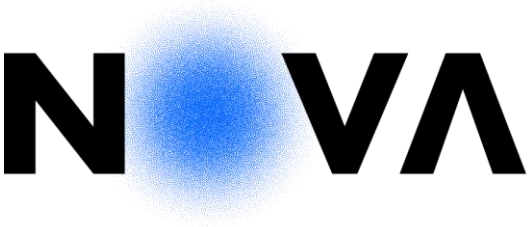


NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

# Software Engineering – 2022/2023

GanttProject Phase 1 Report


**Made by:**

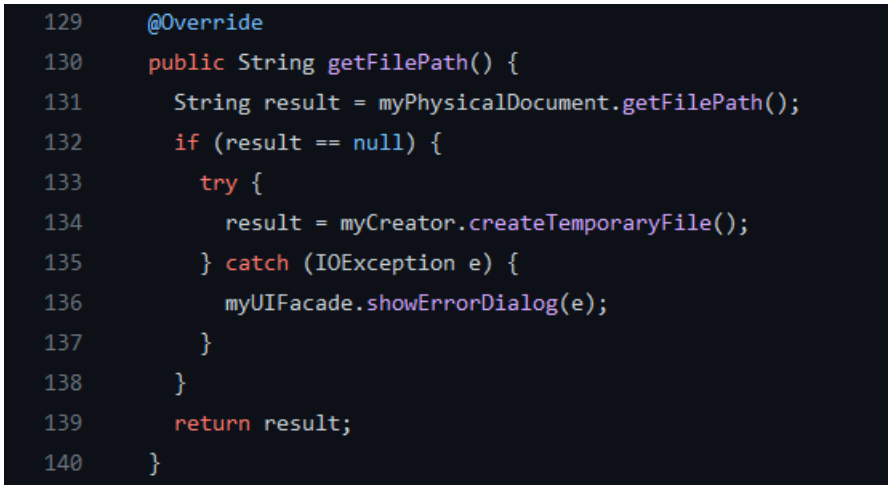
Pedro Lopes (57514)
Guilherme Fernandes (60045)
Rafael Martins (60602)
Pedro Fernandes (60694)
Rafael Pereira (60700)

# Index

# Design Patterns

## Proxy

### Illustrating code snippet



```
129     @Override
130     public String getFilePath() {
131       String result = myPhysicalDocument.getFilePath();
132       if (result == null) {
133         try {
134           result = myCreator.createTemporaryFile();
135         } catch (IOException e) {
136           myUIFacade.showErrorDialog(e);
137         }
138       }
139       return result;
140     }
```

### Class location

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

### Explanation

The ProxyDocument.java class performs the same tasks as Document object, but may delegate requests to the Document object to achieve them

### Author

Guilherme Fernandes (60045)

### Reviewer

Pedro Fernandes (60694)

# Factory

## Illustrating code snippet

```
42    public static GanttCalendar createGanttCalendar(Date date) {
43      return new GanttCalendar(date, ourLocaleApi);
44    }
45
46    public static GanttCalendar createGanttCalendar(int year, int month, int date) {
47      return new GanttCalendar(year, month, date, ourLocaleApi);
48    }
49
50    public static GanttCalendar createGanttCalendar() {
51      return new GanttCalendar(ourLocaleApi);
52    }
```

## Class location

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

## Explanation

The CalendarFactory.java class hide the creation of GanttCalendar objects

## Author

Guilherme Fernandes (60045)

## Reviewer

Rafael Pereira (60700)

# Façade

## Illustrating code snippet

```
391     @Override
392     public void showNotificationDialog(NotificationChannel channel, String message) {
393       String i18nPrefix = channel.name().toLowerCase() + ".channel.";
394       getNotificationManager().addNotifications(
395           channel,
396           Collections.singletonList(new NotificationItem(i18n(i18nPrefix + "itemTitle"),
397               GanttLanguage.getInstance().formatText(i18nPrefix + "itemBody", message), new HyperlinkListener() {
398             @Override
399             public void hyperlinkUpdate(HyperlinkEvent e) {
400               if (e.getEventType() != EventType.ACTIVATED) {
401                 return;
402               }
403               if ("localhost".equals(e.getURL().getHost()) && "/log".equals(e.getURL().getPath())) {
404                 onViewLog();
405               } else {
406                 NotificationManager.DEFAULT_HYPERLINK_LISTENER.hyperlinkUpdate(e);
407               }
408             }
409           })));
410     }
```

## Class location

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

## Explanation

The UIFacadeImpl.java is a class that implements the UIFacade.java interface, and which encapsulates a subsystem to hide the complexity of the subsystem (for example, the showNotificationDialog method hides from the user how the notification will be created and displayed) and acts as a point of entry into a subsystem without adding more functionality itself (e.g. the showNotificationDialog method uses the NotificationManager subsystem and no extra functionality is added to the class)

## Author

Guilherme Fernandes (60045)

## Reviewer

Pedro Lopes (57514)

# Abstract Factory

## Illustrating code snippet

```
public abstract class StylesheetFactoryImpl {
  public List<Stylesheet> createStylesheets(Class<? extends Stylesheet> stylesheetInterface) {
    IExtensionRegistry extensionRegistry = Platform.getExtensionRegistry();
    IConfigurationElement[] configElements = extensionRegistry.getConfigurationElementsFor(stylesheetInterface.getName());
    List<Stylesheet> result = new ArrayList<>();
    for (int i = 0; i < configElements.length; i++) {
      try {
        // Object nextExtension =
        // configElements[i].createExecutableExtension("class");
        // assert nextExtension!=null && nextExtension instanceof HTMLStylesheet
        // :
        // "Extension="+nextExtension+" is expected to be instance of HTMLStylesheet";
        String localizedName = configElements[i].getAttribute( s: "name");
        String pluginRelativeUrl = configElements[i].getAttribute( s: "url");
        String namespace = configElements[i].getDeclaringExtension().getNamespaceIdentifier();
        URL stylesheetUrl = Platform.getBundle(namespace).getResource(pluginRelativeUrl);
        assert stylesheetUrl != null : "Failed to resolve url=" + pluginRelativeUrl;
        URL resolvedUrl = Platform.resolve(stylesheetUrl);
        assert resolvedUrl != null : "Failed to resolve URL=" + stylesheetUrl;
        result.add(newStylesheet(resolvedUrl, localizedName));
      } catch (Exception e) {
        if (!GPLogger.log(e)) {
          e.printStackTrace(System.err);
        }
      }
    }
    return result;
  }

  protected abstract Stylesheet newStylesheet(URL url, String localizedName);
}
```

## Class location

/org.ganttproject.impex.htmlpdf/src/main/java/org/ganttproject/impex/htmlpdf/StylesheetFactoryImpl.java

## Explanation

This class is responsible for constructing lists of Stylesheets potentially of different types that share similarities and can be constructed provided an implementation of Stylesheet.

## Author

Pedro Fernandes (60694)

## Reviewer

Guilherme Fernandes (60045)

# Builder

## Illustrating code snippet

```java
public class DependencySceneBuilder<T extends IdentifiableRow, D extends BarChartConnector<T, D>> {
  private final Canvas myTaskCanvas;
  private final Canvas myOutputCanvas;
  private final ChartApi myChartApi;
  private final TaskApi<T, D> myTaskApi;
  private int myBarHeight;
  private Canvas.Arrow myFinishArrow;

  public interface TaskApi<T extends IdentifiableRow, D> {
    boolean isMilestone(T task);
    Dimension getUnitVector(BarChartActivity<T> activity, D dependency);
    String getStyle(D dependency);
    Iterable<D> getConnectors(T task);
    List<T> getTasks();
  }

  public interface ChartApi {
    int getBarHeight();
  }

  public DependencySceneBuilder(Canvas taskCanvas, Canvas outputCanvas, TaskApi<T, D> taskApi, ChartApi chartApi) {
    myTaskApi = taskApi;
    myChartApi = chartApi;
    //myVisibleTasks = visibleTasks;
    myTaskCanvas = taskCanvas;
    myOutputCanvas = outputCanvas;
    myFinishArrow = Canvas.Arrow.FINISH;
    myBarHeight = -1;
  }

  public void build() {
    List<Connector> dependencyDrawData = prepareDependencyDrawData();
    drawDependencies(dependencyDrawData);
  }
```

## Class location

/biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/scene/gantt/DependencySceneBuilder.java

## Explanation

This class is responsible for building something from a complex set of actions, separating its construction from its representation, in this case, dependency lines between tasks.

## Author

Pedro Fernandes (60694)

## Reviewer

Rafael Pereira (60700)

# Singleton

## Illustrating code snippet

```
public class GanttLookAndFeels {

  protected Map<String, GanttLookAndFeelInfo> infoByClass;

  protected Map<String, GanttLookAndFeelInfo> infoByName;

  protected static GanttLookAndFeels singleton;

  static {...}

  protected GanttLookAndFeels() {...}

  protected void addLookAndFeel(GanttLookAndFeelInfo info) {...}

  public GanttLookAndFeelInfo getInfoByClass(String className) {...}

  public GanttLookAndFeelInfo getInfoByName(String name) { return infoByName.get(name); }

  public GanttLookAndFeelInfo getDefaultInfo() {...}

  public GanttLookAndFeelInfo[] getInstalledLookAndFeels() {...}

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

## Class location

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

## Explanation

Limits the instantiation of its class to a single object, always providing the same underlying object on every call to *getGanttLookAndFeels()*.

## Author

Pedro Fernandes (60694)

## Reviewer

Rafael Martins (60602)

# Builder

## Illustrating code snippet

