

1º Fase

Trabalho de Engenharia de Software (Report)

Trabalho realizado por:

Vasco Malta	58592
Rafael Mira	59243
Bruno David	60011
Mafalda Batalha	60684
Gonçalo Cerveira	61696

Code Smells

Bruno David

- Long method

Package:

biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/canvas/FontChooser.java

Class: Font Chooser

O método `getFont` é muito longo, sendo mais legível, fracionar em métodos mais pequenos, por exemplo, no código a executar em cada condição.

```
Font f = myFonts.get(style);
if (f == null) {
    String propValue = Strings.nullToEmpty(myProperties.getProperty(style + ".font")).trim();
    if (propValue.isEmpty()) {
        // If .font property is not set then we use the base font
        f = myBaseFont.get();
    } else {
        String[] components = propValue.split(regex "\\s+");
        String last = components[components.length - 1];
        String family = "";
        float absoluteSize;
        try {
            // If the last component of .font property is int/float then
            // we check whether it is a relative increment (it should be prefixed with sign)
            // or an absolute value
            if (last.startsWith("+") || last.startsWith("-")) {
                absoluteSize = Float.parseFloat(last) + myBaseFont.get().getSize();
            } else {
                absoluteSize = Float.parseFloat(last);
            }
            if (components.length > 1) {
                family = Joiner.on(separator ' ').join(Arrays.asList(components).subList(0, components.length - 1));
            }
            if (family.isEmpty()) {
                f = myBaseFont.get().deriveFont(absoluteSize);
            } else {
                f = Font.decode(str family + " 10");
                if (f == null) {
                    f = myBaseFont.get();
                }
                f = f.deriveFont(absoluteSize);
            }
        } catch (NumberFormatException e) {
            f = Font.decode(propValue);
        }
    }
}
```

Revisto por:

Gonçalo Cerveira
Vasco Malta

- **Speculative Generality**

Package: net.sourceforge.ganttproject.action.edit (ganttproject.src.main.java)

Class: PasteAction

Esta classe tem dois construtores, porém um deles é privado e nunca chega a ser utilizado, provavelmente porque foi criado anteriormente com a expectativa de ser necessário.

```
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;
}

▴ dbarashev +1
private PasteAction(IGanttProject project, UIFacade uiFacade, GPViewManager viewmanager, IconSize size, GPUndoManager undoManager) {
    super( name: "paste", size);
    myViewmanager = viewmanager;
    myUndoManager = undoManager;
    myProject = project;
    myUiFacade = uiFacade;
}
```

Revisto por:

Vasco Malta

Mafalda Batalha

- **Dead Code**

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

Class: CalendarEvent

Este método não é usado e, por isso, devia ser eliminado

```
▴ dbarashev
public static CalendarEvent newEvent(Date date, boolean isRecurring, Type type, String title, Color color) {
    return new CalendarEvent(date, isRecurring, type, title, color);
}
```

Revisto por:

Gonçalo Cerveira

Vasco Malta

Rafael Mira

- **Speculative Generality**

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

Class: AlwaysWorkingTimeCalendarImpl.java

Na `class AlwaysWorkingTimeCalendarImpl`, há vários métodos que têm o corpo vazio e portanto não fazem nada. Apenas estão na classe porque esta implementa uma interface que tem estes métodos quando não são necessários para a classe em questão.

```
@Override
public void setPublicHolidays(Collection<CalendarEvent> holidays) {
}
```

```
@Override
public void setBaseCalendarID(String id) {
}
```

```
@Override
public void importCalendar(GPCalendar calendar, ImportCalendarOption importOption) {
}
```

Interface em questão:

```
public interface GPCalendar
```

Podemos resolver esta situação utilizando o princípio da Interface Segregation. A interface GPCalendar ficava apenas com os métodos comuns a todas as classes que a implementam e, para cada classe, criava-se uma interface específica com os métodos que lhe são exclusivos evitando assim ter métodos que não fazem nada.

Revisto por:

Bruno David

Vasco Malta

- **Message Chains**

Package: ganttproject/src/main/java/net/sourceforge/ganttproject/task/dependency/constraint

Class: StartFinishConstraintImpl.java

Na `class StartFinishConstraintImpl`, no método `getCollision()`, podemos observar a mesma message chain por duas vezes.

```
@Override
public TaskDependencyConstraint.Collision getCollision() {
    Task dependee = getDependency().getDependee();
    Task dependant = getDependency().getDependant();
    final GanttCalendar dependeeStart = dependee.getStart();
    final GanttCalendar dependantEnd = dependant.getEnd();

    GanttCalendar acceptableEnd = CalendarFactory.createGanttCalendar(dependant.getManager().getCalendar().shiftDate(
        dependeeStart.getTime(), dependant.getManager().createLength(getDependency().getDifference())));

    boolean isActive = getDependency().getHardness() == TaskDependency.Hardness.RUBBER ? dependantEnd.compareTo(acceptableEnd) < 0
        : dependantEnd.compareTo(acceptableEnd) != 0;

    GanttCalendar acceptableStart = isActive ? CalendarFactory.createGanttCalendar(
        dependant.getManager().getCalendar().shiftDate(acceptableEnd.getTime(), dependant.getDuration().reverse())) : dependant.getStart();
    return new TaskDependencyConstraint.DefaultCollision(acceptableStart,
        TaskDependencyConstraint.Collision.START_LATER_VARIATION, isActive);
}
```

