LOYOLA
UNIVERSITY
CHICAGO

COMP 413: Intermediate Object-Oriented Programming

Dr. Robert Yacobellis
Advanced Lecturer
Department of Computer Science

Week 11 Topics

- **Test 2**
  - **90 minutes, including a 10-minute break; open book / notes / course computer content**
- **More About Project 4**
  - **Starter Projects – stopwatch and clickcounter**
- **Android Application Development**
  - **Android framework & activity life cycle**
- **If time: Android simplebatch and simpledraw examples**

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

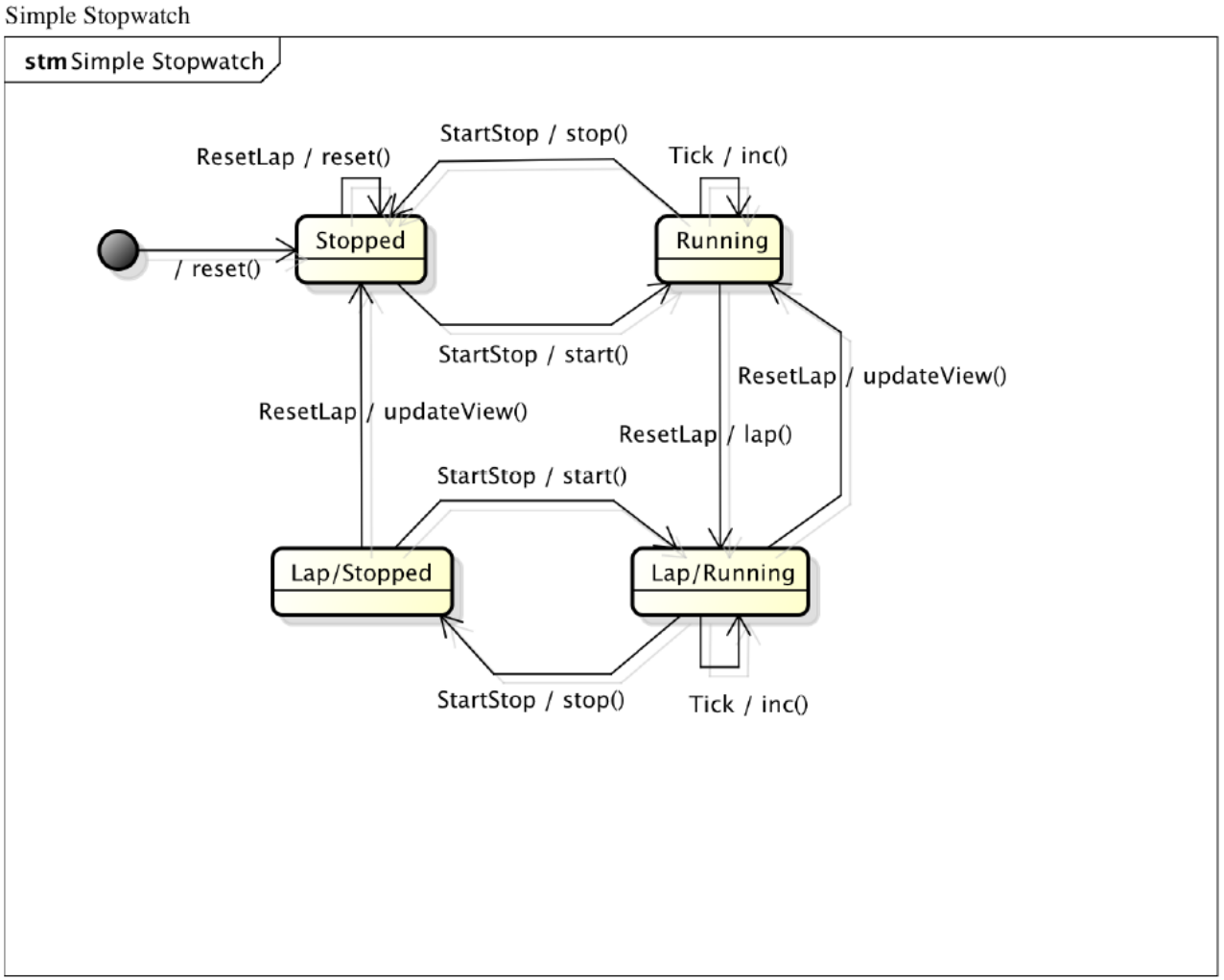
- **Key architectural issues and patterns**
  - Simple dependency injection (DI)
  - Model-view-adapter (MVA) architectural pattern, differences between MVA and model-view-controller (MVC)
  - Mapping MVA to Android
  - Distinguishing among dumb, reactive, and autonomous model components
- **Key design 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

