



NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Departamento de Informática
Engenharia de Software - 2022/2023
1º Semestre

Projeto – Fase 1

Realizado por

Afonso Ribeiro, 59895

Catarina Crespo, 59810

Danny Fernandes, 61162

Guilherme Figueira, 60288

Luana Gonçalves, 60294

Code Smells

[1] Switch statements & repeated code

Autor: Afonso Ribeiro

Review: Luana Gonçalves

Localização:

biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/WeekendCalendarImpl.java, no método *isPublicHoliday* [linhas 222-241]

Explicação: Os *switch statements* neste método torna difícil fazer mudanças no caso de querermos alterar o valor do *return*. O código repete-se duas vezes.

Solução proposta: Criar classes que implementem uma classe abstrata para cada tipo com um método que retorna o valor que corresponde ao tipo específico, eliminando assim a necessidade do *switch statement* e código repetido.

Snippet do Código:

```
private boolean isPublicHoliday(Date curDayStart) {
    CalendarEvent oneOff = myOneOffEvents.get(curDayStart);
    if (oneOff != null) {
        switch (oneOff.getType()) {
            case HOLIDAY:
                return true;
            case WORKING_DAY:
                return false;
            case NEUTRAL:
            default:
                // intentionally fall-through, consult recurring holidays in this case
        }
    }
    CalendarEvent recurring = myRecurringEvents.get(getRecurringDate(curDayStart));
    if (recurring != null) {
        switch (recurring.getType()) {
            case HOLIDAY:
                return true;
            case WORKING_DAY:
                return false;
            case NEUTRAL:
            default:
                // intentionally fall-through, use default answer
        }
    }
    return false;
}
```

[2] Dead Code

Autor: Afonso Ribeiro

Review: Danny Fernandes

Localização:

ganttproject/src/main/java/net/sourceforge/ganttproject/client/RssFeedChecker.java in method createRssReadCommand lines [188-193]

Explicação: O switch statement neste caso é obsoleto

Solução proposta: substituir switch por um if

Snippet do Código:

```
3 |         try {
4 |             for (int i = 0; i < MAX_ATTEMPTS; i++) {
5 |                 HttpResponse result = httpClient.execute(getRssUrl);
6 |                 switch (result.getStatusLine().getStatusCode()) {
7 |                     case HttpStatus.SC_OK:
8 |                         processResponse(result.getEntity().getContent());
9 |                         return;
10 |                 }
11 |             }
12 |         }
```

7

[3] Code duplication

Autor: Afonso Ribeiro

Review: Guilherme Figueira

Localização:

biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/scene/TimelineSceneBuilder.java in constructor TimelineSceneBuilder in lines [61-66]

Explicação: Número repetido de chamadas a funções

Solução proposta: Pôr os métodos dentro de um loop de maneira a que possamos mudar o número de vezes que chamamos a função caso seja necessário no futuro.

Snippet do Código:

```
1 usage 2 dbarashev +1
public TimelineSceneBuilder(InputApi inputApi) {
    myInputApi = inputApi;
    getCanvas().newLayer();
    getCanvas().newLayer();
    getCanvas().newLayer();
    getCanvas().newLayer();
    myTimelineContainer = getCanvas().newLayer();
```

7

[4] Long Method

Autor: Guilherme Figueira

Review: Afonso Ribeiro

Localização:

ganttproject/src/main/java/net/sourceforge/ganttproject/action/BaselineDialogAction.java

Explicação: O método **actionPerformed()** iniciado na linha 53 tem 94 linhas, sendo assim demasiado grande.

Solução proposta: Os métodos aninhados dentro de **actionPerformed()** deviam estar fora.

Snippet do Código:

```
53 public void actionPerformed(ActionEvent arg0) {
54     myBaselines = new ArrayList<GanttPreviousState>(myProject.getBaselines());
55
56     final EditableList<GanttPreviousState> list = new EditableList<>(myBaselines,
57         Collections.<GanttPreviousState> emptyList()) {
58
59         @Override
60         protected GanttPreviousState updateValue(GanttPreviousState newValue, GanttPreviousState curValue) {
61             curValue.setName(newValue.getName());
62             return curValue;
63         }
64
65         @Override
66         protected GanttPreviousState createValue(GanttPreviousState prototype) {
67             try {
68                 prototype.init();
69                 prototype.saveFile();
70                 return prototype;
71             } catch (IOException e) {
72                 myUiFacade.showErrorDialog(e);
73                 return null;
74             }
75         }
76
77         @Override
78         protected GanttPreviousState createPrototype(Object editValue) {
79             if (editValue == null) {
```

```
118         myUiFacade.getGanttChart().reset();
119     }
120 });
121 list.getTableAndActions().addAction(new GAction("baseline.dialog.hide") {
122     @Override
123     public void actionPerformed(ActionEvent actionEvent) { list.getTableAndActions().setSelection(-1); }
124 });
125
126
127 Action[] actions = new Action[] { (GAction) (e) -> {
128     list.stopEditing();
129     myProject.getBaselines().clear();
130     myProject.getBaselines().addAll(myBaselines);
131     for (GanttPreviousState trashBaseline : myTrash) {
132         trashBaseline.remove();
133     }
134     myProject.setModified();
135 }, CancelAction.EMPTY };
136
137
138 OptionsPageBuilder optionsBuilder = new OptionsPageBuilder();
139 optionsBuilder.setUiFacade(myUiFacade);
140 JPanel contentPanel = new JPanel(new BorderLayout());
141 contentPanel.add(list.createDefaultComponent(), BorderLayout.CENTER);
142 contentPanel.add(optionsBuilder.createGroupComponent(myUiFacade.getGanttChart().getBaselineColorOptions()), BorderLayout.SOUTH);
143 myUiFacade.createDialog(contentPanel, actions, getI18n("baseline.dialog.title")).show();
144 }
```

