**Aluno**

Bernardo Atalaia – 59962

Code Smells:

1. Dead Code

In the ArtefactAction Class shown below, there is a variable that receives a value to save it but it’s not used and cant be accessed by other classes, so there is no point in saving this information as it has no purpose in this class.

This is easily fixed by just deleting this variable.

Path: main\java\net\sourceforge\ganttproject\roles\RolePersistentID.java

Uma imagem com texto

Descrição gerada automaticamente

1. Speculative Generality

The RolePresidentID Class has method shown below that is never used, probably the author though that it would be needed in the future, but it never did.

Again, the solution is easy, as the deletion of this method has no impact on the App overall.

Path: main\java\net\sourceforge\ganttproject\roles\RolePersistentID.java

Uma imagem com texto, dispositivo

Descrição gerada automaticamente

1. Primitive Obsession

The ChartUIConfiguration class can be seen as a long class (another code smell) and in this particular case it’s mainly because of the extensive number of variables that this class has, most of them even being useless at some point due to lack of usage. This code smell, in this case, can be solved in a couple of ways, as by deleting all the useless primitives, as grouping primitives that are make sense to be used together and don’t have a big meaning by their own in a new class englobing all of them so this class only has to have the new class as it’s primitive.

Path: main\java\net\sourceforge\ganttproject\chart\ChartUIConfiguration.java

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

Design Patterns:

1. Observer

This Design pattern is usually used to observe a specific class, that is notified once a modification has happened in this class. Other classes that are waiting for a specific state or event, can use their observer to observe it and notify once an important action is made.

In the example below, GPUndoListener is an Interface of an observer, any observer class that implements this is going to observe any actions related to “undo”.

In the second example, there is a class (“UndoManagerImpl”) notifying it’s observers (“listeners”) of an action that happened.

Path: main\java\net\sourceforge\ganttproject\undo\GPUndoListener.java

main\java\net\sourceforge\ganttproject\undo\UndoManagerImpl.java

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

1. Facade

This Pattern is really useful to simplify code and it’s understanding, aswell as giving a simplified interface to a complex system. In the example below, the interface TaskContainmentHierarchyFacade does just that.

Path: main\java\net\sourceforge\ganttproject\task\TaskContainmentHierarchyFacade.java

Uma imagem com texto

Descrição gerada automaticamente

1. **Memento Pattern**

This pattern is basically a backup of a class, or something stored by the class itself of a previous state/version that can be brought back if for some reason an undo is needed.

As the class UndoableEditImpl has in the example below, the old document saved in case an undo is needed along side with the method undo that brings the old document back.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente