

FCT UNL

Code Smells and Code Patterns

Trabalho de ES

Júlio Simões 44599
André Ribeiro 59835
Daniel Gavinho 59889
Joana Fernandes 60065
Filipe Leão 60191
21-10-2022

Code Smells

Code Smells - Júlio Simões - 44599

1. Comments

- **Code snippet:**

```
private boolean export(Exporter exporter, Args args, IGanttProject
project, UIFacade uiFacade) {
    logger.debug("Using exporter {}", new Object[]{exporter}, new
HashMap<>());
    ConsoleUIFacade consoleUI = new ConsoleUIFacade(uiFacade);
    GPLogger.setUIFacade(consoleUI);
    // TODO: bring back task expanding
    // if (myArgs.expandTasks) {
    //     for (Task t : project.getTaskManager().getTasks()) {
    //         project.getUIFacade().getTaskTree().setExpanded(t, true);
    //     }
    // }
```

- **Location:**

ganttproject/src/main/java/net/sourceforge/ganttproject/export/CommandLineExp
ortApplication.java

- **Reasoning:** This is a reminder that something should be done (TODO). In the snippet we can also see that we have commented code. These comments reduce readability and understandability.

- **Refactoring suggestion:** The proposal would be to first; remove the commented code and second remove the TODO comment or implement the functionality that is missing.

```
private boolean export(Exporter exporter, Args args, IGanttProject
project, UIFacade uiFacade) {
    logger.debug("Using exporter {}", new Object[]{exporter}, new
HashMap<>());
    ConsoleUIFacade consoleUI = new ConsoleUIFacade(uiFacade);
    GPLogger.setUIFacade(consoleUI);
```

Sidenote: While investigating the code I noticed that most of the methods on classes/interfaces are not commented. Which makes it very difficult for whoever is maintaining the code or reading it to try to understand it. The lack of comments throughout this project it's a major code smell.

Review – Daniel Gavinho – 59889

Foi bem apontado que a existência destes TODOs é um code smell do tipo Comments. Também muito bem visto que várias classes sofriam de uma falta de comentários.

2. Duplicated Code

- **Code snippet:**

```
@Override
public void startCopyClipboardTransaction() {
    if (isTransactionRunning) {
        cancelClipboardTransaction();
    }
    isTransactionRunning = true;
}

@Override
public void startMoveClipboardTransaction() {
    if (isTransactionRunning) {
        cancelClipboardTransaction();
    }
    isTransactionRunning = true;
}
```

- **Location:**

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

- **Reasoning:** In this example we have a duplicated method. Both methods do the same thing only with a different name. It's confusing and makes maintaining the code more difficult.
- **Refactoring suggestion:** Remove one of them and refactor the other methods that call the deleted one.

Review – André Ribeiro - 59835

O code smell foi bem encontrado, e como o colega disse (e bem), o código torna-se muito mais confuso com a inclusão de um dos métodos equivalentes. A solução embora ligeiramente generalizada é a melhor a se tomar.

3. Dead Code

- **Code snippet:**

```
private URL getSelectedUrl() {
    try {
        switch (ourSelectedSource) {
            case FILE_SOURCE:
                return myChooser.getSelectedURL();
            case URL_SOURCE:
                return new URL(myUrlField.getText());
            default:
                assert false : "Should not be here";
                return null;
        }
    } catch (MalformedURLException e) {
        reportMalformedUrl(e);
        return null;
    }
}

private void reportMalformedUrl(Exception e) {
}
```

- **Location:**

ganttproject/src/main/java/net/sourceforge/ganttproject/wizard/AbstractFileChooserPage.java

- **Reasoning:** In this example we have a case of dead code. The getSelectUrl() method calls the reportMalformedUrl(Exception e). We shouldn't have a parameter that's not used (Exception e), it's misleading. Without the parameter the behavior would be the same.

Also the getSelectUrl() is calling a method that does nothing. Its confusing for whoever is maintaining the code.

- **Refactoring suggestion:** Remove the reportMalformedUrl() altogether or implement the method.

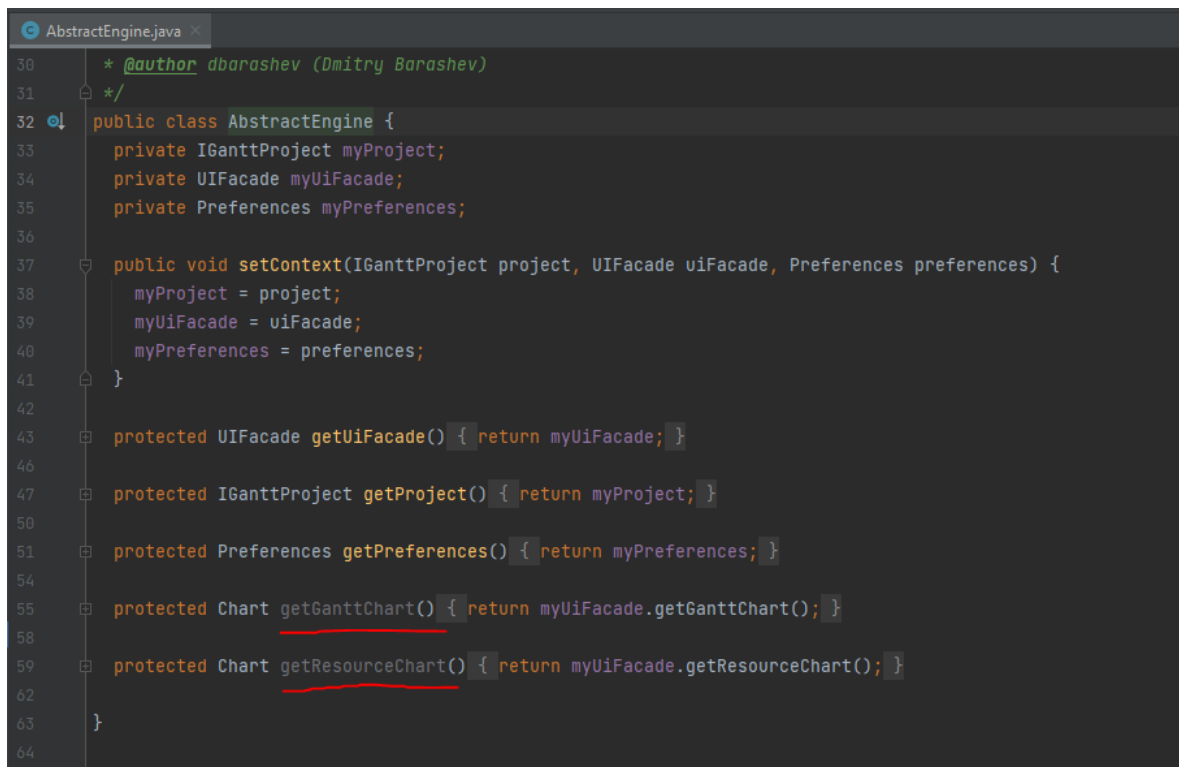
```
private URL getSelectedUrl() {
    try {
        switch (ourSelectedSource) {
            case FILE_SOURCE:
                return myChooser.getSelectedURL();
            case URL_SOURCE:
                return new URL(myUrlField.getText());
            default:
                assert false : "Should not be here";
                return null;
        }
    } catch (MalformedURLException e) {
        return null;
    }
}
```

Review – Joana Fernandes – 60065

Efetivamente, o método `reportMalformedURLException` é inútil, e portanto a melhor solução seria eliminar este método por completo.

