Code Smells

1. Speculative Generality

There is a function in “TimeUnitGraph” class that is never used. Maybe it was created with some purpose, but that purpose is no longer needed.

TimeUnit createTimeUnit(String name, TimeUnit atomUnit, int count) {  
 TimeUnit result = new TimeUnitImpl(name, this, atomUnit);  
 registerTimeUnit(result, count);  
 return result;  
}

Solution: As it is never used, we can remove this method and the code will continue to make sense.

Path: java/biz/ganttproject/core/time/TimeUnitGraph.java

1. Lack of comments

There can be found another smell code in this same class, which is the lack of comments. The only Javadoc comment in this class is the following:

*/\*\*  
 \* Created by IntelliJ IDEA.  
 \*   
 \** ***@author*** *bard Date: 01.02.2004  
 \*/*

Due to the lack of comments, the code becomes much more difficult and confusing to understand. There is a simple solution: comments all the functions so other programmers can easily understand the existing code and improve it.

This are some functions to illustrate the lack of comments:

public TimeUnit createDateFrameableTimeUnit(String name, TimeUnit atomUnit, int atomCount, DateFrameable framer) {  
 TimeUnit result = new TimeUnitDateFrameableImpl(name, this, atomUnit, framer);  
 registerTimeUnit(result, atomCount);  
 return result;  
}  
  
public TimeUnitFunctionOfDate createTimeUnitFunctionOfDate(String name, TimeUnit atomUnit, DateFrameable framer) {  
 TimeUnitFunctionOfDate result;  
 result = new TimeUnitFunctionOfDateImpl(name, this, atomUnit, framer);  
 registerTimeUnit(result, -1);  
 return result;  
}  
  
private void registerTimeUnit(TimeUnit unit, int atomCount) {  
 TimeUnit atomUnit = unit.getDirectAtomUnit();  
 List<Composition> transitiveCompositions = myUnit2compositions.get(atomUnit);  
 if (transitiveCompositions == null) {  
 throw new RuntimeException("Atom unit=" + atomUnit + " is unknown");  
 }

1. Long method

The following smell code can be found in “OffsetBuilderImpl” class and is identified as long method because of the size of the function. Methods with such a big size can be very confusing and difficult to understand. They can be easily corrected by creating some auxiliar functions and separate the big “problem” in some smaller ones. (The lack of comments of this long method helps even more for the difficult interpretation)

1. void constructBottomOffsets(List<Offset> offsets, int initialEnd) {  
    int marginUnitCount = myRightMarginBottomUnitCount;  
    Date currentDate = myStartDate;  
    int shift = 0;  
    OffsetStep step = new OffsetStep();  
    int prevEnd = initialEnd;  
    do {  
    TimeUnit concreteTimeUnit = *getConcreteUnit*(getBottomUnit(), currentDate);  
    calculateNextStep(step, concreteTimeUnit, currentDate);  
    Date endDate = concreteTimeUnit.adjustRight(currentDate);  
    if (endDate.compareTo(myViewportStartDate) <= 0) {  
    shift = (int) (step.parrots \* getDefaultUnitWidth());  
    }  
    int offsetEnd = (int) (step.parrots \* getDefaultUnitWidth()) - shift;  
    Offset offset = Offset.*createFullyClosed*(concreteTimeUnit, myStartDate, currentDate, endDate,   
    prevEnd, initialEnd + offsetEnd, step.dayMask);  
    prevEnd = initialEnd + offsetEnd;  
    offsets.add(offset);  
    currentDate = endDate;  
     
    boolean hasNext = true;  
    if (offsetEnd > getChartWidth()) {  
    hasNext &= marginUnitCount-- > 0;  
    }  
    if (hasNext && myEndDate != null) {  
    hasNext &= currentDate.before(myEndDate);  
    }  
    if (!hasNext) {  
    return;  
    }  
    } while (true);  
   }

Path: biz/ganttproject/core/chart/grid/OffsetBuilderImpl.java

Martin Magdalinchev Nº58172 MIEI