

Departamento de Informática

**Engenharia de Software - 2022/2023**

1º Semestre

**Projeto – Fase 2**

**Realizado por:**

Afonso Ribeiro, 59895

Catarina Crespo, 59810

Danny Fernandes, 61162

Guilherme Figueira, 60288

Luana Gonçalves, 60294

**Link do repositório:** <https://github.com/grfigueira/ganttproject>

**Link do video demo:** https://youtu.be/E780rF1H2OQ

Índice

[**Code Smells**](#_x3u5lrxc9p5t) **4**

[[1] Switch statements & repeated code](#_7pssx8k3v5pp) 4

[[2] Dead Code](#_yzl7fp4hehu2) 5

[[3] Code duplication](#_555m1a631lr0) 5

[[4] Long Method](#_lbmzx78uokts) 7

[[5] Dead Code](#_fno380q4tzpa) 7

[[6] Data Class](#_69yf8dkevucu) 8

[[7] Duplicated code](#_hkz0qronvurh) 9

[[8] Dead code](#_vkjna5v06p7s) 10

[[9] Message Chains](#_6iusmxfm417v) 11

[[10] Long Method & duplicated code](#_be44cyvcqap9) 11

[[11] Dead code](#_x30gcacyrady) 12

[[12] No coments](#_xqvnpql5y035) 14

[[13] Dead code](#_xtnc90e09x85) 14

[[14] Long Parameter List](#_otgo1555vk7h) 15

[[15] Long Method](#_j47xtxq9wc1b) 15

[**Design Patterns**](#_ep4ubvppoam3) **17**

[[1] Builders](#_f7nun6gf78d8) 17

[[2] Singleton](#_7p5au3ftctcw) 18

[[3] Decorator](#_o25r3skeq0gt) 19

[[4] Adapter Pattern](#_bqnqdkps35ai) 19

[[5] Abstract Factory Pattern](#_6ai742hsaq0m) 20

[[6] Memento Pattern](#_7d0ecy3rp8ei) 21

[[7] Iterator](#_m3g35eqja5g3) 21

[[8] Facade](#_xtwxeosnxmjt) 22

[[9] Observer](#_53qzpv1iae1j) 23

[[10] Factory Method](#_hst0gef9ke51) 24

[[11] Iterator](#_wj86n0deb6dd) 24

[[12] Adapter](#_hb8hgztzs3g5) 25

[[13] Iterator](#_txfrp7f2gp1j) 26

[[14] Adapter](#_2febfkwtnvg) 26

[[15] Iterator](#_xcjhzjtov0k) 27

[**GanttProject UseCases**](#_3w7kx7n8l8nh) **28**

[**Task view**](#_xyaz7aad4egf) **28**

[**Recursos**](#_rg0t71s197sx) **29**

[**Settings**](#_vzp9q4lxpdkh) **30**

[**Toolbar Use Case Diagrams**](#_i0ogohtx1mh5) **32**

[**Toolbar Use Case Descriptions**](#_p0qwyt19ti14) **33**

[Toolbar - Project Tab](#_30j0zll) 33

[Toolbar - Edit Tab](#_afqm45iqpoc) 34

[Toolbar - View Tab](#_cgim2vtkzd2f) 34

[Toolbar - Task Tab](#_a2dl5uqmtz4u) 35

[Toolbar - Resources Tab](#_yelsgocddsec) 35

[Toolbar - Help Tab](#_988vsm5u0x2g) 36

[**Save**](#_ms3zjuuvcpoe) **37**

[**Additional Features Use cases**](#_tt67cljzdp4m) **38**

[Feature One:](#_ydt7g8au02n) 38

[Feature Two:](#_wm22ot7sqma9) 39

[**Dependency metrics**](#_cs1g7io8tdp6) **40**

[**Glossário:**](#_hoy9ojetx6yn) **40**

[**Análise das dependências**](#_w5z6yfo5lqlg) **40**

[Classes](#_f7aptp5eqo8t) 40

[Representações Gráficas da Classe GanttProject](#_45h45meyk14o) 43

[Comments:](#_txyf137cwvdq) 47

[**Complexity Metrics**](#_d91vb2rtz9qa) **48**

[Method Metrics](#_q4htdmz3je4m) 48

[Comments](#_7b11iv55q89r) 49

[Complexity Metric Distributions](#_5optoy2z7lqx) 50

[**Line of code metrics**](#_r1wir3rrd9f8) **52**

[**Glossário:**](#_wa7lh8n4eu29) **52**

[**Análise das dependências**](#_3m5t6qvuv9ky) **52**

[Classes](#_9ctii27rxy87) 52

[Comments:](#_z41gr7dyy7z4) 53

[Representações Gráficas da Classe GanttProject](#_pirqvqkamt7x) 53

[**Métricas Chidamber and Kemerer**](#_jczp2q64xly7) **55**

[**Glossário**](#_bip9khda3xlj) **55**

[**Métricas Chidamber and Kemerer - Valores mais relevantes**](#_4ns5ps5tzgp2) **55**

[Coupling Between Objects (CBD)](#_grhud1y17s0w) 55

[Depth of Inheritance Tree (DIT)](#_jl00d1ekgyt7) 56

[Lack of Cohesion Of Methods (LCOM)](#_9p3b4thbyik3) 56

[Number Of Children (NOF)](#_iwd7qx7zz6il) 56

[Response For Class (RFC)](#_7jsbo3bj8o7r) 56

[Weighted Method Complexity (WMC)](#_qlz4vberyiln) 57

[**Comentários (Métricas Chidamber and Kemerer)**](#_h4yq3cvh9rj5) **57**

[**Distribuições das Métricas Chidamber and Kemerer**](#_r3qre7bd0qys) **58**

[**Martin Packaging metrics**](#_no6eprgbk4hr) **60**

[**Glossário:**](#_gmeh64d2gslz) **60**

[**Análise:**](#_tu08tyva3rjr) **60**

[Packages](#_d2vkm8facn0w) 60

[Comments:](#_2xso4f8cfg4u) 61

[Distribuição das Métricas Martin Packaging](#_yjz05dtic7q6) 63

Fase 1 - Relatório

# 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 statment* e código repetido.

Snippet do Código:



## **[2] Dead Code**

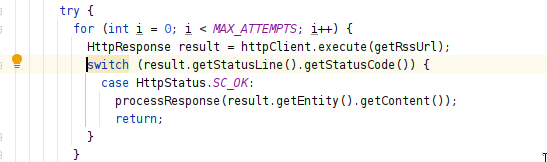
Autor: Afonso Ribeiro  
Review: Danny Fernandes

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] Code duplication**

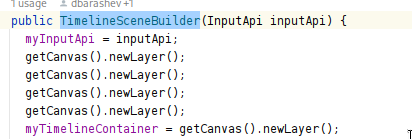
Autor: Afonso Ribeiro  
Review: Guilherme Figueira

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:



## **[4] Long Method**

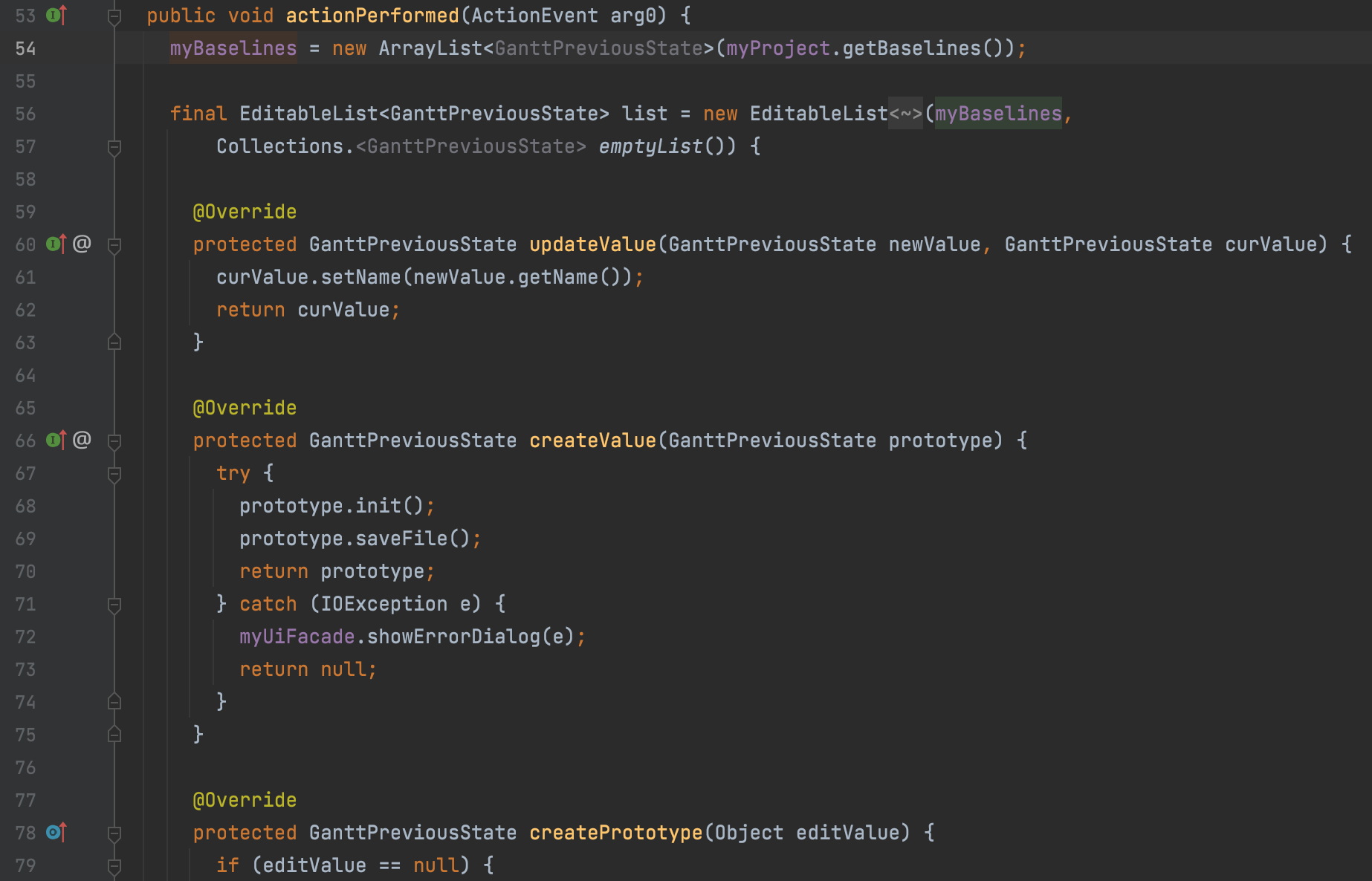
Autor: Guilherme Figueira  
Review: Afonso Ribeiro

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:





## **[5] Dead Code**

Autor: Guilherme Figueira

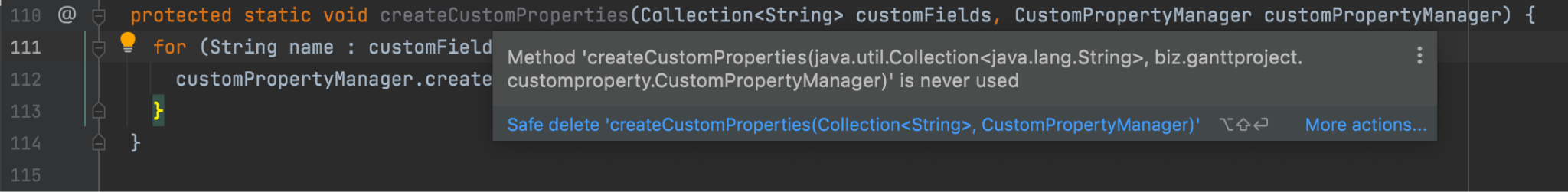
Review: Luana Gonçalves

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:



## **[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:



## **[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:





## 

## 

## **[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: Apagar todos esses métodos

Snippet do Código:



## 

## **[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:

****

## **[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*



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):