Code Smells – André Ribeiro - 59835

Code Smell 1: Speculative Generality



```
30  * @author dbarashev (Dmitry Barashev)
31  */
32  public class AbstractEngine {
33      private IGanttProject myProject;
34      private UIFacade myUiFacade;
35      private Preferences myPreferences;
36
37      public void setContext(IGanttProject project, UIFacade uiFacade, Preferences preferences) {
38          myProject = project;
39          myUiFacade = uiFacade;
40          myPreferences = preferences;
41      }
42
43      protected UIFacade getUiFacade() { return myUiFacade; }
44
45
46
47      protected IGanttProject getProject() { return myProject; }
48
49
50
51      protected Preferences getPreferences() { return myPreferences; }
52
53
54
55      protected Chart getGanttChart() { return myUiFacade.getGanttChart(); }
56
57
58      protected Chart getResourceChart() { return myUiFacade.getResourceChart(); }
59
60
61
62  }
```

Found in:

[org.ganttproject.impex.htmlpdf\src\main\java\org\ganttproject\impex\htmlpdf\AbstractEngine.java](#)

Lines: 55 and 59

A Speculative Generality code smell is described as code that might be useful someday but is not needed at the moment, and this generality may hinder the code by “over-engineering” it.

In the two mentioned lines we can see that both `getGanttChart()` and `getResourceChart()` are two methods that may prove useful, yet are not used right now, therefore these fall in the Speculative Generality smell.

One solution to amend this code would be remove both these methods entirely.

Review – Júlio Simões – 44599

--Speculative Generality--

Estas funções nunca são chamadas.

Depende da intenção do developer na altura da criação dos métodos.

Caso tenham sido criados com a intenção de serem utilizados um dia, então podemos considerar Speculative Generality. Caso contrário é código que terá sido utilizado em tempos, e a dada altura deixou de ser preciso, e nesse caso estaríamos perante Dead Code.

Code Smell 2: Dead Code



```
33 public class VisibleNodesFilter {
34 @ public List<Task> getVisibleNodes(JXTreeTable jtree, int minHeight, int maxHeight, int nodeHeight) {
35     List<MutableTreeTableNode> preorderedNodes = TreeUtil.collectSubtree((MutableTreeTableNode) jtree.getTreeTableModel().getRoot());
36     List<Task> result = new ArrayList<>();
37     int currentHeight = 0;
38     for (int i = 1; i < preorderedNodes.size(); i++) {
39         MutableTreeTableNode nextNode = preorderedNodes.get(i);
40         if (false == nextNode.getUserObject() instanceof Task) {
41             continue;
42         }
43         if ((currentHeight + nodeHeight) > minHeight && jtree.isVisible(TreeUtil.createPath(nextNode))) {
44             result.add((Task) nextNode.getUserObject());
45         }
46         if (jtree.isVisible(TreeUtil.createPath(nextNode))) {
47             currentHeight += nodeHeight;
48         }
49         if (currentHeight > minHeight + maxHeight) {
50             break;
51         }
52     }
53     return result;
54 }
55 }
56 }
```

Found in:

[ganttproject\src\main\java\net\sourceforge\ganttproject\chart\VisibleNodesFilter.java](#)

Code that has no further use and becomes obsolete is considered Dead Code.

The VisibleNodesFilter.java class is not used anywhere in the rest of the code, so it can be considered Dead Code, therefore the solution to take is to delete the class entirely.

Review – Daniel Gavinho – 59889

Como demonstrado na imagem e comentário feito pelo André este código não tem nenhuma utilidade dentro do programa, por isso é o que considerariamos Dead Code.

Code Smell 3: Data Class

```
1  //...
19 package net.sourceforge.ganttproject.client;
20
21 public class RssUpdate {
22
23     final String myVersion;
24     final String myUrl;
25     final String myDescription;
26
27     public RssUpdate(String version, String url, String description) {
28         myVersion = version;
29         myUrl = url;
30         myDescription = description;
31     }
32
33     public String getVersion() { return myVersion; }
34
35     public String getUrl() { return myUrl; }
36
37     public String getDescription() { return myDescription; }
38 }
39
40
41
42
43
44
45
```

Found in:

[ganttproject\src\main\java\net\sourceforge\ganttproject\client\RssUpdate.java](#)

A Data Classes are classes with the mere function of storing data that is used in other classes, characterized by containing only instance variables and simple methods to access said variables(getter/setters).

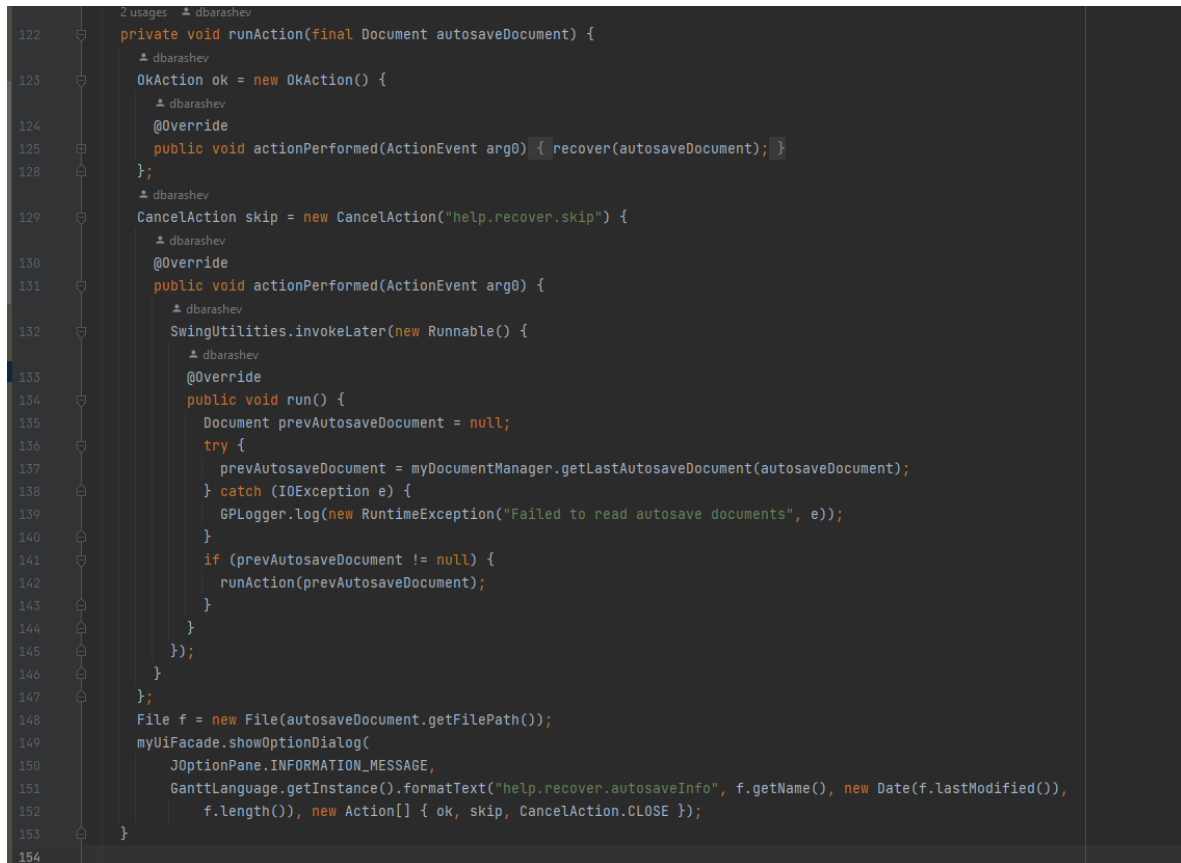
RssUpdate.java is one of these Data Classes, since its composition is 3 instance variables, its constructor and 3 getters. The best way to refactor this code would be to move the methods in this class to another class that could find a better use of said methods.

