

Software Engineering

2022/23

Gantt Project

Mergedoc deliverable

Students:

- Diogo Dias, 59731
- João Ribeiro, 60127
- Pedro Gasparinho, 60590
- Tiago Meirim, 60811
- Afonso Nunes, 64862

Chapter 1 - Patterns

Diogo Dias' Patterns

Pattern 1 - Façade

Location

ganttproject\ganttproject\src\main\java\net.sourceforge\ganttproject\gui\UIFacade

Motive - In this case, UIFacade is an interface that combines features that share a common goal, particularly, to show a message on screen. This interface can offer an abstraction of these features that can be used in the system, without knowing the complexity of the implementation.

Code

snippet

```
public interface UIFacade {  
    4 usages  
    ImageIcon DEFAULT_LOGO = new ImageIcon(UIFacade.class.getResource( name: "/icons/big.png"));  
  
    1 implementation  ↗ dbarashev +2  
    interface Dialog {  
        1 implementation  ↗ dbarashev  
        void show();  
  
        1 implementation  ↗ dbarashev  
        void hide();  
  
        1 implementation  ↗ dbarashev  
        void layout();  
  
        1 implementation  ↗ dbarashev  
        void center(Centering centering);  
        1 usage  1 implementation  ↗ Dmitry Barashev  
        void onShown(Runnable onShown);  
        2 usages  1 implementation  ↗ Dmitry Barashev  
        void onClosed(Runnable onClosed);  
        //void resize();  
    }  
}
```

Pattern 2 - Proxy

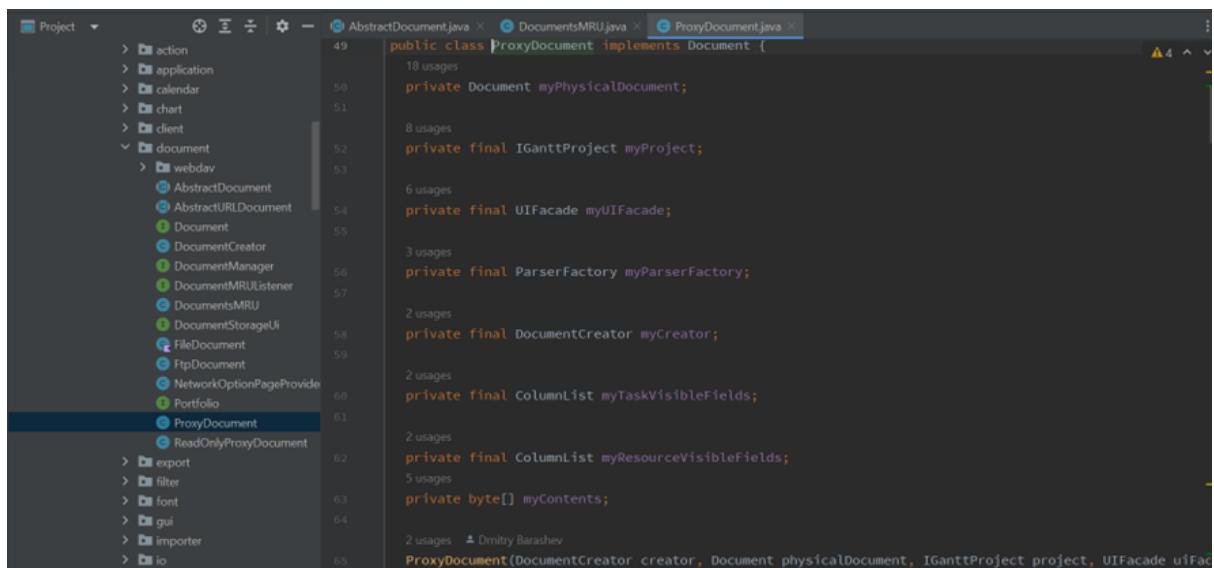
Location

ganttproject\src\main\java\net.sourceforge\ganttproject\document\ProxyDocument.java

Motive - The proxyDocument class is a simplified implementation of the Document interface. It delegates some requests to other non-proxy Document objects that can fulfill them, for example the getFileName() method, which returns the document's name using another class that extends Document. The proxyDocument class can also perform some tasks by itself, such as the write() method, which writes bytes in a file.

Code

snippet



```
public class ProxyDocument implements Document {
    private Document myPhysicalDocument;
    private final IGanttProject myProject;
    private final UIFacade myUIFacade;
    private final ParserFactory myParserFactory;
    private final DocumentCreator myCreator;
    private final ColumnList myTaskVisibleFields;
    private final ColumnList myResourceVisibleFields;
    private byte[] myContents;
    ProxyDocument(DocumentCreator creator, Document physicalDocument, IGanttProject project, UIFacade uiFacad
```

The screenshot shows a Java code editor with the 'ProxyDocument.java' file open. The code defines a class 'ProxyDocument' that implements the 'Document' interface. It contains several private fields: 'myPhysicalDocument', 'myProject', 'myUIFacade', 'myParserFactory', 'myCreator', 'myTaskVisibleFields', 'myResourceVisibleFields', and 'myContents'. The constructor 'ProxyDocument' takes parameters for these fields. The code editor also shows the project structure on the left, with 'ProxyDocument' highlighted in the 'document' package.

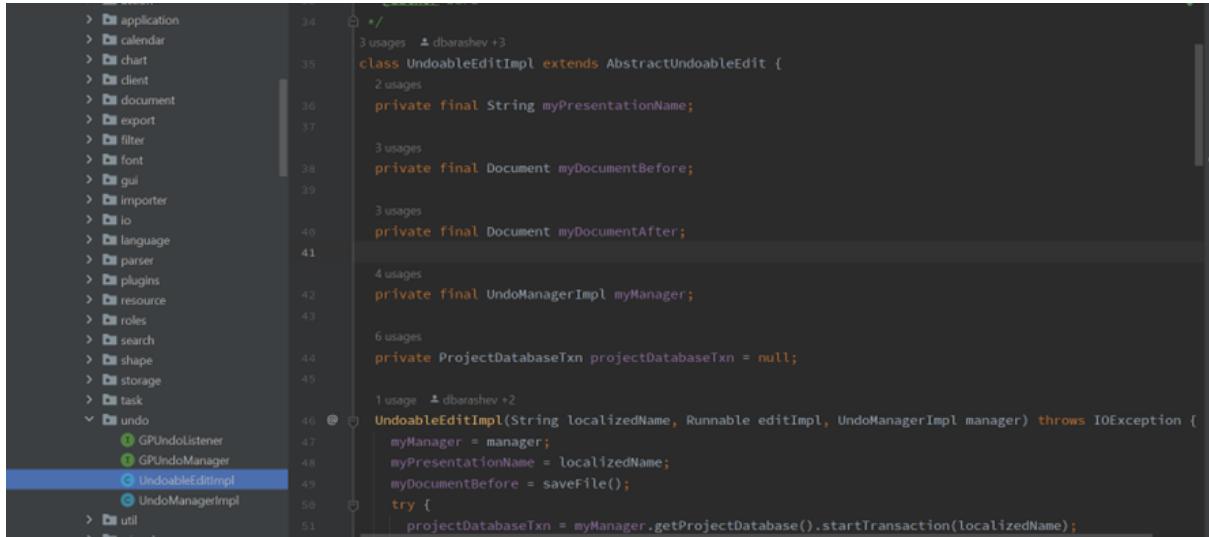
Pattern 3 - Memento Pattern

Location

ganttproject\src\main\java\net.sourceforge\ganttproject\undo\UndoableEditImpl.java

Motive - This class allows the user to save his document or require an old version of his document.

Code snippet -



A screenshot of an IDE showing Java code for the `UndoableEditImpl` class. The code is part of a larger project structure shown on the left. The `UndoableEditImpl` class extends `AbstractUndoableEdit`. It contains several private final fields: `myPresentationName`, `myDocumentBefore`, `myDocumentAfter`, `myManager`, and `projectDatabaseTxn`. The constructor initializes these fields. The code is annotated with usage counts and developer names (dbarashev).

```
> application
> calendar
> chart
> client
> document
> export
> filter
> font
> gui
> importer
> io
> language
> parser
> plugins
> resource
> roles
> search
> shape
> storage
> task
> undo
    GPUndoListener
    GPUndoManager
    UndoableEditImpl
    UndoManagerImpl
> util
34   */
35   3 usages  dbarashev +3
36   class UndoableEditImpl extends AbstractUndoableEdit {
37     2 usages
38       private final String myPresentationName;
39
40       3 usages
41         private final Document myDocumentBefore;
42
43       4 usages
44         private final Document myDocumentAfter;
45
46       6 usages
47         private ProjectDatabaseTxn projectDatabaseTxn = null;
48
49       1 usage  dbarashev +2
50       UndoableEditImpl(String localizedName, Runnable editImpl, UndoManagerImpl manager) throws IOException {
51         myManager = manager;
52         myPresentationName = localizedName;
53         myDocumentBefore = saveFile();
54         try {
55           projectDatabaseTxn = myManager.getProjectDatabase().startTransaction(localizedName);
56         }
57       }
58     }
59   }
```

João Ribeiro's Patterns

Pattern 1 - Builder

Location

[ganttproject/src/main/java/net/sourceforge/ganttproject/gui/options/OptionsPageBuilder.java](#)

Motive - OptionsPageBuilder is a builder class that allows the construction of complex and different objects step by step using the same construction code.

