**Code smells**

1. **Switch statements**

Graphical user interface, text, application, email

Description automatically generated A picture containing graphical user interface

Description automatically generated

Location:



Rationale: here the type of graph is determined by an integer and then there are a bunch of switch statements regarding the behavior of such graph depending on the type of graph it is. This should be resolved by using polymorphism to specify the type of graph.

Also, if there is the need to add functionalities to different types of graphs we would need to add a switch statement for which, and if we want to add a new graph we would need to change every scattered switch statement.

1. **Law of demeter**



Location:



Rationale: Here *PerChartAbstraction*class in its method *getTaskGraphNode( )*is callinga method named *getNestedTasks()*, and this method is not:

·      Encapsulated within the same object – the method *getNestedTasks()*, is from an object of the type *TaskContainmentHierarchyFacade*, so it obviously isn’t from the same class as *getTaskGraphNode(),*or in other words is not from*PerChartAbstraction*class*.*

·      Encapsulated within an object that is in the parameters of M - the object from which this method is from is received from the method *getTaskHierarchy()*of the myTaskManager object, and this object is not from the parameters of *getNestedTasks()*, it is just returned from a function.

·      Encapsulated within an object that is instantiated inside the M - the object from which this method is from is received from the method *getTaskHierarchy()*of the *myTaskManager* object and this object is not instantiated inside *getNestedTasks()*, it is just returned from a function.

·      Encapsulated within an object that is referenced in an instance variable of the class in M - the object from which this method is from is received from the method *getTaskHierarchy()*of the *myTaskManager* object and not from an instance variable of the class *PerChartAbstraction*.

1. **Law of demeter**



Location:



Rationale: Here *PerChart*class in its method *updateFonts( )*is callinga method named *getFactor()*, and this method is not:

·      Encapsulated within the same object – the method *getFactor()*, is from an object of the type *Size*, so it obviously isn’t from the same class as *updateFonts(),*or in other words is not from*PerChart*class*.*

·      Encapsulated within an object that is in the parameters of *updateFonts()* - the object from which this method is from is received from the method *getSize()*of the *fontSpec* object, and this object is not from the parameters of *updatedFonts()*, it is just returned from a function.

·      Encapsulated within an object that is instantiated inside *updateFonts()*- the object from which this method is from is received from the method *getSize()*of the *fontSpec* object and this object is not instantiated inside *updateFonts()*, it is just returned from a function.

·      Encapsulated within an object that is referenced in an instance variable of the class in  *updateFonts()* - the object from which this method is from is received from the method *getSize()*of the *fontSpec* object and not from an instance variable of the class *PerChart*.