```
51 @  protected OffsetBuilderImpl(OffsetBuilder.Factory factory) {
52      myCalendar = factory.myCalendar;
53      myStartDate = factory.myStartDate;
54      myViewportStartDate = factory.myViewportStartDate;
55      myTopUnit = factory.myTopUnit;
56      myBottomUnit = factory.myBottomUnit;
57      myDefaultUnitWidth = factory.myAtomicUnitWidth;
58      myChartWidth = factory.myEndOffset;
59      myWeekendDecreaseFactor = factory.myWeekendDecreaseFactor;
60      myEndDate = factory.myEndDate;
61      baseUnit = factory.myBaseUnit;
62      myRightMarginBottomUnitCount = factory.myRightMarginTimeUnits;
63      myOffsetStepFn = factory.myOffsetStepFn;
64    }
65
        1 usage  dbarashev
66    private TimeUnit getBottomUnit() { return myBottomUnit; }
69
        1 usage  dbarashev
70    private TimeUnit getTopUnit() { return myTopUnit; }
73
```

## Class location

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

## Explanation

In the OffsetBuilderImpl.java class the initialization of various constants is handled, step by step, y a creator that is not the class, this makes it easier to modify their creation and makes the code more understandable.

## Author

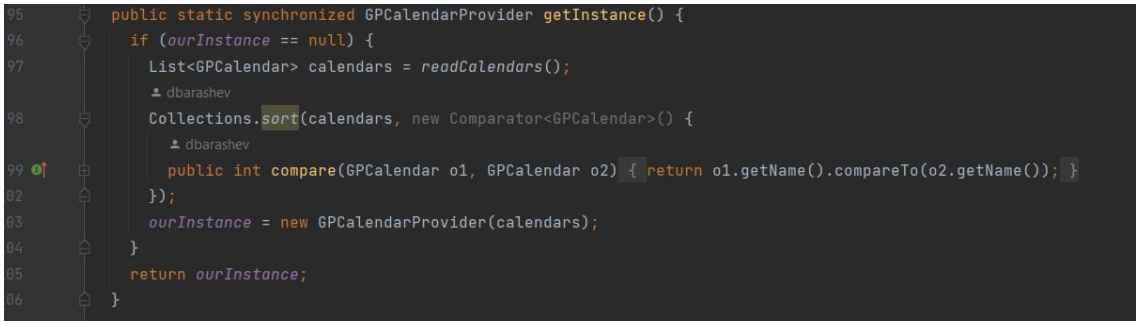
Pedro Lopes (57514)

## Reviewer

Guilherme Fernandes (60045)

# Singleton

## Illustrating code snippet




## Class location

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

## Explanation

The class provides access through a single instance (line 71), the constructor to the class is private (line 109) and the method that instantiates the class is public (line 95).

## Author

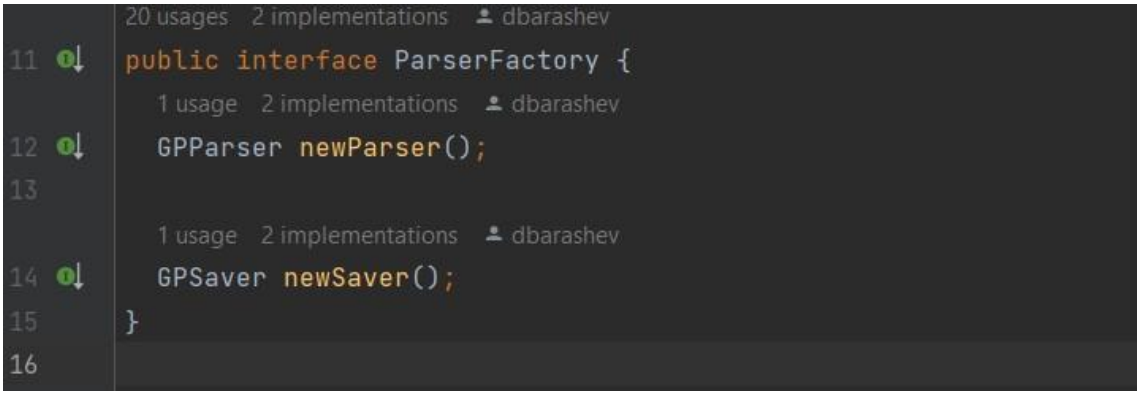
Pedro Lopes (57514)

## Reviewer

Rafael Pereira (60700)

# Factory

## Illustrating code snippet




## Class location

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

## Explanation

The interface ParserFactory.java declares 2 methods that return an object, the interface is implemented by the BufferProject.java class and the ParserFactoryImpl class in GanttProject.java. This allows the subclasses to alter the creation of the new objects to better fit their necessities.

## Author

Pedro Lopes (57514)

## Reviewer

Rafael Martins (60602)

# Factory

## Illustrating code snippet

```
public BeanFactory() {
}

public Class[] keyClasses() { return this.keyClasses; }

public String[] keyIds() { return null; }

public Registration insert(RootContext context, Context requestContext) {
    Object o = this.instantiateBean();
    return this.isRequestScope() ? requestContext.put(o) : context.put(o);
}
```

```
private Object instantiateBean() {
    try {
        return this.beanClass.newInstance();
    } catch (InstantiationException var2) {
        throw new RuntimeException(var2);
    } catch (IllegalAccessException var3) {
        throw new RuntimeException(var3);
    }
}
```

## Class location

/biz.ganttproject.app.libs/lib/milton-api-2.7.4.4.jar/io/milton/context/BeanFactory.java

## Explanation

This Class is used to create and setup new Bean instances, therefore it is being used as a Factory Class.

## Author

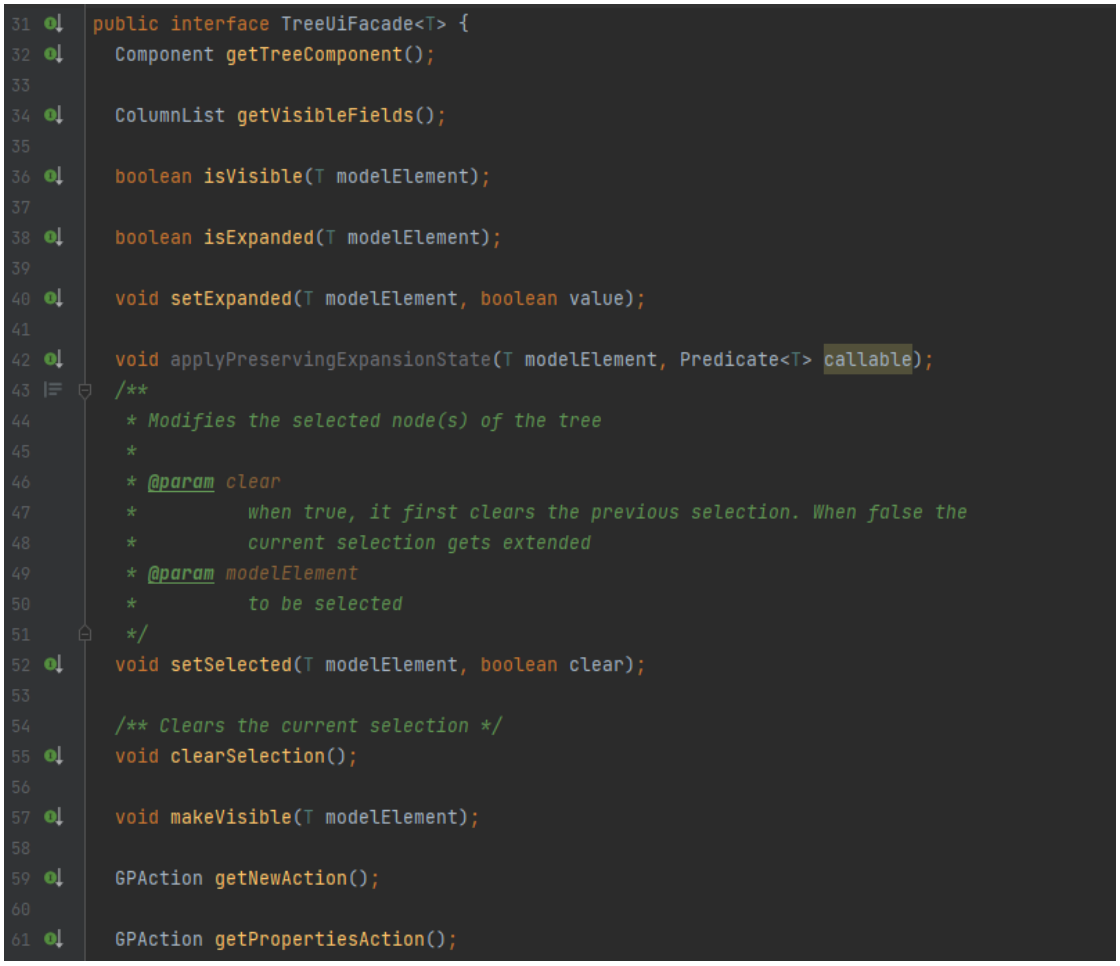
Rafael Martins (60602)

## Reviewer

Guilherme Fernandes (60045)

# Facade

## Illustrating code snippet



```java
public interface TreeUiFacade<T> {
  Component getTreeComponent();

  ColumnList getVisibleFields();

  boolean isVisible(T modelElement);

  boolean isExpanded(T modelElement);

  void setExpanded(T modelElement, boolean value);

  void applyPreservingExpansionState(T modelElement, Predicate<T> callable);
  /**
   * Modifies the selected node(s) of the tree
   *
   * @param clear
   *          when true, it first clears the previous selection. When false the
   *          current selection gets extended
   * @param modelElement
   *          to be selected
   */
  void setSelected(T modelElement, boolean clear);

  /** Clears the current selection */
  void clearSelection();

  void makeVisible(T modelElement);

  GPAction getNewAction();

  GPAction getPropertiesAction();
```

## Class location

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

## Explanation

This Interface is used to hide the complexity of the Tree subsystem, while it doesn't add new functionalities.

## Author

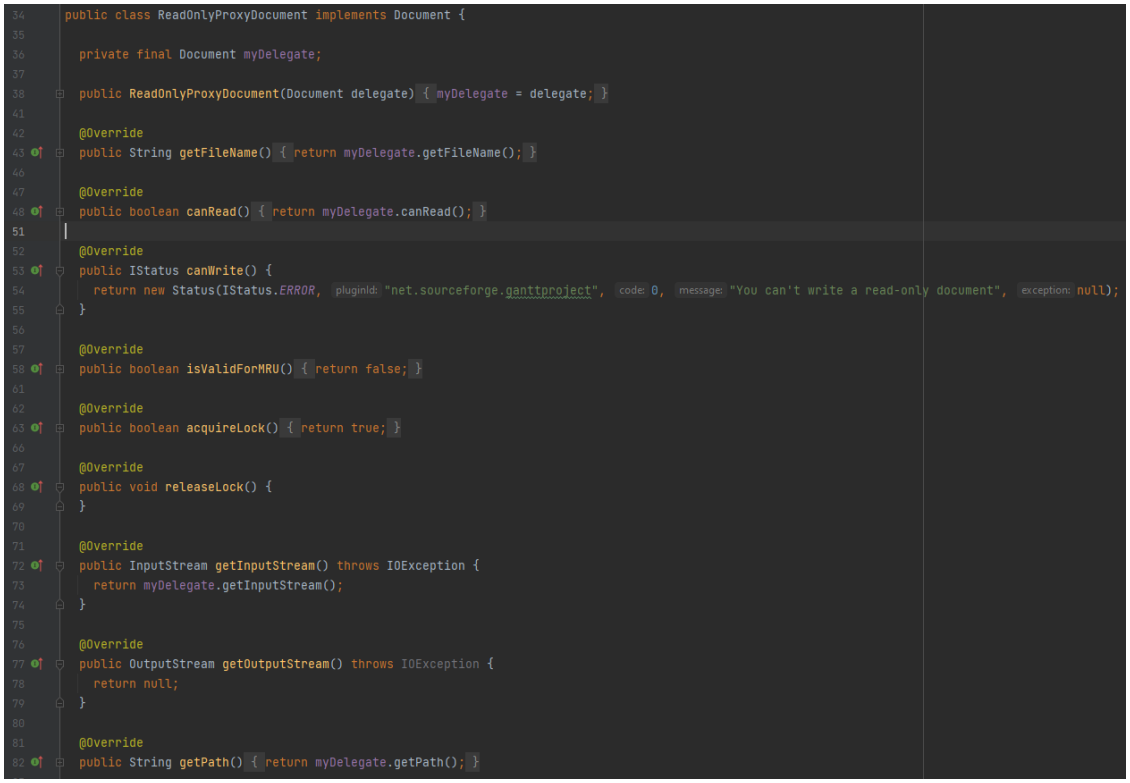
Rafael Martins (60602)

## Reviewer

Pedro Fernandes (60694)

# Proxy

## Illustrating code snippet



