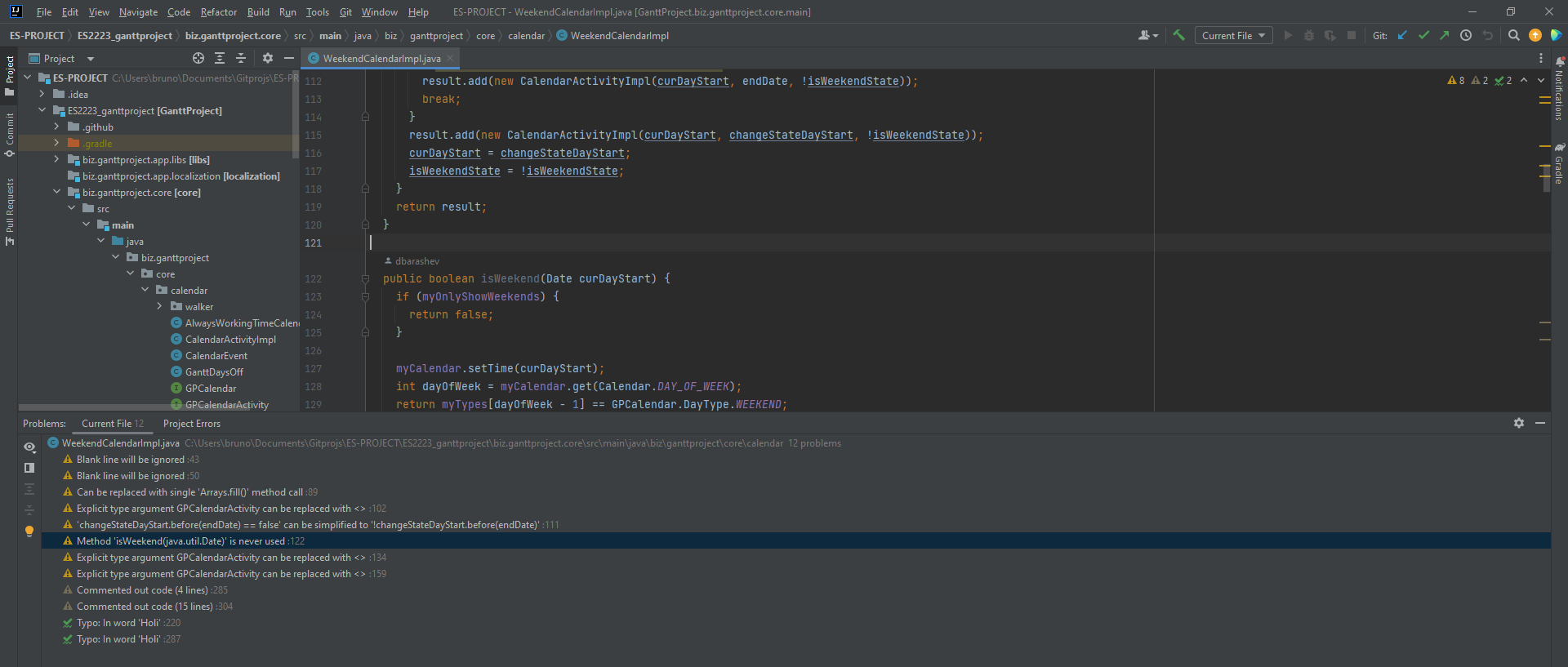
The CalendarActivityImpl.java class is too small and is only used to store, having no real functionality. This is a code smell known as Data Class.

**Refactoring proposal:**

This class may not be needed if its only functionality is to store data. Alternatively, some functionality could be added to it in order to make it a necessary class.

Review 2: This code smell has been reviewed and approved by Ricardo Bessa.

### Code Smell 3: Dead Code



**Location of the code smell from repository root:**

biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/WeekendCalendarImpl.java , line number 122.

**Explanation:**

The method isWeekend(Date curDayStart) is never used, also known as dead code.

**Refactoring proposal:**

This method should be deleted.

Review 3

This code pattern was viewed and approved by Neel.

# 

# Duarte Cruz

## Design Patterns

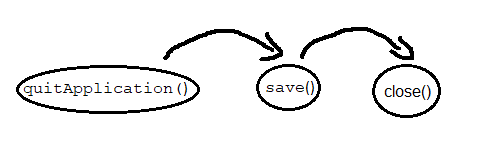
### Pattern 1: Chain of Responsibility Pattern(Behavioral)

**Localization**:

ganttproject\src\main\java\net\sourceforge\ganttproject\ganttproject.java, line 595 -> 604

ganttproject\src\main\java\net\sourceforge\ganttproject\ganttoprtions.java, line 196 -> 204

java/io/FilterOutputStream.java , line 196



**Explanation**:

The chain of responsibility pattern allows the client object to send a request, the first handler in the chain will try to process it. If the handler can process the request, then the request ends with this handler. However, if the handler cannot handle the request, then the

request is sent to the next handler in the chain. This process will continue until a handler can process the request. If the entire chain is unable to handle the request, then the request is not satisfied.