Em concreto `dependant.getManager().getCalendar().shiftDate()`.

Visto que em ambas a chamadas se está a usar o método `dependant.getManager()`, podíamos utilizar uma variável local `TaskManager manager = dependant.getManager();` E faríamos o mesmo para o calendário do manager.

`GPCalendarCalc managerCalendar = manager.getCalendar();`

Evitando assim estar a ir buscar os mesmo objetos por várias vezes, diminuindo a complexidade do código.

Revisto por:

Vasco Malta

- **Large Class**

Package: biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/canvas/

Class: Canvas.java

A `class Canvas` é um aglomerado de várias classes contendo 10 classes no total. O facto de haver 9 classes todas agrupadas dentro de uma torna a classe principal muito extensa e muito mais complexa do que devia ser.

```
public static class Shape |
public static class Polygon extends Shape
public static class Rectangle extends Polygon
public static class Rhombus extends Polygon
public static class Arrow extends Shape
public static class Label
public static class Line extends Shape
public static class Text extends Shape
public static class TextGroup
```

Para melhorar esta parte do código podiam-se criar as classes fora desta classe Canvas, cada uma num ficheiro separado, para tornar a classe mais pequena, legível e mais objetiva.

Revisto por:

Gonçalo Cerveira

Vasco Malta

- **Comments**

Package: biz.ganttproject.core.option

As classes da package acima referida não apresentam quaisquer comentários, o que é considerado code smell pois uma pessoa que não conheça o projeto ou mesmo o programador original pode ter dificuldade em entender as funcionalidades do projeto.

Revisto por:

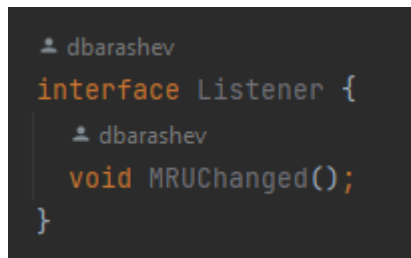
Rafael Mira

- **Speculative Generality**

Package: net.sourceforge.ganttproject.document (ganttproject.src.main.java)

Class: DocumentMRU

Nesta classe foi criada uma interface Listener possivelmente com a probabilidade de ser necessária posteriormente, mas que acabou por nunca ser utilizada.

A screenshot of a code editor showing the definition of an interface named 'Listener'. The code is written in Java and includes a package declaration 'net.sourceforge.ganttproject.document' and a class declaration 'DocumentMRU'. The interface 'Listener' is defined with a single method 'MRUChanged()' of type 'void'. The code is as follows:

```
package net.sourceforge.ganttproject.document;

import net.sourceforge.ganttproject.core.option;

public class DocumentMRU {

    interface Listener {

        void MRUChanged();

    }

}
```

Revisto por:

Mafalda Batalha

- **Data Class**

Package: biz.ganttproject.core.option

Class: DefaultStringOption, DefaultMoneyOption, DefaultIntegerOption

As Classes acima referidas são exemplos de code smell pois são usadas apenas para armazenar data.

```
/**
 * Default implementation of MoneyOption
 *
 * @author dbarashev (Dmitry Barashev)
 */
2 usages  ▲ dbarashev
public class DefaultMoneyOption extends GPAbstractOption<BigDecimal> implements MoneyOption {

    2 usages  ▲ dbarashev
    public DefaultMoneyOption(String id) { super(id); }

    ▲ dbarashev
    public DefaultMoneyOption(String id, BigDecimal initialValue) { super(id, initialValue); }

    ▲ dbarashev
    @Override
    public String getPersistentValue() { return getValue().toPlainString(); }

    ▲ dbarashev
    @Override
    public void loadPersistentValue(String value) { resetValue(new BigDecimal(value), resetInitial: true); }
}
```

```
/**
 *
 */
package biz.ganttproject.core.option;

▲ dbarashev
public class DefaultIntegerOption extends GPAbstractOption<Integer> implements IntegerOption {
    ▲ dbarashev
    public DefaultIntegerOption(String id) { this(id, initialValue: 0); }

    5 usages  ▲ dbarashev
    public DefaultIntegerOption(String id, Integer initialValue) { super(id, initialValue); }

    ▲ dbarashev
    @Override
    public String getPersistentValue() {
        int value = getValue();
        return String.valueOf(value);
    }

    ▲ dbarashev
    @Override
    public void loadPersistentValue(String value) {
        int intValue = Integer.parseInt(value);
        resetValue(intValue, resetInitial: true);
    }
}
```

```
/**
 *
 */
package biz.ganttproject.core.option;

▲ dbarashev
public class DefaultStringOption extends GPAbstractOption<String> implements StringOption {
    ▲ dbarashev
    public DefaultStringOption(String id) { super(id); }

    ▲ dbarashev
    public DefaultStringOption(String id, String initialValue) { super(id, initialValue); }

    2 overrides  ▲ dbarashev
    @Override
    public String getPersistentValue() { return getValue(); }

    1 override  ▲ dbarashev
    @Override
    public void loadPersistentValue(String value) { setValue(value); }
}
```

