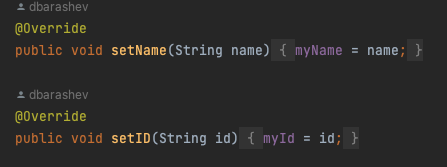
Identified Design Patterns

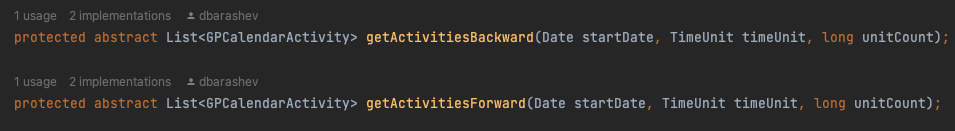
* Inês Carvalho 45345:
  + Pattern 1 - Template Method Pattern:
  + Illustrating code snippet:

\* On the GPCalenderBase class:

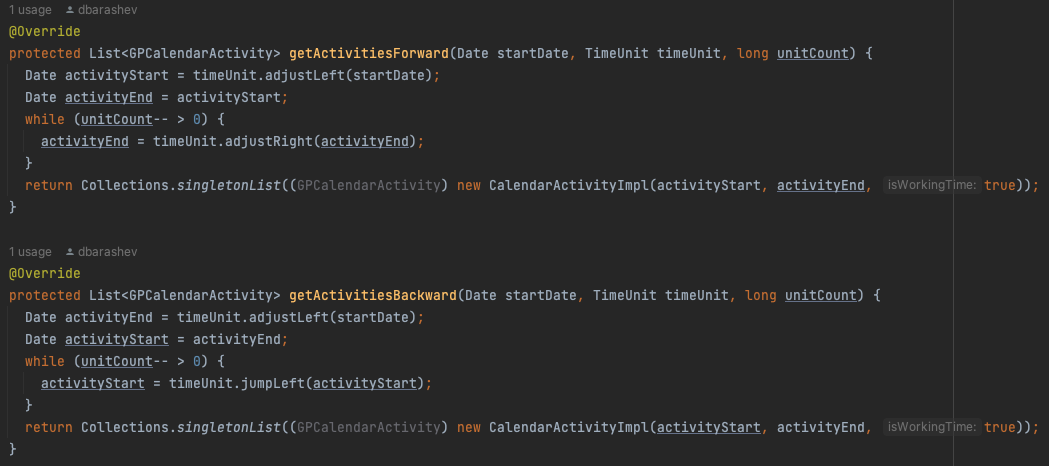
Examples of methods in common



Examples of methods implemented differently:

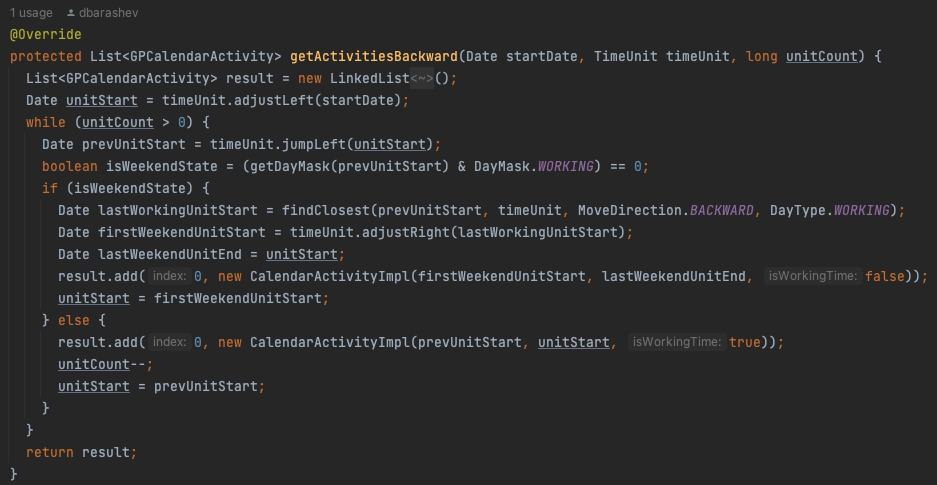


\* On the AlwaysWorkingTimeCalendarImpl class:



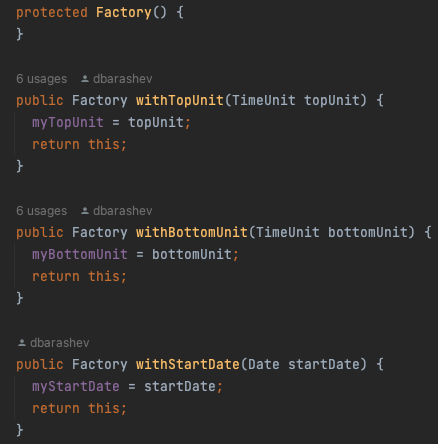
\* On the WeekendCalendarImpl:



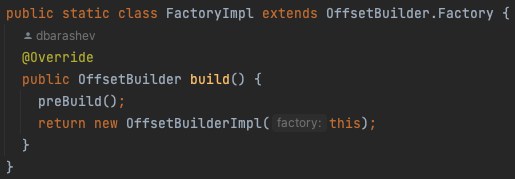


* + The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> calendar
  + An explanation of the rationale for identifying this as a pattern instantiation: Both AlwaysWorkingTimeCalendarImpl and Weekend CalendarImpl classes extend GPCalendarBase abstract class. Both of these classes implement the methods that they have in common but execute them differently. Abstract class on the other hand implements the methods in common for all classes that extend it.
  + Pattern 2 - Factory Object :
    - Illustraring code snippet:

\* On the OffsetBuilder interface:



\* On OffsetBuilderImpl:



* + - The exact location on the codebase: Project -> Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> grid -> OffsetBuilder
    - An explanation of the rationale for identifying this as a pattern instantiation: Hiding the creation of instances of a given type
  + Pattern 3 - Facade:
    - Illustraring code snippet:









* + - The exact location on the codebase: Project -> biz.ganttproject.core -> src -> main -> java -> biz.ganttproject -> core -> chart -> scene -> AbstractSceneBuilder, BottomUnitSceneBuilder, DayGridSceneBuilder, TimelineSceneBuilder
    - An explanation of the rationale for identifying this as a pattern instantiation: Hide complexity behind an interface. Provides a unified interface to a complex subsystem