Code snippet

```
67  public class OptionsPageBuilder {
68      22 usages
69      I18N myI18n = new I18N();
70      1 usage
71      private Component myParentComponent;
72      2 usages
73      private final LayoutApi myLayoutApi;
74      3 usages
75      private UIFacade myUiFacade;
76      private DecimalFormat myFormat;
77
78      6 usages 2 implementations ▲ dbarashev
79      public static interface LayoutApi {
80          2 implementations ▲ dbarashev
81          void layout(JPanel panel, int componentsCount);
82
83          4 usages ▲ dbarashev
84          public static LayoutApi TWO_COLUMN_LAYOUT = new LayoutApi() {
85              ▲ dbarashev
86              @Override
87              public void layout(JPanel panel, int componentsCount) {
88                  panel.setLayout(new SpringLayout());
89                  SpringUtilities.makeCompactGrid(panel, componentsCount, cols: 2, initialX: 0, initialY: 0, xPad: 5, yPad: 3);
90              };
91
92              1 usage ▲ dbarashev
93              public static LayoutApi ONE_COLUMN_LAYOUT = new LayoutApi() {
```

Pattern 2 - Prototype

Location

ganttproject/src/main/java/net/sourceforge/ganttproject/gui/EditableList.java

Motive - This method, located in the EditableList class, allows for the creation of copies of existing objects without making the code dependent on their original classes.

Code snippet -

```
OL @ protected T createPrototype(Object editValue) {  
    ComboItem _setItem = null;  
    if (ComboItem.class.equals(editValue.getClass())) {  
        _setItem = (ComboItem) editValue;  
    } else {  
        for (int i = 0; i < myComboBox.getModel().getSize(); i++) {  
            if (((ComboItem) myComboBox.getModel().getElementAt(i)).myText.equals(editValue)) {  
                _setItem = (ComboItem) myComboBox.getModel().getElementAt(i);  
                break;  
            }  
        }  
    }  
    return _setItem == null ? null : _setItem.myObject;  
}
```

Pattern 3 - Proxy

Location

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

Motive - `ReadOnlyProxyDocument` is a proxy class that permits the usage of itself in order to allow other classes of the `Document` type to be used in codes that normally wouldn't be accepted. For example, this proxy is used to surround a `Document` in the usage of the `openProject()` method inside the `recover` method in the `HelpMenu` class.

Code

snippet

```
34     public class ReadOnlyProxyDocument implements Document {  
35  
36         12 usages  
37         private final Document myDelegate;  
38     □ public ReadOnlyProxyDocument(Document delegate) { myDelegate = delegate; }  
39  
40         ▲ dbarashev  
41  
42             @Override  
43     □ public String getFileName() { return myDelegate.getFileName(); }  
44  
45             ▲ dbarashev  
46  
47             @Override  
48     □ public boolean canRead() { return myDelegate.canRead(); }  
49  
50             ▲ dbarashev  
51  
52             @Override  
53     □ public IStatus canWrite() {  
54         return new Status(IStatus.ERROR, pluginId: "net.sourceforge.ganttproject", code: 0,  
55     }
```

Pedro Gasparinho's Patterns

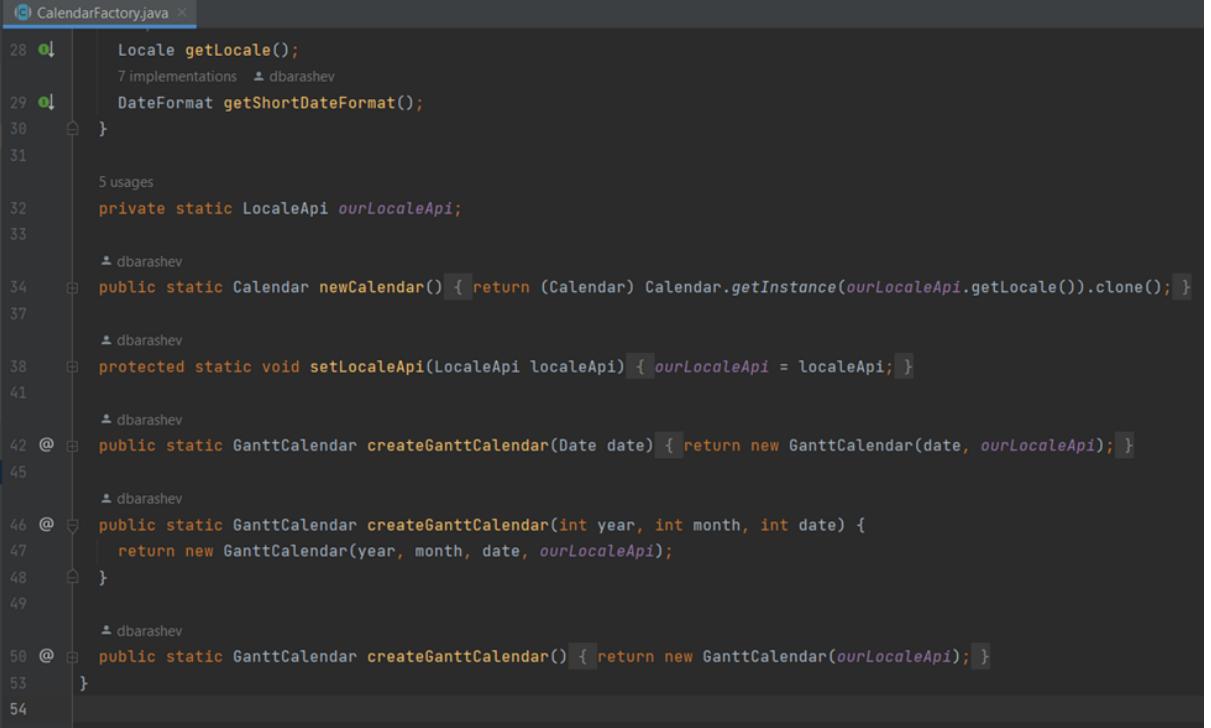
Pattern 1 - Factory

Location

ganttproject\biz.ganttproject.core\src\main\java\biz.ganttproject\core\time\CalenderFactory.

Motive - This abstract class is used to create GanttCalenders given different arguments. This helps to hide the complete process behind the creation of a GanttCalender to the user, starting off with a predefined localeApi, but also giving the possibility to change localeApi's for users with more knowledge about the system.

Code snippet



```
28     Locale getLocale();
29     DateFormat getShortDateFormat();
30 }
31
32     5 usages
33     private static LocaleApi ourLocaleApi;
34
35     public static Calendar newCalendar() { return (Calendar) Calendar.getInstance(ourLocaleApi.getLocale()).clone(); }
36
37     protected static void setLocaleApi(LocaleApi localeApi) { ourLocaleApi = localeApi; }
38
39     public static GanttCalendar createGanttCalendar(Date date) { return new GanttCalendar(date, ourLocaleApi); }
40
41     public static GanttCalendar createGanttCalendar(int year, int month, int date) {
42         return new GanttCalendar(year, month, date, ourLocaleApi);
43     }
44
45     public static GanttCalendar createGanttCalendar() { return new GanttCalendar(ourLocaleApi); }
46 }
```

Pattern 2 - Façade

Location

ganttproject\ganttproject\src\main\java\net.sourceforge.ganttproject\export\ConsoleUIFacade

Motive - ConsoleUIFacade is a façade class used to hide complex details about the system while also formatting the outputs and other functionalities in a more user-friendly way to be used in the console UI.

Code snippet -

```
ConsoleUIFacade.java
51     public ConsoleUIFacade(UIFacade realFacade) { myRealFacade = realFacade; }
54
55     ▲ Dmitry Barashev
56     @Override
57     public IntegerOption getDpiOption() { return null; }
59
60     4 usages ▲ Dmitry Barashev
61     @Override
62     public GPOption<String> getLafOption() { return null; }
64
65     5 usages ▲ dbarashev
66     @Override
67     public ScrollingManager getScrollingManager() {
68         // TODO Auto-generated method stub
69         return null;
70     }
71
72     ▲ dbarashev +1
73     @Override
74     public ZoomManager getZoomManager() {
75         return myRealFacade != null ? myRealFacade.getZoomManager() : new ZoomManager(new GPTimeUnitStack());
76     }
77
78     4 usages ▲ dbarashev
79     @Override
80     public ZoomActionSet getZoomActionSet() { return myRealFacade.getZoomActionSet(); }
81
82     3 usages ▲ dbarashev
83     @Override
84     public Choice showConfirmationDialog(String message, String title) {
85
86         5 usages ▲ dbarashev
87         @Override
88         public void showOptionDialog(int messageType, String message, Action[] actions) {
89             System.err.println("[ConsoleUIFacade]: " + message);
90         }
91
92         ▲ dbarashev
93         @Override
94         public void showErrorDialog(String errorMessage) { System.err.println("[ConsoleUIFacade] ERROR: " + errorMessage); }
95
96         4 usages ▲ dbarashev
97         @Override
98         public void showNotificationDialog(NotificationChannel channel, String message) {
99             System.err.println("[ConsoleUIFacade] " + channel.toString() + ": " + message);
100
101         ▲ dbarashev
102         @Override
103         public void showErrorDialog(Throwable e) {
104             System.err.println("[ConsoleUIFacade] ERROR: " + e.getMessage());
105             e.printStackTrace();
106         }
107
108     }
```