In this example, the code is trying to quit the application but before that he needs to save the data, for that he calls the function save(), however the function save() needs to close the File Writer, so she calls the function close(). If any of this requests fail, the application will not end properly

Review 1

This design pattern has been reviewed and approved by Bruno Melo,

### 

### 

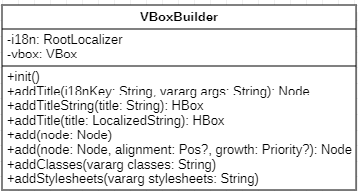
### Pattern 2: Builder Pattern(creational)

**Localization**:

ganttproject\src\main\java\biz\ganttproject\lib\fx\Components.kt, line 88

**Used in**:

ganttproject\src\main\java\biz\ganttproject\storage\cloud\GPCloudSignupPane.kt, line 54



**Explanation**:

A Builder is a creational design pattern that lets the user construct objectives in a more personalized way. This is possible because this pattern separates the code the normally would be in the constructor method in to other “sub” methods

In this example, we have a VBoxBuilder that allows the creation of a personalized panel for the SignUp panel of the GanttProject Cloud

Review 2: This design pattern has been reviewed and approved by Ricardo Bessa.

### 

### Pattern 3: Iterator

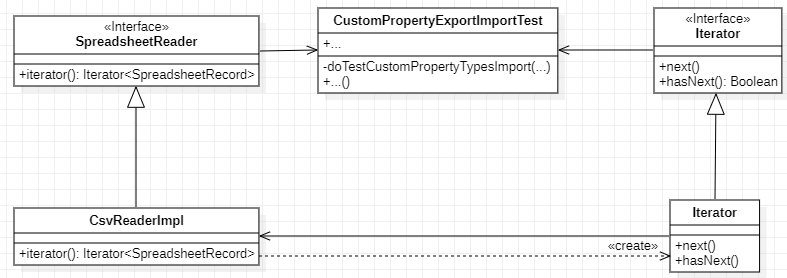
**Localization**:

ganttproject\src\main\java\biz\ganttproject\impex\csv\CSVImport.kt,line55

ganttproject\src\main\java\biz\ganttproject\impex\csv\Spreadsheet.kt

**Usase**:

ganttproject-tester\test\biz\ganttproject\impex\csv, line 107



**Explanation**:

An Iterator provides a way to traverse a collection of objects without exposing its implementation.

In this example, we have an Iterator that allow us to navigate through all the Spreadsheet Records without exposing its implementation to the main system

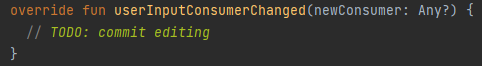
Review 3

This code pattern was viewed and approved by Neel.

## 

## Code Smells

### Code Smell 1: Comments and Dead Code



**Location of the code smell from repository root:**

ganttproject\src\main\java\biz\ganttproject\ganttview, line number 433.

**Explanation:**

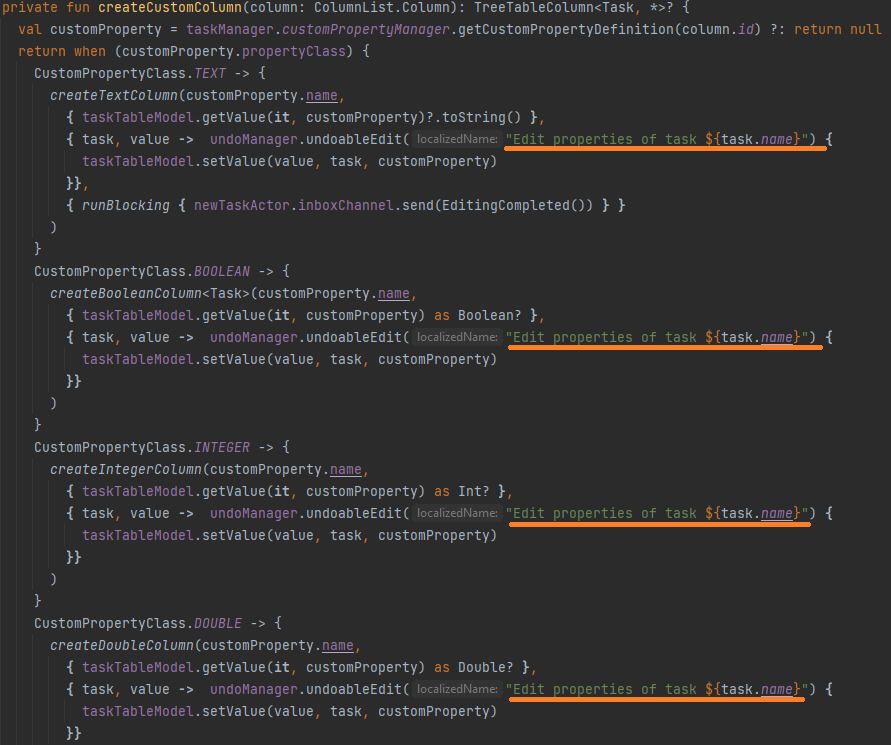
This function is currently doing nothing and has a reminder that some work will be done here in the future

**Refactoring proposal:**

Remove this function

Review 1

### Code Smell 2: Shotgun surgery



**Location of the code smell from repository root:**

ganttproject\src\main\java\biz\ganttproject\ganttview, line number 566.

