NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Department of Computer Science
NOVA School of Science and Technology

THE S TEAM

Cláudia Santos	57049
Pedro Grilo	59213
Guilherme Fernandes	60173
Rui Correia	60390
Tomás Mondim	60747

21 de outubro de 2022

1 Design Patterns

1.1 Factory Method Pattern - Task 8

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

  5 usages
  private static LocaleApi ourLocaleApi;

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

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

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

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

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

biz.ganttproject.core.time.CalendarFactory

This is a Factory Method design pattern because it hides the creation of instances of the class GanttCalendar behind a factory class CalendarFactory.

- **Product Object:** GanttCalendar
  The object type varies depending on the Locale, instead of having subclasses. The way to specify the product object type here is by setting the Locale (via the setLocaleApi method).
- **Factory Object:** CalendarFactory
- **Factory Method:** createGanttCalendar
  It's a static method, so that, even though CalendarFactory isn't a singleton, the methods may be called directly from the factory object.

1.2 Façade Pattern - Task 10

```java
public class CustomColumnsManager implements CustomPropertyManager {
  13 usages
  private final CustomColumnsStorage myStorage;

  dbarashev
  public CustomColumnsManager() {
    myStorage = new CustomColumnsStorage( manager: this);
  }

  2 usages  dbarashev
  private void addNewCustomColumn(CustomColumn customColumn) {
    assert customColumn != null;
    myStorage.addCustomColumn(customColumn);
  }

  dbarashev
  @Override
  public void addListener(CustomPropertyListener listener) { myStorage.addCustomColumnsListener(listener); }

  dbarashev
  @Override
  public List<CustomPropertyDefinition> getDefinitions() {
    return new ArrayList<CustomPropertyDefinition>(myStorage.getCustomColums());
  }

  dbarashev +1
  @Override
  public CustomPropertyDefinition createDefinition(String id, String typeAsString, String name,
      String defaultValueAsString) {
    CustomPropertyDefinition stub = PropertyTypeEncoder.INSTANCE.decodeTypeAndDefaultValue(typeAsString,
        defaultValueAsString);
    CustomColumn result = new CustomColumn( manager: this, name, stub.getPropertyClass(), stub.getDefaultValue());
    result.setId(id);
    addNewCustomColumn(result);
    return result;
  }
```

net.sourceforge.ganttproject.task.CustomColumnsManager

This is a Façade design pattern because it provides a unified interface to a subsystem (the subsystem of classes handling "custom properties") hiding its complexity, and provides a point of entry to it: the CustomColumnsManager class.

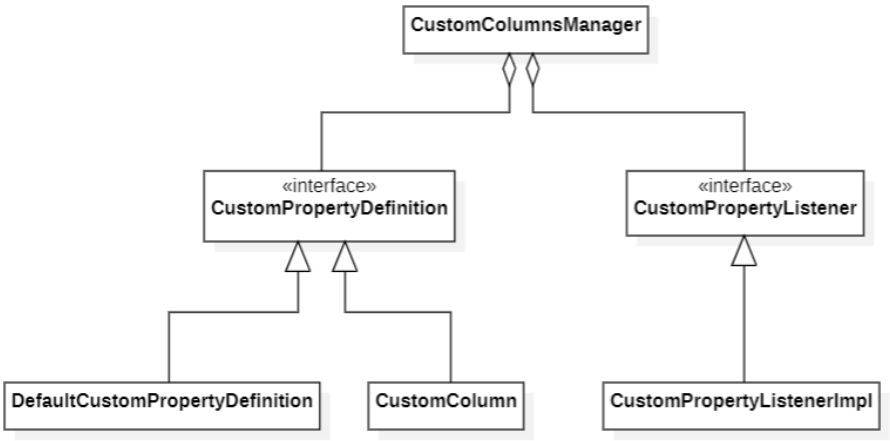
- **Façade class:** CustomColumnsManager
- **Interfaces it provides access to:** CustomPropertiesDefinition and CustomPropertyListener (using methods like addNewCustomColumn, addListener, to name a few)

```java
public class CustomColumn implements CustomPropertyDefinition {
  6 usages
  private String id = null;

  6 usages
  private String name = null;
```

net.sourceforge.ganttproject.task.CustomColumn

```java
public class DefaultCustomPropertyDefinition implements CustomPropertyDefinition {
  5 usages
  private String myName;
  7 usages
  private final String myID;
```

net.sourceforge.ganttproject.DefaultCustomPropertyDefinition



An excerpt of the classes that make up this design pattern.