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

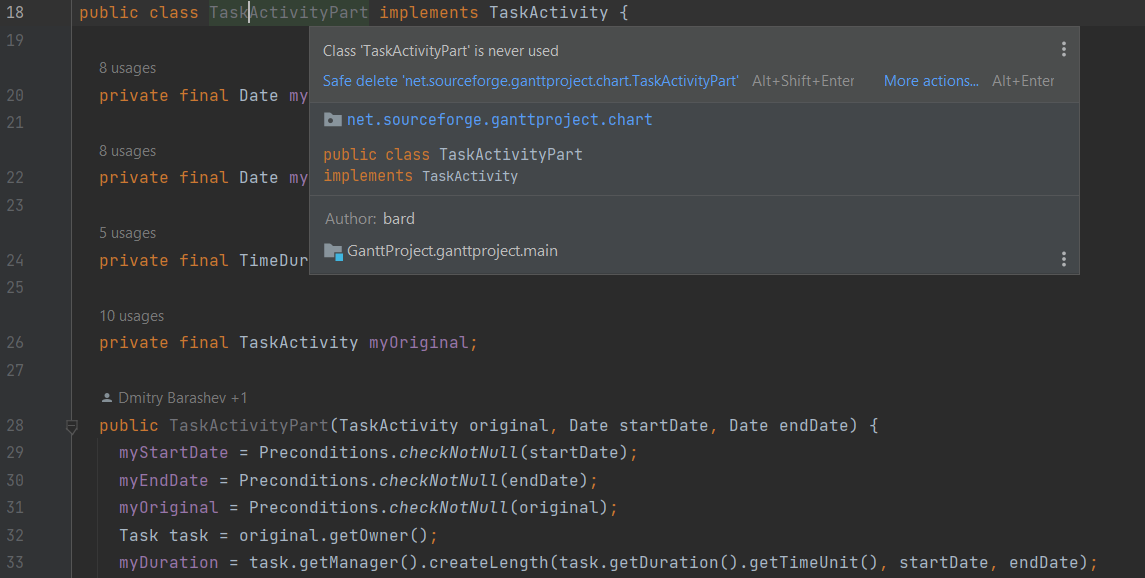
Review: Danny Fernandes

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:



## **[12] No coments**

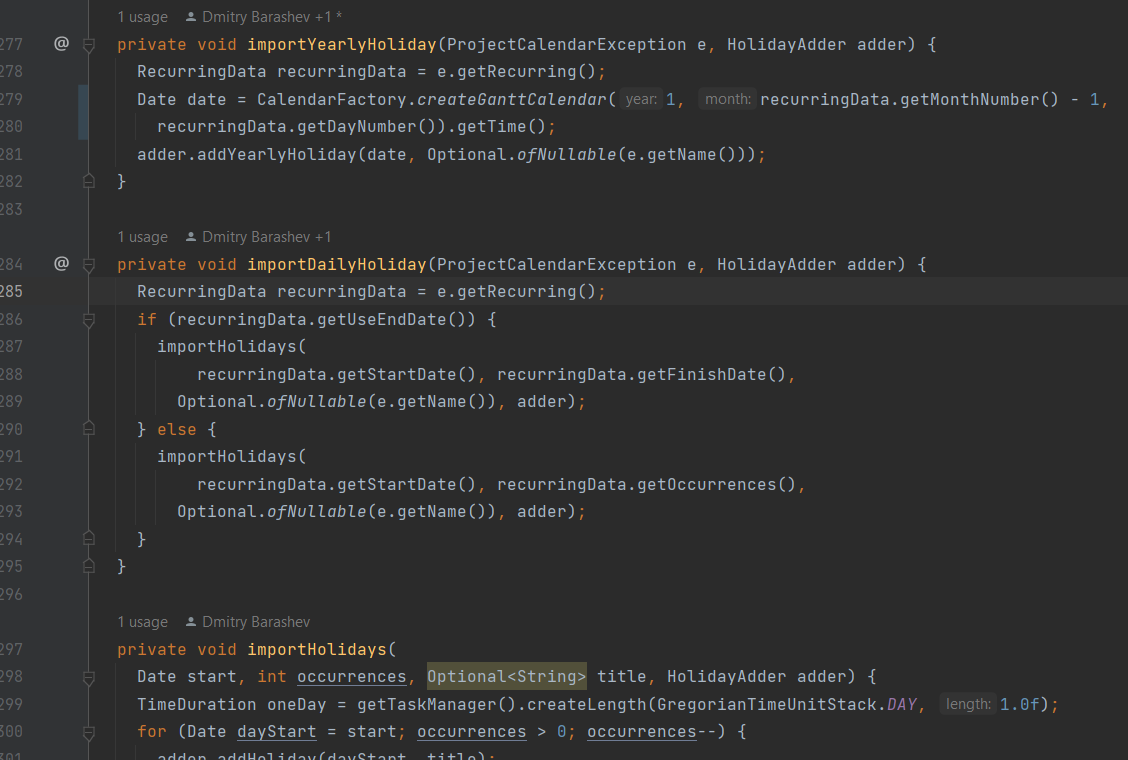
Autor: Luana Gonçalves

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:

## 

## 

## 

## 

## 

## 

## 

## 

## **[13] Dead code**

Autor: Danny Fernandes

Review: Catarina Crespo

Localização:

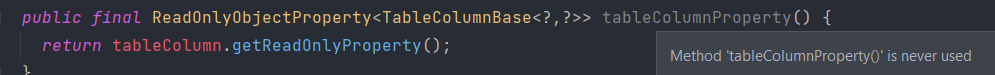
ganttproject\ganttproject\src\main\java\biz\ganttproject\lib\fx\treetable/TableColumnHeader.java

linha 345

Explicação: O método tableColumnProperty() nunca é usado.

Solução: Apagar o método.

Snippet do código:



## **[14] Long Parameter List**

Autor: Danny Fernandes

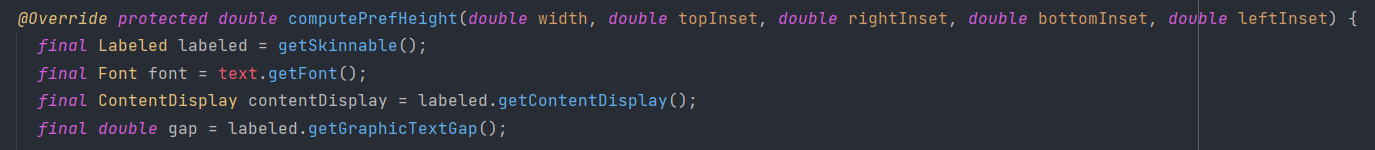
Review: Catarina Crespo

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

Explicação: O método computePreHeight() 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:



## **[15] Long Method**

Autor: Danny Fernandes

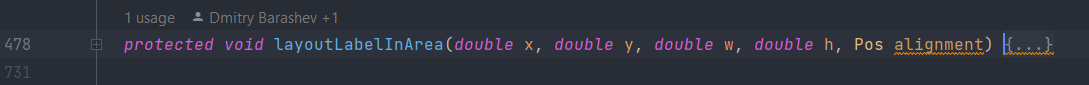
Review: Catarina Crespo

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:



# Design Patterns

## **[1] Builders**

Autor: Afonso Ribeiro

Review: Luana Gonçalves

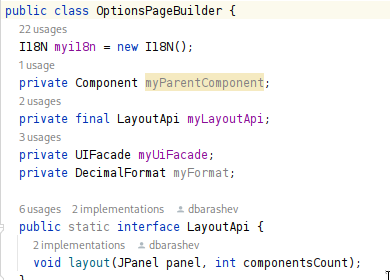
Localização: OptionsPage Builder in ganttproject/src/main/java/net/sourceforge/ganttproject/gui/options/OptionsPageBuilder.java

e

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

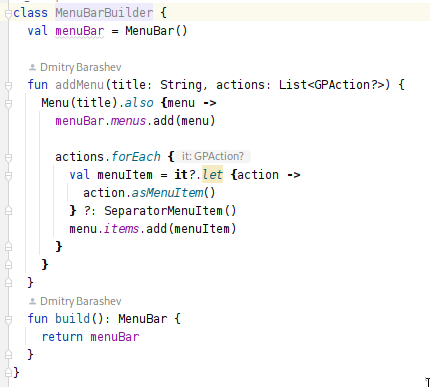
Explicação**:** Esta classe constrói objetos do tipo JComponent.

Snippet do código:



Explicação**:** Esta classe constrói objetos do tipo MenuBar.

Snippet do código:



## **[2] Singleton**

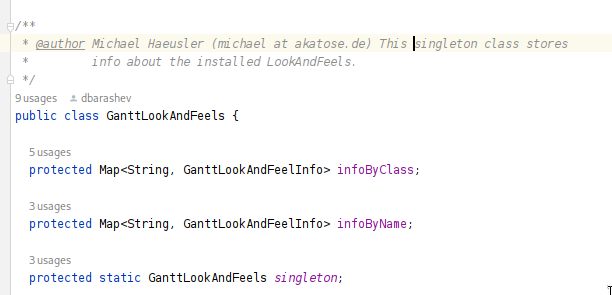
Autor: Afonso Ribeiro

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:



## **[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:



## **[4] Adapter Pattern**

Autor: Catarina Crespo

Review: Afonso Ribeiro

Localização: *ganttproject/ganttproject/src/main/java/biz.ganttproject/storage/webdav/WebdavBrowserPane.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:



## **[5] Abstract Factory Pattern**

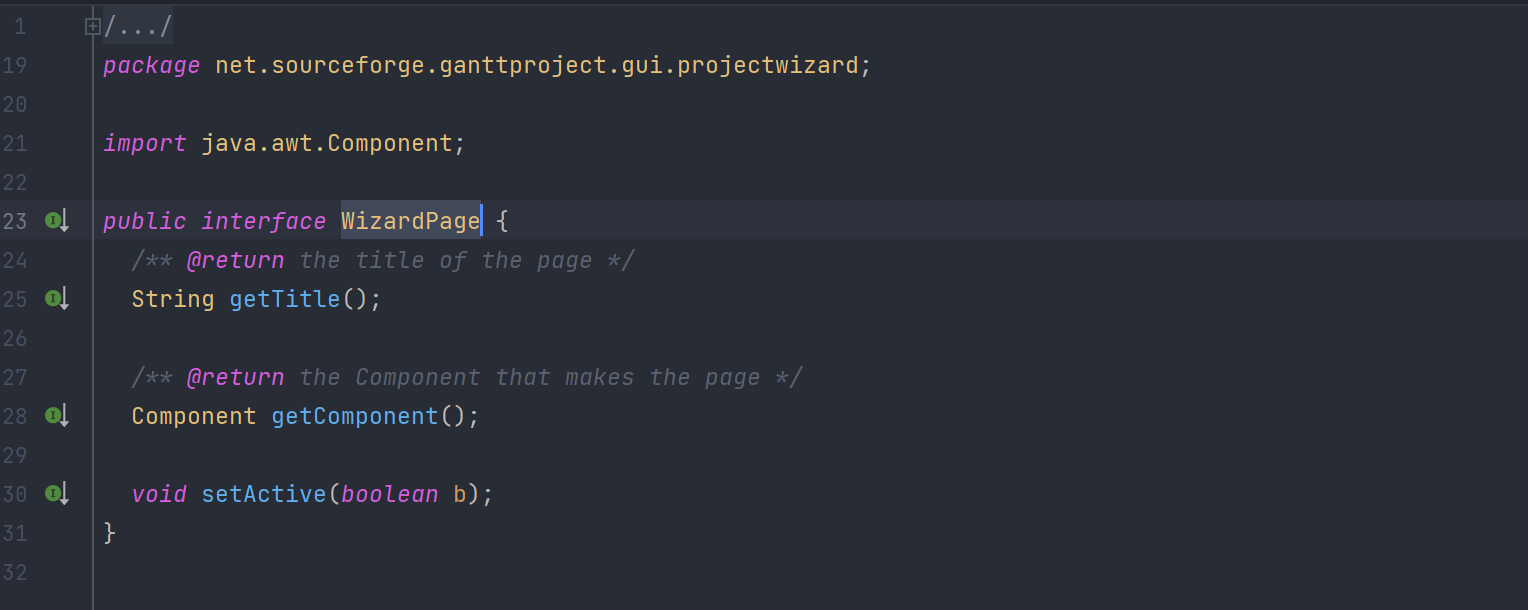
Autor: Catarina Crespo

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:

