# Rafael Mira

* **Speculative Generality**

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

Na , 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.





Interface em questão:



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  , no método  , podemos observar a mesma message chain por duas vezes.

Em concreto .

Visto que em ambas a chamadas se está a usar o método , podíamos utilizar uma variável local TaskManager manager =  ;

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.

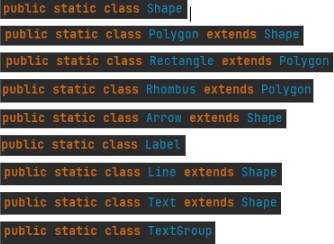
## Revisto por:

Vasco Malta

# Large Class

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

A  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.



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