Review – Joana Fernandes – 60065

Como foi apontado, a classe tem apenas métodos getter. Portanto, é efetivamente uma Data Class.

Code Smells – Daniel Gavinho - 59889

Long Methods



```
122 private void runAction(final Document autosaveDocument) {
123     OkAction ok = new OkAction() {
124         @Override
125         public void actionPerformed(ActionEvent arg0) { recover(autosaveDocument); }
126     };
127
128     CancelAction skip = new CancelAction("help.recover.skip") {
129         @Override
130         public void actionPerformed(ActionEvent arg0) {
131             SwingUtilities.invokeLater(new Runnable() {
132                 @Override
133                 public void run() {
134                     Document prevAutosaveDocument = null;
135                     try {
136                         prevAutosaveDocument = myDocumentManager.getLastAutosaveDocument(autosaveDocument);
137                     } catch (IOException e) {
138                         GPLogger.log(new RuntimeException("Failed to read autosave documents", e));
139                     }
140                     if (prevAutosaveDocument != null) {
141                         runAction(prevAutosaveDocument);
142                     }
143                 }
144             });
145         }
146     };
147
148     File f = new File(autosaveDocument.getFilePath());
149     myUiFacade.showOptionDialog(
150         JOptionPane.INFORMATION_MESSAGE,
151         GanttLanguage.getInstance().formatText("help.recover.autosaveInfo", f.getName(), new Date(f.lastModified()),
152             f.length()), new Action[] { ok, skip, CancelAction.CLOSE });
153 }
154
```

Folder Encontrada:

ganttproject\src\main\java\net\sourceforge\ganttproject\action\help\HelpMenu.java

Rationale: O programa contém vários métodos como o mostrado acima, onde têm tantos argumentos que ocupam várias linhas para correr um só método eg. Linhas 149-152

Suggestion: Este code smell pode ser resolvido pela criação prévia anteriormente dos argumentos, ou utilização de uma classe para tratar da criação e manutenção desses argumentos numa maneira mais concisa

Review – Filipe Leão – 60191

--Long Method--

O método apontado tem de facto vários argumentos que tornam o código pouco legível.

Comments

```
247 * A {@link ConvolveOp} using a very light "blur" kernel that acts like an
248 * anti-aliasing filter (softens the image a bit) when applied to an image.
249 * <p/>
250 * A common request by users of the library was that they wished to "soften"
251 * resulting images when scaling them down drastically. After quite a bit of
252 * A/B testing, the kernel used by this Op was selected as the closest match
253 * for the target which was the softer results from the deprecated
254 * {@link AreaAveragingScaleFilter} (which is used internally by the
255 * deprecated {@link Image#getScaledInstance(int, int, int)} method in the
256 * JDK that imgscalr is meant to replace).
257 * <p/>
258 * This ConvolveOp uses a 3x3 kernel with the values:
259 * <table cellpadding="4" border="1">
260 * <tr>
261 * <td>.0f</td>
262 * <td>.08f</td>
263 * <td>.0f</td>
264 * </tr>
265 * <tr>
266 * <td>.08f</td>
267 * <td>.68f</td>
268 * <td>.08f</td>
269 * </tr>
270 * <tr>
271 * <td>.0f</td>
272 * <td>.08f</td>
273 * <td>.0f</td>
```

Folder Encontrada: ganttproject\src\main\java\org\imgscalr\Scalr.java

Rationale: Quase todos, senão todos, os comandos desta classe têm comentários que ocupam muito mais espaço do que os comandos em si

Suggestion: Diminuir a explicação do trabalho de cada comando, ou ainda separar o trabalho de cada comando por vários que realizam as várias tarefas enunciadas nos comentários

Review – André Ribeiro – 59835

Um code smell bem apontado, visto que estes comentários extensos efetivamente dificultam a leitura do código. Concordo também com a solução proposta.

Dead Code

```
1 package org.ganttproject;  
2  
3 public class WebStartIDClass {  
4  
5 }  
6
```

Folder Encontrada: ganttproject\src\main\java\org\gantProject\WebStartIDClass.java

Rationale: Esta classe não é utilizada em nenhum local, e como visto na imagem está completamente vazia

Suggestion: Apagar a classe

Review – Júlio Simões – 59835

--Dead Code--

Concordo com o comentário do Daniel. Uma classe sem conteúdo e/ou que não é utilizada deve ser removida.

Caso tenha sido colocada como placeholder para futuros desenvolvimentos deverá ser construída.