## **[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:



## **[7] Iterator**

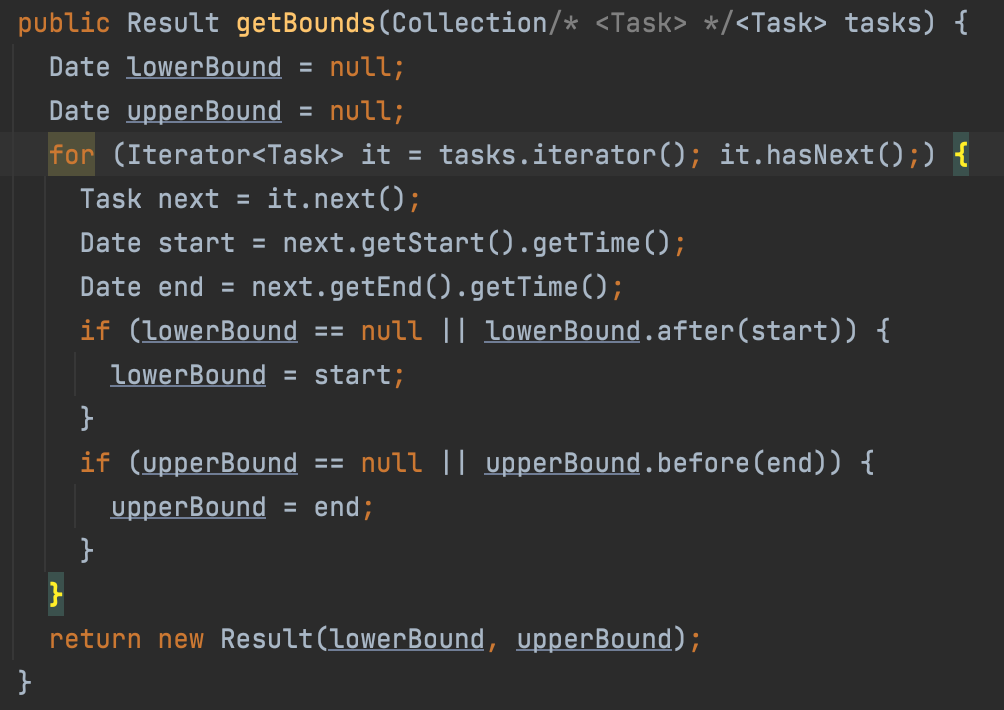
Autor: Guilherme Figueira

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:



## **[8] Facade**

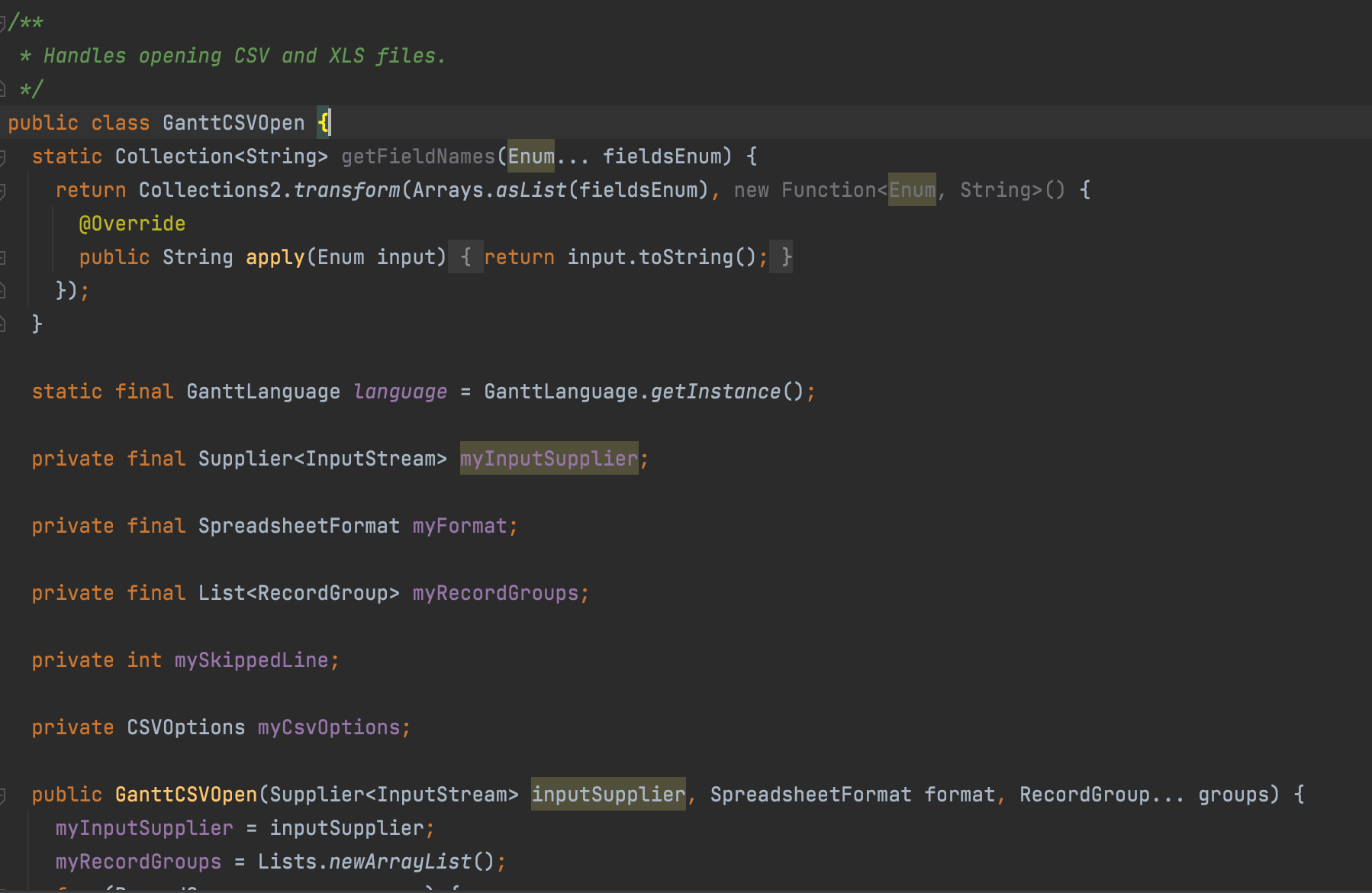
Autor: Guilherme Figueira

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:



## **[9] Observer**

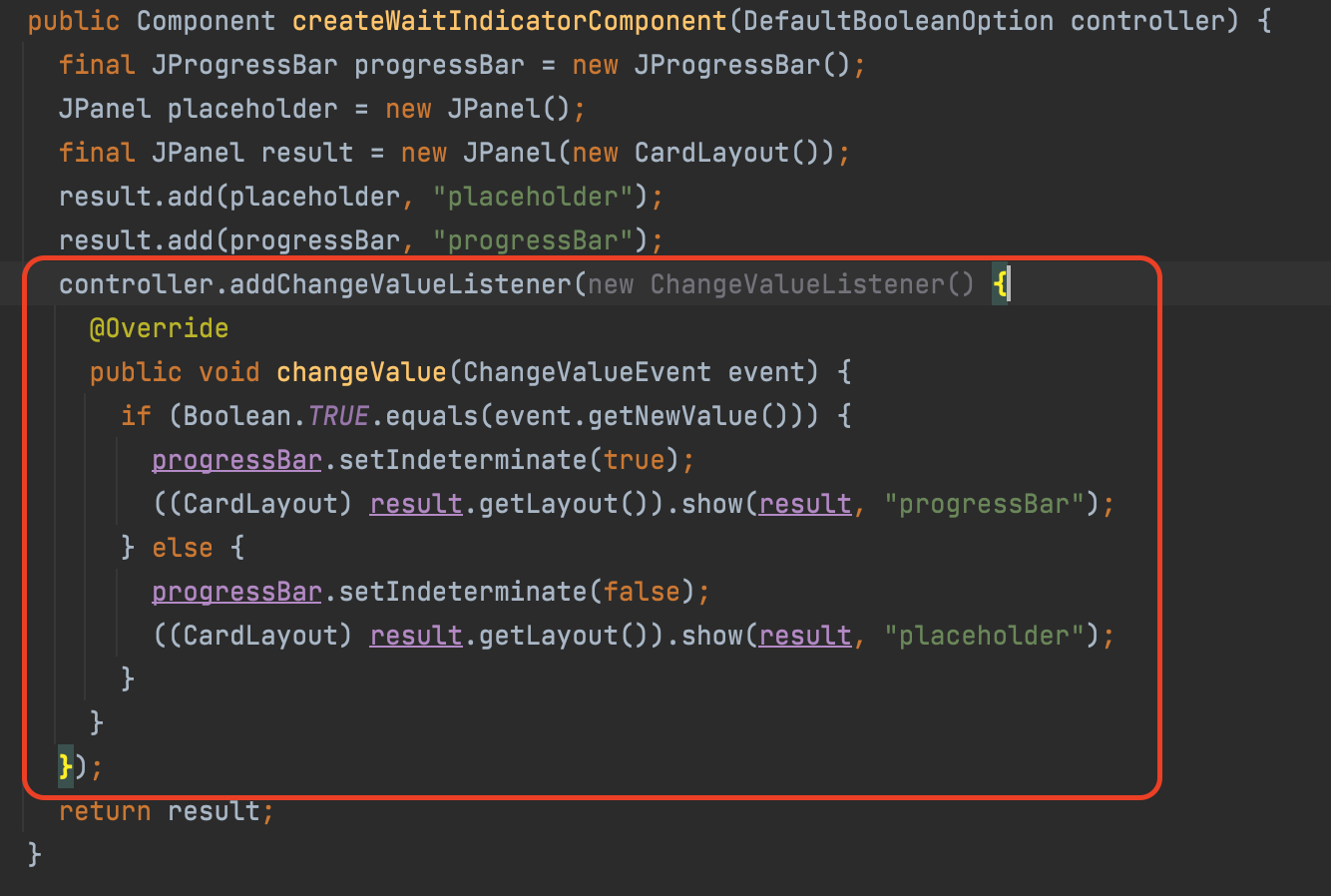
Autor: Guilherme Figueira

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:



## **[10] Factory Method**

## 

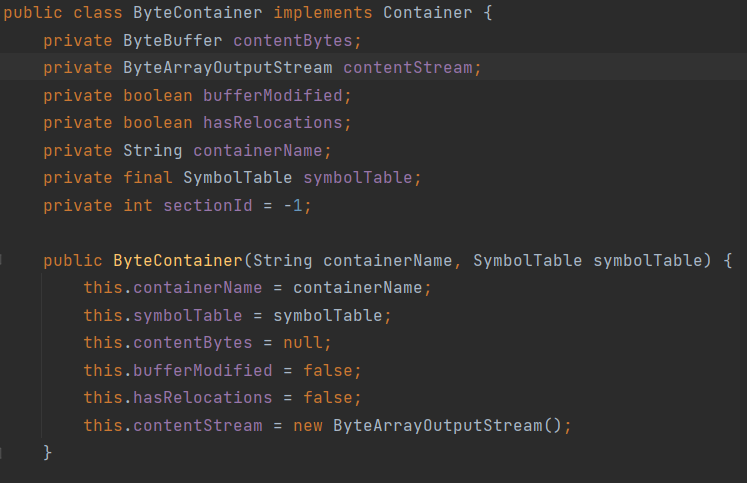
Autor: Luana Gonçalves

Review: Danny Fernandes

Localização:jdk-15!\jdk.aot\jdk\tools\jaotc\binformat\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:



## **[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 *InstructorHandle*, sem expor a sua representação subjacente.

