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

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

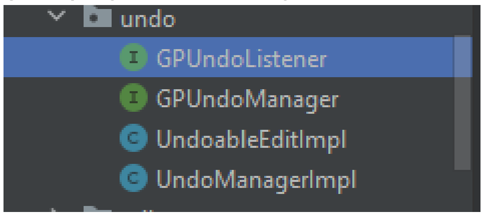
Each team member will be responsible for identifying three different design patterns. Design patterns may overlap from one team member to another, as long as they correspond to different locations in the code. Each team member should also review three other design patterns identified by the other team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

Bruno Melo

Design Patterns

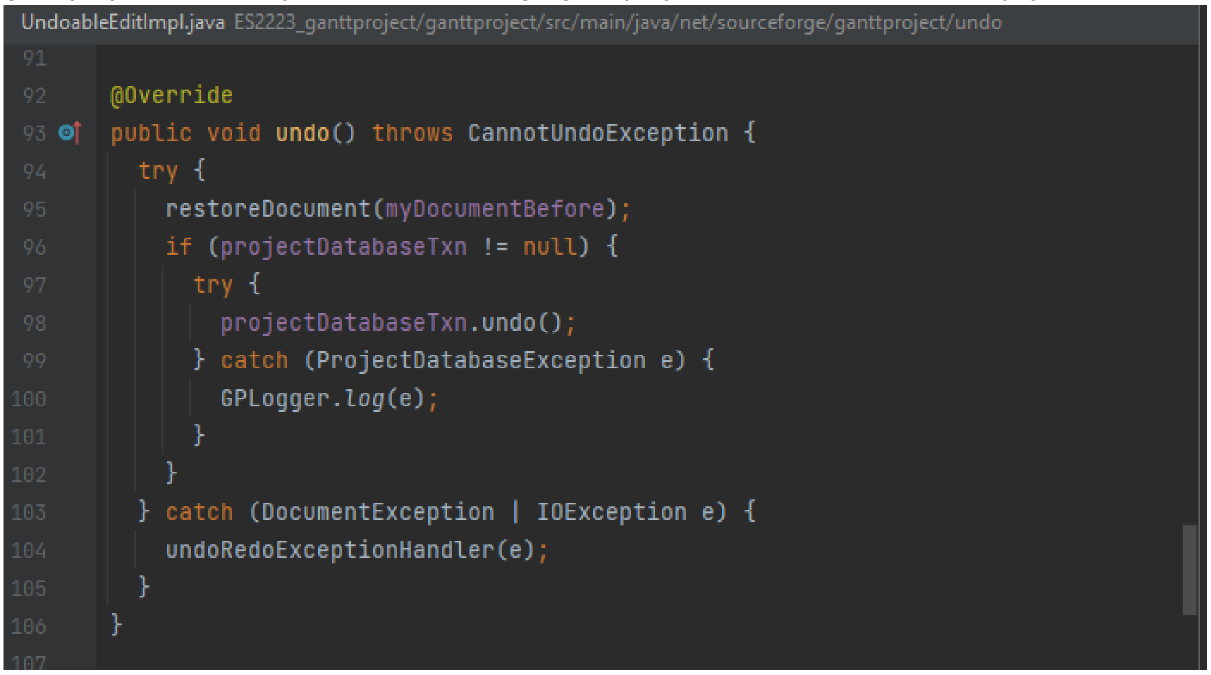
Pattern 1: Memento Pattern

Location of the pattern:
ganttproject/src/main/java/net/sourceforge/ganttproject/undo:



```
undo
    GPUndoListener
    GPUndoManager
    UndoableEditImpl
    UndoManagerImpl
```

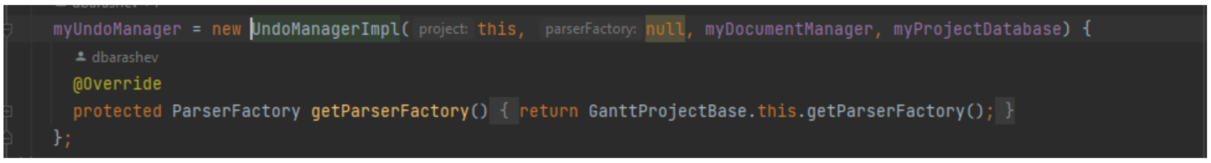
ganttproject/src/main/java/net/sourceforge/ganttproject/undo/UndoableEditImpl.java:



UndoableEditImpl.java ES2223_ganttproject/ganttproject/src/main/java/net/sourceforge/ganttproject/undo

```java
@Override
public void undo() throws CannotUndoException {
  try {
    restoreDocument(myDocumentBefore);
    if (projectDatabaseTxn != null) {
      try {
        projectDatabaseTxn.undo();
      } catch (ProjectDatabaseException e) {
        GPLogger.log(e);
      }
    }
  } catch (DocumentException | IOException e) {
    undoRedoExceptionHandler(e);
  }
}
```

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



```java
myUndoManager = new UndoManagerImpl( project: this,  parserFactory: null, myDocumentManager, myProjectDatabase) {
  dbarashev
  @Override
  protected ParserFactory getParserFactory() { return GanttProjectBase.this.getParserFactory(); }
};
```

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.

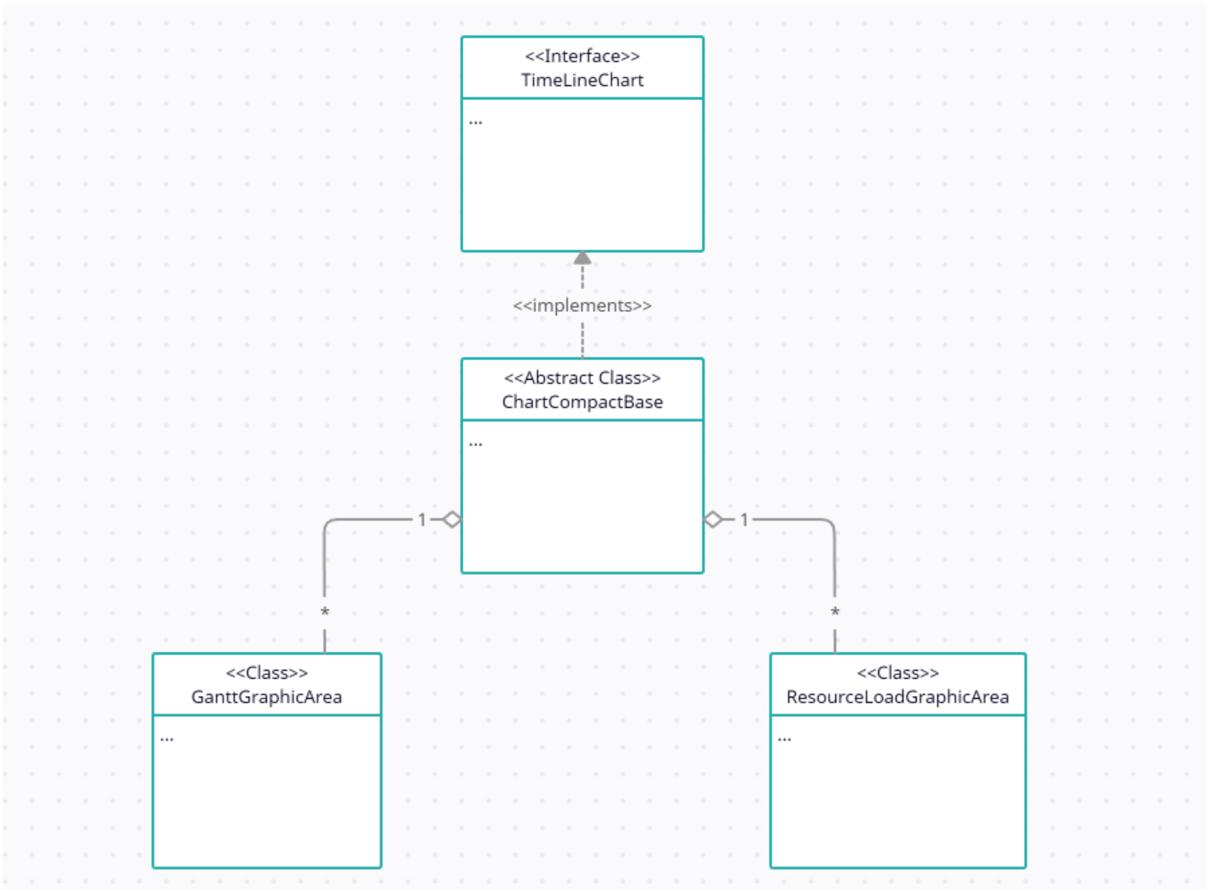
Review 1: This design pattern has been reviewed and approved by Ricardo Bessa.