[5] Dead Code

Autor: Guilherme Figueira

Review: Luana Gonçalves

Localização: *ganttproject/src/main/java/biz.ganttproject/impex.csv/GanttCSVOpen.java*

Explicação: O método **createCustomProperties()** iniciado na linha 110 não é utilizado

Solução proposta: Pôr os métodos dentro de um loop de maneira a que possamos mudar o número de vezes que chamamos a função caso seja necessário no futuro.

Snippet do Código:



```
110 @ protected static void createCustomProperties(Collection<String> customFields, CustomPropertyManager customPropertyManager) {
111     for (String name : customField
112         customPropertyManager.create
113     }
114 }
115 }
```

[6] Data Class

Autor: Guilherme Figueira

Review: Danny Fernandes

Localização:

ganttproject/src/main/java/biz/ganttproject/lib/fx/treetable/CellSkinBase.java

Explicação: Na classe **StyleableProperties** iniciada na linha 154, o tipo da variável final **CELL_SIZE** é definido na própria instância da variável.

Solução Proposta: Se esta classe só é usada pela classe **StyableProperties** então a informação guardada e operações de **CssMetaData** deviam pertencer à classe mãe, caso contrário a classe **CssMetaData** devia ser criada em separado para o caso de esta ser usada novamente noutra parte do código.

Snippet do código:



```
/*
 * Super-lazy instantiation pattern from Bill Pugh.
 */
private static class StyleableProperties {
    private final static CssMetaData<Cell<?>, Number> CELL_SIZE =
        new CssMetaData<~>("-fx-cell-size",
            SizeConverter.getInstance(), DEFAULT_CELL_SIZE) {

        @Override
        public boolean isSettable(Cell<?> n) {
            final CellSkinBase<?> skin = (CellSkinBase<?>) n.getSkin();
            return skin.cellSize == null || !skin.cellSize.isBound();
        }

        @Override
        public StyleableProperty<Number> getStyleableProperty(Cell<?> n) {
            final CellSkinBase<?> skin = (CellSkinBase<?>) n.getSkin();
            return (StyleableProperty<Number>)(WritableValue<Number>)skin.cellSizePropertyImpl();
        }
    }
}
```

[7] Duplicated code

Autor: Catarina Crespo

Review: Afonso Ribeiro

Localização: **createRecurringComponent()** [linhas 138-142] e **createNonRecurringComponent()** [linhas 154-158] em *CalendarEditorPanel.java*

Full path:

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

Explicação: Nestes dois métodos mencionados acima, podemos ver as mesmas cinco linhas de código, sem qualquer diferença de um para outro. Tal situação, no caso de queremos fazer alterações a esta parte do código, obrigar-nos-ia a ter de alterar as mesmas coisas em dois sítios diferentes.

Solução proposta: Criar um método auxiliar que contenha estas cinco linhas de código destacadas [linhas 138-142] e devolva um objeto do tipo *JPanel* (equivalente ao *result*).

Snippet do Código:

```
129
130 @ private Component createRecurringComponent() {
131     DateFormat dateFormat = GanttLanguage.getInstance().getRecurringDateFormat();
132     AbstractTableAndActionsComponent<CalendarEvent> tableAndActions = createTableComponent(myRecurringModel, dateFormat, myRecurringModel);
133     JPanel result = AbstractTableAndActionsComponent.createDefaultTableAndActions(tableAndActions.getTable(), tableAndActions.getTableAndActions());
134
135     Date today = CalendarFactory.newCalendar().getTime();
136     final String hint = GanttLanguage.getInstance().formatText("calendar.editor.dateHint", dateFormat.format(today));
137     Pair<JLabel, ? extends TableCellEditor> validator = createDateValidatorComponents(hint, dateFormat);
138     TableColumn dateColumn = tableAndActions.getTable().getColumnModel().getColumn(TableModelImpl.Column.DATES.ordinal());
139     dateColumn.setCellEditor(validator.second());
140     result.add(validator.first(), BorderLayout.SOUTH);
141     result.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
142     return result;
143 }
```

```
144
145 public JPanel createNonRecurringComponent() {
146     AbstractTableAndActionsComponent<CalendarEvent> tableAndActions = createTableComponent(myOneOffModel, GanttLanguage.getInstance().getRecurringDateFormat(), myOneOffModel);
147     JPanel result = AbstractTableAndActionsComponent.createDefaultTableAndActions(tableAndActions.getTable(), tableAndActions.getTableAndActions());
148
149     Date today = CalendarFactory.newCalendar().getTime();
150     final String hint = GanttLanguage.getInstance().formatText("calendar.editor.dateHint",
151         GanttLanguage.getInstance().getMediumDateFormat().format(today), GanttLanguage.getInstance().getShortDateFormat().format(today));
152
153     Pair<JLabel, ? extends TableCellEditor> validator = createDateValidatorComponents(hint, GanttLanguage.getInstance().getRecurringDateFormat());
154     TableColumn dateColumn = tableAndActions.getTable().getColumnModel().getColumn(TableModelImpl.Column.DATES.ordinal());
155     dateColumn.setCellEditor(validator.second());
156     result.add(validator.first(), BorderLayout.SOUTH);
157     result.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
158     return result;
159 }
```

[8] Dead code

Autor: Catarina Crespo

Review: Afonso Ribeiro

Localização: `getSpanningHeaderFont()`, `getHorizontalGutterColor1()`,
`getHorizontalGutterColor2()`, `getBottomUnitGridColor()`,
`getWorkingTimeBackgroundColor()`, `getPublicHolidayTimeBackgroundColor()`,
`getWeekEndColor()` e `isRedlineOn()` [linhas 142-154, 162, 174 e 178] em
`ChartUIConfiguration.java`

Full path:

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

Explicação: Todos os métodos referidos acima nunca são utilizados

Proposta de Solução: Criar um método auxiliar que contenha estas cinco linhas de código destacadas [linhas 138-142] e devolva um objeto do tipo `JPanel` (equivalente ao `result`).

Snippet do Código:

```
129
130 @ private Component createRecurringComponent() {
131     DateFormat dateFormat = GanttLanguage.getInstance().getRecurringDateFormat();
132     AbstractTableAndActionsComponent<CalendarEvent> tableAndActions = createTableComponent(myRecurringModel, dateFormat,
133     JPanel result = AbstractTableAndActionsComponent.createDefaultTableAndActions(tableAndActions.getTable(), tableAndActions
134
135     Date today = CalendarFactory.newCalendar().getTime();
136     final String hint = GanttLanguage.getInstance().formatText("calendar.editor.dateHint", dateFormat.format(today));
137     Pair<JLabel, ? extends TableCellEditor> validator = createDateValidatorComponents(hint, dateFormat);
138     TableColumn dateColumn = tableAndActions.getTable().getColumnModel().getColumn(TableModelImpl.Column.DATES.ordinal());
139     dateColumn.setCellEditor(validator.second());
140     result.add(validator.first(), BorderLayout.SOUTH);
141     result.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
142     return result;
143 }
```

```
144
145 public JPanel createNonRecurringComponent() {
146     AbstractTableAndActionsComponent<CalendarEvent> tableAndActions = createTableComponent(myOneOffModel, GanttLanguage.ge
147     JPanel result = AbstractTableAndActionsComponent.createDefaultTableAndActions(tableAndActions.getTable(), tableAndActi
148
149     Date today = CalendarFactory.newCalendar().getTime();
150     final String hint = GanttLanguage.getInstance().formatText("calendar.editor.dateHint",
151         GanttLanguage.getInstance().getMediumDateFormat().format(today), GanttLanguage.getInstance().getShortDateFormat().
152
153     Pair<JLabel, ? extends TableCellEditor> validator = createDateValidatorComponents(hint, GanttLanguage.getInstance().get
154     TableColumn dateColumn = tableAndActions.getTable().getColumnModel().getColumn(TableModelImpl.Column.DATES.ordinal());
155     dateColumn.setCellEditor(validator.second());
156     result.add(validator.first(), BorderLayout.SOUTH);
157     result.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
158     return result;
159 }
```

[9] Message Chains

Autor: Catarina Crespo

Review: Guilherme Figueira

Localização: **getChartEndDate()** [linhas 68-70] em ResourceLoadRenderer.java

Full Path:

ganttproject/ganttproject/src/main/java/net.sourceforge.ganttproject/chart/overview/
ResourceLoadRenderer.java

Explicação: Neste método, um objeto do tipo *ChartModel* é chamado, que por sua vez chama um objeto do tipo *OffsetList* que por sua vez chama um método *get()* que retorna um objeto. O parâmetro que este método recebe é novamente o mesmo objeto *ChartModel* a chamar o objeto *OffsetList* (e posteriormente este chama um método *size()*). Por fim, tudo isto é usado para obter um objeto do tipo *Date*. Podemos então ver que estamos claramente na presença de uma série sucessiva de chamadas de métodos que podiam ser simplificadas.