Snippet do Código:



## 

## 

## **[12] Adapter**

## 

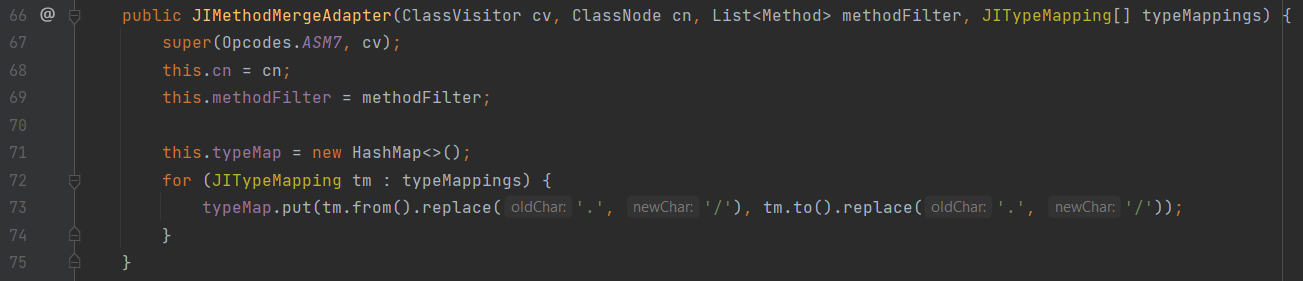
Autor: Luana Gonçalves

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:

****

## **[13] Iterator**

## 

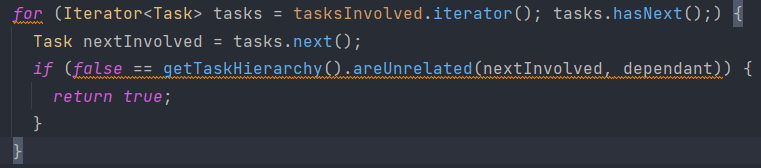
Autor: Danny Fernandes

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:



## **[14] Adapter**

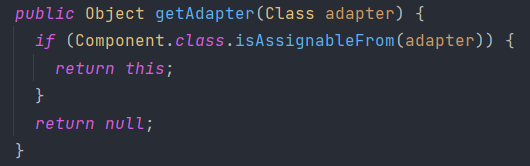
## 

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:



## **[15] Iterator**

## 

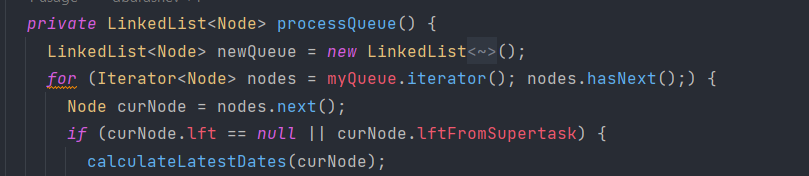
Autor: Danny Fernandes

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:



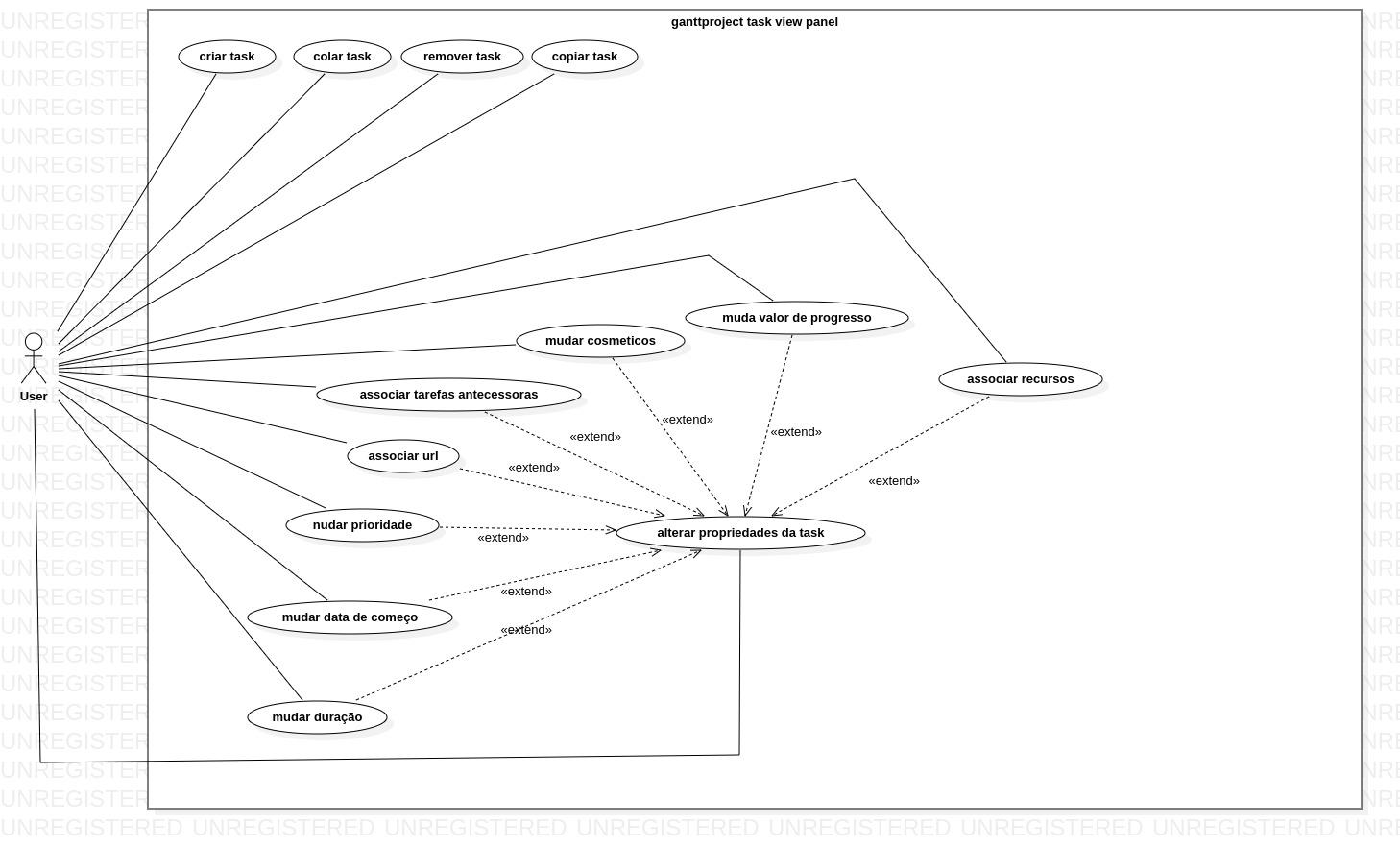
Fase 2 - Relatório

# GanttProject UseCases

## Task view

*Autor: Afonso*

*Review por: Luana*

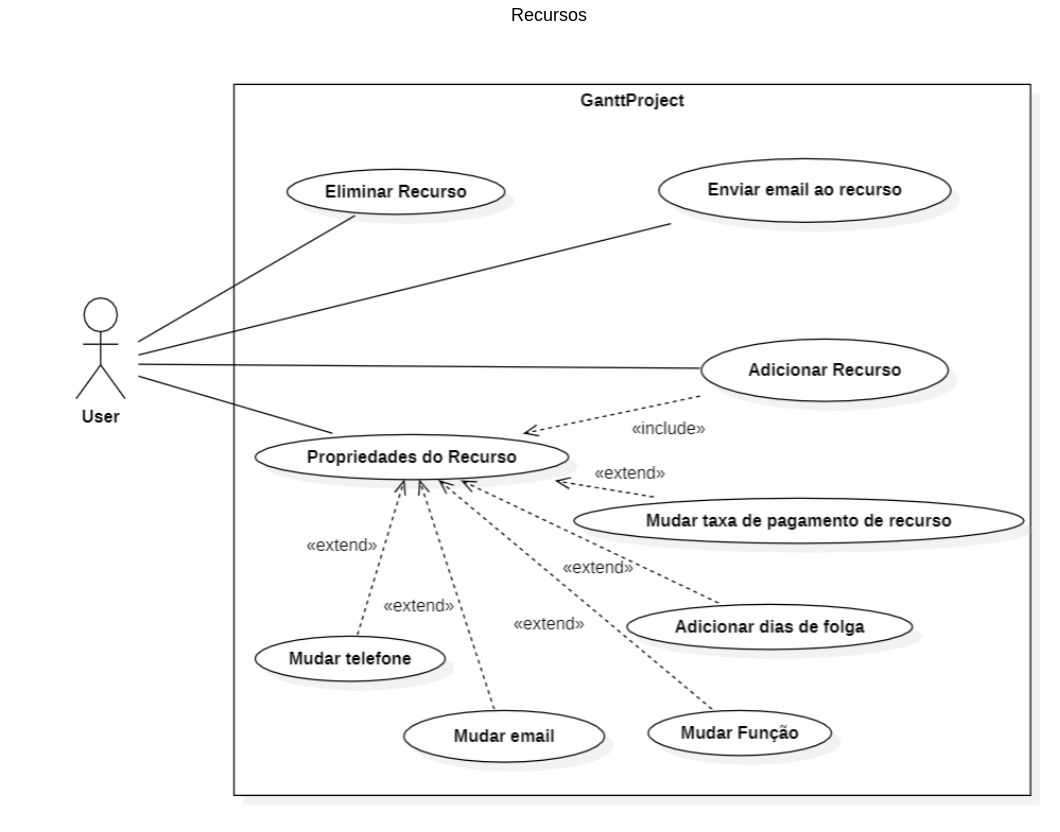


| **Use Case: Tasks** |
| --- |
| *Brief description: the user creates tasks and manages them according to his needs*.He can also copy and paste the task he as already created and delete them as well. All the task will be displayed in a timeline making it more accessible for the User to understand |
| *Primary actors: User* |
| *Secondary actors: none* |
| *Main flow:*  1. create task  2. change task properties to the desired needs  3. link them to other tasks |
| *Postconditions:* task is created |

## Recursos

*Autor: Luana*

*Review por: Guilherme*



# 

| **Use Case: Resources** |
| --- |
| *Brief description:* The user will be able to create resources and assign all of his properties. The resource will be able to be deleted. The User can also email a specific resource. |
| *Primary actors: User* |
| *Secondary actors: None* |
| *Postconditions:* resource is created |

## Settings

*Autor: Guilherme*

*Review por: Afonso*

# 

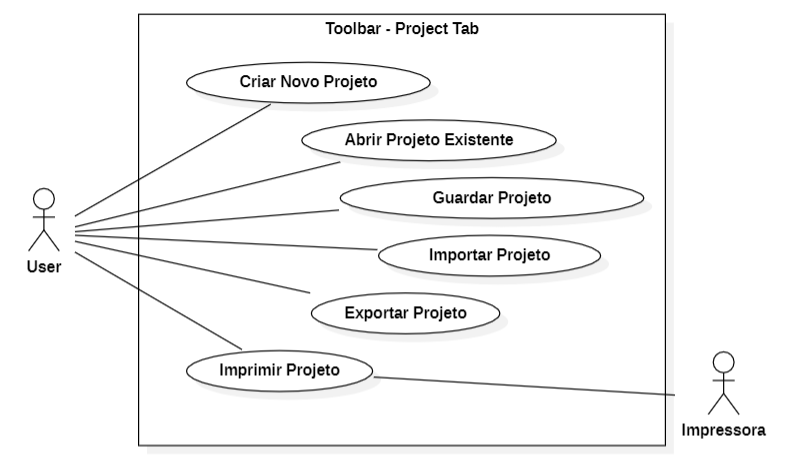
| **Use Case: Settings** |
| --- |
| *Brief description:* The user will be able to change some aspects of the program to fit their preferences. |
| *Primary actors: User* |
| *Secondary actors: None* |
| *Postconditions: The program's settings will change based on the user's actions on the settings tab.* |