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:



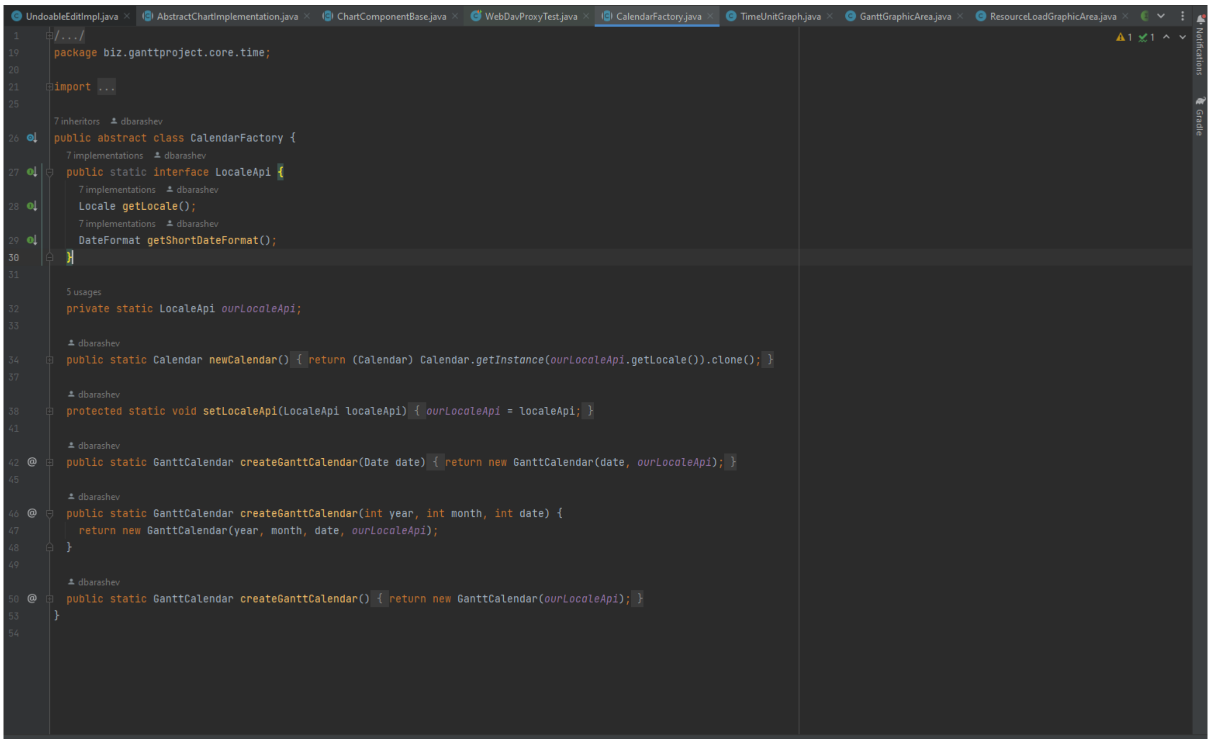
Decorator pattern allows a user to add new functionality to an existing object without altering its structure. This type of design pattern comes under structural pattern as this pattern acts as a wrapper to existing class.
This pattern creates a decorator class which wraps the original class and provides 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

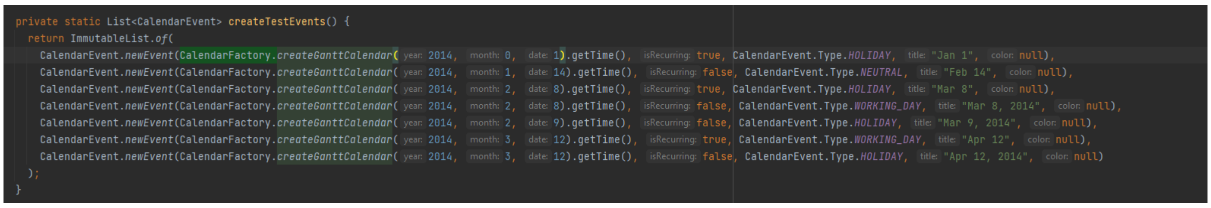
Pattern 3: Factory Method Pattern

Location of the pattern:
biz.ganttproject.core/src/main/java/biz/ganttproject/core/time/CalendarFactory.java


```
/.../
package biz.ganttproject.core.time;

import ...

7 inheritors  dbarashev
public abstract class CalendarFactory {
  7 implementations  dbarashev
  public static interface LocaleApi {
    7 implementations  dbarashev
    Locale getLocale();
    7 implementations  dbarashev
    DateFormat getShortDateFormat();
  }

  5 usages
  private static LocaleApi ourLocaleApi;

  dbarashev
  public static Calendar newCalendar() { return (Calendar) Calendar.getInstance(ourLocaleApi.getLocale()).clone(); }

  dbarashev
  protected static void setLocaleApi(LocaleApi localeApi) { ourLocaleApi = localeApi; }

  dbarashev
  public static GanttCalendar createGanttCalendar(Date date) { return new GanttCalendar(date, ourLocaleApi); }

  dbarashev
  public static GanttCalendar createGanttCalendar(int year, int month, int date) {
    return new GanttCalendar(year, month, date, ourLocaleApi);
  }

  dbarashev
  public static GanttCalendar createGanttCalendar() { return new GanttCalendar(ourLocaleApi); }
}
```

Explanation:

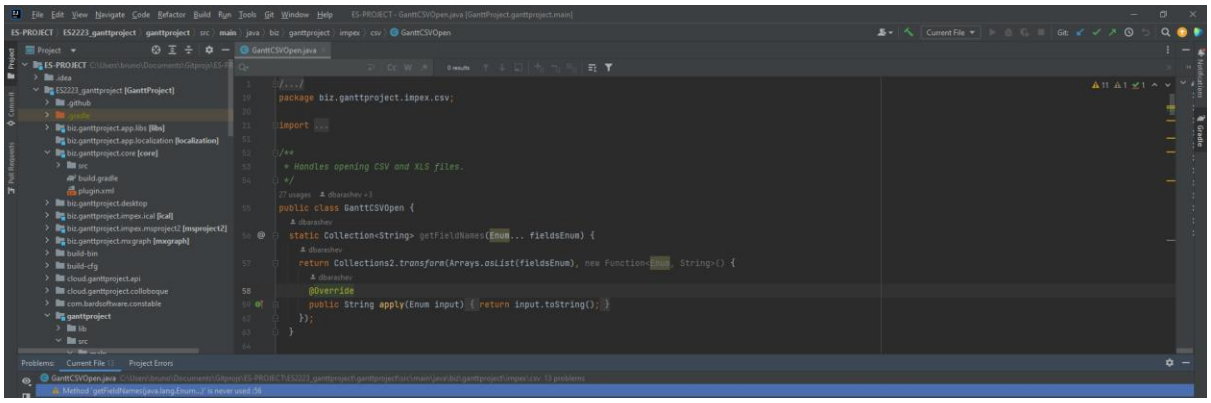
In the Factory pattern, we create objects without exposing the creation logic to the client and refer to newly created objects using a common interface.
This is exactly what the CalendarFactory class does. It creates calendar objects without exposing its creation logic to the user.


```
private static List<CalendarEvent> createTestEvents() {
  return ImmutableList.of(
    CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014,  month: 0,  date: 1).getTime(),  isRecurring: true, CalendarEvent.Type.HOLIDAY,  title: "Jan 1",  color: null),
    CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014,  month: 1,  date: 14).getTime(),  isRecurring: false, CalendarEvent.Type.NEUTRAL,  title: "Feb 14",  color: null),
    CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014,  month: 2,  date: 8).getTime(),  isRecurring: true, CalendarEvent.Type.HOLIDAY,  title: "Mar 8",  color: null),
    CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014,  month: 2,  date: 8).getTime(),  isRecurring: false, CalendarEvent.Type.WORKING_DAY,  title: "Mar 8, 2014",  color: null),
    CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014,  month: 2,  date: 9).getTime(),  isRecurring: false, CalendarEvent.Type.HOLIDAY,  title: "Mar 9, 2014",  color: null),
    CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014,  month: 3,  date: 12).getTime(),  isRecurring: true, CalendarEvent.Type.WORKING_DAY,  title: "Apr 12",  color: null),
    CalendarEvent.newEvent(CalendarFactory.createGanttCalendar( year: 2014,  month: 3,  date: 12).getTime(),  isRecurring: false, CalendarEvent.Type.HOLIDAY,  title: "Apr 12, 2014",  color: null)
  );
}
```

Review 3

Code Smells

Code Smell 1: Dead Code


```java
/.../
package biz.ganttproject.impex.csv;

import ...

/**
 * Handles opening CSV and XLS files.
 */
public class GanttCSVOpen {
  static Collection<String> getFieldNames(Enum... fieldsEnum) {
    return Collections2.transform(Arrays.asList(fieldsEnum), new Function<Enum, String>() {
      @Override
      public String apply(Enum input) { return input.toString(); }
    });
  }
```

```
GanttCSVOpen.java C:\Users\bruno\Documents\Gitprojs\ES-PROJECT\ES2223_ganttproject\ganttproject\src\main\java\biz\ganttproject\impex\csv 13 problems
Method 'getFieldNames(java.lang.Enum...)' is never used :56
```

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.