```
public class ReadOnlyProxyDocument implements Document {

  private final Document myDelegate;

  public ReadOnlyProxyDocument(Document delegate) { myDelegate = delegate; }

  @Override
  public String getFileName() { return myDelegate.getFileName(); }

  @Override
  public boolean canRead() { return myDelegate.canRead(); }

  @Override
  public IStatus canWrite() {
    return new Status(IStatus.ERROR, pluginId: "net.sourceforge.ganttproject", code: 0, message: "You can't write a read-only document", exception: null);
  }

  @Override
  public boolean isValidForMRU() { return false; }

  @Override
  public boolean acquireLock() { return true; }

  @Override
  public void releaseLock() {
  }

  @Override
  public InputStream getInputStream() throws IOException {
    return myDelegate.getInputStream();
  }

  @Override
  public OutputStream getOutputStream() throws IOException {
    return null;
  }

  @Override
  public String getPath() { return myDelegate.getPath(); }
```

## Class location

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

## Explanation

This Class is used to perform the tasks that the "myDelegate" document would perform, using that same object to delegate some of the requests.

## Author

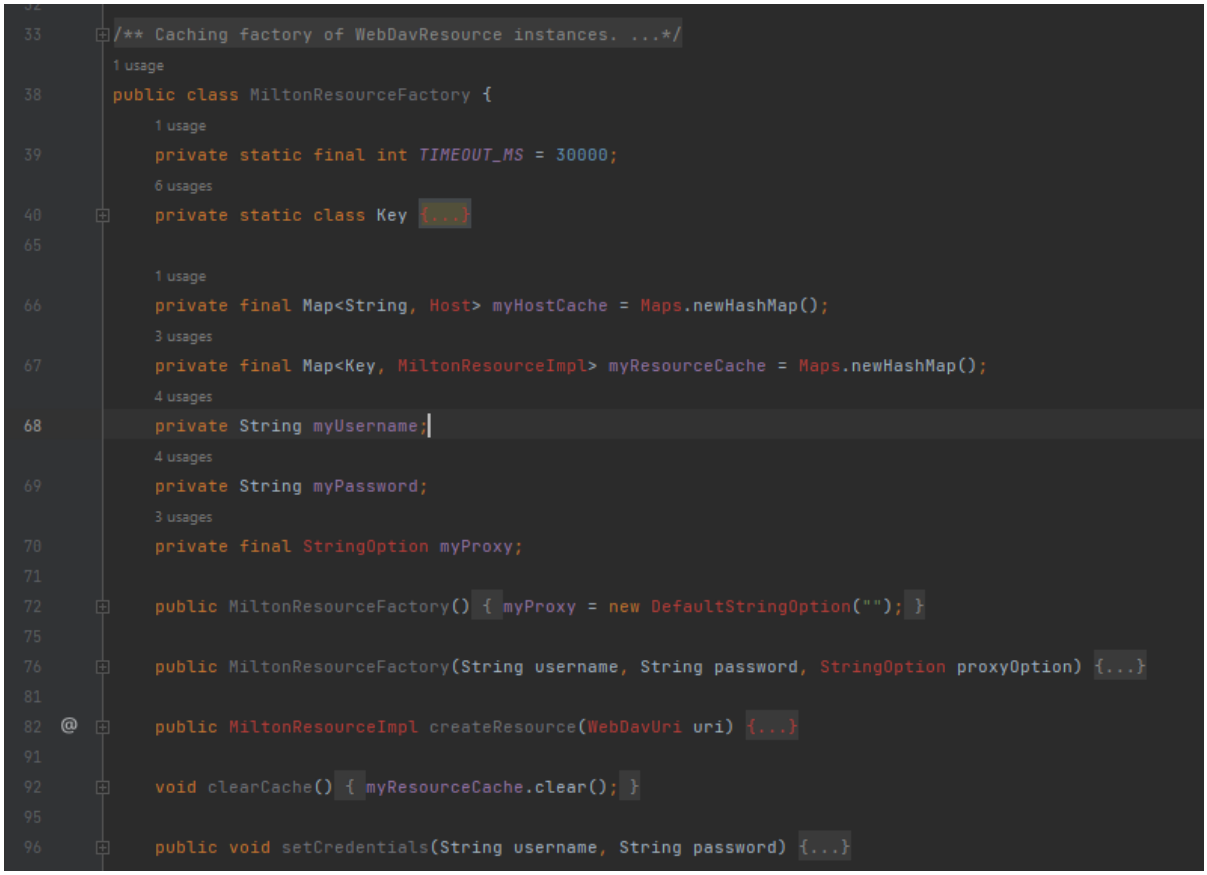
Rafael Martins (60602)

## Reviewer

Pedro Lopes (57514)

# Factory

## Illustrating code snippet



```
/** Caching factory of WebDavResource instances. ...*/
1 usage
public class MiltonResourceFactory {
    1 usage
    private static final int TIMEOUT_MS = 30000;
    6 usages
    private static class Key {...}

    1 usage
    private final Map<String, Host> myHostCache = Maps.newHashMap();
    3 usages
    private final Map<Key, MiltonResourceImpl> myResourceCache = Maps.newHashMap();
    4 usages
    private String myUsername;
    4 usages
    private String myPassword;
    3 usages
    private final StringOption myProxy;

    public MiltonResourceFactory() { myProxy = new DefaultStringOption(""); }

    public MiltonResourceFactory(String username, String password, StringOption proxyOption) {...}

    public MiltonResourceImpl createResource(WebDavUri uri) {...}

    void clearCache() { myResourceCache.clear(); }

    public void setCredentials(String username, String password) {...}
```

## Class location

/ganttproject/src/main/java/net/sourceforge/ganttproject/document/webdav/MiltonResourceFactory.java

## Explanation

Here we have a “Factory” design pattern that has the purpose to hide and simplify the creation of instances of MiltonResourceImpl.

## Author

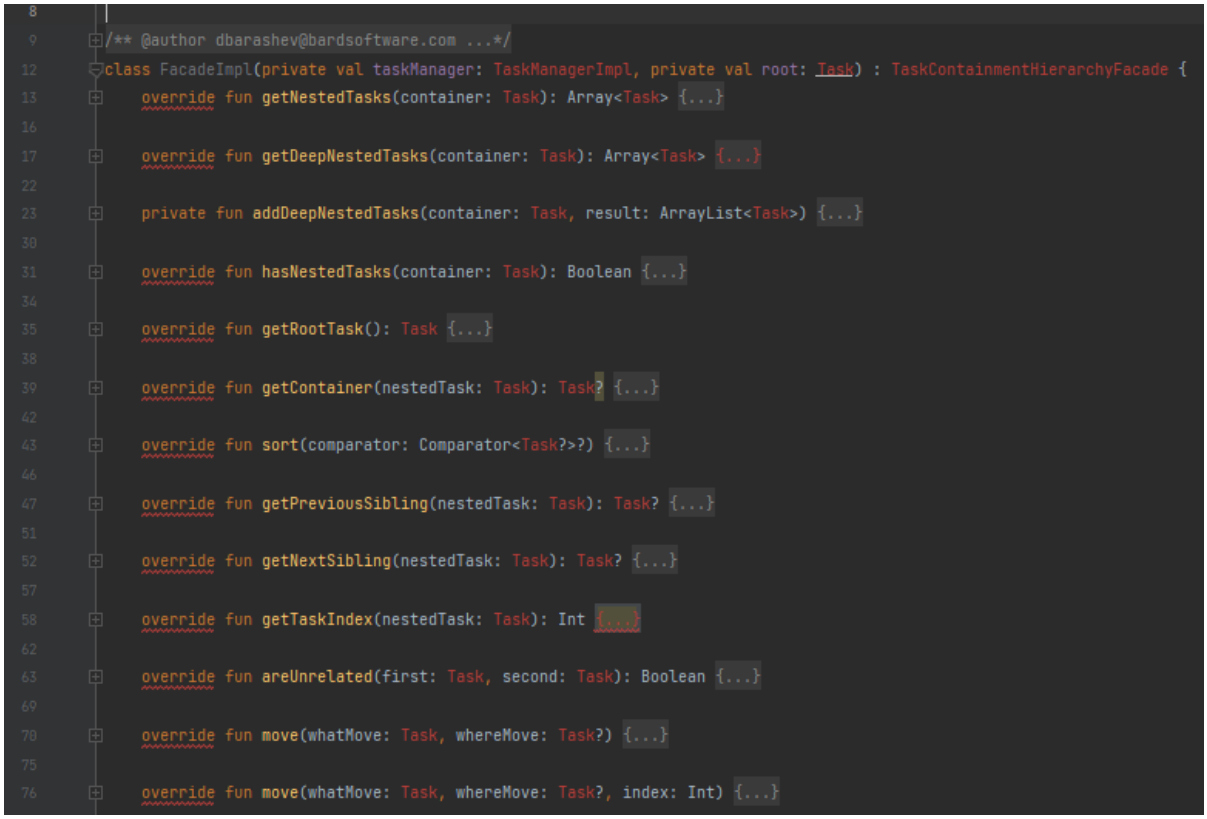
Rafael Pereira (60700)

## Reviewer

Rafael Martins (60602)

# Façade

## Illustrating code snippet



