

Code Smells

Francisco Vale:

Code Smell 1: Dead Code/Speculative Generality

```
42
43     protected UIFacade getUiFacade() { return myUiFacade; }
44
45     protected IGanttProject getProject() { return myProject; }
46
47     protected Preferences getPreferences() { return myPreferences; }
48
49     protected Chart getGanttChart() { return myUiFacade.getGanttChart(); }
50
51     protected Chart getResourceChart() { return myUiFacade.getResourceChart(); }
52
53     }
54
55 }
```

Location:

ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > AbstractEngine

Line 55 and 59;

Synopsis: Dead Code is found when we have obsolete code that is never called throughout the entire project, which is similar to Speculative generality, but here the code is made with future intentions, even tho it is not being used at the moment.

Rationale: This class has methods that are never called in the entire code, which makes this obsolete, even if it is with future intentions, if is not something requested from the client then it is only a waste of time .

Refactor proposal: I would delete this piece of code since it is not being used and only makes the code more extensive.

Code Smell 2: Data Class

```
1  /...
19 package org.ganttproject.impex.htmlpdf;
20
21 import ...
22
23 /**
24  * Simple base class for the rendering engines.
25  *
26  * @author dbarashev (Dmitry Barashev)
27 */
28
29 public class AbstractEngine {
30     private IGanttProject myProject;
31     private UIFacade myUiFacade;
32     private Preferences myPreferences;
33
34     public void setContext(IGanttProject project, UIFacade uiFacade, Preferences preferences) {
35         myProject = project;
36         myUiFacade = uiFacade;
37         myPreferences = preferences;
38     }
39
40     protected UIFacade getUiFacade() { return myUiFacade; }
41
42     protected IGanttProject getProject() { return myProject; }
43
44     protected Preferences getPreferences() { return myPreferences; }
45
46     protected Chart getGanttChart() { return myUiFacade.getGanttChart(); }
47
48     protected Chart getResourceChart() { return myUiFacade.getResourceChart(); }
49
50 }
```

Location:

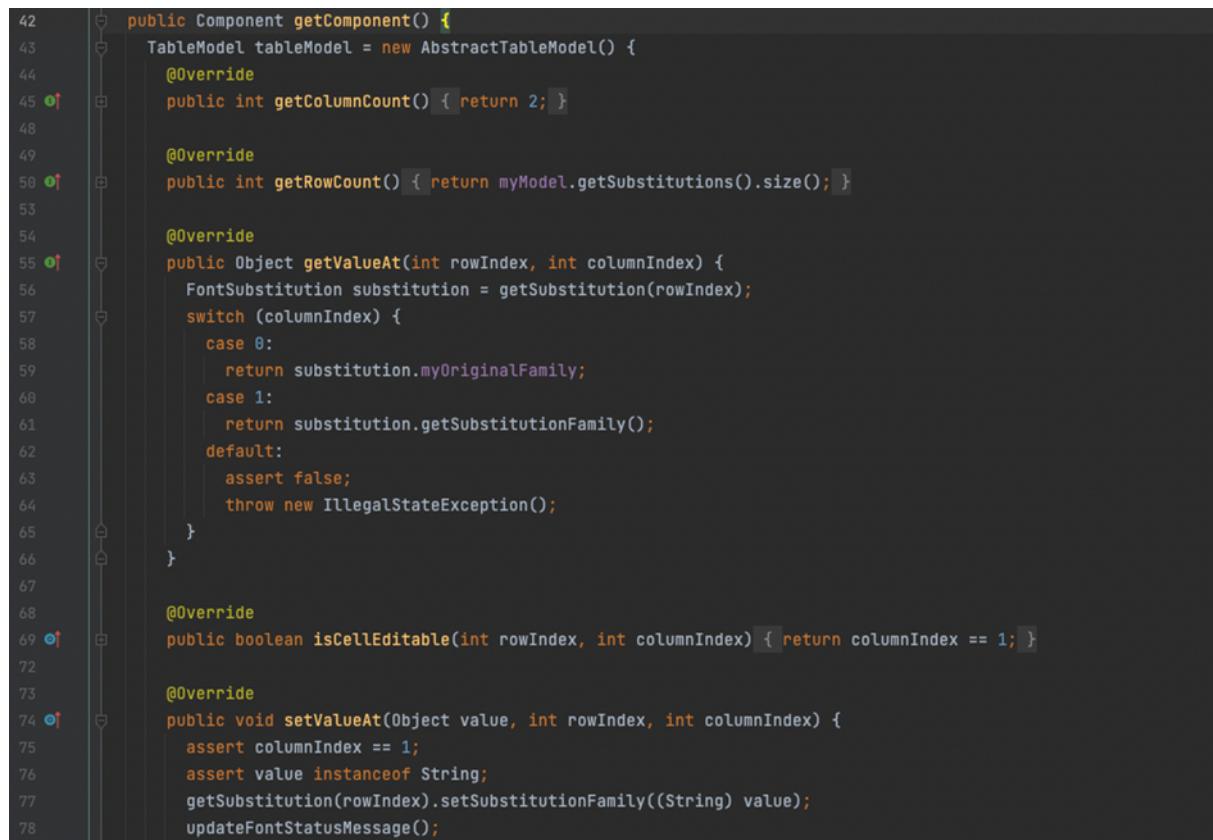
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf >  AbstractEngine

Synopsis: A Data Class is a class in which we only have getter and setter methods, and so it has no real functionality, which may indicate it may not be a good abstraction, or a necessary class.

Rationale: As we can observe the entire class as nothing but get methods.

Refactor: I would suggest trying to understand if there is something other than getters that would make sense to be placed in this class or, on the other hand, trying to see if the classes that use this gets can place this behaviours in their own class.

Code Smell 3: Long Method



A screenshot of a Java code editor showing a long method named `getComponent`. The code is annotated with several red and orange markers, indicating potential issues or warnings. The method implements several methods from the `TableModel` interface, including `getColumnCount`, `getRowCount`, `getValueAt`, `isCellEditable`, and `setValueAt`. The code uses a switch statement to handle the column index, asserting that the column index is 1, and updating a font status message.

```
42     public Component getComponent() {
43         TableModel tableModel = new AbstractTableModel() {
44             @Override
45             public int getColumnCount() { return 2; }
46
47             @Override
48             public int getRowCount() { return myModel.getSubstitutions().size(); }
49
50             @Override
51             public Object getValueAt(int rowIndex, int columnIndex) {
52                 FontSubstitution substitution = getSubstitution(rowIndex);
53                 switch (columnIndex) {
54                     case 0:
55                         return substitution.myOriginalFamily;
56                     case 1:
57                         return substitution.getSubstitutionFamily();
58                     default:
59                         assert false;
60                         throw new IllegalStateException();
61                 }
62             }
63
64             @Override
65             public boolean isCellEditable(int rowIndex, int columnIndex) { return columnIndex == 1; }
66
67             @Override
68             public void setValueAt(Object value, int rowIndex, int columnIndex) {
69                 assert columnIndex == 1;
70                 assert value instanceof String;
71                 getSubstitution(rowIndex).setSubstitutionFamily((String) value);
72                 updateFontStatusMessage();
73             }
74         };
75     }
```

Location:

ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > itext > FontSubstitutionPanel > getComponent

From line 42 to 134;

Synopsis: A Long Method, as its name suggests, is a method which is too long, and so, it may suggest that the method is too complex and does too much stuff that it probably didn't need.

Rationale: As we can see in the previous image the code is really extensive, and it continues for sixty more lines. Also it does various operations that probably could be divided throughout the code.

Refactor: As I said before this method deals with a lot of operations, and so I would try to divide them and try to put them in more adequate spots, rather than being all in the same method.

Guilherme Franco:

Code Smell 1: Dead Code/Speculative Generality

```
└─ dbarashev
public static CustomPropertyClass fromJavaClass(Class<?> javaClass) {
    for (CustomPropertyClass klass : CustomPropertyClass.values()) {
        if (klass.getJavaClass().equals(javaClass)) {
            return klass;
        }
    }
    return null;
}
```

Location:

Grantt > biz.ganttproject.core > src > main > java > biz > ganttproject > core > OperationStatus

Line 66;

Rationale: This method is not used or called anywhere in the application.

Refactor proposal: I believe this was added to be used in the future but since it's not used anywhere at the moment and there is no current need for it, I would delete this piece of obsolete code.