# 

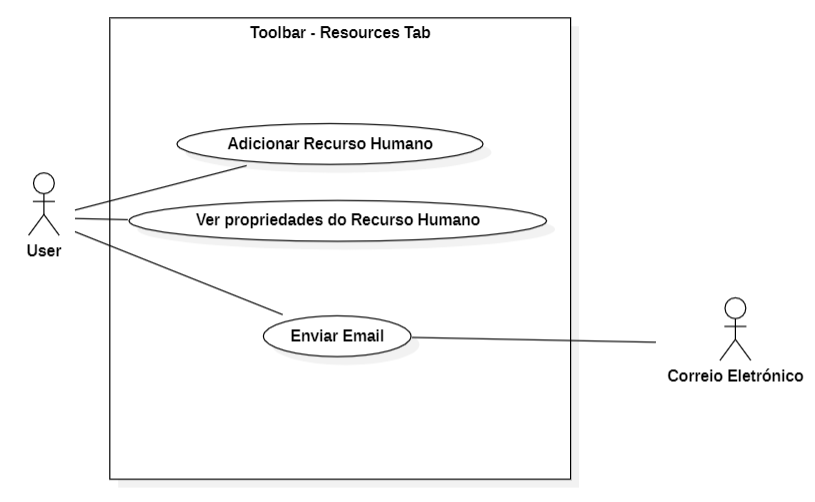
## Toolbar Use Case Diagrams

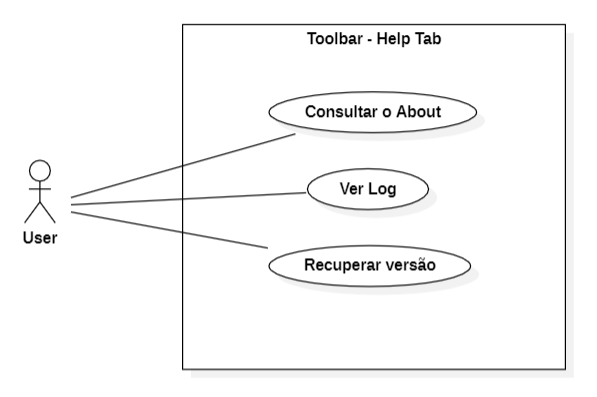
*Autor: Catarina Crespo*

*Review por: Danny*

**







## Toolbar Use Case Descriptions

### Toolbar - Project Tab

| **Use Case: Criar Novo Projeto** |
| --- |
| *Brief description:* O user cria um novo projeto ao carregar no botão do menu e especifica os detalhes do projeto, seja de identificação do mesmo (nome, organização…), seja relacionados com o funcionamento do mesmo e termina carregando em “OK” |
| *Primary actors:* User |
| *Secondary actors: None* |

| **Use Case: Abrir projeto existente** |
| --- |
| *Brief description:* O user abre um projeto previamente criado e guardado |
| *Primary actors:* User |
| *Secondary actors: None* |

| **Use Case: Guardar projeto** |
| --- |
| *Brief description:* O user guarda as definições do projeto atual/aberto |
| *Primary actors:* User |
| *Secondary actors: None* |

| **Use Case: Importar projeto** |
| --- |
| *Brief description:* O user carrega o projeto de um ficheiro externo e abre-o no GanttProject |
| *Primary actors:* User |
| *Secondary actors: None* |

| **Use Case: Exportar projeto** |
| --- |
| *Brief description:* O user exporta o projeto corrente num dos cinco formatos disponíveis |
| *Primary actors:* User |
| *Secondary actors: None* |

| **Use Case: Imprimir projeto** |
| --- |
| *Brief description:* O user manda imprimir o projeto corrente a uma impressora |
| *Primary actors:* User |
| *Secondary actors:* Impressora |

### Toolbar - Edit Tab

| **Use Case: Desfazer Ação (Undo)** |
| --- |
| *Brief description:* O user anula a última ação efetuada |
| *Primary actors:* User |
| *Secondary actors: none* |

| **Use Case: Refazer Ação (Redo)** |
| --- |
| *Brief description:* O user refaz a última ação efetuada |
| *Primary actors:* User |
| *Secondary actors: none* |

| **Use Case: Atualizar Projeto (Refresh)** |
| --- |
| *Brief description:* O user carrega para recarregar as páginas e mostrar possíveis alterações que foram efetuadas mas ainda não estavam a mostrar |
| *Primary actors:* User |
| *Secondary actors: none* |

| **Use Case: Procurar no Projeto** |
| --- |
| *Brief description:* O user insere um termo na caixa de pesquisa que é selecionada ao carregar no botão do menu da toolbar e as correspondências que forem encontradas são mostradas |
| *Primary actors:* User |
| *Secondary actors: none* |

Os use cases “Cortar Elemento”, “Copiar Elemento” e “Colar Elemento” correspondem às operações normais de, respectivamente, *cut, copy* e *paste*. Em todos só há um único ator (o utilizador) que executa essas ações.

| **Use Case: Abrir as definições** |
| --- |
| *Brief description:* O user carrega no botão para abrir o painel com as definições e tem a possibilidade de as editar nesse momento |
| *Primary actors:* User |
| *Secondary actors: none* |

### Toolbar - View Tab

| **Use Case: Ver PERT CHART** |
| --- |
| *Brief description:* Uma nova tab é aberta onde o user pode ver as propriedades das suas tasks (de forma mais gráfica) |
| *Primary actors:* User |
| *Secondary actors: none* |

### Toolbar - Task Tab

| **Use Case: Adicionar Task** |
| --- |
| *Brief description:* O user cria uma nova task com os parâmetros inseridos ou já atribuídos antes dessa chamada |
| *Primary actors:* User |
| *Secondary actors:* Correio Eletrónico |

| **Use Case: Ver propriedades da Task** |
| --- |
| *Brief description:* O user abre uma página com as properties da task e permite a edição das mesmas |
| *Primary actors:* User |
| *Secondary actors: none* |

| **Use Case: Apagar Task** |
| --- |
| *Brief description:* O user apaga a Task selecionada |
| *Primary actors:* User |
| *Secondary actors: none* |

### Toolbar - Resources Tab

| **Use Case: Adicionar Recurso Humano** |
| --- |
| *Brief description:* O user adiciona um novo recurso humano ao projeto |
| *Primary actors:* User |
| *Secondary actors: none* |

| **Use Case: Ver propriedas do Recurso Humano** |
| --- |
| *Brief description:* O user abre uma janela com a informação do recurso humano selecionado e tem a possibilidade de editar essas informações |
| *Primary actors:* User |
| *Secondary actors: none* |

| **Use Case: Enviar email** |
| --- |
| *Brief description:* Ao carregar no botão para enviar email, um sistema de correio eletrónico torna-se responsável pelo envio dessa mensagem |
| *Primary actors:* User |
| *Secondary actors:* Correio Eletrónica |

### Toolbar - Help Tab

| **Use Case: Consultar o About** |
| --- |
| *Brief description:* O user é levado para uma página onde tem a informação da aplicação para nós podermos consultar |
| *Primary actors:* User |
| *Secondary actors: none* |

| **Use Case: Ver log** |
| --- |
| *Brief description:* o user consulta os valores dos logs antigos |
| *Primary actors:* User |
| *Secondary actors: none* |

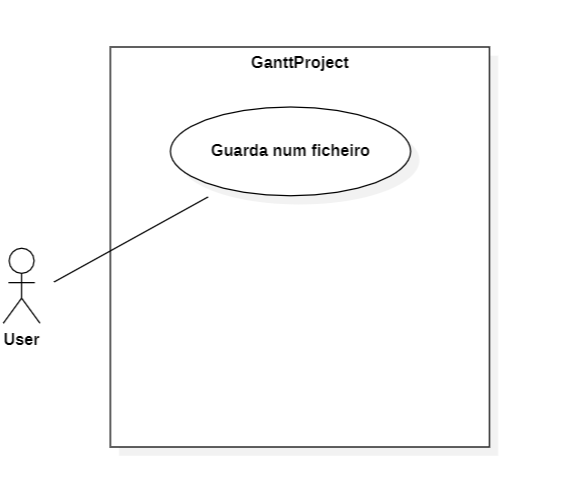
| **Use Case: Recover** |
| --- |
| *Brief description:* Ao carregar no botão, o GanttProject apresenta uma mensagem a perguntar se quer utilizar o último atosaved state |
| *Primary actors:* User |
| *Secondary actors: none* |

# 

## Save

*Autor: Danny*

*Review por: Catarina*



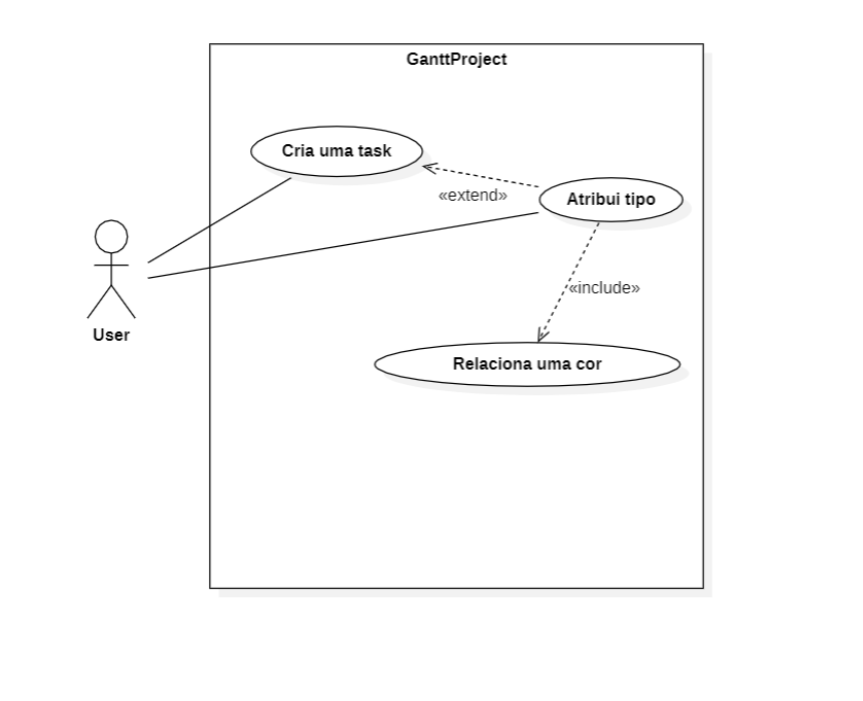
| **Use Case: Guarda num ficheiro** |
| --- |
| *Brief description:* Na primeira vez o user deve escolher onde o ficheiro vai ser guardado.  Nas próximas vezes o ficheiro só é atualizado ao clicar em Guardar. |
| *Primary actors:* User |
| *Secondary actors: none* |

# Additional Features Use cases

## Feature One:

*Autor: Luana*

*Review por: Guilherme*

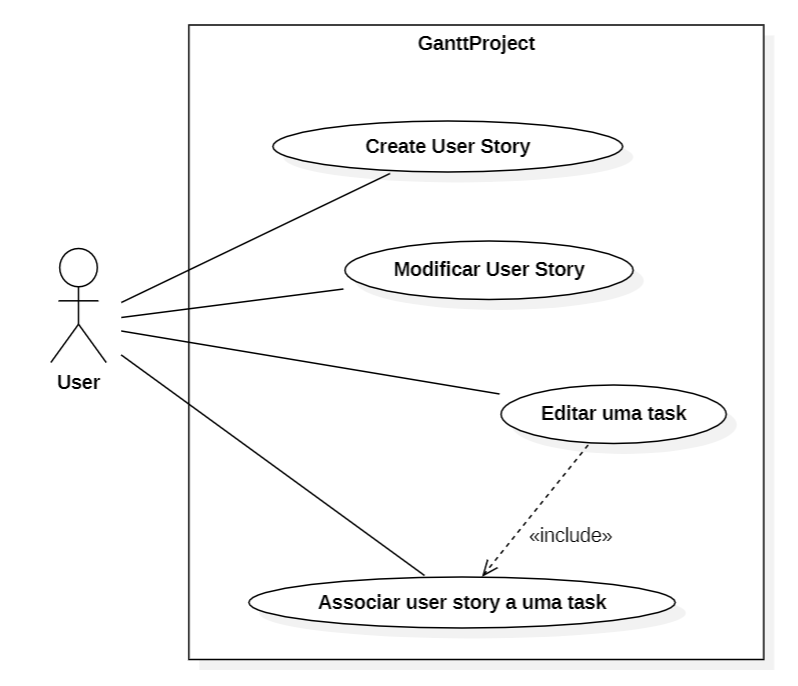


| **Use Case: Associate Task Types with tasks** |
| --- |
| *Brief description: The user will be able to select from a few task types (Planning, Research, Design, Implementation, Follow up, Other), each of them with a unique color association, to set the task type of each task.* |
| *Primary actors: The user* |
| *Secondary actors: None* |

## Feature Two:

*Autor: Luana*

*Review por: Catarina*



| **Use Case: Create User Story** |
| --- |
| *Brief description:*  Create a new User Story and associate them to a task |
| *Primary actors:* Project Manager/User |
| *Secondary actors:*  None |

# 

# Dependency metrics

*Autor: Afonso*

*Review por: Danny*

## Glossário:

* **Dependências cíclicas** - São relações entre duas ou mais classes tal que essas classes dependem, diretamente ou indiretamente uma da outra mutuamente.
* **dependência direta** -classe que é diretamente uma dependência de outra.
* dependência transitiva - Classe diretamente ou indiretamente que é dependência de uma dada classe.
* **dependente direto**- classe que depende diretamente de uma ou mais classes.
* **dependente transitivo** - classe que depende diretamente ou indiretamente de uma ou mais classes.