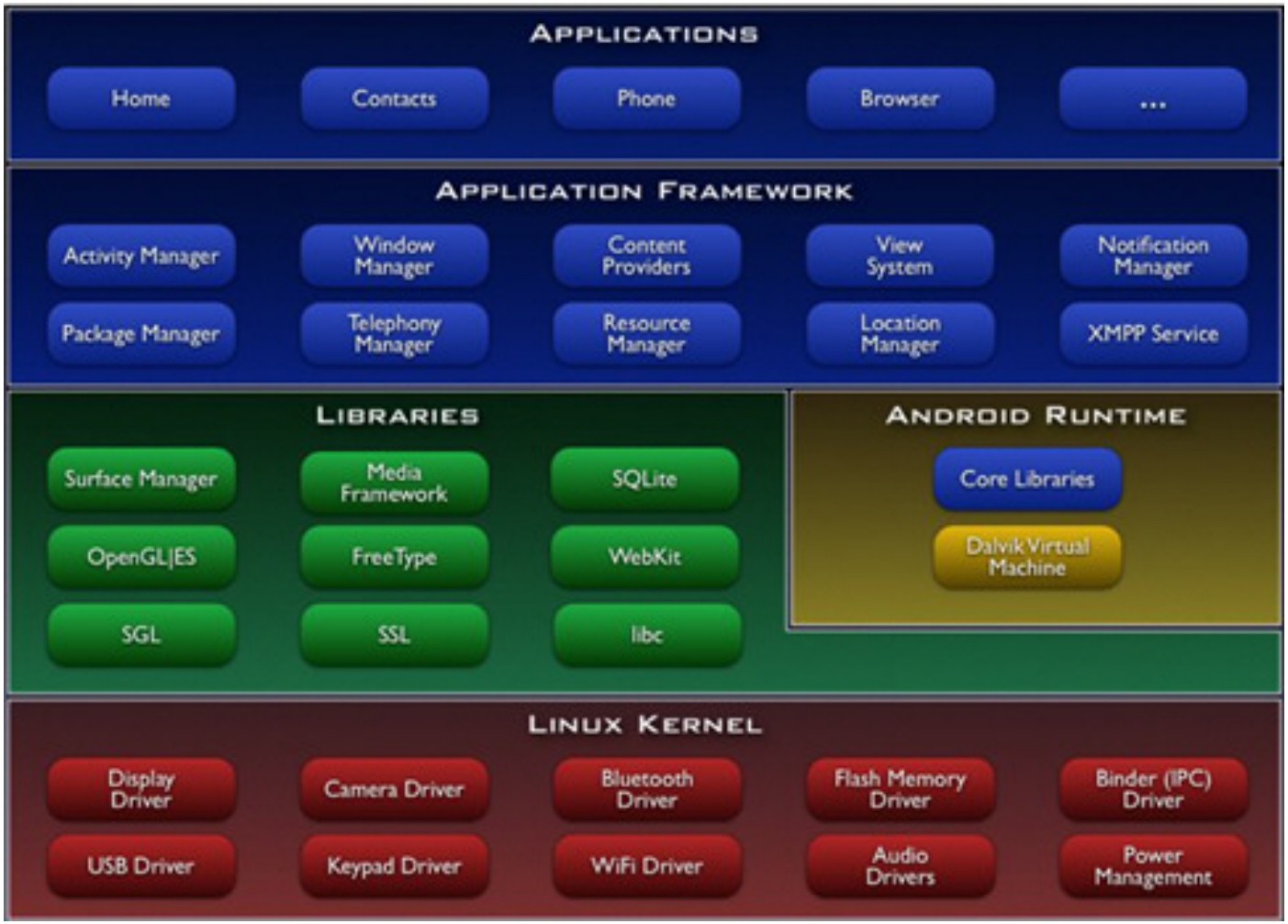


LOYOLA UNIVERSITY CHICAGO

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

Android Framework/Architecture

