Design Patterns

# Command Design Pattern

Como se pode observar neste exemplo, existem classes que representam ações específicas (no caso, "Copy", "Cut", "Edit", "Paste", "Redo", "RefreshView", "Search Dialog", "SettingsDialog" e "Undo"). Todas estas classes guardam linhas de código para serem executadas mais tarde as vezes que forem necessárias e vão ser enviadas pelo Invoker para o Receiver para este corre o comando ou a açãoUma imagem com texto

Descrição gerada automaticamente

Localização: ganttproject/src/main/java/net/sourceforge/ganttproject/action/edit

# Abstract Factory Pattern

No exemplo apresentado, é definida a interface CalendarFactory que controla todos os tipos de unidade que aparecem no calendário relativamente ao tempo e duração. No entanto, para cada um dos Produtos que o implementam, ainda são implementadas classes específicas, isto é, conseguimos observar uma "fábrica de fábricas", como descrito nas teóricas.

Uma imagem com texto

Descrição gerada automaticamente

Localização: biz.ganttproject.core/src/main/java/biz/ganttproject/core/time/CalendarFactory.java

# Observer Pattern(Behavioral Pattern)

Usamos este tipo de padrão quando se pretende atualizar algum parâmetro que irá alterar por consequência muitos outros parâmetros. Para tal usamos Interfaces Listeners que automaticamente atualizam todos os outros parâmetros a atualizar quando recebe o update do parâmetro original. Neste exemplo podemos observar o ScrollingListenner que recebe o update quando se dá scroll a algum número de dias.

Uma imagem com texto

Descrição gerada automaticamente

**Location:** ganttproject/src/main/java/net/sourceforge/ganttproject all interface