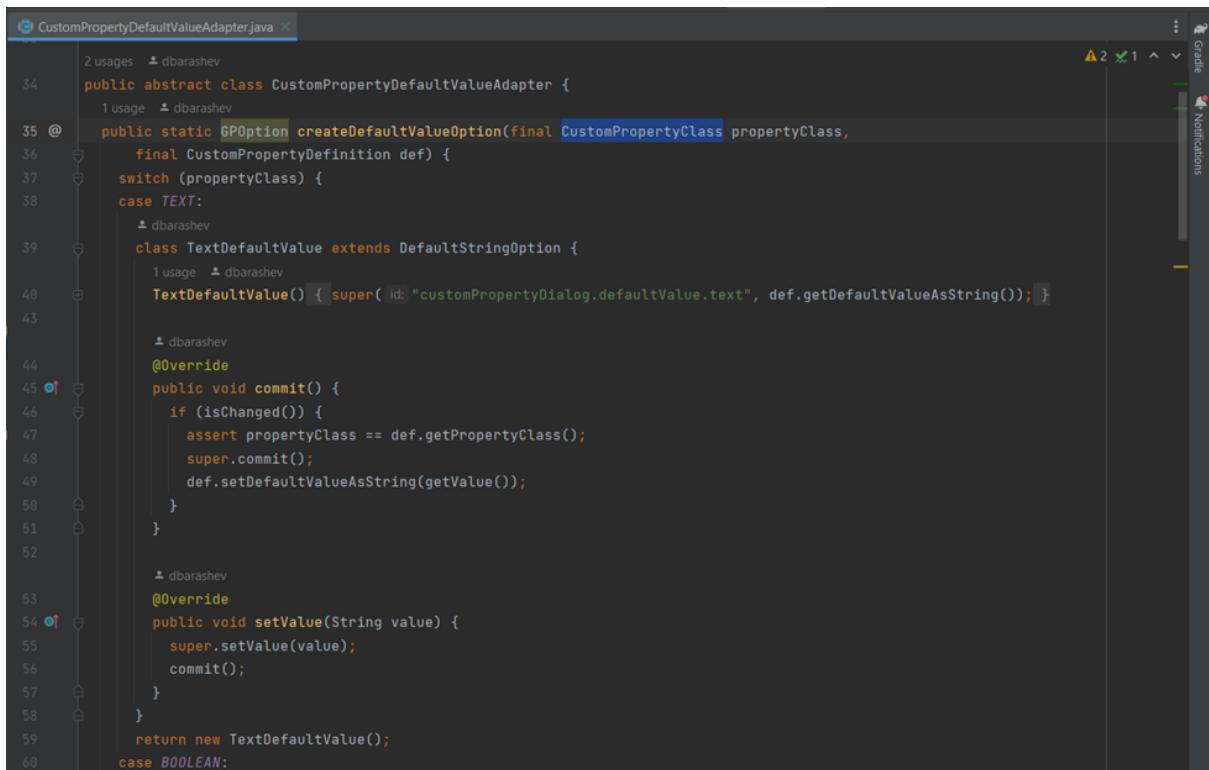
Pattern 3 - Adapter

Location

ganttproject\ganttproject\src\main\java\net.sourceforge.ganttproject\gui\options\model\CustomPropertyValueAdapter

Motive - Given a CustomPropertyClass object and a CustomPropertyDefinition object this class implements the ability to adapt the default value from the CustomPropertyDefinition object to a GPOption implemented class base on a primitive type given by the CustomPropertyClass object. While also providing a commit and a setValue method.

Code snippet -



The screenshot shows a Java code editor with the file 'CustomPropertyValueAdapter.java' open. The code defines an abstract class 'CustomPropertyValueAdapter' with static methods for creating options based on property classes and definitions. It includes a switch statement for different property types (TEXT, BOOLEAN) and overrides for 'commit()' and 'setValue(String)' methods. The code is annotated with Javadoc comments and includes assertions. The editor interface shows tabs for 'CustomPropertyValueAdapter.java', 'Gradle', and 'Notifications'.

```
CustomPropertyValueAdapter.java
34  public abstract class CustomPropertyValueAdapter {
35 @  public static GPOption createDefaultValueOption(final CustomPropertyClass propertyClass,
36   final CustomPropertyDefinition def) {
37     switch (propertyClass) {
38       case TEXT:
39         class TextDefaultValue extends DefaultStringOption {
40           TextDefaultValue() { super(id: "customPropertyDialog.defaultValue.text", def.getDefaultValueAsString()); }
41
42         @Override
43         public void commit() {
44           if (isChanged()) {
45             assert propertyClass == def.getPropertyClass();
46             super.commit();
47             def.setDefaultValueAsString(getValue());
48           }
49         }
50
51       }
52
53       @Override
54       public void setValue(String value) {
55         super.setValue(value);
56         commit();
57       }
58
59       return new TextDefaultValue();
60     case BOOLEAN:
61   }
```

Tiago Meirim's Patterns

Pattern 1 - Builder

Location

[biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java](https://github.com/tmeirim/ganttproject/blob/main/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java)

Motive - Although this pattern is not yet explored and explained by the teacher, with a little googling I found that this class fit perfectly in the builder pattern. The class lets you build the different types of objects offsets step by step, like the method constructOffsets, by exporting the object construction code, in this case from the offset, and using it in the object builders/constructors.

Code

snippet

```
// protected RegularFrameOffsetBuilder(GPCalendar calendar, TimeUnit topUnit, TimeUnit bottomUnit, Date startDate,
//     Date viewportStartDate, int defaultUnitWidth, int chartWidth, float weekendDecreaseFactor, Date endDate,
//     int rightMarginTimeUnits) {
// usage: ^ dbarashev
protected OffsetBuilderImpl(OffsetBuilder.Factory factory) {
    myCalendar = factory.myCalendar;
    myStartDate = factory.myStartDate;
    myViewportStartDate = factory.myViewportStartDate;
    myTopUnit = factory.myTopUnit;
    myBottomUnit = factory.myBottomUnit;
    myDefaultUnitWidth = factory.myAtomicUnitWidth;
    myChartWidth = factory.myEndOffset;
    myWeekendDecreaseFactor = factory.myWeekendDecreaseFactor;
    myEndDate = factory.myEndDate;
    baseUnit = factory.myBaseUnit;
    myRightMarginBottomUnitCount = factory.myRightMarginTimeUnits;
    myOffsetStepFn = factory.myOffsetStepFn;
}
```

Pattern 2 - Façade

Location

[biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetLookup.java](#)

Motive - The class creates other comparator classes that implement the same interface but for different types that are used as arguments when creating the class. Because both classes use the same name to compare the only difference between them is the type.

Code

snippet

```
22 usages ▲ dbarashev
public class OffsetLookup {
    8 usages 3 implementations ▲ dbarashev
    public static interface ComparatorBy<T> {
        3 implementations ▲ dbarashev
        int compare(T point, int offsetIdx, List<Offset> offsets);
    }

    3 usages ▲ dbarashev
    static class ComparatorByStartDate implements ComparatorBy<Date> {
        ▲ dbarashev
        @Override
        public int compare(Date point, int offsetIdx, List<Offset> offsets) {
            return point.compareTo(offsets.get(offsetIdx).getOffsetStart());
        }
    }

    5 usages ▲ dbarashev
    static class ComparatorByEndDate implements ComparatorBy<Date> {
        ▲ dbarashev
        @Override
        public int compare(Date point, int offsetIdx, List<Offset> offsets) {
            return point.compareTo(offsets.get(offsetIdx).getOffsetEnd());
        }
    }
}
```

Pattern 3 - Factory

Location

[biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java](#)

Motive - This class is hiding the creation of instances of the given type behind an interface, so the client doesn't depend on concrete classes and one class can be easily replaceable or added. In this specific implementation the class builds grid offsets for timelines where top cells are always constructed from the integer number of bottom cells.

Code

snippet

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

Afonso Nunes's Patterns

Pattern 1 - Façade

Location

ganttproject/src/main/java/net/sourceforge/ganttproject/task/TaskContainmentHierarchyFacade.java

Motive - After a discussion with my colleague and reviewing the code a little bit more I came to terms with his review and can also agree that this interface is indeed used to encapsulate Tasks whose implementation is done in a tree and not any type of Task.