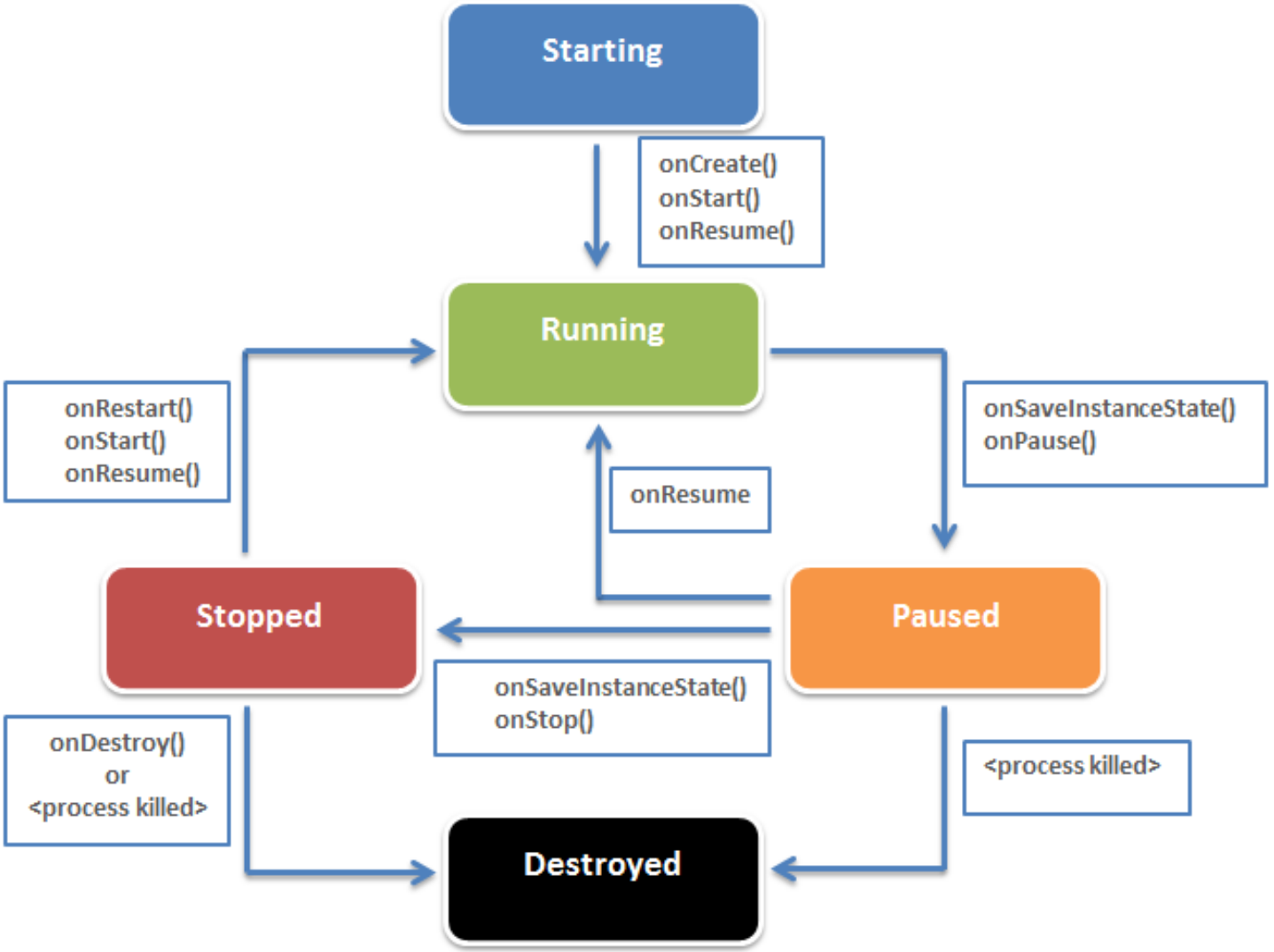


Android Class Hierarchy

Classes in yellow are those a developer typically deals with directly.