Caso contrário deve ser eliminada

Data Class

- Code snippet:

```
public enum CustomPropertyClass {
    TEXT( i18ntifier: "text",   defaultValue: "", String.class),
    INTEGER( i18ntifier: "integer",   defaultValue: "0", Integer.class),
    DOUBLE( i18ntifier: "double",   defaultValue: "0.0", Double.class),
    DATE( i18ntifier: "date",   defaultValue: null, GregorianCalendar.class),
    BOOLEAN( i18ntifier: "boolean",   defaultValue: "false", Boolean.class);

    private final String myI18Ntifier;
    private final Class myJavaClass;
    private final String myDefaultValue;

    private CustomPropertyClass(String i18ntifier, String defaultValue, Class<?> javaClass) {
        myI18Ntifier = i18ntifier;
        myDefaultValue = defaultValue;
        myJavaClass = javaClass;
    }

    public String getDisplayName() { return GanttLanguage.getInstance().getText(myI18Ntifier); }

    public Class<?> getJavaClass() { return myJavaClass; }

    @Override
    public String toString() { return getDisplayName(); }

    public String getID() { return myI18Ntifier; }

    public String getDefaultValueAsString() { return null; }

    public boolean isNumeric() { return this == INTEGER || this == DOUBLE; }

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

- Localização:

ganttproject\Project\ProjectFiles\ganttproject\src\main\java\biz\ganttproject\custompr
operty\CustomPropertyClass.java

- Rationale: a classe contém quase exclusivamente getter methods.

- Sugestão de refatorização: possivelmente existem métodos na classe que chamam estes getters que estariam melhor localizados nesta classe. Por outro lado, estes getters poderiam ser distribuídos pelas classes que os chamam, eliminando a necessidade desta classe existir.

Review – Júlio Simões – 59835

Como apontado pelo elemento 4 (Joana) esta class é constituída basicamente só de getters, a sugestão que ela dá para a resolução deste smell, parece-me ser a mais apropriada

Message Chain

- **Code snippet:**

```
174         case OUTLINE_NUMBER:  
175             List<Integer> outlinePath = task.getManager().getTaskHierarchy().getOutlinePath(task);  
176             writer.print(Joiner.on('.').join(outlinePath));  
177             break;
```

- **Localização:**

ganttproject\Project\ProjectFiles\ganttproject\src\main\java\biz\ganttproject\impex\csv\GanttCSVExport.java

- **Rationale:** é chamado um método, que chama outro método, que chama um terceiro, criando dependência excessiva entre classes. Qualquer mudança na relação entre as classes irá requerer alteração deste código.

- **Sugestão de refatorização:** relocar o método de modo a não ser necessário percorrer várias classes para o utilizar.

Review – Júlio Simões - 44599

--Message Chain--

A logica parece correta. Não há necessidade de precorrer 3 objectos para obter um. Realocar a funcionalidade final para o início da cadeia, seria uma eventual estratégia a implementar nesta situação.

Dead Code

- Code snippet:

```
public void flush() throws BackingStoreException {  
}
```

- **Localização:** ganttproject\Project\Project
Files\ganttproject\src\main\java\net\sourceforge\ganttproject\PluginPreferencesImpl.
java
- **Rationale:** o método está vazio, ou seja, mesmo sendo chamado, não faz nada.
- **Sugestão de refatorização:** eliminar o método.

Review – André Ribeiro – 59835

O código apresentado evidentemente não tem utilidade para o resto do código, e por isso concordo com a sugestão de refatorização proposta.

Code Smells – Filipe Leão – 60191

1. Speculative Generality

- Code Snippet

```
57  @Override
58  public void process(XmlProject xmlProject) {
59      xmlProject.getAllocations().forEach(this::loadAllocation);
60      processRoleBindings();
61  }
```

- Path

ganttproject\src\main\java\net\sourceforge\ganttproject\parser\AllocationTagHandler.java

- Rationale

The method in question is implemented and never called upon to which leaves this feature unused.

- Refactoring Suggestion

Add features which require this method elsewhere or delete the method.

Review – Joana Fernandes – 60065

Code smell bem apontado, efetivamente a melhor opção é simplesmente eliminar o método, já que este não é usado de todo.

2. Comments

- Code Snippet

```
@Override
public RoleSet createRoleSet(String name) {
    RoleSet result = new RoleSetImpl(name, this);
    myRoleSets.add(result);
    // System.err.println("[RoleManagerImpl] createRoleSet():
    // created:"+name);
    return result;
}
```

- Path

ganttproject\src\main\java\net\sourceforge\ganttproject\roles\RoleManagerImpl.java

- Rationale

The commented code serves no purpose. It is probably remains of debugging.

- Refactoring Suggestion

Elimination of useless comments.

Review – André Ribeiro - 59835

Apesar de parecer irrelevante, código desnecessário prejudica a leitura do código tornando-o mais confuso. Sendo assim concordo com a sugestão de refatorização proposta

3. Middleman

- Code Snippet

```
25  class ColorValueParser {
26      public static Color parseString(String value) {
27          return ColorConversion.determineColor(value);
28      }
29  }
30
```

- Path

ganttproject\src\main\java\net\sourceforge\ganttproject\parser\ColorValueParser.java

- Rationale

The purpose of this class is questionable since it doesn't take advantage of the oop Java offers and it is only used to implement one method.

- Refactoring Suggestion

Take this method to where it is called and implement it there.

Review – Daniel Gavinho – 59889

O element 5 (Filipe Leão) aponta bem como este comando serve simplesmente de middleman

Code Patterns

Code Patterns – Júlio Simões – 44599

Template method pattern

- **Code snippet:**

GPCalendarBase:

```
abstract class GPCalendarBase implements GPCalendarCalc {  
...  
...  
...  
protected abstract List<GPCalendarActivity> getActivitiesBackward(Date  
startDate, TimeUnit timeUnit, long unitCount);  
  
protected abstract List<GPCalendarActivity> getActivitiesForward(Date  
startDate, TimeUnit timeUnit, long unitCount);  
  
public List<GPCalendarActivity> getActivities(Date startingFrom,  
TimeDuration period) {  
    return getActivities(startingFrom, period.getTimeUnit(),  
period.getLength());  
}  
...  
...  
...  
}
```

WeekendCalenderImpl:

```

public class WeekendCalendarImpl extends GPCalendarBase implements
GPCalendarCalc {
...
...
...
@Override
protected List<GPCalendarActivity> getActivitiesBackward(Date
startDate, TimeUnit timeUnit, long unitCount) {
...
...
...
}
@Override
protected List<GPCalendarActivity> getActivitiesBackward(Date
startDate, TimeUnit timeUnit, long unitCount) {
...
...
...
}
@Override
public List<GPCalendarActivity> getActivities(Date startingFrom, Time-
Duration period) {
...
}

```

AlwaysWorkingTimeCalendarImpl:

```

public class AlwaysWorkingTimeCalendarImpl extends GPCalendarBase
implements GPCalendarCalc {
...
...
...
@Override
protected List<GPCalendarActivity> getActivitiesForward(Date
startDate, TimeUnit timeUnit, long unitCount) {
...
...
...
}

@Override
protected List<GPCalendarActivity> getActivitiesBackward(Date
startDate, TimeUnit timeUnit, long unitCount) {
...
...
...
}

@Override
public List<GPCalendarActivity> getActivities(Date startDate, Date
endDate) {
...
}

```

- **Location:**
biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/GPCalendarBase.java
biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/WeekendCalendarImpl.java

biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/AlwaysWorkingTimeCalendarImpl.java
- **Reasoning:** This is an example of the template method pattern. We have 2 classes (AlwaysWorkingTimeCalendarImpl, WeekendCalendarImpl) that extend from another one GPCalendarBase. The two classes that extend from the base class are enforced to apply the behavior set by the base.

Review – Daniel Gavinho – 59889

O pattern está bem identificado, as localizações bem demonstradas e o reasoning parece estar correto. Em geral bom, está bem feito.

Builder pattern

- **Code snippet:**

ProjectDatabase.kt