Solução Proposta: Criar um método em *ChartRendererBase.java* (a classe onde é criado método *getChartModel()*) e criar um método que devolva logo a *OffsetList* retornada por *getBottomUnitOffsets()* em relação a este *ChartModel* (chamemos ao método *getChartModelBottomUnitOffsets()*). Criar uma variável do tipo

OffsetList oList = getChartModelBottomUnitOffsets() e uma ***int index = oList.size() - 1***

Para o código ficar mais legível, podia-se criar uma variável ***Offset offset = oList.get(index)*** e usá-la no *return* da função em vez do que temos no código atualmente

return offset.getOffsetEnd()

Snippet do Código:

```
66  @NotNull
67  @Override
68  public Date getChartEndDate() {
69      return getChartModel().getBottomUnitOffsets().get(getChartModel().getBottomUnitOffsets().size() - 1).getOffsetEnd();
70  }
```

[10] Long Method & duplicated code

Autor: Luana Gonçalves

Review: Afonso Ribeiro

Localização:

biz.ganttproject.core/src/main/java/biz/ganttproject/core/time/impl/GPTimeUnitStack.java
a


```

161 public TimeDuration parseDuration(String lengthAsString) throws ParseException {...}
255

```

Explicação: Método demasiado longo, começa na linha 161 e acaba na 254, sendo que tem bastante código repetido com intercalações de **switch** com **if** ficando um código bastante confuso. Como se pode ver no destacado das imagens (linhas 180-196 e linhas 211-227):

```

211 case 2:
212     TimeUnit timeUnit = findTimeUnit(valueBuffer.toString());
213     if (timeUnit == null) {
214         throw new ParseException(lengthAsString, i);
215     }
216     assert currentValue != null;
217     TimeDuration localResult = createLength(timeUnit, currentValue.floatValue());
218     if (currentLength == null) {
219         currentLength = localResult;
220     } else {
221         if (currentLength.getTimeUnit().isConstructedFrom(timeUnit)) {
222             float recalculatedLength = currentLength.getLength(timeUnit);
223             currentLength = createLength(timeUnit, length: localResult.getValue() + recalculatedLength);
224         } else {
225             throw new ParseException(lengthAsString, i);
226         }
227     }
228     state = 0;

```

Solução: Fazer métodos à parte com o código repetido e conseguir assim diminuir o tamanho e complexidade do **parseDuration**.

[11] Dead code

Autor: Luana Gonçalves

Review: Danny Fernandes

Localização:

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

Explicação: A classe **TaskActivityPart** não é usada.

Proposta de Solução: Apagar a classe.

Snippet do código:

```
18 public class TaskActivityPart implements TaskActivity {
19
20     8 usages
    private final Date my
21
22     8 usages
    private final Date my
23
24     5 usages
    private final TimeDur
25
26     10 usages
    private final TaskActivity myOriginal;
27
28     Dmitry Barashev +1
    public TaskActivityPart(TaskActivity original, Date startDate, Date endDate) {
29         myStartDate = Preconditions.checkNotNull(startDate);
30         myEndDate = Preconditions.checkNotNull(endDate);
31         myOriginal = Preconditions.checkNotNull(original);
32         Task task = original.getOwner();
33         myDuration = task.getManager().createLength(task.getDuration().getTimeUnit(), startDate, endDate);
```

[12] No coments

Autor: Luana Gonçalves

Review: Guilherme Figueira

Localização:

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

Explicação: O código não tem comentários, o que faz com que no futuro se tornará complicado perceber os métodos, seja por outra pessoa ou para quem os desenvolveu.

Proposta de Solução: Mal se escreve o código deve se comentar com uma breve explicação do seu objetivo.

Snippet do código:

```
1 usage  Dmitry Barashev +1 *
277 @ private void importYearlyHoliday(ProjectCalendarException e, HolidayAdder adder) {
278     RecurringData recurringData = e.getRecurring();
279     Date date = CalendarFactory.createGanttCalendar( year: 1, month: recurringData.getMonthNumber() - 1,
280     recurringData.getDayNumber()).getTime();
281     adder.addYearlyHoliday(date, Optional.ofNullable(e.getName()));
282 }
283
1 usage  Dmitry Barashev +1
284 @ private void importDailyHoliday(ProjectCalendarException e, HolidayAdder adder) {
285     RecurringData recurringData = e.getRecurring();
286     if (recurringData.getUseEndDate()) {
287         importHolidays(
288             recurringData.getStartDate(), recurringData.getFinishDate(),
289             Optional.ofNullable(e.getName()), adder);
290     } else {
291         importHolidays(
292             recurringData.getStartDate(), recurringData.getOccurrences(),
293             Optional.ofNullable(e.getName()), adder);
294     }
295 }
296
1 usage  Dmitry Barashev
297 private void importHolidays(
298     Date start, int occurrences, Optional<String> title, HolidayAdder adder) {
299     TimeDuration oneDay = getTaskManager().createLength(GregorianCalendar.DAY, length: 1.0f);
300     for (Date dayStart = start; occurrences > 0; occurrences--) {
301         adder.addHoliday(dayStart, title);
302     }
303 }
```