```
/** @author dbarashev@bardsoftware.com ...*/
class FacadeImpl(private val taskManager: TaskManagerImpl, private val root: Task) : TaskContainmentHierarchyFacade {
    override fun getNestedTasks(container: Task): Array<Task> {...}

    override fun getDeepNestedTasks(container: Task): Array<Task> {...}

    private fun addDeepNestedTasks(container: Task, result: ArrayList<Task>) {...}

    override fun hasNestedTasks(container: Task): Boolean {...}

    override fun getRootTask(): Task {...}

    override fun getContainer(nestedTask: Task): Task? {...}

    override fun sort(comparator: Comparator<Task?>?) {...}

    override fun getPreviousSibling(nestedTask: Task): Task? {...}

    override fun getNextSibling(nestedTask: Task): Task? {...}

    override fun getTaskIndex(nestedTask: Task): Int {...}

    override fun areUnrelated(first: Task, second: Task): Boolean {...}

    override fun move(whatMove: Task, whereMove: Task?) {...}

    override fun move(whatMove: Task, whereMove: Task?, index: Int) {...}
```

## Class location

/ganttproject/src/main/java/net/sourceforge/ganttproject/task/TaskTreeFacade.kt

## Explanation

Here we have a “Façade” design pattern that encapsulates the whole Task tree hiding its complexity and making it easy to access the functionality of this subsystem.

## Author

Rafael Pereira (60700)

## Reviewer

Pedro Fernandes (60694)

# Adapter

## Illustrating code snippet

```kotlin
open class TaskListenerAdapter(private val allEventsHandler: ()->Unit = {}) : TaskListener {
  var dependencyAddedHandler: ((TaskDependencyEvent) -> Unit)? = null
  var dependencyRemovedHandler: ((TaskDependencyEvent) -> Unit)? = null
  var dependencyChangedHandler: ((TaskDependencyEvent) -> Unit)? = null
  var taskAddedHandler: ((TaskHierarchyEvent) -> Unit)? = null
  var taskRemovedHandler: ((TaskHierarchyEvent) -> Unit)? = null
  var taskMovedHandler: ((TaskHierarchyEvent) -> Unit)? = null
  var taskPropertiesChangedHandler: ((TaskPropertyEvent) -> Unit)? = null
  var taskProgressChangedHandler: ((TaskPropertyEvent) -> Unit)? = null
  var taskScheduleChangedHandler: ((TaskScheduleEvent) -> Unit)? = null
  var taskModelResetHandler: (() -> Unit)? = null

  override fun taskScheduleChanged(e: TaskScheduleEvent) {
    taskScheduleChangedHandler?.also { it(e) } ?: allEventsHandler()
  }

  override fun dependencyAdded(e: TaskDependencyEvent) {
    dependencyAddedHandler?.also { it(e) } ?: allEventsHandler()
  }

  override fun dependencyRemoved(e: TaskDependencyEvent) {
    dependencyRemovedHandler?.also { it(e) } ?: allEventsHandler()
  }

  override fun dependencyChanged(e: TaskDependencyEvent) {
    dependencyChangedHandler?.also { it(e) } ?: allEventsHandler()
  }
```

## Class location

/ganttproject/src/main/java/net/sourceforge/ganttproject/task/event/TaskListenerAdapter.kt

## Design pattern

Here we have a “Adapter” design pattern which facilitates the handling of the events by delegating a single handler for all them and allowing two different classes that use different types of interfaces to communicate with each other.

## Author

Rafael Pereira (60700)

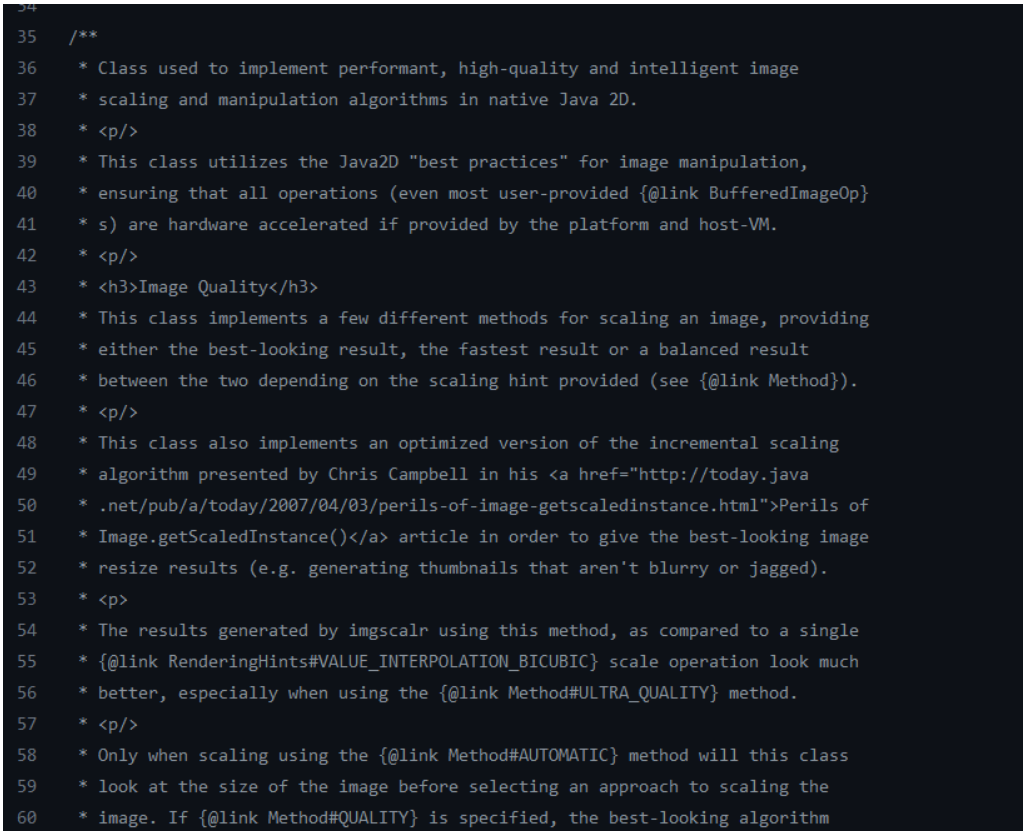
## Reviewer

Pedro Lopes (57514)

# Code Smells

## Too many comments

### Illustrating code snippet



```
35  /**
36   * Class used to implement performant, high-quality and intelligent image
37   * scaling and manipulation algorithms in native Java 2D.
38   * <p/>
39   * This class utilizes the Java2D "best practices" for image manipulation,
40   * ensuring that all operations (even most user-provided {@link BufferedImageOp}
41   * s) are hardware accelerated if provided by the platform and host-VM.
42   * <p/>
43   * <h3>Image Quality</h3>
44   * This class implements a few different methods for scaling an image, providing
45   * either the best-looking result, the fastest result or a balanced result
46   * between the two depending on the scaling hint provided (see {@link Method}).
47   * <p/>
48   * This class also implements an optimized version of the incremental scaling
49   * algorithm presented by Chris Campbell in his <a href="http://today.java
50   * .net/pub/a/today/2007/04/03/perils-of-image-getscaledinstance.html">Perils of
51   * Image.getScaledInstance()</a> article in order to give the best-looking image
52   * resize results (e.g. generating thumbnails that aren't blurry or jagged).
53   * <p>
54   * The results generated by imgscalr using this method, as compared to a single
55   * {@link RenderingHints#VALUE_INTERPOLATION_BICUBIC} scale operation look much
56   * better, especially when using the {@link Method#ULTRA_QUALITY} method.
57   * <p/>
58   * Only when scaling using the {@link Method#AUTOMATIC} method will this class
59   * look at the size of the image before selecting an approach to scaling the
60   * image. If {@link Method#QUALITY} is specified, the best-looking algorithm
```

### Class location

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

### Explanation

In the Scalr.java class there are very extensive comments (for example the comment of line 35 to 196) that explain the functioning of the methods and the class, which indicate that for the author the code is not very clear

### Refactoring proposal

Summarize the comments to be smaller, clearer, and more direct

### Author

Guilherme Fernandes (60045)

### Reviewer

Pedro Fernandes (60694)

# Large methods

## Illustrating code snippet

```
34
35      private static Calendar getCalendar(String isodate)
36              throws InvalidDateException {
37          // YYYY-MM-DDThh:mm:ss.sTZD
38          StringTokenizer st = new StringTokenizer(isodate, "-T:.+Z", true);
39
40          Calendar calendar = new GregorianCalendar();
41          calendar.clear();
42          try {
43              // Year
44              if (st.hasMoreTokens()) {
45                  int year = Integer.parseInt(st.nextToken());
46                  calendar.set(Calendar.YEAR, year);
47              } else {
48                  return calendar;
49              }
50              // Month
51              if (check(st, "-") && (st.hasMoreTokens())) {
52                  int month = Integer.parseInt(st.nextToken()) - 1;
53                  calendar.set(Calendar.MONTH, month);
54              } else {
55                  return calendar;
56              }
57              // Day
58              if (check(st, "-") && (st.hasMoreTokens())) {
59                  int day = Integer.parseInt(st.nextToken());
60                  calendar.set(Calendar.DAY_OF_MONTH, day);
61              } else {
62                  return calendar;
63              }
```

## Class location

/biz.ganttproject.core/src/main/java/org/w3c/util/DateParser.java

## Explanation

In the DateParser.java class there are very extensive methods that are difficult to understand (for example the method from lines 35 to 156)

## Refactoring proposal

The most appropriate solution would be to divide the method into sub methods

## Author

Guilherme Fernandes (60045)

## Reviewer

Rafael Pereira (60700)

# Shotgun surgery

## Illustrating code snippet

