***Code Smells***

## Long Method(doSave)

1. Illustrating code snippet:



**Class location**:

/ganttproject/src/main/java/net/[sourceforge](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net/sourceforge)/[ganttproject](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net/sourceforge/ganttproject)/

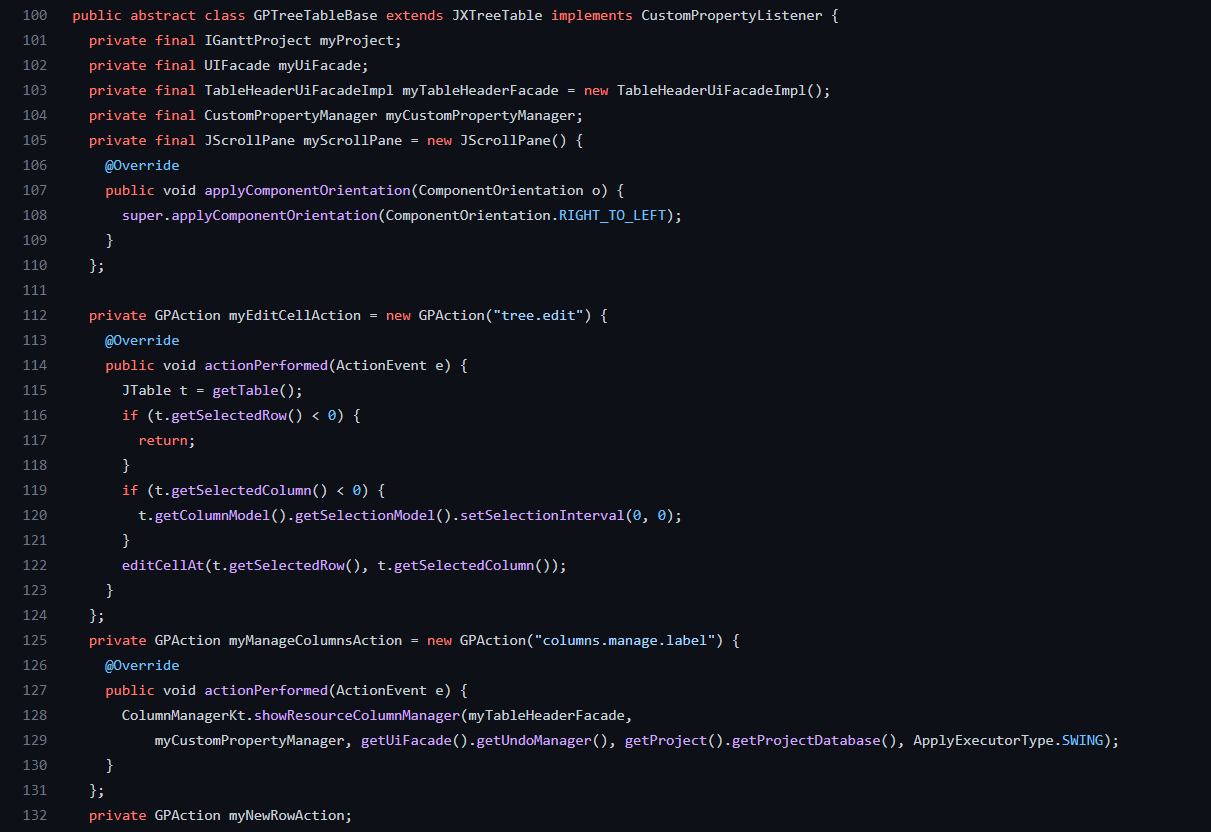
GanttOptions.java

**Code smell:** In GanttOptions.java the method is too extensive (line 210 to line 358) which makes it too complex and hard to understand.

**Refactoring proposal:** this method should be broken down into smaller methods to make the code simpler.

## Long Class(GPTreeTableBase)

Illustrating code snippet:



**Class location**:

/biz.ganttproject.core/src/main/java/[net](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net)/[sourceforge](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net/sourceforge)/[ganttproject](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net/sourceforge/ganttproject)/

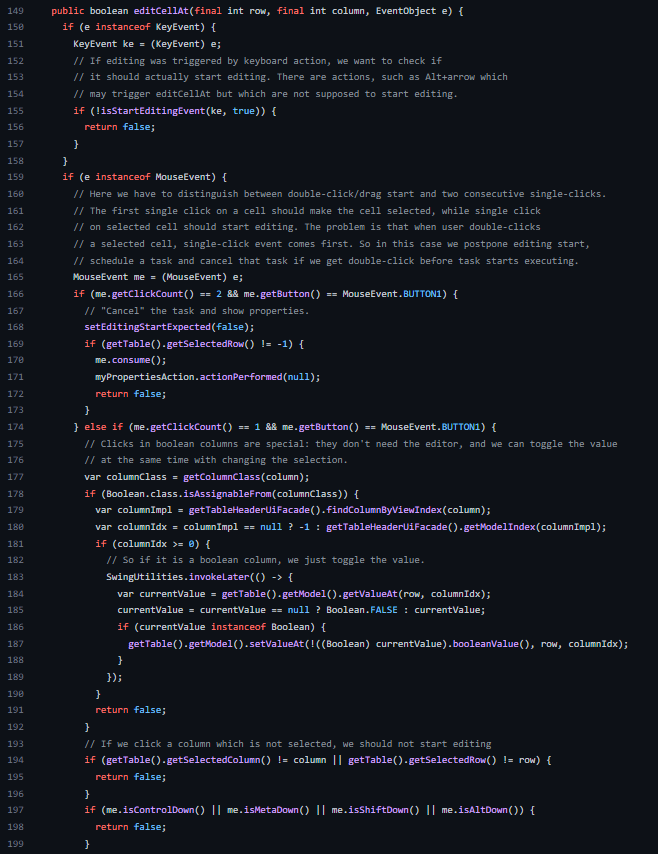
GPTreeTableBase.java

**Code smell:** The GPTreeTableBase.java class is long, having 1271 lines of code, which is a sign that it has too many responsibilities.

**Refactoring proposal:** The most appropriate solution would be to split the class into subclasses and redistribute the responsibilities.

## Comments

Illustrating code snippet:



**Class location**:

/biz.ganttproject.core/src/main/java/[net](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net)/[sourceforge](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net/sourceforge)/[ganttproject](https://github.com/Gui28F/ganttproject/tree/master/ganttproject/src/main/java/net/sourceforge/ganttproject)/

GPTreeTableBase.java

**Code smell:** In this section of the GPTreeTableBase.java class there is an excessive use of comments to explain the editCellAt() method which indicates poor design.

**Refactoring proposal:** The best way to deal with this problem would be to decompose the method into simpler to understand submethods and decrease the use of comments within the methods.