[13] Dead code

Autor: Danny Fernandes

Review: Catarina Crespo

Localização:

ganttproject\ganttproject\src\main\java\biz\ganttproject\lib\fx\treetable/TableColumnHeader.java

linha 345

Explicação: O método tableColumnProperty() nunca é usado.

```
public final ReadOnlyObjectProperty<TableColumnBase<?,?>> tableColumnProperty() {  
    return tableColumn.getReadOnlyProperty();  
}
```

Method 'tableColumnProperty()' is never used

Solução: Apagar o método.

[14] Long Parameter List

Autor: Danny Fernandes

Review: Catarina Crespo

Localização:

ganttproject\ganttproject\src\main\java\biz\ganttproject\lib\fx\treetable/LabeledSkinBase.java

Explicação: O método computePrefHeight() tem uma longa lista de parâmetros (mais de três, neste caso cinco).

Proposta de Solução: Visto que os parâmetros são dimensões, uma solução seria passá-los como um objeto de uma classe criada previamente, representando as dimensões.

Snippet do código:

```
@Override protected double computePrefHeight(double width, double topInset, double rightInset, double bottomInset, double leftInset) {  
    final Labeled labeled = getSkinnable();  
    final Font font = text.getFont();  
    final ContentDisplay contentDisplay = labeled.getContentDisplay();  
    final double gap = labeled.getGraphicTextGap();
```

[15] Long Method

Autor: Danny Fernandes

Review: Catarina Crespo

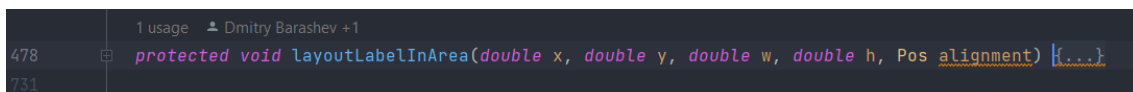
Localização:

ganttproject\ganttproject\src\main\java\biz\ganttproject\lib\fx\treetable\LabeledSkinBase.java [linhas 478-730]

Explicação: O método `layoutLabelInArea()` é muito longo (tem 252 linhas).

Proposta de Solução: Separar algumas instruções em métodos comentados para se poder entender o porquê de cada operação. Operações como ifs (muito usado neste método) podem ser separados do método.

Snippet do código:



The screenshot shows a code editor with a dark theme. At the top, it says '1 usage' and 'Dmitry Barashev +1'. Below that, the code snippet is: `protected void layoutLabelInArea(double x, double y, double w, double h, Pos alignment) {...`. The line numbers 478 and 731 are visible on the left side of the editor.

Design Patterns

[1] Builders

Autor: Afonso Ribeiro

Review: Luana Gonçalves

Localização: OptionsPage Builder in

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

java

e

MenuBuilder in ganttproject/src/main/java/biz/ganttproject/app/Menu.kt

Explicação: Esta classe constrói objetos do tipo JComponent.

Snippet do código:

```
public class OptionsPageBuilder {  
    22 usages  
    I18N myI18n = new I18N();  
    1 usage  
    private Component myParentComponent;  
    2 usages  
    private final LayoutApi myLayoutApi;  
    3 usages  
    private UIFacade myUiFacade;  
    private DecimalFormat myFormat;  
  
    6 usages 2 implementations 1 dbarashev  
    public static interface LayoutApi {  
        2 implementations 1 dbarashev  
        void layout(JPanel panel, int componentsCount);  
    }  
}
```

Explicação: Esta classe constrói objetos do tipo MenuBar.

Snippet do código:

```

class MenuBarBuilder {
    val menuBar = MenuBar()

    Dmitry Barashev
    fun addMenu(title: String, actions: List<GPAAction?>) {
        Menu(title).also { menu ->
            menuBar.menus.add(menu)

            actions.forEach { it: GPAAction?
                val menuItem = it?.let { action ->
                    action.asMenuItem()
                } ?: SeparatorMenuItem()
                menu.items.add(menuItem)
            }
        }
    }

    Dmitry Barashev
    fun build(): MenuBar {
        return menuBar
    }
}

```

1

[2] Singleton

Autor: Afonso Ribeiro

Review: Catarina Crespo

Localização:

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

Explicação: Esta classe é apenas instanciada uma vez com o objetivo de guardar objetos LookAndFeels.