Code snippet -

```
145     /** Move whatMove to whereMove, added as a child at index */
146     void move(Task whatMove, Task whereMove, int index);
147
148     boolean areUnrelated(Task dependant, Task dependee);
149
150     int getDepth(Task task);
151
152     int compareDocumentOrder(Task next, Task dependeeTask);
153
154     List<Task> getTasksInDocumentOrder();
155
156     List<Task> breadthFirstSearch(Task root, final Predicate<Pair<Task, Task>> predicate);
157     List<Task> breadthFirstSearch(Task root, boolean includeRoot);
158
159     boolean contains(Task task);
160
161     interface Factory {
162         TaskContainmentHierarchyFacade createFacade();
163     }
164
165     TaskContainmentHierarchyFacade STUB = new TaskContainmentHierarchyFacade() {
166         @Override
167         public Task[] getNestedTasks(Task container) { return new Task[0]; }
168
169         @Override
170         public Task[] getDeepNestedTasks(Task container) {
171             // TODO Auto-generated method stub
172             return null;
173         }
174
175         @Override
176         public boolean hasNestedTasks(Task container) { return false; }
177     };
178 }
```

```
102 ① @ □
103     ▲ dbarashov
104     @Override
105     public Task getRootTask() { return null; }
106
107 ① @ □
108     ▲ dbarashov
109     @Override
110     public Task getContainer(Task nestedTask) { return null; }
111
112 ① @ □
113     ▲ Kambus
114     @Override
115     public void sort(Comparator<Task> comparator) {
116
117     3 usages ▲ dbarashov
118     @Override
119     public Task getPreviousSibling(Task nestedTask) { return null; }
120
121     1 usage ▲ dbarashov
122     @Override
123     public Task getNextSibling(Task nestedTask) { return null; }
124
125     9 usages ▲ dbarashov
126     @Override
127     public int getTaskIndex(Task nestedTask) { return 0; }
128
129     4 usages ▲ dbarashov
130     @Override
131     public List<Integer> getOutlinePath(Task task) { return Collections.emptyList(); }
132
133     ▲ dbarashov
134     @Override
135     public void move(Task whatMove, Task whereMove) {
136
137     ▲ dbarashov
138     @Override
139     public void move(Task whatMove, Task whereMove, int index) {
140
141     7 usages ▲ dbarashov
142     @Override
143     public boolean areUnrelated(Task dependant, Task dependee) { return false; }
144
145     4 usages ▲ dbarashov
146     @Override
147     public int getDepth(Task task) { return 0; }
148
149     4 usages ▲ dbarashov
150     @Override
151     public int compareDocumentOrder(Task next, Task dependeeTask) { throw new UnsupportedOperationException(); }
152
153     ▲ dbarashov
154     @Override
155     public boolean contains(Task task) { throw new UnsupportedOperationException(); }
156
```

```

④ TaskContainmentHierarchyFacade.java -->
1  /**
2   * ...
3   */
4  package net.sourceforge.ganttproject.task;
5
6  import ...
7
8  /**
9   * @author band
10  */
11
12  public interface TaskContainmentHierarchyFacade {
13
14      Task[] getNestedTasks(Task container);
15
16      Task[] getDeepNestedTasks(Task container);
17
18      boolean hasNestedTasks(Task container);
19
20      Task getRootTask();
21
22      Task getContainer(Task nestedTask);
23
24      void sort(Comparator<Task> comparator);
25
26      /**
27       * @return the previous sibling or null if task is the first child of the
28       *         parent task
29      */
30
31      Task getPreviousSibling(Task nestedTask);
32
33      /**
34       * @return the next sibling or null if task is the last child of the parent
35       *         task
36      */
37
38      Task getNextSibling(Task task);
39
40      /**
41       * @return the index of the nestedTask with respect of its siblings */
42      int getTaskIndex(Task nestedTask);
43
44      List<Integer> getOutlinePath(Task task);
45
46      /**
47       * Move whatMove to whereMove, added as a child at the end */
48      void move(Task whatMove, Task whereMove);
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63

```

Pattern 2 - Factory

Location

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

Motive - This interface is used to hide the creation of instances of type Parser behind it making it so we can replace the concrete parser classes with others if we need to and it has no impact on the client code.

Code

snippet

```
1  /...
2  package net.sourceforge.ganttproject.parser;
3
4  import net.sourceforge.ganttproject.io.GPSaver;
5
6  /**
7   * @author bard
8   */
9  public interface ParserFactory {
10
11     GPParser newParser();
12
13
14     GPSaver newSaver();
15
16 }
```

Pattern 3 - Command

Location

ganttproject/src/main/java/net/sourceforge/ganttproject/action/edit

Motive - Since I didn't specify any of the operations, I can understand my colleague's review. As it can be **seen from the screenshots** there are many operations supported in this implementation of the command pattern and that are useful for the user.

Code snippet -

```
39 package net.sourceforge.ganttproject.action.edit;
40
41 import ...
42
43 /**
44  * Action to show the options dialog for the application. It will search and show
45  * all available OptionPageProvider classes
46 */
47 2 usages ▲ dbarashev+2
48 public class SettingsDialogAction extends GPAction {
49     2 usages
50     private final UIFacade myUIFacade;
51     2 usages
52     private final IGanttProject myProject;
53
54     2 usages ▲ dbarashev
55     public SettingsDialogAction(IGanttProject project, UIFacade uiFacade) {
56         super( name: "settings.app");
57         myUIFacade = uiFacade;
58         myProject = project;
59     }
60
61     ▲ dbarashev+2
62     @Override
63     public void actionPerformed(ActionEvent e) {
64         if (calledFromAppleScreenMenu(e)) {
65             return;
66         }
67         SettingsDialog2 dialog = new SettingsDialog2(myProject, myUIFacade, pageOrderKey: "settings.app.pageOrder",
68             titleKey: "settings.app");
69         dialog.show();
70     }
71 }
```

```
19 package net.sourceforge.ganttproject.action.edit;
20
21 import ...
22
23 2 usages ▲ dbarashev+1
24 public class SearchDialogAction extends GPAction {
25     2 usages
26     private final Runnable mySearchUi;
27
28     1 usage ▲ dbarashev+1
29     public SearchDialogAction(Runnable searchUi) {
30         super( name: "search.dialog.open");
31         mySearchUi = searchUi;
32     }
33
34     ▲ dbarashev+1
35     @Override
36     public void actionPerformed(ActionEvent e) { mySearchUi.run(); }
37 }
```

```
19 package net.sourceforge.ganttproject.action.edit;
20
21 import ...
22
23 1 usage  ± dbarashev +1
24
25 public class RefreshViewAction extends GPAAction {
26
27     2 usages
28     private UIFacade myUIFacade;
29
30     1 usage  ± dbarashev
31     public RefreshViewAction(UIFacade uiFacade) {
32         super( name: "refresh");
33         myUIFacade = uiFacade;
34     }
35
36     ± Dmitry Barashev +1
37     @Override
38     public void actionPerformed(ActionEvent e) {
39         if (calledFromAppleScreenMenu(e)) {
40             return;
41         }
42         myUIFacade.refresh();
43     }
44     ± dbarashev
45     @Override
46     protected String getIconFilePrefix() { return "refresh_"; }
47
48 }
```

```

package net.sourceforge.ganttpoint.action.edit;

import ...


/*
 * Another card
 */
class A {
    A() {
        ...
    }
}

public class RedoAction extends GPAction implements GPUndoListener {
    @usage
    private final GPUndoManager myUndoManager;
    ...
    public RedoAction(GPUndoManager undoManager) { this(undoManager, IconSize.MENU); }

    @usage A doable
    private RedoAction(GPUndoManager undoManager, IconSize size) {
        super( name: "redo", size);
        myUndoManager = undoManager;
        myUndoManager.addUndoableEditListener(this);
        setEnabled(myUndoManager.canRedo());
    }

    A Dorothy Kostachev <!
    @Override
    public void actionPerformed(ActionEvent e) {
        if (calledFromAppleScreenMenu()) {
            return;
        }
        myUndoManager.redo();
    }

    A doable <!
    @Override
    public void undoOrRedoHappened(UndoableEditEvent e) {
        setEnabled(myUndoManager.canRedo());
        updateTooltip();
    }

    E usages & doables <!
    @Override
    public void undoOrRedoHappened() {
        setEnabled(myUndoManager.canRedo());
        updateTooltip();
    }

    A Dorothy Kostachev
    @Override
    public void undoReset() { undoOrRedoHappened(); }

    A doable <!
    @Override
    public String getLocalizedName() {
        if (myUndoManager == null || myUndoManager.canRedo() == false) {
            return super.getLocalizedName();
        }
        // use name of redoable action
        return myUndoManager.getRedoPresentationName();
    }
}

```

```

package net.sourceforge.ganttpoint.action.edit;

import ...

// TODO Enable/Disable action depending on clipboard contents
A Your Name <!
public class PasteAction extends GPAction {
    @usage
    private final GPViewManager myViewmanager;
    @usage
    private final GPUndoManager myUndoManager;
    @usage
    private final IGanttProject myProject;
    @usage
    private final UIFacade myUIFacade;

    E usages & doables <!
    public PasteAction(IGanttProject project, UIFacade uiFacade, GPViewManager viewManager, GPUndoManager undoManager) {
        super( name: "paste");
        myViewmanager = viewManager;
        myUndoManager = undoManager;
        myProject = project;
        myUIFacade = uiFacade;
    }

    A doable <!
    private PasteAction(IGanttProject project, UIFacade uiFacade, GPViewManager viewmanager, IconSize size, GPUndoManager undoManager) {
        super( name: "paste", size);
        myViewmanager = viewmanager;
        myUndoManager = undoManager;
        myProject = project;
        myUIFacade = uiFacade;
    }

    A Your Name <!
    @Override
    public void actionPerformed(ActionEvent evt) {
        if (calledFromAppleScreenMenu(evt)) {
            return;
        }
        ChartSelection selection = myViewmanager.getSelectedArtefacts();
        if (!selection.isEmpty()) {
            pasteInternalFlavor(selection);
            return;
        }
        var clipboardProject = ClipboardContentsKt.getProjectFromClipboard(new BufferProject(myProject, myUIFacade));
        if (clipboardProject != null) {
            try {
                myUndoManager.undoableEdit(getLocalizedName(), () -> pasteExternalDocument(clipboardProject));
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