## Análise das dependências

## Classes

Top 3 dependencias ciclicas:

| Class | cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| --- | --- | --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.AbstractChartImplementation | 513.0 | 34.0 | 864.0 | 8.0 | 649.0 | 9.0 | 3.0 |
| net.sourceforge.ganttproject.AbstractChartImplementation.MouseHoverLayerUi | 513.0 | 8.0 | 864.0 | 1.0 | 649.0 | 6.0 | 1.0 |
| net.sourceforge.ganttproject.CalendarEventAction | 513.0 | 7.0 | 864.0 | 1.0 | 649.0 | 5.0 | 1.0 |

Top 3 dependências diretas:

| Class | cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| --- | --- | --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.GanttProject | 513.0 | 100.0 | 864.0 | 29.0 | 649.0 | 29.0 | 9.0 |
| net.sourceforge.ganttproject.task.TaskManagerImpl | 513.0 | 78.0 | 864.0 | 16.0 | 649.0 | 15.0 | 5.0 |
| net.sourceforge.ganttproject.chart.ChartModelBase | 513.0 | 56.0 | 864.0 | 20.0 | 649.0 | 13.0 | 5.0 |

top 3 dependências transitivas:

| Class | cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| --- | --- | --- | --- | --- | --- | --- | --- |
| org.ganttproject.impex.htmlpdf.ExporterToPDF | 1.0 | 8.0 | 887.0 | 1.0 | 1.0 | 5.0 | 1.0 |
| org.ganttproject.impex.htmlpdf.itext.ITextEngine | 1.0 | 18.0 | 887.0 | 1.0 | 1.0 | 8.0 | 1.0 |
| org.ganttproject.impex.htmlpdf.itext.ThemeImpl | 0.0 | 33.0 | 880.0 | 1.0 | 2.0 | 14.0 | 1.0 |

top 3 dependentes diretos:

| Class | cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| --- | --- | --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.language.GanttLanguage | 513.0 | 7.0 | 864.0 | 130.0 | 649.0 | 4.0 | 37.0 |
| net.sourceforge.ganttproject.action.GPAction | 513.0 | 7.0 | 864.0 | 96.0 | 649.0 | 4.0 | 20.0 |
| net.sourceforge.ganttproject.GPLogger | 513.0 | 3.0 | 864.0 | 89.0 | 649.0 | 2.0 | 32.0 |

top 3 dependentes transitivos:

| Class | cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| --- | --- | --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.GanttProject | 513.0 | 100.0 | 864.0 | 29.0 | 649.0 | 29.0 | 9.0 |
| net.sourceforge.ganttproject.GanttProjectBase | 513.0 | 48.0 | 864.0 | 14.0 | 649.0 | 21.0 | 4.0 |
| net.sourceforge.ganttproject.GanttGraphicArea | 513.0 | 48.0 | 864.0 | 4.0 | 649.0 | 16.0 | 3.0 |

top 3 PDcy:

| Class | cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| --- | --- | --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.GanttProject | 513.0 | 100.0 | 864.0 | 29.0 | 649.0 | 29.0 | 9.0 |
| net.sourceforge.ganttproject.GanttProjectBase | 513.0 | 48.0 | 864.0 | 14.0 | 649.0 | 21.0 | 4.0 |
| net.sourceforge.ganttproject.GanttGraphicArea | 513.0 | 48.0 | 864.0 | 4.0 | 649.0 | 16.0 | 3.0 |

top 3 DPpt:

| Class | cyclic | Dcy | Dcy\* | Dpt | Dpt\* | PDcy | PDpt |
| --- | --- | --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.GanttProject | 513.0 | 100.0 | 864.0 | 29.0 | 649.0 | 29.0 | 9.0 |
| net.sourceforge.ganttproject.GanttProjectBase | 513.0 | 48.0 | 864.0 | 14.0 | 649.0 | 21.0 | 4.0 |
| net.sourceforge.ganttproject.GanttGraphicArea | 513.0 | 48.0 | 864.0 | 4.0 | 649.0 | 16.0 | 3.0 |

Dcy - número de dependências diretas

Dcy\* - número de dependências transitivas

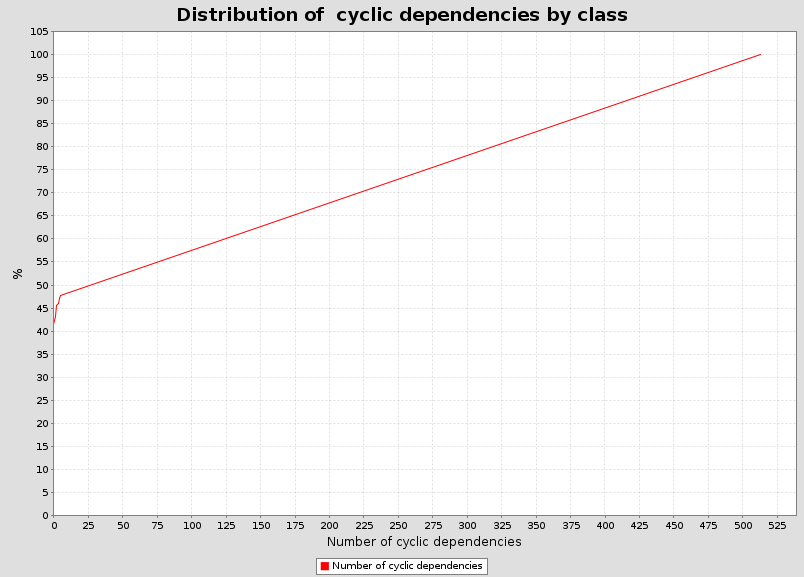
Dpt - número de dependentes diretos

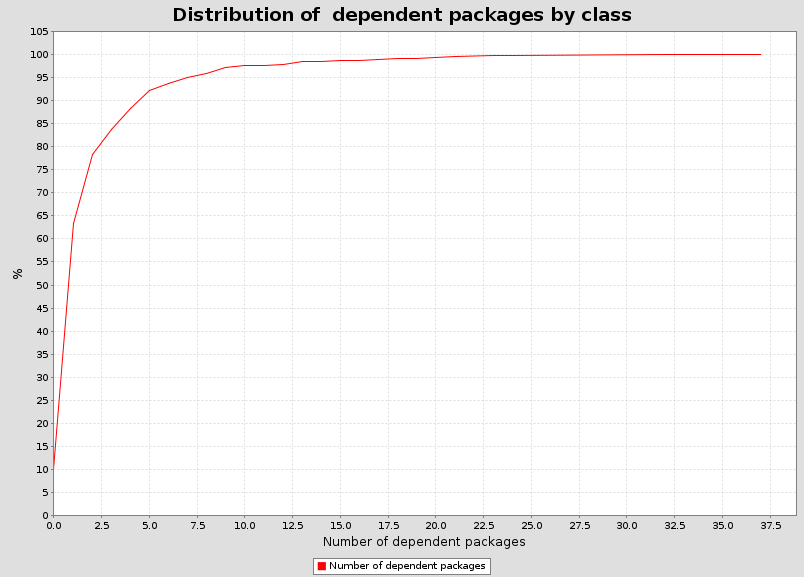
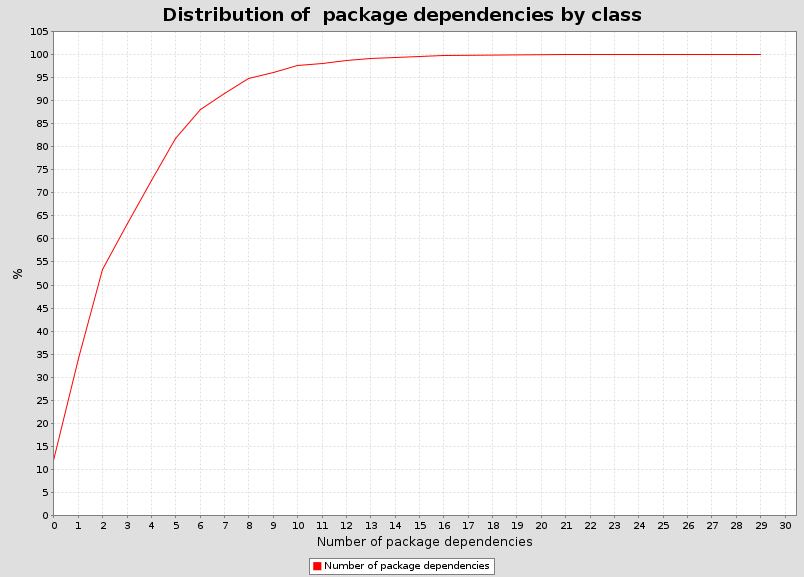
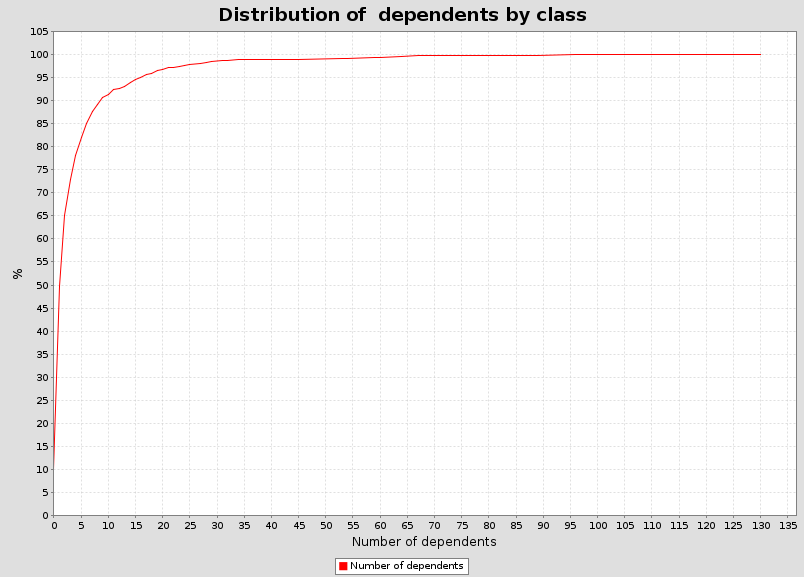
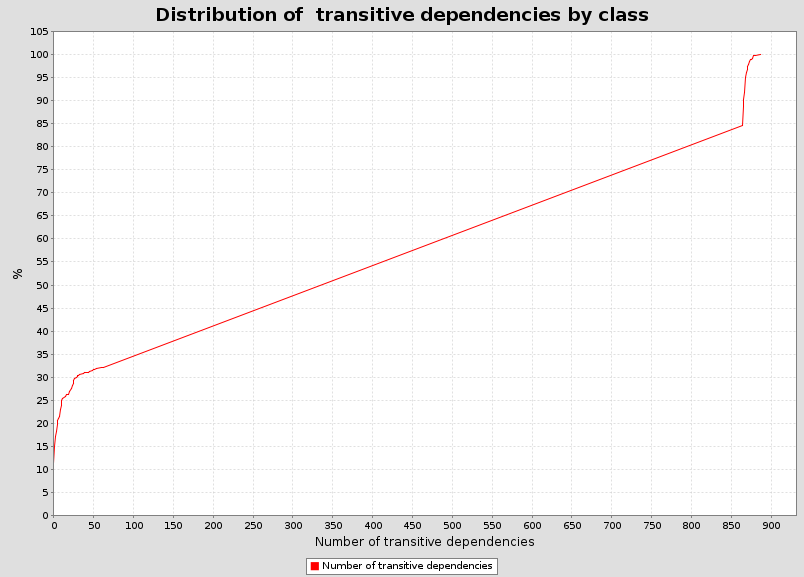
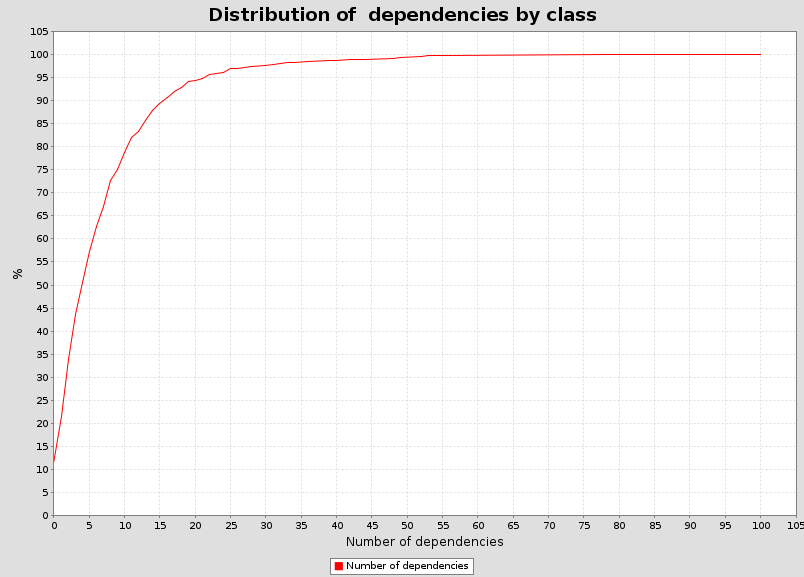
Dpt\* -número de dependentes transitivos

PDcy - número de pacotes que a classe diretamente depende

PDpt - número de pacotes que depende da classe

## Representações Gráficas da Classe GanttProject





## Comments:

Pelas tabelas, nós podemos ver muitos problemas com o programa. A classe ganttProject, que é a classe principal onde as estruturas de dados necessárias são criadas, é a classe com com o maior número de dependências diretas embora devesse ser a com maior abstrações e por consequência a com um menor número de dependências diretas.