Snippet do código:

```

/**
 * @author Michael Haeusler (michael at akatose.de) This singleton class stores
 * info about the installed LookAndFeels.
 */
9 usages Dmitry Barashev
public class GanttLookAndFeels {

    5 usages
    protected Map<String, GanttLookAndFeelInfo> infoByClass;

    3 usages
    protected Map<String, GanttLookAndFeelInfo> infoByName;

    3 usages
    protected static GanttLookAndFeels singleton;
}

```

1

[3] Decorator

Autor: Afonso Ribeiro

Review: Guilherme Figueira

Localização:

ganttproject/src/main/java/biz/ganttproject/storage/cloud/GPCloudStatusBar.kt in
method updateOnlineMode

Explicação: Um objeto que muda de comportamento durante a execução. Neste caso
pode adicionar decorações ao menu ou removê-las.

Snippet do código:

```
private fun updateOnlineMode(mode: OnlineDocumentMode) {
    when (mode) {
        OnlineDocumentMode.ONLINE_ONLY -> {
            this.btnOffline.run { this: Button {
                text = STATUS_BAR_LOCALIZER.formatText( key: "mode.onlineOnly")
                graphic = FontAwesomeIconView(FontAwesomeIcon.CLOUD)
                tooltip = Tooltip(STATUS_BAR_LOCALIZER.formatText( key: "mode.onlineOnly.tooltip"))
                Decorator.removeAllDecorations( target: this)
                isDisable = false
            }
            this.btnLock.isDisable = false
        }
        OnlineDocumentMode.MIRROR -> {
            this.btnOffline.run { this: Button {
                text = STATUS_BAR_LOCALIZER.formatText( key: "mode.mirror")
                graphic = FontAwesomeIconView(FontAwesomeIcon.CLOUD_DOWNLOAD)
                tooltip = Tooltip(STATUS_BAR_LOCALIZER.formatText( key: "mode.mirror.tooltip"))
                Decorator.removeAllDecorations( target: this)
                isDisable = false
            }
            this.btnLock.isDisable = false
        }
        OnlineDocumentMode.OFFLINE_ONLY -> {
            this.btnOffline.run { this: Button {
                text = STATUS_BAR_LOCALIZER.formatText( key: "mode.offline")
                graphic = FontAwesomeIconView(FontAwesomeIcon.CLOUD_DOWNLOAD)
                tooltip = Tooltip(STATUS_BAR_LOCALIZER.formatText( key: "mode.offline.tooltip"))
                Decorator.addDecoration( target: this, GraphicDecoration(createWarningDecoration(), Pos.BOTTOM_LEFT, xOffset: 6.0, yOffset: -4.0))
                isDisable = true
            }
        }
    }
}
```

[4] Adapter Pattern

Autor: Catarina Crespo

Review: Afonso Ribeiro

Localização:

ganttproject/ganttproject/src/main/java/biz/ganttproject/storage/webdav/WebdavBrowse
rPane.kt

Explicação: Tal como é dito em comentário no código, esta classe serve como
adaptador entre **WebDavResource** e **FolderView**, sendo esta última uma classe do
sistema.

Snippet do código:

```
182  */
183  class WebDavResourceAsFolderItem(val myResource: WebDavResource) : FolderItem {
184      override val isLocked: Boolean
185      get() {...}
193
194      override val isLockable: Boolean
195      get() = myResource.isLockSupported(exclusive: true)
196
197      override val canChangeLock: Boolean
198      get() = isLockable
199
200      override val name: String
201      get() = myResource.name
202
203      override val basePath: String
204      get() = myResource.parent.absolutePath
205
206      override val isDirectory: Boolean
207      get() {...}
215      override val tags = listOf<String>()
216  }
```

[5] Abstract Factory Pattern

Autor: Catarina Crespo

Review: Afonso Ribeiro

Localização: interface **WizardPage** em WizardPage.java

Full Path:

ganttproject/ganttproject/src/main/java/net.sourceforge.ganttproject/gui/projectwizard/
WizardPage.java

Explicação: A interface WizardPage serve como uma abstração que é depois usada por várias classes distintas, mas relacionadas entre si, sendo estas: ProjectNamePage(), CalendarEditorPage(), FileChooserPageBase(), entre outras. Isto permite-nos produzir objetos semelhantes sem necessidade de especificar a sua classe, tal como é feito, por exemplo, na classe NewProjectWizardWindow.java.