**Explanation:**

This function has a lot of repeated code, if for example, we wanted to change the message we would need to change it individually for each line

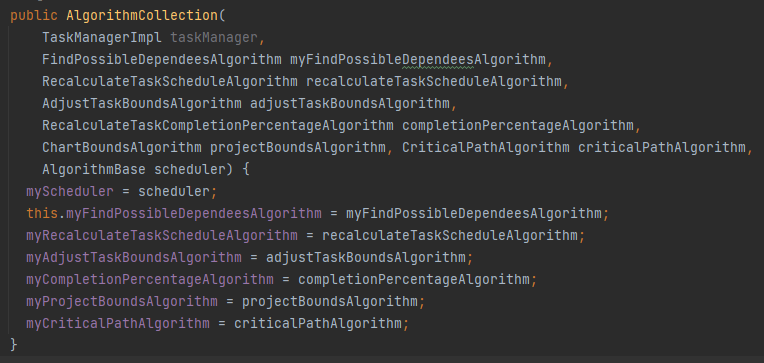
**Refactoring proposal:**

Create a constant for the message and when showing the message use that constant instead of the whole string.

Review 2

This code smell has been reviewed and approved by Bruno Melo.

### Code Smell 3: Dead code and Long parameter list



**Location of the code smell from repository root:**

ganttproject\src\main\java\net\sourceforge\ganttproject\task\algorithm

**Explanation:**

This function has a dead variable(“taskManager”) and has a pretty long parameter list.

**Refactoring proposal:**

Remove the variable taskManager from the parameter list and create parameter objects

Review 3: This code smell has been reviewed and approved by Ricardo Bessa.

# 

# João Pereira

## Design Patterns

### Pattern 1: Decorator Pattern

**Location of the pattern:**

\ganttproject\src\main\java\net\sourceforge\ganttproject\export\Exporter

\ganttproject\src\main\java\net\sourceforge\ganttproject\export\ExporterBase

\ganttproject\src\main\java\net\sourceforge\ganttproject\export\ExporterToCSV

\ganttproject\src\main\java\net\sourceforge\ganttproject\export\ExporterToImage

**Explanation:**

The decorator pattern allows to dynamically attach additional behaviors to an object, making use of interfaces and inheritance, building a coherent combination of behavior overall.

Here, the decorator is the abstract class ExporterBase.java that implements the interface Exporter.java, and the classes ExporterToCSV.java and ExporterToImage are expanded from it.

Review 1

This design pattern was viewed and reviewed by Duarte

### Pattern 2: Façade Pattern

**Location:**

\ganttproject\src\main\java\net\sourceforge\ganttproject\task\TaskContainmentHierarchyFacade

\ganttproject\src\main\java\net\sourceforge\ganttproject\task\TaskTreeFacade.kt\FacadeImpl

**Explanation:**

This façade pattern has the interface TaskContainmentHierarchyFacade.java which is used to hide the complexity, being implemented by FacadeImpl.

Review 2

### 

### 

### 

### Pattern 3: Factory Method Pattern

**Location:**

\ganttproject\src\main\java\net\sourceforge\ganttproject\action\task\OutdentTargetFunctionFactory

\ganttproject-tester\test\net\sourceforge\ganttproject\action\task\TaskMoveEnabledPredicateTest

**Explanation:**

This factory method pattern is used to hide the creation of objects. In this case, the OutdentTargetFunctionFactory is responsible for creating an object that is used in TaskMoveEnabledPredicateTest.

Review 3: This design pattern has been reviewed and approved by Ricardo Bessa.

## 

## Code Smells

### Code Smell 1: Data Class

**Location of the code smell from repository root:**

\ganttproject\src\main\java\net\sourceforge\ganttproject\resource\ResourceEvent

**Explanation:**

This is a Data Class because it only has getter methods and data. It doesn’t have any functionalities.

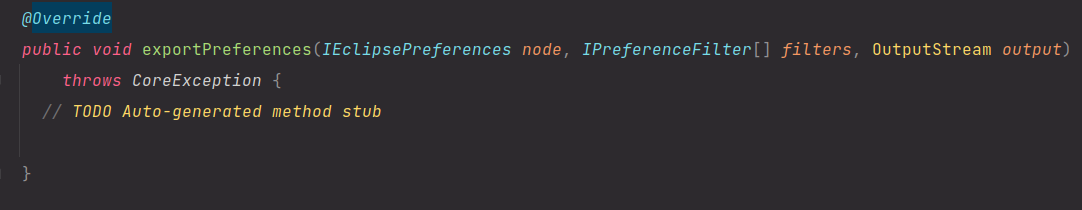
**Refactoring proposal:**

A solution might be implementing the methods used by the HumanResourceManager class to interact with the HumanResource class.

Review 1

This code pattern was viewed and approved by Neel.

### Code Smell 2: Comments and Dead Code



**Location of the code smell from repository root:**

\ganttproject\src\main\java\net\sourceforge\ganttproject\PreferenceServiceImpl

**Explanation:**

This method is dead code because it’s not being used and it also has a comment to remind of some unfinished work.

