COMP 313: Project 4

Due date (part 1): Fri 30 Oct
Due date (part 2): Mon 16 Nov
Pair project

Objectives

Familiarity with

- Unified Modeling Language (UML)
- State diagrams (STD)
- Modeling dynamic, event-driven behavior with STDs
- Event-driven program execution
- State dependence in application
- Model-View-Adapter architecture
- Android framework

Description

In this project, you will implement a very simple countdown timer as an Android application. There is no explicit skeleton for this project, but you are strongly encouraged to start from this stopwatch example project: https://github.com/lucoodevcourse/stopwatch-android-java

You may also find parts of this clickcounter example helpful for managing the Activity life cycle, especially saving and restoring state (eg, when the device is rotated) and playing a sound as part of the Activity: https://github.com/lucoodevcourse/clickcounter-android-java

Part 1: Model

For this part, you are encouraged to join forces with one or two other groups and have in-person and/or online discussions.

Deliverable: State diagram

Create a state machine diagram (SMD) as an abstract description of the observable event-driven behavior of the device you will implement in project 4b (please see details below). This diagram will be part of your project 3b submission (in the doc folder of your project).

General rules:

- Represent each visually or logically distinct state of the system as a distinct state in the diagram, while differences in internal data values should not lead to distinct states.
- Do not directly represent UI elements in the diagram; represent events coming from UI elements abstractly and name them according to their UI element source, e.g., increment.

You are welcome to choose from the following tools for this job:

- manual drawing (usually best for in-person group activities); remember to scan your drawing (e.g., using a scanner app on your phone)
- general-purpose drawing tool, such as OmniGraffle or Visio
- UML modeling tool; the top two cross-platform choices are:
    - Astah (free community edition available, recommended)
    - ArgoUML (open source but a bit clunky)

Submission

Please put your model in the doc subdirectory of your project repository (cs313f15p4) and notify us of your submission through Sakai.

Part 2: Android Implementation

Functional Requirements (55%)

The timer has the following controls:

- (0.5) One two-digit display of the form 88.
- (0.5) One multi-function button.

The timer behaves as follows:

- (0.5) The timer always displays the remaining time in seconds.

- (0.5) Initially, the timer is stopped and the (remaining) time is zero.
- (0.5) If the button is pressed when the timer is stopped, the time is incremented by one up to a preset maximum of 99. (The button acts as an increment button.)
- (0.5) If the time is greater than zero and three seconds elapse from the most recent time the button was pressed, then the timer beeps once and starts running. Note: if the time reaches the preset maximum of 99 the timer acts the same way as if three seconds had elapsed - it beeps and starts running.
- (0.5) While running, the timer subtracts one from the time for every second that elapses. In particular, the display only changes 1 second after the 3-second timeout occurs or the timer value reaches 99.
- (0.5) If the timer is running and the button is pressed, the timer stops and the time is reset to zero. (The button acts as a cancel button.)
- (0.5) If the timer is running and the time reaches zero by itself (without the button being pressed), then the timer stops and the alarm starts beeping continually and indefinitely.
- (0.5) If the alarm is sounding and the button is pressed, the alarm stops sounding; the timer is now stopped and the (remaining) time is zero. (The button acts as a stop button.)
- (0.5) The timer handles rotation by continuing in its current state.

Extra credit (worth 20%)

- (2.0 extra credit) Add a 2-digit text area, editable only when the timer is stopped, where the user can type in the time and then press enter or click the button to start the timer. Note that this will require the emulator or device to have a physical or on-screen keyboard to enter the digits, and it is a non-trivial addition to the project!!

Nonfunctional Requirements (25%)

- Ensure your application is testable.
- (1.5) Ensure your application includes comprehensive unit, integration, and functional tests using the techniques from the clickcounter and stopwatch examples where appropriate. (See also http://developer.android.com/tools/testing/testing_android.html)
- (0.5) Follow the design principles discussed so far.
- Maintain a clear, responsibility-based separation among the different building blocks. It is recommended that you start with the stopwatch example.
    - The adapter should be lean (as in the stopwatch example).
    - Most of the complexity should be buried in the model (also as in the stopwatch example).
    - Unlike the stopwatch example, the dynamic model (state machine) has guards involving the current time. Think carefully how to inject the dependency on the time model into the state machine. Do not use static members!
    - (0.5 extra credit) Use of the state pattern (APPP chapter 36) is recommended. Note: If you reuse and update the stopwatch example, you will likely get this extra credit.

- Generally follow good Android development practice: http://developer.android.com
- Use this Timer class: http://developer.android.com/reference/java/util/Timer.html
- (0.5) For beeping, use media playback to play a notification sound (or some other suitable mechanism):
    - http://stackoverflow.com/questions/10335057/play-notification-default-sound-only-android
    - use getApplicationContext() to obtain the required context reference
    - note that clickcounter plays a sound, so you can review how it does that

Written Part/Documentation (20%)

Please include these deliverables in the doc folder of your project repository (cs313f15p4).

- (0.5) Include the model from project 3a. (Minimally, a cell phone scan of your drawing is acceptable.)
- (0.25) Use inline comments to document design details in the code.
- (0.25) Use javadoc comments to document how to use the abstractions (interfaces, classes) you designed.
- Include a brief (300-500 words) report on
    - (0.5) Your pair development journey during this project. Focus on aspects you find noteworthy, e.g., process, pairing, testing, design decisions, refactoring, use of the repository.
    - (0.5) The relationship between your state machine model from this project and your actual code. Possible talking points are:
        - What are the similarities and differences between model and code?
        - Is it more effective to code or model first?
        - Now that you have the code, what, if any, changes would you make to your model?

Grading Criteria

- Stated percentages for the major categories
- Stated points for the specific items
- Total 10 points with opportunities for up to 2.5 points of extra credit
- Deductions of up to 1 point for
    - deviation from the required project folder structure (see clickcounter)
    - inability to run and/or test

Submission

Please work in your project repository (cs313f15p4) from the beginning and share this repository with TA and instructor early on. Then notify us of your submission through Sakai. *IMPORTANT: Please make sure you are not including any generated files in your submission.*