Revisto por: Gonalo Cerveira

Mafalda Batalha

- **Dead Code**

Package: biz.ganttproject.core.option (biz.ganttproject.core.src.main.java)

A classe DefaultIntegerOption tem dois construtores, porém um deles não é utilizado e, portanto, é um exemplo de Dead Code e deveria ser removido.

```
public class DefaultIntegerOption extends GPAbstractOption<Integer> implements IntegerOption {  
    public DefaultIntegerOption(String id) { this(id, initialValue: 0); }  
  
    public DefaultIntegerOption(String id, Integer initialValue) { super(id, initialValue); }
```

Classe DefaultIntegerOption com dois construtores

Revisto por:

Bruno David

Vasco Malta

- **Speculative Generality**

Package: net.sourceforge.ganttproject.resource (biz.ganttproject.core.src.main.java)

Na classe AssignmentNode, existe um método (setStandardField) que tem um switch statement que contém apenas um case. No caso, seria mais correto usar um if, dado que no momento só se está a considerar apenas um cenário.

```
@Override  
public void setStandardField(ResourceDefaultColumn def, Object value) {  
    switch (def) {  
        case ROLE_IN_TASK:  
            setRoleForAssignment((Role) value);  
            return;  
    }  
}
```

Método setStandardField da classe AssignmentNode

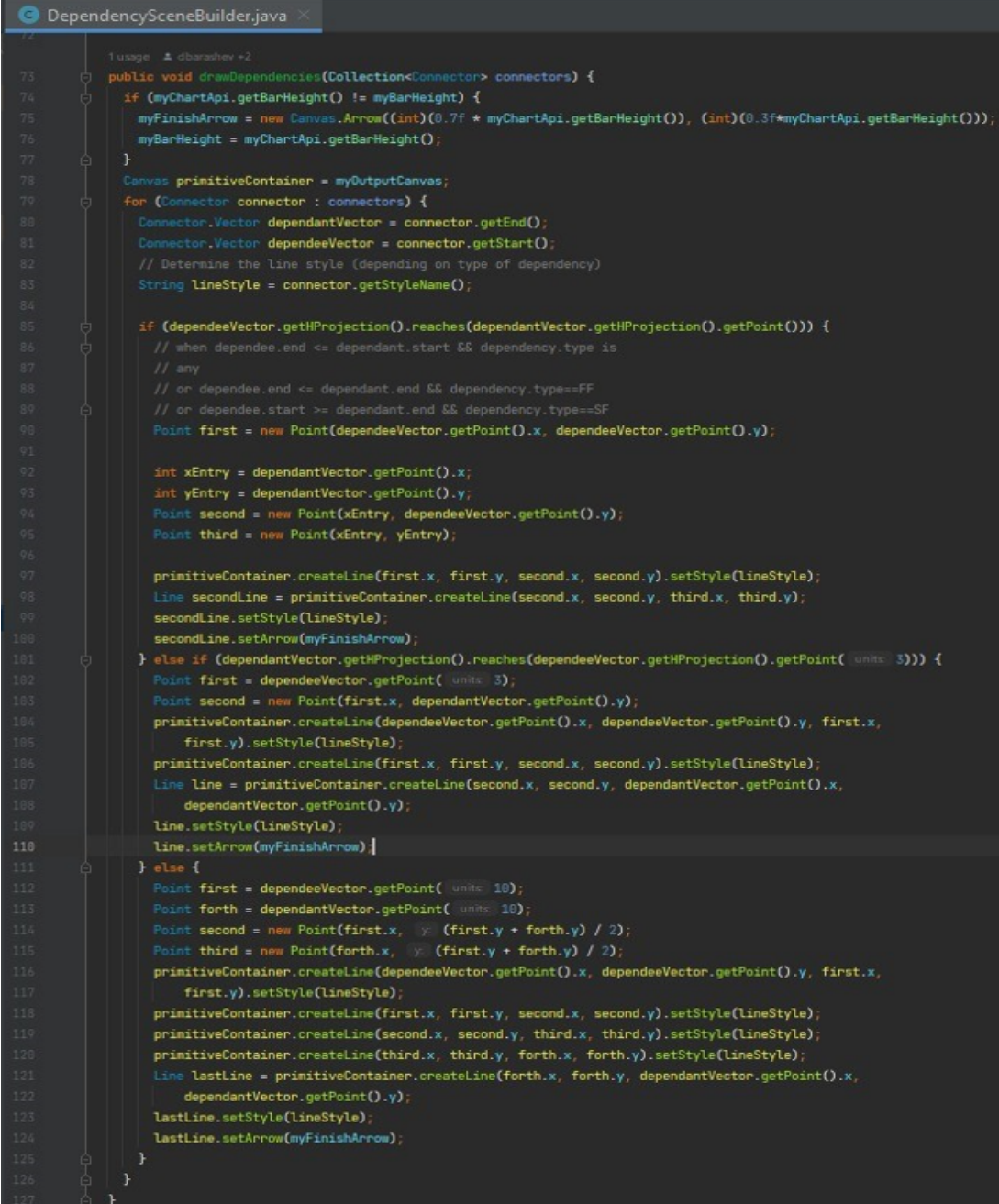
Revisto por:

Rafael Mira

- Long Method

Package: biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/scene

A classe DependencySceneBuilder tem alguns métodos como DrawDependencies e ConnectEndArrow que apresentam um tamanho considerável. Isso causa dificuldade na leitura e compreensão do código e, por isso, seria recomendável repartir em partes, por exemplo, em métodos auxiliares.



```
72
73 1 usage 2 dbarashev +2
74 public void drawDependencies(Collection<Connector> connectors) {
75     if (myChartApi.getBarHeight() != myBarHeight) {
76         myFinishArrow = new Canvas.Arrow((int)(0.7f * myChartApi.getBarHeight()), (int)(0.3f*myChartApi.getBarHeight()));
77         myBarHeight = myChartApi.getBarHeight();
78     }
79     Canvas primitiveContainer = myOutputCanvas;
80     for (Connector connector : connectors) {
81         Connector.Vector dependantVector = connector.getEnd();
82         Connector.Vector dependeeVector = connector.getStart();
83         // Determine the line style (depending on type of dependency)
84         String lineStyle = connector.getStyleName();
85
86         if (dependeeVector.getHProjection().reaches(dependantVector.getHProjection().getPoint())) {
87             // when dependee.end <= dependant.start && dependency.type is
88             // any
89             // or dependee.end <= dependant.end && dependency.type==FF
90             // or dependee.start >= dependant.end && dependency.type==SF
91             Point first = new Point(dependeeVector.getPoint().x, dependeeVector.getPoint().y);
92
93             int xEntry = dependantVector.getPoint().x;
94             int yEntry = dependantVector.getPoint().y;
95             Point second = new Point(xEntry, dependeeVector.getPoint().y);
96             Point third = new Point(xEntry, yEntry);
97
98             primitiveContainer.createLine(first.x, first.y, second.x, second.y).setStyle(lineStyle);
99             Line secondLine = primitiveContainer.createLine(second.x, second.y, third.x, third.y);
100             secondLine.setStyle(lineStyle);
101             secondLine.setArrow(myFinishArrow);
102         } else if (dependantVector.getHProjection().reaches(dependeeVector.getHProjection().getPoint( units: 3))) {
103             Point first = dependeeVector.getPoint( units: 3);
104             Point second = new Point(first.x, dependantVector.getPoint().y);
105             primitiveContainer.createLine(dependeeVector.getPoint().x, dependeeVector.getPoint().y, first.x,
106             first.y).setStyle(lineStyle);
107             primitiveContainer.createLine(first.x, first.y, second.x, second.y).setStyle(lineStyle);
108             Line line = primitiveContainer.createLine(second.x, second.y, dependantVector.getPoint().x,
109             dependantVector.getPoint().y);
110             line.setStyle(lineStyle);
111             line.setArrow(myFinishArrow);
112         } else {
113             Point first = dependeeVector.getPoint( units: 10);
114             Point forth = dependantVector.getPoint( units: 10);
115             Point second = new Point(first.x, y: (first.y + forth.y) / 2);
116             Point third = new Point(forth.x, y: (first.y + forth.y) / 2);
117             primitiveContainer.createLine(dependeeVector.getPoint().x, dependeeVector.getPoint().y, first.x,
118             first.y).setStyle(lineStyle);
119             primitiveContainer.createLine(first.x, first.y, second.x, second.y).setStyle(lineStyle);
120             primitiveContainer.createLine(second.x, second.y, third.x, third.y).setStyle(lineStyle);
121             primitiveContainer.createLine(third.x, third.y, forth.x, forth.y).setStyle(lineStyle);
122             Line lastLine = primitiveContainer.createLine(forth.x, forth.y, dependantVector.getPoint().x,
123             dependantVector.getPoint().y);
124             lastLine.setStyle(lineStyle);
125             lastLine.setArrow(myFinishArrow);
126         }
127     }
128 }
```