1.3 Adapter Pattern - Task 11

```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

  Dmitry Barashev +1
  override fun taskScheduleChanged(e: TaskScheduleEvent) {
    taskScheduleChangedHandler?.also { it(e) } ?: allEventsHandler()
  }

  Dmitry Barashev +1
  override fun dependencyAdded(e: TaskDependencyEvent) {
    dependencyAddedHandler?.also { it(e) } ?: allEventsHandler()
  }
```

net.sourceforge.ganttproject.task.event.TaskListenerAdapter

This is an Adapter design pattern because it provides a compatible interface between the event handling services and the client that wants to adapt the handling of events to its needs, which would be incompatible otherwise.

- **Adapter class:** TaskListenerAdapter
- **Target interface:** TaskListener

```java
public interface TaskListener extends EventListener {

  1 usage  7 implementations  dbarashev
  void taskScheduleChanged(TaskScheduleEvent e);

  1 usage  5 implementations  dbarashev
  void dependencyAdded(TaskDependencyEvent e);

  1 usage  5 implementations  dbarashev
  void dependencyRemoved(TaskDependencyEvent e);

  1 usage  4 implementations  dbarashev
  void dependencyChanged(TaskDependencyEvent e);
```

net.sourceforge.ganttproject.task.event.TaskListener

An example of a client class that uses this adapter class to adapt the handling of events is TaskManagerImpl.

```java
TaskManagerImpl.java
                Dmitry Barashev +1
238             addTaskListener(new TaskListenerAdapter() {
                  1 usage  Dmitry Barashev +1
239               @Override
240               public void dependencyChanged(@NotNull TaskDependencyEvent e) {
241                 if (areEventsEnabled) {
242                   myScheduler.run();
243                 }
244               }
245
                  1 usage  Dmitry Barashev +1
246               @Override
247               public void taskScheduleChanged(@NotNull TaskScheduleEvent e) { processCriticalPath(getRootTask()); }
250
                  1 usage  Dmitry Barashev +1
251               @Override
252               public void dependencyAdded(@NotNull TaskDependencyEvent e) { processCriticalPath(getRootTask()); }
255
                  1 usage  Dmitry Barashev +1
256               @Override
257               public void dependencyRemoved(@NotNull TaskDependencyEvent e) { processCriticalPath(getRootTask()); }
260
                  1 usage  Dmitry Barashev +1
261               @Override
262               public void taskAdded(@NotNull TaskHierarchyEvent e) { processCriticalPath(getRootTask()); }
```

net.sourceforge.ganttproject.task.TaskManagerImpl

1.4 Reviews

Reviewer Name: Pedro Grilo
Design Pattern: Factory Method Pattern
Deverias mostrar um bocado do construtor da class GanttCalendar. De resto parece-me tudo bem, pois tens uma interface LocalApi que é implementada nas classes e depois tens o metodo createGanttCalendar que utiliza o objeto que tem a tua interface. Isto tudo dentro de uma classe chamada CalendarFactory.
Reviewer Name: Guilherme Fernandes
Design Pattern: Facade Pattern
Façade class and interfaces nicely pointed out. Bonus points for the uml diagram! Looks good to me!
Reviewer Name: Rui Correia
Design Pattern: Adapter Pattern
Maybe show some more classes where the adapter is used. Other than that it looks fine.