O alto número de dependências cíclicas também é um dos sintomas do que foi mencionado acima. Como está tudo interligado e a abstração é baixa, o número de dependências cíclicas da classe de topo é muito maior do que deveria ser esperado desta classe.

# Complexity Metrics

*Autor: Guilherme*

*Review por: Luana*

## Method Metrics

Worst 3 - Cognitive Complexity

| **Method** | **CogC** |
| --- | --- |
| GanttXMLOptionsParser.startElement() | 183.0 |
| TaskProperties.getProperty() | 58.0 |
| GPTimeUnitStack.parseDuration() | 53.0 |

Worst 3 - Essential Complexity

| **Method** | **ev(G)** |
| --- | --- |
| DateParser.getCalendar() | 16.0 |
| GPTimeUnitStack.parseDuration() | 12.0 |
| TaskManagerImpl.createLength() | 12.0 |

Worst 3 - Design Complexity

| **Method** | **iv(G)** |
| --- | --- |
| GanttXMLOptionsParser.startElement() | 78.0 |
| GanttTaskPropertiesBean.applySettings() | 33.0 |
| TaskManagerImpl.newTaskBuilder() | 22.0 |

Worst 3 - Cyclomatic Complexity

| **Method** | **v(G)** |
| --- | --- |
| GanttXMLOptionsParser.startElement() | 89.0 |
| GanttTaskPropertiesBean.applySettings() | 33.0 |
| GanttTreeTableModel.getValueAt() | 28.0 |

**Cognitive Complexity** - Calculates the Cognitive Complexity of each non-abstract method. The metric is similar to Cyclomatic Complexity, but is intended to explicitly measure understandability, which can be quite different from testability. Cognitive Complexity is increased with each control structure used and is higher the more nested control structures are. Abbreviation: *CogC*

**Essential Complexity** - Calculates the Essential Complexity of each non-abstract method. Essential Complexity is a graph-theoretic measure of just how ill-structured a method's control flow is. Essential Complexity ranges from 1 to v(G), the Cyclomatic Complexity of the method. Abbreviation: *ev(G)*

**Design Complexity** - Calculates the design complexity of a method. The design complexity is related to how interlinked a method's control flow is with calls to other methods. Design complexity ranges from 1 to v(G), the cyclomatic complexity of the method. Design complexity also represents the minimal number of tests necessary to exercise the integration of the method with the methods it calls. Abbreviation: *iv(G)*

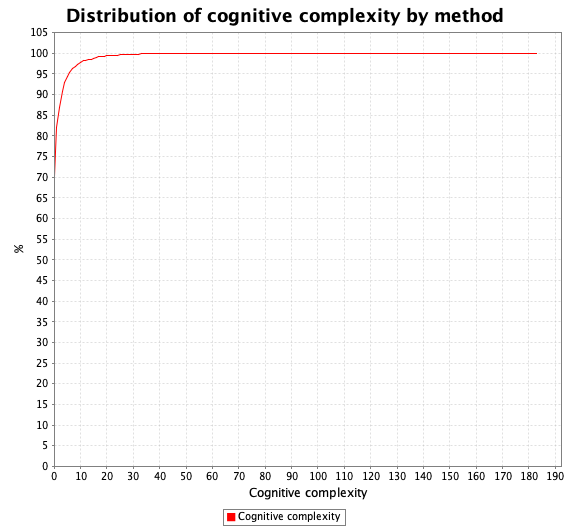
**Cyclomatic Complexity -** Calculates the Cyclomatic Complexity of each non-abstract method. Cyclomatic complexity is a measure of the number of distinct execution paths through each method. This can also be considered as the minimal number of tests necessary to completely exercise a method's control flow. In practice, this is 1 + the number of if's, while's, for's, do's, switch cases, catches, conditional expressions, &&'s and ||'s in the method. Abbreviation: v(G)

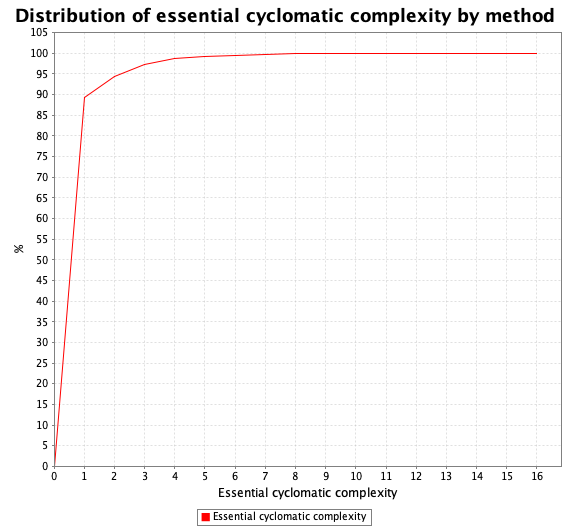
## Comments

The most significant observation we can make with this data is the problematic complexity of the method *startElement()* from the class *GanttXMLOptionsParser*. Just taking a quick look at this method shows us the problem. It has **207 lines** mostly containing *if*’s and *else if*’s. This makes testing the method a very difficult and tedious task.

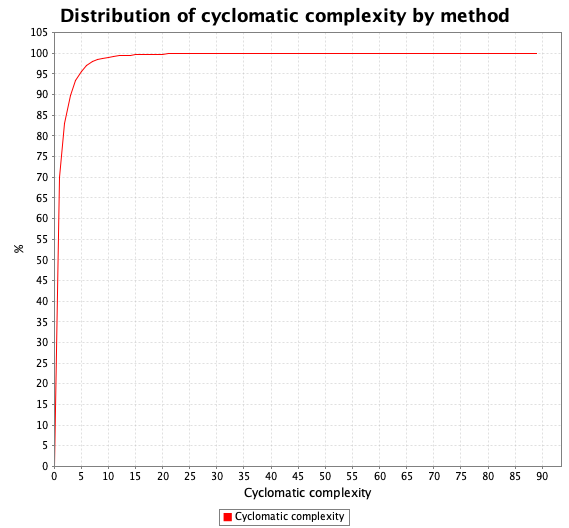
I would suggest, as usual, to divide the method in various separate methods, as this way we could test each method individually and thus more easily test the entire logic.

## Complexity Metric Distributions

****







# Line of code metrics

*Autor: Luana*

*Review por: Catarina*

## Glossário:

* (CLOC) Comment lines of code - Calcula o número de linhas de comentários em cada classe. Linhas em branco não são contadas.
* (JLOC) Javadoc lines of code - Calcula o número de linhas de comentários javadoc em cada classe. Linhas em branco não são contadas.
* (LOC) Lines of code - Calcula o número de linhas de código em cada classe. Os comentários são contados, mas os espaços em branco não.

## Análise das dependências

## Classes

Top 3 de linhas em comentário:

| Class | CLOC | JLOC | LOC |
| --- | --- | --- | --- |
| net.sourceforge.ganttproject.chart.SimpleRenderedImage | 220.0 | 207.0 | 436.0 |
| net.sourceforge.ganttproject.GanttOptions | 189.0 | 149.0 | 689.0 |
| net.sourceforge.ganttproject.language.CharSetMap | 185.0 | 173.0 | 425.0 |

Top 3 de linhas em javadoc:

| Class | CLOC | JLOC | LOC |
| --- | --- | --- | --- |
| net.sourceforge.ganttproject.chart.SimpleRenderedImage | 220.0 | 207.0 | 436.0 |
| net.sourceforge.ganttproject.language.CharSetMap | 185.0 | 173.0 | 425.0 |
| net.sourceforge.ganttproject.GanttOptions | 189.0 | 149.0 | 689.0 |

Top 3 de linhas de código:

| Class | CLOC | JLOC | LOC |
| --- | --- | --- | --- |
| net.sourceforge.ganttproject.GanttProject | 129.0 | 66.0 | 1016.0 |
| net.sourceforge.ganttproject.task.TaskManagerImpl | 10.0 | 3.0 | 765.0 |
| net.sourceforge.ganttproject.GanttOptions | 189.0 | 149.0 | 689.0 |

## Comments:

Pelas tabelas, nós podemos ver muitos problemas com o programa. As classes com mais comentários, são também as classes que mais comentários em javadoc tem.

A classe SimpleRenderedImage é a classe com com o maior número de linhas de código em comentário.

A classe GanttProject, que é a classe principal, é a que tem mais linhas de código do programa, e faz todo o sentido pois é onde as estruturas de dados necessárias são criadas.

O elevado número de linhas de comentários pode ser um possível code smells, isto é o uso excessivo de comentários para explicar um design complicado no código que pode indicar um design a ser encoberto por exemplo…

O elevado número de linhas de código pode ser também um code smells, como por exemplo complexidade elevada que o código não precisa de ter ou métodos longos demais.

## Representações Gráficas da Classe GanttProject



## 

## 

# Métricas Chidamber and Kemerer

*Autor: Catarina Crespo*

*Review por:*

## Glossário

* **Coupling Between Objects (CBO)** - calcula o número de classes ou interfaces acopladas com a classe selecionada. Uma classe diz-se “acoplada” com outra se estabelece uma relação de dependência com a mesma (ignoram-se relações de dependência devido à hereditariedade)
* **Depth of Inheritance Tree (DIT)** - calcula a profundidade da árvore de hereditariedade para cada classe, isto é, a profundidade corresponde ao número de passos de hereditariedade entre a classe e java.lang.Object
* **Lack of Cohesion Of Methods (LCOM)** - calcula o grau de coesão de uma classe. Dois métodos de uma classe estão relacionados se eles partilham o uso de uma variável ou um chama o outro. A métrica é então o número de componentes do gráfico da relação. Valor 1 indica uma classe fortemente coesa (tornando difícil a separação em classes mais pequenas). Valores mais altos indicam geralmente que a classe devia ser separada noutras mais específicas.
* **Number Of Children (NOF)** - calcula o número de subclasses diretas de cada classe do projeto
* **Response For Class (RFC)** - dá-nos o total do número de construtores e métodos de uma classe mais o número de construtores e métodos que essa mesma classe pode chamar. Classes com um RFC elevado têm grande complexidade e podem ser menos estáveis.
* **Weighted Method Complexity (WMC)** - calcula o total da complexidade ciclomática (mede o número de caminhos de execução distintos ao longo de cada método - soma de if’s, while’s, for’s, etc) dos métodos de cada classe

## Métricas Chidamber and Kemerer - Valores mais relevantes

### Coupling Between Objects (CBD)

| **Classe** | **CBD** |
| --- | --- |
| net.sourceforge.ganttproject.language.GanttLanguage | 138 |
| net.sourceforge.ganttproject.GanttProject | 122 |
| net.sourceforge.ganttproject.action.GPAction | 103 |

## 

### Depth of Inheritance Tree (DIT)

| **Classe** | **DIT** |
| --- | --- |
| net.sourceforge.ganttproject.ResourceTreeTable | 8 |
| net.sourceforge.ganttproject.GanttTreeTable | 8 |
| net.sourceforge.ganttproject.GanttProject | 7 |

### Lack of Cohesion Of Methods (LCOM)

| **Classe** | **LCOM** |
| --- | --- |
| net.sourceforge.ganttproject.export.ConsoleUIFacade | 33 |
| net.sourceforge.ganttproject.GanttProjectBase | 31 |
| net.sourceforge.ganttproject.gui.UIUtil | 22 |