Revisto por:

Rafael Mira

Gonçalo Cerveira

- **Speculative Generality**

Package: biz.ganttproject.core/src/main/java/biz/ganttproject/core/time/impl

Class: GregorianTimeUnitStack

Nesta classe existem métodos não implementados apenas porque existem na interface, mesmo estes não sendo usados em lado nenhum.

```
dbarashev
@Override
public TimeDuration createDuration(TimeUnit timeUnit, int count) {
    // TODO Auto-generated method stub
    return null;
}

dbarashev
@Override
public TimeUnit findTimeUnit(String code) {
    // TODO Auto-generated method stub
    return null;
}

dbarashev
@Override
public TimeDuration parseDuration(String duration) throws ParseException {
    // TODO Auto-generated method stub
    return null;
}
```

Revisto por:

Bruno David

- **Dead Code**

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

Class: WeekendCalendarImpl

Este método não é usado, pelo que deveria ser removido do código.

```
dbarashev
public CalendarEvent getEvent(Date date) {
    CalendarEvent result = myOneOffEvents.get(date);
    if (result == null) {
        result = myRecurringEvents.get(getRecurringDate(date));
    }
    return result;
}
```

Revisto por:

Rafael Mira

- **Long Method**

Package: biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/render/RectangleRenderer.java

Class: RectangleRenderer

Este método é longo, tornando-se confuso para a pessoa que o tenta perceber, uma solução seria fragmentar o código em pequenas partes, conseguindo assim fazer as verificações necessárias para esta método perceptíveis e, caso necessário acrescentar informação mais detalhada, as verificações e os métodos continuavam a ser legíveis.

```
public boolean render(Canvas.Rectangle rect) {
    Graphics2D g = (Graphics2D) myGraphics.create();
    Style style = Style.getStyle(myProperties, rect.getStyle());

    if (style.getVisibility(rect) == Style.Visibility.HIDDEN) {
        return false;
    }
    Style.Color background = style.getBackgroundColor(rect);
    if (background != null) {
        g.setColor(background.get());
    }
    Paint paint = style.getBackgroundPaint(rect);
    if (paint != null) {
        g.setPaint(paint);
    }
    Float opacity = style.getOpacity(rect);
    if (opacity != null) {
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, opacity.floatValue()));
    }
    Style.Padding padding = style.getPadding();
    if (style.getBackgroundImage() != null) {
        g.drawImage(style.getBackgroundImage(), rect.getLeftX() + padding.getLeft(), rect.getTopY() + padding.getTop(), null);
    } else {
        g.fillRect(rect.getLeftX() + padding.getLeft(), rect.getTopY() + padding.getTop(),
            rect.getWidth() - (padding.getLeft() + padding.getRight()), rect.getHeight() - (padding.getTop() + padding.getBottom()));
    }
    Style.Borders border = style.getBorder(rect);
    if (border != null) {
        renderBorders(g, border, rect.getLeftX(), rect.getTopY(), rect.getWidth(), rect.getHeight());
    }
    return true;
}
```

Revisto por:

Rafael Mira

Design Patterns

Bruno David

- **Decorator**

Package: biz.ganttproject.core/src/main/java/biz/ganttproject/core/calendar/walker
Class: WorkingUnitCounter

A classe WorkingUnitCounter implementa os métodos inerentes aos seus objetos, no entanto esses mesmos objetos podem comportar-se de modo a que é inerente a outros (ForwardTimeWalker)

```
25 usages  👤 dbarashev  
public class WorkingUnitCounter extends ForwardTimeWalker {
```

```
3 usages  2 inheritors  👤 dbarashev +1  
public abstract class ForwardTimeWalker {
```

Revisto por:

Gonçalo Cerveira

- **Iterator**

Package: biz.ganttproject.impex.csv
Class: WorkingUnitCounter

Nesta interface é utilizado um iterador, aparentemente para iterar o registo de uma linha na spreadsheet.

```
👤 Dmitry Barashev  
interface SpreadsheetReader : Closeable {  
    👤 Dmitry Barashev  
    operator fun iterator(): Iterator<SpreadsheetRecord>  
}
```

Revisto por:

Gonçalo Cerveira

- **Abstract Factory**

Package: ganttproject.task.dependency

Class: WorkingUnitCounter

Ambas as classes estendem uma classe abstrata, o que mostra claramente a existência de dois objetos que têm semelhanças e, por isso, partilham métodos, todavia não representam a mesma instância.