```
1 package net.sourceforge.ganttproject.action.edit;
2
3 import ...
4
5 1 usage. A @WebService
6 public class EditMenu extends JMenu {
7     2 usages
8     private final UndoAction myUndoAction;
9     2 usages
10    private final RedoAction myRedoAction;
11    3 usages
12    private final SearchDialogAction mySearchAction;
13
14    1 usage. A @WebService
15    public EditMenu(IIGanttProject project, UIFacade uiFacade, GPViewManager viewManager, Runnable searchUi, String key) {
16        super(GPAction.createVoidAction(key));
17        final GPUndoManager undoManager = uiFacade.getUndoManager();
18        myUndoAction = new UndoAction(undoManager);
19        myRedoAction = new RedoAction(undoManager);
20        mySearchAction = new SearchDialogAction(searchUi);
21
22        add(getUndoAction());
23        add(getRedoAction());
24        addSeparator();
25        add(new RefreshViewAction(uiFacade));
26        add(mySearchAction);
27        addSeparator();
28        add(viewManager.getCutAction());
29        add(viewManager.getCopyAction());
30        add(viewManager.getPasteAction());
31        addSeparator();
32        add(new SettingsDialogAction(project, uiFacade));
33        setToolTipText(null);
34    }
35
36    A @WebService
37    @Override
38    public JMenuItem add(Action a) {
39        a.putValue(Action.SHOW_DESCRIPTION, null);
40        return super.add(a);
41    }
42
43    2 usages. A @WebService
44    public GPAction getUndoAction() { return myUndoAction; }
45
46    2 usages. A @WebService
47    public GPAction getRedoAction() { return myRedoAction; }
48
49    2 usages. A Direct Listener
50    public GPAction getSearchAction() { return mySearchAction; }
51}
```

```
1  //...
19 package net.sourceforge.ganttproject.action.edit;
20
21 import ...
22
23 //TODO Enable/Disable action on selection changes
24 6 usages ▲ Dmitry Barashev +2
25
26 public class CopyAction extends GPAction {
27     3 usages
28     private final GPViewManager myViewmanager;
29
30     2 usages ▲ dbarashev
31     public CopyAction(GPViewManager viewManager) {
32         super( name: "copy");
33         myViewmanager = viewManager;
34     }
35
36     ▲ dbarashev +1
37     @Override
38     public void actionPerformed(ActionEvent e) {
39         if (calledFromAppleScreenMenu(e)) {
40             return;
41         }
42         myViewmanager.getSelectedArtefacts().startCopyClipboardTransaction();
43     }
44
45     ▲ Dmitry Barashev
46     @Override
47     public CopyAction asToolbarAction() {
48         final CopyAction result = new CopyAction(myViewmanager);
49         result.setFontAwesomeLabel(UIUtil.getFontAwesomeLabel(result));
50         ▲ Dmitry Barashev
51         this.addPropertyChangeListener(new PropertyChangeListener() {
52             ▲ Dmitry Barashev
53             @Override
54             public void propertyChange(PropertyChangeEvent evt) {
55                 if ("enabled".equals(evt.getPropertyName())) {
56                     result.setEnabled((Boolean)evt.getNewValue());
57                 }
58             }
59         });
59         result.setEnabled(this.isEnabled());
60         return result;
61     }
62 }
```

```
1  //...
2  package net.sourceforge.ganttpoint.action.edit;
3
4  import ...
5
6  //TODO Enable/Disable action on selection changes
7  6 usages + Dmitry Barashev +3
8  public class CutAction extends GPAction {
9      3 usages
10     private final GPViewManager myViewmanager;
11     2 usages
12     private final GPUndoManager myUndoManager;
13
14     2 usages + dbarashev
15     public CutAction(GPViewManager viewManager, GPUndoManager undoManager) {
16         super(name: "cut");
17         myViewmanager = viewManager;
18         myUndoManager = undoManager;
19     }
20
21     + Dmitry Barashev +2
22     @Override
23     public void actionPerformed(ActionEvent e) {
24         if (calledFromAppleScreenMenu(e)) {
25             return;
26         }
27         //+
28         //+
29         public void run() {
30             myViewmanager.getSelectedArtefacts().startMoveClipboardTransaction();
31         }
32     }
33
34     + Dmitry Barashev
35     @Override
36     public CutAction asToolbarAction() {
37         final CutAction result = new CutAction(myViewmanager, myUndoManager);
38         result.setFontAwesomeLabel(UIUtil.getFontAwesomeLabel(result));
39         + Dmitry Barashev
40         this.addPropertyChangeListener(new PropertyChangeListener() {
41             + Dmitry Barashev
42             @Override
43             public void propertyChange(PropertyChangeEvent evt) {
44                 if ("enabled".equals(evt.getPropertyName())) {
45                     result.setEnabled((Boolean)evt.getNewValue());
46                 }
47             }
48         });
49         result.setEnabled(this.isEnabled());
50         return result;
51     }
52 }
```

```
31  */
32  * $ usages ▾ dbarashev +2
33  * public class UndoAction extends GPUAction implements GPUUndolistener {
34  *   10 usages
35  *   private final GPUUndoManager myUndoManager;
36  *
37  *   2 usages ▾ dbarashev
38  *   public UndoAction(GPUUndoManager undoManager) { this(undoManager, IconSize.MENU); }
39  *
40  *   1 usage ▾ dbarashev
41  *   @private UndoAction(GPUUndoManager undoManager, IconSize size) {
42  *     super( name: "undo", size.asString());
43  *     myUndoManager = undoManager;
44  *     myUndoManager.addUndoableEditListener(this);
45  *     setEnabled(myUndoManager.canUndo());
46  *   }
47  *
48  *   ▾ Dmitry Barashev +1
49  *   @Override
50  *   public void actionPerformed(ActionEvent e) {
51  *     if (calledFromAppleScreenMenu(e)) {
52  *       return;
53  *     }
54  *
55  *     myUndoManager.undo();
56  *   }
57  *
58  *   ▾ dbarashev +1
59  *   @Override
60  *   public void undoableEditHappened(UndoableEditEvent e) {
61  *     setEnabled(myUndoManager.canUndo());
62  *     updateTooltip();
63  *   }
64  *
65  *   3 usages ▾ dbarashev +1
66  *   @Override
67  *   public void undoOrRedoHappened() {
68  *     setEnabled(myUndoManager.canUndo());
69  *     updateTooltip();
70  *   }
71  *
72  *   1 usage ▾ Dmitry Barashev
73  *   @Override
74  *   public void undoReset() { undoOrRedoHappened(); }
75  *
```

Chapter 2 - Code Smells

Diogo Dias' Code Smells

Code Smell 1 - Duplicated Code

Location:

- biz.ganttproject.core\src\main\java\biz\ganttproject\core\chart\renderer\RectangleRenderer.java
- biz.ganttproject.core\src\main\java\biz\ganttproject\core\chart\renderer\PolygonRenderer.java
- biz.ganttproject.core\src\main\java\biz\ganttproject\core\chart\renderer\LineRenderer.java
- biz.ganttproject.core\src\main\java\biz\ganttproject\core\chart\renderer\SummaryTaskRenderer.java

Motive - All these classes have the same variables, one method equal and one method that is different in each class but has the same name and same purpose.

Solution - It's possible to eliminate this repetition using an abstract class, AbstractRenderer. This class should have the variables myProperties and myGraphics. In terms of methods, this class implements the method setGraphics and may have an abstract method, render, that allows each class' son to implement this method.

Code snippet -

```
public class SummaryTaskRenderer {  
    2 usages  
    private final Properties myProperties;  
    2 usages  
    private Graphics2D myGraphics;  
  
    1 usage  ▲ Dmitry Kazakov  
    public SummaryTaskRenderer(Properties props) { myProperties = props; }  
  
    ▲ Dmitry Kazakov  
    public void setGraphics(Graphics2D graphics) { myGraphics = graphics; }  
  
    ▲ Dmitry Kazakov  
    public void render(Canvas.Rectangle rect) {  
        Graphics2D g = (Graphics2D) myGraphics.create();  
        Style style = Style.getStyle(myProperties, rect.getStyle());  
  
        Style.Color background = style.getBackgroundColor(rect);  
        if (background != null) {  
            g.setColor(background.get());  
        }  
        Style.Padding padding = style.getPadding();  
        g.fillRect( x: rect.getLeftX() + padding.getLeft(), y: rect.getTopY() + padding.getTop(),
```