Code Smell 2: Message Chains

```
2 usages  ± oihanealbizuri +3
@Override
protected void doInit() {
    super.doInit();
    myResourceTreeModel.updateResources();
    getModel().addTableModelListener(new ModelListener());
    getVerticalScrollBar().addAdjustmentListener(vscrollController.get());
    getTableHeader().addMouseListener((MouseAdapter) mouseClicked(mouseEvent) → {
        if (!mouseEventHandling(mouseEvent)) {
            return;
        }

        final TableHeaderUiFacadeImpl tableHeader = getTableHeaderUiFacade();
        final ColumnImpl column = tableHeader.findColumnByViewIndex(getTable().columnAtPoint(mouseEvent.getPoint()));
        final ResourceDefaultColumn resourceColumn = ResourceDefaultColumn.find(column.getID());

        ± oihanealbizuri +1
        getUiFacade().getUndoManager().undoableEdit(GanttLanguage.getInstance().getText(key: "task.sort"), new Runnable() {
```

Location:

Grantt > ganttproject > src > main > java > net > sourceforge > ganttproject > C ResourceTreeTable > m doInit()

Line 152 or 155

Rationale: In line 152 the code calls 4 successive objects just to receive the column.

In line 155 the code calls a total of 5 successive objects.

Refactor: I would suggest structuring the classes differently, so they don't have to call so many objects or at least creating local variables so it's more understandable.

Code Smell 3: Duplicated Code

1)

```
2 usages  ± dbarashov
@Override
protected TableColumnExt newTableColumnExt(int modelIndex) {
    TableColumnExt tableColumn = super.newTableColumnExt(modelIndex);
    if (modelIndex == ResourceDefaultColumn.ROLE.ordinal() || modelIndex == ResourceDefaultColumn.ROLE_IN_TASK.ordinal()) {
        JComboBox comboBox = new JComboBox(getRoleManager().getEnabledRoles());
        comboBox.setEditable(false);
        tableColumn.setCellEditor(new DefaultCellEditor(comboBox));
    }
    return tableColumn;
}
```

2)

```
71      2 usages  ↗ dbarashev
72      private void setEditor(ColumnImpl column) {
73          if (column == null || column.getTableColumnExt() == null) {
74              return;
75          }
76          JComboBox comboBox = new JComboBox(getRoleManager().getEnabledRoles());
77          comboBox.setEditable(false);
78          column.getTableColumnExt().setCellEditor(new DefaultCellEditor(comboBox));
79      });

```

Location:

[Grantt](#) > [ganttproject](#) > src > main > java > net > sourceforge > [ganttproject](#) >  ResourceTreeTable

On lines 75-76 and 194-195

Rationale: Inside these two methods there is duplicated code so if we needed to alter the method of creating “JcomboBoxes” we would need to do it in two different places

Refactor: I would create a new method that creates these jumbo boxes and just call that method inside “setEditor” and “newTableColumnExt”.

Luís Abreu:

Code Smell 1: Data Class

```
 9  ▲ dbarashev
10 public class TimeUnitPair {
11     2 usages
12     private final TimeUnit myBottomTimeUnit;
13
14     2 usages
15     private final TimeUnit myTopTimeUnit;
16
17     2 usages
18     private final TimeUnitStack myTimeUnitStack;
19
20     /** Used scale for this TimeUnit */
21     2 usages
22     private final int myDefaultUnitWidth;
23
24     10 usages ▲ dbarashev
25     public TimeUnitPair(TimeUnit topUnit, TimeUnit bottomUnit, TimeUnitStack timeUnitStack, int defaultUnitWidth) {
26         myTopTimeUnit = topUnit;
27         myBottomTimeUnit = bottomUnit;
28         myTimeUnitStack = timeUnitStack;
29         myDefaultUnitWidth = defaultUnitWidth;
30     }
31
32     2 usages ▲ dbarashev
33     □ public TimeUnit getTopTimeUnit() { return myTopTimeUnit; }
34
35     2 usages ▲ dbarashev
36     □ public TimeUnit getBottomTimeUnit() { return myBottomTimeUnit; }
37
38     ▲ dbarashev
39     □ public TimeUnitStack getTimeUnitStack() { return myTimeUnitStack; }
40
41     /** @return the scale for this TimeUnit */
42     1 usage ▲ dbarashev
43     □ public int getDefaultUnitWidth() { return myDefaultUnitWidth; }
44 }
```

Location:

ganttproject > biz.ganttproject.core > src > main > java > biz > ganttproject > core > time > TimeUnitPair

Lines: whole class, but more important methods in between 26-39

Rationale: Inside this class there is only data, no real functionality, only getter and setters methods.

Refactor: I would create a better abstraction: see if i could put something else in this class (other methods), see what classes are manipulating this data. If I failed to solve the problem, I would delete this class and try to figure out if this data would be better store in another class.

Code Smell 2: Long Method

```
59      ▲ dbarashev
60  ↳ public JComponent getComponent() {
61    ↳   if (myPanel == null) {
62      ↳     SpringLayout topPanelLayout = new SpringLayout();
63      ↳     JPanel topPanel = new JPanel(topPanelLayout);
64
65      ↳     JComponent depsComponent = myList.getTableComponent();
66      ↳     JComponent titleComponent = new JLabel(myList.getTitle());
67      ↳     JComponent actionsComponent = myList.getActionsComponent();
68      ↳     topPanel.add(titleComponent);
69      ↳     topPanel.add(actionsComponent);
70
71      ↳     topPanelLayout.putConstraint(SpringLayout.WEST, titleComponent, pad: 0, SpringLayout.WEST, topPanel);
72      ↳     topPanelLayout.putConstraint(SpringLayout.NORTH, titleComponent, pad: 0, SpringLayout.NORTH, topPanel);
73
74      ↳     topPanelLayout.putConstraint(SpringLayout.NORTH, actionsComponent, pad: 2, SpringLayout.SOUTH, titleComponent);
75      ↳     topPanelLayout.putConstraint(SpringLayout.SOUTH, topPanel, pad: 2, SpringLayout.SOUTH, actionsComponent);
76      ↳     topPanelLayout.putConstraint(SpringLayout.WEST, actionsComponent, pad: 0, SpringLayout.WEST, topPanel);
77
78      ↳     JPanel centerPanel = new JPanel(new BorderLayout());
79      ↳     centerPanel.add(depsComponent, BorderLayout.CENTER);
80
81      ↳     myFields.setBorder(BorderFactory.createEmptyBorder( top: 0, left: 10, bottom: 0, right: 0));
82      ↳     centerPanel.add(myFields, BorderLayout.EAST);
83
84      ↳     myPanel = Box.createVerticalBox();
85      ↳     myPanel.setBorder(BorderFactory.createEmptyBorder( top: 5, left: 5, bottom: 5, right: 5));
86      ↳     myPanel.add(topPanel);
87      ↳     myPanel.add(Box.createVerticalStrut( height: 5));
88      ↳     myPanel.add(centerPanel);
89    ↳   }
90
91    ↳   return myPanel;
92  ↳ }
93 }
```

Location:

ganttpoint > ganttpoint > src > main > java > net > sourceforge > ganttpoint > gui > ListAndFieldsPanel

Lines: 40-72

Rationale: This method is way too big, which can lead to bad understanding of the code and can be a sign of being more complex than it needs to be.

Refactor: I would create smaller private methods that would be called inside the method `public JComponent getComponent()`.

This method will be better documented and will provide better understanding of the code, for example one of this smaller private methods could be named `putConstraint()` and would be responsible for the lines 51-56.

Code Smell 3: Duplicated Code

```
184     /** Move Up the selected resource */
185     1 usage  ↗ dbarashev
186     public boolean moveUp(HumanResource resource) {
187         myResourceManager.up(resource);
188         ResourceNode rn = getNodeForResource(resource);
189         int index = TreeUtil.getPrevSibling(root, rn);
190         if (index == -1) {
191             return false;
192         }
193         removeNodeFromParent(rn);
194         insertNodeInto(rn, root, index);
195         return true;
196     }
197
198     1 usage  ↗ dbarashev
199     public boolean moveDown(HumanResource resource) {
200         myResourceManager.down(resource);
201         ResourceNode rn = getNodeForResource(resource);
202         int index = TreeUtil.getNextSibling(root, rn);
203         if (index == -1) {
204             return false;
205         }
206         removeNodeFromParent(rn);
207         insertNodeInto(rn, root, index);
208     }
```

Location:

ganttproject > ganttproject > src > main > java > net > sourceforge > ganttproject >  ResourceTreeTableModel >  moveUp

Lines: 189-194 and 201-206

Rationale: Inside these two red boxes there is duplicated code so if we needed to alter something in the way the code deals with the fact that there is or is not a previous/next sibling we would need to do it in two different places.