2 usages  dbarashev

```
public class TaskDependencySliceAsDependee extends TaskDependencySliceImpl {
```

2 usages  dbarashev

```
public class TaskDependencySliceAsDependant extends TaskDependencySliceImpl {
```

Revisto por:

Mafalda Batalha

Rafael Mira

- **Chain of Responsibility**

Package:

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

Class: DependencySceneBuilder

No método `createConnector` da `class DependencySceneBuilder`, podemos ver uma cadeia de if que são responsáveis por devolver um valor para a variável `xDantEntry`.

```
int xDantEntry;
if (myTaskApi.isMilestone(dependant.getOwner())) {
    xDantEntry = dependantRectangle.getMiddleX();
} else if (dependantDirection == Connector.Vector.WEST) {
    xDantEntry = endArrow == ConnectorEndArrow.VERTICAL ? dependantRectangle.getLeftX() + 3 : dependantRectangle.getLeftX();
} else if (dependantDirection == Connector.Vector.EAST) {
    xDantEntry = endArrow == ConnectorEndArrow.VERTICAL ? dependantRectangle.getRightX() - 3 : dependantRectangle.getRightX();
} else {
    xDantEntry = dependantRectangle.getMiddleX();
}
```

Revisto por:

Bruno David

- **Iterator**

Package:

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

Class: WeekendCalendarImpl

No método `setPublicHolidays`, da classe `WeekendCalendarImpl`, está-se a iterar sobre os elementos da coleção `holidays` utilizando um `for each`.

```
public void setPublicHolidays(Collection<CalendarEvent> holidays) {
    myRecurringEvents.clear();
    myOneOffEvents.clear();
    for (CalendarEvent h : holidays) {
        if (h.isRecurring) {
            myCalendar.setTime(h.myDate);
            myCalendar.set(Calendar.YEAR, DUMMY_YEAR_FOR_RECURRING_EVENTS);
            myRecurringEvents.put(myCalendar.getTime(), h);
        } else {
            myOneOffEvents.put(h.myDate, h);
        }
    }
}
```

Revisto por:

Mafalda Batalha

- **Template Method**

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

Class: GPCalendarBase, AlwaysWorkingTimeCalendarImpl, WeekendCalendarImpl

Na `abstract class GPCalendarBase`, o método `getDayMask(Date date)` é implementado de formas diferentes pelas suas subClasses.

```
public class AlwaysWorkingTimeCalendarImpl extends GPCalendarBase implements GPCalendarCalc
```

```
@Override  
public int getDayMask(Date date) { return GPCalendar.DayMask.WORKING; }
```

```
public class WeekendCalendarImpl extends GPCalendarBase implements GPCalendarCalc
```

```
@Override  
public int getDayMask(Date date) {  
    int result = 0;  
    myCalendar.setTime(date);  
    int dayOfWeek = myCalendar.get(Calendar.DAY_OF_WEEK);  
    boolean isHoliday = isPublicHoliday(date);  
    boolean isWeekend = myTypes[dayOfWeek - 1] == DayType.WEEKEND;  
    if (isWeekend) {  
        result |= DayMask.WEEKEND;  
        CalendarEvent oneOff = myOneOffEvents.get(date);  
        if (oneOff != null && oneOff.getType() == Type.WORKING_DAY) {  
            result |= DayMask.WORKING;  
        }  
    }  
    if (isHoliday) {  
        result |= DayMask.HOLIDAY;  
        return result;  
    }  
    if (!isWeekend || myOnlyShowWeekends) {  
        result |= DayMask.WORKING;  
    }  
    return result;  
}
```

Revisto por:

Gonçalo Cerveira

Vasco Malta

- **Decorator**

Package: biz.ganttproject.core.option

Escolhi a Decorator Design Pattern pois todas as classes implementam a mesma abstract class que esconde a complexidade do programa.

```
public class DefaultBooleanOption extends GPAbstractOption<Boolean> implements BooleanOption {  
  
public class DefaultDateOption extends GPAbstractOption<Date> implements DateOption {  
  
public class DefaultDoubleOption extends GPAbstractOption<Double> implements DoubleOption {  
  
public class DefaultEnumerationOption<T> extends GPAbstractOption<String> implements EnumerationOption {  
  
public class DefaultFontOption extends GPAbstractOption<FontSpec> implements FontOption {  
  
public class DefaultIntegerOption extends GPAbstractOption<Integer> implements IntegerOption {  
  
public class DefaultMoneyOption extends GPAbstractOption<BigDecimal> implements MoneyOption {  
  
public class DefaultStringOption extends GPAbstractOption<String> implements StringOption {
```