```
public class RectangleRenderer {  
    2 usages  
    private final Properties myProperties;  
    2 usages  
    private Graphics2D myGraphics;  
  
    1 usage  ▲ dbarashev  
    public RectangleRenderer(Properties props) { myProperties = props; }  
  
    ▲ dbarashev  
    public void setGraphics(Graphics2D graphics) { myGraphics = graphics; }  
  
    ▲ dbarashev  
    public boolean render(Canvas.Rectangle rect) {  
        Graphics2D g = (Graphics2D) myGraphics.create();  
        Style style = Style.getStyle(myProperties, rect.getStyle());  
  
        if (style.getVisibility(rect) == Style.Visibility.HIDDEN) {  
            return false;  
        }  
        Style.Color background = style.getBackgroundColor(rect);  
        if (background != null) {  
            g.setColor(background.get());  
        }  
        Paint paint = style.getBackgroundPaint(rect);  
        if (paint != null) {
```

```

public class PolygonRenderer {
    2 usages
    private final Properties myProperties;
    2 usages
    private Graphics2D myGraphics;

    1 usage  ↳ dbarashev
    public PolygonRenderer(Properties props) { myProperties = props; }

    ↳ dbarashev
    public void setGraphics(Graphics2D graphics) { myGraphics = graphics; }

    ↳ dbarashev
    public void render(Canvas.Polygon p) {
        Graphics2D g = (Graphics2D) myGraphics.create();
        Style style = Style.getStyle(myProperties, p.getStyle());

        if (style.getVisibility(p) == Style.Visibility.HIDDEN) {
            return;
        }
        Style.Color background = style.getBackgroundColor(p);
        if (background != null) {
            g.setColor(background.get());
        }
        g.fillPolygon(p.getPointsX(), p.getPointsY(), p.getPointCount());
    }
}

```

```

public class LineRenderer {
    2 usages
    private final Properties myProperties;
    2 usages
    private Graphics2D myGraphics;

    1 usage  ↳ dbarashev
    public LineRenderer(Properties props) { myProperties = props; }

    ↳ dbarashev
    public void setGraphics(Graphics2D graphics) { myGraphics = graphics; }

    1 usage  ↳ dbarashev +1
    public void renderLine(Canvas.Line line) {
        Graphics2D g = (Graphics2D) myGraphics.create();
        Style style = Style.getStyle(myProperties, line.getStyle());

        Style.Borders border = style.getBorder(line);
        g.setColor(border == null ? java.awt.Color.BLACK : border.getTop().getColor());
        g.setStroke(border == null ? Style.DEFAULT_STROKE : border.getTop().getStroke());
        g.drawLine(line.getStartX(), line.getStartY(), line.getFinishX(), line.getFinishY());
        if (line.getArrow() != Canvas.Arrow.NONE) {
            Canvas.Arrow arrow = line.getArrow();
            int xsign = Integer.signum( line.getFinishX() - line.getStartX());
            int ysign = Integer.signum( line.getFinishY() - line.getStartY());
        }
    }
}

```

Code Smell 2 - Message Chain

Location

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

Motive - This method is calling in chains methods that return other objects. The method `getProject` returns an `IGanttProject`. After that is called the method `getGanttOptions` that is implemented in `IGanttProject`, this method returns a `GanttOptions`. Finally, the object `GanttOptions` calls the method `getCSVOptions` which returns a `CSVOptions` object.

Solution - The way to resolve this problem the class `IGanttProject` may have a method that returns a `CSVOptions` object. After that, the class `ExporterToCSV` should use the method created in `IGanttProject` and create a new method that also returns `CSVOptions`. So we can use the class that we pretend to without knowing the complexity of the system.

Code snippet -

```
86
87     @Override
88     protected ExporterJob[] createJobs(final File outputFile, List<File> resultFiles) {
89         ExporterJob job = createCVSExportJob(outputFile);
90         resultFiles.add(outputFile);
91         return new ExporterJob[]{job};
92     }
93
94     private ExporterJob createCVSExportJob(final File outputFile) {
95         ExporterJob result = new ExporterJob(name: "Export project") {
96             @Override
97             protected IStatus run() {
98                 OutputStream outputStream = null;
99                 try {
100                     outputFile.createNewFile();
101                     outputStream = new BufferedOutputStream(new FileOutputStream(outputFile));
102                     CSVOptions csvOptions = ((GanttProject) getProject()).getGanttOptions().getCSVOptions();
```

Code Smell 3 - Speculative Generality

Location

ganttproject\src\main\java\net.sourceforge\ganttproject\PreferenceServiceimpl.java

Motive - This class is not implemented.

Solution - This class should be eliminated because at this moment it is not necessary for this app.

Code snippet

```
public class PreferenceServiceImpl implements IPreferencesService {  
  
    ▲ dbarashev  
    @Override  
    public IStatus applyPreferences(IExportedPreferences preferences) throws CoreException {  
        // TODO Auto-generated method stub  
        return null;  
    }  
  
    ▲ dbarashev  
    @Override  
    public void applyPreferences(IEclipsePreferences node, IPreferenceFilter[] filters) throws CoreException {  
        // TODO Auto-generated method stub  
    }  
  
    ▲ dbarashev  
    @Override  
    public IStatus exportPreferences(IEclipsePreferences node, OutputStream output, String[] excludesList)  
        throws CoreException {  
        // TODO Auto-generated method stub  
        return null;  
    }  
  
    ▲ dbarashev  
    @Override  
    public void exportPreferences(IEclipsePreferences node, IPreferenceFilter[] filters, OutputStream output)  
        throws CoreException {  
        // TODO Auto-generated method stub  
    }  
  
    ▲ dbarashev  
    @Override  
    public String get(String key, String defaultValue, Preferences[] nodes) {  
        // TODO Auto-generated method stub  
        return null;  
    }  

```

João Ribeiro's Code Smells

Code Smell 1 - Dead Code

Location

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

Motive - Commented code from previous versions that isn't being run and still is occupying space.

Solution - Delete the commented code.

Code

snippet

```
146     // public java.awt.Rectangle getBoundingClientRect(Task task) {  
147     //     java.awt.Rectangle result = null;  
148     //     TaskActivity[] activities = task.getActivities();  
149     //     for (int i = 0; i < activities.length; i++) {  
150     //         GraphicPrimitiveContainer.Rectangle nextRectangle = myTaskRendererImpl  
151     //             .getPrimitive(activities[i]);  
152     //         if (nextRectangle != null) {  
153     //             java.awt.Rectangle nextAwtRectangle = new java.awt.Rectangle(  
154     //                 nextRectangle.myLeftX, nextRectangle.myTopY,  
155     //                 nextRectangle.myWidth, nextRectangle.myHeight);  
156     //             if (result == null) {  
157     //                 result = nextAwtRectangle;  
158     //             } else {  
159     //                 result = result.union(nextAwtRectangle);  
160     //             }  
161     //         }  
162     //     }  
163     //     return result;  
164     // }  
165  
166     // GraphicPrimitiveContainer.Rectangle[] getTaskActivityRectangles(Task task)  
167     // {  
168     //     List<Rectangle> result = new ArrayList<Rectangle>();  
169     //     TaskActivity[] activities = task.getActivities();  
170     //     for (int i = 0; i < activities.length; i++) {  
171     //         GraphicPrimitiveContainer.Rectangle nextRectangle = myTaskRendererImpl  
172     //             .getPrimitive(activities[i]);  
173     //         if (nextRectangle!=null) {  
174     //             result.add(nextRectangle);  
175     //         }  
176     //     }
```

Code Smell 2 - Too many comments

Location

/ganttproject/src/main/java/biz.ganttproject/app/Internationalization.kt

Motive - Way too big of a comment for the size of the class.

Solution - Sum up some of the comments into a simpler and easy to read comment.

Code snippet -

```
107  /**
108   * This localizer allows for flexible use of shared resource bundles.
109  * When searching for a message by the given message key, it first prepends the rootKey prefix to the
110 * message key. If prefixed localizer is set, it is consulted first. This way we can just
111 * use shorter message keys for a group of logically related keys (e.g. use root key "exitDialog" and
112 * keys "title", "message", "ok" instead of "exitDialog.title", "exitDialog.message" and "exitDialog.ok").
113 *
114 * If root localizer is not set or returns no message, the message is searched by prefixed key in the local
115 * resource bundle of this localizer. In case of success it is formatted with MessageFormat, otherwise
116 * base localizer is consulted with original message key. This way we can use a pool of common messages
117 * which is shared between more specific localizers. E.g., for a set of dialogs where submit and cancel
118 * buttons are usually labeled with "OK" and "Cancel", we can create a shared base localizer L0 with keys
119 * "ok" and "cancel". For a dialog which requests user to accept some terms, we can create a localizer L1
120 * with root key "acceptTerms", key "acceptTerms.ok"="Accept" and L0 as a base localizer.
121 *
122 * When submit and cancel buttons in accept terms dialog are constructed, they will call localizer L1
123 * and pass "ok" and "cancel" keys. L1 will find "acceptTerms.ok" in its own bundle and will pass "cancel"
124 * to the base localizer.
125 *
126 * @author dbarashev@bardsoftware.com
127 */
128 open class DefaultLocalizer(
129     private val rootKey: String = "",
130     private val baseLocalizer: Localizer = DummyLocalizer,
131     private val prefixedLocalizer: Localizer? = null,
132     private val currentTranslation: SimpleObjectProperty< ResourceBundle? > = SimpleObjectProperty( initialValue: null) :
```

Code Smell 3 - Speculative Generality

Location

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

Motive - An auto generated method without implementation inside it. Could lead to errors and, for now, it's not doing anything.

Solution - Delete the method or fully implement it.

Code snippet -