Refactor: I would create a new method that has this code and call inside both methods *moveUp()* and *moveDown()*. By doing this the program would perform the same tasks but with less code.

Pedro Gouveia:

1- Switch statements

```
switch (type) {  
    case PertChartAbstraction.Type.NORMAL:  
        color = NORMAL_COLOR;  
        break;  
    case PertChartAbstraction.Type.SUPER:  
        color = SUPER_COLOR;  
        break;  
    case PertChartAbstraction.Type.MILESTONE:  
        color = MILESTONE_COLOR;  
        break;  
    default:  
        color = NORMAL_COLOR;  
}
```

```
static class Type {  
    2 usages  
    public static final int NORMAL = 0;  
  
    2 usages  
    public static final int SUPER = 1;  
  
    2 usages  
    public static final int MILESTONE = 2;  
}
```

Location:

ganttproject > org.ganttproject.chart.pert > src > main > java > org > ganttproject > chart > pert > ActivityOnNodePertChart > GraphicalNode > paintMe

Rationale: here the type of graph is determined by an integer and then there are a bunch of switch statements regarding the behavior of such graph depending on the type of graph it is. This should be resolved by using polymorphism to specify the type of graph. Also, if there is the need to add functionalities to different types of graphs we would need to add a switch statement for which, and if we want to add a new graph we would need to change every scattered switch statement.

Refactor: Should be created an abstract class from which 3 different subclasses inherit.

2- Law of demeter

74 | } else if (myTaskManager.getTaskHierarchy().getNestedTasks(task).length == 0) {

Location:

ganttproject > org.ganttproject.chart.pert > src > main > java > org > ganttproject > chart > pert > PertChartAbstraction > getTaskGraphNode

Rationale: Here *PertChartAbstraction* class in its method *getTaskGraphNode()* is calling a method named *getNestedTasks()*, and this method is not:

- Encapsulated within the same object – the method *getNestedTasks()*, is from an object of the type *TaskContainmentHierarchyFacade*, so it obviously isn't from the same class as *getTaskGraphNode()*, or in other words is not from *PertChartAbstraction* class.

- Encapsulated within an object that is in the parameters of M - the object from which this method is from is received from the method `getTaskHierarchy()` of the `myTaskManager` object, and this object is not from the parameters of `getNestedTasks()`, it is just returned from a function.
- Encapsulated within an object that is instantiated inside the M - the object from which this method is from is received from the method `getTaskHierarchy()` of the `myTaskManager` object and this object is not instantiated inside `getNestedTasks()`, it is just returned from a function.
- Encapsulated within an object that is referenced in an instance variable of the class in M - the object from which this method is from is received from the method `getTaskHierarchy()` of the `myTaskManager` object and not from an instance variable of the class `PerChartAbstraction`.

Refractor: There should be a method in task manager that returned the nested tasks.

3-Dead Code

```

1 package biz.ganttproject.impex.msproject2;
2
3 ± dbarashev
4 public class WebStartIDClass {
5
6 }
```

Location:

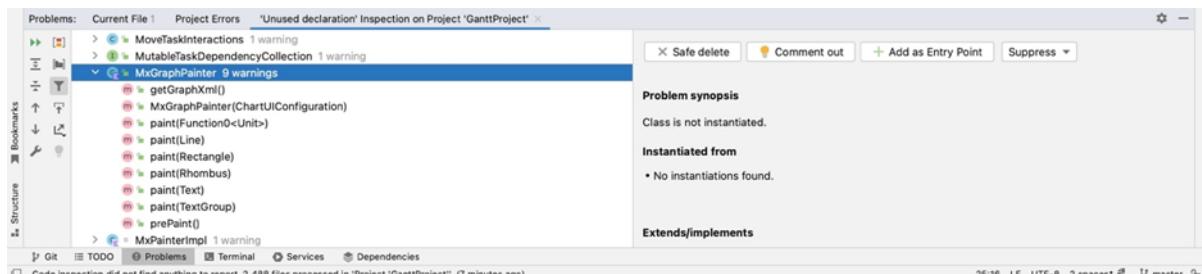
[ganttproject](#) > [biz.ganttproject.impex.msproject2](#) > [src](#) > [main](#) > [java](#) > [biz](#) > [ganttproject](#) > [impex](#) > [msproject2](#) > [WebStartIDClass](#)

Rationale: This class doesn't have anything and is not implemented anywhere or used. It was probably created to implement something that stopped making sense before it was implemented or is a placeholder for something to do in the future.

Refractor: This class should not exist before it is needed.

Tiago Francisco:

Code smell 1: Dead code:



Location:

ganttproject > biz.ganttproject.mxgraph > src > main > kotlin > biz > ganttproject > mxgraph > MxGraphPainter

Rationale: This class is not used or instantiated anywhere in the application.

Refactor proposal: It might be a class that is not used anymore, so it should be deleted.

Code smell 2: Speculative Generality:

```
89     private void run0(Function<CustomPropertyDefinition, S> fxnTaskField) {
90         for (Iterator<CustomPropertyDefinition> it = allDefs.iterator(); it.hasNext(); ) {
91             CustomPropertyDefinition def = it.next();
92             try {
93                 FieldType tf = fxnTaskField.apply(def);
94                 if (tf != null) {
95                     result.put(def, tf);
96                     mpxjFields.remove(tf);
97                     it.remove();
98                 }
99             } catch (IllegalArgumentException e) {
100                 // That's somewhat okay. We have not found such value in the enum, but it might come from the future
101                 // versions of MPXJ, so it is not the reason to fail
102             }
103         }
104     }
```

Location:

ganttproject > biz.ganttproject.impex.msproject2 > src > main > java > biz > ganttproject > impex > msproject2 > CustomPropertyMapping

Line 99:

Rationale: This catch clause was written thinking in something that might happen in the future but doesn't happen at this moment.

Refactor proposal: It should be deleted to make the code clearer.

Code Smell 3: Law of Demeter

72

```
float scaleFactor = fontSpec.getSize().getFactor() * getDpi();
```

Location:

ganttpoint > org.ganttpoint.chart.pert > src > main > java > org > ganttpoint > chart > pert > PertChart > updateFonts

Rationale: Here *PertChart* class in its method *updateFonts()* is calling a method named *getFactor()*, and this method is not:

Encapsulated within the same object – the method *getFactor()*, is from an object of the type *Size*, so it obviously isn't from the same class as *updateFonts()*, or in other words is not from *PertChart* class.

Encapsulated within an object that is in the parameters of *updateFonts()* - the object from which this method is from is received from the method *getSize()* of the *fontSpec* object, and this object is not from the parameters of *updatedFonts()*, it is just returned from a function.

Encapsulated within an object that is instantiated inside *updateFonts()*- the object from which this method is from is received from the method *getSize()* of the *fontSpec* object and this object is not instantiated inside *updateFonts()*, it is just returned from a function.

Encapsulated within an object that is referenced in an instance variable of the class in *updateFonts()* - the object from which this method is from is received from the method *getSize()* of the *fontSpec* object and not from an instance variable of the class *PertChart*.

Refactor proposal: I would create a method in the *FontSpec* class that returns the factor since the object is received from the constructor parameters in that class.

