**Written part documentation:**

The process of morphing a stopwatch to a timer embodies a variety of fundamentals. A solid first step was to capitalize on the use of branches given the basis of the project whereby we (the members of the team) may kick on making individual or collective changes to the initial source code without running the risk of losing or overriding successful chunks of code with erroneous ones.

It proved to be indispensable to fathom the sheer structure of the timer’s design, and its UML state machine diagram. Originally, we were convinced that the state names from the stopwatch would remain intact in the timer. However, when rendered a successful UML diagram whereby Increment and Decrement are deemed individual states as opposed to methods, it enabled us to focus on each event (click and tick).

Since this exercise involved a transformation of a distinct application, refactoring was key to its completion. Certain methods issued by interfaces were kept, though others with no function (lap states methods) were removed. The methods that we added were kept concise with simple tasks. Inevitably, each removal or addition of a method meant that we had to assure the app compiled before moving forward.

Modeling should absolutely precede coding given a certain degree of abstraction. Otherwise, we must consider the need for extra methods and data fields. We inevitably come across numerous expectancies while coding, but those can be reduced if our code follows a coherent UML diagram. Under the right circumstances, tests may ensure the code follows the model.

In hindsight, we would put more focus on Roboelectric tests since they are the only ones that enable Android Studio Activities as opposed to Abstract Stopwatch State Machine Tests. These activities are paramount to both of the application’s events: onStart/Stop and onTick. As far as the start/stop button, it occasionally involves a feature that makes the text view editable such that the adapter needs to access the displayed value via the Activity framework. Similarly, the latter occasionally encompasses beeps.

Diagram

Description automatically generated