Revisto por:

Gonalo Cerveira

- **Singleton**

Package: biz.ganttproject.core

A classe OperationStatus implementa o Singleton Pattern uma vez que pode apenas existir uma instanciao da mesma, tendo em conta que o seu construtor   privado.

```
6 usages  Dmitry Barashev  
private OperationStatus(boolean ok, T result, String message, @NonNull Code resultCode) {  
    myOperationResult = result;  
    myMessage = message;  
    myOk = ok;  
    myResultCode = Preconditions.checkNotNull(resultCode);  
}
```

Revisto por:

Mafalda Batalha

- **Facade**

(Deverá ser corrigido posteriormente)

Package: biz.ganttproject.core.chart.text

Escolhi a facade pattern pois as 5 seguintes classes implementam diretamente na mesma interface tendo métodos com assinatura iguais mas com implementações diferentes.

```
public class WeekTextFormatter extends CachingTextFormatter implements TimeFormatter {  
public class QuarterTextFormatter extends CachingTextFormatter implements TimeFormatter {  
public class MonthTextFormatter extends CachingTextFormatter implements TimeFormatter {  
public class MonthTextFormatter extends CachingTextFormatter implements TimeFormatter {
```

Revisto por:

Bruno David

Mafalda Batalha

- **Template Method**

Package: biz.ganttproject.core.chart.text (biz.ganttproject.core.src.main.java)

A classe abstrata `CachingTextFormatter` contém um método inicializado `CreateTimeUnixText()`, utilizado no método implementado `format()`, que será implementado de forma diferente por cada classe que a estende.

```
public abstract class CachingTextFormatter {
    private final HashMap<Date, TimeUnitText[]> myTextCache = new HashMap();
    private TimeFormatters.LocaleApi myLocale;

    protected CachingTextFormatter() {
    }

    public TimeUnitText[] format(Offset curOffset) {
        return this.format(curOffset.getOffsetUnit(), curOffset.getOffsetStart());
    }

    public TimeUnitText[] format(TimeUnit timeUnit, Date baseDate) {
        TimeUnitText[] result = null;
        Date adjustedLeft = timeUnit.adjustLeft(baseDate);
        result = this.getCachedText(adjustedLeft);
        if (result == null) {
            result = this.createTimeUnitText(adjustedLeft);
            this.myTextCache.put(adjustedLeft, result);
        }

        return result;
    }
}
```

Classe CachingTextFormatter

As classes `WeekTextFormatter`, `QuarterTextFormatter`, `MonthTextFormatter` e `YearTextFormatter` são as classes mencionadas anteriormente que estendem a classe abstrata.

```
protected TimeUnitText[] createTimeUnitText(Date startDate) {
    this.myCalendar.setTime(startDate);
    return new TimeUnitText[]{this.createTopText(), this.createBottomText()};
}
```

Implementação de WeekTextFormatter

```
protected TimeUnitText[] createTimeUnitText(Date startDate) {
    this.myCalendar.setTime(startDate);
    int month = this.myCalendar.get(2);
    int quarter = month / 4 + 1;
    String shortText = "Q" + quarter;
    return new TimeUnitText[]{new TimeUnitText(shortText)};
}
```

Implementação de QuarterTextFormatter

```
protected TimeUnitText[] createTimeUnitText(Date adjustedLeft) {
    String longText = this.myLongFormat.format(adjustedLeft);
    String mediumText = this.myMediumFormat.format(adjustedLeft);
    String shortText = this.myShortFormat.format(adjustedLeft);
    TimeUnitText result = new TimeUnitText(longText, mediumText, shortText);
    return new TimeUnitText[]{result};
}
```

Implementação de MonthTextFormatter

```
protected TimeUnitText[] createTimeUnitText(Date startDate) {
    this.myCalendar.setTime(startDate);
    String shortText = MessageFormat.format("pattern: \"{0,date,yyyy}\"", this.myCalendar.getTime());
    return new TimeUnitText[]{new TimeUnitText(shortText)};
}
```

Implementação de YearTextFormatter

Revisto por:

Bruno David

Vasco Malta

- **Memento**

Package: net.sourceforge.ganttproject.undo (ganttproject.src.main.java)

A classe UndoableEditImpl implementa o Memento Pattern uma vez que guarda duas versões de um objeto Document (myDocumentBefore e myDocumentAfter), permitindo, assim, executar operação de “redo” e “undo”.

```
3 usages  dbarashev +3
class UndoableEditImpl extends AbstractUndoableEdit {
    2 usages
    private final String myPresentationName;

    3 usages
    private final Document myDocumentBefore;

    3 usages
    private final Document myDocumentAfter;

    4 usages
    private final UndoManagerImpl myManager;

    6 usages
    private ProjectDatabaseTxn projectDatabaseTxn = null;

    1 usage  dbarashev +2
    UndoableEditImpl(String localizedName, Runnable editImpl, UndoManagerImpl manager) throws IOException {
        myManager = manager;
        myPresentationName = localizedName;
        myDocumentBefore = saveFile();

        try {
            projectDatabaseTxn = myManager.getProjectDatabase().startTransaction(localizedName);
            editImpl.run();
            projectDatabaseTxn.commit();
        } catch (ProjectDatabaseException ex) {
            GPLogger.log(ex);
        }
    }
}
```

