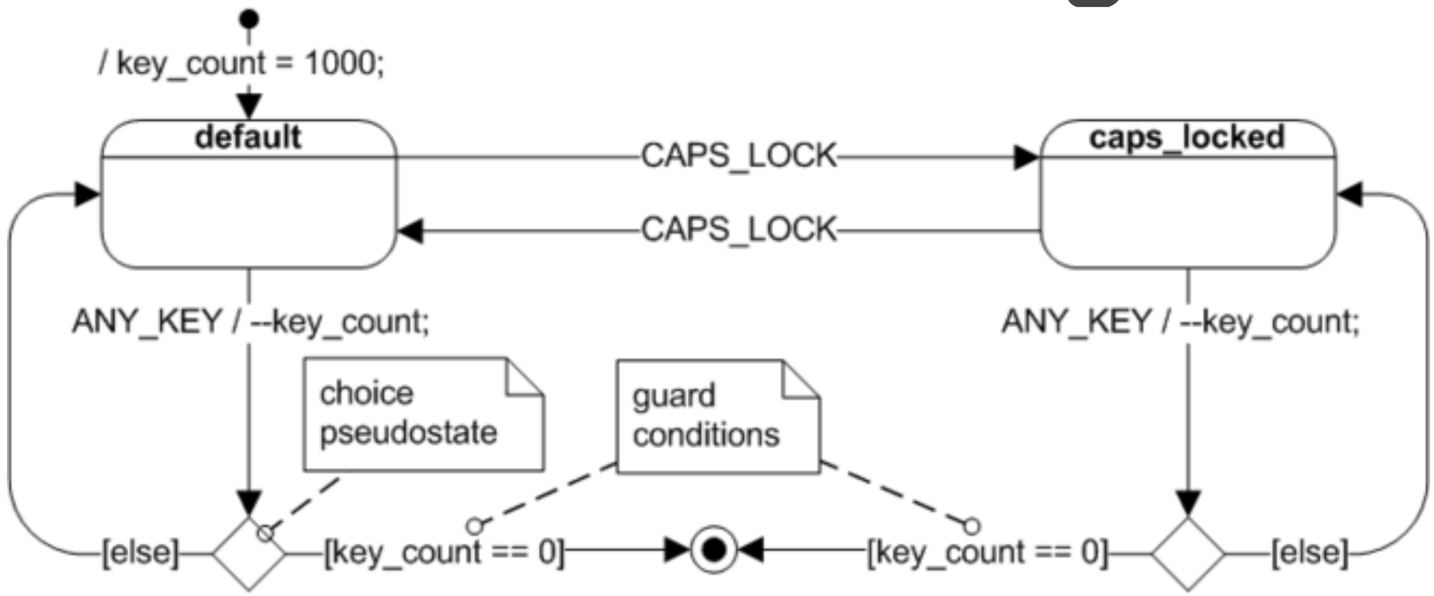
Extended UML State Diagrams



This is an example of an extended state machine, in which the complete condition of the system (called the extended state) is the combination of a qualitative aspect —the "state"—and the quantitative aspects—the extended state variables (such as the down-counter **key_count**). This keyboard "dies" after 1000 keystrokes.
The obvious advantage of extended state machines is flexibility: for example, extending the lifespan of this "cheap keyboard" from 1,000 to 10,000 keystrokes would not complicate the extended state machine at all.
This also shows the use of **guard conditions**, boolean expressions based on the value of extended state variables, like **key_count**, and/or event parameters.
Time-based guard conditions will prove useful in the Project 4 state diagram.

<u>Project 4</u> – Functional Requirements – Slide 1 of 2

- The timer has the following controls:
  - One two-digit display of the form 88.
  - One multi-function button. **(Clicking the button causes a <u>click</u> event.)**
- The timer behaves as follows (part 1 of 2):
  - The timer always displays the remaining time in seconds.
  - Initially, the timer is stopped and the (remaining) time is zero.
  - 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.)
  - 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. **(When the remaining time is greater than 0 the <u>clock</u> model (see stopwatch) is used to send <u>tick</u> events to the state machine.)**