Design Patterns

Francisco Vale:

Design Pattern 1: Template Method Pattern

SuperClass:

```
1  /...
19 package org.ganttproject.impex.htmlpdf;
20
21 import ...
22
23
24 public abstract class StylesheetExporterBase extends ExporterBase {
25
26     private GPOptionGroup myOptions;
27
28     @Override
29     protected EnumerationOption createStylesheetOption(String optionID, final List<Stylesheet> stylesheets) {
30         final List<String> names = new ArrayList<>();
31         for (Stylesheet s : stylesheets) {
32             names.add(s.getLocalizedString());
33         }
34         EnumerationOption stylesheetOption = new DefaultEnumerationOption<Stylesheet>(optionID, names) {
35             @Override
36             public void commit() {
37                 super.commit();
38                 String value = getValue();
39                 int index = names.indexOf(value);
40                 if (index >= 0) {
41                     setSelectedStylesheet(stylesheets.get(index));
42                 }
43             }
44         };
45         return stylesheetOption;
46     }
47 }
```

SubClasses:

```
1  .../
19 package org.ganttproject.impex.htmlpdf;
20
21 import ...
22
23 public class ExporterToHTML extends StylesheetExporterBase {
24     static final String GANTT_CHART_FILE_EXTENSION = "png";
25     static final String RESOURCE_CHART_FILE_EXTENSION = "res.png";
26     private static final String PNG_FORMAT_NAME = "png";
27     private HTMLStylesheet mySelectedStylesheet;
28     private final HtmlSerializer mySerializer = new HtmlSerializer( engine: this);
29
30 }
```

```
1  .../
19 package org.ganttproject.impex.htmlpdf;
20
21 import ...
22
23
24 public class ExporterToPDF extends StylesheetExporterBase {
25
26     private final ITextEngine myITextEngine = new ITextEngine( exporter: this);
27     private Stylesheet mySelectedStylesheet;
28
29     @Override
30     protected ExporterJob[] createJobs(File outputFile, List<File> resultFiles) {
31         super.setCommandLineStylesheet();
32     }
33
34 }
```

Locations:

```
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > StylesheetExporterBase
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > ExporterToHTML
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > ExporterToPDF
```

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

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

Design Pattern 2: Chain of Responsibility Design Pattern

ITextEngine:

```
94
95     private Component createFontPanel() {
96         return new FontSubstitutionPanel(mySubstitutionModel).getComponent();
97     }
98 }
```

FontSubstitutionPanel:

```
119
120     myFamiliesComboBox = new JComboBox(myModel.getAvailableSubstitutionFamilies().toArray(new String[0]));
121     table.getColumnModel().getColumn( columnIndex: 0 ).setCellRenderer(new CellRendererImpl());
```

(Inside getComponent method in line 42)

FontSubstitutionModel:

```
109     public List<String> getAvailableSubstitutionFamilies() {
110         return myFontCache.getRegisteredFamilies();
111     }
112 }
```

TTFontCache:

```
79
80     public List<String> getRegisteredFamilies() {
81         return new ArrayList<>(myMap_Family_RegularFont.keySet());
82     }
83 }
```

Location:

```
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > itext >  ITextEngine
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > itext >  FontSubstitutionPanel
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > itext >  FontSubstitutionModel
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > fonts >  TTFontCache
```

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

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

Design Pattern 3: Visitor Design Pattern

Visitor Class:

```
1  .../
19 package org.ganttproject.impex.htmlpdf.itext;
20
21 import ...
22
23 public class FontSubstitutionModel {
24
25     public static class FontSubstitution {
26         final TTFontCache myFontCache;
27         final String myOriginalFamily;
28         private final Preferences myPrefs;
29
30         public FontSubstitution(String family, Preferences prefs, TTFontCache fontCache) {
31             myFontCache = fontCache;
32             myOriginalFamily = family;
33             myPrefs = prefs;
34         }
35
36         public boolean isResolved() { return getSubstitutionFont() != null; }
37
38         public void setSubstitutionFamily(String family) { myPrefs.put(myOriginalFamily, family); }
39
40         public String getSubstitutionFamily() { return myPrefs.get(myOriginalFamily, myOriginalFamily); }
41
42         public Font getSubstitutionFont() {
43             return myFontCache.getAwfFont(getSubstitutionFamily());
44         }
45
46     }
47
48     private final TTFontCache myFontCache;
49     private final Map<String, FontSubstitution> mySubstitutions = new LinkedHashMap<>();
50     private final ArrayList<FontSubstitution> myIndexedSubstitutions = new ArrayList<>();
51     private final ITextStylesheet myStylesheet;
52     private final Preferences myPrefs;
53
54     public FontSubstitutionModel(TTFontCache fontCache, ITextStylesheet stylesheet, Preferences prefs) {
55         myFontCache = fontCache;
56         myStylesheet = stylesheet;
57         myPrefs = prefs;
58     }
59
60 }
```

One of the Composite Class:

```
1  .../
19 package org.ganttproject.impex.htmlpdf.fonts;
20
21 import ...
22
23 /**
24  * This class collects True Type fonts from .ttf files in the registered
25  * directories and provides mappings of font family names to plain AWT fonts and
26  * iText fonts.
27  *
28  * @author dbarashev
29 */
30
31 public class TTFontCache {
32     private static final org.slf4j.Logger ourLogger = GILogger.create("Export.Pdf.Fonts").delegate();
33     private final Map<String, AwtFontSupplier> myMap_Family_RegularFont = new TreeMap<>();
34     private final Map<FontKey, com.itextpdf.text.Font> myFontCache = new HashMap<>();
35     private final Map<String, Function<String, BaseFont>> myMap_Family_ItextFont = new HashMap<>();
36     private Properties myProperties;
37     private Function<String, BaseFont> myFallbackFont;
38
39
40     public TTFontCache() {
41         try {
42             myFallbackFont = createFontSupplier(FontManager.INSTANCE.getFallbackFontFile(), isEmbedded: true);
43             registerFontFile(FontManager.INSTANCE.getFallbackFontFile());
44         } catch (Exception e) {
45             ourLogger.error("Failed to create fallback font", e);
46         }
47     }
48
49 }
```

Location:

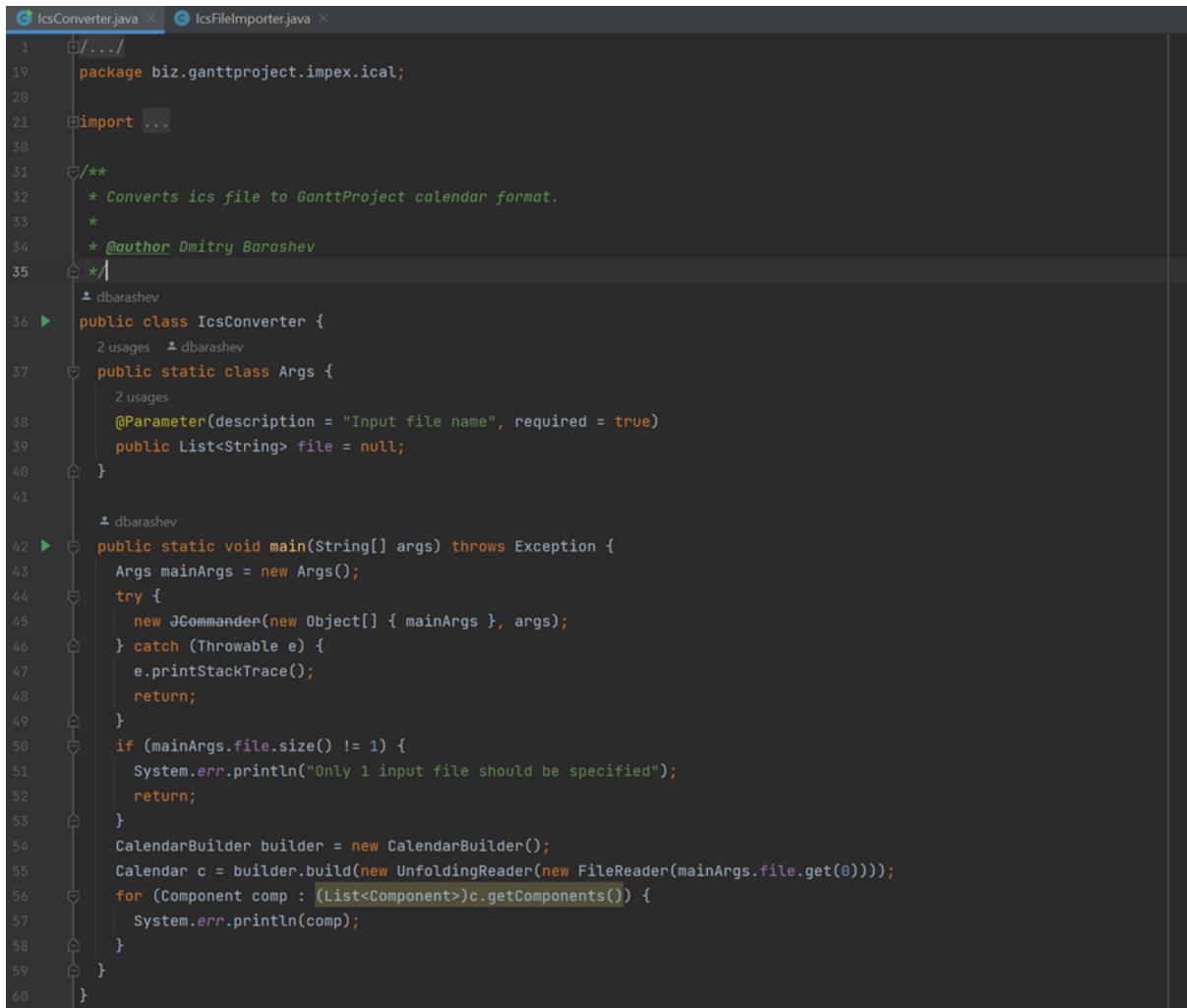
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > itext >  FontSubstitutionModel
ganttproject > org.ganttproject.impex.htmlpdf > src > main > java > org > ganttproject > impex > htmlpdf > fonts >  TTFontCache