```
interface TaskUpdateBuilder {
    /** Commit task update. */
    @Throws(ProjectDatabaseException::class)
    fun commit()

    fun setColor(oldValue: Color?, newValue: Color?)
    fun setCompletionPercentage(oldValue: Int, newValue: Int)
    fun setCost(oldValue: Task.Cost, newValue: Task.Cost)
    fun setCritical(oldValue: Boolean, newValue: Boolean)
    fun setCustomProperties(oldCustomProperties: CustomPropertyHolder,
        newCustomProperties: CustomPropertyHolder)
    fun setDuration(oldValue: TimeDuration, newValue: TimeDuration)
    fun setMilestone(oldValue: Boolean, newValue: Boolean)
    fun setName(oldName: String?, newName: String?)
    fun setNotes(oldValue: String?, newValue: String?)
    fun setPriority(oldValue: Task.Priority?, newValue: Task.Priority?)
    fun setProjectTask(oldValue: Boolean, newValue: Boolean)
    fun setShape(oldValue: ShapePaint?, newValue: ShapePaint?)
    fun setStart(oldValue: GanttCalendar, newValue: GanttCalendar)
    fun setWebLink(oldValue: String?, newValue: String?)

    fun interface Factory {
        fun createTaskUpdateBuilder(task: Task): TaskUpdateBuilder
    }
}
```

- **TaskImpl.kt:**

```
override fun commit() {
    ...
    ...
    myThirdChange.ifChanged {
        taskImpl.setThirdDate(it)
    }
    milestoneChange.ifChanged {
        taskImpl.isMilestone = it
        taskUpdateBuilder?.setMilestone(milestoneChange.oldValue, it)
    }
    if (hasDateFieldsChange) {
        hasActualDatesChange = taskImpl.start != myStartChange.oldValue ||
        taskImpl.duration != myDurationChange.oldValue
        if (hasActualDatesChange && taskUpdateBuilder != null) {
            if (taskImpl.start != myStartChange.oldValue) {
                taskUpdateBuilder.setStart(myStartChange.oldValue,
                taskImpl.start)
            }
            if (taskImpl.duration != myDurationChange.oldValue) {
                taskUpdateBuilder.setDuration(myDurationChange.oldValue,
                taskImpl.duration)
            }
        }
    }

    colorChange.ifChanged {
        taskImpl.color = it
        taskUpdateBuilder?.setColor(colorChange.oldValue, it)
    }
    myCompletionPercentageChange.ifChanged { completion ->
        taskImpl.completionPercentage = completion

        taskUpdateBuilder?.setCompletionPercentage(myCompletionPercentageChange
        e.oldValue, completion)
    }
    ...
    ...
}
```

- **Location:**

ganttproject/src/main/java/net/sourceforge/ganttproject/storage/ProjectDatabase.kt
ganttproject/src/main/java/net/sourceforge/ganttproject/task/TaskImpl.kt

- **Reasoning:** In these code snippets we can see an example of the builder pattern, by having multiple functions that allow the object to be built. Highlighted in yellow are the calls to the functions that build the object, the parts are assembled when needed.

Review – André Ribeiro – 59835

Código muito bem localizado e apresentado. O destaque do código a amarelo é um excelente detalhe.

Singleton

Code snippet:

```
public class GanttLanguage {
    ...
    ...
    ...
    private static final GanttLanguage ganttLanguage = new
    GanttLanguage();
    ...
    ...
    ...

    private GanttLanguage() {
        new GPAbstractOption.I18N() {
            {
                setI18N(this);
            }
            @Override
            protected String i18n(String key) {
                return getText(key);
            }
        };
        Properties charsets = new Properties();
        PropertiesUtil.loadProperties(charsets, "/charsets.properties");
        myCharSetMap = new CharSetMap(charsets);
        setLocale(Locale.getDefault());
        PropertiesUtil.loadProperties(myExtraLocales,
        "/language/extra.properties");
    }

    public static GanttLanguage getInstance() {
        return ganttLanguage;
    }
}
```

- **Location:**
ganttproject/src/main/java/net/sourceforge/ganttproject/language/GanttLanguage.java
- **Reasoning:** This is an example of a singleton. The object can only be instantiated once, since no one has access to its constructor (private). It can only exist one instance of this object and a global point of access to this instance is provided by the class.

Review – Joana Fernandes – 60065

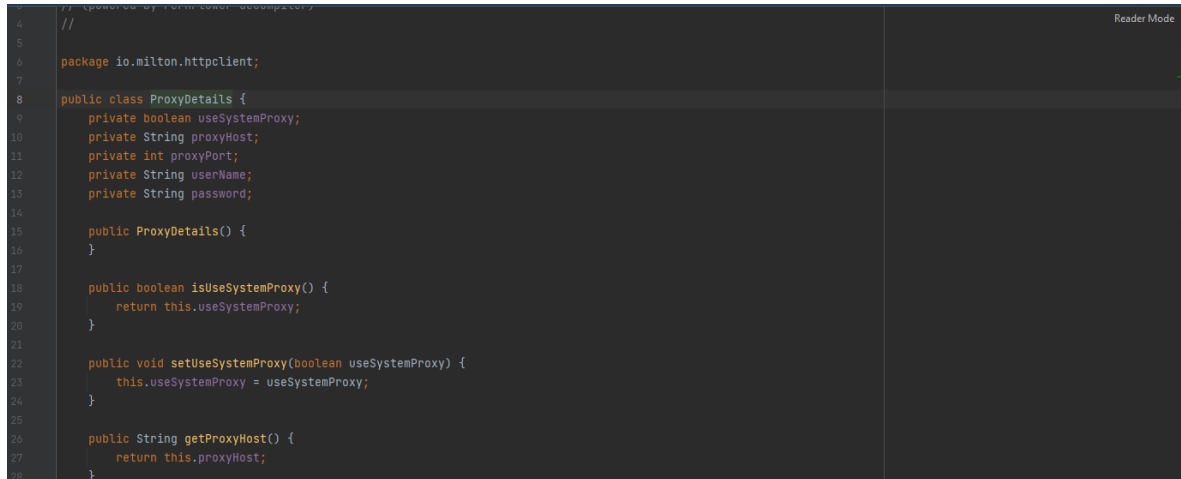
Pattern bem localizado, com uma rationale racional. Bem feito, nada a apontar.

Code Patterns – André Ribeiro – 59835

Code Pattern 1 – Proxy

Found in:

[biz.ganttproject.app.libs\lib\milton-client-2.7.4.4-bs.jar\io\milton\httpClient -> ProxyDetails.class](#)