57049, Cláudia Santos

1.5 Façade Pattern - Task 1

```
7 usages  dbarashev +4
class UIFacadeImpl extends ProgressProvider implements UIFacade {
  6 usages
  private final JFrame myMainFrame;
  2 usages
  private final ScrollingManager myScrollingManager;
  2 usages
  private final ZoomManager myZoomManager;
  5 usages
  private final GanttStatusBar myStatusBar;
  17 usages
  private final UIFacade myFallbackDelegate;
  3 usages
  private final TaskSelectionManager myTaskSelectionManager;
  3 usages
  private final List<GPOptionGroup> myOptionGroups = Lists.newArrayList();
  5 usages
  private final GPOptionGroup myOptions;
  5 usages
  private final LafOption myLafOption;
  3 usages
  private final GPOptionGroup myLogoOptions;
  5 usages
  private final DefaultFileOption myLogoOption;
  3 usages
  private final NotificationManagerImpl myNotificationManager;
  1 usage
  private final TaskView myTaskView = new TaskView();
```

```
UIFacadeImpl(JFrame mainFrame, GanttStatusBar statusBar, NotificationManagerImpl notificationManager,
             final IGanttProject project, UIFacade fallbackDelegate) {
  myMainFrame = mainFrame;
  myProject = project;
  myDialogBuilder = new DialogBuilder(mainFrame);
  myScrollingManager = new ScrollingManagerImpl();
  myZoomManager = new ZoomManager(project.getTimeUnitStack());
  myStatusBar = statusBar;
  myStatusBar.setNotificationManager(notificationManager);
  myFallbackDelegate = fallbackDelegate;
  Job.getJobManager().setProgressProvider(this);
  myTaskSelectionManager = new TaskSelectionManager(() -> project.getTaskManager());
  myNotificationManager = notificationManager;

  myLafOption = new LafOption( uiFacade: this);
  final ShortDateFormatOption shortDateFormatOption = new ShortDateFormatOption();
  final DefaultStringOption dateSampleOption = new DefaultStringOption( id: "ui.dateFormat.sample");
  dateSampleOption.setWritable(false);
  final DefaultBooleanOption dateFormatSwitchOption = new DefaultBooleanOption( id: "ui.dateFormat.switch",  initialValue: true);

  dbarashev +1
  myLanguageOption = new LanguageOption() {
    {
      dbarashev
      GanttLanguage.getInstance().addListener(new GanttLanguage.Listener() {
        dbarashev
        @Override
        public void languageChanged(GanttLanguage.Event event) {
          Locale selected = getSelectedValue();
          reloadValues(GanttLanguage.getInstance().getAvailableLocales());
          setSelectedValue(selected);
        }
```

Façade class: UIFacadeImpl
E é utilizada no GanttProjectBase.java na **linha 215**.

```
214     NotificationManagerImpl notificationManager = new NotificationManagerImpl(myContentPaneBuilder.getAnimationHost());
215     myUIFacade = new UIFacadeImpl( mainFrame: this, statusBar, notificationManager, getProject(),  fallbackDelegate: this);
216     myUiInitializationPromise = new TwoPhaseBarrierImpl<>(myUIFacade);
```

1.6 Factory Pattern - Task 2

Existe a class Factory (FontAwesomeIconFactory), que dá extends à GlyphsFactory.

```
8 usages
public class FontAwesomeIconFactory extends GlyphsFactory {

    3 usages
    private static FontAwesomeIconFactory me;

    1 usage
    private FontAwesomeIconFactory() { super(FontAwesomeIconView.class); }

    public static FontAwesomeIconFactory get() {
        if (me == null) {
            me = new FontAwesomeIconFactory();
        }
        return me;
    }

}
```

Na class Components.kt, no método buildFontAwesomeButton, usa se para criar um botão do tipo FontAwesomeIcon.