```
168  4 usages  ▲ dbarashev
169  @Override
170  public void setVisibleTasks(List<Task> visibleTasks) {
171      // TODO Auto-generated method stub
172  }
```

Pedro Gasparinho's Code Smells

Code Smell 1 - Dead Code

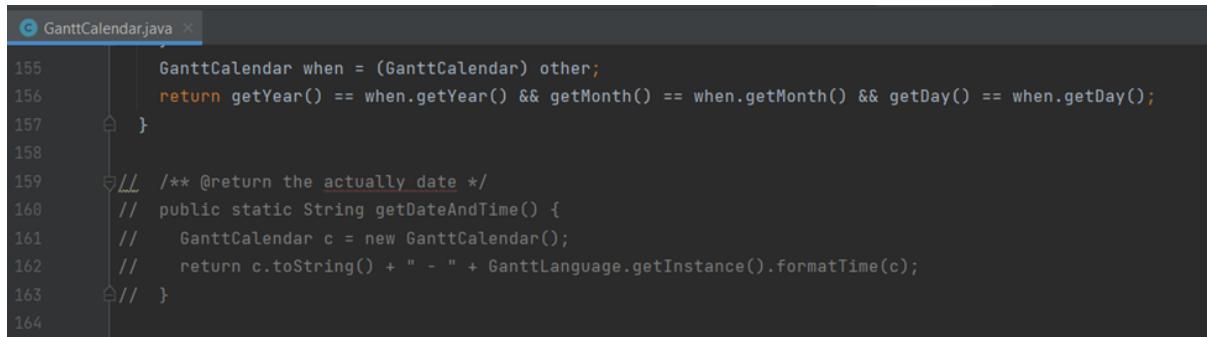
Location

ganttproject\biz.ganttproject.core\src\main\java\biz.ganttproject\core\time\\GanttCalender

Motive - When a method, such as this one, is commented it becomes obsolete since the functionality is either not supported anymore or implemented somewhere else, because of that the comments clutter the file where it's written.

Solution - Given the fact that this piece of code is commented and there is no indication of a reminder that it is still being worked on, I suggest that we delete those lines to improve file size and that class's readability.

Code snippet



```
GanttCalendar.java x
155     GanttCalendar when = (GanttCalendar) other;
156     return getYear() == when.getYear() && getMonth() == when.getMonth() && getDay() == when.getDay();
157 }
158
159 // /**
160 //  * @return the actually date */
161 // public static String getDateAndTime() {
162 //     GanttCalendar c = new GanttCalendar();
163 //     return c.toString() + " - " + GanttLanguage.getInstance().formatTime(c);
164 }
```

Code Smell 2 - Speculative Generality + Reminder Code

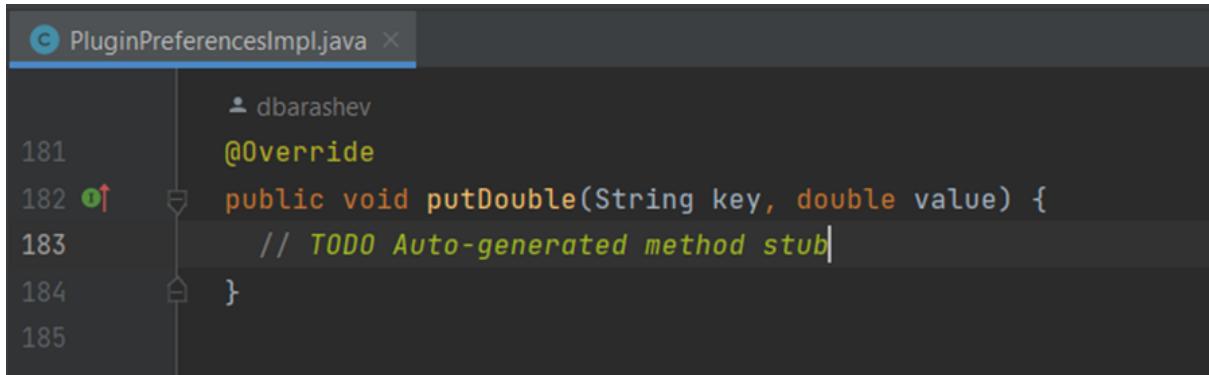
Location

ganttproject\ganttproject\src\main\java\net.sourceforge.ganttproject\PluginPreferencesImpl

Motive - The comment shown below on the codebase snippet indicates that the method is still being worked on, besides its auto generated, meaning that it must have been created by necessity of an Interface that the class implements.

Solution - In the previous version I suggested implementing this method, however it isn't used as of now, meaning it's an instance of a speculative generality code smell, so it's not a very good solution. As suggested by my colleague Afonso we should delete this method, however we first have to delete it from the interface since it's the only instance.

Code snippet



The screenshot shows a Java code editor with a file named 'PluginPreferencesImpl.java'. The code contains a single method:

```
181
182 ①    public void putDouble(String key, double value) {
183      // TODO Auto-generated method stub
184  }
```

The line '①' is highlighted with a red arrow pointing up, indicating it is the target for refactoring. The line '// TODO Auto-generated method stub' is also highlighted in green.

Code Smell 3 - Long Method

Location

ganttproject\biz.ganttproject.core\src\main\java\biz.ganttproject\core\time\impl\GPTTimeUnitStack

Motive - When looking at this method, it's easy to lose track of what it accomplishes and how so, not only because of the length by itself but also the accumulated complexity.

Solution - To refactor this piece of code what I suggest is to create a few auxiliary private methods to reduce the size of the original method and make it more digestible.

Code

snippet

```
① GPTimeUnitStack.java ×
160
161     @Override
162     public TimeDuration parseDuration(String lengthAsString) throws ParseException {
163         int state = 0;
164         StringBuffer valueBuffer = new StringBuffer();
165         Integer currentValue = null;
166         TimeDuration currentLength = null;
167         lengthAsString += " ";
168         for (int i = 0; i < lengthAsString.length(); i++) {
169             char nextChar = lengthAsString.charAt(i);
170             if (Character.isDigit(nextChar)) {
171                 switch (state) {
172                     case 0:
173                         if (currentValue != null) {
174                             throw new ParseException(lengthAsString, i);
175                         }
176                         state = 1;
177                         valueBuffer.setLength(0);
178                         valueBuffer.append(nextChar);
179                         break;
180                     case 1:
181                         TimeUnit timeUnit = findTimeUnit(valueBuffer.toString());
182                         if (timeUnit == null) {
183                             throw new ParseException(lengthAsString, i);
184                         }
185                         assert currentValue != null;
186                         TimeDuration localResult = createLength(timeUnit, currentValue.floatValue());
187
188                         if (currentLength == null) {
189                             currentLength = localResult;
190                         } else {
191                             if (currentLength.getTimeUnit().isConstructedFrom(timeUnit)) {
192                                 float recalculatedLength = currentLength.getLength(timeUnit);
193                                 currentLength = createLength(timeUnit, length: localResult.getValue() + recalculatedLength);
194                             } else {
195                                 throw new ParseException(lengthAsString, i);
196                             }
197                             state = 1;
198                             currentValue = null;
199                             valueBuffer.setLength(0);
200                             valueBuffer.append(nextChar);
201                             break;
202                         }
203                     } else if (Character.isWhitespace(nextChar)) {
204                         switch (state) {
205                             case 0:
206                             break;
207                             case 1:
208                             currentValue = Integer.valueOf(valueBuffer.toString());
209                             state = 0;
210                             break;
211                             case 2:
212                             TimeUnit timeUnit = findTimeUnit(valueBuffer.toString());
213                             if (timeUnit == null) {
214                                 throw new ParseException(lengthAsString, i);
215                             }
216                         }
217                     }
218                 }
219             }
220         }
221     }
```