```
187     public static String getIsoDate(Date date) {
188         Calendar calendar = new GregorianCalendar();
189         calendar.setTime(date);
190         StringBuffer buffer = new StringBuffer(getIsoDateNoHours(date));
191         buffer.append("T");
192         buffer.append(twoDigit(calendar.get(Calendar.HOUR_OF_DAY)));
193         buffer.append(":");
194         buffer.append(twoDigit(calendar.get(Calendar.MINUTE)));
195         buffer.append(":");
196         buffer.append(twoDigit(calendar.get(Calendar.SECOND)));
197         buffer.append(".");
198         buffer.append(twoDigit(calendar.get(Calendar.MILLISECOND) / 10));
199         buffer.append("Z");
200         return buffer.toString();
201     }
202 
203     /**
204      * Generate a ISO 8601 date
205      *
206      * @param date
207      *            a Date instance
208      * @return a string representing the date in the ISO 8601 format
209      */
210     public static String getIsoDateNoMillis(Date date) {
211         Calendar calendar = new GregorianCalendar();
212         calendar.setTime(date);
213         StringBuffer buffer = new StringBuffer(getIsoDateNoHours(date));
214         buffer.append("T");
215         buffer.append(twoDigit(calendar.get(Calendar.HOUR_OF_DAY)));
216         buffer.append(":");
217         buffer.append(twoDigit(calendar.get(Calendar.MINUTE)));
218         buffer.append(":");
219         buffer.append(twoDigit(calendar.get(Calendar.SECOND)));
220         buffer.append("Z");
221         return buffer.toString();
```

## Class location

/biz.ganttproject.core/src/main/java/org/w3c/util/DateParser.java

## Code smell

In the DateParser.java class there are several equal constants scattered throughout the code, if one of them changed it would be necessary to change it in several places in the class

## Refactoring proposal

The best way to deal with this problem would be to have a constant defined for each of these constants and if it was necessary to change its value, it would only be changed in one place.

## Author

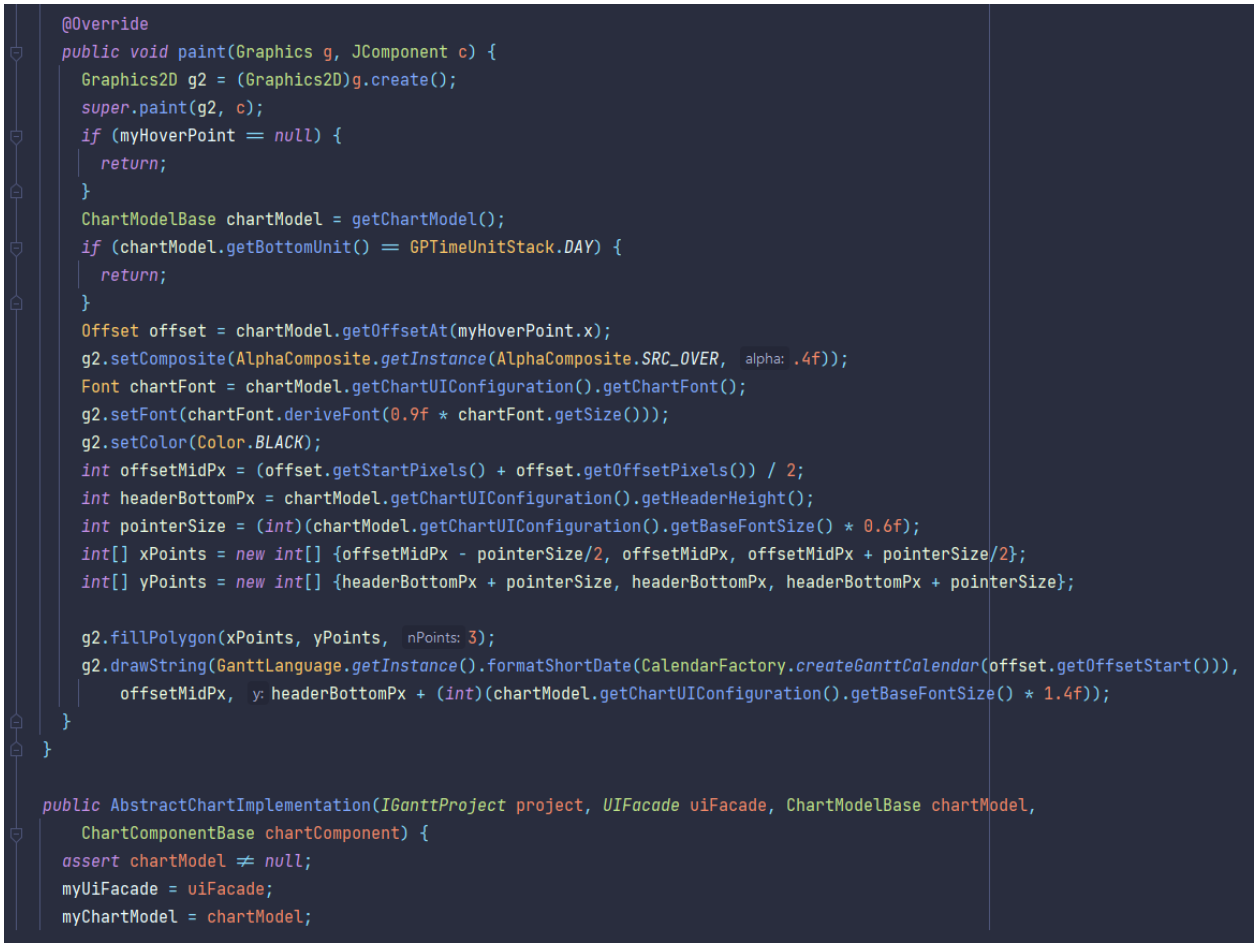
Guilherme Fernandes (60045)

## Reviewer

Rafael Martins (60602)

# No comments

## Illustrating code snippet



```
@Override
public void paint(Graphics g, JComponent c) {
  Graphics2D g2 = (Graphics2D)g.create();
  super.paint(g2, c);
  if (myHoverPoint == null) {
    return;
  }
  ChartModelBase chartModel = getChartModel();
  if (chartModel.getBottomUnit() == GPTimeUnitStack.DAY) {
    return;
  }
  Offset offset = chartModel.getOffsetAt(myHoverPoint.x);
  g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, alpha: .4f));
  Font chartFont = chartModel.getChartUIConfiguration().getChartFont();
  g2.setFont(chartFont.deriveFont(0.9f * chartFont.getSize()));
  g2.setColor(Color.BLACK);
  int offsetMidPx = (offset.getStartPixels() + offset.getOffsetPixels()) / 2;
  int headerBottomPx = chartModel.getChartUIConfiguration().getHeaderHeight();
  int pointerSize = (int)(chartModel.getChartUIConfiguration().getBaseFontSize() * 0.6f);
  int[] xPoints = new int[] {offsetMidPx - pointerSize/2, offsetMidPx, offsetMidPx + pointerSize/2};
  int[] yPoints = new int[] {headerBottomPx + pointerSize, headerBottomPx, headerBottomPx + pointerSize};

  g2.fillPolygon(xPoints, yPoints, nPoints: 3);
  g2.drawString(GanttLanguage.getInstance().formatShortDate(CalendarFactory.createGanttCalendar(offset.getOffsetStart())),
      offsetMidPx, y: headerBottomPx + (int)(chartModel.getChartUIConfiguration().getBaseFontSize() * 1.4f));
 }
}

public AbstractChartImplementation(IGanttProject project, UIFacade uiFacade, ChartModelBase chartModel,
    ChartComponentBase chartComponent) {
  assert chartModel != null;
  myUiFacade = uiFacade;
  myChartModel = chartModel;
```

## Class location

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

## Explanation

The class is missing any form of comments describing the functionality of its methods, making it harder to understand.

## Refactoring proposal

Add descriptive enough comments to make it easier to read the code.

## Author

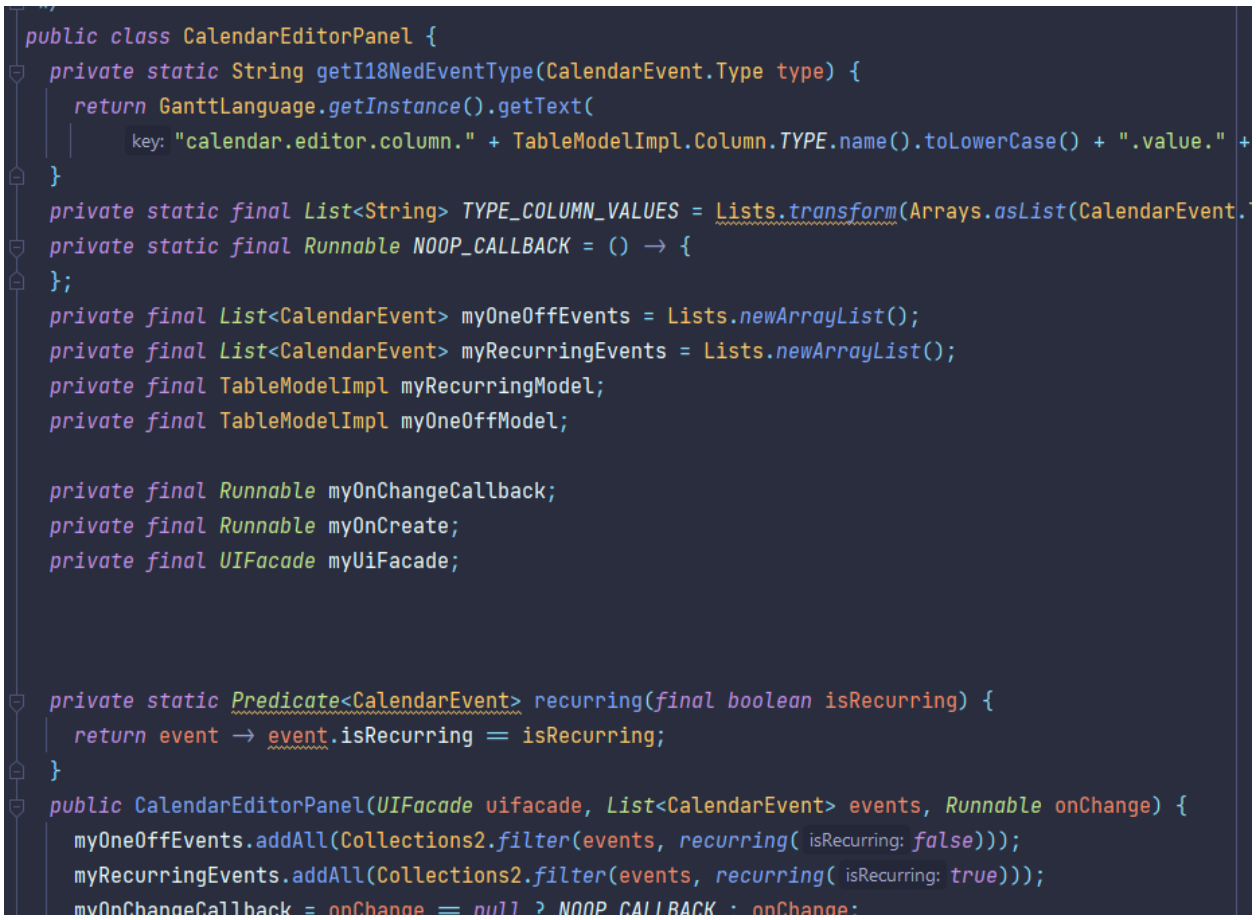
Pedro Fernandes (60694)

## Reviewer

Guilherme Fernandes (60045)

# Large class

## Illustrating code snippet



