More About Project 4 – Two Useful "Starter" Projects

- stopwatch-android-java:
  - Two two-digit displays of the form 88 showing elapsed minutes & seconds.
  - Two buttons: Start/Stop and Reset/Lap.
  - Once started, counts clock "tick" events until one of the buttons is clicked.
  - Various other functionality related to Lap timing, etc.
  - Implemented using the State design pattern.
  - Only implements a few Android Activity methods.
- clickcounter-android-java:
  - Display area showing current count of "click" events between MIN and MAX.
  - Three buttons: +, –, 0 (to reset the counter).
  - Counts clicks and increases, decreases, or resets the displayed count.
  - Does not do timing, does not implement the State design pattern
  - Implements most Android Activity methods.

stopwatch-android-java

Objectives

- **Modeling**
  - Modeling state-dependent behavior with state machine diagrams
  - Distinguishing between view states and (behavioral) model states
- **Semantics**
  - Event-driven/asynchronous program execution
  - User-triggered input events
  - Internal events from background timers
  - Concurrency issues: single-thread rule of accessing/updating the view in the GUI thread

stopwatch-android-java

Architecture, Design, and Testing – 1

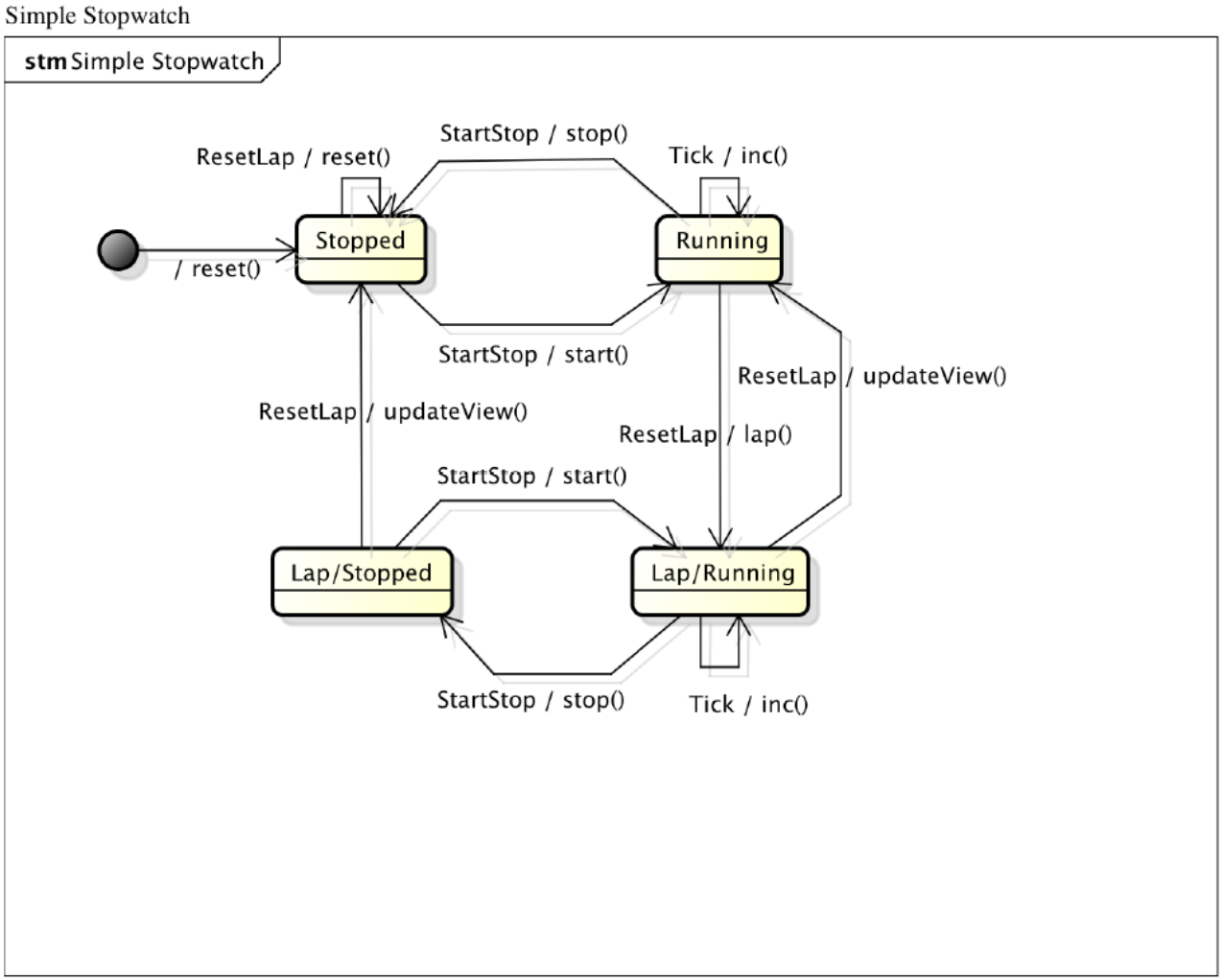
- **Key architectural issues and patterns**
  - Simple dependency injection (DI)
  - Model-view-adapter (MVA) architectural pattern, differences between MVA and model-view-controller (MVC)
  - Mapping MVA to Android
  - Distinguishing among dumb, reactive, and autonomous model components
- **Key design patterns**
  - Implementing event-driven behavior using the Observer pattern
  - Implementing state-dependent behavior using the State pattern
  - Command pattern for representing tasks as objects
  - Façade pattern to hide complexity in the model from the adapter

stopwatch-android-java

Architecture, Design, and Testing – 2

- **Relevant class-level design principles**
  - Dependency Inversion Principle (DIP)
  - Single Responsibility Principle (SRP)
  - Interface Segregation Principle (ISP)
- **Package-level architecture and relevant principles**
  - Dependency graph
  - Stable Dependencies Principle (SDP)
  - Acyclic Dependencies Principle (ADP)
- **Testing**
  - Different types of testing - component-level unit testing, system testing, instrumentation testing
  - Mock-based testing, testcase Superclass pattern, test coverage

stopwatch-android-java
State Machine Mapped to Java Code and stopwatch demo



LOYOLA UNIVERSITY CHICAGO

clickcounter Example – overview and demo

- **Objectives**
  - Simple dependency injection
  - Event-driven program execution
  - State dependence in applications
  - Mapping the model-view-adapter architecture to Android
  - Android application life cycle management
  - 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