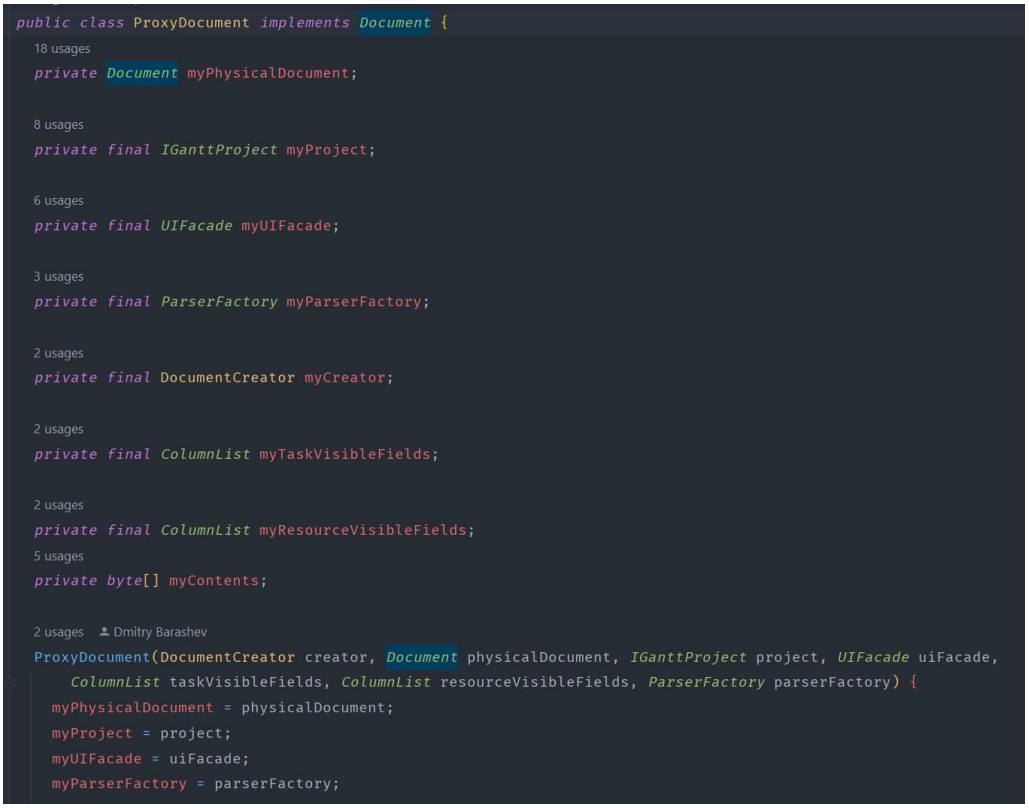
1.7 Proxy Pattern - Task 5

```
9 implementations  dbarashev +1
35  public interface Document {
        10 usages
36      String PLUGIN_ID = "net.sourceforge.ganttproject";
37
        10 usages  dbarashev
38      public enum ErrorCode { ... }
41
42      /** @return the filename of the document (can be used forthe application's ... */
        7 implementations  dbarashev
46      public String getFileName();
47
48      /** Checks, whether the document is readable. ... */
        7 implementations  dbarashev
53      public boolean canRead();
54
55      /**
56       * Checks, whether the document is writable.
57       *
58       * @return writability
59       */
        7 implementations  dbarashev
60      public IStatus canWrite();
```

Concrete classes implementing the same interface:
AbstractDocument.java

```
public abstract class AbstractDocument implements Document {

    dbarashev
    @Override
    public boolean equals(Object o) {
        if (o instanceof Document) {
            return ((Document) o).getPath().equals(this.getPath());
        }
        return false;
    }

    2 usages  1 override  dbarashev
    @Override
    public boolean acquireLock() { return true; }

    3 usages  1 override  dbarashev
    @Override
    public void releaseLock() {
    }

    8 usages  1 override  dbarashev
    @Override
    public String getFilePath() { return null; }

    1 override  dbarashev
    @Override
    public String getUsername() { return null; }
```

ProxyDocument.java



```
public class ProxyDocument implements Document {
    18 usages
    private Document myPhysicalDocument;

    8 usages
    private final IGanttProject myProject;

    6 usages
    private final UIFacade myUIFacade;

    3 usages
    private final ParserFactory myParserFactory;

    2 usages
    private final DocumentCreator myCreator;

    2 usages
    private final ColumnList myTaskVisibleFields;

    2 usages
    private final ColumnList myResourceVisibleFields;
    5 usages
    private byte[] myContents;

    2 usages  Dmitry Barashev
    ProxyDocument(DocumentCreator creator, Document physicalDocument, IGanttProject project, UIFacade uiFacade,
            ColumnList taskVisibleFields, ColumnList resourceVisibleFields, ParserFactory parserFactory) {
        myPhysicalDocument = physicalDocument;
        myProject = project;
        myUIFacade = uiFacade;
        myParserFactory = parserFactory;
```

1.8 Reviews

Reviewer Name: Tomás Mondim
Design Pattern: Facade Pattern
Parece-me bem, apenas peca de uma explicação sobre o porque deste ser um Facade Pattern. Falta também a localização das classes.

Reviewer Name: Cláudia Santos
Design Pattern: Factory Method Pattern
The exact location of the code snippets in the codebase should be provided. Should show a code snippet of the Factory Method "createIconFactory" and explain that that's the main way to obtain instances of the class FontAwesomeIcon. Hiding the creation of objects is the purpose of this pattern.

Reviewer Name: Rui Correia
Design Pattern: Proxy Pattern
Maybe explain how that interface is used in those classes.