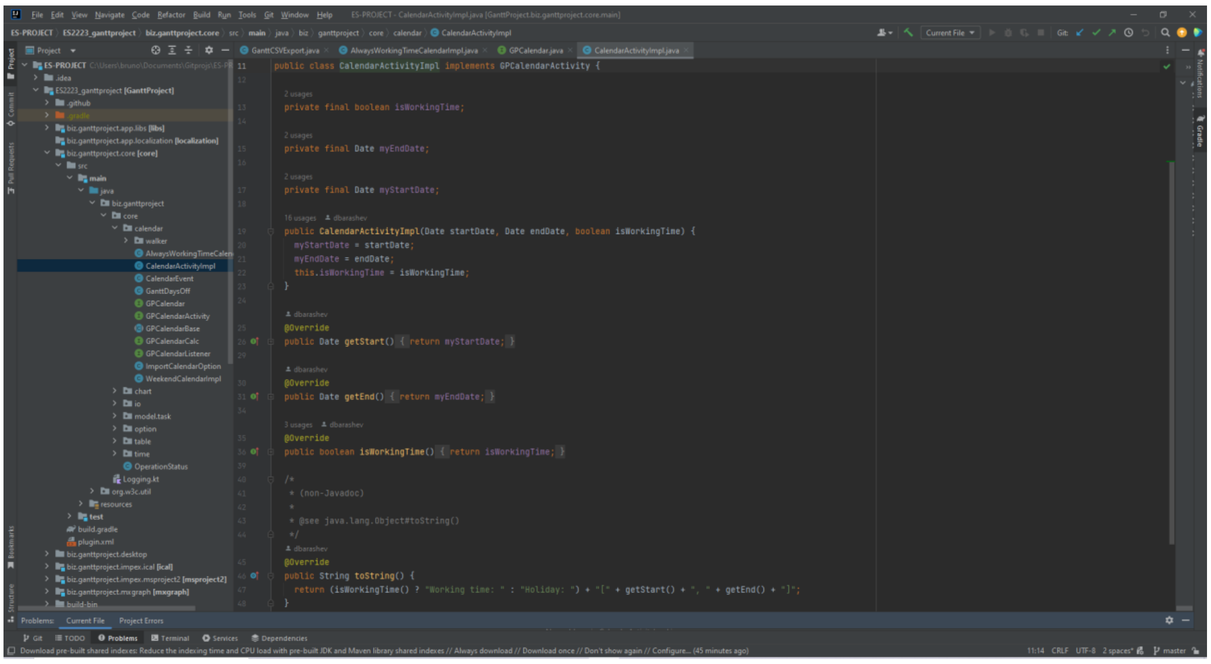
Refactoring proposal:

This method should be deleted.

Review 1-

This code smell was viewed and approved by Duarte

Code Smell 2: Data Class



```java
public class CalendarActivityImpl implements GPCalendarActivity {

  private final boolean isWorkingTime;

  private final Date myEndDate;

  private final Date myStartDate;

  public CalendarActivityImpl(Date startDate, Date endDate, boolean isWorkingTime) {
    myStartDate = startDate;
    myEndDate = endDate;
    this.isWorkingTime = isWorkingTime;
  }

  @Override
  public Date getStart() { return myStartDate; }

  @Override
  public Date getEnd() { return myEndDate; }

  @Override
  public boolean isWorkingTime() { return isWorkingTime; }

  /*
   * (non-Javadoc)
   *
   * @see java.lang.Object#toString()
   */
  @Override
  public String toString() {
    return (isWorkingTime() ? "Working time: " : "Holiday: ") + "[" + getStart() + ", " + getEnd() + "]";
  }
```

Location of the code smell from repository root:
biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/CalendarActivityImpl.java

Explanation:

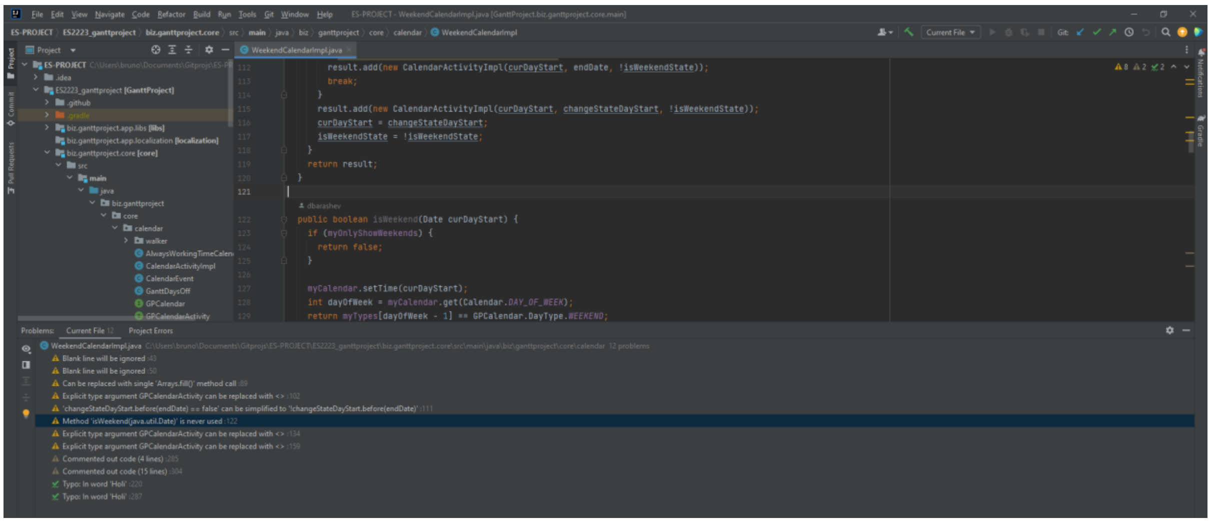
The `CalendarActivityImpl.java` class is too small and is only used to store, having no real functionality. This is a code smell known as Data Class.

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



```
112         result.add(new CalendarActivityImpl(curDayStart, endDate, !isWeekendState));
113         break;
114       }
115       result.add(new CalendarActivityImpl(curDayStart, changeStateDayStart, !isWeekendState));
116       curDayStart = changeStateDayStart;
117       isWeekendState = !isWeekendState;
118     }
119     return result;
120   }
121
      dbarashev
122   public boolean isWeekend(Date curDayStart) {
123     if (myOnlyShowWeekends) {
124       return false;
125     }
126
127     myCalendar.setTime(curDayStart);
128     int dayOfWeek = myCalendar.get(Calendar.DAY_OF_WEEK);
129     return myTypes[dayOfWeek - 1] == GPCalendar.DayType.WEEKEND;
```

```
Problems: Current File 12  Project Errors
WeekendCalendarImpl.java C:\Users\bruno\Documents\Gitprojs\ES-PROJECT\ES2223_ganttproject\biz.ganttproject.core\src\main\java\biz\ganttproject\core\calendar 12 problems
  Blank line will be ignored :43
  Blank line will be ignored :50
  Can be replaced with single 'Arrays.fill()' method call :89
  Explicit type argument GPCalendarActivity can be replaced with <> :102
  'changeStateDayStart.before(endDate) == false' can be simplified to '!changeStateDayStart.before(endDate)' :111
  Method 'isWeekend(java.util.Date)' is never used :122
  Explicit type argument GPCalendarActivity can be replaced with <> :134
  Explicit type argument GPCalendarActivity can be replaced with <> :159
  Commented out code (4 lines) :285
  Commented out code (15 lines) :304
  Typo: In word 'Holi' :220
  Typo: In word 'Holi' :287
```

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

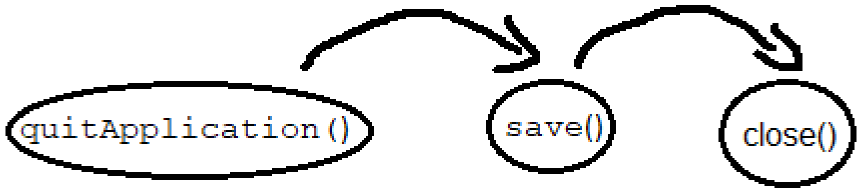
This code pattern was viewed and approved by Neel.

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

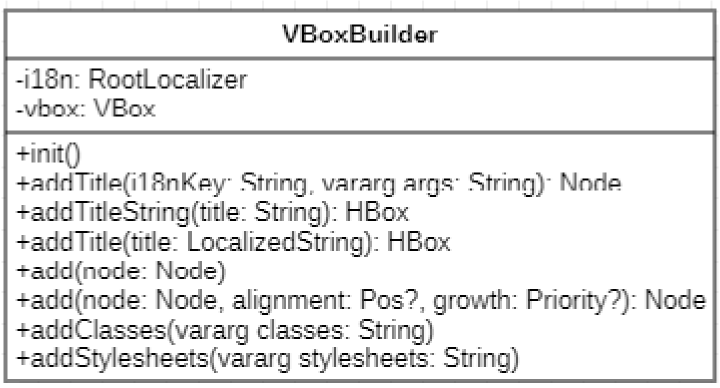
Review 1

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


VBoxBuilder
-i18n: RootLocalizer
-vbox: VBox
+init()
+addTitle(i18nKey: String, vararg args: String): Node
+addTitleString(title: String): HBox
+addTitle(title: LocalizedString): HBox
+add(node: Node)
+add(node: Node, alignment: Pos?, growth: Priority?): Node
+addClasses(vararg classes: String)
+addStylesheets(vararg stylesheets: String)

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

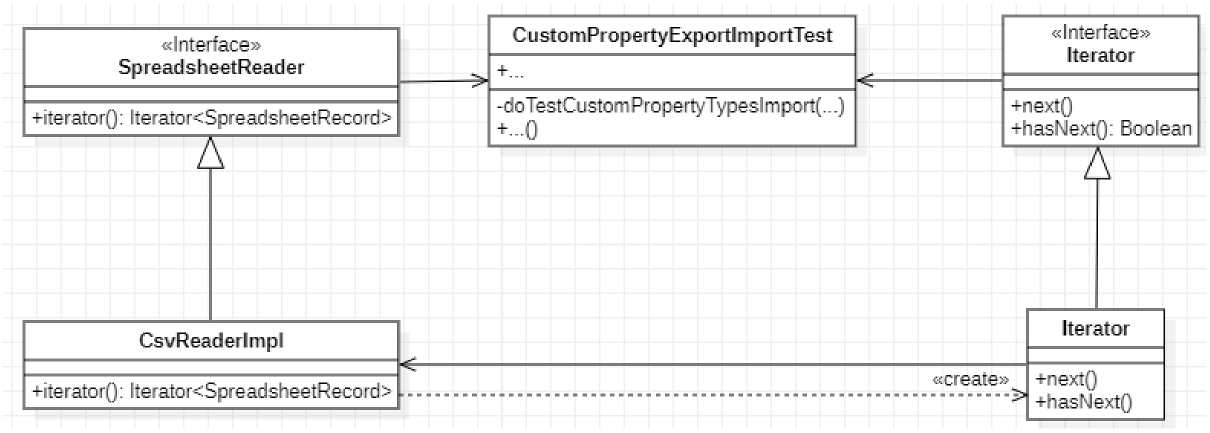
In this example, we have a VBoxBuilder that allows the creation of a personalized panel for the SignUp panel of the GanttProject Cloud