Source: http://androidcooltipsandtricks.blogspot.com/

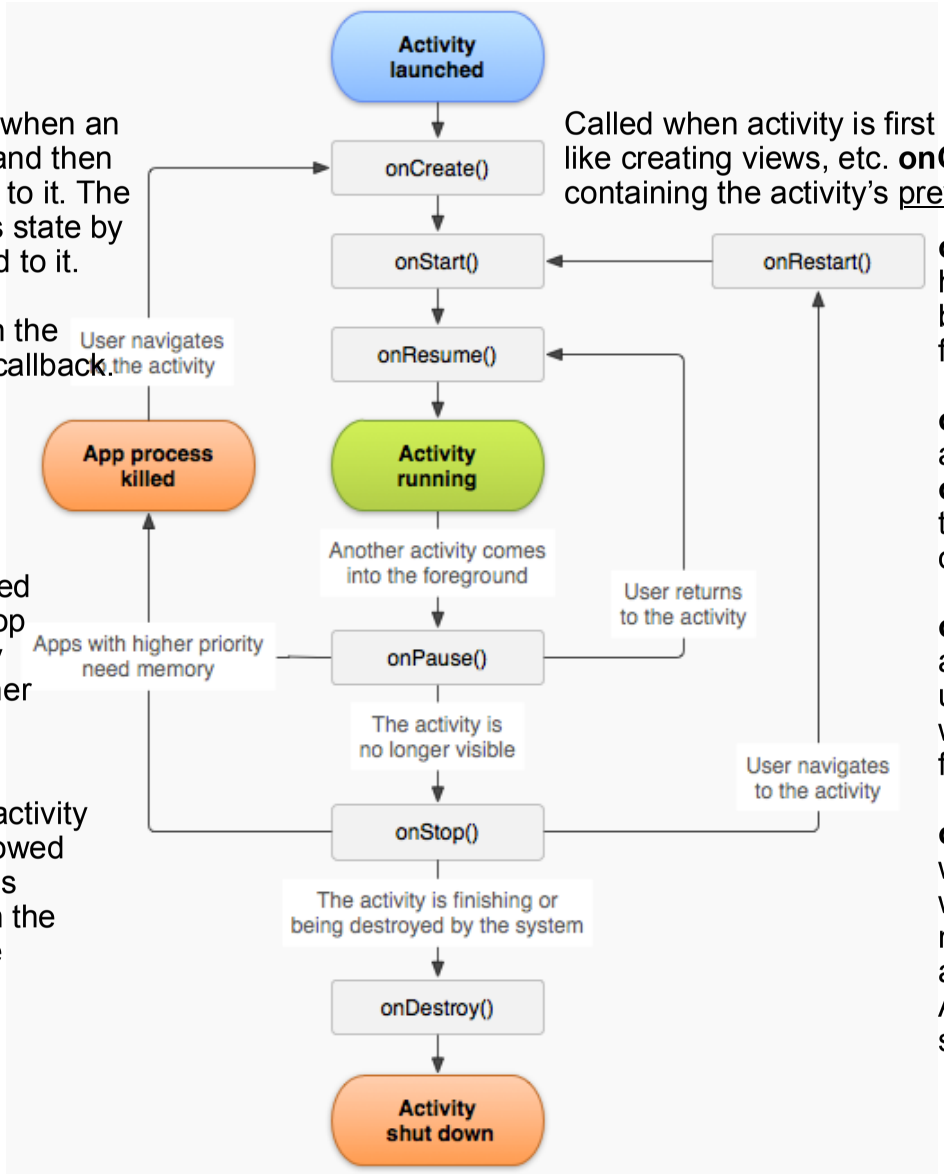
Android Activity Life Cycle



Android Activity Life Cycle

onCreate is also called when an activity has been killed and then the user navigates back to it. The activity must recreate its state by using the Bundle passed to it.

The Bundle is created in the **onSaveInstanceState** callback.



Called when activity is first created – do all normal static setup, like creating views, etc. **onCreate** is passed a Bundle object containing the activity's previous state, if any, or null on startup.

onRestart is called when activity has been stopped, just prior to it being started again – it is always followed by **onStart**.

onStart is called just before the activity becomes visible to the user. **onStart** is followed by **onResume** if the activity comes to the foreground, or by **onStop** if it becomes hidden.