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

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

Guilherme Franco:

Design Pattern 1: Adapter Pattern



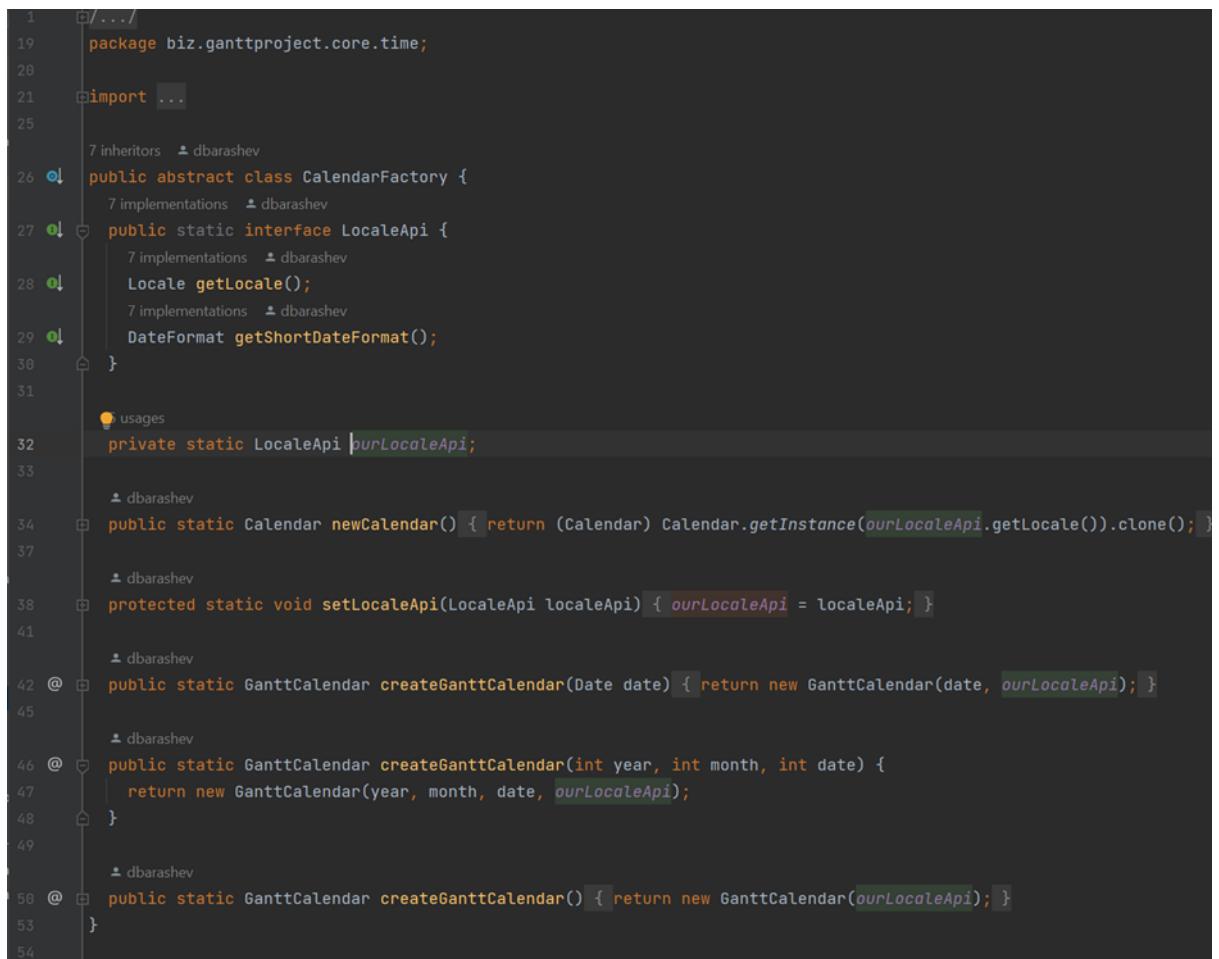
```
1  /...
19 package biz.ganttproject.impex.ical;
28
31 /**
32  * Converts ics file to GanttProject calendar format.
33  *
34  * @author Dmitry Barashev
35 */
36 public class IcsConverter {
37     public static class Args {
38         @Parameter(description = "Input file name", required = true)
39         public List<String> file = null;
40     }
41
42     public static void main(String[] args) throws Exception {
43         Args mainArgs = new Args();
44         try {
45             new JCommander(new Object[] { mainArgs }, args);
46         } catch (Throwable e) {
47             e.printStackTrace();
48             return;
49         }
50         if (mainArgs.file.size() != 1) {
51             System.err.println("Only 1 input file should be specified");
52             return;
53         }
54         CalendarBuilder builder = new CalendarBuilder();
55         Calendar c = builder.build(new UnfoldingReader(new FileReader(mainArgs.file.get(0))));
56         for (Component comp : (List<Component>)c.getComponents()) {
57             System.out.println(comp);
58         }
59     }
60 }
```

Location:

Grantt > biz.ganttproject.impex.ical > src > main > java > biz > ganttproject > impex > ical > IcsConverter

Rationale: This class receives a ics file and converts it into a format our Calendar Class can read so I believe this is an Adaptor Design pattern.

Design Pattern 2: Factory Method Pattern



The screenshot shows a Java code editor with the following code:

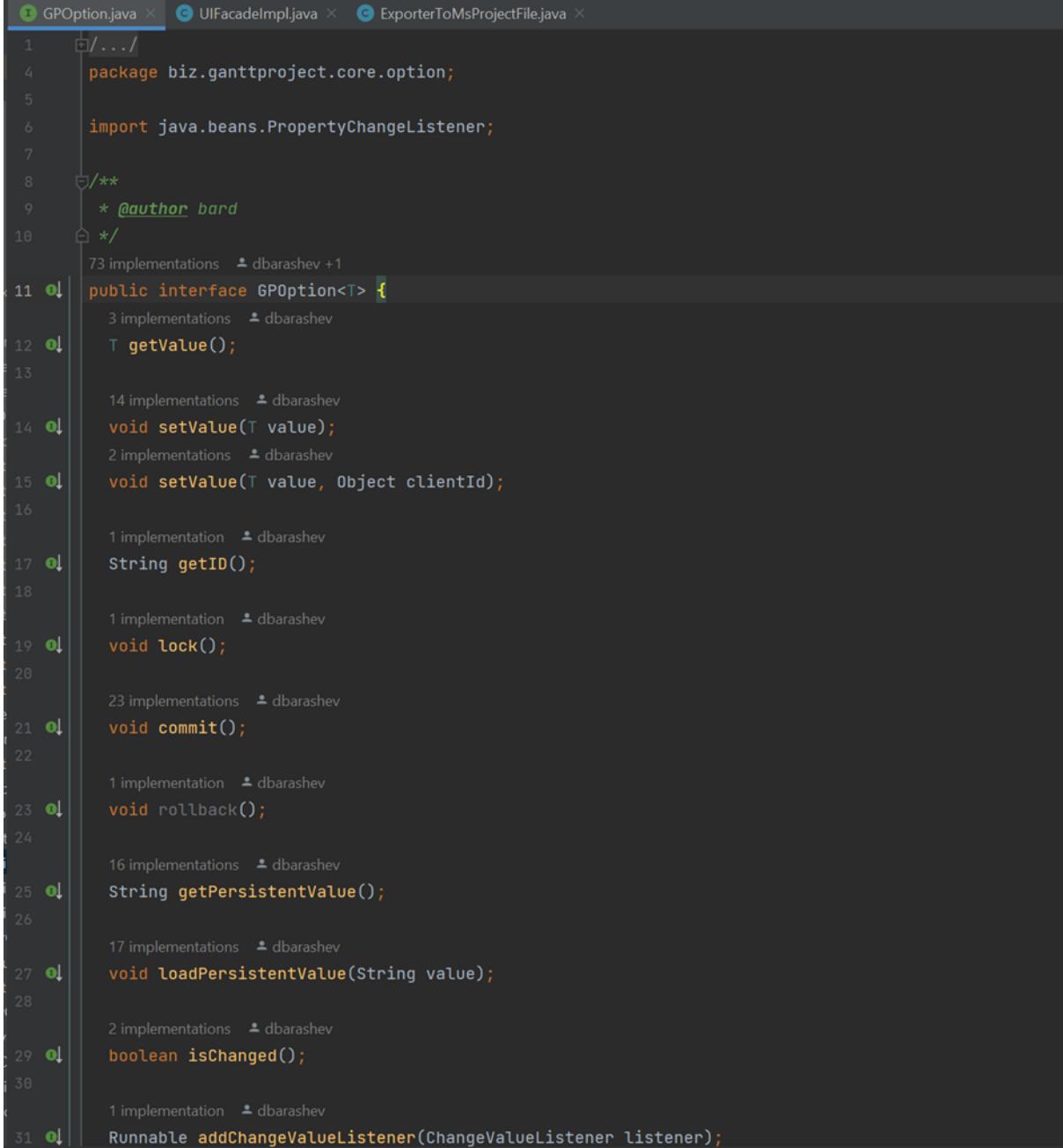
```
1  package biz.ganttproject.core.time;
2
3  import ...
4
5
6  public abstract class CalendarFactory {
7
7  implementations  ↳ dbarashev
8    public static interface LocaleApi {
9
9  implementations  ↳ dbarashev
10   Locale getLocale();
11   DateFormat getShortDateFormat();
12 }
13
14
15  usages
16  private static LocaleApi ourLocaleApi;
17
18  ↳ dbarashev
19  public static Calendar newCalendar() { return (Calendar) Calendar.getInstance(ourLocaleApi.getLocale()).clone(); }
20
21  ↳ dbarashev
22  protected static void setLocaleApi(LocaleApi localeApi) { ourLocaleApi = localeApi; }
23
24  ↳ dbarashev
25  @ public static GanttCalendar createGanttCalendar(Date date) { return new GanttCalendar(date, ourLocaleApi); }
26
27  ↳ dbarashev
28  @ public static GanttCalendar createGanttCalendar(int year, int month, int date) {
29    return new GanttCalendar(year, month, date, ourLocaleApi);
30  }
31
32  ↳ dbarashev
33  @ public static GanttCalendar createGanttCalendar() { return new GanttCalendar(ourLocaleApi); }
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54 }
```

Location:

Grantt > biz.ganttproject.core > src > main > java > biz > ganttproject > core > time > CalendarFactory

Rationale: This class's purpose is to hide the creation of instances of Calendars behind an abstract class "CalendarFactory". Therefore, I believe this is a Factory Method design Pattern.

Design Pattern 3: Decorator Pattern



The screenshot shows a Java code editor with the file `GPOption.java` open. The code defines an interface `GPOption<T>` with various methods and annotations. The code is color-coded, and the editor interface shows tabs for other files like `UIFacadeImpl.java` and `ExporterToMsProjectFile.java`.

```
1  .../
4  package biz.ganttproject.core.option;
5
6  import java.beans.PropertyChangeListener;
7
8  /**
9   * @author bard
10  */
11 public interface GPOption<T> {
12     T getValue();
13
14     void setValue(T value);
15     void setValue(T value, Object clientId);
16
17     String getID();
18
19     void lock();
20
21     void commit();
22
23     void rollback();
24
25     String getPersistentValue();
26
27     void loadPersistentValue(String value);
28
29     boolean isChanged();
30
31     Runnable addChangeValueListener(ChangeValueListener listener);
}
```

This is the Interface

```
1  package biz.ganttproject.core.option;
2
3  import ...
4
5
6  public abstract class GPAbstractOption<T> implements GPOption<T> {
7
8      public abstract static class I18N {
9
10         private static I18N ourInstance;
11
12         protected static void setI18N(I18N i18n) { ourInstance = i18n; }
13
14         protected abstract String i18n(String key);
15     }
16
17     private final String myID;
18
19     //private final List<ChangeValueListener> myListeners = new ArrayList<ChangeValueListener>();
20
21     private final Listeners myListeners = new Listeners();
22
23     private final PropertyChangeSupport myPropertyChangeSupport = new PropertyChangeSupport( sourceBean, this );
24
25     private boolean isWritable = true;
26
27     private T myValue;
28
29     private T myInitialValue;
30
31     private boolean isScreened;
32 }
```

This is the Decorator Class

Location:

Grantt > biz.ganttproject.core > src > main > java > biz > ganttproject > core > option >

Grantt > biz.ganttproject.core > src > main > java > biz > ganttproject > core > option > GPAbstractOption

Rationale: There are 73 implementations of the Interface GPOption, each one using a different subclass of the GPAbstractOption (which implements the GPOption Interface) and most of these subclasses override certain methods, adding their own touch to it, So I believe that GPAbstractOption is the Decorator class, and we are in the presence of a Decorator Design Pattern.

Tiago Francisco:

Design Pattern 1: Open/ Closed

Exporter base

```
1     .../
19    package net.sourceforge.ganttproject.export;
20
21    import ...
22
23
24
25    12 usages  6 inheritors  ± dbarashev +1
26    public abstract class ExporterBase implements Exporter {
27        2 usages
28
29        private IGanttProject myProject;
30        6 usages
31
32        private Chart myGanttChart;
33        2 usages
34
35        private Chart myResourceChart;
36        2 usages
37
38        private UIFacade myUIFacade;
39        6 usages
40
41        private Preferences myRootPreferences;
42        6 usages
43
44        private DefaultDateOption myExportRangeStart;
45        6 usages
46
47        private DefaultDateOption myExportRangeEnd;
48
49
50        protected static final GanttLanguage language = GanttLanguage.getInstance();
51
52
53
54        1 usage
55        static protected Object EXPORT_JOB_FAMILY = "Export job family";
56
57
58        1 override  ± Dmitry Barashev +1
59        @Override
60        public void setContext(IGanttProject project, UIFacade uiFacade, Preferences prefs) {
61            myGanttChart = uiFacade.getGanttChart();
62            myResourceChart = uiFacade.getResourceChart();
63            myProject = project;
64            myUIFacade = uiFacade;
65            myRootPreferences = prefs;
66            Preferences prefNode = prefs.node( s: "/instance/net.sourceforge.ganttproject/export");
67            myExportRangeStart = new DefaultDateOption( id: "export.range.start", myGanttChart.getStartDate());
68            myExportRangeStart.loadPersistentValue(prefNode.get(
69                s: "export-range-start", DateParser.getIsoDate(myGanttChart.getStartDate())));
70            myExportRangeStart.addValueChangeListener(event -> {
71                prefNode.put( s: "export-range-start", myExportRangeStart.getPersistentValue());
72            });
73            myExportRangeEnd = new DefaultDateOption( id: "export.range.end", myGanttChart.getEndDate());
74            myExportRangeEnd.loadPersistentValue(prefNode.get(
75                s: "export-range-end", DateParser.getIsoDate(myGanttChart.getEndDate())));
76            myExportRangeEnd.addValueChangeListener(event -> {
77                prefNode.put( s: "export-range-end", myExportRangeEnd.getPersistentValue());
78            });
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
945
946
947
947
948
949
949
950
951
952
953
954
955
956
956
957
958
958
959
959
960
961
962
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
16
```