```
// Generated by IntelliJ IDEA  
//  
package io.milton.httpClient;  
  
public class ProxyDetails {  
    private boolean useSystemProxy;  
    private String proxyHost;  
    private int proxyPort;  
    private String userName;  
    private String password;  
  
    public ProxyDetails() {  
    }  
  
    public boolean isUseSystemProxy() {  
        return this.useSystemProxy;  
    }  
  
    public void setUseSystemProxy(boolean useSystemProxy) {  
        this.useSystemProxy = useSystemProxy;  
    }  
  
    public String getProxyHost() {  
        return this.proxyHost;  
    }  
}
```

A proxy is a substitute for an object, whether it is to make an object more “lightweight” if this object requires heavy resource management, to protect its access or for remote access of the original object.

In this case this “ProxyDetails” class doesn’t get much use besides some tests in: [ganttproject-tester\test\net\sourceforge\ganttproject\document\webdav\WebDavProxyTest.java](#)

Review – Daniel Gavinho – 59889

Element 2 (André) identificou e bem uma proxy, o rationale está bem elaborado e permite perceber a utilidade do pattern

Code Pattern 2 – Prototype/Clone

Found in:

ganttproject\src\main\java\org\ganttproject -> WebStartIDClass.java

```
public class GanttPreviousState {
    private final List<GanttPreviousStateTask> myTasks;

    private String myName;

    private File myFile;

    public GanttPreviousState(String name, List<GanttPreviousStateTask> tasks) {
        myName = name;
        myTasks = tasks;
    }

    public void init() throws IOException {
        myFile = createTemporaryFile();
        myFile.deleteOnExit();
    }

    public void setName(String name) { myName = name; }

    private class BaselineSaver extends SaverBase {
        void save(File file, List<GanttPreviousStateTask> tasks) throws TransformerConfigurationException, SAXException {
            StreamResult result = new StreamResult(file);
            TransformerHandler handler = createHandler(result);
            HistorySaver saver = new HistorySaver();
            handler.startDocument();
            saver.saveBaseline(myName, tasks, handler);
            handler.endDocument();
        }
    }

    public void saveFile() throws IOException {
        BaselineSaver saver = new BaselineSaver();
        try {
            saver.save(myFile, myTasks);
        } catch (TransformerConfigurationException e) {
            throw new IOException(e);
        } catch (SAXException e) {
            throw new IOException(e);
        }
    }
}
```

A Prototype or Clone is creational code pattern that allows the copy of an existing object without making the code dependent on their classes, by cloning a new object from the original model or prototype.

This class (WebStartIDClass.java) is an example of this code pattern, as it is used as a prototype in various areas of the code, like the following:

Found in:

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

Lines: 66 - 85

```
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) {
80              return null;
81          }
82          GanttPreviousState newBaseline = new GanttPreviousState(String.valueOf(editValue),
83              GanttPreviousState.createTasks(myProject.getTaskManager()));
84          return newBaseline;
85      }
86
```

Review – Júlio Simões – 44599

-- Prototype/Clone --

Como a primeira propriedade do objecto (private String myName) está dependente do que é recebido no método -> createPrototype(Object editValue), o objeto não é clonado na sua integridade. Para ser considerado um clone este teria que ser uma copia exata do original.

Code Pattern 3 – Strategy

Found in:

[ganttproject\src\main\java\net\sourceforge\ganttproject\gui\ProjectOpenStrategy.kt](#)

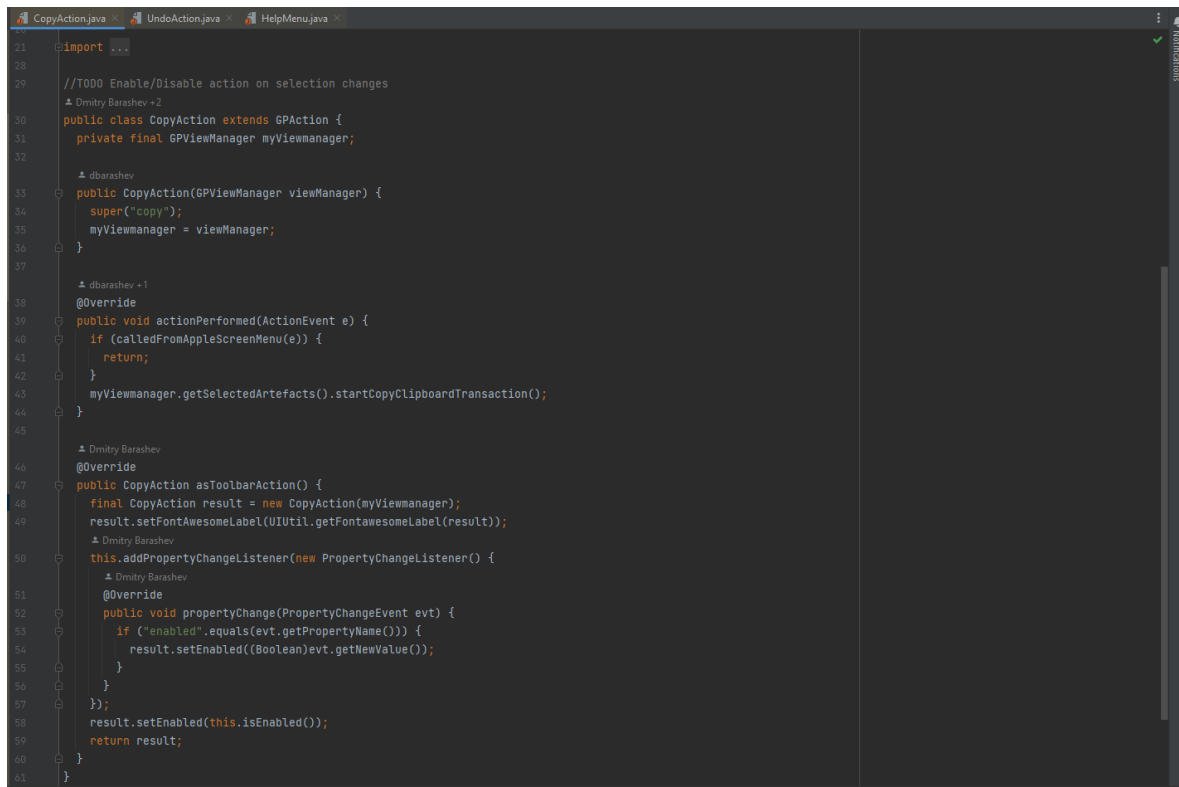
```
58
59  /**
60   * When we open a file, we need to complete a number of steps in order to be sure
61   * that should task dates change after the first scheduler run, user is informed about that.
62   *
63   * The steps are chained so that client can't invoke them in wrong order. Strategy is auto-closeable
64   * and should be closed after using.
65   *
66   * @author bard
67   */
68  internal class ProjectOpenStrategy(
69      private val project: IGanttProject,
70      private val uiFacade: UIFacade,
71      private val signIn: AuthenticationFlow,
72  ) : AutoCloseable {
73      private val myDiagnostics: ProjectOpenDiagnosticImpl
74      private val myCloseables = Lists.newArrayList<AutoCloseable>()
75      private val myEnableAlgorithmsCmd: AutoCloseable
76      private val myAlgs: AlgorithmCollection
77      private val myTasks = Lists.newArrayList<Runnable>()
78      private var myOldDuration: TimeDuration? = null
79      private val i18n = GanttLanguage.getInstance()
80      private var myResetModifiedState = true
81      enum class ConvertMilestones {
82          UNKNOWN, TRUE, FALSE
83      }
84  }
```

A strategy pattern is used to implement different algorithms while maintaining the rest of the code unchanged, since different algorithms are more appropriate for different situations. So instead of having a long and complex class with various algorithms, a strategy class is created for each algorithm needed, being ProjectOpenStrategy one of these classes.

Review – Joana Fernandes – 60065

Pattern bem localizado, apresenta características habituais do Strategy, a rationale aponta bem como funciona e porque faz sentido o seu uso.

Command Pattern



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

Folder Encontrada:

ganttproject\src\main\java\net\sourceforge\ganttproject\action\edit\CopyAction.java

Rationale: Esta classe e as outras também contidas dentro da pasta edit, permitem correr comandos efetivamente e até reverter certos comandos, mantendo todos organizados dentro da sua própria pasta estendendo a classe abstrata GPAction

Review – André Ribeiro – 59835

Código muito bem identificado e demonstrado. Pattern bem descrito no campo "Rationale".

Factory Pattern