### Number Of Children (NOF)

| **Classe** | **NOF** |
| --- | --- |
| net.sourceforge.ganttproject.action.GPAction | 86 |
| net.sourceforge.ganttproject.test.task.TaskTestCase | 32 |
| net.sourceforge.ganttproject.parser.AbstractTagHandler | 19 |

### Response For Class (RFC)

| **Classe** | **RFC** |
| --- | --- |
| net.sourceforge.ganttproject.GanttProject | 382 |
| net.sourceforge.ganttproject.task.TaskManagerImpl | 260 |
| biz.ganttproject.impex.msproject2.ProjectFileImporter | 248 |

### Weighted Method Complexity (WMC)

| **Classe** | **WMC** |
| --- | --- |
| net.sourceforge.ganttproject.task.TaskManagerImpl | 173 |
| net.sourceforge.ganttproject.GanttProject | 172 |
| org.ganttproject.chart.pert.ActivityOnNodePertChart | 142 |

## 

## 

# 

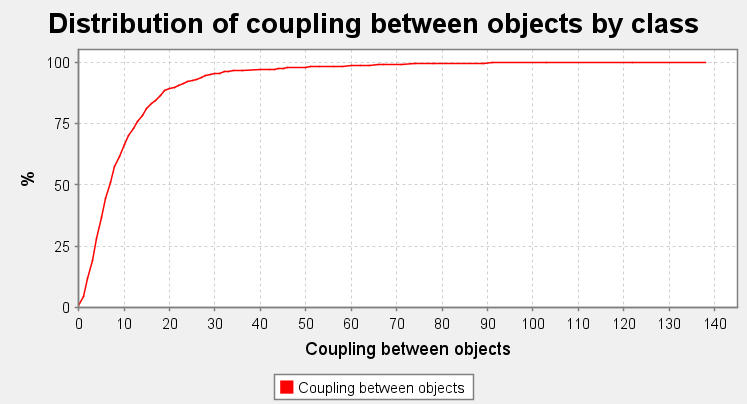
## Comentários (Métricas Chidamber and Kemerer)

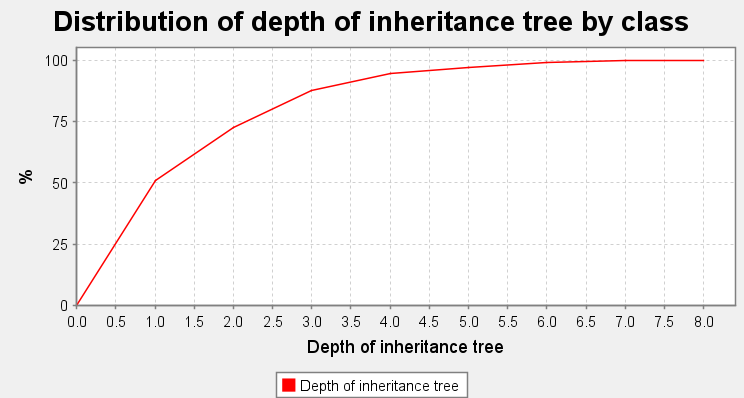
Através da análise dos gráficos e tabelas, podemos concluir que há diversas classes com uma complexidade extremamente elevada, o que origina dificuldades na escrita de testes unitários e pode tornar a classe pouco estável. Exemplo disto é por exemplo a classe GanttProject (uma das classes centrais do programa), que é extremamente extensa e, como podemos ver pelas tabelas acima, tem um CBD, um RFC e um WMC muito elevados (o que nos leva a concluir o que expliquei acima).

Também com base nas tabelas acima, podemos concluir que classes como GanttProjectBase poderiam ser subdivididas em classes mais pequenas de forma a tornar o código mais fácil de testar, já que, embora não seja uma das classes mais extensas, os métodos que lhe pertencem são pouco coesos e era possível melhorar o código nesta parte. Durante o desenvolvimento das funcionalidades, reparámos que esta classe tinha em comentário uma espécie de “separador” que indicava que os métodos a partir daquele ponto estavam mais relacionados com a interface UIFacade, o que reforça a possibilidade de haver essa separação em classes mais concretas.

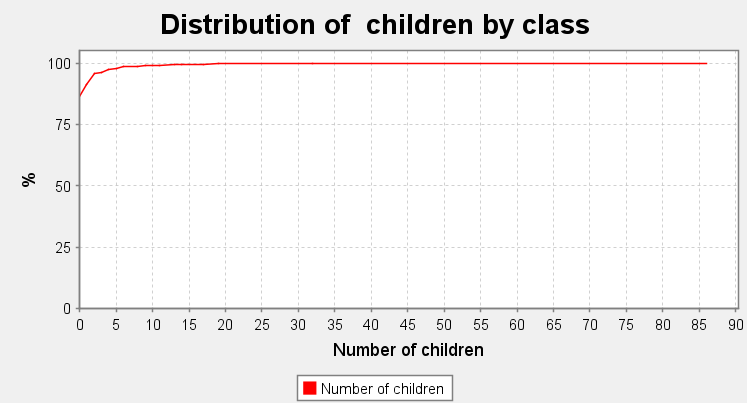
Com base nos valores da Depth of Inheritance Tree, podemos reforçar novamente o facto de o código ser demasiado complexo, o que se revela diversas vezes problemático. É certo que uma classe com um elevado DIT, não implica code smells, mas pode tornar difícil prever o comportamento da classe devido aos métodos herdados. Este foi certamente um dos problemas com que mais nos deparámos ao desenvolver as funcionalidades porque tivémos de visitar diversas classes para entender como funcionaria certo método que estávamos a tentar chamar, tornando muito mais lenta a implementação de novos acrescentos ao programa.

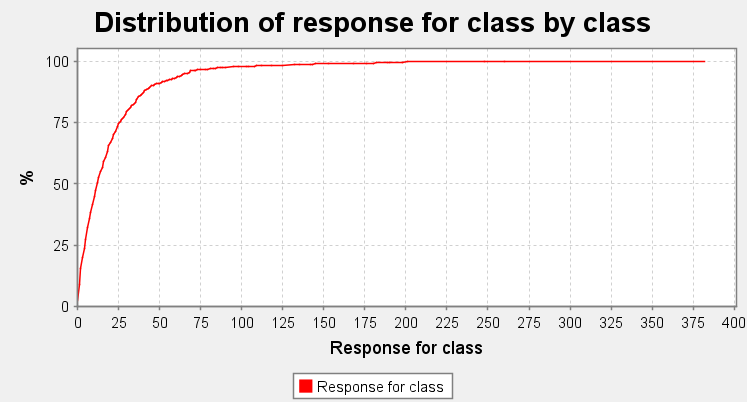
## Distribuições das Métricas Chidamber and Kemerer

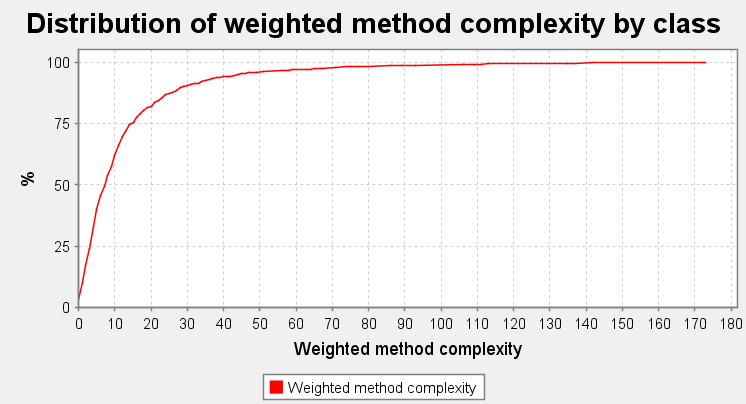












# Martin Packaging metrics

*feito por: danny*

*reviewed por: Afonso*

## Glossário:

* Efferent Coupling (Ce) - é o número de classes em um determinado pacote, que depende das classes em outros pacotes.
* Afferent Coupling (Ca) - é uma adição à métrica Ce e é usada para medir outro tipo de dependência entre pacotes, ou seja, dependências de entrada. Isso nos permite medir a sensibilidade dos pacotes restantes às mudanças no pacote analisado.
* Instability (I) - é usada para medir a suscetibilidade relativa da classe a mudanças. De acordo com a definição de instabilidade, é a proporção de Ce para todas as dependências de pacotes.
* Abstractness (A) - é usada para medir o grau de abstração do pacote e é um pouco semelhante à instabilidade. Em relação à definição, a abstração é o número de classes abstratas no pacote para o número de todas as classes.
* Normalized Distance from Main Sequence (D) - é usada para medir o equilíbrio entre I e A referidos acima..

## Análise:

### Packages

Top 3 Abstractness (A) :

| Package | A | Ca | Ce | D | I |
| --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.gui.options.model | 1.00 | 42 | 55 | 0.57 | 0.57 |
| biz.ganttproject.core.chart.scene | 0.70 | 131 | 163 | 0.25 | 0.53 |
| net.sourceforge.ganttproject.gui.scrolling | 0.67 | 14 | 3 | 0.16 | 0.18 |

Top 3 Afferent Coupling (Ca) :

| Package | A | Ca | Ce | D | I |
| --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.task | 0.32 | 4412 | 770 | 0.53 | 0.14 |
| net.sourceforge.ganttproject | 0.70 | 131 | 163 | 0.30 | 0.51 |
| biz.ganttproject.core.option | 0.67 | 14 | 3 | 0.16 | 0.18 |

Top 3 Efferent Coupling (Ce) :

| Package | A | Ca | Ce | D | I |
| --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject | 0.16 | 2454 | 2829 | 0.30 | 0.51 |
| net.sourceforge.ganttproject.chart | 0.28 | 439 | 1523 | 0.06 | 0.76 |
| net.sourceforge.ganttproject.task.algorithm | 0.29 | 303 | 1233 | 0.09 | 0.79 |

Top 3 Normalized Distance from Main Sequence (D) :

| Package | A | Ca | Ce | D | I |
| --- | --- | --- | --- | --- | --- |
| net.sourceforge.ganttproject.shape | 0 | 5 | 0 | 1 | 0 |
| org.ganttproject | 0 | 0 | 0 | 1 | 1 |
| net.sourceforge.ganttproject.font | 0 | 14 | 0 | 1 | 0 |

Top 3 Instability (I) :

| Package | A | Ca | Ce | D | I |
| --- | --- | --- | --- | --- | --- |
| org.ganttproject | 0 | 0 | 0 | 1 | 1 |
| net.sourceforge.ganttproject.test.task.hierarchy | 0 | 0 | 47 | 0 | 1 |
| net.sourceforge.ganttproject.test.task.dependency | 0 | 0 | 321 | 0 | 1 |

## 

## 

### Comments:

Pelas tabelas, nós podemos ver muitos problemas com o programa:

-O alto valor da métrica Ce>20 indica instabilidade de um pacote, a alteração em qualquer uma das inúmeras classes externas pode ocasionar a necessidade de alterações no pacote. Os valores preferenciais para a métrica Ce estão na faixa de 0 a 20, valores mais altos causam problemas com cuidado e desenvolvimento do código. E como se constata nos valores da tabela, a média da métrica Ce é de 239.05, muito superior ao esperado, indicando instabilidade.

-Altos valores de Ca geralmente sugerem alta estabilidade do componente. Isso se deve ao fato de que a classe depende de muitas outras classes. Portanto, não pode ser modificado significativamente porque, neste caso, aumenta a probabilidade de propagação de tais alterações.Os valores preferidos para a métrica Ca estão na faixa de 0 a 500. O pacote net.sourceforge.ganttproject.task mostra-se muito instável (com Ca igual 4412).

-Os valores preferenciais para a métrica I devem estar dentro dos intervalos de 0 a 0,3 ou 0,7 a 1. Os pacotes devem ser muito estáveis ​​ou instáveis, portanto devemos evitar pacotes de estabilidade intermediária, mas há muitas no programa.

-Os valores preferenciais para a métrica A devem assumir valores extremos próximos de 0 ou 1. Os pacotes que são estáveis ​​(métrica I próxima de 0), o que significa que dependem em um nível muito baixo de outros pacotes, também devem ser abstratos (o fechamento da métrica A é 1). Por sua vez, os pacotes muito instáveis ​​(métrica I próxima de 1) devem ser constituídos por classes concretas (métrica A próxima de 0).

-O valor da métrica D deveria ser o mais baixo possível para que os componentes estivessem localizados próximos à sequência principal.

## Distribuição das Métricas Martin Packaging