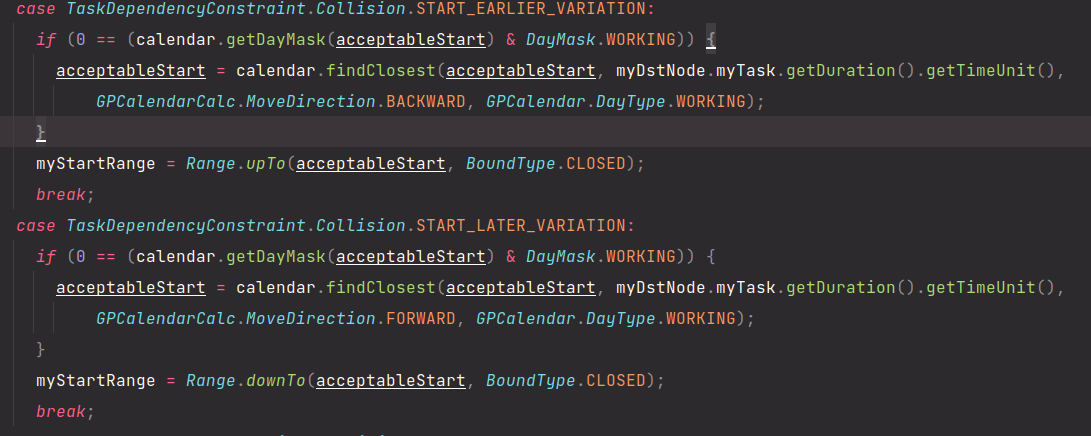
**Refactoring proposal:**

Since the code is not being used, a simple solution is to remove it from the code.

Review 2

This code smell was viewed and approved by Duarte

### Code Smell 3: Shotgun Surgery



**Location of the code smell from repository root:**

\ganttproject\src\main\java\net\sourceforge\ganttproject\task\algorithm\DependecyGraph\ExplicitdependecyImpl

**Explanation:**

This method has repeated code and if we want to change some of this code we need to change everywhere it’s used.

**Refactoring Proposal:**

A solution is to create methods to put this code and that way if we want to change the code we just need to change the method.

Review 3

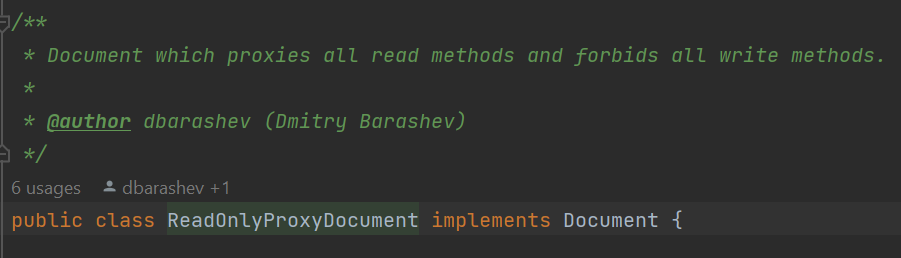
This code smell has been reviewed and approved by Bruno Melo.

# 

# Neel Badracim

## Design Patterns

### Pattern 1: Proxy Pattern (Structural Design Pattern)



**Location of the design pattern from repository root:**

* \ganttproject\src\main\java\net\sourceforge\ganttproject\document\ReadOnlyProxyDocument.java

**Explanation:**

We can find a Proxy Pattern in the ReadOnlyProxyDocument class. This class acts as a simplified version of the ProxyDocument class since it performs some methods as the “original” class but is restricted from anothers. In this case the ReadOnlyProxyDocument class only handles the read methods and rules out the write ones, as indicated by the programmer.

Review 1

### Pattern 2: Iterator



**Location of the design pattern from repository root:**

* ganttproject\src\main\java\net\sourceforge\ganttproject\ResourceTreeTableModel.java

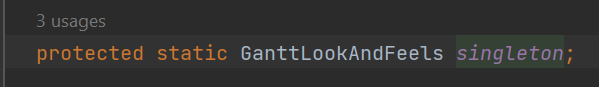
**Explanation:**

In this case, we have an Iterator that allows us to navigate through all the elements of type HumanResource without having to expose its high complexity. The iterator provides simple methods for us to work with and helps to keep our code simple and secure.

Review 2

This code pattern was viewed and reviewed by Duarte

### Pattern 3: Singleton Method (Creational)



****

****

**Location of the design pattern from repository root:**

* ganttproject\src\main\java\net\sourceforge\ganttproject\gui\GanttLookAndFeels.java

**Explanation:**

This is a Singleton design pattern. The class has a protected constructor and a protected static instance of this class as displayed above. This instance is only accessed through the method GanttLookAndFeels() and this method certificates that there’s a single instance of this class because it makes a new one only if this instance is null, otherwise returns the previous created one.

Review 3

## 

## Code Smells

### Code Smell 1: Data Class

**Location of the code smell from repository root:**

* ganttproject\src\main\java\net\sourceforge\ganttproject\client\RssUpdate.java.

**Explanation:**

This is a Data Class since it only has getter methods which means it only contains data and no real functionality. According to lecture 8, this indicates that it may not be an essential class or a good abstraction.

**Refactoring proposal:**

To change the situation, a possible solution would be to implement methods that use the class data in a suitable way, by adding some relevant functions other than getter and setter methods.

Review 1: This code smell has been reviewed and approved by Ricardo Bessa.

### Code Smell 2: Dead Code

**Location of the code smell from repository root:**

* ganttproject\src\main\java\net\sourceforge\ganttproject\chart\ChartModelImpl.java

Line 53.