```
1  /**
2   * Created on 20.08.2003
3   *
4   */
5   package net.sourceforge.ganttproject.document;
6
7   import ...
8
9   /**
10    * This is a helper class, to create new instances of Document easily. It
11    * chooses the correct implementation based on the given path.
12    *
13    * @author Michael Haeusler (michael at akatose.de)
14    */
15
16   public class DocumentCreator implements DocumentManager {
17       private final IGanttProject myProject;
18
19       private final UIFacade myUIFacade;
20
21       private final ParserFactory myParserFactory;
22
23       private final WebDavStorageImpl myWebDavStorage;
24       private final StringOption myWorkingDirectory = new StringOptionImpl( optionName: "working-dir", legacyTagName: "work
25
26       private final GPOptionGroup myOptionGroup;
```

Folder Encontrada:

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

Rationale: Como clarificado no comentário no topo da classe, esta classe é uma classe “helper” que ajuda na criação de vários tipos de documentos, isto pode-se identificar como um design pattern do tipo Factory Pattern.

Review - Filipe Leão - 60191

--Factory--

Esta classe de facto encapsula a gestão das operações de outras classes, funcionando como uma fábrica.

Facade

```
159
160 UIFacadeImpl(JFrame mainFrame, GanttStatusBar statusBar, NotificationManagerImpl notificationManager,
161             final IGanttProject project, UIFacade fallbackDelegate) {
162     myMainFrame = mainFrame;
163     myProject = project;
164     myDialogBuilder = new DialogBuilder(mainFrame);
165     myScrollingManager = new ScrollingManagerImpl();
166     myZoomManager = new ZoomManager(project.getTimeUnitStack());
167     myStatusBar = statusBar;
168     myStatusBar.setNotificationManager(notificationManager);
169     myFallbackDelegate = fallbackDelegate;
170     Job.getJobManager().setProgressProvider(this);
171     myTaskSelectionManager = new TaskSelectionManager(() -> project.getTaskManager());
172     myNotificationManager = notificationManager;
173
174     myLafOption = new LafOption(uiFacade: this);
175     final ShortDateFormatOption shortDateFormatOption = new ShortDateFormatOption();
176     final DefaultStringOption dateSampleOption = new DefaultStringOption("ui.dateFormat.sample");
177     dateSampleOption.setWritable(false);
178     final DefaultBooleanOption dateFormatSwitchOption = new DefaultBooleanOption("ui.dateFormat.switch", true);
179
180     dbarashev +1
181     myLanguageOption = new LanguageOption() {
182         dbarashev
183         GanttLanguage.getInstance().addListener(new GanttLanguage.Listener() {
```

Folder Encontrada:

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

Rationale: Esta classe acumula várias classes para fácil acesso por classe superiores, e maior organização das classes inferiores, servindo de um gênero de “middleman” para o programa

Review – Júlio Simões – 44599

--Facade--

Concordo com o comentário do Daniel. Esta classe aparenta ser uma interface que disponibiliza métodos que escondem outros métodos mais complexos do sistema. Tornando-o mais fácil de utilizar.

Adapter Class

- Code snippet:

```
1 // Copyright (C) 2018 Bard Software
2 package biz.ganttproject.desktop;
3
4 import ...
5
14
15 /**
16  * @author dbarashev@bardsoftware.com
17  */
18 public class DesktopAdapter {
19     public static void install(final GanttProjectApi api) {
20         Desktop desktop = Desktop.getDesktop();
21         desktop.setAboutHandler(new AboutHandler() {
22             @Override
23             public void handleAbout(AboutEvent e) { api.showAboutDialog(); }
24         });
25         desktop.setPreferencesHandler(new PreferencesHandler() {
26             @Override
27             public void handlePreferences(PreferencesEvent e) { api.showPreferencesDialog(); }
28         });
29         desktop.setQuitHandler(new QuitHandler() {
30             @Override
31             public void handleQuitRequestWith(QuitEvent e, final java.awt.desktop.QuitResponse response) {
32                 api.maybeQuit(new QuitResponse() {
33                     @Override
34                     public void performQuit() { response.performQuit(); }
35
36                     @Override
37                     public void cancelQuit() { response.cancelQuit(); }
38                 });
39             }
40         });
41         desktop.setOpenFileHandler(new OpenFilesHandler() {
42             @Override
43             public void openFiles(OpenFilesEvent e) {
44                 List<File> files = e.getFiles();
45                 if (files.isEmpty()) {
46                     return;
47                 }
48                 File file = files.get(0);
49                 if (!file.isFile() || !file.canRead()) {
50                     return;
51                 }
52                 api.openFile(file);
53             }
54         });
55     }
56 }
```

- Localização:

ganttproject\Project\ProjectFiles\biz.ganttproject.desktop\src\biz\ganttproject\desktop

- Rationale: a classe é composta por handlers que estabelecem a comunicação entre duas classes que seriam de outro modo incompatíveis.

Como apontado pelo elemento 4 (Joana) esta classe permite a comunicação entre duas outras. Então concordo que isto enquadra-se no pattern de Adapter Class

Template Method Pattern

- Code snippets:

ChartItem.java