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


«Interface»
SpreadsheetReader
+iterator(): Iterator<SpreadsheetRecord>
CustomPropertyExportImportTest
+...
-doTestCustomPropertyTypesImport(...)
+...()
«Interface»
Iterator
+next()
+hasNext(): Boolean
CsvReaderImpl
+iterator(): Iterator<SpreadsheetRecord>
«create»
Iterator
+next()
+hasNext()

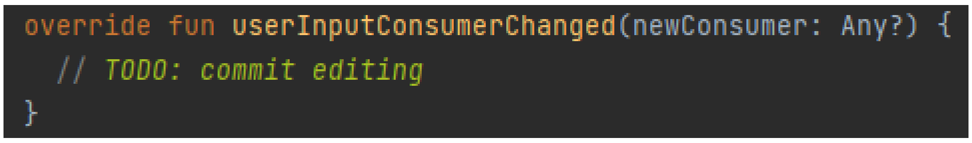
Explanation:
An Iterator provides a way to traverse a collection of objects without exposing its implementation.
In this example, we have an Iterator that allow us to navigate through all the Spreadsheet Records without exposing its implementation to the main system

Review 3

This code pattern was viewed and approved by Neel.

Code Smells

Code Smell 1: Comments and Dead Code



```
override fun userInputConsumerChanged(newConsumer: Any?) {
  // TODO: commit editing
}
```

Location of the code smell from repository root:
ganttproject\src\main\java\biz\ganttproject\ganttview, line number 433.

Explanation:

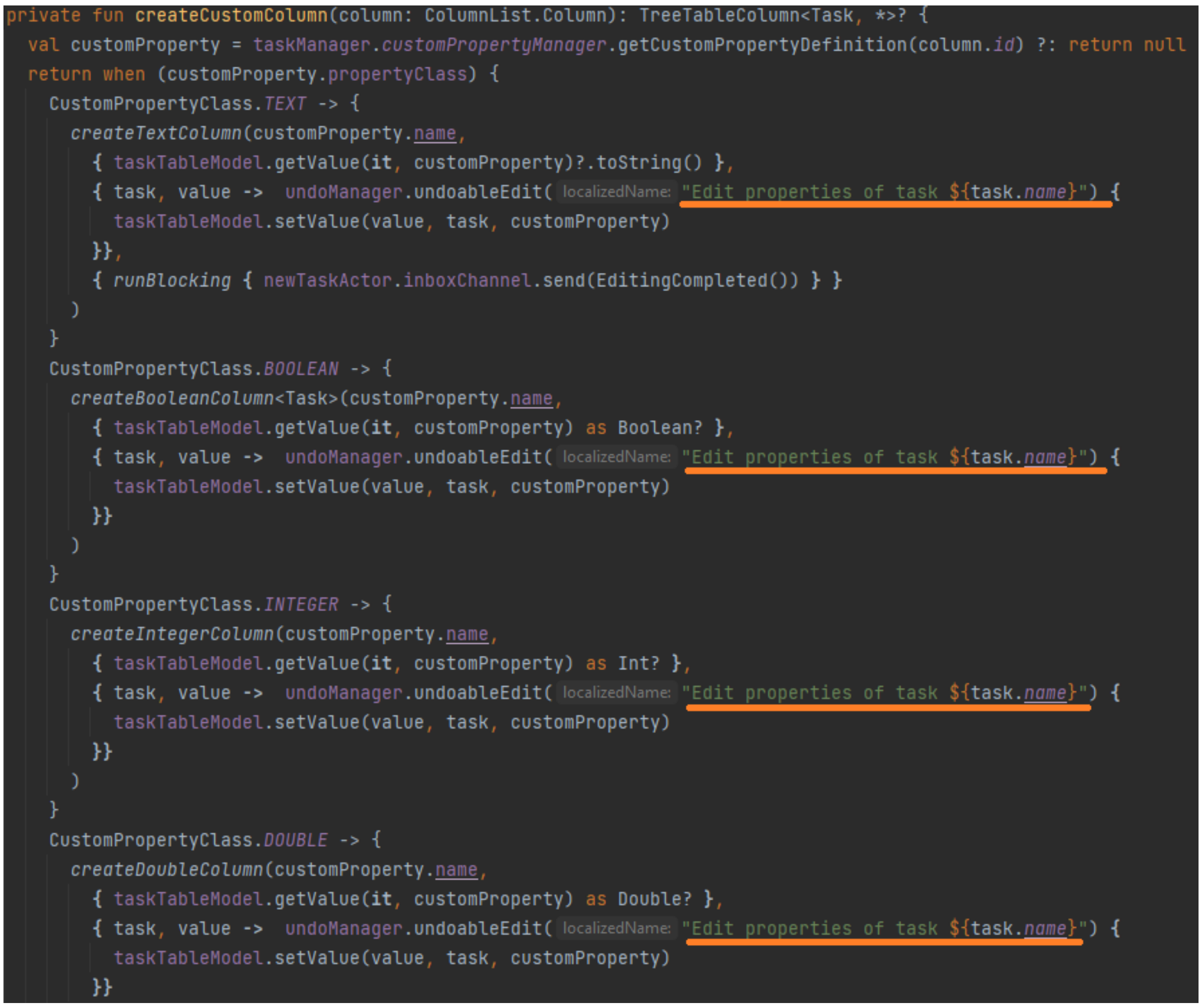
This function is currently doing nothing and has a reminder that some work will be done here in the future

Refactoring proposal:

Remove this function

Review 1

Code Smell 2: Shotgun surgery



```
private fun createCustomColumn(column: ColumnList.Column): TreeTableColumn<Task, *>? {
  val customProperty = taskManager.customPropertyManager.getCustomPropertyDefinition(column.id) ?: return null
  return when (customProperty.propertyClass) {
    CustomPropertyClass.TEXT -> {
      createTextColumn(customProperty.name,
        { taskTableModel.getValue(it, customProperty)?.toString() },
        { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
          taskTableModel.setValue(value, task, customProperty)
        }},
        { runBlocking { newTaskActor.inboxChannel.send(EditingCompleted()) } }
      )
    }
    CustomPropertyClass.BOOLEAN -> {
      createBooleanColumn<Task>(customProperty.name,
        { taskTableModel.getValue(it, customProperty) as Boolean? },
        { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
          taskTableModel.setValue(value, task, customProperty)
        }}
      )
    }
    CustomPropertyClass.INTEGER -> {
      createIntegerColumn(customProperty.name,
        { taskTableModel.getValue(it, customProperty) as Int? },
        { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
          taskTableModel.setValue(value, task, customProperty)
        }}
      )
    }
    CustomPropertyClass.DOUBLE -> {
      createDoubleColumn(customProperty.name,
        { taskTableModel.getValue(it, customProperty) as Double? },
        { task, value -> undoManager.undoableEdit( localizedName: "Edit properties of task ${task.name}") {
          taskTableModel.setValue(value, task, customProperty)
        }}
```

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

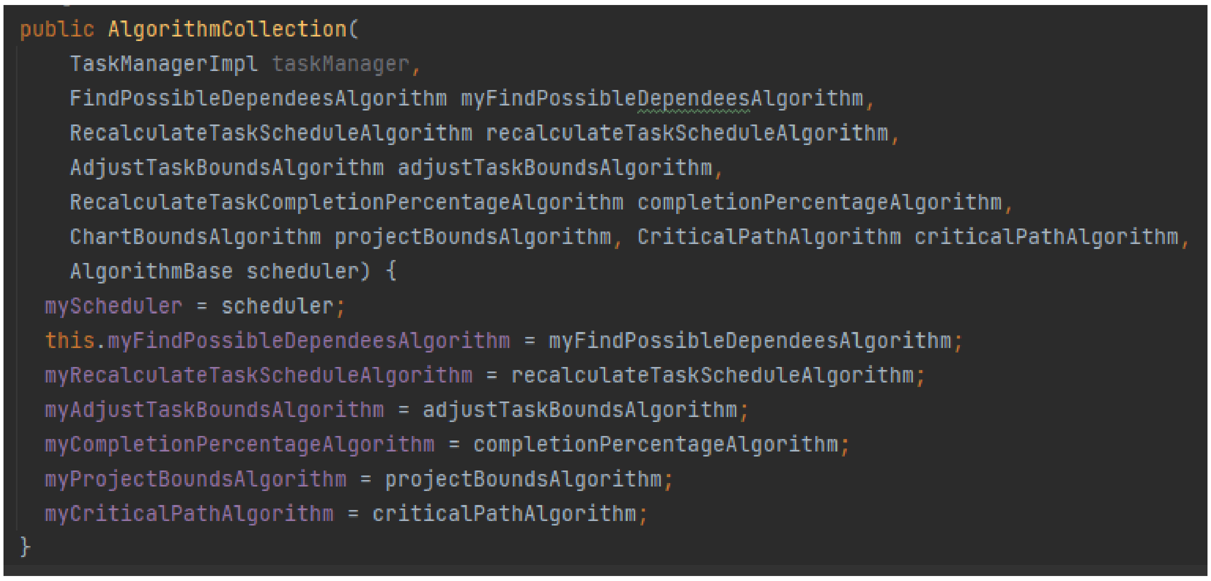
Refactoring proposal:

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



```
public AlgorithmCollection(
    TaskManagerImpl taskManager,
    FindPossibleDependeesAlgorithm myFindPossibleDependeesAlgorithm,
    RecalculateTaskScheduleAlgorithm recalculateTaskScheduleAlgorithm,
    AdjustTaskBoundsAlgorithm adjustTaskBoundsAlgorithm,
    RecalculateTaskCompletionPercentageAlgorithm completionPercentageAlgorithm,
    ChartBoundsAlgorithm projectBoundsAlgorithm, CriticalPathAlgorithm criticalPathAlgorithm,
    AlgorithmBase scheduler) {
  myScheduler = scheduler;
  this.myFindPossibleDependeesAlgorithm = myFindPossibleDependeesAlgorithm;
  myRecalculateTaskScheduleAlgorithm = recalculateTaskScheduleAlgorithm;
  myAdjustTaskBoundsAlgorithm = adjustTaskBoundsAlgorithm;
  myCompletionPercentageAlgorithm = completionPercentageAlgorithm;
  myProjectBoundsAlgorithm = projectBoundsAlgorithm;
  myCriticalPathAlgorithm = criticalPathAlgorithm;
}
```

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

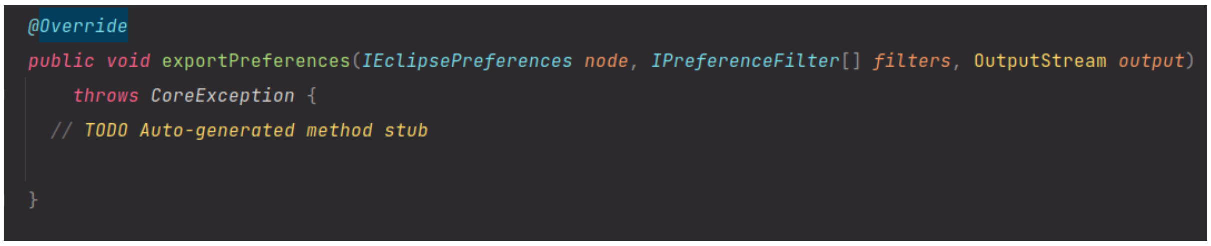
Code Smell 1: Data Class

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

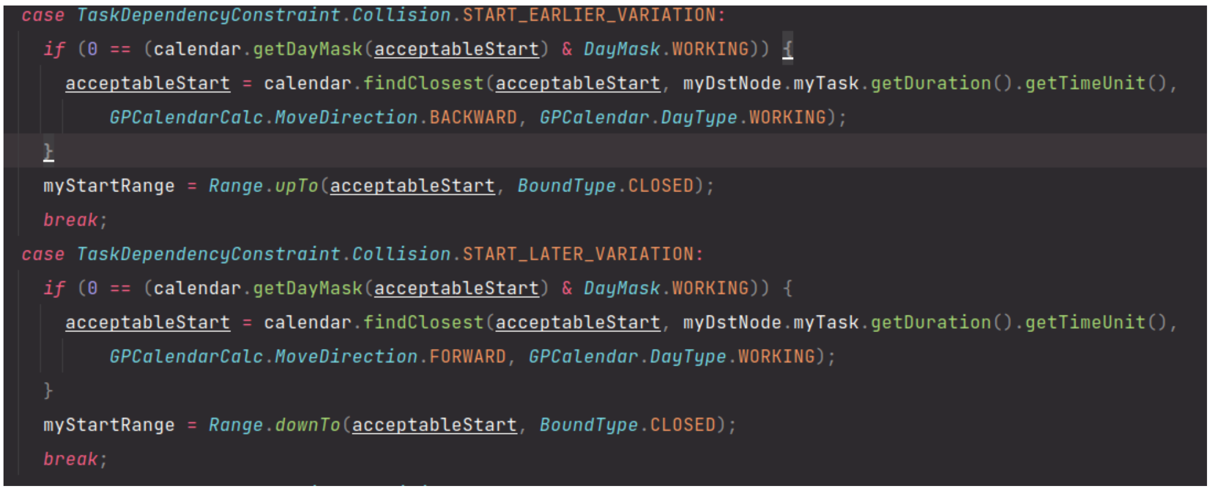

```
@Override
public void exportPreferences(IEclipsePreferences node, IPreferenceFilter[] filters, OutputStream output)
    throws CoreException {
  // TODO Auto-generated method stub

}
```

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



```
case TaskDependencyConstraint.Collision.START_EARLIER_VARIATION:
  if (0 == (calendar.getDayMask(acceptableStart) & DayMask.WORKING)) {
    acceptableStart = calendar.findClosest(acceptableStart, myDstNode.myTask.getDuration().getTimeUnit(),
        GPCalendarCalc.MoveDirection.BACKWARD, GPCalendar.DayType.WORKING);
  }
  myStartRange = Range.upTo(acceptableStart, BoundType.CLOSED);
  break;
case TaskDependencyConstraint.Collision.START_LATER_VARIATION:
  if (0 == (calendar.getDayMask(acceptableStart) & DayMask.WORKING)) {
    acceptableStart = calendar.findClosest(acceptableStart, myDstNode.myTask.getDuration().getTimeUnit(),
        GPCalendarCalc.MoveDirection.FORWARD, GPCalendar.DayType.WORKING);
  }
  myStartRange = Range.downTo(acceptableStart, BoundType.CLOSED);
  break;
```

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.

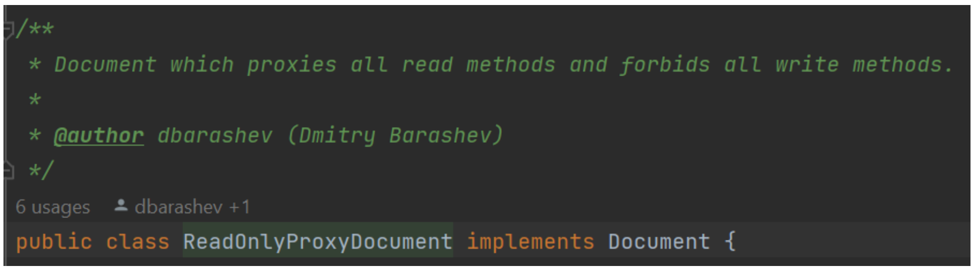
Review 3

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

Neel Badracim

Design Patterns

Pattern 1: Proxy Pattern (Structural Design Pattern)



```
/**
 * Document which proxies all read methods and forbids all write methods.
 *
 * @author dbarashev (Dmitry Barashev)
 */
6 usages   dbarashev +1
public class ReadOnlyProxyDocument implements Document {
```

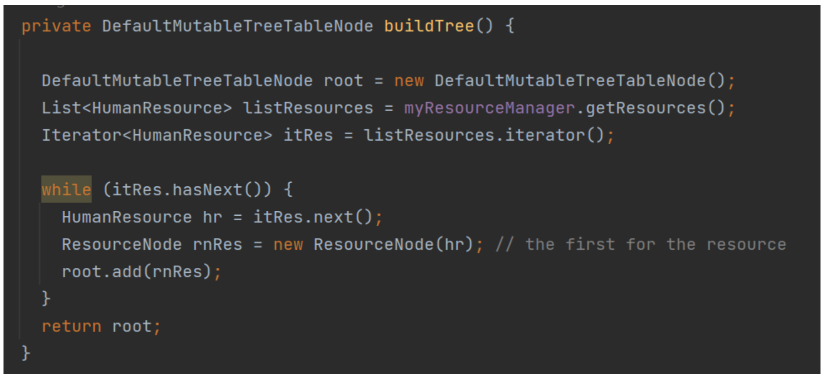
Location of the design pattern from repository root:

- \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



```
private DefaultMutableTreeTableNode buildTree() {

  DefaultMutableTreeTableNode root = new DefaultMutableTreeTableNode();
  List<HumanResource> listResources = myResourceManager.getResources();
  Iterator<HumanResource> itRes = listResources.iterator();

  while (itRes.hasNext()) {
    HumanResource hr = itRes.next();
    ResourceNode rnRes = new ResourceNode(hr); // the first for the resource
    root.add(rnRes);
  }
  return root;
}
```

Location of the design pattern from repository root:

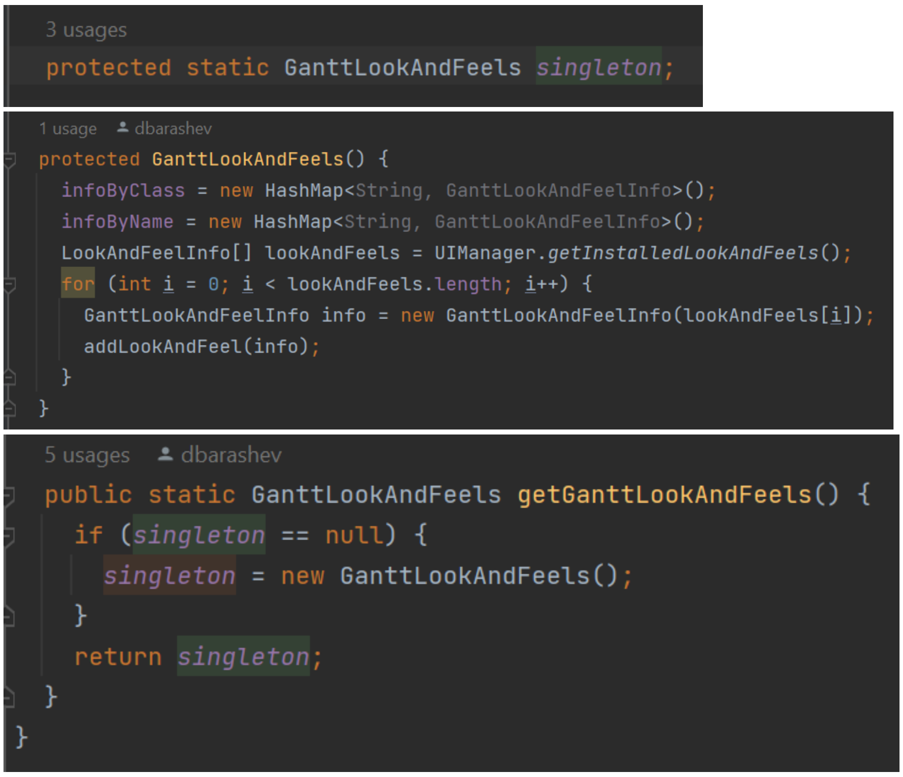
- ganttproject\src\main\java\net\sourceforge\ganttproject\ResourceTreeTableModel.java

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)



```
3 usages
protected static GanttLookAndFeels singleton;
```

```
1 usage  dbarashev
protected GanttLookAndFeels() {
  infoByClass = new HashMap<String, GanttLookAndFeelInfo>();
  infoByName = new HashMap<String, GanttLookAndFeelInfo>();
  LookAndFeelInfo[] lookAndFeels = UIManager.getInstalledLookAndFeels();
  for (int i = 0; i < lookAndFeels.length; i++) {
    GanttLookAndFeelInfo info = new GanttLookAndFeelInfo(lookAndFeels[i]);
    addLookAndFeel(info);
  }
}
```

```
5 usages  dbarashev
 public static GanttLookAndFeels getGanttLookAndFeels() {
   if (singleton == null) {
     singleton = new GanttLookAndFeels();
   }
   return singleton;
 }
}
```

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.

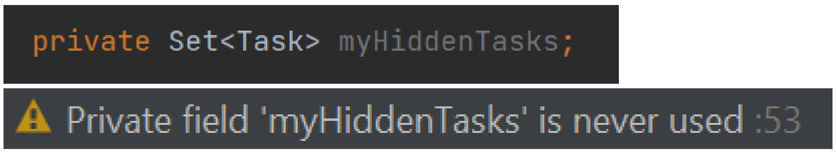
Code Smell 2: Dead Code

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.

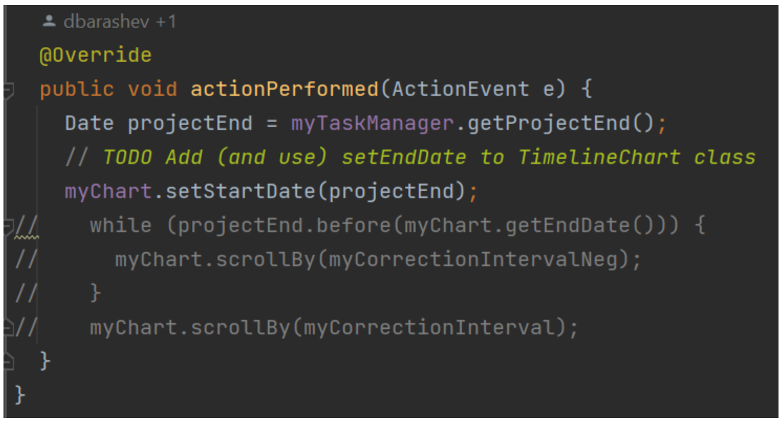


```
private Set<Task> myHiddenTasks;
```

```
Private field 'myHiddenTasks' is never used :53
```

Review 2

Code Smell 3: Comments and Speculative Generality



```
dbarashev +1
@Override
public void actionPerformed(ActionEvent e) {
  Date projectEnd = myTaskManager.getProjectEnd();
  // TODO Add (and use) setEndDate to TimelineChart class
  myChart.setStartDate(projectEnd);
//    while (projectEnd.before(myChart.getEndDate())) {
//      myChart.scrollBy(myCorrectionIntervalNeg);
//    }
//    myChart.scrollBy(myCorrectionInterval);
  }
}
```

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

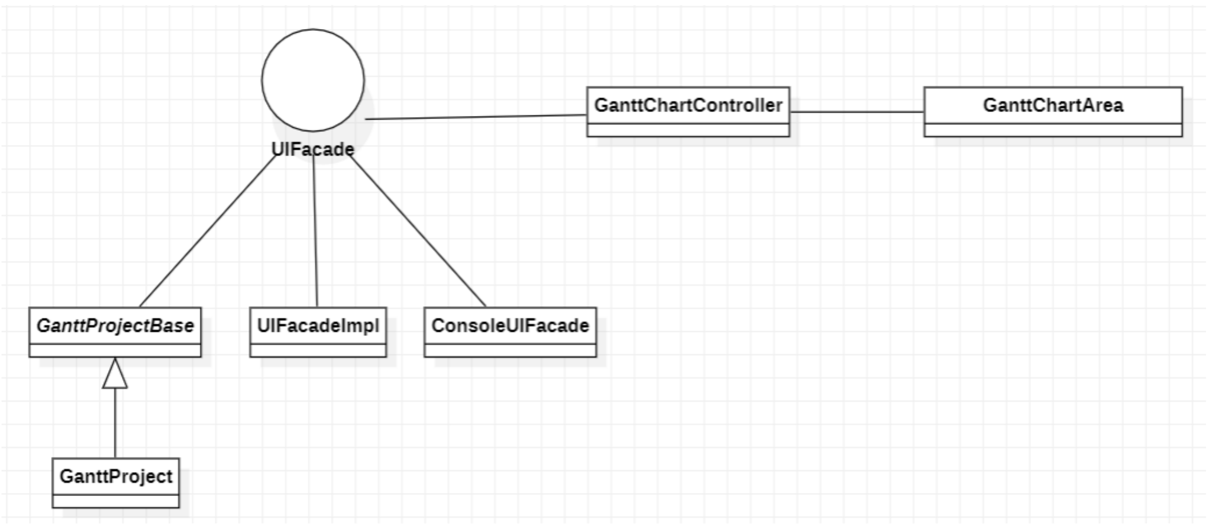
Design Patterns

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.



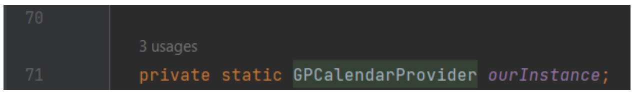
Review 1

This code pattern was viewed and approved by Neel

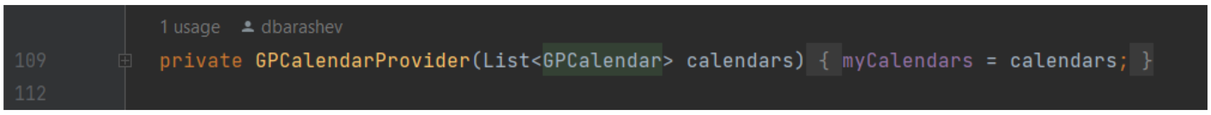
Pattern 2: Singleton Pattern (Creational Design Pattern)

Location of the design pattern from repository root:
ganttproject/src/main/java/net/sourceforge/ganttproject/calendar/GPCalendarProvider.java

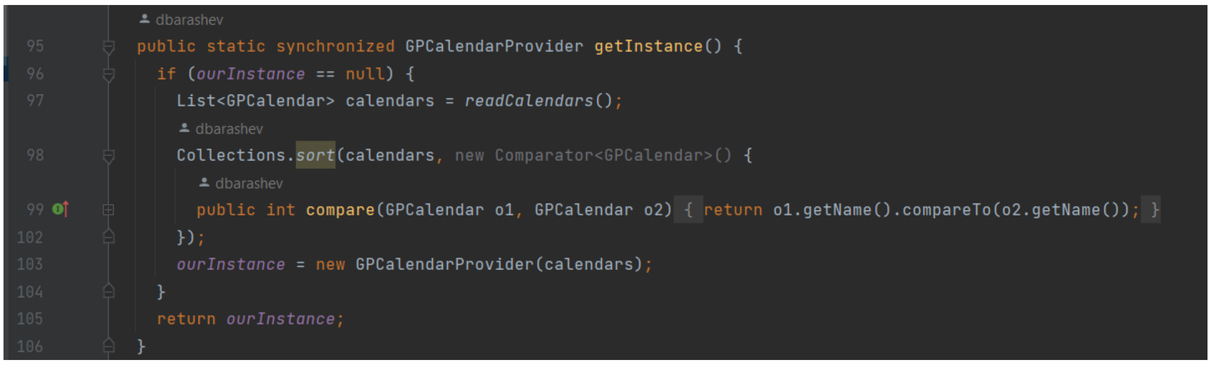
Explanation:
The class GPCalensarProvider.java has a private constructor (line 109) and a private static instance of this class. This instance is only accessed through the method getInstance() and this method guarantees that there will be one and only instance of this class because it creates a new one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



```
70
    3 usages
71  private static GPCalendarProvider ourInstance;
```



```
    1 usage  dbarashev
109 private GPCalendarProvider(List<GPCalendar> calendars) { myCalendars = calendars; }
112
```



```
    dbarashev
95  public static synchronized GPCalendarProvider getInstance() {
96    if (ourInstance == null) {
97      List<GPCalendar> calendars = readCalendars();
        dbarashev
98      Collections.sort(calendars, new Comparator<GPCalendar>() {
          dbarashev
99        public int compare(GPCalendar o1, GPCalendar o2) { return o1.getName().compareTo(o2.getName()); }
102     });
103     ourInstance = new GPCalendarProvider(calendars);
104   }
105   return ourInstance;
106 }
```