Snippet do código:

```
1  .../
19  package net.sourceforge.ganttproject.gui.projectwizard;
20
21  import java.awt.Component;
22
23  public interface WizardPage {
24      /** @return the title of the page */
25      String getTitle();
26
27      /** @return the Component that makes the page */
28      Component getComponent();
29
30      void setActive(boolean b);
31  }
32
```

[6] Memento Pattern

Autor: Catarina Crespo

Review: Guilherme Figueira

Localização: classe **GanttPreviousState.java**

Full Path: *ganttproject/ganttproject/src/main/java/net.sourceforge.ganttproject/GanttPreviousState.java*

Explicação: A classe GanttPreviousState.java guarda o estado anterior para que este possa ser restaurado a qualquer altura.

Snippet do código:



```
1  .../
19 package net.sourceforge.ganttproject;
20
21 import ...
22
23 /**
24  * @author nbohn
25  */
26 public class GanttPreviousState {
27     private final List<GanttPreviousStateTask> myTasks;
28
29     private String myName;
30
31     private File myFile;
32
33     public GanttPreviousState(String name, List<GanttPreviousStateTask> tasks) {
34         myName = name;
35         myTasks = tasks;
36     }
37 }
```

[7] Iterator

Autor: Guilherme Figueira

Review: Danny Fernandes

Localização:

ganttproject/src/main/java/net/sourceforge/ganttproject/task/algorithm/ProjectBoundsAlgorithm.java

Explicação: Neste método, é utilizado um iterador da classe *Task* para iterar os seus elementos sem expor a representação da estrutura de dados da classe.

Snippet do Código:

```

public Result getBounds(Collection/* <Task> */<Task> tasks) {
    Date lowerBound = null;
    Date upperBound = null;
    for (Iterator<Task> it = tasks.iterator(); it.hasNext();) {
        Task next = it.next();
        Date start = next.getStart().getTime();
        Date end = next.getEnd().getTime();
        if (lowerBound == null || lowerBound.after(start)) {
            lowerBound = start;
        }
        if (upperBound == null || upperBound.before(end)) {
            upperBound = end;
        }
    }
    return new Result(lowerBound, upperBound);
}

```

[8] Facade

Autor: Guilherme Figueira

Review: Catarina Crespo

Localização: *ganttproject/src/main/java/biz/ganttproject/impex/csv/GanttCSVOpen.java*

Explicação: Esta classe facilita a leitura de ficheiros CSV e XLS a outras classes utilizando bibliotecas da *Google* e *Apache*. Deste modo, outras classes que precisem de ler ficheiros CSV e XLS têm uma interface simples de o fazer sem ter de estar constantemente a utilizar bibliotecas de terceiros.

Snippet do Código:

```

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

    static final GanttLanguage language = GanttLanguage.getInstance();

    private final Supplier<InputStream> myInputSupplier;

    private final SpreadsheetFormat myFormat;

    private final List<RecordGroup> myRecordGroups;

    private int mySkippedLine;

    private CSVOptions myCsvOptions;

    public GanttCSVOpen(Supplier<InputStream> inputSupplier, SpreadsheetFormat format, RecordGroup... groups) {
        myInputSupplier = inputSupplier;
        myRecordGroups = Lists.newArrayList();
    }
}

```

[9] Observer

Autor: Guilherme Figueira

Review: Luana Gonçalves

Localização:

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

Explicação: Esta classe, como o nome indica, espera pela alteração de um valor e reage assim que tal acontece. Neste caso, este pattern facilita a implementação de uma *progress bar* que ajuda o utilizador a visualizar a execução de outra parte do código.

Snippet do Código:

```

public Component createWaitIndicatorComponent(DefaultBooleanOption controller) {
    final JProgressBar progressBar = new JProgressBar();
    JPanel placeholder = new JPanel();
    final JPanel result = new JPanel(new CardLayout());
    result.add(placeholder, "placeholder");
    result.add(progressBar, "progressBar");
    controller.addChangeListener(new ChangeValueListener() {
        @Override
        public void changeValue(ChangeValueEvent event) {
            if (Boolean.TRUE.equals(event.getNewValue())) {
                progressBar.setIndeterminate(true);
                ((CardLayout) result.getLayout()).show(result, "progressBar");
            } else {
                progressBar.setIndeterminate(false);
                ((CardLayout) result.getLayout()).show(result, "placeholder");
            }
        }
    });
    return result;
}

```

[10] Factory Method

Autor: Luana Gonçalves

Review: Danny Fernandes

Localização: jdk-15!\jdk.aot\jdk\tools\jaotc\bin\format\Container.class

Explicação: A interface **Container** é implementada pela superclasse **ByteContainer**, que por sua vez é estendida pelas classes **ReadOnlyDataContainer** e **CodeContainer**. Assim permite que nas subclasses alterem o tipo de objetos que serão criados.