ExporterToMsProjectFile

```
1  .../
19   package biz.ganttproject.impex.msproject2;
20
21  import ...
22
23  /**
24   * @author dbarashev (Dmitry Barashev)
25   */
26
27  4 usages ± dbarashev +1
28  public class ExporterToMsProjectFile extends ExporterBase {
29
30
31  10 usages
32  private static final String[] FILE_FORMAT_IDS = new String[] { "impex.msproject.fileformat.mpx",
33  | "impex.msproject.fileformat.mspdi" };
34
35
36  4 usages
37  private static final String[] FILE_EXTENSIONS = new String[] { "mpx", "xml" };
38
39
40  7 usages
41  private String myFileFormat = FILE_FORMAT_IDS[0];
42
43
44  6 usages ± dbarashev
45  private EnumerationOption myFileFormatOption = new DefaultEnumerationOption<Object>(
46  | id: "impex.msproject.fileformat",
47  | FILE_FORMAT_IDS) {
48
49  | ± dbarashev
50  | @Override
51  | public void commit() {
52  | | super.commit();
53  | | ExporterToMsProjectFile.this.myFileFormat = getValue();
54  | }
55  | };
56
57
58  4 usages
59  private LocaleOption myLanguageOption = new LocaleOption();
60
61
62  2 usages
63  private GPOptionGroup myOptions = new GPOptionGroup( id: "exporter.msproject", new GPOption[] { myFileFormatOption });
64
65
66  2 usages
67  private GPOptionGroup myMPXOptions = new GPOptionGroup( id: "exporter.msproject.mpx", new GPOption[] { myLanguageOption });
68
69
70  ± dbarashev
71  public ExporterToMsProjectFile() {
72  | myOptions.setTitled(false);
73  | myMPXOptions.setTitled(false);
74  | myFileFormatOption.lock();
75  | myFileFormatOption.setValue(FILE_FORMAT_IDS[0]);
76  | myFileFormatOption.commit();
77 }
```

Location:

ganttproject > biz.ganttproject.impex.msproject2 > src > main > java > biz > ganttproject > impex > msproject2 > ExporterToMsProjectFile

Rationale: ExporterBase is the abstract class that is closed because it is used in eleven more classes beyond ExporterToMsProjectFile, so if we changed something we would need to change it in twelve different places .The ExporterToMsProjectFile is the class that extends ExporterBase to implement new functionalities avoiding introducing undesirable side effects.

Design Pattern 2: Decorator Design Pattern

```
./...
package net.sourceforge.ganttproject.export;

import ...

12 usages  6 inheritors  ± dbarashev +1*
public abstract class
ExporterBase implements Exporter {
    2 usages
    private IGanttProject myProject;
    6 usages
    private Chart myGanttChart;
    2 usages
    private Chart myResourceChart;
    2 usages
    private UIFacade myUIFacade;
    6 usages
    private Preferences myRootPreferences;
    6 usages
    private DefaultDateOption myExportRangeStart;
    6 usages
    private DefaultDateOption myExportRangeEnd;

    protected static final GanttLanguage language = GanttLanguage.getInstance();

    1 usage
    static protected Object EXPORT_JOB_FAMILY = "Export job family";

    1 override  ± Dmitry Barashev +1
@Override
public void setContext(IGanttProject project, UIFacade uiFacade, Preferences prefs) {
    myGanttChart = uiFacade.getGanttChart();
    myResourceChart = uiFacade.getResourceChart();
    myProject = project;
    myUIFacade = uiFacade;
    myRootPreferences = prefs;
    Preferences prefNode = prefs.node( s: "/instance/net.sourceforge.ganttproject/export");
    myExportRangeStart = new DefaultDateOption( id: "export.range.start", myGanttChart.getStartDate());
    myExportRangeStart.loadPersistentValue(prefNode.get(
        s: "export-range-start", DateParser.getIsoDate(myGanttChart.getStartDate())));
    myExportRangeStart.addChangeListener(event -> {
        prefNode.put( s: "export-range-start", myExportRangeStart.getPersistentValue());
    });
    myExportRangeEnd = new DefaultDateOption( id: "export.range.end", myGanttChart.getEndDate());
    myExportRangeEnd.loadPersistentValue(prefNode.get(
        s: "export-range-end", DateParser.getIsoDate(myGanttChart.getEndDate())));
    myExportRangeEnd.addChangeListener(event -> {
        prefNode.put( s: "export-range-end", myExportRangeEnd.getPersistentValue());
    });
}
```

The decorator class

Location:

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

Rationale: The decorator is an abstract class that implements Exporter the component interface that defines the common type for all the classes. The StylesheetExporterBase is a concrete decorator that extends ExporterBase the decorator.

Design Pattern 3: Façade

```
1 implementation ± dbarashev
38 ①↓ static interface TimelineFacade {
    1 usage 1 implementation ± dbarashev
    39 ①↓     ScrollingSession createScrollingSession(int xpos, int ypos);
    40
    6 usages 1 implementation ± dbarashev
    41 ①↓     Date getDateAt(int x);
    42
    1 usage 1 implementation ± dbarashev
    43 ①↓     TimeDuration createTimeInterval(TimeUnit timeUnit, Date startDate, Date endDate);
    44
    1 implementation ± dbarashev
    45 ①↓     TimeUnitStack getTimeUnitStack();
    46
    1 implementation ± dbarashev
    47 ①↓     GPCalendarCalc getCalendar();
    48
    1 implementation ± dbarashev
    49 ①↓     Date getEndDateAt(int i);
    50 }
    51 }
```

Location:

ganttpoint > ganttpoint > src > main > java > net > sourceforge > ganttpoint > chart > mouse > ① MouseInteraction > ① TimelineFacade

Rationale: Façade provides a unified interface to a complex subsystem and that complexity in this case is hidden in TimelineFacade.

Pedro Gouveia:

Open/Closed

```
public abstract class ChartModelBase implements /* TimeUnitStack.Listener, */ChartModel, TimelineLabelRendererImpl.ChartModelApi {
    10 usages 2 Implementations ± dbarashev
    public static interface ScrollingSession {
        2 implementations ± dbarashev
        void scrollTo(int xpos, int ypos);
        2 implementations ± dbarashev
        void finish();
    }
}
```

Location:

ganttpoint > ganttpoint > src > main > java > net > sourceforge > ganttpoint > chart > ② ChartModelBase

Rationale: There is an abstract class called *ChartModelBase* to which there are 2 inheritors implementations called *ChartModelImpl* and *ChartModelResource* that are open to change

and the abstract class is closed because methods in the abstract superclass are preserved, and the system can be extended by providing different implementations for each method in the subclasses.

Observer

```
public ActivityOnNodePertChart() {
    setBackground(Color.WHITE.brighter());

    ± dbarashev +1
    this.addMouseListener(new MouseListener() {
        ± dbarashev +1
        @Override
        public void mouseDragged(MouseEvent e) {
            if (myPressedGraphicalNode != null) {
                myPressedGraphicalNode.x = e.getX() - myXClickedOffset;
                myPressedGraphicalNode.y = e.getY() - myYClickedOffset;
                if (e.getX() > getPreferredSize().getWidth()) {
                    ActivityOnNodePertChart.this.setPreferredSize(new Dimension( width: myPressedGraphicalNode.x + getNodeWidth() + getxGap(),
                        (int) getPreferredSize().getHeight()));
                    revalidate();
                }
                if (e.getY() > getPreferredSize().getHeight()) {
                    ActivityOnNodePertChart.this.setPreferredSize(new Dimension((int) getPreferredSize().getWidth(),
                        height: myPressedGraphicalNode.y + getNodeHeight() + getyGap()));
                    revalidate();
                }
                repaint();
            }
        }
    });

    ± dbarashev
    @Override
    public void mouseMoved(MouseEvent e) {
        // nothing to do...
    }
};

± dbarashev +1
this.addMouseMotionListener(new MouseMotionListener() {
    ± dbarashev
    @Override
    public void mouseDragged(MouseEvent e) {
        // nothing to do...
    }
});
```

Location:

ganttproject > org.ganttproject.chart.pert > src > main > java > org > ganttproject > chart > pert >  ActivityOnNodePertChart

Rationale:

This class methods are called whenever there are some actions on a node of the Pert Chart, changing some data in the main app. These actions can be a mouse drag, a mouse move, a mouse click, etc.