59213, Pedro Grilo

1.9 Singleton Pattern - Task 7

```
1 usage   Dmitry Barashev
public WebDavResource createResource(WebDavResource parent, String name) {
  myResourceFactory.setCredentials(myServer.getUsername(), myServer.getPassword());
  return myResourceFactory.createResource(parent.getWebDavUri().buildChild(name));
}
```

```
public MiltonResourceImpl createResource(WebDavUri uri) {
  Key key = new Key(uri.buildUrl(), myUsername, myPassword);
  MiltonResourceImpl result = myResourceCache.get(key);
  if (result == null) {
    result = new MiltonResourceImpl(uri, getHost(uri), factory: this);
    myResourceCache.put(key, result);
  }
  return result;
}
```

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

Ensures the creation of only one Milton Resource for each user (given its username and password).

1.10 Decorator Pattern - Task 9

```java
/.../
package net.sourceforge.ganttproject.gui.options;

import ...

public abstract class OptionPageProviderBase implements OptionPageProvider {
  private String myPageID;
  private IGanttProject myProject;
  private UIFacade myUiFacade;

  protected OptionPageProviderBase(String pageID) { myPageID = pageID; }

  @Override
  public String getPageID() { return myPageID; }

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

  @Override
  public Component buildPageComponent() { return null; }

  @Override
  public void init(IGanttProject project, UIFacade uiFacade) {
    myProject = project;
    myUiFacade = uiFacade;
  }

  @Override
  public void commit() {
    for (GPOptionGroup optionGroup : getOptionGroups()) {
      optionGroup.commit();
    }
  }

  @Override
  public void setActive(boolean isActive) {
  }

  @Override
  public abstract GPOptionGroup[] getOptionGroups();
```

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

Decorator: OptionPageProviderBase
ConcreteDecorators: ProjectBasicOptionPageProvider, ProjectCalendarOptionPageProvider, ProjectRolesOptionPageProvider, ResourceChartOptionPageProvider, entre outros.

1.11 Interpreter Pattern - Task 15

```java
public class GanttLanguage {
  public class Event extends EventObject {
    public Event(GanttLanguage language) { super(language); }

    public GanttLanguage getLanguage() { return (GanttLanguage) getSource(); }
  }

  public interface Listener extends EventListener {
    public void languageChanged(Event event);
  }

  private static class CalendarFactoryImpl extends CalendarFactory implements CalendarFactory.LocaleApi {
    static void setLocaleImpl() { CalendarFactory.setLocaleApi(new CalendarFactoryImpl()); }

    @Override
    public Locale getLocale() { return GanttLanguage.getInstance().getLocale(); }

    @Override
    public DateFormat getShortDateFormat() { return GanttLanguage.getInstance().getShortDateFormat(); }
  }

  private static final GanttLanguage ganttLanguage = new GanttLanguage();

  private final SimpleDateFormat myRecurringDateFormat = new SimpleDateFormat( pattern: "MMM dd");

  private ArrayList<Listener> myListeners = new ArrayList<Listener>();

  private Locale currentLocale = null;

  private final CharSetMap myCharSetMap;

  private SimpleDateFormat currentDateFormat = null;

  private SimpleDateFormat shortCurrentDateFormat = null;

  private SimpleDateFormat myLongFormat;

  private DateFormat currentTimeFormat = null;

  private DateFormat currentDateTimeFormat = null;
```

ganttproject/src/main/java/net.sourceforge.ganttproject/language/GanttLanguage.java

Interpretador para a linguagem Gantt.

1.12 Reviews

Reviewer Name: Cláudia Santos
Design Pattern: Singleton Pattern
All code snippets should have a label identifying their exact location on the codebase. The explanation as for why it's a Singleton is not enough.
There should be an explanation about how the "myUIConfig" is this class' uniqueInstance, and that clients access the sole instance of UIConfiguration only using its access point "getUIConfiguration".

Reviewer Name: Pedro Grilo
Design Pattern: Decorator Pattern
Parece tudo ótimo, pois mostra a classe abstrata OptionPageProviderBase e identificas todas as classes que utilizam essa abstrata. Todas essas classes adicionam funcionalidades ao objeto sem alterar a estrutura, tal como é pedido no Decorator Pattern.
Podia ter prints de snippets de code das classes que utilizam OptionPageProviderBase.

Reviewer Name: Tomás Mondim
Design Pattern: Interpreter Pattern
Deveria ter uma explicação sobre o porquê desta classe usar o Interpreter Method. De resto está simples e claro!