Snippet do Código:

```
public class ByteContainer implements Container {
    private ByteBuffer contentBytes;
    private ByteArrayOutputStream contentStream;
    private boolean bufferModified;
    private boolean hasRelocations;
    private String containerName;
    private final SymbolTable symbolTable;
    private int sectionId = -1;

    public ByteContainer(String containerName, SymbolTable symbolTable) {
        this.containerName = containerName;
        this.symbolTable = symbolTable;
        this.contentBytes = null;
        this.bufferModified = false;
        this.hasRelocations = false;
        this.contentStream = new ByteArrayOutputStream();
    }
}
```

```
public final class ReadOnlyDataContainer extends ByteContainer {
```

```
public final class CodeContainer extends ByteContainer {
```

[11] Iterator

Autor: Luana Gonçalves

Review: Catarina Crespo

Localização: com/sun/org/apache/bcel/internal/util/InstructionFinder.java

Explicação: Este padrão de comportamento permite percorrer elementos de uma coleção, neste caso do tipo *InstructionHandle*, sem expor a sua representação subjacente.

Snippet do Código:

```
208 @      public final Iterator<InstructionHandle[]> search( final String pattern,
209      final InstructionHandle from, final CodeConstraint constraint ) {
210      final String search = compilePattern(pattern);
211      int start = -1;
212      for (int i = 0; i < handles.length; i++) {
213          if (handles[i] == from) {
214              start = i; // Where to start search from (index)
215              break;
216          }
217      }
218      if (start == -1) {
219          throw new ClassGenException("Instruction handle " + from
220              + " not found in instruction list.");
221      }
222      final Pattern regex = Pattern.compile(search);
223      final List<InstructionHandle[]> matches = new ArrayList<>();
224      final Matcher matcher = regex.matcher(il_string);
225      while (start < il_string.length() && matcher.find(start)) {
226          final int startExpr = matcher.start();
227          final int endExpr = matcher.end();
228          final int lenExpr = endExpr - startExpr;
229          final InstructionHandle[] match = getMatch(startExpr, lenExpr);
230          if ((constraint == null) || constraint.checkCode(match)) {
231              matches.add(match);
232          }
233          start = endExpr;
234      }
235      return matches.iterator();
236  }
```

[12] Adapter

Autor: Luana Gonçalves

Review: Danny Fernandes

Localização: jdk/jfr/internal/instrument/JIMethodMergeAdapter.java

Explicação: Esta classe serve como adaptador de métodos da **ClassNode** para a **ClassVisitor** como descrito nos comentários acima.

Snippet do Código:

```
66 @ public JIMethodMergeAdapter(ClassVisitor cv, ClassNode cn, List<Method> methodFilter, JITypeMapping[] typeMappings) {  
67     super(Opcodes.ASM7, cv);  
68     this.cn = cn;  
69     this.methodFilter = methodFilter;  
70  
71     this.typeMap = new HashMap<>();  
72     for (JITypeMapping tm : typeMappings) {  
73         typeMap.put(tm.from().replace( oldChar: '.', newChar: '/' ), tm.to().replace( oldChar: '.', newChar: '/' ));  
74     }  
75 }
```

[13] Iterator

Autor: Danny Fernandes

Review: Afonso Ribeiro

Localização:

ganttproject\ganttproject\src\main\java\net\sourceforge\ganttproject\task\dependency\TaskDependencyCollectionImpl.java

Explicação: Percorre os elementos que são do tipo Task sem expor as suas representações subjacentes.

Snippet do Código:

```
for (Iterator<Task> tasks = tasksInvolved.iterator(); tasks.hasNext();) {  
    Task nextInvolved = tasks.next();  
    if (false == getTaskHierarchy().areUnrelated(nextInvolved, dependant)) {  
        return true;  
    }  
}
```

[14] Adapter

Autor: Danny Fernandes

Review: Luana Gonçalves

Localização:

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

Explicação: Converte a interface de um objeto Chart para que outro objeto Component possa entendê-lo.

Snippet do Código:

```
public Object getAdapter(Class adapter) {  
    if (Component.class.isAssignableFrom(adapter)) {  
        return this;  
    }  
    return null;  
}
```

```
GPViewImpl(Chart chart) {  
    myChart = chart;  
    myComponent = (Component) chart.getAdapter(Container.class);  
}
```

[15] Iterator

Autor: Danny Fernandes

Review: Guilherme Figueira

Localização:

ganttproject\ganttproject\src\main\java\net\sourceforge\ganttproject\task\algorithm\CriticalPathAlgorithmImpl.java

Explicação: Percorre os elementos que são do tipo Node sem expor as suas representações subjacentes.

Snippet do Código:

Package: Searcher

```
private LinkedList<Node> processQueue() {  
    LinkedList<Node> newQueue = new LinkedList<>();  
    for (Iterator<Node> nodes = myQueue.iterator(); nodes.hasNext();) {  
        Node curNode = nodes.next();  
        if (curNode.lft == null || curNode.lftFromSupertask) {  
            calculateLatestDates(curNode);  
        }  
    }  
}
```