```
11 public class ChartItem {
12     private final Task myTask;
13
14     protected ChartItem(Task task) { myTask = task; }
15
16
17
18     public Task getTask() { return myTask; }
19
20 }
21
22
```

TaskBoundaryChartItem.java

```
11 public class TaskBoundaryChartItem extends ChartItem {
12
13     private final boolean isStart;
14
15     public TaskBoundaryChartItem(Task task, boolean isStart) {
16         super(task);
17         this.isStart = isStart;
18     }
19
20     public boolean isStartBoundary() { return isStart; }
21
22 }
23
```

CalendarChartItem.java

```
28 public class CalendarChartItem extends ChartItem {
29     private final Date myDate;
30
31     public CalendarChartItem(Date date) {
32         super(null);
33         myDate = date;
34     }
35
36     public Date getDate() { return myDate; }
37
38 }
39
```

- **Localização:**

ganttproject\Project\ProjectFiles\ganttproject\src\main\java\net\sourceforge\
ganttproject\chart\item\ChartItem.java

ganttproject\Project\ProjectFiles\ganttproject\src\main\java\net\sourceforge\
ganttproject\chart\item\TaskBoundaryChartItem.java

ganttproject\Project\ProjectFiles\ganttproject\src\main\java\net\sourceforge\
ganttproject\chart\item\CalendarChartItem.java

- **Rationale:** duas das classes(TaskBoundaryItem, CalendarChartItem) estendem-se a partir da classe ChartItem. Estas classes realizam operações semelhantes, pelo que faz sentido aplicar a herança de classes.

Review – Júlio Simões – 44599

Não acho que este exemplo possa ser considerado como um template method pattern.

A ideia de um template, é a de que um classe (classe base) decide uma serie de passos necessários para resolver determinado problema. As sub classes implementam os métodos abstratos determinado pela classe base, e podem fazer override de métodos já definidos pela classe superior, para se poderem adaptar a resolver problemas mais específicos.

Facade Pattern

- Code snippet:

```
53 public class TimelineFacadeImpl implements MouseInteraction.TimelineFacade {
54     private final ChartModelBase myChartModel;
55     private final TaskManager myTaskManager;
56     private VScrollController myVScrollController;
57
58     public TimelineFacadeImpl(ChartModelBase chartModel, TaskManager taskManager) {
59         myChartModel = chartModel;
60         myTaskManager = taskManager;
61     }
62
63     @Override
64     public Date getDateAt(int x) { return myChartModel.getOffsetAt(x).getOffsetStart(); }
65
66     @Override
67     public TimeDuration createTimeInterval(TimeUnit timeUnit, Date startDate, Date endDate) {
68         WorkingUnitCounter workingUnitCounter = new WorkingUnitCounter(getCalendar(), timeUnit);
69         if (startDate.before(endDate)) {
70             return workingUnitCounter.run(startDate, endDate);
71         }
72         return workingUnitCounter.run(endDate, startDate).reverse();
73     }
74
75     @Override
76     public TimeUnitStack getTimeUnitStack() { return myChartModel.getTimeUnitStack(); }
77
78     @Override
79     public GPCalendarCalc getCalendar() { return myTaskManager.getCalendar(); }
80
81     @Override
82     public Date getEndDateAt(int x) { return myChartModel.getOffsetAt(x).getOffsetEnd(); }
83
84     @Override
85     public ScrollingSession createScrollingSession(final int xpos, final int ypos) {
86         return new ScrollingSession() {
87             private final ScrollingSession myDelegate = myChartModel.createScrollingSession(xpos);
88             private int myStartYpos = ypos;
89         };
90     }
91 }
```

- Localização:

Ganttproject\Project\ProjectFiles\ganttpproject\src\main\java\net\sourceforge\ganttpproject\chart\mouse\TimelineFacadeImpl.java

- **Rationale:** a classe agrupa dentro de si diversas classes, assim facilitando o acesso aos métodos necessários e diminuindo dependências entre classes.

Review – André Ribeiro – 59835

O código apresentado corresponde evidentemente ao pattern facade e por isso é uma boa inclusão aos patterns do documento.

Code Patterns – Filipe Leão – 60191

1. Factory Method

- Code Snippet

```
50  @Override
51  public Role createRole(String name) {
52      int maxId = 0;
53      for (Role role : myRoles) {
54          if (role.getID() > maxId) {
55              maxId = role.getID();
56          }
57      }
58      return createRole(name, maxId + 1);
59  }
60
61  @Override
62  public Role createRole(String name, int persistentID) {
63      RoleImpl result = new RoleImpl(persistentID, name, this);
64      myRoles.add(result);
65      myRoleManager.fireRolesChanged(this);
66      return result;
67  }
```

- Path

ganttproject\src\main\java\net\sourceforge\ganttproject\roles\RoleSetImpl.java

- Rationale

The method in question, RoleSet.createRole() is used to create a Role object and to process it as needed by the Role set and Manager so the Role constructor is never called outside of this structure.

Review – André Ribeiro – 59835

Pattern bem identificado, localizado e explicado no Rationale, explicando o pattern no contexto do projeto.

2. Proxy

- Code Snippet

```
34 public class ReadOnlyProxyDocument implements Document {
35
36     private final Document myDelegate;
37
38     public ReadOnlyProxyDocument(Document delegate) {
39         myDelegate = delegate;
40     }
41
42     @Override
43     public String getFileName() {
44         return myDelegate.getFileName();
45     }
46
47     @Override
48     public boolean canRead() {
49         return myDelegate.canRead();
50     }
51
52     @Override
53     public IStatus canWrite() {
54         return new Status(IStatus.ERROR, "net.sourceforge.ganttproject", 0, "You can't write a read-only document", null);
55     }
56 }
```

- Path

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

- Rationale

The purpose of this class is to protect the access to data stored in Documents tagged as read only. This is very clear, for example, in the Document.canWrite() method, which in this implementation it is not possible, since it returns an error Status object.

Review – Joana Fernandes – 60065

Rationale bem apontada, é possível perceber claramente que é um Proxy Pattern, e porque é necessário.

3. Template Method

- Code Snippet

```
105     protected void onEndElement() {  
106     }
```

```
59     @Override  
60     public void onEndElement() {  
61         if (hasCdata()) {  
62             myOption.loadPersistentValue(getCdata());  
63             clearCdata();  
64         }  
65     }  
66 }
```

```
35     @Override  
36     protected void onEndElement() {  
37         myFieldsHandler.setEnabled(enabled: false);  
38     }
```

- Path

ganttproject\src\main\java\net\sourceforge\ganttproject\parser\AbstractTagHandler.java

- Rationale

This abstract method, TagHandler.onEndElement(), represents a behaviour all subclasses of AbstractTagHandler should have, opening up the possibility of adding subclass-specific actions. In this case, we can see two different extensions of said method.

Review – Daniel Gavinho – 59889

O elemento 5 apresenta bem o seu rationale em como esta classe é um pattern to tipo template