60390, Rui Correia

1.13 Template Pattern - Task 3

```
class ResourceSaver extends SaverBase {
  void save(IGanttProject project, TransformerHandler handler) throws SAXException {
    final AttributesImpl attrs = new AttributesImpl();
    startElement( name: "resources", handler);
    saveCustomColumnDefinitions(project, handler);

    for (HumanResource p : project.getHumanResourceManager().getResources()) {
      addAttribute( name: "id", p.getId(), attrs);
      addAttribute( name: "name", p.getName(), attrs);
      addAttribute( name: "function", p.getRole().getPersistentID(), attrs);
      addAttribute( name: "contacts", p.getMail(), attrs);
      addAttribute( name: "phone", p.getPhone(), attrs);
      startElement( name: "resource", attrs, handler);
      {
        saveRates(p, handler);
        saveCustomProperties(project, p, handler);
      }
      endElement( name: "resource", handler);
    }
    endElement( name: "resources", handler);
  }

  private void saveRates(HumanResource p, TransformerHandler handler) throws SAXException {
    if (!BigDecimal.ZERO.equals(p.getStandardPayRate())) {
      AttributesImpl attrs = new AttributesImpl();
      addAttribute( name: "name",  value: "standard", attrs);
      addAttribute( name: "value", p.getStandardPayRate().toPlainString(), attrs);
      emptyElement( name: "rate", attrs, handler);
    }
```

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

Superclasse: SaverBase
Subclasses: ResourceSaver, VacationSaver, ViewSaver, HistorySaver, OptionSaver, entre outros.

1.14 Singleton Pattern - Task 4

```
dbarashev
public UIConfiguration getUIConfiguration() {
  if (myUIConfig == null) {
    myUIConfig = new UIConfiguration(new Color( r: 140,  g: 182,  b: 206), redline);
  }
  return myUIConfig;
}
```

java/net/sourceforge/ganttproject/GanttOptions.java

Ensures that only one UIConfig is created for each GanttOption.

```
dbarashev +5
public GanttProject(boolean isOnlyViewer) {
  LoggerApi<Logger> startupLogger = GPLogger.create("Window.Startup");
  startupLogger.debug( msg: "Creating main frame...");
  ToolTipManager.sharedInstance().setInitialDelay(200);
  ToolTipManager.sharedInstance().setDismissDelay(60000);

  getProjectImpl().getHumanResourceManager().addView(this);
  myCalendar.addListener(GanttProject.this::setModified);

  setFocusable(true);
  startupLogger.debug( msg: "1. loading look'n'feels");
  options = new GanttOptions(getRoleManager(), getDocumentManager(), isOnlyViewer);
  myUIConfiguration = options.getUIConfiguration();
  myUIConfiguration.setChartFontOption(getUiFacadeImpl().getChartFontOption());
  myUIConfiguration.setDpiOption(getUiFacadeImpl().getDpiOption());
```

```
public UIConfiguration(Color taskColor, boolean isRedlineOn) {
//    myChartMainFont = chartMainFont == null ? Fonts.DEFAULT_CHART_FONT : chartMainFont;
  this.isRedlineOn = isRedlineOn;
  myResColor = new Color( r: 140,  g: 182,  b: 206);
  myResOverColor = new Color( r: 229,  g: 50,  b: 50);
  myResUnderColor = new Color( r: 50,  g: 229,  b: 50);
  myEarlierPreviousTaskColor = new Color( r: 50,  g: 229,  b: 50);
  myLaterPreviousTaskColor = new Color( r: 229,  g: 50,  b: 50);
  myPreviousTaskColor = Color.LIGHT_GRAY;
  myWeekEndColor = Color.GRAY;
  myDayOffColor = new Color( r: 0.9f,  g: 1f,  b: 0.17f);
  myWeekendAlphaRenderingOption = new AlphaRenderingOption();
   Dmitry Barashev +1
  myAppFontSize = new Supplier<Integer>() {
     Dmitry Barashev +1
    @Override
    public Integer get() {
      //Font tableFont = (Font) UIManager.get("Table.font");
      //return tableFont.getSize() + 8;
      return (int)TreeTableCellsKt.getMinCellHeight().get();
    }
  };
}
```

1.15 Command Pattern - Task 6

```
1 usage   dbarashev
ViewHolder(ViewManagerImpl manager, GanttTabbedPane tabs, GPView view, Icon icon) {
  myManager = manager;
  myTabs = tabs;
  myView = view;
  myIcon = icon;
  GanttLanguage.getInstance().addListener(this);
  assert myView != null;
}
```

java/net/sourceforge/ganttproject/gui/view/ViewManagerImpl.java

ViewHolder has access to a ViewManager that is responsible to create and call classes to perform certain actions.

```
3 usages
private final CopyAction myCopyAction;
3 usages
private final CutAction myCutAction;
2 usages
private final PasteAction myPasteAction;

1 usage   dbarashev +2
public ViewManagerImpl(IGanttProject project, UIFacade uiFacade, GanttTabbedPane tabs, GPUndoManager undoManager) {
  myTabs = tabs;
  project.addProjectEventListener(getProjectEventListener());
  // Create actions
  myCopyAction = new CopyAction( viewManager: this);
  myCutAction = new CutAction( viewManager: this, undoManager);
  myPasteAction = new PasteAction(project, uiFacade,  viewManager: this, undoManager);
```

Base Class (object creator): ViewManager
Created Classes (actions):

- CopyAction
- CutAction
- PasteAction

```
2 usages  dbarashev
public CopyAction(GPViewManager viewManager) {
  super( name: "copy");
  myViewmanager = viewManager;
}
```

```
2 usages  dbarashev
public CopyAction(GPViewManager viewManager) {
  super( name: "copy");
  myViewmanager = viewManager;
}
```

```
2 usages  dbarashev +1
public PasteAction(IGanttProject project, UIFacade uiFacade, GPViewManager viewManager, GPUndoManager undoManager) {
  super( name: "paste");
  myViewmanager = viewManager;
  myUndoManager = undoManager;
  myProject = project;
  myUiFacade = uiFacade;
}
```

1.16 Reviews

Reviewer Name: Rui Correia
Design Pattern: Template Pattern
Should add some more info about the subclasses (maybe some pics or explanation of implementation). Apart from that looks good to me!

Reviewer Name: Tomás Mondim
Design Pattern: Singleton Pattern
Deverias explicar a print 1, passa um pouco despercebido o porque de teres colocado a print. De resto parece me certo!

Reviewer Name: Rui Correia
Design Pattern: Command Pattern
Explain how each command is interpreted and executed.

60173, Guilherme Fernandes

1.17 Façade Pattern - Task 12

```
abstract class GanttProjectBase extends JFrame implements IGanttProject, UIFacade {
  protected final static GanttLanguage language = GanttLanguage.getInstance();
  protected final WeekendCalendarImpl myCalendar = new WeekendCalendarImpl();
  3 usages
  private final ViewManagerImpl myViewManager;
  35 usages
  private final UIFacadeImpl myUIFacade;
  3 usages
  private final GanttStatusBar statusBar;
  1 usage
  private final TimeUnitStack myTimeUnitStack = new GPTimeUnitStack();
  2 usages
  private final ProjectUIFacadeImpl myProjectUIFacade;
  4 usages
  private final DocumentManager myDocumentManager;
  7 usages
  protected final SimpleObjectProperty<Document> myObservableDocument = new SimpleObjectProperty<>();
  /** The tabbed pane with the different parts of the project */
  3 usages
  private final GanttTabbedPane myTabPane;
  3 usages
  private final GPUndoManager myUndoManager;

  3 usages
  private final RssFeedChecker myRssChecker;
  3 usages
  protected final ContentPaneBuilder myContentPaneBuilder;
```

```
protected GanttProjectBase() {
  super( title: "GanttProject");
  var databaseProxy = new LazyProjectDatabaseProxy(SqlProjectDatabaseImpl.Factory::createInMemoryDatabase, this::getTaskManager

  myProjectDatabase = databaseProxy;
  myTaskManagerConfig = new TaskManagerConfigImpl();
  myTaskManager = TaskManager.Access.newInstance( containmentFacadeFactory: null, myTaskManagerConfig,
    myProjectDatabase::createTaskUpdateBuilder);
  myProjectImpl = new GanttProjectImpl((TaskManagerImpl) myTaskManager, databaseProxy);
  addProjectEventListener(databaseProxy.createProjectEventListener());
  myTaskManager.addTaskListener(databaseProxy.createTaskEventListener());
  statusBar = new GanttStatusBar( mainFrame: this);
  myTabPane = new GanttTabbedPane();
  myContentPaneBuilder = new ContentPaneBuilder(getTabs(), getStatusBar());

  NotificationManagerImpl notificationManager = new NotificationManagerImpl(myContentPaneBuilder.getAnimationHost());
  myUIFacade = new UIFacadeImpl( mainFrame: this, statusBar, notificationManager, getProject(),  fallbackDelegate: this);
  myUiInitializationPromise = new TwoPhaseBarrierImpl<>(myUIFacade);

  GPLogger.setUIFacade(myUIFacade);
  var newTaskActor = new NewTaskActor<Task>();
  newTaskActor.start();
```

