BORG CALENDAR \_ M4(Individual)

**Manouchehr Azizi (523228)**

# Observer (listener) pattern

**Introduction**

In this document I tried to avoid explaining clear things which is explained in comments of selected code here. And I just briefly explain 3 key points (Publisher, Listener (subscriber), notify) of observer pattern which is implemented in Borg Calendar. After introduction you will see in sequence: publisher, notify, Subscriber, Sequence diagram and Partial class diagram section. For extract class “Aid UML diagram” is used. Following is reference to observer: <http://www.vogella.com/articles/DesignPatternObserver/article.html>, <http://java.dzone.com/articles/design-patterns-uncovered>.

**Publisher**

In “net.sf.borg.common.profs “ class, Listener class is implemented which is responsible to keep list of listeners and Notify listeners of a pref change (following notifyListeners method).the point is that prefsChanged method is going to be override in all subscriber that we will explain in subscriber section.

**Remark:** prefsChanged method is going to be override to do proper action for each of listener member in case of preference changes.

|  |
| --- |
| /\*\*  \* Interface for classes that want to be notified of preference changes  \*/  **public** **interface** Listener {    /\*\*called when preferences changed.  \*/  **public** **abstract** **void** prefsChanged();  }  /\*\* list of listeners \*/  **static** **private** ArrayList<Listener> *listeners* = **new** ArrayList<Listener>();  /\*\*  \* add a listener  \*  \* **@param** listener the listener  \*/  **static** **public** **void** addListener(Listener listener) {  *listeners*.add(listener);  }  /\*\*  \* Notify listeners of a pref change.  \*/  **static** **public** **void** notifyListeners() {  **for** (**int** i = 0; i < *listeners*.size(); i++) {  Listener v = *listeners*.get(i);  v.prefsChanged();  }  } |

**Notify**

**“Prefs.*notifyListeners*();”** is used in two following class to apply proper action to Observer members:

net.sf.borg.model.Theme

|  |
| --- |
| /\*\*  \* sync with the db. called if the options table is changed by something other than the UI (such as import)  \*/  **public** **static** **void** sync()  {  *loadThemes*();    // notify listeners that Prefs may have changed  **Prefs.*notifyListeners*();**  } |

1. net.sf.borg.ui.options.OptionsView

|  |
| --- |
| /\*\*  \* save all preferences to the preference store based on the current UI  \* values  \*/  **private** **void** applyChanges() {  **for** (OptionsPanel panel : panels)  panel.applyChanges();  // notify all parts of borg that have registered to know about  // options changes  **Prefs.*notifyListeners*();**    **if** (SunTrayIconProxy.*hasTrayIcon*())  SunTrayIconProxy.*getReference*().updateImage();  } |