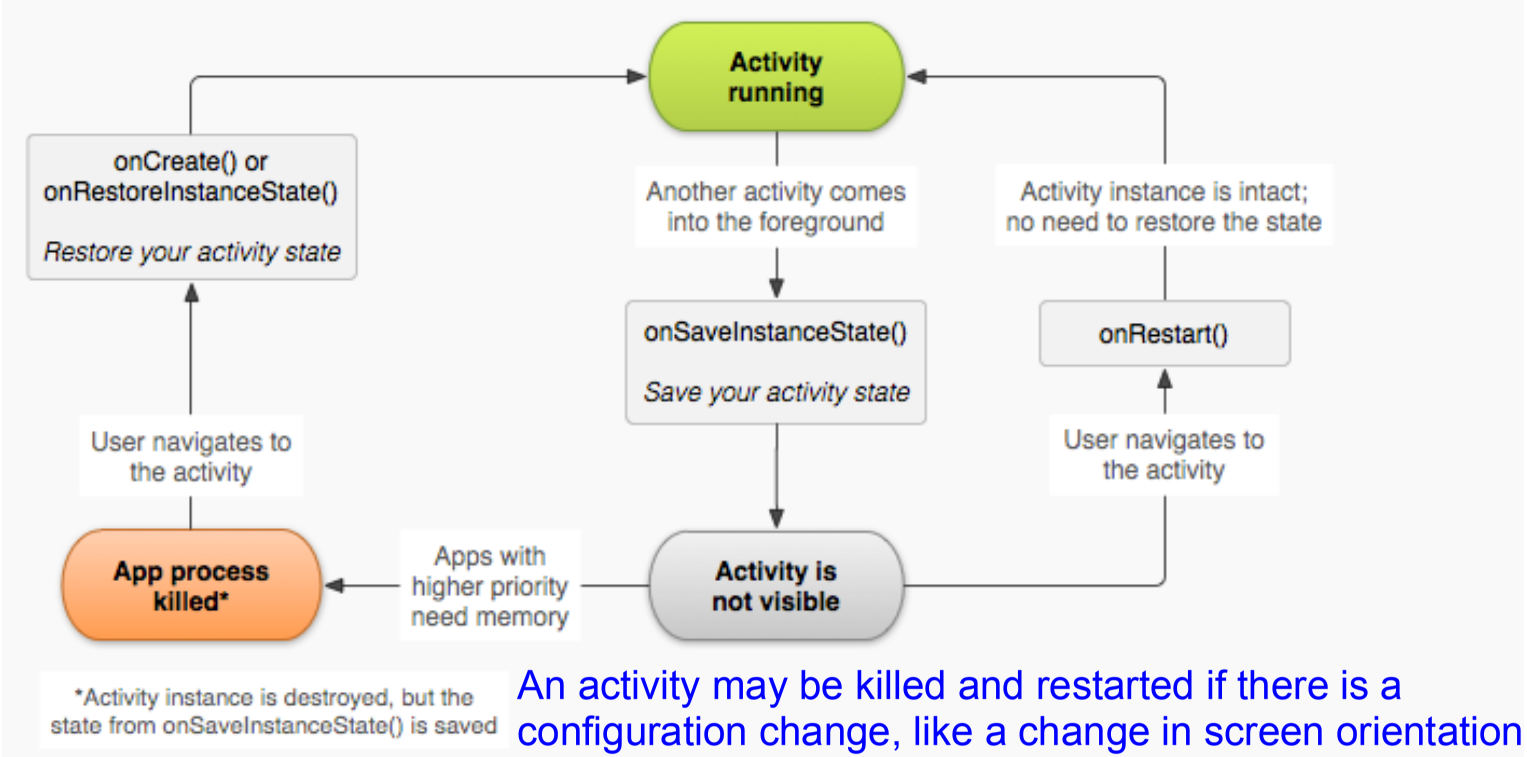
onResume is called just before the activity starts interacting with the user; always followed by **onPause** when another activity comes to the foreground.

onPause is followed by **onResume** when the user returns to the activity while it is still (partially) visible. It may be followed by **onStop** if the activity is no longer visible, or the App process may be killed if the system needs memory.

onPause should commit unsaved resources to persistent data, stop animations, etc, but do this very quickly so it doesn't block another activity that is being resumed.

onStop is called when the activity is no longer visible. It is followed by **onRestart** if the activity is coming back to interact with the user, or by **onDestroy** if the activity is going away.

Android Activity Life Cycle – Activity Regaining Focus



An activity may be killed and restarted if there is a configuration change, like a change in screen orientation.

Figure 2. The two ways in which an activity returns to user focus with its state intact: either the activity is destroyed, then recreated and the activity must restore the previously saved state, or the activity is stopped, then resumed and the activity state remains intact.

Week 11 Topics

- **Test 2**
  - **90 minutes, including a 10-minute break; open book / notes / course computer content**
- **More About Project 4**
  - **Starter Projects – stopwatch and clickcounter**
- **Android Application Development**
  - **Android framework & activity life cycle**
- **If time: Android simplebatch and simpledraw examples**