net/sourceforge/ganttproject/GanttProjectBase.java

Encontrei uma Façade classe (GanttProjectBase) que vai servir de “interface” para a criação dos outros objetos ditos subclasses.

1.18 Singleton Pattern - Task 13

```
9 usages
public class GanttLookAndFeels {

  5 usages
  protected Map<String, GanttLookAndFeelInfo> infoByClass;

  3 usages
  protected Map<String, GanttLookAndFeelInfo> infoByName;

  3 usages
  protected static GanttLookAndFeels singleton;

  static {
    UIManager.put("ClassLoader", LookUtils.class.getClassLoader());
    UIManager.installLookAndFeel( name: "Plastic",  className: "com.jgoodies.looks.plastic.PlasticLookAndFeel");
  }
```

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

net/sourceforge/ganttproject/gui/GanttLookAndFeels.java

Basicamente, este vai ser o único ponto de acesso para criação deste objeto assegurando a criação de apenas um, neste caso sendo usada no UIFacade...

1.19 Decorator Pattern - Task 14

```java
17 usages
public class CalendarActivityImpl implements GPCalendarActivity {

  2 usages
  private final boolean isWorkingTime;

  2 usages
  private final Date myEndDate;

  2 usages
  private final Date myStartDate;

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

```java
1 inheritor
public class AlwaysWorkingTimeCalendarImpl extends GPCalendarBase implements GPCalendarCalc {
  1 override
  @Override
  public List<GPCalendarActivity> getActivities(Date startDate, Date endDate) {
    return Collections.singletonList((GPCalendarActivity) new CalendarActivityImpl(startDate, endDate,  isWorkingTime: true));
  }

  1 usage
  @Override
  protected List<GPCalendarActivity> getActivitiesForward(Date startDate, TimeUnit timeUnit, long unitCount) {
    Date activityStart = timeUnit.adjustLeft(startDate);
    Date activityEnd = activityStart;
    while (unitCount-- > 0) {
      activityEnd = timeUnit.adjustRight(activityEnd);
    }
    return Collections.singletonList((GPCalendarActivity) new CalendarActivityImpl(activityStart, activityEnd,  isWorkingTime: true));
  }
```

```java
1 inheritor
public class WeekendCalendarImpl extends GPCalendarBase implements GPCalendarCalc {

  2 usages
  private static final int DUMMY_YEAR_FOR_RECURRING_EVENTS = 2000;
  10 usages
  private final Calendar myCalendar = CalendarFactory.newCalendar();

  3 usages
  private final FramerImpl myFramer = new FramerImpl(Calendar.DAY_OF_WEEK);

  7 usages
  private final DayType[] myTypes = new DayType[7];
```

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

Encontrei um Decorator, que tem como base o GpcCalendarBase e tem pelo menos as duas decorator classes que estão nos prints, todas com o mesmo tipo e que adicionam coisas diferentes ao mesmo objeto.

- **Component interface:** GPCCalendar
- **Base object:** GPCCalendarBase
- **Decorators:** WeekendCalendarImpl and AlwaysWorkingTimeCalendarImpl

1.20 Reviews

Reviewer Name: Guilherme Fernandes
Design Pattern: Façade Pattern
Should explain the complexity hidden behind the façade.

Reviewer Name: Pedro Grilo
Design Pattern: Singleton Pattern
Está correto, pois garante que o GanttLookAndFeels cria apenas um, usando assim o return do Singleton. Podias mostrar uma print do snippet do code onde é usado.

Reviewer Name: Cláudia Santos
Design Pattern: Decorator Pattern
The identification of the Decorator design pattern seems to be correct, with all the class relationships required by this pattern. There should be some grammar fixes in the report.

60747, Tomás Mondim