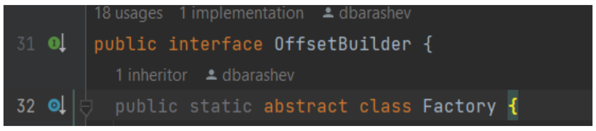
Review 2

Pattern 3: Abstract Factory Pattern (Creational)

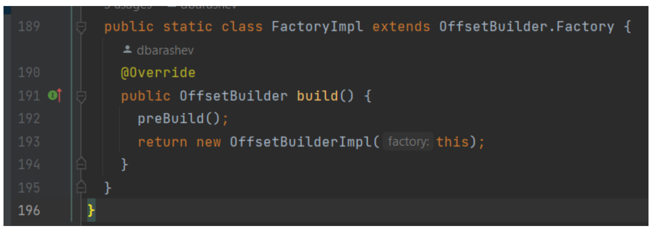
Location of the code smell from repository root:

- 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

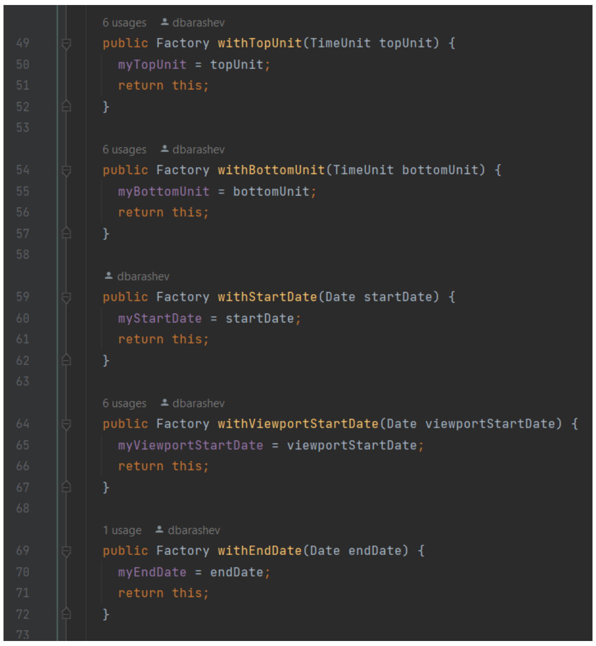
Explanation:
The interface OffsetBuilder.java includes an abstract class called Factory. This Factory is extended by the class FactoryImpl, which can be found within the class OffsetBuilderImpl.java. This factory is used to create instances of the class OffsetBuilderImpl.java. This creation occurs, for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



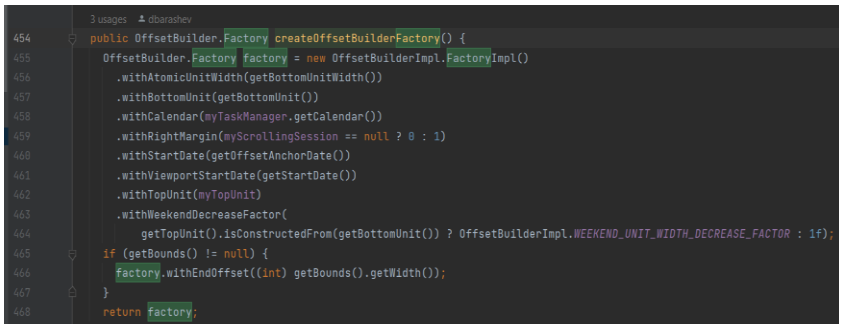
```
18 usages   1 implementation   dbarashev
public interface OffsetBuilder {
  1 inheritor   dbarashev
  public static abstract class Factory {
```



```
public static class FactoryImpl extends OffsetBuilder.Factory {
  dbarashev
  @Override
  public OffsetBuilder build() {
    preBuild();
    return new OffsetBuilderImpl( factory: this);
  }
}
}
```

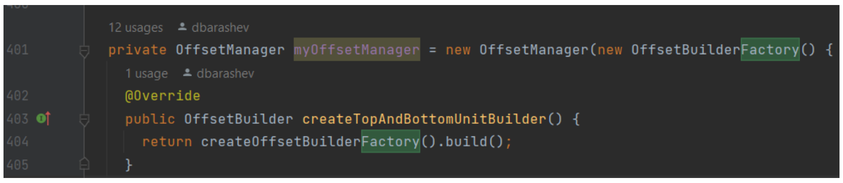


```
6 usages   dbarashev
public Factory withTopUnit(TimeUnit topUnit) {
  myTopUnit = topUnit;
  return this;
}

6 usages   dbarashev
public Factory withBottomUnit(TimeUnit bottomUnit) {
  myBottomUnit = bottomUnit;
  return this;
}

dbarashev
public Factory withStartDate(Date startDate) {
  myStartDate = startDate;
  return this;
}

6 usages   dbarashev
public Factory withViewportStartDate(Date viewportStartDate) {
  myViewportStartDate = viewportStartDate;
  return this;
}

1 usage   dbarashev
public Factory withEndDate(Date endDate) {
  myEndDate = endDate;
  return this;
}
```



```
3 usages   dbarashev
public OffsetBuilder.Factory createOffsetBuilderFactory() {
  OffsetBuilder.Factory factory = new OffsetBuilderImpl.FactoryImpl()
    .withAtomicUnitWidth(getBottomUnitWidth())
    .withBottomUnit(getBottomUnit())
    .withCalendar(myTaskManager.getCalendar())
    .withRightMargin(myScrollingSession == null ? 0 : 1)
    .withStartDate(getOffsetAnchorDate())
    .withViewportStartDate(getStartDate())
    .withTopUnit(myTopUnit)
    .withWeekendDecreaseFactor(
      getTopUnit().isConstructedFrom(getBottomUnit()) ? OffsetBuilderImpl.WEEKEND_UNIT_WIDTH_DECREASE_FACTOR : 1f);
  if (getBounds() != null) {
    factory.withEndOffset((int) getBounds().getWidth());
  }
  return factory;
```

Review 3
This code pattern was viewed and approved by Duarte



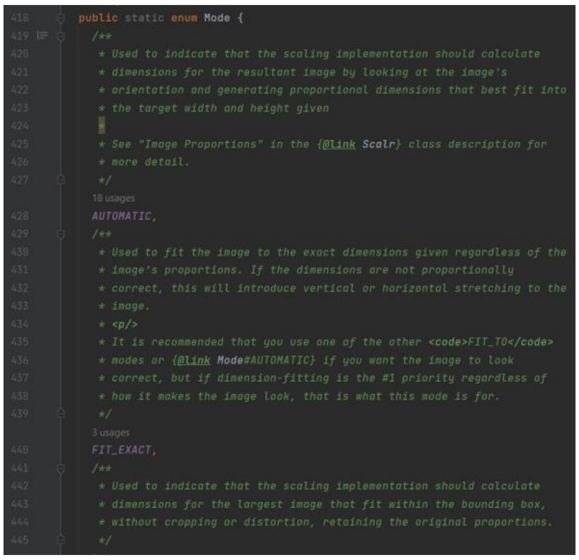
```
12 usages   dbarashev
private OffsetManager myOffsetManager = new OffsetManager(new OffsetBuilderFactory() {
  1 usage   dbarashev
  @Override
  public OffsetBuilder createTopAndBottomUnitBuilder() {
    return createOffsetBuilderFactory().build();
  }
```

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.



```
public static enum Mode {
  /**
   * Used to indicate that the scaling implementation should calculate
   * dimensions for the resultant image by looking at the image's
   * orientation and generating proportional dimensions that best fit into
   * the target width and height given
   *
   * See "Image Proportions" in the {@link Scalr} class description for
   * more detail.
   */
  18 usages
  AUTOMATIC,
  /**
   * Used to fit the image to the exact dimensions given regardless of the
   * image's proportions. If the dimensions are not proportionally
   * correct, this will introduce vertical or horizontal stretching to the
   * image.
   * <p/>
   * It is recommended that you use one of the other <code>FIT_TO</code>
   * modes or {@link Mode#AUTOMATIC} if you want the image to look
   * correct, but if dimension-fitting is the #1 priority regardless of
   * how it makes the image look, that is what this mode is for.
   */
  3 usages
  FIT_EXACT,
  /**
   * Used to indicate that the scaling implementation should calculate
   * dimensions for the largest image that fit within the bounding box,
   * without cropping or distortion, retaining the original proportions.
   */
```

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.