**Explanation:**

The private variable is not used anywhere in the code making its existence unnecessary in the class.

**Refactoring proposal:**

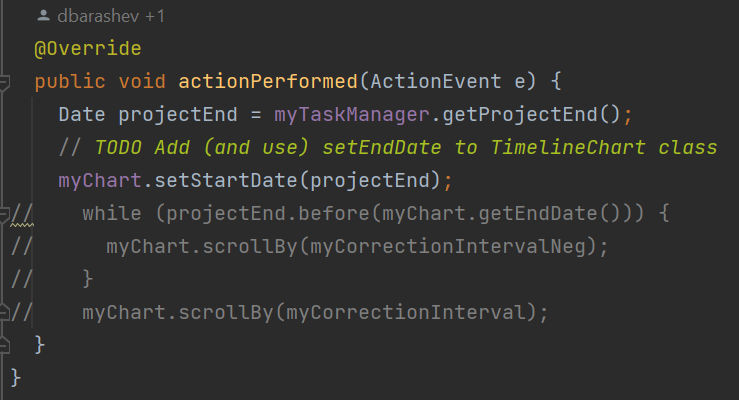
It should be deleted since it’s not used anywhere. Removing it will help the code be simpler.





Review 2

### Code Smell 3: Comments and Speculative Generality



**Location of the code smell from repository root:**

* ganttproject\src\main\java\net\sourceforge\ganttproject\action\scroll\ScrollToEndAction.java

Line 44-54.

**Explanation:**

The TODO comments made by the programmer take on a “reminder” nature. The code is unfinished and the comments are warning us about that. Taking a closer look we can see commented code that indicates that the programmer was already thinking about a future implementation that was left incomplete and is not needed at this time.

**Refactoring proposal:**

The best solution would be to choose simpler code rather than features that are not needed at this time. This means that the code should be deleted.

Review 3

This code smell was reviewed and approved by Duarte

# 

# Ricardo Bessa

## Design Patterns

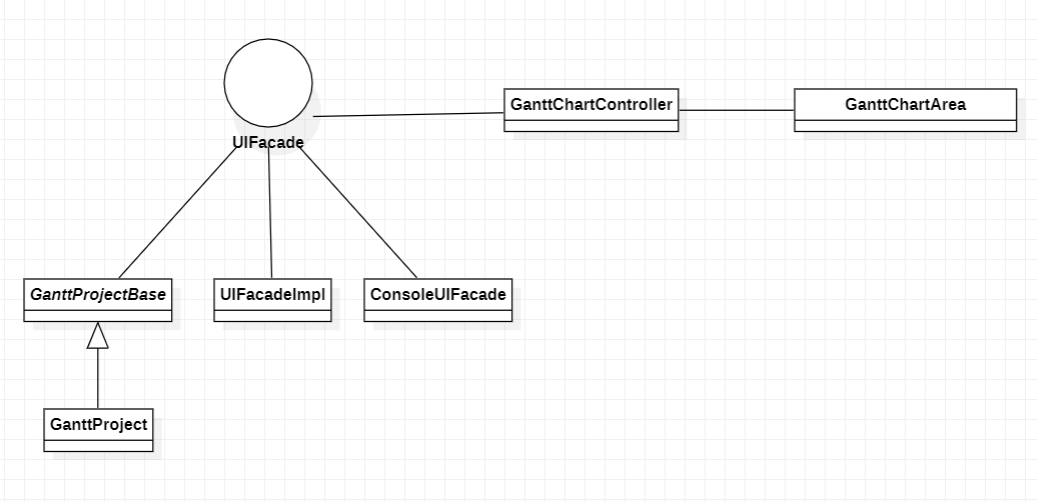
### Pattern 1: Facade Pattern (Structural Design Pattern)

**Location of the design pattern from repository root:**

* ganttproject/src/main/java/net/sourceforge/ganttproject/gui/UIFacade.java
* ganttproject/src/main/java/net/sourceforge/ganttproject/chart/gantt/GanttChartController.java
* ganttproject/src/main/java/net/sourceforge/ganttproject/export/ConsoleUIFacade.java
* ganttproject/src/main/java/net/sourceforge/ganttproject/GanttProject.java
* ganttproject/src/main/java/net/sourceforge/ganttproject/GanttProjectBase.java
* ganttproject/src/main/java/net/sourceforge/ganttproject/UIFacadeImpl.java
* ganttproject/src/main/java/net/sourceforge/ganttproject/GanttGraphicArea.java

**Explanation:**

We can find a Facade Pattern in the way the Interface UIFacade hides a more complex subsystem including classes like GanttProjectBase.java, UIFacadeImpl.java and ConsoleUIFacade.java, which can be accessed, for example, through the class GanttChartController.java. Other classes like GanttChartArea.java can interact with this subsystem through the GanttChartController.java. This is illustrated in the next simplified UML diagram of this situation.