Project 4 – Functional Requirements – Slide 2 of 2

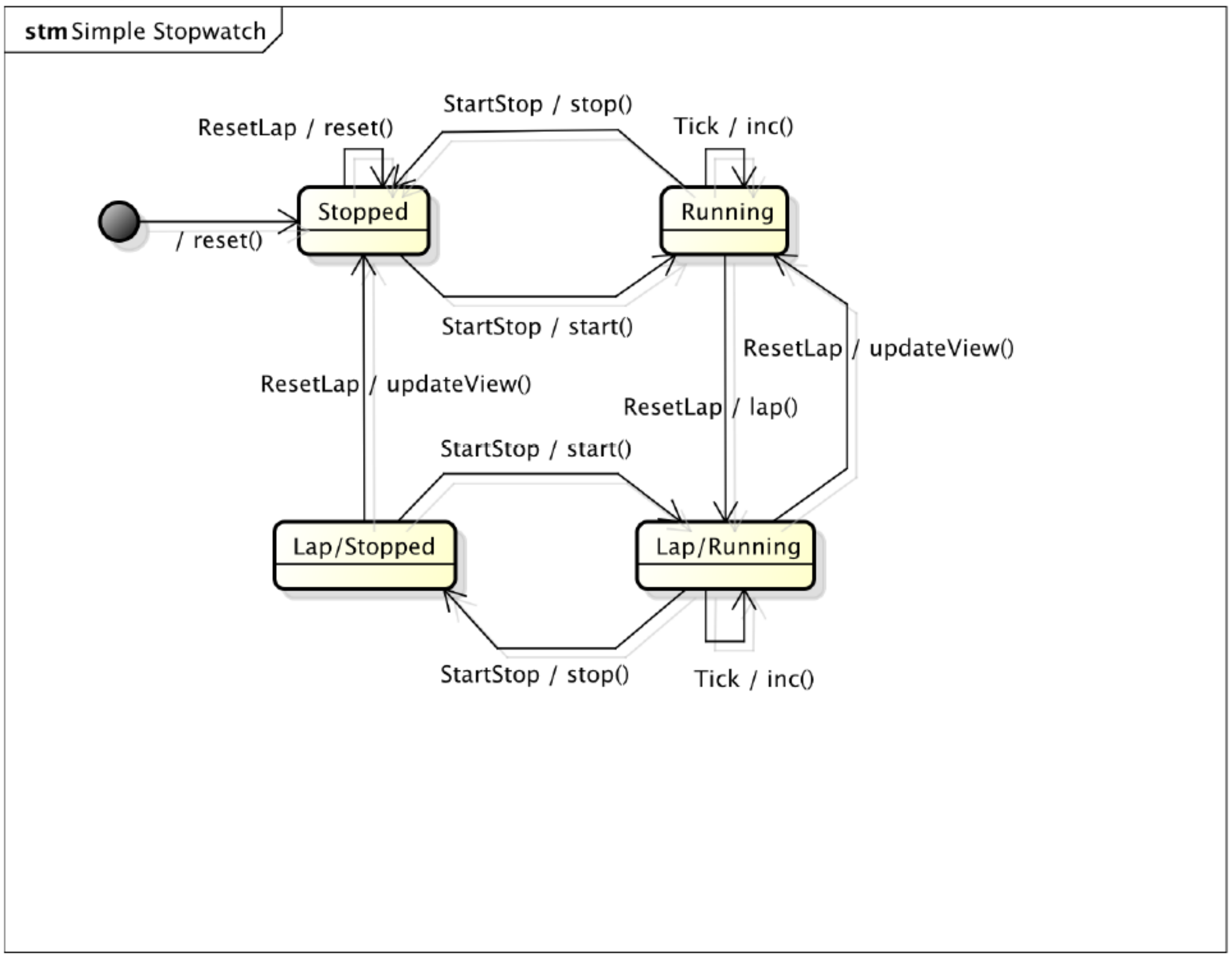
- The timer behaves as follows (part 2 of 2):
  - While running, the timer subtracts one from the time for every second that elapses. **(Caused by a clock model tick event.)**
  - If the timer is running and the button is pressed, the timer stops and the time is reset to zero. (The button acts as a cancel button.)
  - 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.
  - 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.)
  - The timer handles rotation by continuing in its current state.
- **In your groups, develop an extended state diagram for Project 4 – keep a copy for your team's Project 4!!**

LOYOLA UNIVERSITY CHICAGO

stopwatch-android-java

Reminder: the stopwatch State Machine

Simple Stopwatch



clickcounter Example

- **Objectives**
  - Simple dependency injection
  - Event-driven program execution
  - State dependence in applications
  - Mapping the model-view-adapter architecture to Android (and the command-line)
  - Android application life cycle management (including rotation and back button)
  - Playing a notification sound in Android
  - Adapter pattern (wrapper, as opposed to the adapter in MVA)
  - Dependency inversion principle (DIP)
  - Automated unit and integration testing with JUnit
  - Testcase Superclass pattern for xUnit testing
  - Automated system testing by interacting with the GUI
  - Automated GUI testing in Android

Implicit clickcounter State Machine