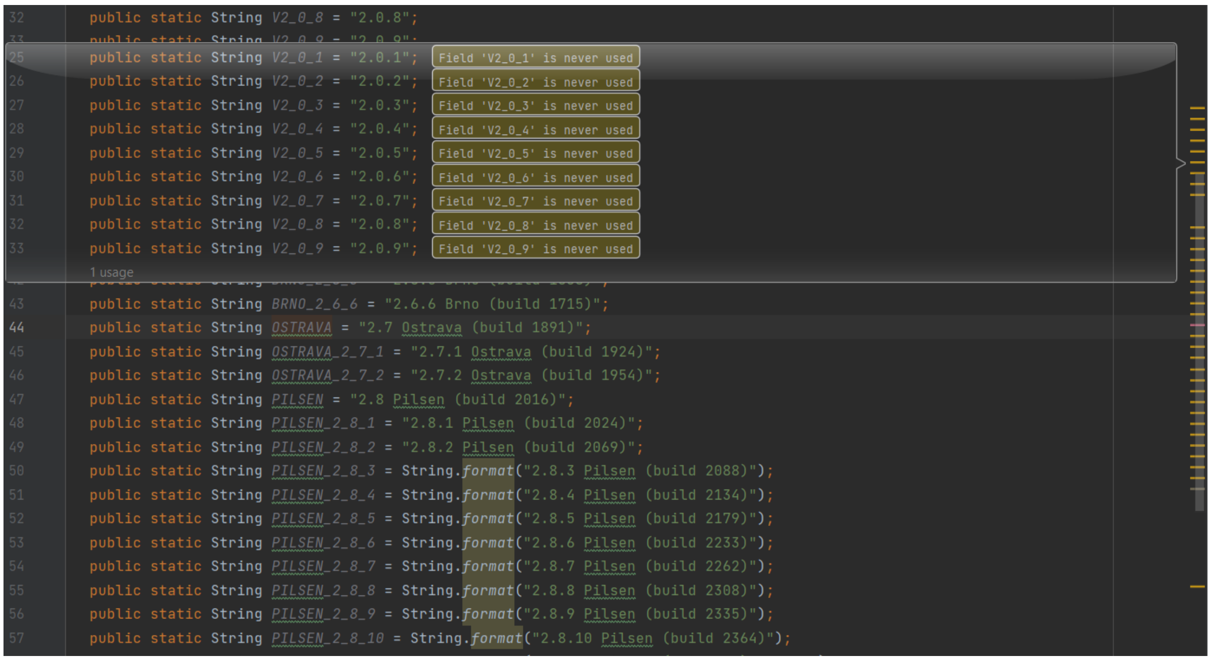
Review 1

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.



```
public static String V2_0_8 = "2.0.8";
public static String V2_0_9 = "2.0.9";
public static String V2_0_1 = "2.0.1";  Field 'V2_0_1' is never used
public static String V2_0_2 = "2.0.2";  Field 'V2_0_2' is never used
public static String V2_0_3 = "2.0.3";  Field 'V2_0_3' is never used
public static String V2_0_4 = "2.0.4";  Field 'V2_0_4' is never used
public static String V2_0_5 = "2.0.5";  Field 'V2_0_5' is never used
public static String V2_0_6 = "2.0.6";  Field 'V2_0_6' is never used
public static String V2_0_7 = "2.0.7";  Field 'V2_0_7' is never used
public static String V2_0_8 = "2.0.8";  Field 'V2_0_8' is never used
public static String V2_0_9 = "2.0.9";  Field 'V2_0_9' is never used
1 usage
public static String BRNO_2_6_6 = "2.6.6 Brno (build 1715)";
public static String OSTRAVA = "2.7 Ostrava (build 1891)";
public static String OSTRAVA_2_7_1 = "2.7.1 Ostrava (build 1924)";
public static String OSTRAVA_2_7_2 = "2.7.2 Ostrava (build 1954)";
public static String PILSEN = "2.8 Pilsen (build 2016)";
public static String PILSEN_2_8_1 = "2.8.1 Pilsen (build 2024)";
public static String PILSEN_2_8_2 = "2.8.2 Pilsen (build 2069)";
public static String PILSEN_2_8_3 = String.format("2.8.3 Pilsen (build 2088)");
public static String PILSEN_2_8_4 = String.format("2.8.4 Pilsen (build 2134)");
public static String PILSEN_2_8_5 = String.format("2.8.5 Pilsen (build 2179)");
public static String PILSEN_2_8_6 = String.format("2.8.6 Pilsen (build 2233)");
public static String PILSEN_2_8_7 = String.format("2.8.7 Pilsen (build 2262)");
public static String PILSEN_2_8_8 = String.format("2.8.8 Pilsen (build 2308)");
public static String PILSEN_2_8_9 = String.format("2.8.9 Pilsen (build 2335)");
public static String PILSEN_2_8_10 = String.format("2.8.10 Pilsen (build 2364)");
```

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

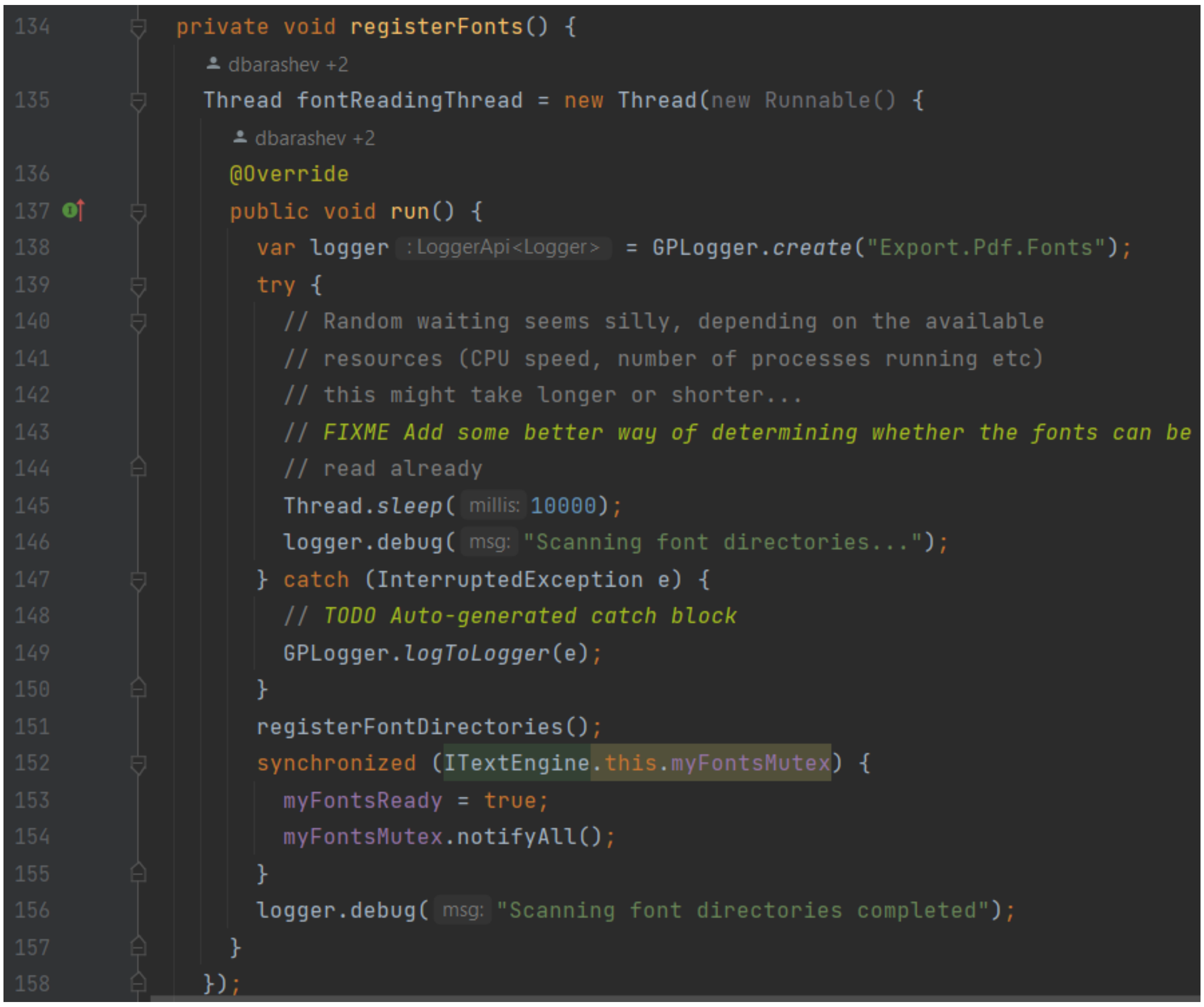
Review 2

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).



```
134   private void registerFonts() {
        dbarashev +2
135     Thread fontReadingThread = new Thread(new Runnable() {
          dbarashev +2
136       @Override
137       public void run() {
138         var logger :LoggerApi<Logger> = GPLogger.create("Export.Pdf.Fonts");
139         try {
140           // Random waiting seems silly, depending on the available
141           // resources (CPU speed, number of processes running etc)
142           // this might take longer or shorter...
143           // FIXME Add some better way of determining whether the fonts can be
144           // read already
145           Thread.sleep( millis: 10000);
146           logger.debug( msg: "Scanning font directories...");
147         } catch (InterruptedException e) {
148           // TODO Auto-generated catch block
149           GPLogger.logToLogger(e);
150         }
151         registerFontDirectories();
152         synchronized (ITextEngine.this.myFontsMutex) {
153           myFontsReady = true;
154           myFontsMutex.notifyAll();
155         }
156         logger.debug( msg: "Scanning font directories completed");
157       }
158     });
```

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

Review 3