Review 1

This code pattern was viewed and approved by Neel

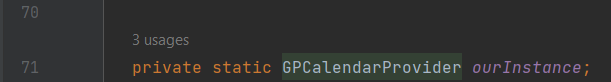
### Pattern 2: Singleton Pattern (Creational Design Pattern)

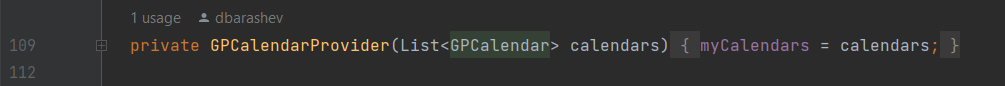
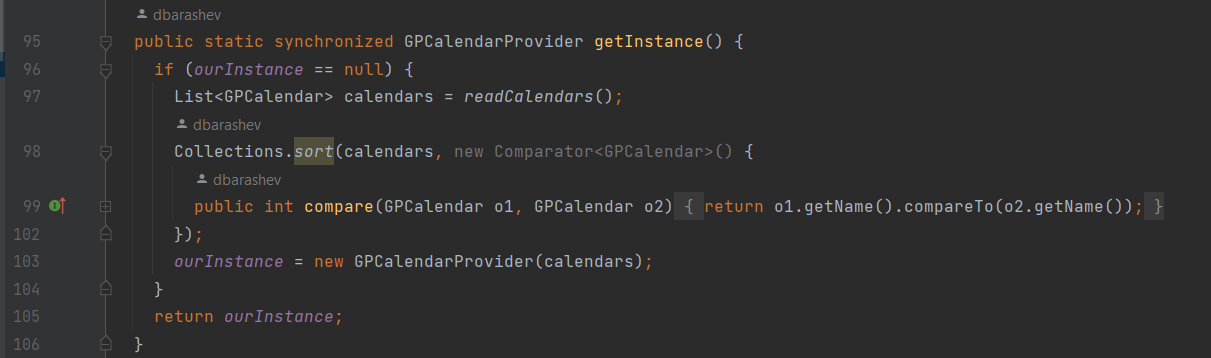
**Location of the design pattern from repository root:**

ganttproject/src/main/java/net/sourceforge/ganttproject/calendar/GPCalendarProvider.java

**Explanation:**

The class GPCalensarProvider.java has a private constructor (line 109) and a private static instance of this class. This instance is only accessed through the method getInstance() and this method guarantees that there will be one and only instance of this class because it creates a new one only if this instance is null, otherwise returns the previous created one.





Review 2

### 

### 

### 

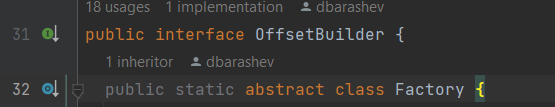
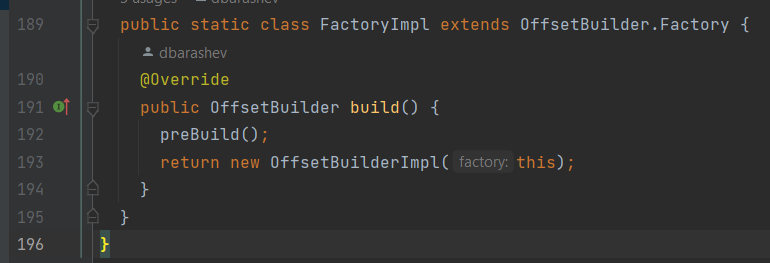
### Pattern 3: Abstract Factory Pattern (Creational)

**Location of the code smell from repository root:**

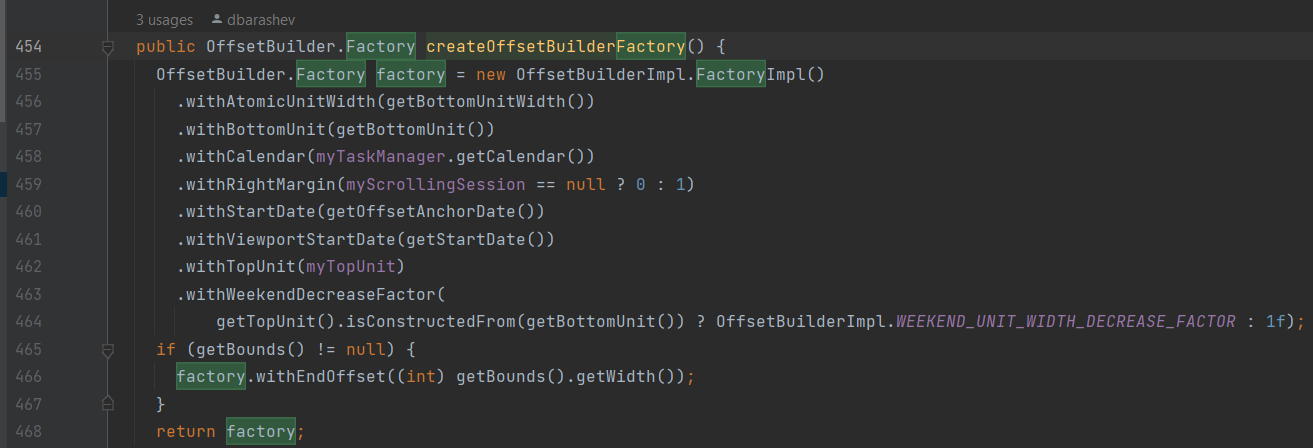
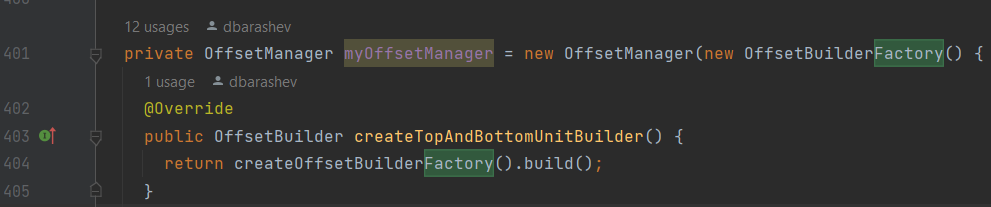
* biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilder.java
* biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java
* ganttproject/src/main/java/net/sourceforge/ganttproject/chart/ChartModelBase.java