Implementação de UndoableEditImpl onde é possível observar que são guardadas diferentes versões de um objeto

```

dbarashev +2
@Override
public void redo() throws CannotRedoException {
    try {
        restoreDocument(myDocumentAfter);
        if (projectDatabaseTxn != null) {
            try {
                projectDatabaseTxn.redo();
            } catch (ProjectDatabaseException e) {
                GPLLogger.log(e);
            }
        }
    } catch (DocumentException | IOException e) {
        undoRedoExceptionHandler(e);
    }
}

```

Método de redo

```

dbarashev +2
@Override
public void undo() throws CannotUndoException {
    try {
        restoreDocument(myDocumentBefore);
        if (projectDatabaseTxn != null) {
            try {
                projectDatabaseTxn.undo();
            } catch (ProjectDatabaseException e) {
                GPLLogger.log(e);
            }
        }
    } catch (DocumentException | IOException e) {
        undoRedoExceptionHandler(e);
    }
}

```

Método de undo

Revisto por:
Rafael Mira

- **Proxy**

Package: net.sourceforge.ganttproject.document (ganttproject.src.main.java)

Tal como o próprio nome indica, a classe ProxyDocument demonstra o design pattern Proxy. Esta implementa a mesma interface (Document) que a classe do objeto real (myPhysicalDocument) que contém e sobre a qual delega algum dos pedidos.

```
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;
    myCreator = creator;
    myTaskVisibleFields = taskVisibleFields;
    myResourceVisibleFields = resourceVisibleFields;
}
```

Construtor da classe ProxyDocument

```
3 usages  Dmitry Barashev
@Override
public void releaseLock() { myPhysicalDocument.releaseLock(); }
```

Exemplo de um método que demonstra a delegação de pedidos para o objeto original

Revisto por:

Gonçalo Cerveira

Gonçalo Cerveira

- **Abstract Factory**

Package:

biz.ganttproject.core/src/main/java/biz/ganttproject/core/chart/scene/AbstractSceneBuilder.java

```
public class DayGridSceneBuilder extends AbstractSceneBuilder {  
    2 usages  
    private final BooleanOption myRedlineOption;  
    2 usages
```

```
public class TimelineSceneBuilder extends AbstractSceneBuilder {  
    5 usages  
    private Canvas myTimelineContainer;  
    11 usages
```

```
public class BottomUnitSceneBuilder extends AbstractSceneBuilder {  
    1 usage  
    dbarashev
```

As classes **DayGrindSceneBuilder**, **TimeLineSceneBuilder** e **BottomUnitSceneBuilder** funcionam através de uma superclasse, esta sendo a **AbstractSceneBuilder**, que assim arranja uma boa forma de criar objetos sem ter especificar as suas classes.

```
public abstract class AbstractSceneBuilder implements SceneBuilder {  
    3 usages  
    private final Canvas myCanvas;  
    2 usages
```

Revisto por:

Mafalda Batalha

● State Pattern

Package:

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

A classe **OperationStatus** trata-se de um padrão que muda o comportamento de acordo com o estado.

```
/**
 * Status object which contains status of the operation.
 * If operation was successful it holds result returned by operation.
 * Otherwise it contains message describing result of the operation.
 *
 * @param <Code> type of result code
 * @param <T> type of result returned by operation
 * @author gkalabin@bardsoftware.com
 */
13 usages Dmitry Barashev
public class OperationStatus<T, Code extends Enum> {
    6 usages Dmitry Barashev
    public enum DefaultCode {
        OK, FAIL
    }
    2 usages
```

Revisto por:

Rafael Mira

● Facade

Package:

ganttproject/src/main/java/net/sourceforge/ganttproject/task/TaskActivity.java

```
2 usages Dmitry Barashev +1
public class TaskActivityPart implements TaskActivity {
```

```
1 usage dbarashev +1
class TaskActivityImpl implements TaskActivity {
```

As classes **TaskActivityPart** e **TaskActivityImpl** implementam diretamente na interface **TaskActivity**, ajudando o código estruturalmente, fazendo com que desse para esconder a complexidade da interface.

```
dbarashev
public interface TaskActivity extends BarChartActivity<Task> {
    dbarashev
```

Revisto por: Rafael Mira