Façade

```
± Dmitry Barashev
class FacadeImpl(private val taskManager: TaskManagerImpl, private val root: Task) : TaskContainmentHierarchyFacade {
    ± Dmitry Barashev
    override fun getNestedTasks(container: Task): Array<Task> {
        return container.nestedTasks
    }

    ± Dmitry Barashev
    override fun getDeepNestedTasks(container: Task): Array<Task> {
        val result = ArrayList<Task>()
        addDeepNestedTasks(container, result)
        return result.toTypedArray()
    }

    ± Dmitry Barashev
    private fun addDeepNestedTasks(container: Task, result: ArrayList<Task>) {
        val nested = container.nestedTasks
        result.addAll(Arrays.asList(*nested))
        for (i in nested.indices) {
            addDeepNestedTasks(nested[i], result)
        }
    }

    ± Dmitry Barashev
    override fun hasNestedTasks(container: Task): Boolean {
        return container.nestedTasks.size > 0
    }

    ± Dmitry Barashev
    override fun getRootTask(): Task {
        return root
    }

    ± Dmitry Barashev
```

Location:

ganttpoint > ganttpoint > src > main > java > net > sourceforge > ganttpoint > task >  TaskTreeFacade.kt >  FacadeImpl

Rationale: This façade serves as a way for the user to interact with the task hierarchy tree hiding more complex functionalities like interacting with left and right nodes, iterating the tree and so on from the other classes. It saves the user the task and trouble to make a class to interact with the Task Hierarchy tree.

Luís Abreu:

Design Pattern 1: Memento Pattern

This is the interface:

```
2 implementations ▲ dbarashev
27  ↴ public interface GPUndoManager {
28    ↴     1 implementation ▲ dbarashev
29         void undoableEdit(String localizedName, Runnable runnableEdit);
30    ↴     4 usages 1 implementation ▲ dbarashev
31         boolean canUndo();
32    ↴     4 usages 1 implementation ▲ dbarashev
33         boolean canRedo();
34    ↴     1 implementation ▲ dbarashev
35         void undo() throws CannotUndoException;
36    ↴     1 implementation ▲ dbarashev
37         void redo() throws CannotRedoException;
38    ↴     1 usage 1 implementation ▲ dbarashev
39         String getUndoPresentationName();
40    ↴     1 usage 1 implementation ▲ dbarashev
41         String getRedoPresentationName();
42    ↴     4 usages 1 implementation ▲ dbarashev
43         void addUndoableEditListener(GPUndoListener listener);
44    ↴     1 implementation ▲ dbarashev
45         void removeUndoableEditListener(GPUndoListener listener);
46    ↴     2 usages 1 implementation ▲ dbarashev
47         void die();
48 }
```

This is the class that implements the interface:

```
43 4 usages 1 inheritor ▲ dbarashev +2
44 public class UndoManagerImpl implements GPUndoManager {
45   2 usages
46   private final ProjectDatabase myProjectDatabase;
47   6 usages
48   private final UndoableEditSupport myUndoEventDispatcher;
49
50   10 usages
51   private final UndoManager mySwingUndoManager;
52
53   2 usages
54   private final DocumentManager myDocumentManager;
55
56   2 usages
57   private final ParserFactory myParserFactory;
58
59   2 usages
60   private final IGanttProject myProject;
61
62   5 usages
63   private UndoableEditImpl swingEditImpl;
64
65   1 usage ▲ dbarashev +1
66   public UndoManagerImpl(@NotNull IGanttProject project,
67                         @NotNull ParserFactory parserFactory,
68                         @NotNull DocumentManager documentManager,
69                         @NotNull ProjectDatabase projectDatabase) {
70     myProject = project;
71     myParserFactory = parserFactory;
72     myDocumentManager = documentManager;
73     mySwingUndoManager = new UndoManager();
74   }
```

Location:

```
ganttpoject > ganttpoject > src > main > java > net > sourceforge > ganttpoject > undo > UndoManagerImpl
```

Rationale: This class allows to return an object to one of its previous states. In this case, our class extends an interface that has methods (*canUndo()* and *undo()*) that allow the object to check if it has saved a previous state and if that is the case, it can return to them.

Design Pattern 2: Factory Method Pattern

```
34 */  
35     3 usages  ▲ dbarashev +1  
36     public class IndentTargetFunctionFactory implements Function<Collection<Task>, Function<Task, Task>> {  
37         2 usages  
38         private final TaskManager myTaskManager;  
39         3 usages  ▲ dbarashev +1  
40         □ public IndentTargetFunctionFactory(TaskManager taskManager) { myTaskManager = taskManager; }  
41  
42         ▲ dbarashev  
43         @Override  
44         ① □ public Function<Task, Task> apply(final Collection<Task> indentRoots) {  
45             ▲ dbarashev  
46             return new Function<Task, Task>() {  
47                 1 usage  
48                 private final TaskContainmentHierarchyFacade myTaskHierarchy = myTaskManager.getTaskHierarchy();  
49  
50                 ▲ dbarashev  
51                 @Override  
52                 public Task apply(Task whatMove) {  
53                     Task moveTarget = whatMove;  
54                     for (; moveTarget != null && indentRoots.contains(moveTarget);  
55                         moveTarget = myTaskHierarchy.getPreviousSibling(moveTarget));  
56                     return moveTarget;  
57                 }  
58             };  
59         }  
60     }  
61 }
```

Location:

ganttproject > ganttproject > src > main > java > net > sourceforge > ganttproject > action > task > IndentTargetFunctionFactory

Rationale: This class's purpose is to hide the creation of instances of a given type

Design Pattern 3: Facade Pattern

This is the interface:

```
31 ① 7 usages 4 implementations ▲ dbarashev +1
31 ① public interface TreeUiFacade<T> {
32 ①     4 usages 1 implementation ▲ dbarashev
32 ①     Component getTreeComponent();
33
34 ①     1 implementation ▲ dbarashev
34 ①     ColumnList getVisibleFields();
35
36 ①     1 implementation ▲ dbarashev
36 ①     boolean isVisible(T modelElement);
37
38 ①     1 implementation ▲ dbarashev
38 ①     boolean isExpanded(T modelElement);
39
40 ①     2 usages 1 implementation ▲ dbarashev
40 ①     void setExpanded(T modelElement, boolean value);
41
42 ①     1 implementation ▲ dbarashev
42 ①     void applyPreservingExpansionState(T modelElement, Predicate<T> callable);
43 ① /**
44 ① * Modifies the selected node(s) of the tree
45 ① *
46 ① * @param clear
47 ① *         when true, it first clears the previous selection. When false the
48 ① *         current selection gets extended
49 ① * @param modelElement
50 ① *         to be selected
```

This is the class that implements the interface:

```
67 ① 3 usages 1 inheritor ▲ dbarashev +3
67 ① public abstract class TreeTableContainer<ModelObject, TreeTableClass extends GPTreeTableBase, TreeTableModelClass extends DefaultTreeTableModel>
68 ①     extends JPanel implements TreeUiFacade<ModelObject> {
69
70 ①     16 usages
70 ①     private final TreeTableClass myTreeTable;
71 ①     2 usages
71 ①     private final TreeTableModelClass myTreeTableModel;
72 ①     3 usages
72 ①     private GPAAction myNewAction;
73 ①     3 usages
73 ①     private GPAAction myPropertiesAction;
74 ①     2 usages
74 ①     private GPAAction myDeleteAction;
75
75 ①     1 usage ▲ dbarashev
75 ①     private class ExpandCollapseAction extends GPAAction {
76 ①         1 usage ▲ dbarashev
76 ①         ExpandCollapseAction() { super( name: "tree.expand"); }
77
77 ①         ▲ dbarashev
78 ①         @Override
78 ①         public void actionPerformed(ActionEvent e) {
79 ①             TreePath currentSelection = getTree().getTreeSelectionModel().getSelectionPath();
80 ①             if (currentSelection != null) {
81 ①                 if (getTree().isCollapsed(currentSelection)) {
82 ①                     getTree().expandPath(currentSelection);
83 ①                 } else {
84 ①                     getTree().collapsePath(currentSelection);
85 ①                 }
86 ①             }
87 ①         }
88 ①     }
89 ① }
90
91 ① 1 usage ▲ dbarashev +1
91 ①     private static ExpandCollapseAction expandCollapseAction /
```

Location:

ganttproject › ganttproject › src › main › java › net › sourceforge › ganttproject › gui ›  TreeUiFacade

ganttproject › ganttproject › src › main › java › net › sourceforge › ganttproject ›  TreeTableContainer

Rationale: A façade is a wrapper class that encapsulates a subsystem in order to hide the subsystem complexity. The intent is to hide complexity behind an interface. In this case, we hide the complexity in the TreeUiFacade.