```
public class CalendarEditorPanel {
  private static String getI18NedEventType(CalendarEvent.Type type) {
    return GanttLanguage.getInstance().getText(
        key: "calendar.editor.column." + TableModelImpl.Column.TYPE.name().toLowerCase() + ".value." +
  }
  private static final List<String> TYPE_COLUMN_VALUES = Lists.transform(Arrays.asList(CalendarEvent.
  private static final Runnable NOOP_CALLBACK = () -> {
  };
  private final List<CalendarEvent> myOneOffEvents = Lists.newArrayList();
  private final List<CalendarEvent> myRecurringEvents = Lists.newArrayList();
  private final TableModelImpl myRecurringModel;
  private final TableModelImpl myOneOffModel;

  private final Runnable myOnChangeCallback;
  private final Runnable myOnCreate;
  private final UIFacade myUiFacade;



  private static Predicate<CalendarEvent> recurring(final boolean isRecurring) {
    return event -> event.isRecurring == isRecurring;
  }
  public CalendarEditorPanel(UIFacade uifacade, List<CalendarEvent> events, Runnable onChange) {
    myOneOffEvents.addAll(Collections2.filter(events, recurring( isRecurring: false)));
    myRecurringEvents.addAll(Collections2.filter(events, recurring( isRecurring: true)));
    myOnChangeCallback = onChange == null ? NOOP_CALLBACK : onChange;
```

## Class location

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

## Explanation

This class is too large, indicating it has too many responsibilities and less organization.

## Refactoring proposal

Creating more specific classes that take care of smaller sets of actions the original class was responsible for.

## Author

Pedro Fernandes (60694)

## Reviewer

Rafael Pereira (60700)

# Shotgun surgery

## Illustrating code snippet

```
public ChartUIConfiguration(UIConfiguration projectConfig) {
  mySpanningRowTextFont = Fonts.TOP_UNIT_FONT;
  mySpanningHeaderBackgroundColor = new Color( r: 0.93f,  g: 0.93f,  b: 0.93f);
  myHeaderBorderColor = new Color( r: 0.482f,  g: 0.482f,  b: 0.482f);
  myWorkingTimeBackgroundColor = Color.WHITE;
  myHolidayTimeBackgroundColor = new Color( r: 0.9f,  g: 0.9f,  b: 0.9f);
  myPublicHolidayTimeBackgroundColor = new Color( r: 240,  g: 220,  b: 240);
  // myHeaderBorderColor = new Color(0f, 1f, 0f);
  myBottomUnitGridColor = new Color( r: 0.482f,  g: 0.482f,  b: 0.482f);
  myProjectConfig = projectConfig;
  myChartStylesOption = new ChartPropertiesOption();
}
```

## Class location

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

## Explanation

Changing colors in any part of the code may be harder than intended given that there is no general definition for such.

## Refactoring proposal

Create constants responsible for holding each color used throughout the code.

## Author

Pedro Fernandes (60694)

## Reviewer

Rafael Martins (60602)

# Long parameter list

## Illustrating code snippet

```
public static Offset createFullyClosed(TimeUnit timeUnit, Date anchor, Date closedStartDate, Date closedEndDate,
    int startPixels, int endPixels, int dayMask) {
  return new Offset(timeUnit, anchor, closedStartDate, closedEndDate, startPixels, endPixels, dayMask);
}
```

## Class location

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

## Code smell (Large Parameter List)

In the Offset.java class there is a method with an extensive parameter list in line 109, this increases the chance of something going wrong.

## Refactoring proposal

The best solution would be to introduce parameter objects.

## Author

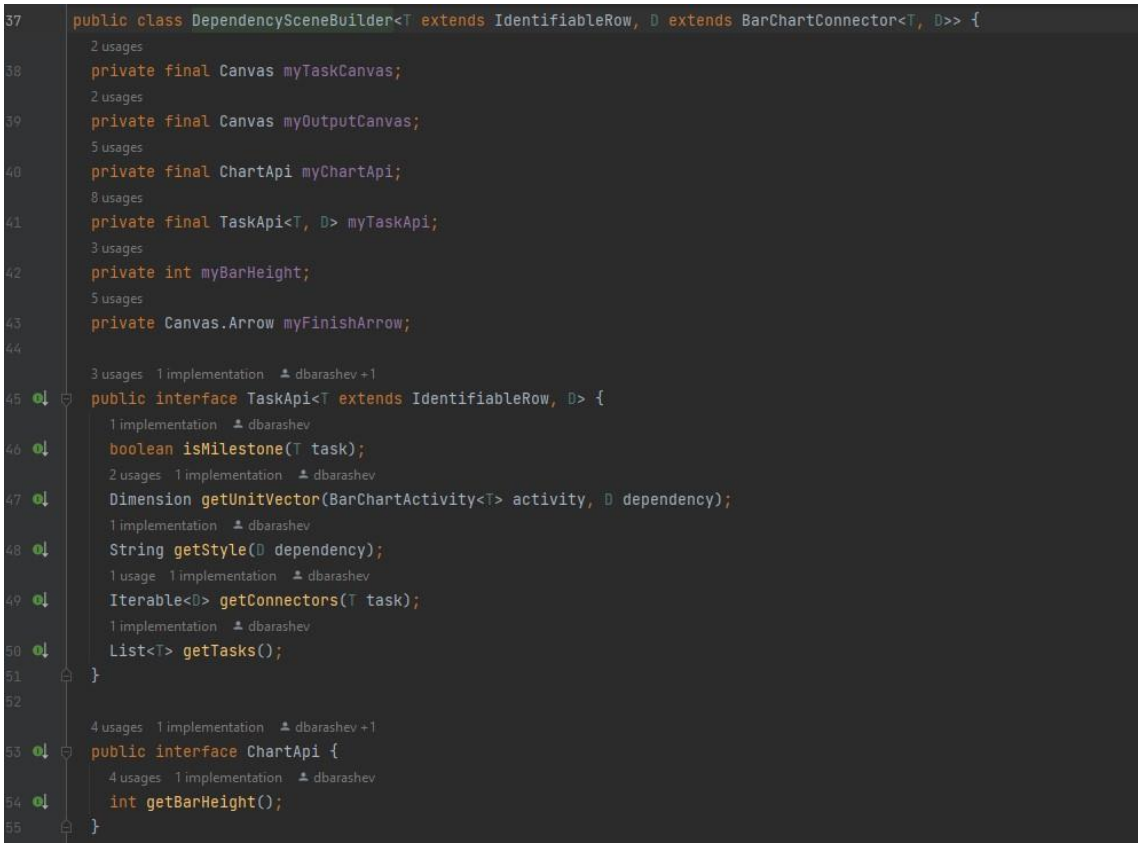
Pedro Lopes (57514)

## Reviewer

Guilherme Fernandes (60045)

# No comments

## Illustrating code snippet



```
37  public class DependencySceneBuilder<T extends IdentifiableRow, D extends BarChartConnector<T, D>> {
      2 usages
38    private final Canvas myTaskCanvas;
      2 usages
39    private final Canvas myOutputCanvas;
      5 usages
40    private final ChartApi myChartApi;
      8 usages
41    private final TaskApi<T, D> myTaskApi;
      3 usages
42    private int myBarHeight;
      5 usages
43    private Canvas.Arrow myFinishArrow;
44
      3 usages  1 implementation  dbarashev +1
45    public interface TaskApi<T extends IdentifiableRow, D> {
        1 implementation  dbarashev
46      boolean isMilestone(T task);
        2 usages  1 implementation  dbarashev
47      Dimension getUnitVector(BarChartActivity<T> activity, D dependency);
        1 implementation  dbarashev
48      String getStyle(D dependency);
        1 usage  1 implementation  dbarashev
49      Iterable<D> getConnectors(T task);
        1 implementation  dbarashev
50      List<T> getTasks();
51    }
52
      4 usages  1 implementation  dbarashev +1
53    public interface ChartApi {
        4 usages  1 implementation  dbarashev
54      int getBarHeight();
55    }
```

## Class location

/ganttproject/biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/scene/gantt/DependencySceneBuilder.java

## Explanation

The DependencySceneBuilder.java does not contain any comments regarding what the implemented functions do. Considering their size this could cause confusion to someone that is not familiar with the code.

## Refactoring proposal

Add enough comments so that at least the purpose of the most extensive functions is clear.

## Author

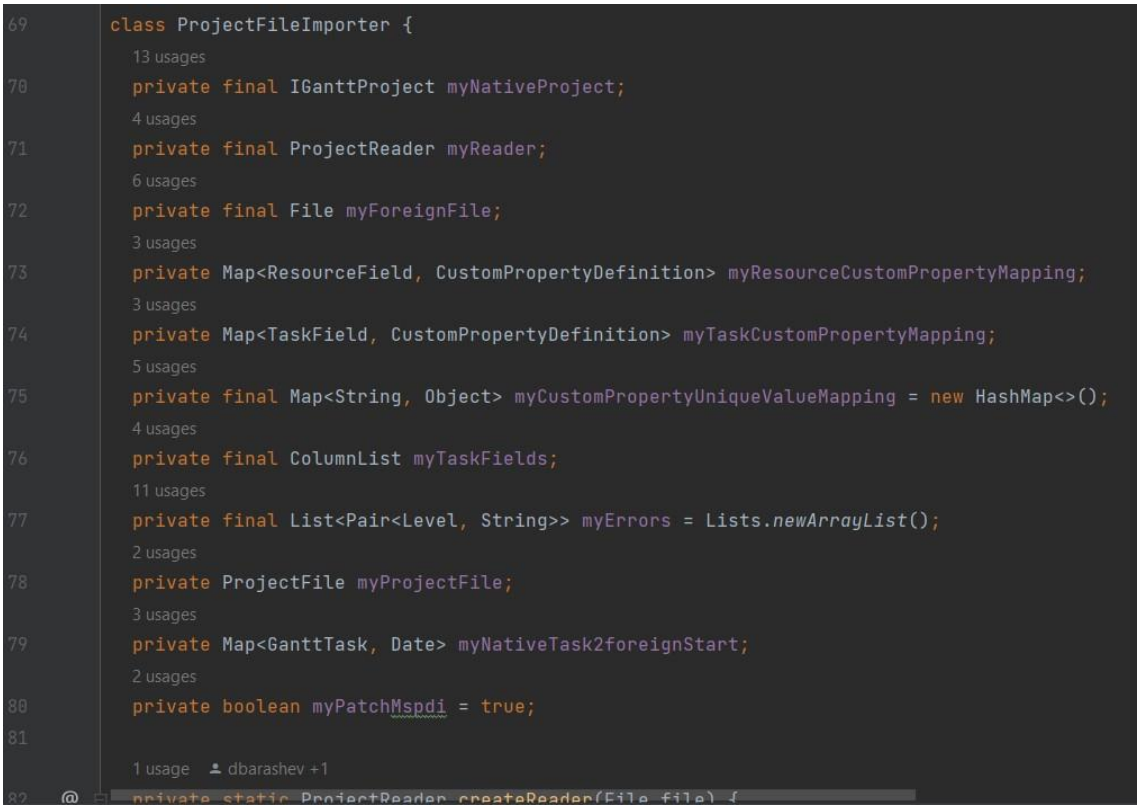
Pedro Lopes (57514)

## Reviewer

Rafael Pereira (60700)

# Large class

## Illustrating code snippet