```

216         assert currentValue != null;
217         TimeDuration localResult = createLength(timeUnit, currentValue.floatValue());
218         if (currentLength == null) {
219             currentLength = localResult;
220         } else {
221             if (currentLength.getTimeUnit().isConstructedFrom(timeUnit)) {
222                 float recalculatedLength = currentLength.getLength(timeUnit);
223                 currentLength = createLength(timeUnit, length: localResult.getValue() + recalculatedLength);
224             } else {
225                 throw new ParseException(lengthAsString, i);
226             }
227         }
228         state = 0;
229         currentValue = null;
230         break;
231     }
232     if (currentValue == null) {
233         switch (state) {
234             case 1:
235                 currentValue = Integer.valueOf(valueBuffer.toString());
236             case 0:
237                 if (currentValue == null) {
238                     throw new ParseException(lengthAsString, i);
239                 }
240                 state = 2;
241                 valueBuffer.setLength(0);
242             case 2:
243                 valueBuffer.append(nextChar);
244                 break;
245             }
246         }
247     if (currentValue != null) {
248         currentValue = Integer.valueOf(valueBuffer.toString());
249         TimeUnit dayUnit = findTimeUnit( code: "d");
250         currentLength = createLength(dayUnit, currentValue.floatValue());
251     }
252     return currentLength;
253 }
254

```

Tiago Meirim' Code Smells

Code Smell 1 - Dead Code

Location

[biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java](https://github.com/ganttproject/ganttproject/blob/main/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java)

Motive - The method is commented so it is probably an idea for a code that was not finished or didn't have purpose to the code at the moment.

Solution - Since the code is not being used in any part of the project, it should be erased to reduce code size and to make the class simpler.

Code snippet -

```
// public void setRightMarginBottomUnitCount(int value) {  
//     myRightMarginBottomUnitCount = value;  
// }
```

Code Smell 2 - Long Method

Location

biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilder.java

Motive - The constructor is not used to simplify the class giving it methods for every attribute that is saved in the class, it gives the code this unnecessary 53 lines.

Solution - Using the constructor as it should be we can initialize every attribute with only 12 lines saving us a total of 41 lines of code.

Code snippet

```
+ dbarashev  
protected Factory() {  
}
```

Code Smell 3 - Data Clumps

Location

biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java

Motive - There are a lot of parameters that appear numerous times in the methods.

Solution - Some of the parameters should be firstly stored in the class with the right conditions to be used and modified if needed like initialEnd.

Code

snippet

```

usage = <bottomUnit>
private void constructTopOffsets(TimeUnit timeUnit, List<Offset> topOffsets, List<Offset> bottomOffsets,
    int initialEnd, int baseUnitWidth) {
    int lastBottomOffset = bottomOffsets.get(bottomOffsets.size() - 1).getOffsetPixels();
    OffsetLookup offsetLookup = new OffsetLookup();
    Date currentDate = myStartDate;
    int prevEnd = initialEnd;
    int offsetEnd;
    do {
        TimeUnit concreteTimeUnit = getConcreteUnit(timeUnit, currentDate);
        Date endDate = concreteTimeUnit.adjustRight(currentDate);
        int bottomOffsetLowerBound = offsetLookup.lookupOffsetByEndDate(endDate, bottomOffsets);
        if (bottomOffsetLowerBound >= 0) {
            offsetEnd = bottomOffsets.get(bottomOffsetLowerBound).getOffsetPixels();
        } else {
            if (-bottomOffsetLowerBound > bottomOffsets.size()) {
                offsetEnd = lastBottomOffset + 1;
            } else {
                Offset ubOffset = bottomOffsetLowerBound <= -2 ? bottomOffsets.get(-bottomOffsetLowerBound - 2) : null;
                Date ubEndDate = ubOffset == null ? myStartDate : ubOffset.getOffsetEnd();
                int ubEndPixel = ubOffset == null ? 0 : ubOffset.getOffsetPixels();
                WorkingUnitCounter counter = new WorkingUnitCounter(GPCalendarCalc.PLAIN, baseUnit);
                offsetEnd = ubEndPixel + counter.run(ubEndDate, endDate).getLength() * baseUnitWidth;
            }
        }
        topOffsets.add(Offset.createFullyClosed(concreteTimeUnit, myStartDate, currentDate, endDate, prevEnd, endPixels: initialEnd
            + offsetEnd, DayMask.WORKING));
        prevEnd = initialEnd + offsetEnd;
        currentDate = endDate;
    } while (offsetEnd <= lastBottomOffset && (myEndDate == null || currentDate.before(myEndDate)));
}

```

```

usage = <bottomUnit>
void constructBottomOffsets(List<Offset> offsets, int initialEnd) {
    int marginUnitCount = myRightMarginBottomUnitCount;
    Date currentDate = myStartDate;
    int shift = 0;
    OffsetStep step = new OffsetStep();
    int prevEnd = initialEnd;
    do {
        TimeUnit concreteTimeUnit = getConcreteUnit(getBottomUnit(), currentDate);
        calculateNextStep(step, concreteTimeUnit, currentDate);
        Date endDate = concreteTimeUnit.adjustRight(currentDate);
        if (endDate.compareTo(myViewportStartDate) <= 0) {
            shift = (int) (step.parrots * getDefaultUnitWidth());
        }
        int offsetEnd = (int) (step.parrots * getDefaultUnitWidth()) - shift;
        Offset offset = Offset.createFullyClosed(concreteTimeUnit, myStartDate, currentDate, endDate,
            prevEnd, endPixels: initialEnd + offsetEnd, step.dayMask);
        prevEnd = initialEnd + offsetEnd;
        offsets.add(offset);
        currentDate = endDate;

        boolean hasNext = true;
        if (offsetEnd > getChartWidth()) {
            hasNext &= marginUnitCount-- > 0;
        }
        if (hasNext && myEndDate != null) {
            hasNext &= currentDate.before(myEndDate);
        }
        if (!hasNext) {
            return;
        }
    } while (true);
}

```

```
1 usage  ± dbarashev
void constructOffsets(List<Offset> topUnitOffsets, List<Offset> bottomUnitOffsets, int initialEnd) {

    // bottomUnitOffsets.add(new Offset(getBottomUnit(), myStartDate,
    // myStartDate, myStartDate, 0, GPCalendar.DayType.WORKING));
    constructBottomOffsets(bottomUnitOffsets, initialEnd);
    if (topUnitOffsets != null) {
        constructTopOffsets(getTopUnit(), topUnitOffsets, bottomUnitOffsets, initialEnd, getDefaultUnitWidth());
    }
}
```

Afonso Nunes' Code Smells

Code Smell 1 - No comments

Location

[biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/CalendarActivityImpl.java](https://github.com/biz/ganttproject/blob/master/src/main/java/biz/ganttproject/core/calendar/CalendarActivityImpl.java)

Motive - When a class on a big project like in this case (or maybe even in a small one) has no comments, it becomes difficult to understand what it is supposed to do.

Solution - Add a simple class comment on what the class should actually be doing.

Code snippet

```
1 // Generated on 18.10.2004
2 // 
3 package biz.ganttproject.core.calendar;
4 
5 import java.util.Date;
6 
7 /**
8  * @author hard
9  */
10 
11 public class CalendarActivityImpl implements GPCalendarActivity {
12 
13     private final boolean isWorkingTime;
14 
15     private final Date myEndDate;
16 
17     private final Date myStartDate;
18 
19     /**
20      * @param startDate
21      * @param endDate
22      * @param isWorkingTime
23      */
24     public CalendarActivityImpl(Date startDate, Date endDate, boolean isWorkingTime) {
25         myStartDate = startDate;
26         myEndDate = endDate;
27         this.isWorkingTime = isWorkingTime;
28     }
29 
30     /**
31      * @return
32      */
33     @Override
34     public Date getStart() { return myStartDate; }
35 
36     /**
37      * @return
38      */
39     @Override
40     public Date getEnd() { return myEndDate; }
41 
42     /**
43      * @return
44      */
45     @Override
46     public boolean isWorkingTime() { return isWorkingTime; }
47 
48     /**
49      * @see java.lang.Object#toString()
50      */
51     @Override
52     public String toString() {
53         return (isWorkingTime() ? "Working time: " : "Holiday: ") + "[" + getStart() + ", " + getEnd() + "]";
54     }
55 }
```

Code Smell 2 - Speculative Generality

Location

[biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/AlwaysWorkingTimeCalendarImpl.java](https://github.com/robert-hibberd/refactoring-guru/blob/master/java/design-patterns/creational/always-working-time/src/main/java/biz/ganttproject/core/calendar/AlwaysWorkingTimeCalendarImpl.java)

Motive - This piece of code indicates that the original writer implements an interface that has those methods declared, but he does not need to implement them in this specific class.

Solution - After a discussion with my colleague, he made me realize that I was making a foolish mistake and that I didn't pay enough attention. Since we are already extending one abstract class we can just implement those methods there and delete them from the subclass by extending that existing abstract class.

Code

snippet

```
└ dbarashev
@Override
public void setPublicHolidays(Collection<CalendarEvent> holidays) {
}
```

```
@Override
public void setBaseCalendarID(String id) {
}
```

Code Smell 3 - Data Class

Location

[biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/CalendarActivityImpl.java](#)

Motive - This is a very small class consisting of only data with no real functionality besides getter methods and a simple `toString()` method.

Solution - Defining these methods in another class where all this data is easily accessible from and where it makes more sense.

Code snippet -

```
package biz.ganttproject.core.calendar;

import java.util.Date;

/**
 * @author bard
 */
17 usages ▲ dbarashev
public class CalendarActivityImpl implements GPCalendarActivity {

    2 usages
    private final boolean isWorkingTime;

    2 usages
    private final Date myEndDate;

    2 usages
    private final Date myStartDate;

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

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

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

    3 usages ▲ dbarashev
    @Override
    public boolean isWorkingTime() { return isWorkingTime; }

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