**Explanation:**

The interface OffsetBuilder.java includes an abstract class called Factory. This Factory is extended by the class FactoryImpl, which can be found within the class OffsetBuilderImpl.java. This factory is used to create instances of the class OffsetBuilderImpl.java. This creation occurs, for example, in the class ChartModelBase.java. This way it is possible to create instances of a certain instance family without specifying their classes.







Review 3

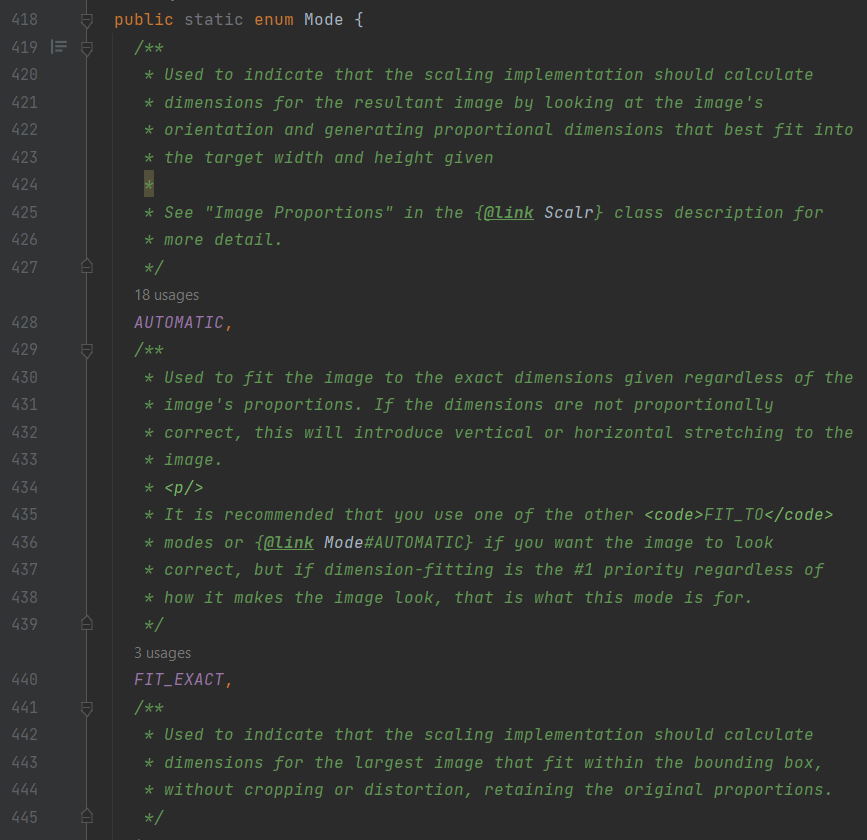
This code pattern was viewed and approved by Duarte

## Code Smells

### Code Smell 1: Comments

**Location of the code smell from repository root:**

ganttproject/src/main/java/org/imgscalr/Scalr.java



**Explanation:**

This class has too many comments with an exaggerated amount of indications and explanations about every method. We can find some very long comments like the initial one with a total of 161 lines. In this class we can also find the Enum Mode with a comment for each constant. In general we can find more comments here than real code. All these indications may suggest that the code has not the appropriate design or that it is too complex, demanding lots of time from the new developers in order to understand it.

**Refactoring proposal:**

Simplify the code and the comments, deleting the unnecessary indications and leaving the real essential ones. This way, this class will become shorter and easier to understand.

Review 1

This code smell has been reviewed and approved by Bruno Melo.

