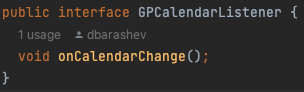
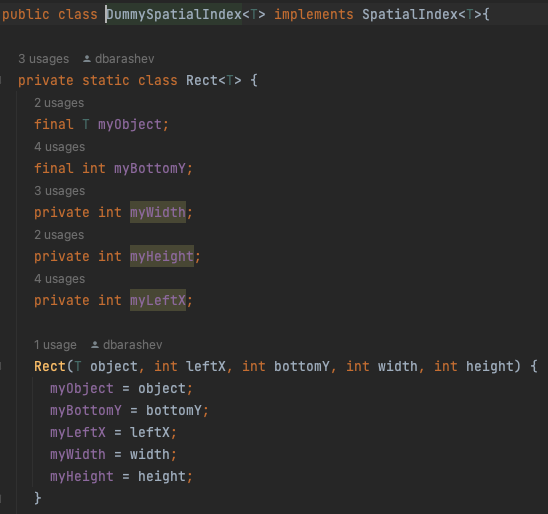
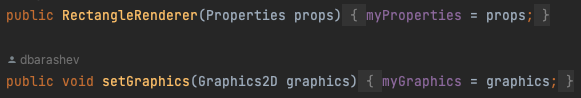
Identified Code Smells

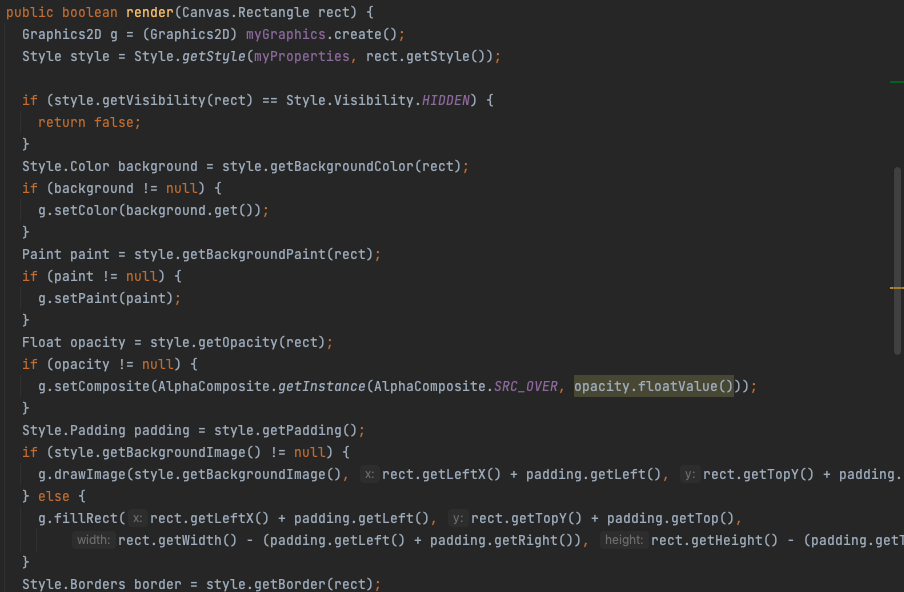
* Inês Carvalho 45345:
  + Code Smell 1 - Dead Code:
    - Illustrating code snippet:



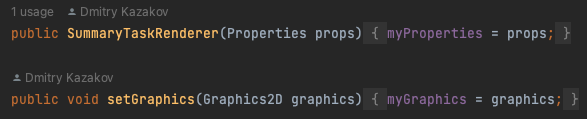
* + - The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> calendar -> GPCalendarListener
    - An explanation of the rationale for identifying this code smell: This interface has only one void method. Classes do not implement this interface, yet the method has 4 usages calls on the WeekendCalendarImpl fileCalendarChanged().
    - A refactoring proposal: Use a variable on the class as a flag to register the change and then call a method that implements the change. In this case it would be the notification
  + Code Smell 2 - Primitive obsession:
    - Illustrating code snippet:
    - The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> chart -> DummySpatialIndex.java
    - An explanation of the rationale for identifying this code smell: Unnecessary use of a lot of primitive types, thus it makes sense to identify this as a code smell
    - A refactoring proposal: The variables that represent the coordinates could be changed to an abstract type such as a tuple Point
  + Code Smell 3 - Shotgun surgery:
    - Illustrating code snippet:

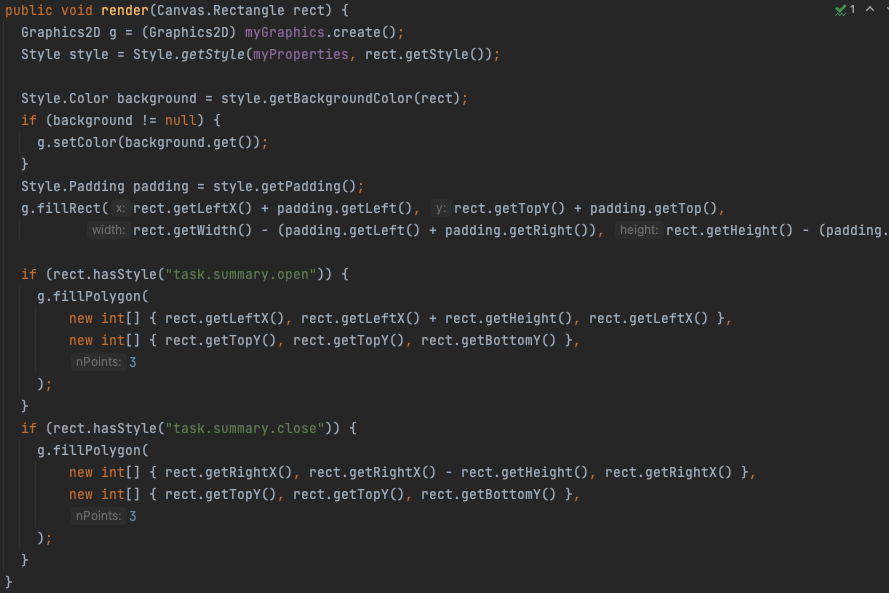
\* On RectangleRenderer.java:





\* On SummaryTaskRenderer.java:





* + - The exact location on the codebase: Project -> Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> chart -> render -> LineRenderer.java, PolygonRenderer.java, Rectangle Renderer, SummaryTaskRenderer
    - An explanation of the rationale for identifying this code smell: LineRenderer, PolygonRenderer, RectangleRenderer and SummaryTaskRenderer have some similar methods and methods with the same name implemented differently. If an update is made change needs to happen in multiple places
    - A refactoring proposal: This appears to be an abstraction. As a possible refactoring I would make an AbstractClass called Renderer and would implement the similarities there, extend the other classes and implement the differences there. That way when an update is to be made that is equal to all types of renderer, then change only happens in one place