```
69      class ProjectFileImporter {
          13 usages
70        private final IGanttProject myNativeProject;
          4 usages
71        private final ProjectReader myReader;
          6 usages
72        private final File myForeignFile;
          3 usages
73        private Map<ResourceField, CustomPropertyDefinition> myResourceCustomPropertyMapping;
          3 usages
74        private Map<TaskField, CustomPropertyDefinition> myTaskCustomPropertyMapping;
          5 usages
75        private final Map<String, Object> myCustomPropertyUniqueValueMapping = new HashMap<>();
          4 usages
76        private final ColumnList myTaskFields;
          11 usages
77        private final List<Pair<Level, String>> myErrors = Lists.newArrayList();
          2 usages
78        private ProjectFile myProjectFile;
          3 usages
79        private Map<GanttTask, Date> myNativeTask2foreignStart;
          2 usages
80        private boolean myPatchMspdi = true;
81
            1 usage  dbarashev +1
82    @     private static ProjectReader createReader(File file) {
```

## Class location

/ganttproject/biz.ganttproject.impex.msproject2/src/main/java/biz/ganttproject/impex/msproject2/ProjectFileImporter.java

## Explanation

The ProjectFileImporter.java is a very extensive class (733 lines), this means it contains a lot of responsabilities and would require a lot of comments to document it's functionalities.

## Refactoring proposal

The proper way to handle this is to divide the responsibilities into subclasses and hide the implementation from the large class.

## Author

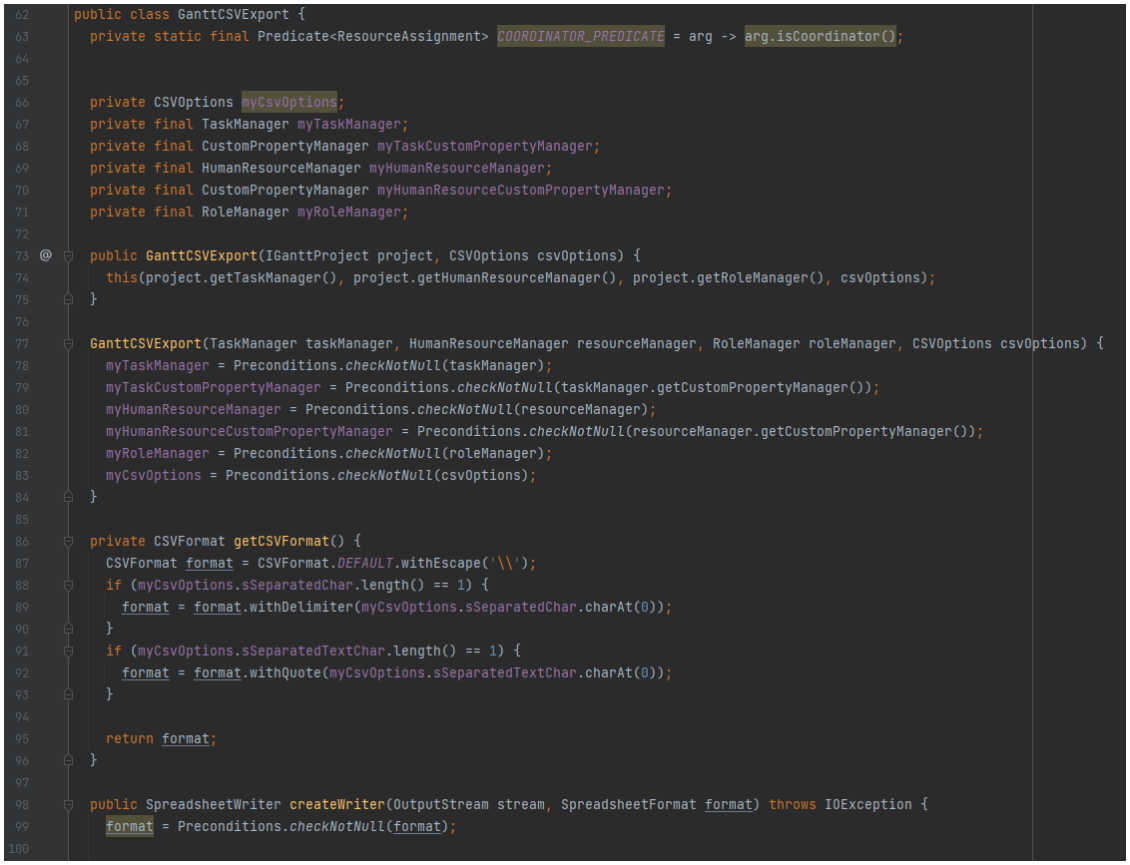
Pedro Lopes (57514)

## Reviewer

Rafael Martins (60602)

# No comments

## Illustrating code snippet



```
public class GanttCSVExport {
  private static final Predicate<ResourceAssignment> COORDINATOR_PREDICATE = arg -> arg.isCoordinator();


  private CSVOptions myCsvOptions;
  private final TaskManager myTaskManager;
  private final CustomPropertyManager myTaskCustomPropertyManager;
  private final HumanResourceManager myHumanResourceManager;
  private final CustomPropertyManager myHumanResourceCustomPropertyManager;
  private final RoleManager myRoleManager;

@ public GanttCSVExport(IGanttProject project, CSVOptions csvOptions) {
    this(project.getTaskManager(), project.getHumanResourceManager(), project.getRoleManager(), csvOptions);
  }

  GanttCSVExport(TaskManager taskManager, HumanResourceManager resourceManager, RoleManager roleManager, CSVOptions csvOptions) {
    myTaskManager = Preconditions.checkNotNull(taskManager);
    myTaskCustomPropertyManager = Preconditions.checkNotNull(taskManager.getCustomPropertyManager());
    myHumanResourceManager = Preconditions.checkNotNull(resourceManager);
    myHumanResourceCustomPropertyManager = Preconditions.checkNotNull(resourceManager.getCustomPropertyManager());
    myRoleManager = Preconditions.checkNotNull(roleManager);
    myCsvOptions = Preconditions.checkNotNull(csvOptions);
  }

  private CSVFormat getCSVFormat() {
    CSVFormat format = CSVFormat.DEFAULT.withEscape('\\');
    if (myCsvOptions.sSeparatedChar.length() == 1) {
      format = format.withDelimiter(myCsvOptions.sSeparatedChar.charAt(0));
    }
    if (myCsvOptions.sSeparatedTextChar.length() == 1) {
      format = format.withQuote(myCsvOptions.sSeparatedTextChar.charAt(0));
    }

    return format;
  }

  public SpreadsheetWriter createWriter(OutputStream stream, SpreadsheetFormat format) throws IOException {
    format = Preconditions.checkNotNull(format);

```

## Class Location

/ganttproject/src/main/java/biz/ganttproject/impex/csv/GanttCSVExport.java

## Explanation

This Class has almost no comments explaining the code, making it confusing for someone who is looking at the code for the first time.

## Refactoring Proposal

Adding more comments explaining what each method does.

## Author

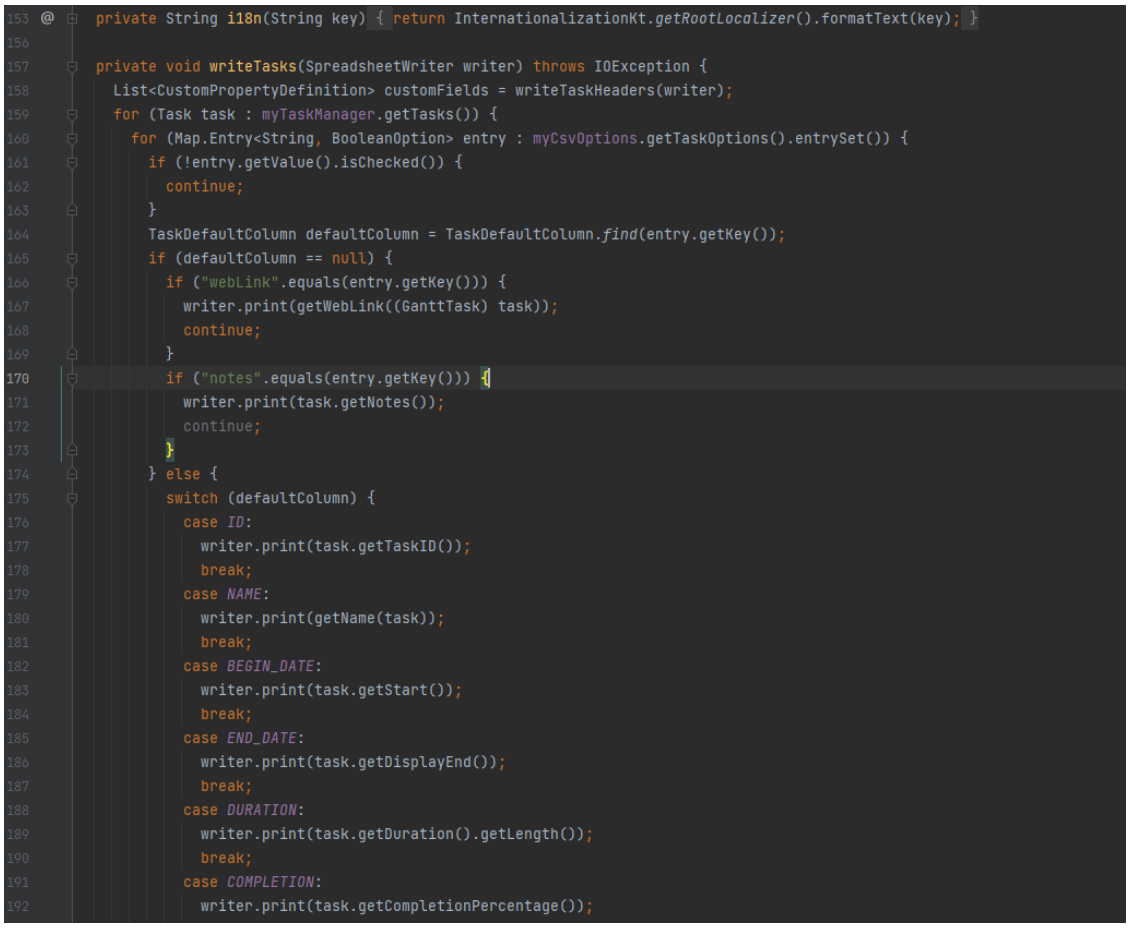
Rafael Martins (60602)

## Reviewer

Guilherme Fernandes (60045)

# Long method

## Illustrating code snippet



```
153 @   private String i18n(String key) { return InternationalizationKt.getRootLocalizer().formatText(key); }
156
157     private void writeTasks(SpreadsheetWriter writer) throws IOException {
158       List<CustomPropertyDefinition> customFields = writeTaskHeaders(writer);
159       for (Task task : myTaskManager.getTasks()) {
160         for (Map.Entry<String, BooleanOption> entry : myCsvOptions.getTaskOptions().entrySet()) {
161           if (!entry.getValue().isChecked()) {
162             continue;
163           }
164           TaskDefaultColumn defaultColumn = TaskDefaultColumn.find(entry.getKey());
165           if (defaultColumn == null) {
166             if ("webLink".equals(entry.getKey())) {
167               writer.print(getWebLink((GanttTask) task));
168               continue;
169             }
170             if ("notes".equals(entry.getKey())) {
171               writer.print(task.getNotes());
172               continue;
173             }
174           } else {
175             switch (defaultColumn) {
176               case ID:
177                 writer.print(task.getTaskID());
178                 break;
179               case NAME:
180                 writer.print(getName(task));
181                 break;
182               case BEGIN_DATE:
183                 writer.print(task.getStart());
184                 break;
185               case END_DATE:
186                 writer.print(task.getDisplayEnd());
187                 break;
188               case DURATION:
189                 writer.print(task.getDuration().getLength());
190                 break;
191               case COMPLETION:
192                 writer.print(task.getCompletionPercentage());
```

## Class Location

/ganttproject/src/main/java/biz/ganttproject/impex/csv/GanttCSVExport.java

## Explanation

This method is too long ( line 157 to 228 ) making it confusing and hard to read.

## Refactoring Proposal

Creating sub methods that help dividing the writeTasks method into smaller methods.

## Author

Rafael Martins (60602)

## Reviewer

Pedro Fernandes (60694)

# Large class

## Illustrating code snippet

```
862
863     private class ParserFactoryImpl implements ParserFactory {
864       @Override
865       public GPParser newParser() { return new GanttXMLOpen(prjInfos, getTaskManager(), getUIFacade()); }
868
869       @Override
870       public GPSaver newSaver() {
871         return new GanttXMLSaver( project: GanttProject.this, getArea(), getUIFacade(),
872           () -> myTaskTableSupplier.get().getColumnList(), () -> myTaskFilterManager);
873       }
874     }
875
876     @Override
877     public int getViewIndex() {
878       if (getTabs() == null) {
879         return -1;
880       }
881       return getTabs().getSelectedIndex();
882     }
883
884     @Override
885     public void setViewIndex(int viewIndex) {
886       if (getTabs().getTabCount() > viewIndex) {
887         getTabs().setSelectedIndex(viewIndex);
888       }
889     }
890
891     public static void setApplicationQuitCallback(Consumer<Boolean> callback) { ourQuitCallback = callback; }
894
895     public static void doQuitApplication(boolean withSystemExit) { ourQuitCallback.accept(withSystemExit); }
898     @Override
899     public void refresh() {
900       getResourcePanel().getResourceTreeTableModel().updateResources();
901       getResourcePanel().getResourceTreeTable().setRowHeight(getResourceChart().getModel().calculateRowHeight());
902       for (Chart chart : PluginManager.getCharts()) {
903         chart.reset();
```

## Class Location

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

## Explanation

This class is too long (900 lines of code). This is bad because this way it holds too many responsibilities and has to communicate with a lot of other classes.

## Refactoring Proposal

Creating smaller classes, where each one holds less responsabilities, but, when they're all put together, serve the same purpose as the GanttProject class.

## Author

Rafael Martins (60602)

## Reviewer

Pedro Lopes (57514)

# Long method

## Illustrating code snippet

```
private void doSave(OutputStream out) throws Exception {
    final TransformerHandler handler = ((SAXTransformerFactory) SAXTransformerFactory.newInstance()).newTransformerHandler();
    Transformer serializer = handler.getTransformer();
    serializer.setOutputProperty(OutputKeys.ENCODING, value: "UTF-8");
    serializer.setOutputProperty(OutputKeys.INDENT, value: "yes");
    serializer.setOutputProperty( name: "{http://xml.apache.org/xslt}indent-amount", value: "4");
    handler.setResult(new StreamResult(out));
    handler.startDocument();
    // handler.startDTD("ganttproject.sourceforge.net",
    // "-//GanttProject.org//DTD GanttProject-1.x//EN",
    // "http://ganttproject.sourceforge.net/dtd/ganttproject.dtd");
    // handler.endDTD();

    final AttributesImpl attrs = new AttributesImpl();
    addAttribute("version", GPVersion.getCurrentVersionNumber(), attrs);
    handler.startElement( uri: "", localName: "ganttproject-options", qName: "ganttproject-options", attrs);

    attrs.clear();
    // write the task Color

    // Color color = getUIConfiguration().getTaskColor();
    // attrs.addAttribute("", "red", "red", "CDATA", "" + color.getRed());
    // attrs.addAttribute("", "green", "green", "CDATA", ""
    // + color.getGreen());
    // attrs.addAttribute("", "blue", "blue", "CDATA", ""
    // + color.getBlue());
    // handler.startElement("", "task-color", "task-color", attrs);
    // handler.endElement("", "task-color", "task-color"); attrs.clear();

    Color resourceColor = myUIConfig.getResourceColor();
    if (resourceColor != null) {
        attrs.addAttribute( uri: "", localName: "resources", qName: "resources", type: "CDATA", ColorConvertion.getColor(resourceColor));
    }
    Color resourceOverloadColor = myUIConfig.getResourceOverloadColor();
    if (resourceOverloadColor != null) {
        attrs.addAttribute( uri: "", localName: "resourcesOverload", qName: "resourcesOverload", type: "CDATA",
                ColorConvertion.getColor(resourceOverloadColor));
    }
    Color resourceUnderloadColor = myUIConfig.getResourceUnderloadColor();
    if (resourceUnderloadColor != null) {
        attrs.addAttribute( uri: "", localName: "resourcesUnderload", qName: "resourcesUnderload", type: "CDATA",
                ColorConvertion.getColor(resourceUnderloadColor));
    }
    Color weekEndColor = myUIConfig.getWeekEndColor();
    if (weekEndColor != null) {
```

## Class location

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

## Explanation

In GanttOptions.java the method is too extensive (line 210 to line 358) which makes it too complex and hard to understand.

## Refactoring proposal

This method should be broken down into smaller methods to make the code simpler.

## Author

Rafael Pereira (60700)

## Reviewer

Pedro Fernandes (60694)

# Large class

## Illustrating code snippet

```
100  public abstract class GPTreeTableBase extends JXTreeTable implements CustomPropertyListener {
101    private final IGanttProject myProject;
102    private final UIFacade myUiFacade;
103    private final TableHeaderUiFacadeImpl myTableHeaderFacade = new TableHeaderUiFacadeImpl();
104    private final CustomPropertyManager myCustomPropertyManager;
105    private final JScrollPane myScrollPane = new JScrollPane() {
106      @Override
107      public void applyComponentOrientation(ComponentOrientation o) {
108        super.applyComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
109      }
110    };
111
112    private GPAction myEditCellAction = new GPAction("tree.edit") {
113      @Override
114      public void actionPerformed(ActionEvent e) {
115        JTable t = getTable();
116        if (t.getSelectedRow() < 0) {
117          return;
118        }
119        if (t.getSelectedColumn() < 0) {
120          t.getColumnModel().getSelectionModel().setSelectionInterval(0, 0);
121        }
122        editCellAt(t.getSelectedRow(), t.getSelectedColumn());
123      }
124    };
125    private GPAction myManageColumnsAction = new GPAction("columns.manage.label") {
126      @Override
127      public void actionPerformed(ActionEvent e) {
128        ColumnManagerKt.showResourceColumnManager(myTableHeaderFacade,
129            myCustomPropertyManager, getUiFacade().getUndoManager(), getProject().getProjectDatabase(), Apply
130      }
131    };
```

## Class location

/biz.ganttproject.core/src/main/java/net/sourceforge/ganttproject/GPTreeTableBase.java

## Explanation

The GPTreeTableBase.java class is long, having 1271 lines of code, which is a sign that it has too many responsibilities.

## Refactoring proposal

The most appropriate solution would be to split the class into subclasses and redistribute the responsibilities.

## Author

Rafael Pereira (60700)

## Reviewer

Rafael Martins (60602)

# Too many comments

## Illustrating code snippet

```
149   public boolean editCellAt(final int row, final int column, EventObject e) {
150     if (e instanceof KeyEvent) {
151       KeyEvent ke = (KeyEvent) e;
152       // If editing was triggered by keyboard action, we want to check if
153       // it should actually start editing. There are actions, such as Alt+arrow which
154       // may trigger editCellAt but which are not supposed to start editing.
155       if (!isStartEditingEvent(ke, true)) {
156         return false;
157       }
158     }
159     if (e instanceof MouseEvent) {
160       // Here we have to distinguish between double-click/drag start and two consecutive single-clicks.
161       // The first single click on a cell should make the cell selected, while single click
162       // on selected cell should start editing. The problem is that when user double-clicks
163       // a selected cell, single-click event comes first. So in this case we postpone editing start,
164       // schedule a task and cancel that task if we get double-click before task starts executing.
165       MouseEvent me = (MouseEvent) e;
166       if (me.getClickCount() == 2 && me.getButton() == MouseEvent.BUTTON1) {
167         // "Cancel" the task and show properties.
168         setEditingStartExpected(false);
169         if (getTable().getSelectedRow() != -1) {
170           me.consume();
171           myPropertiesAction.actionPerformed(null);
172           return false;
173         }
174       } else if (me.getClickCount() == 1 && me.getButton() == MouseEvent.BUTTON1) {
175         // Clicks in boolean columns are special: they don't need the editor, and we can toggle the value
176         // at the same time with changing the selection.
177         var columnClass = getColumnClass(column);
```

## Class location

/biz.ganttproject.core/src/main/java/net/sourceforge/ganttproject/GPTreeTableBase.java

## Explanation

In this section of the GPTreeTableBase.java class there is an excessive use of comments to explain the editCellAt() method which indicates poor design.

## Refactoring proposal

The best way to deal with this problem would be to decompose the method into simpler to understand submethods and decrease the use of comments within the methods.

## Author

Rafael Pereira (60700)

## Reviewer

Pedro Lopes (57514)