### 

### 

### 

### 

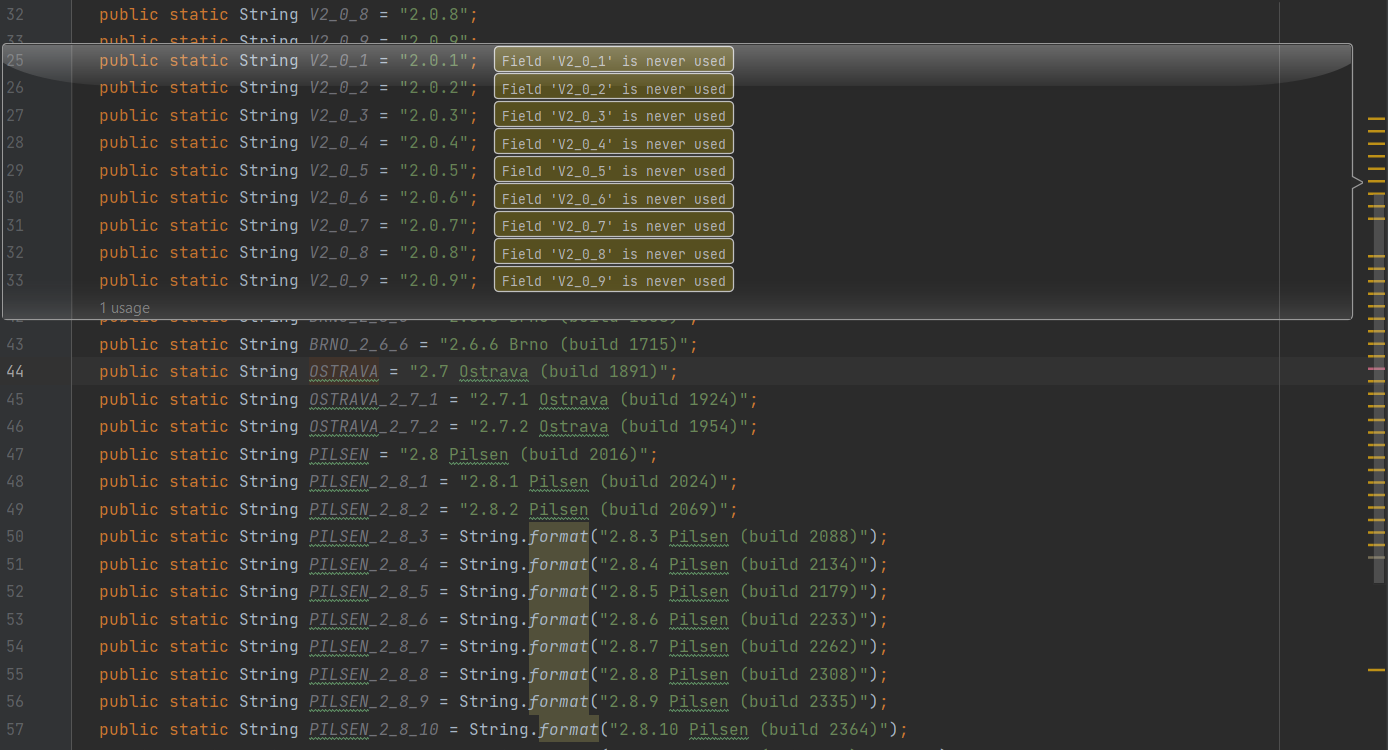
### Code Smell 2: Dead Code

**Location of the code smell from repository root:**

ganttproject/src/main/java/net/sourceforge/ganttproject/GPVersion.java

**Explanation:**

In the abstract class GPVersion.java we can find some static variables that are never used.



**Refactoring proposal:**

Remove these variables and only add them to the program when they are needed.

Review 2

This code pattern was viewed and approved by Neel.

### 

### 

### 

### 

### 

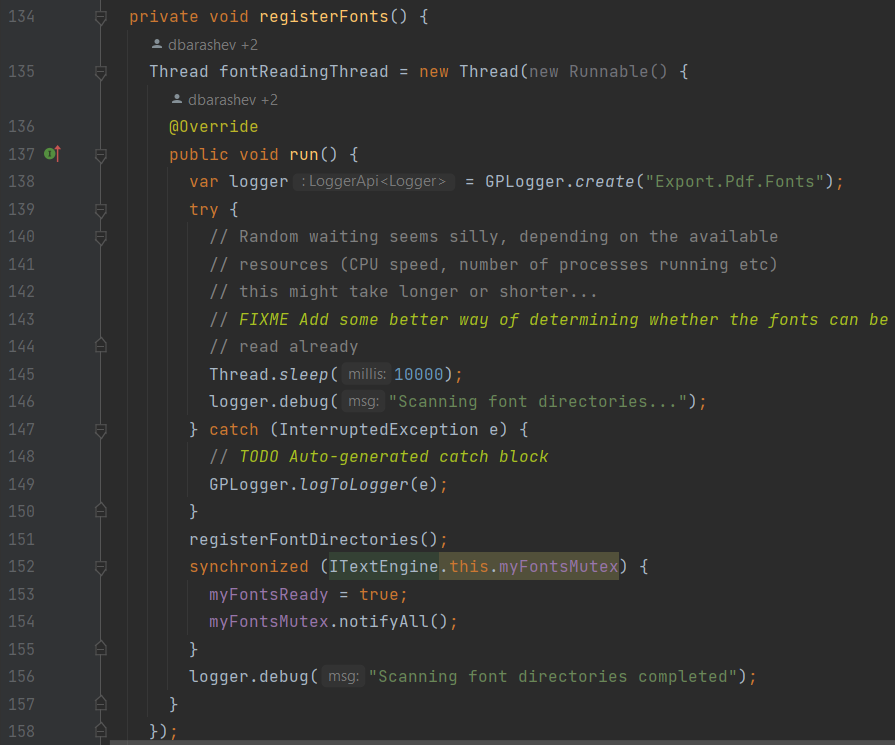
### Code Smell 3: Comments and Speculative generality

**Location of the code smell from repository root:**

org.ganttproject.impex.htmlpdf/src/main/java/org/ganttproject/impex/htmlpdf/itext/ITextEngine.java

**Explanation:**

In this class we can find the method run (line 137) which contains some code that needs to be fixed and completed in the future, like the comments suggest (lines 143 and 148).



**Refactoring proposal:**

Fix the code before advancing into other tasks or delete it until that is possible.

Review 3