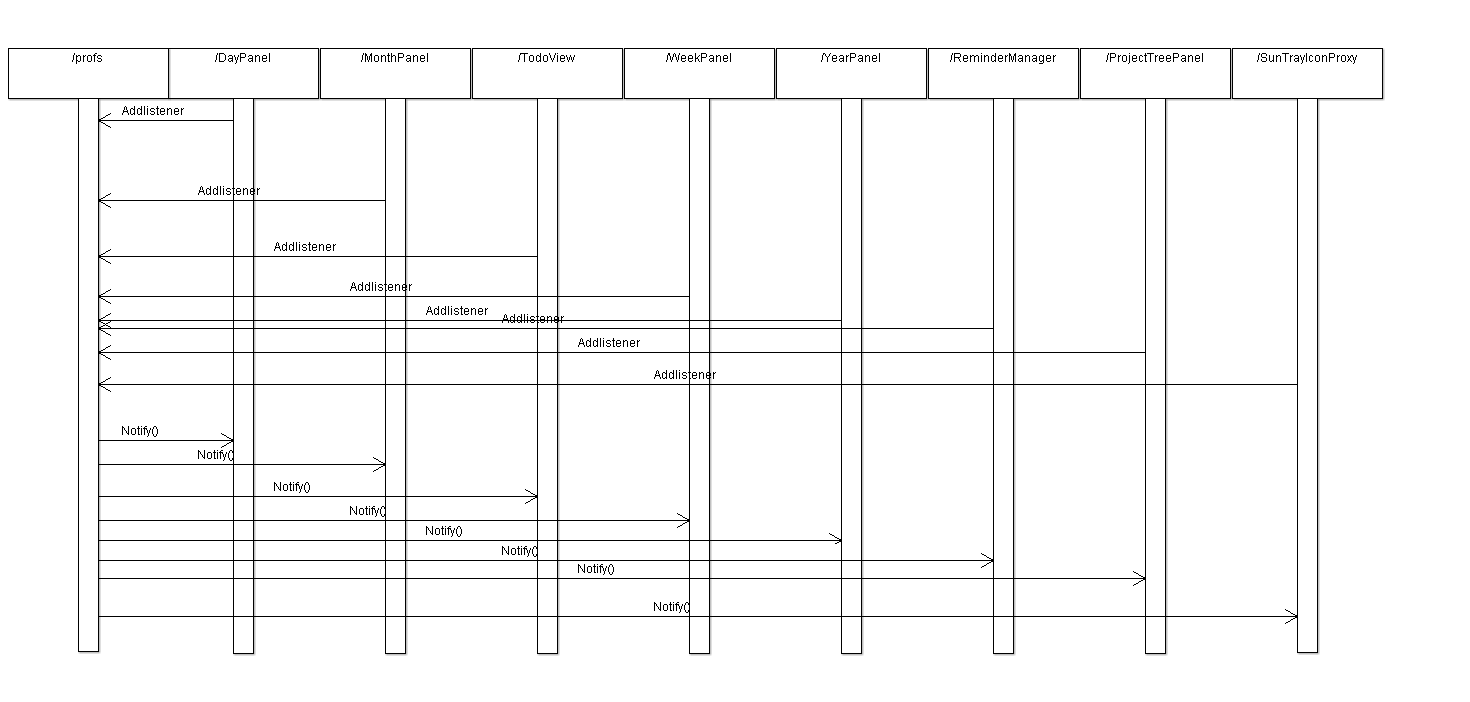
**Subscriber**

Following class with “Prefs.*addListener*(**this**);”add themselves to Listener list:

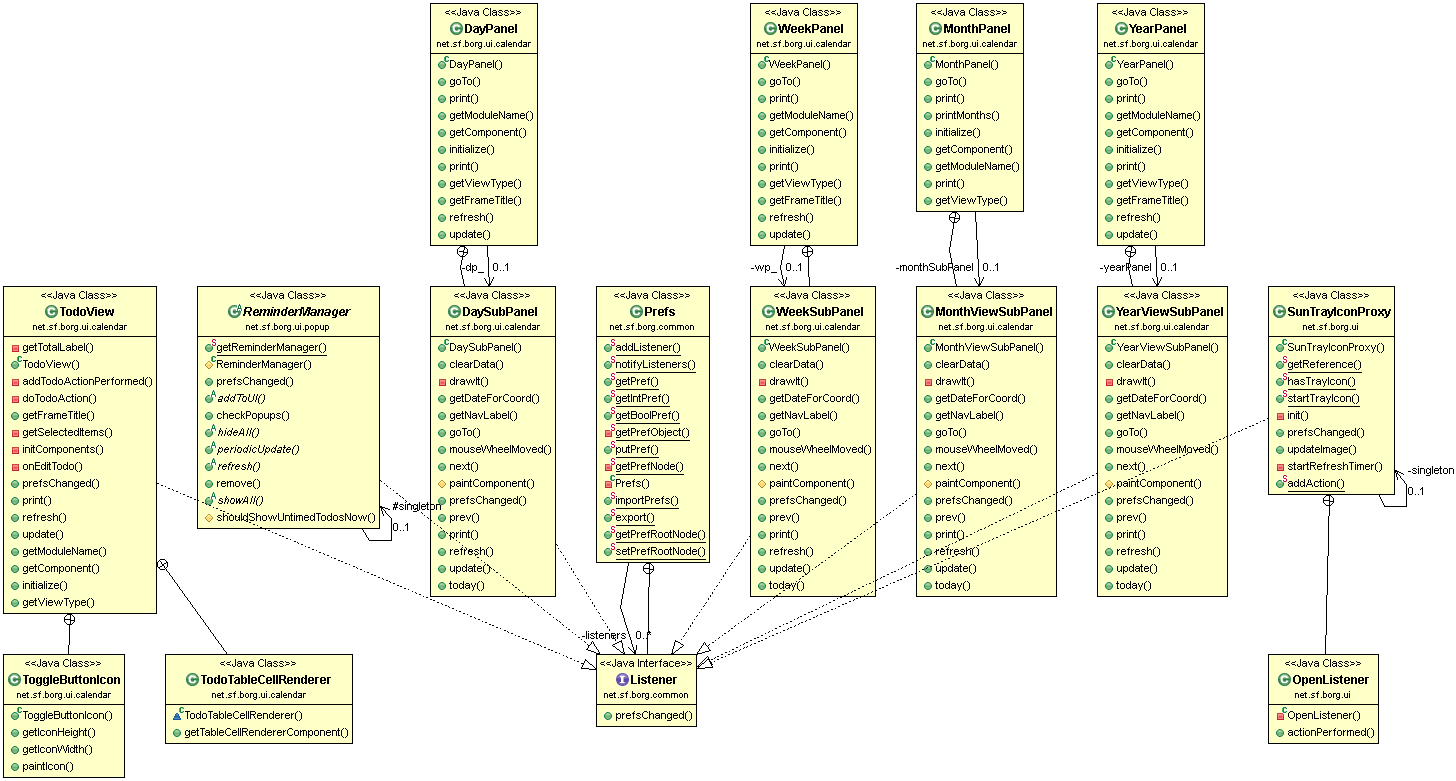
* net.sf.borg.ui.calendar.DayPanel
* net.sf.borg.ui.calendar. MonthPanel
* net.sf.borg.ui.calendar. TodoView
* net.sf.borg.ui.calendar. WeekPanel
* net.sf.borg.ui.calendar. YearPanel
* net.sf.borg.ui.popup. ReminderManager
* net.sf.borg.ui.task. ProjectTreePanel
* net.sf.borg.ui. SunTrayIconProxy

**Remark**: in all of upcoming class, prefsChanged method is override to do proper action in case of performance change.

**Sequence diagram**



**Partial Class diagram**

****