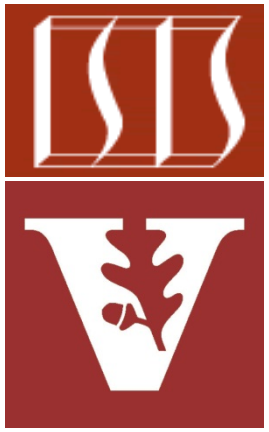
Introduction:
MOOC Prerequisites, Workload, & Learning Strategies

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt



Professor of Computer Science

Institute for Software Integrated Systems

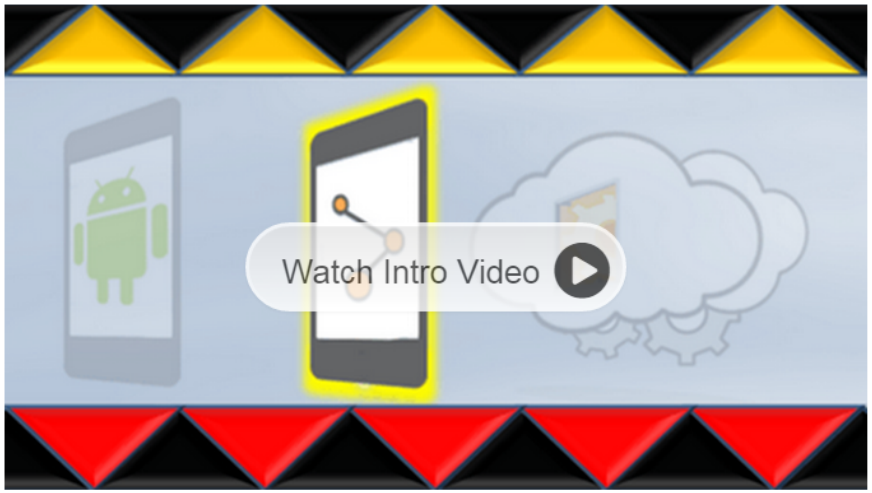
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Module

- Understand MOOC prerequisites, workload & how to complete it successfully

VANDERBILT UNIVERSITY

Programming Mobile Services for Android Handheld Systems: Concurrency



Part of the Mobile Cloud Computing with Android Specialization »

In this MOOC, we will learn how to apply patterns and frameworks to alleviate the complexity of developing concurrent applications on mobile devices running Android that connect to popular cloud computing platforms.

About the Course

This MOOC describes by example how to apply patterns and frameworks to alleviate the complexity of developing concurrent software for mobile devices via the use of object-

Sessions

Mar 25, 2015 - Apr 12th 2015

See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ

MOOC
“Prerequisites”

MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites

CS 279. Software Engineering Project. Students work in teams to specify, design, implement, document, and test a nontrivial software project. The use of CASE (Computer-Assisted Software Engineering) tools is stressed. Prerequisite: CS 278. SPRING. [3]

CS 278. Principles of Software Engineering. The nature of software The object-oriented paradigm. Software life-cycle models. Requirements, specification, design, implementation, documentation, and testing of software. Object-oriented analysis and design. Software maintenance. Prerequisite: CS 251. FALL. [3]

CS 251. Intermediate Software Design. High quality development and reuse of architectural patterns, design patterns, and software components. Theoretical and practical aspects of developing, documenting, testing, and applying reusable class libraries and object-oriented frameworks using object-oriented and component-based programming languages and tools. Prerequisite: CS 201. FALL, SPRING. [3]

CS 101. Programming and Problem Solving. An intensive introduction to algorithm development and problem solving on the computer. Structured problem definition, top down and modular algorithm design. Running, debugging, and testing programs. Program documentation. FALL, SPRING. [3]

CS 282. Principles of Operating Systems II. Projects involving modification of a current operating system. Lectures on memory management policies, including virtual memory. Protection and sharing of information, including general models for implementation of various degrees of sharing. Resource allocation in general, including deadlock detection and prevention strategies. Introduction to operating system performance measurement, for both efficiency and logical correctness. Two hours lecture and one hour laboratory. Prerequisite: CS 281. SPRING. [3]

CS 281. Principles of Operating Systems I. Resource allocation and control functions of operating systems. Scheduling of processes and processors. Concurrent processes and primitives for their synchronization. Use of parallel processes in designing operating system subsystems. Methods of implementing parallel processes on conventional computers. Virtual memory, paging, protection of shared and non-shared information. Structures of data files in secondary storage Security issues. Case studies. Prerequisite: CS 231, CS 251. FALL, SPRING. [3]

CS 201. Program Design and Data Structures. Continuation of CS 101. The study of elementary data structures, their associated algorithms and their application in problems; rigorous development of programming techniques and style; design and implementation of programs with multiple modules, using good data structures and good programming style. Prerequisite: CS 101. FALL, SPRING. [3]

See engineering.vanderbilt.edu/eecs

MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites

CS 279. Software Engineering Project. Students work in teams to specify, design, implement, document, and test a nontrivial software project. The use of CASE (Computer-Assisted Software Engineering) tools is stressed. Prerequisite: CS 278. SPRING. [3]

CS 278. Principles of Software Engineering. The nature of software The object-oriented paradigm. Software life-cycle models. Requirements, specification, design, implementation, documentation, and testing of software. Object-oriented analysis and design. Software maintenance. Prerequisite: CS 251. FALL. [3]

CS 251. Intermediate Software Design. High quality development and reuse of architectural patterns, design patterns, and software components. Theoretical and practical aspects of developing, documenting, testing, and applying reusable class libraries and object-oriented frameworks using object-oriented and component-based programming languages and tools. Prerequisite: CS 201. FALL, SPRING. [3]

CS 101. Programming and Problem Solving. An intensive introduction to algorithm development and problem solving on the computer. Structured problem definition, top down and modular algorithm design. Running, debugging, and testing programs. Program documentation. FALL, SPRING. [3]

CS 282. Principles of Operating Systems II. Projects involving modification of a current operating system. Lectures on memory management policies, including virtual memory. Protection and sharing of information, including general models for implementation of various degrees of sharing. Resource allocation in general, including deadlock detection and prevention strategies. Introduction to operating system performance measurement, for both efficiency and logical correctness. Two hours lecture and one hour laboratory. Prerequisite: CS 281. SPRING. [3]

CS 281. Principles of Operating Systems I. Resource allocation and control functions of operating systems. Scheduling of processes and processors. Concurrent processes and primitives for their synchronization. Use of parallel processes in designing operating system subsystems. Methods of implementing parallel processes on conventional computers. Virtual memory, paging, protection of shared and non-shared information. Structures of data files in secondary storage Security issues. Case studies. Prerequisite: CS 231, CS 251. FALL, SPRING. [3]

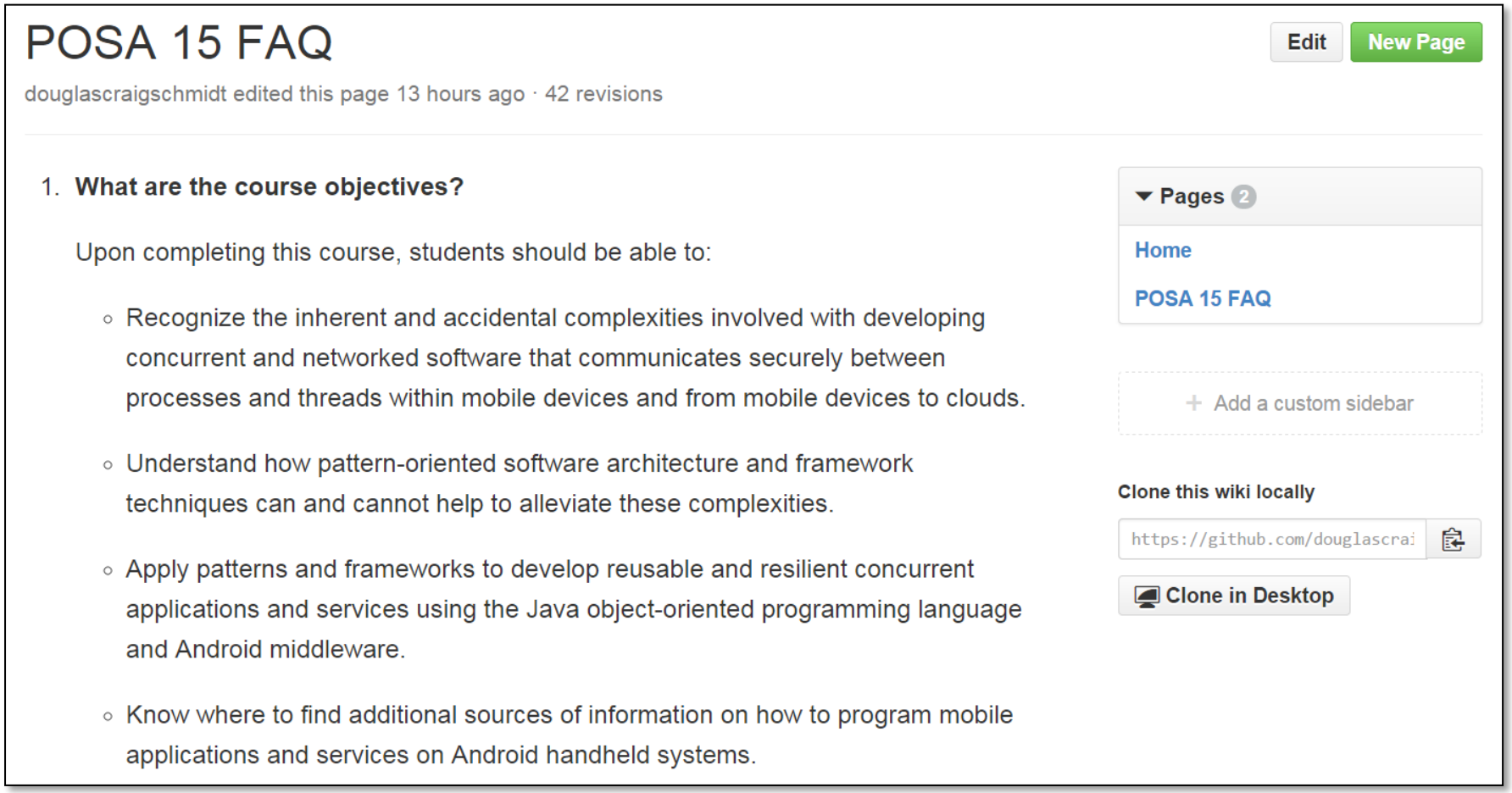
CS 201. Program Design and Data Structures. Continuation of CS 101. The study of elementary data structures, their associated algorithms and their application in problems; rigorous development of programming techniques and style; design and implementation of programs with multiple modules, using good data structures and good programming style. Prerequisite: CS 101. FALL, SPRING. [3]

MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites
- We don't know what you know or whether you're prepared or not!

MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites
- We don't know what you know or whether you're prepared or not!
- This MOOC has assumptions about—& expectations of—the students



POSA 15 FAQ

Edit | New Page

douglascraigschmidt edited this page 13 hours ago · 42 revisions

1. **What are the course objectives?**

   Upon completing this course, students should be able to:

   - Recognize the inherent and accidental complexities involved with developing concurrent and networked software that communicates securely between processes and threads within mobile devices and from mobile devices to clouds.
   - Understand how pattern-oriented software architecture and framework techniques can and cannot help to alleviate these complexities.
   - Apply patterns and frameworks to develop reusable and resilient concurrent applications and services using the Java object-oriented programming language and Android middleware.
   - Know where to find additional sources of information on how to program mobile applications and services on Android handheld systems.

Pages 2

Home

POSA 15 FAQ

+ Add a custom sidebar

Clone this wiki locally

https://github.com/douglascrai

Clone in Desktop

See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #5

MOOC “Prerequisites”

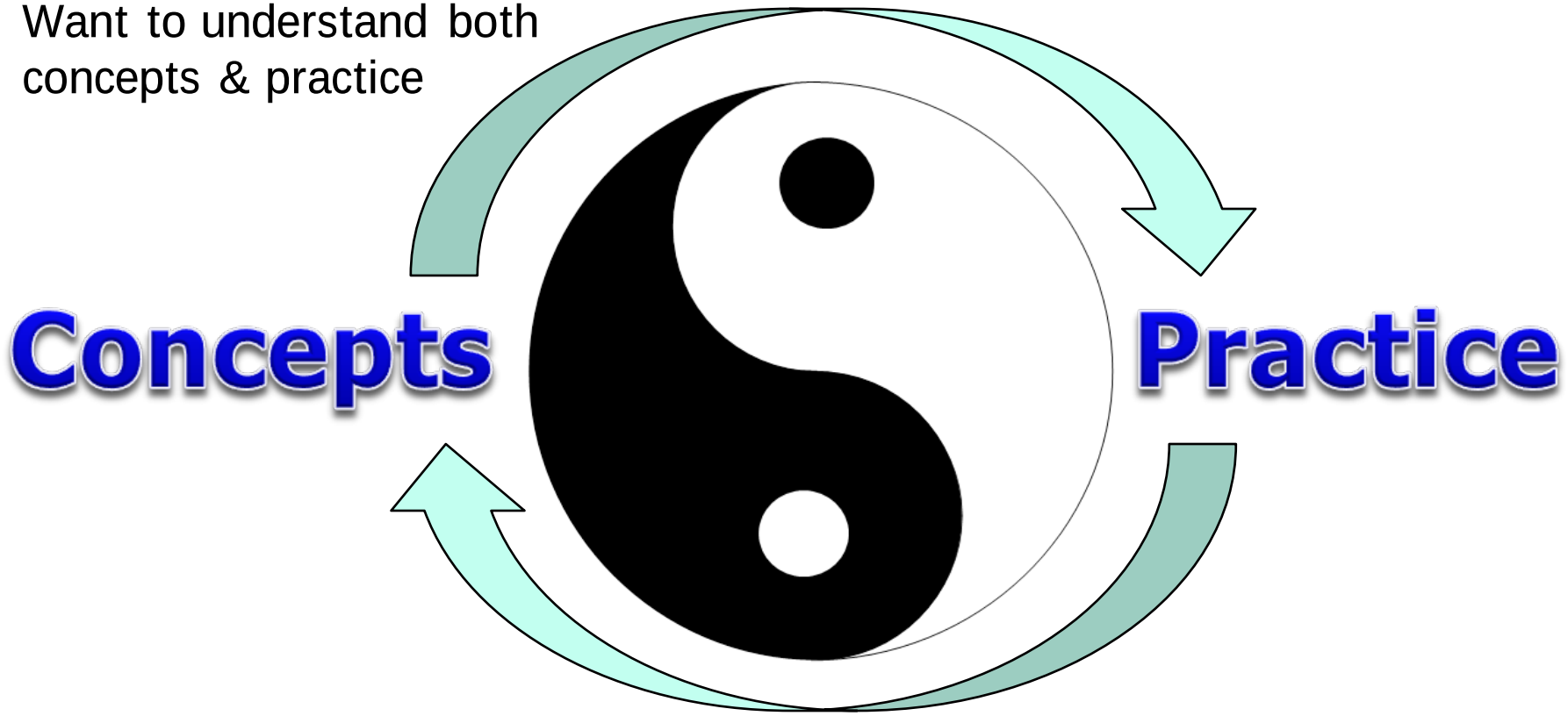
- We know students at our universities have taken the course prerequisites
- We don’t know what you know or whether you’re prepared or not!
- This MOOC has assumptions about—& expectations of—the students

MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites
- We don't know what you know or whether you're prepared or not!
- This MOOC has assumptions about—& expectations of—the students
  - Know Java, Eclipse, & Git

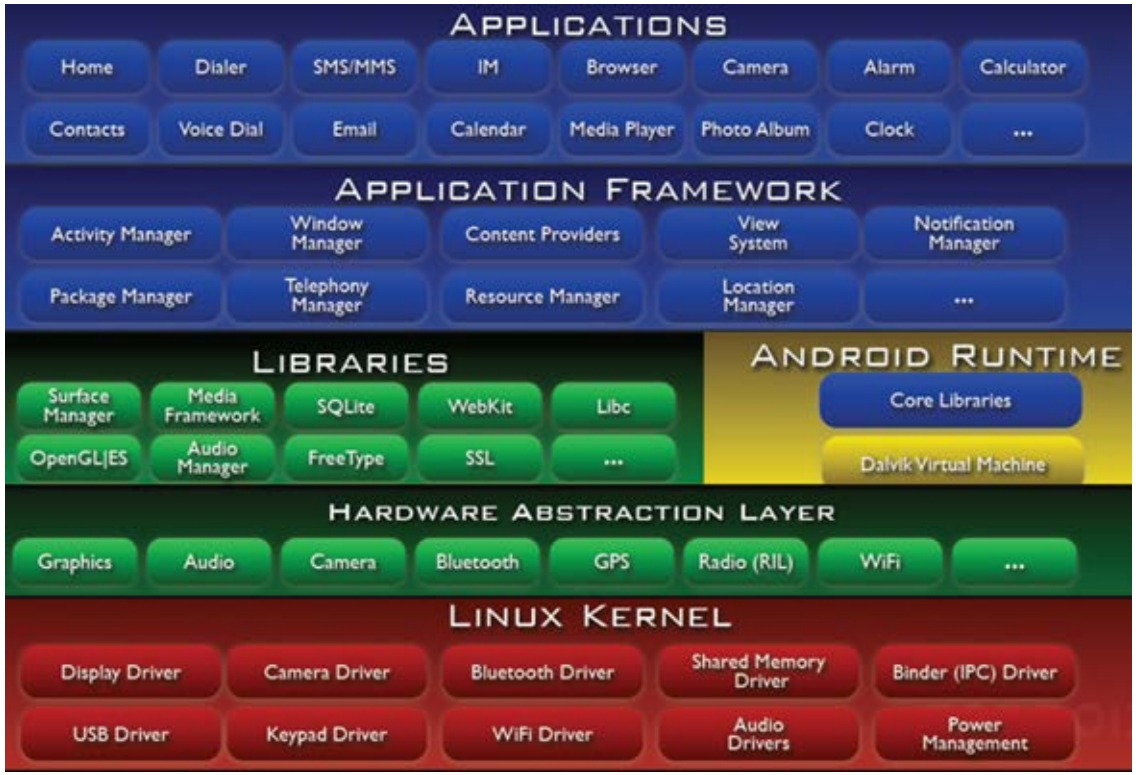
MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites
- We don't know what you know or whether you're prepared or not!
- This MOOC has assumptions about—& expectations of—the students
  - Know Java, Eclipse, & Git
  - Want to understand both concepts & practice



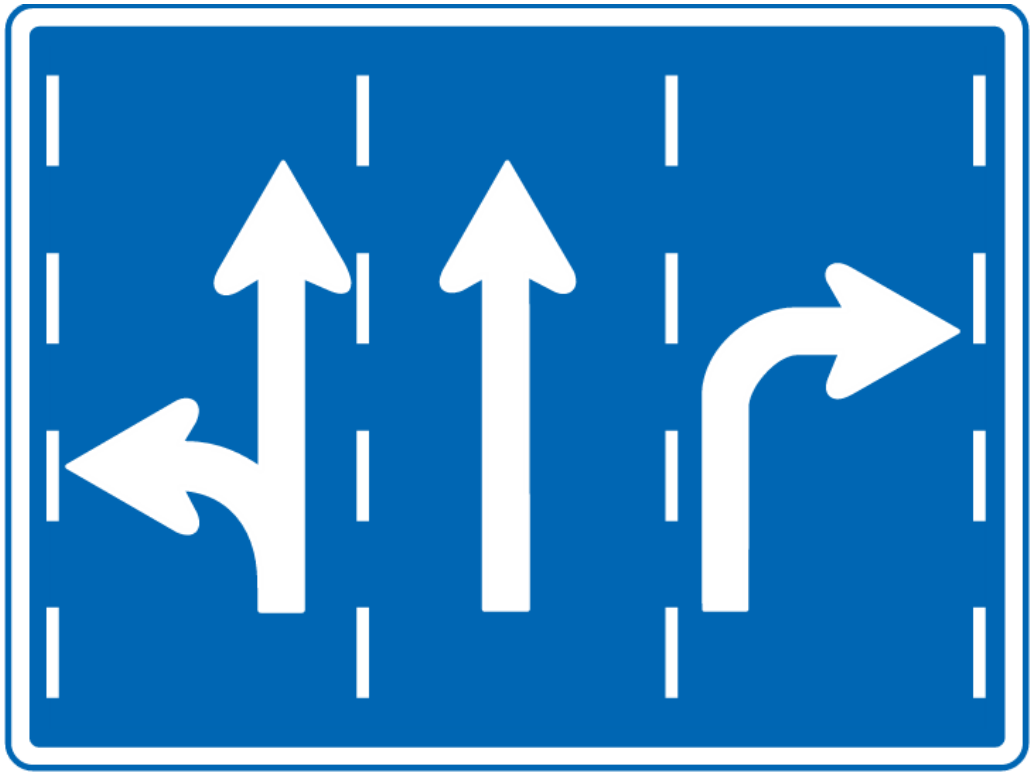
MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites
- We don't know what you know or whether you're prepared or not!
- This MOOC has assumptions about—& expectations of—the students
  - Know Java, Eclipse, & Git
  - Want to understand both concepts & practice
  - Are curious about the Android software stack



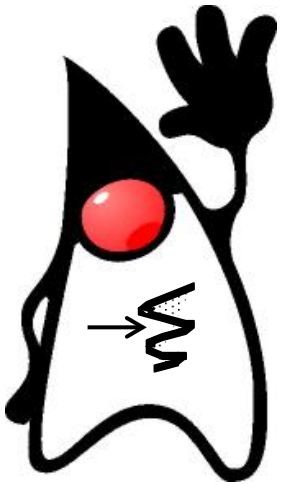
MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites
- We don't know what you know or whether you're prepared or not!
- This MOOC has assumptions about—& expectations of—the students
  - Know Java, Eclipse, & Git
  - Want to understand both concepts & practice
  - Are curious about the Android software stack
  - Are willing to read & follow the instructions



MOOC "Prerequisites"

- We know students at our universities have taken the course prerequisites
- We don't know what you know or whether you're prepared or not!
- This MOOC has assumptions about—& expectations of—the students
- Concurrency is not an easy subject to master



See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #38

What We'd Like Students to Know (Part 1)

What We'd Like Students to Know

- Ideally, students know certain things

What We'd Like Students to Know

- Ideally, students know certain things
  - **OO programming languages**

```
public class EventHandler
    extends Observer {
  public void update(Observable o,
                     Object arg)
  { /*…*/ }
    …

public class EventSource
    extends Observable,
    implements Runnable {
  public void run()
  { /*…*/ notifyObservers(/*…*/); }
    …

EventSource source =
        new EventSource();
EventHandler handler =
        new EventHandler();
eventSource.addObserver(handler);
Thread thread =
   new Thread(eventSource);
thread.start();
…
```

See en.wikipedia.org/wiki/Object-oriented_programming

What We'd Like Students to Know

- Ideally, students know certain things
  - **OO programming languages**


```
public class EventHandler
    extends Observer {
  public void update(Observable o,
                     Object arg)
  { /*…*/ }
    …

public class EventSource
    extends Observable,
    implements Runnable {
  public void run()
  { /*…*/ notifyObservers(/*…*/); }
    …

EventSource source =
        new EventSource();
EventHandler handler =
        new EventHandler();
eventSource.addObserver(handler);
Thread thread =
   new Thread(eventSource);
thread.start();
…
```

See en.wikipedia.org/wiki/Java_(programming_language)

What We'd Like Students to Know

- Ideally, students know certain things
  - **OO programming languages**
    - e.g., Java classes, inheritance, dynamic binding, generics, dynamic memory allocation, garbage collection, etc.

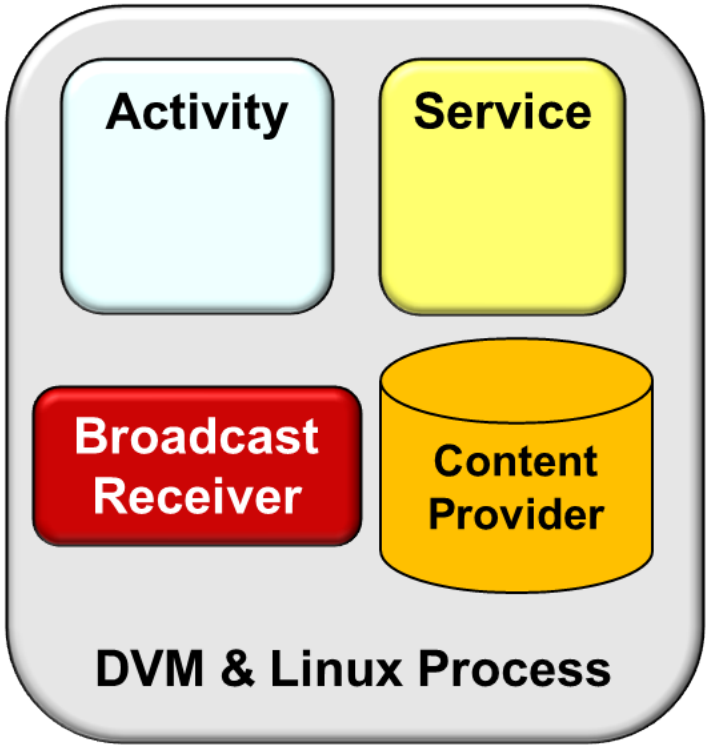
```
public class EventHandler
    extends Observer {
  public void update(Observable o,
                     Object arg)
  { /*…*/ }
    …

public class EventSource
    extends Observable,
    implements Runnable {
  public void run()
  { /*…*/ notifyObservers(/*…*/); }
    …

EventSource source =
        new EventSource();
EventHandler handler =
        new EventHandler();
eventSource.addObserver(handler);
Thread thread =
   new Thread(eventSource);
thread.start();
…
```

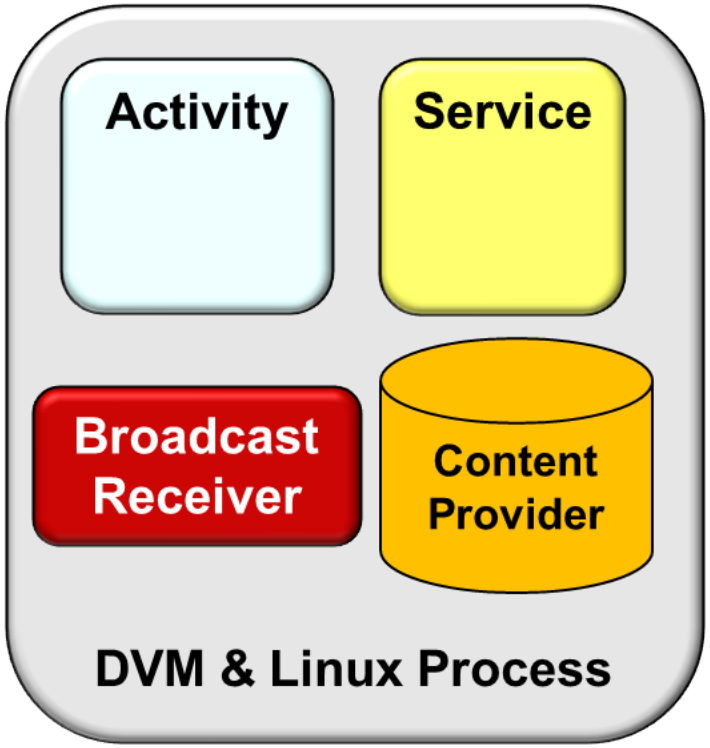
What We'd Like Students to Know

- Ideally, students know certain things
  - OO programming languages
  - **Android programming elements**



What We'd Like Students to Know

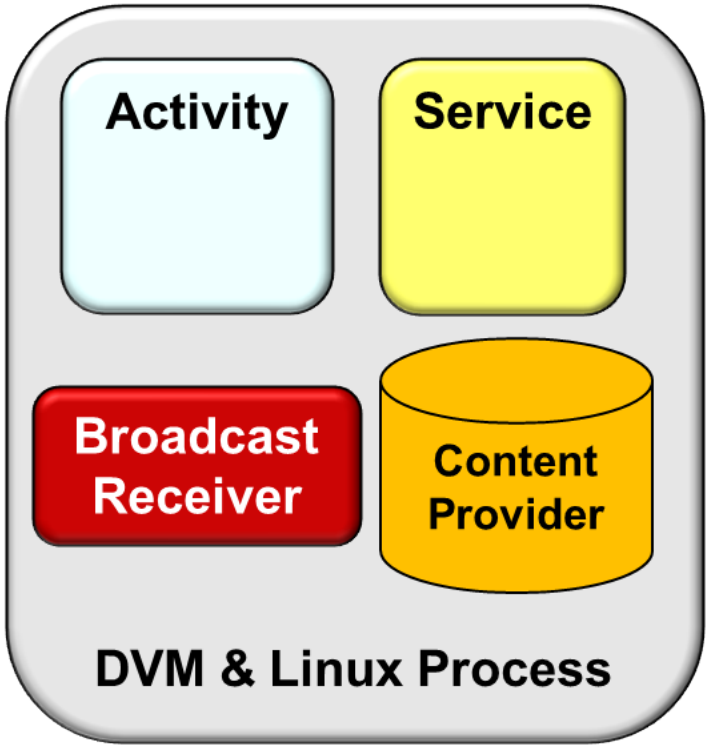
- Ideally, students know certain things
  - OO programming languages
  - **Android programming elements, e.g.**
    - Activities, Intents, UI components



See www.coursera.org/course/androidpart1 & www.coursera.org/course/androidpart2

What We'd Like Students to Know

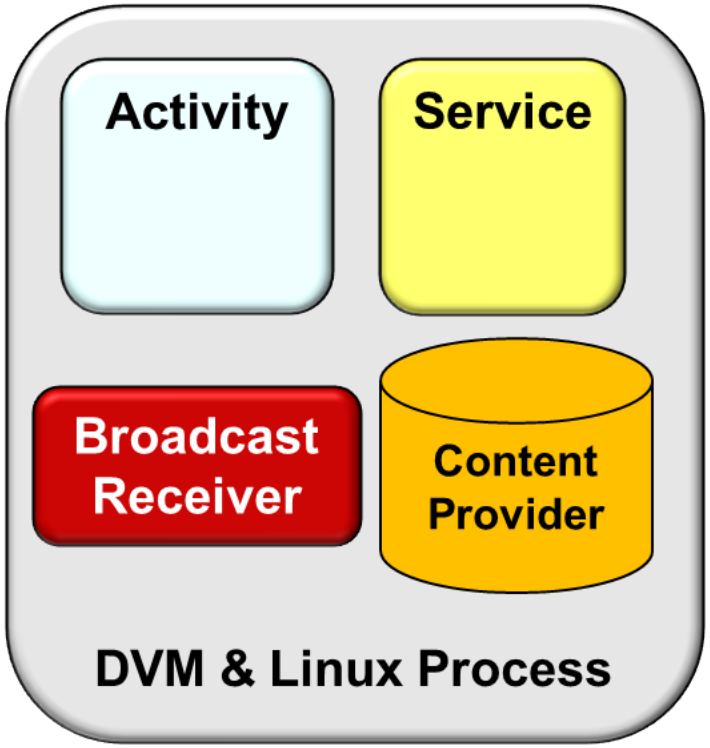
- Ideally, students know certain things
  - OO programming languages
  - **Android programming elements, e.g.**
    - Activities, Intents, UI components
    - Eclipse ADT



See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #7

What We'd Like Students to Know

- Ideally, students know certain things
  - OO programming languages
  - **Android programming elements, e.g.**
    - Activities, Intents, UI components
    - Eclipse ADT

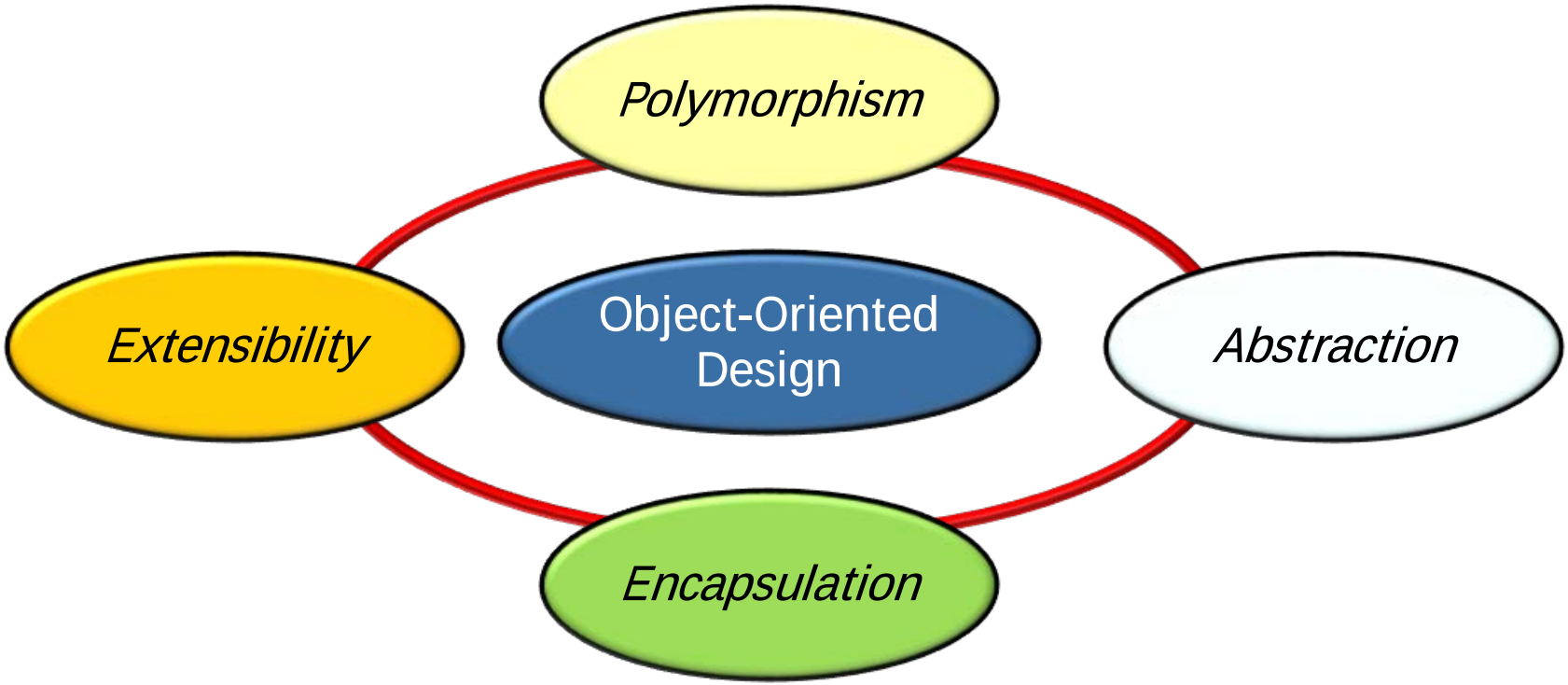


See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #48

What We'd Like Students to Know (Part 2)

What We'd Like Students to Know

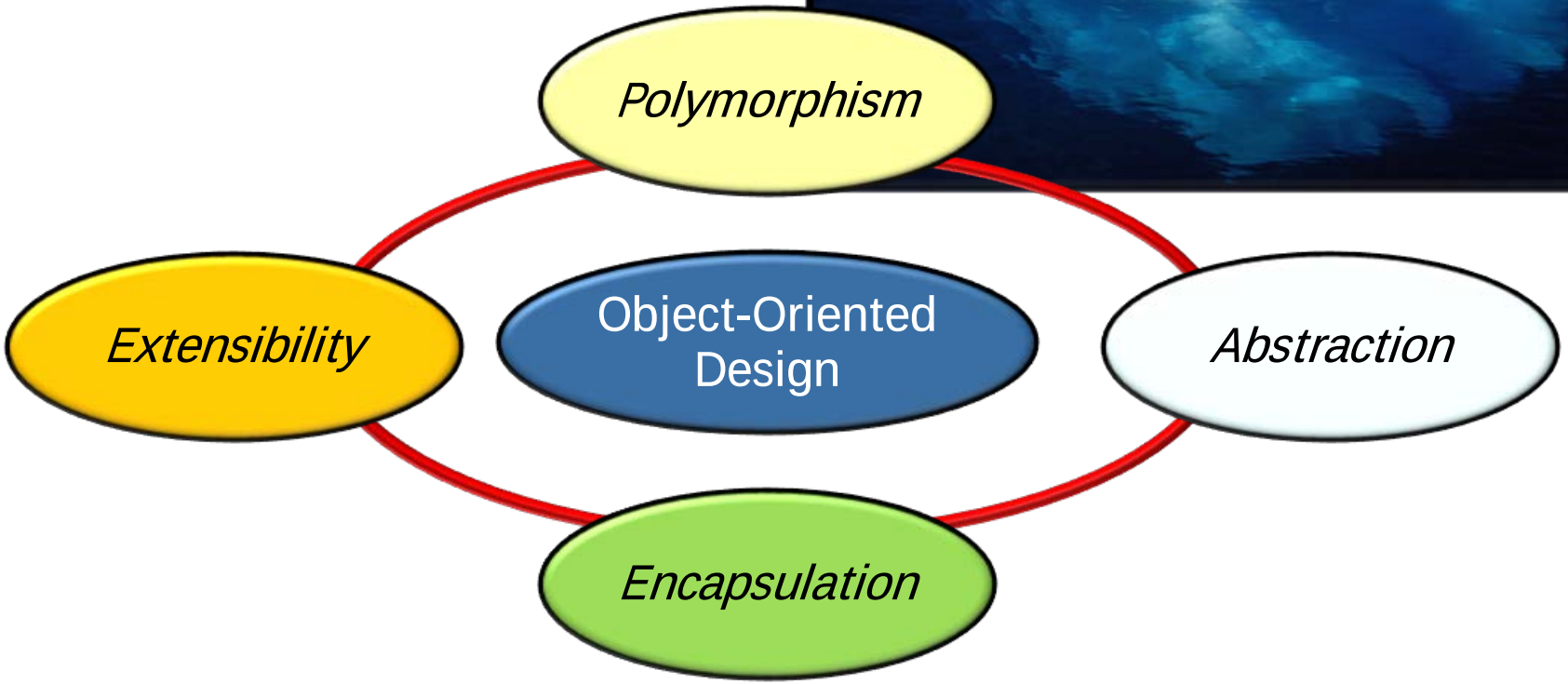
- Ideally, students know certain things
  - OO programming languages
  - Android programming elements
  - **OO design concepts & notations**



See en.wikipedia.org/wiki/Object-oriented_design

What We'd Like Students to Know

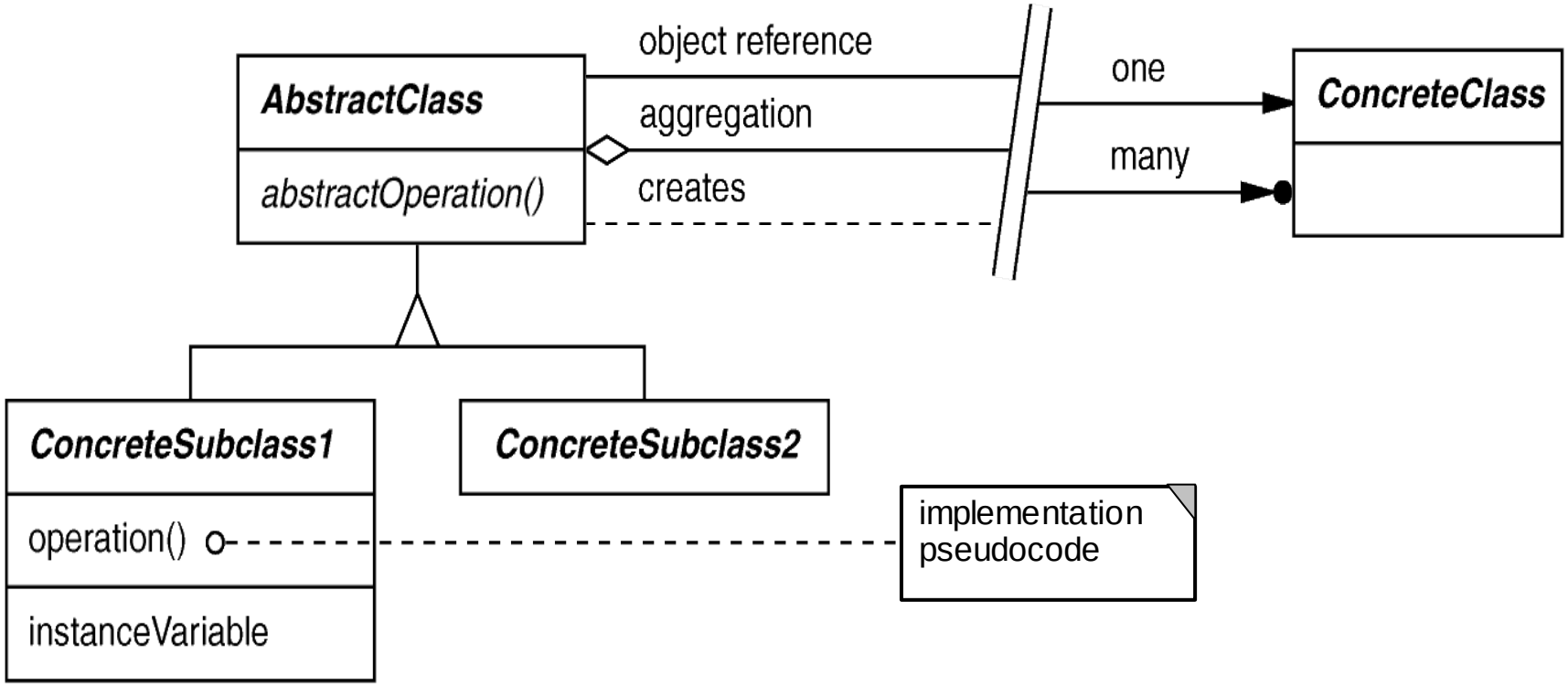
- Ideally, students know certain things
  - OO programming languages
  - Android programming elements
  - **OO design concepts & notations**



See en.wikipedia.org/wiki/Object-oriented_design

What We'd Like Students to Know

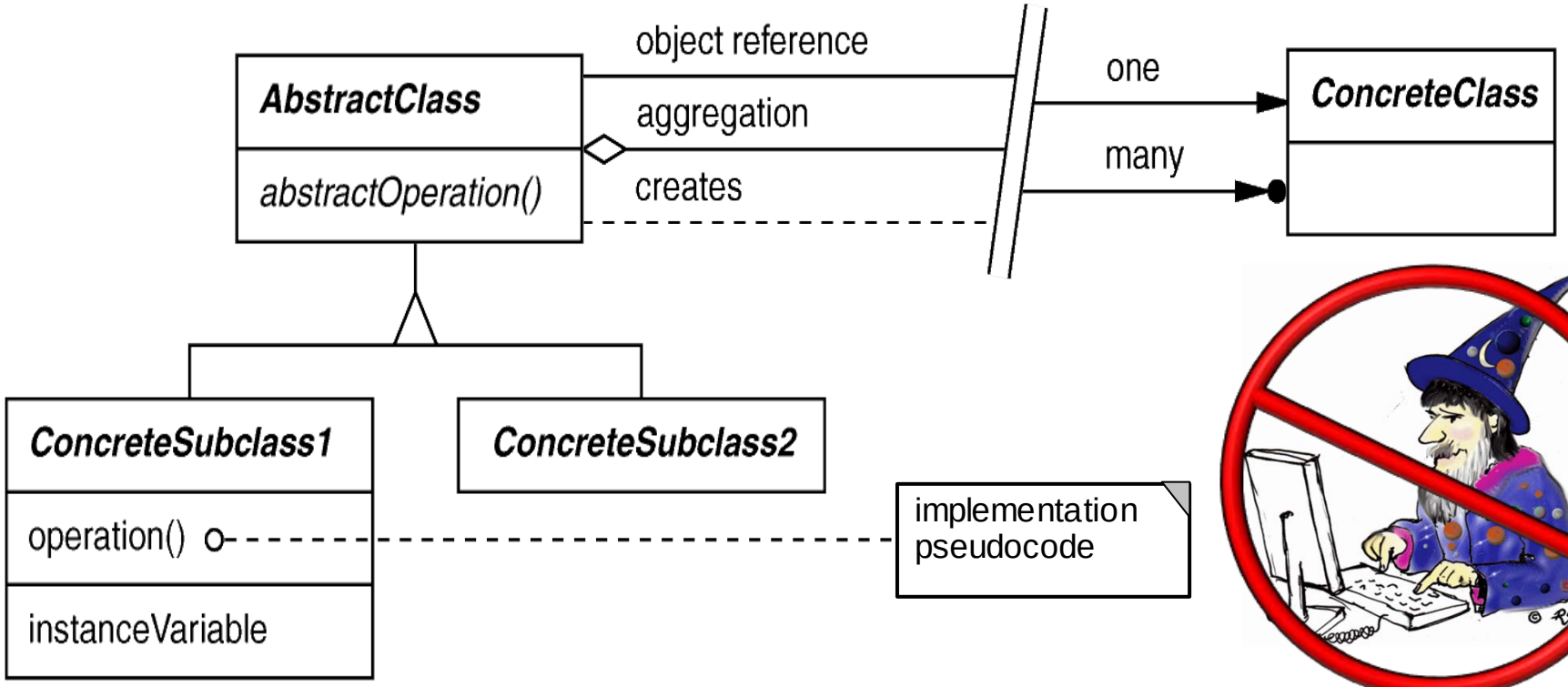
- Ideally, students know certain things
  - OO programming languages
  - Android programming elements
  - **OO design concepts & notations**



See en.wikipedia.org/wiki/Unified_Modeling_Language

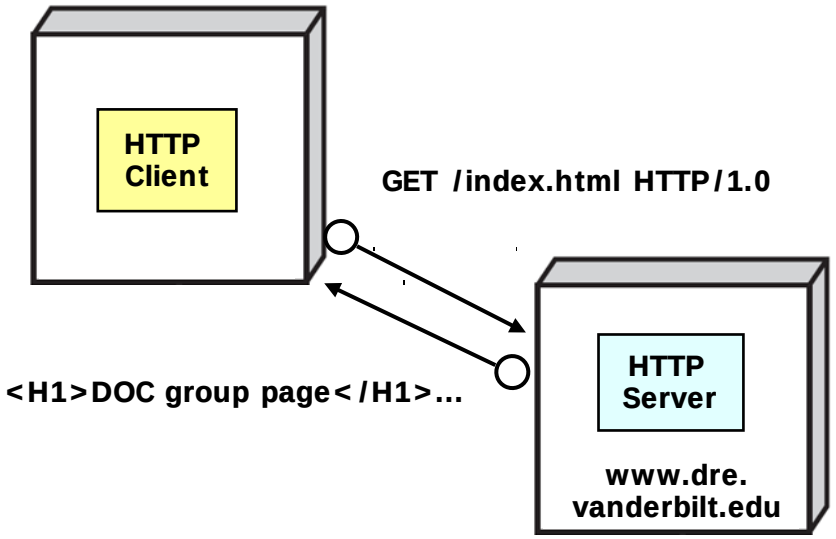
What We'd Like Students to Know

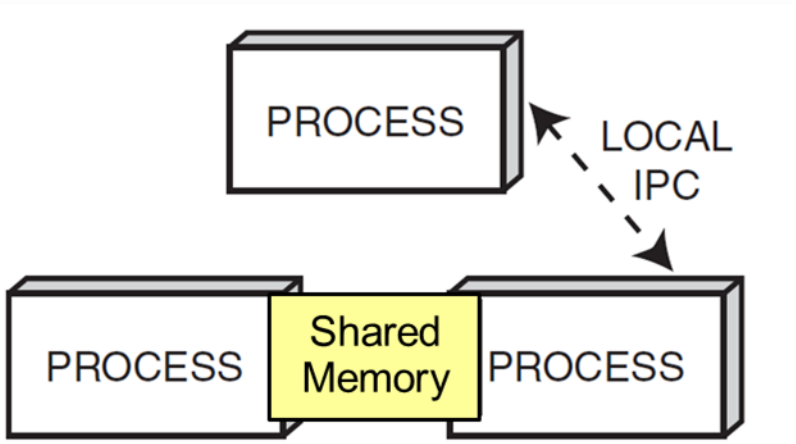
- Ideally, students know certain things
  - OO programming languages
  - Android programming elements
  - **OO design concepts & notations**

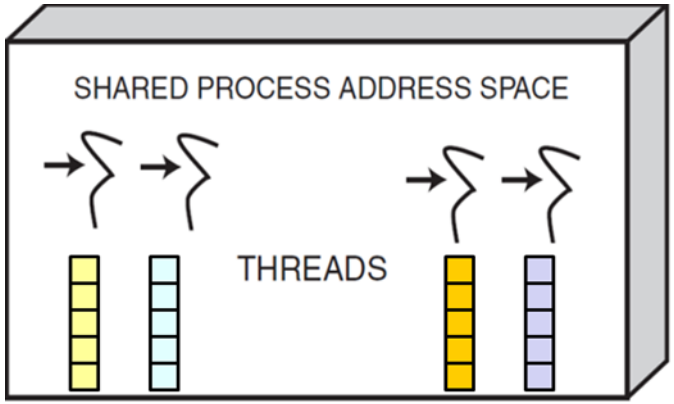


What We'd Like Students to Know

- Ideally, students know certain things
  - OO programming languages
  - Android programming elements
  - OO design concepts & notations
  - **Systems & network programming concepts**

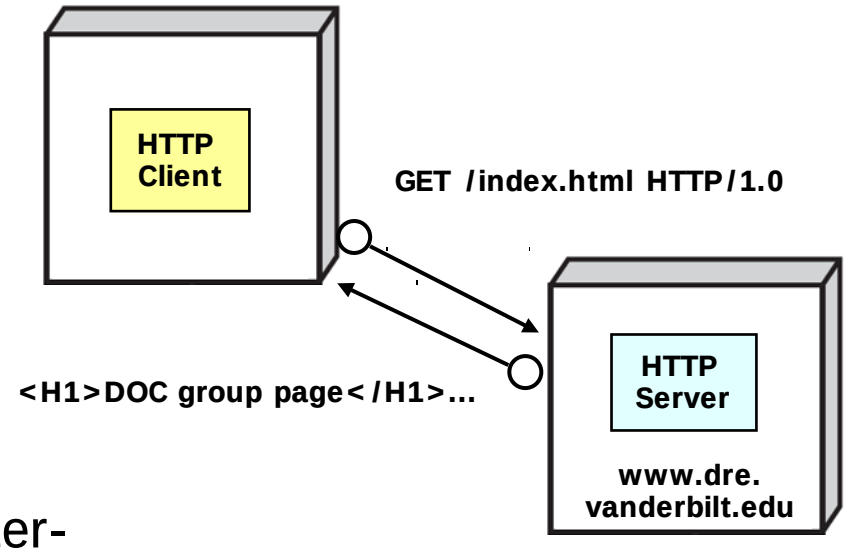


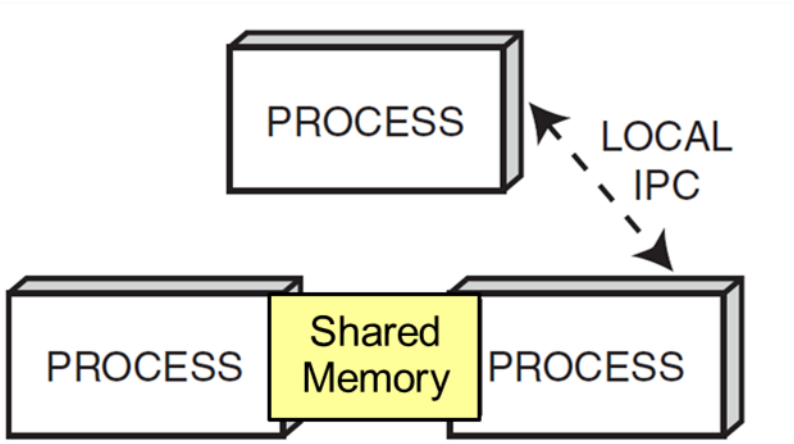


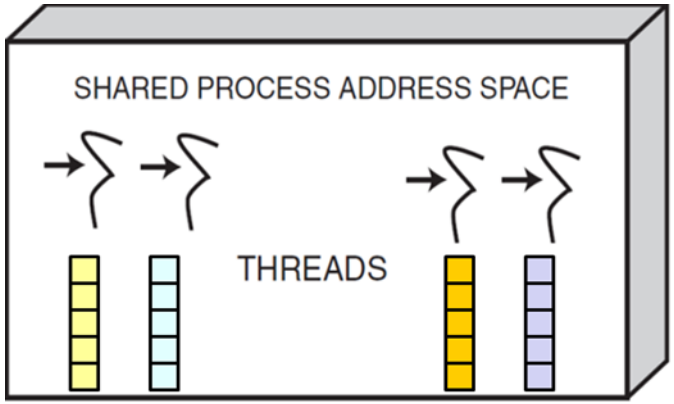


What We'd Like Students to Know

- Ideally, students know certain things
  - OO programming languages
  - Android programming elements
  - OO design concepts & notations
  - **Systems & network programming concepts**
    - e.g., event loops, multi-threading & -processing, synchronization, & inter-process communication (IPC)





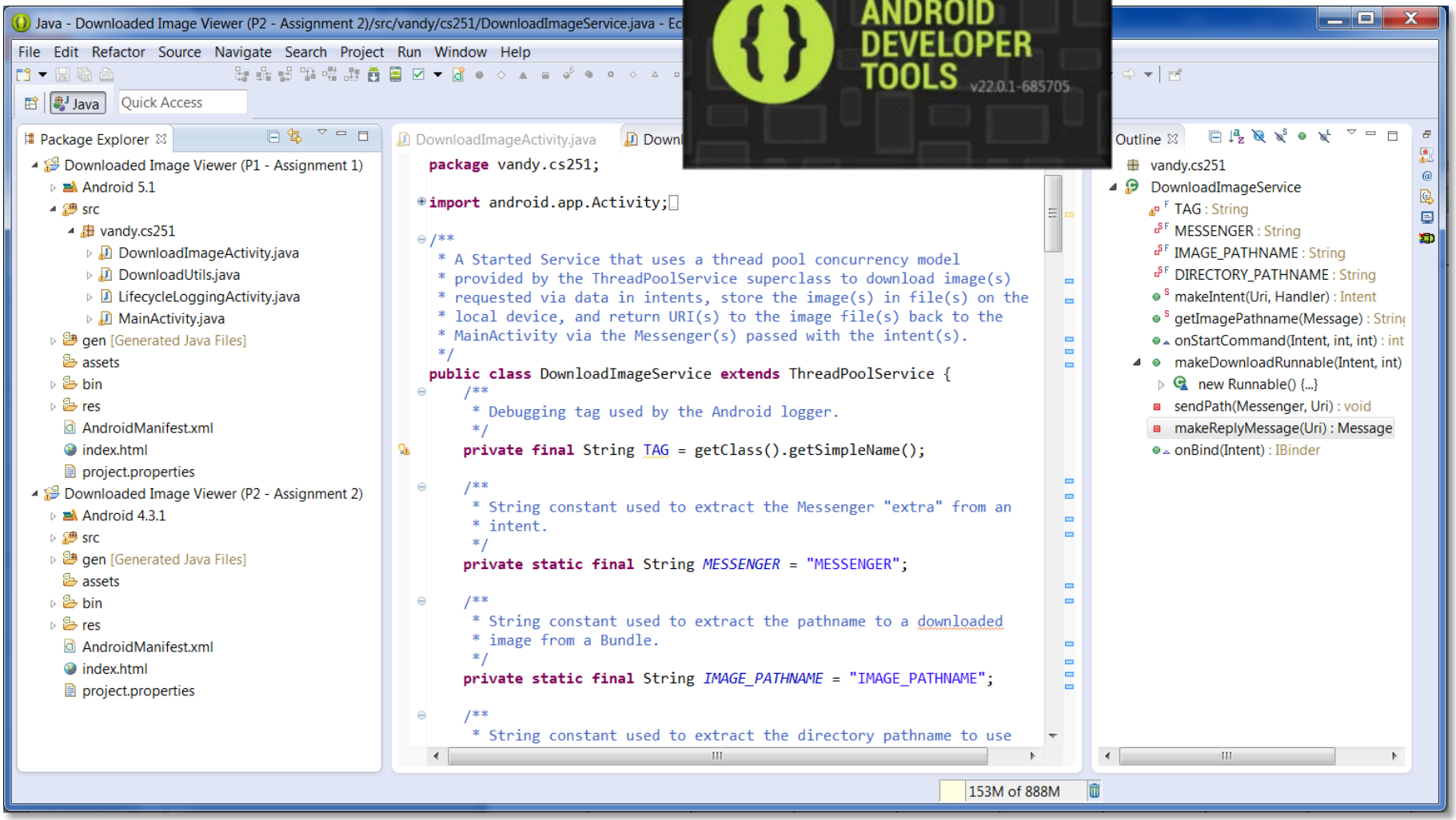


See class.coursera.org/posa-001/lecture Section 1 videos

Overview of the Assignments & Assessments

Overview of Assignments & Assessments

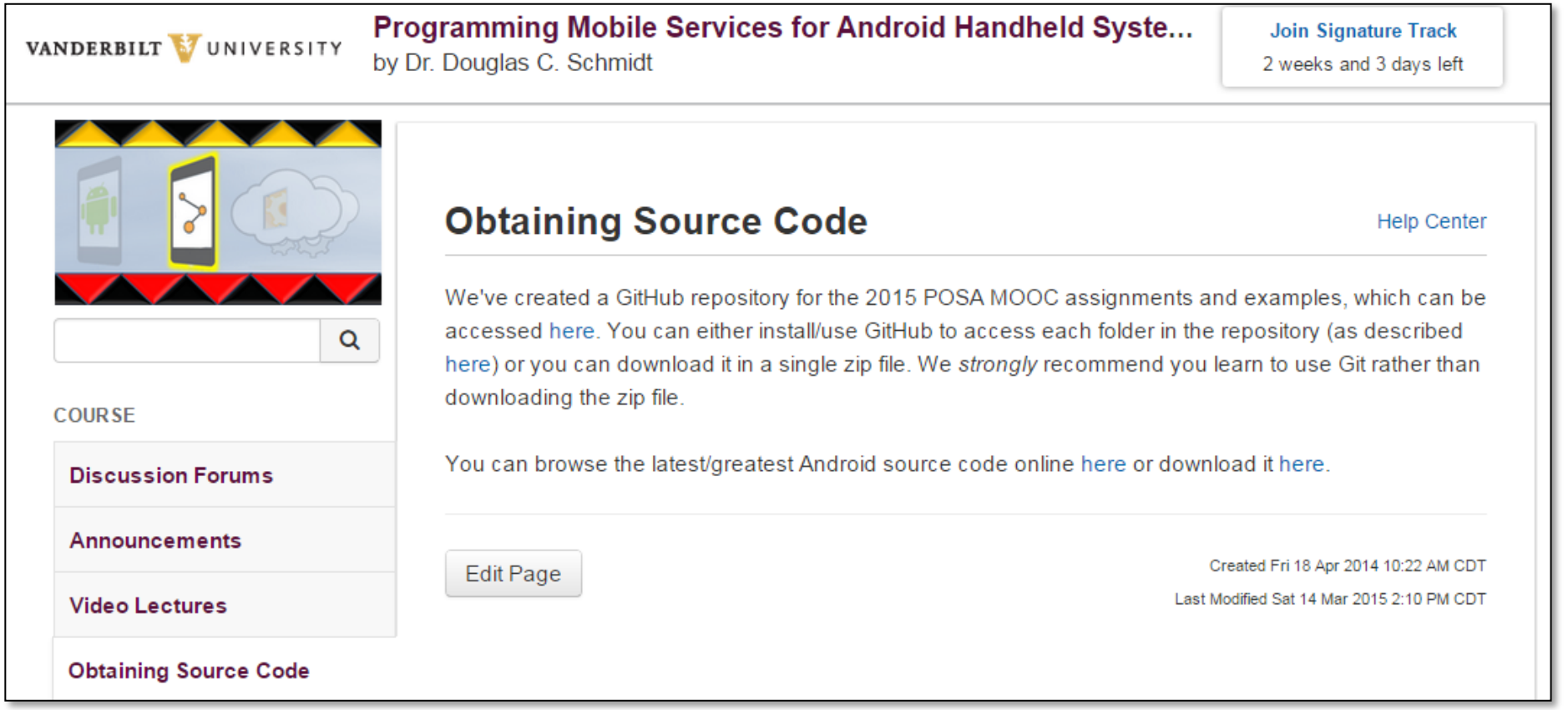
- Programming assignments should be written in Java using Eclipse



See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #23

Overview of Assignments & Assessments

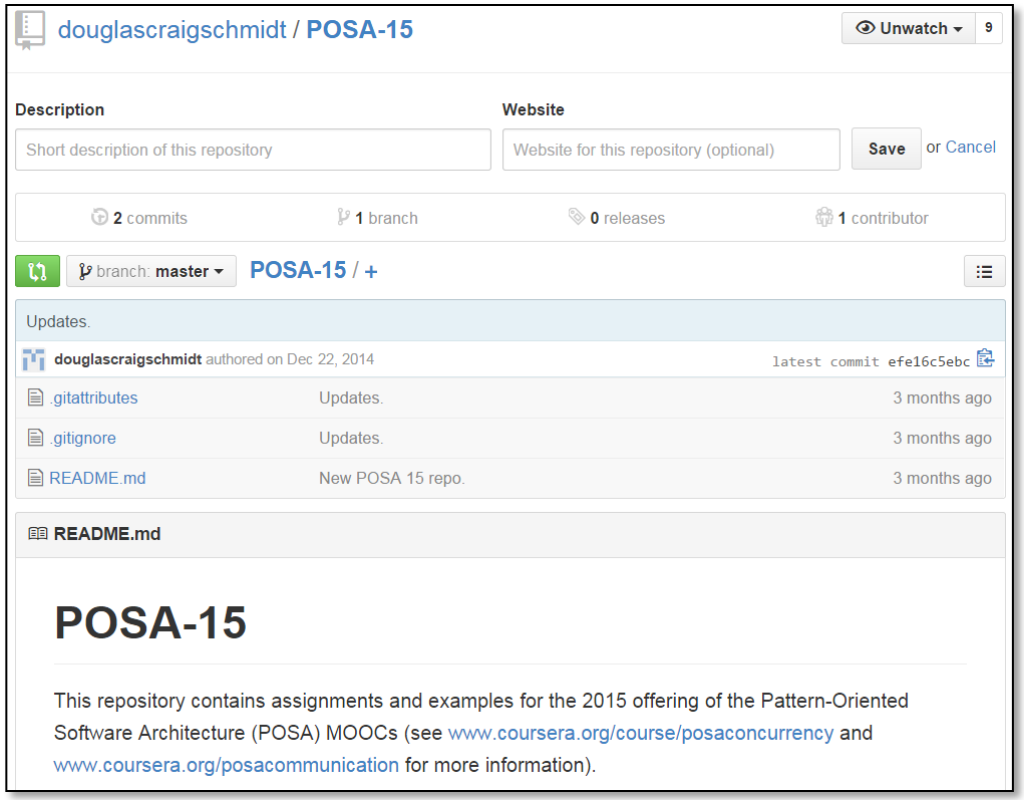
- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online



See class.coursera.org/posaconcurrency-001/wiki/Source_Code

Overview of Assignments & Assessments

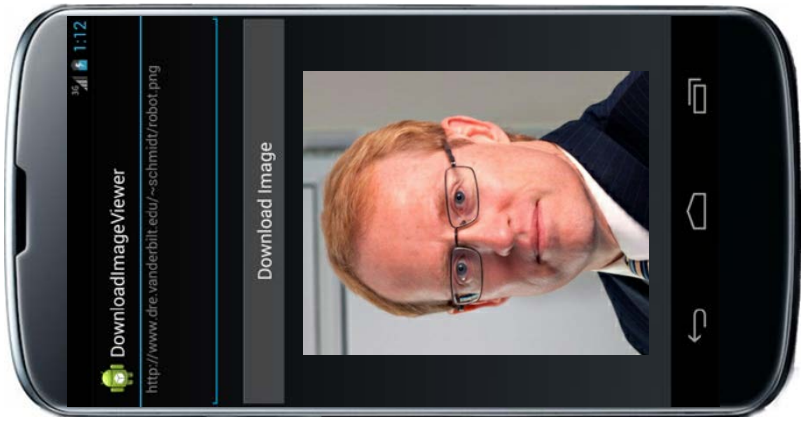
- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming

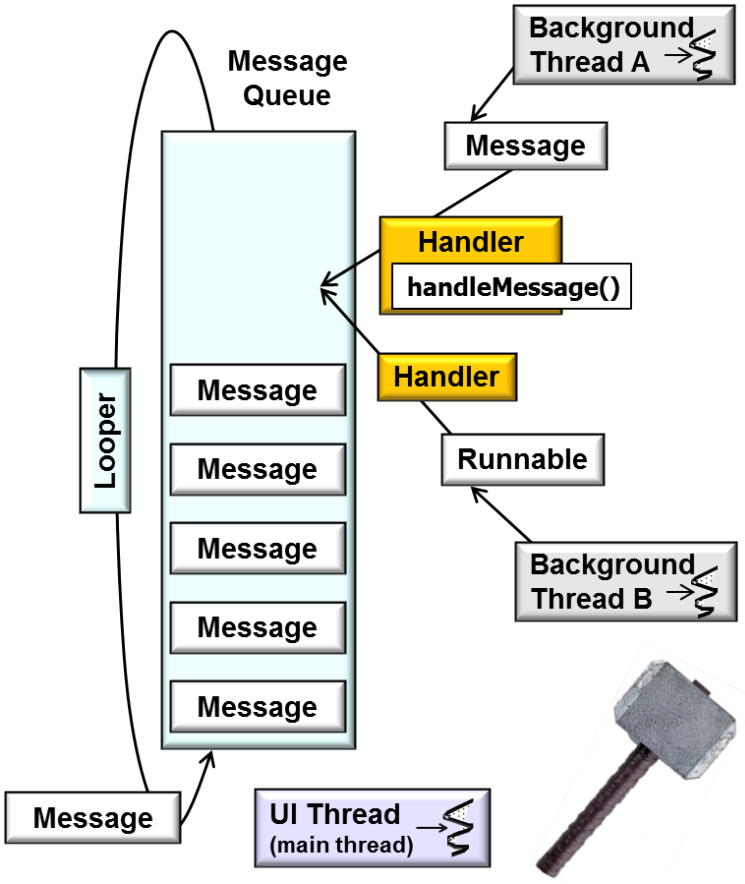


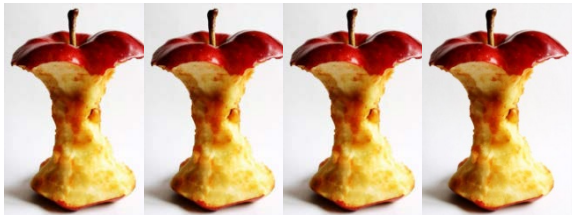
See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #10

Overview of Assignments & Assessments

- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming, e.g.
  - Implement an app to download, store, & display images

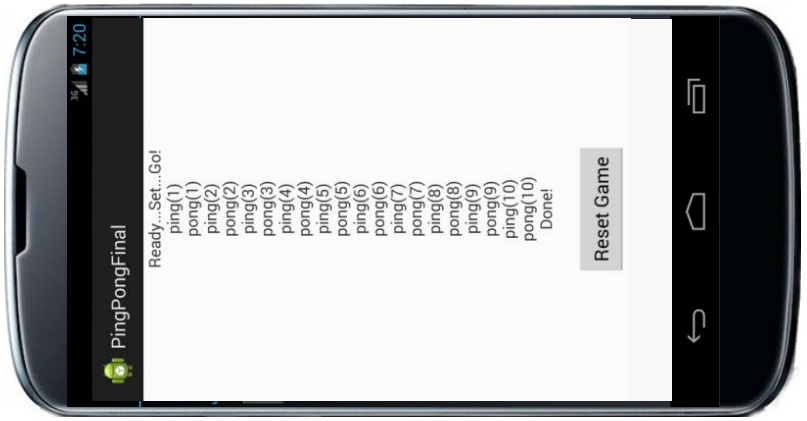


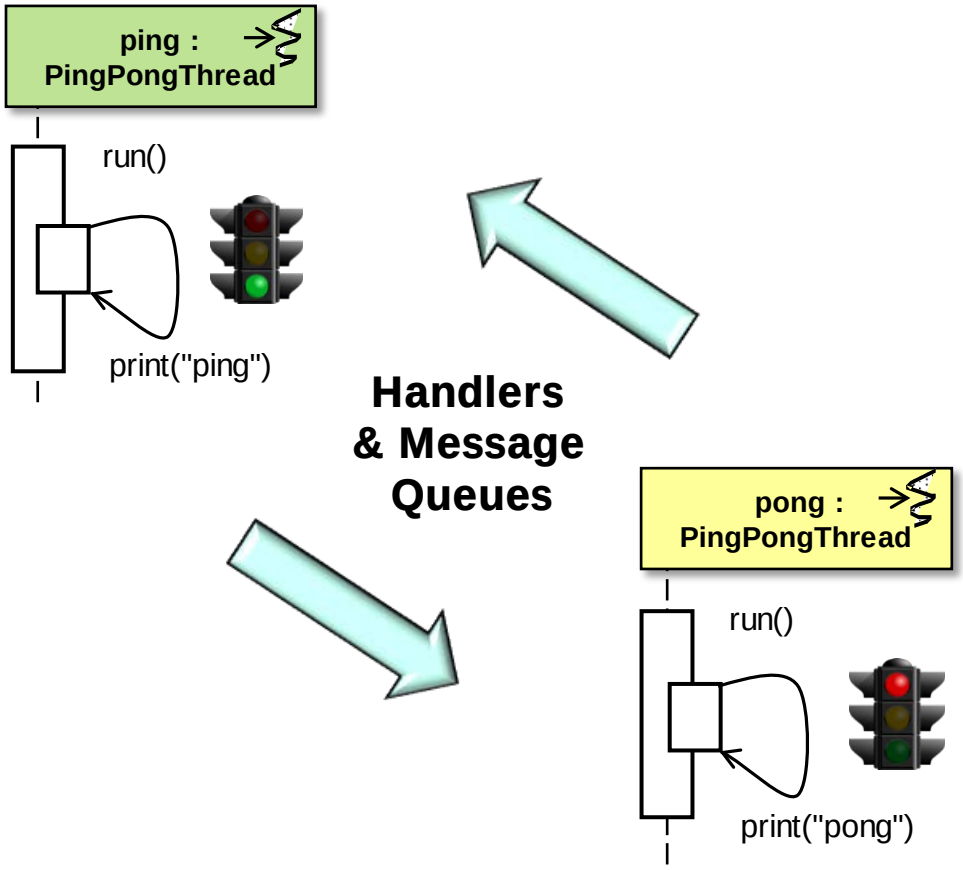


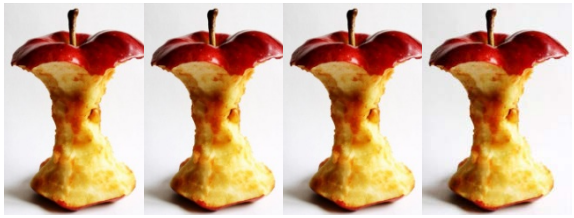


Overview of Assignments & Assessments

- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming, e.g.
  - Implement an app to download, store, & display images
  - Implement a “pong-pong” app

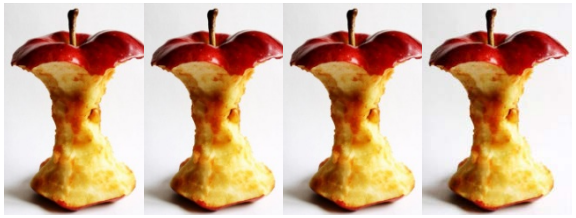






Overview of Assignments & Assessments

- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming, e.g.
  - Implement an app to download, store, & display images
  - Implement a "pong-pong" app
  - Implement a concurrent version of the "Daily Selfie" app



See www.coursera.org/course/androidpart2

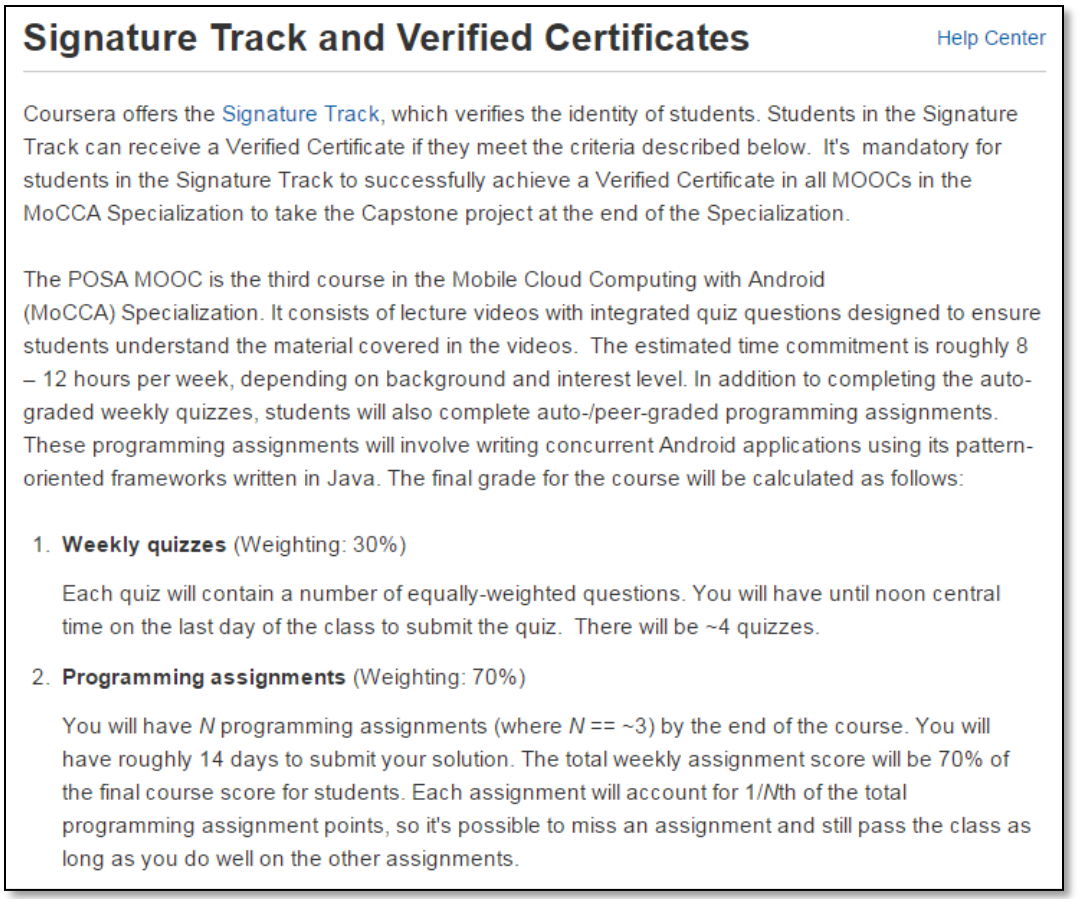
Overview of Assignments & Assessments

- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming
- Assessments will be done via auto-grading & peer-grading

See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #34

Overview of Assignments & Assessments

- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming
- Assessments will be done via auto-grading & peer-grading
- Verified Certificates require doing assignments & quizzes



Signature Track and Verified Certificates

Help Center

Coursera offers the Signature Track, which verifies the identity of students. Students in the Signature Track can receive a Verified Certificate if they meet the criteria described below. It's mandatory for students in the Signature Track to successfully achieve a Verified Certificate in all MOOCs in the MoCCA Specialization to take the Capstone project at the end of the Specialization.

The POSA MOOC is the third course in the Mobile Cloud Computing with Android (MoCCA) Specialization. It consists of lecture videos with integrated quiz questions designed to ensure students understand the material covered in the videos. The estimated time commitment is roughly 8 – 12 hours per week, depending on background and interest level. In addition to completing the auto-graded weekly quizzes, students will also complete auto-/peer-graded programming assignments. These programming assignments will involve writing concurrent Android applications using its pattern-oriented frameworks written in Java. The final grade for the course will be calculated as follows:

1. **Weekly quizzes** (Weighting: 30%)

   Each quiz will contain a number of equally-weighted questions. You will have until noon central time on the last day of the class to submit the quiz. There will be ~4 quizzes.

2. **Programming assignments** (Weighting: 70%)

   You will have *N* programming assignments (where *N* == ~3) by the end of the course. You will have roughly 14 days to submit your solution. The total weekly assignment score will be 70% of the final course score for students. Each assignment will account for 1/*N*th of the total programming assignment points, so it's possible to miss an assignment and still pass the class as long as you do well on the other assignments.

See class.MOOCra.org/posaconcurrency-001/wiki/verified_certificate

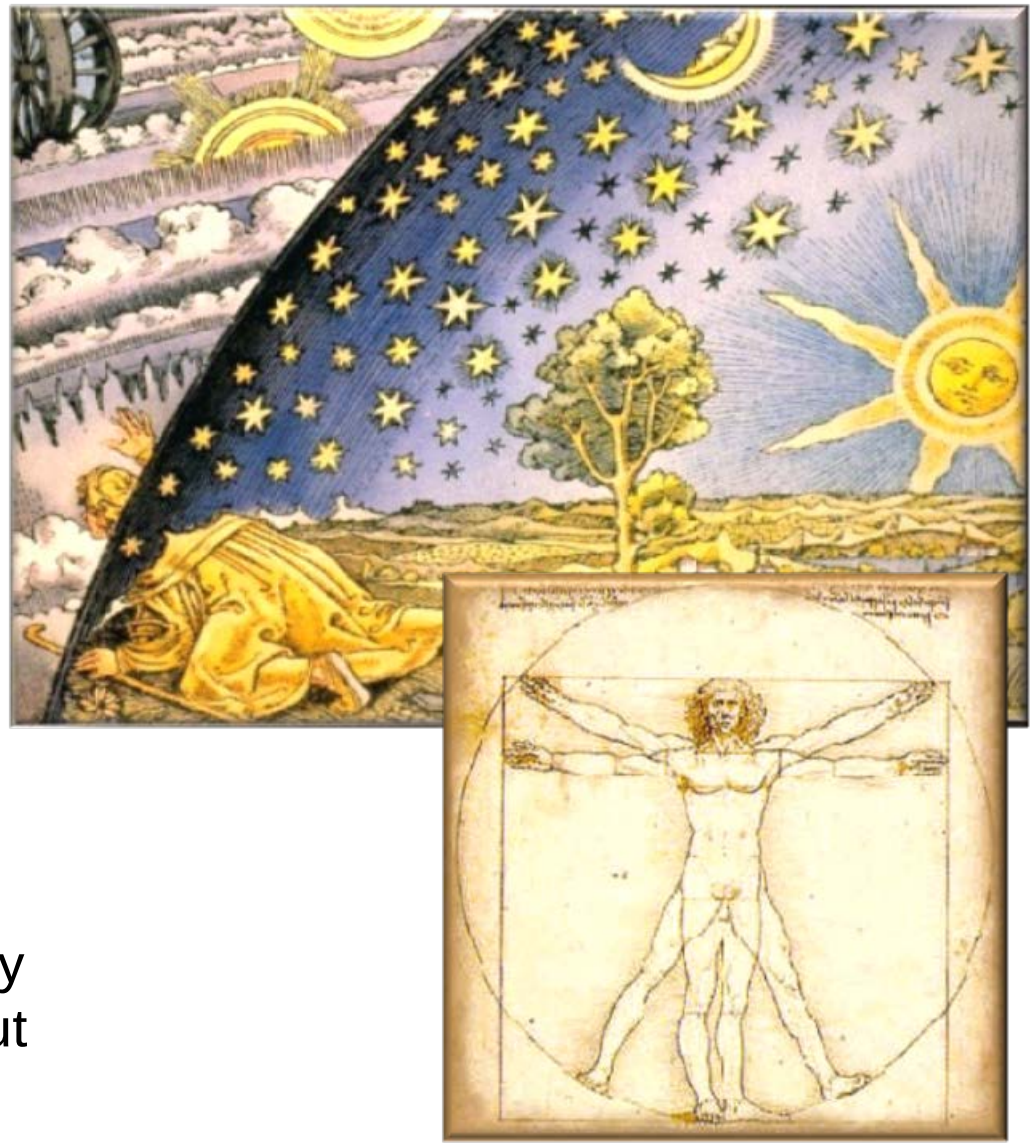
Overview of Assignments & Assessments

- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming
- Assessments will be done via auto-grading & peer-grading
- Verified Certificates require doing assignments & quizzes
- It's hard to precisely estimate the amount of time for this MOOC

See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #38

Overview of Assignments & Assessments

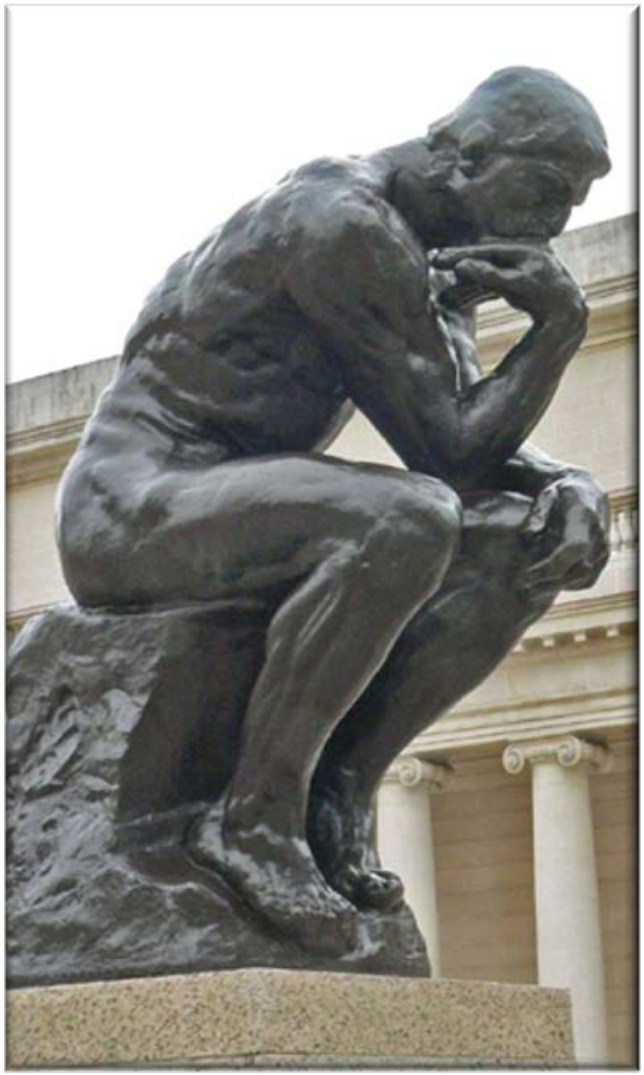
- Programming assignments should be written in Java using Eclipse
- Source code for assignments, examples, & Android is online
- Assignments provide a range of experience with concurrent Android programming
- Assessments will be done via auto-grading & peer-grading
- Verified Certificates require doing assignments & quizzes
- It's hard to precisely estimate the amount of time for this MOOC
- You can also take this MOOC solely to expand your knowledge, without submitting quizzes or assignments



Strategies for Learning All this Material (Part 1)

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following



There are some strategies to consider when taking the POSA MOOCs

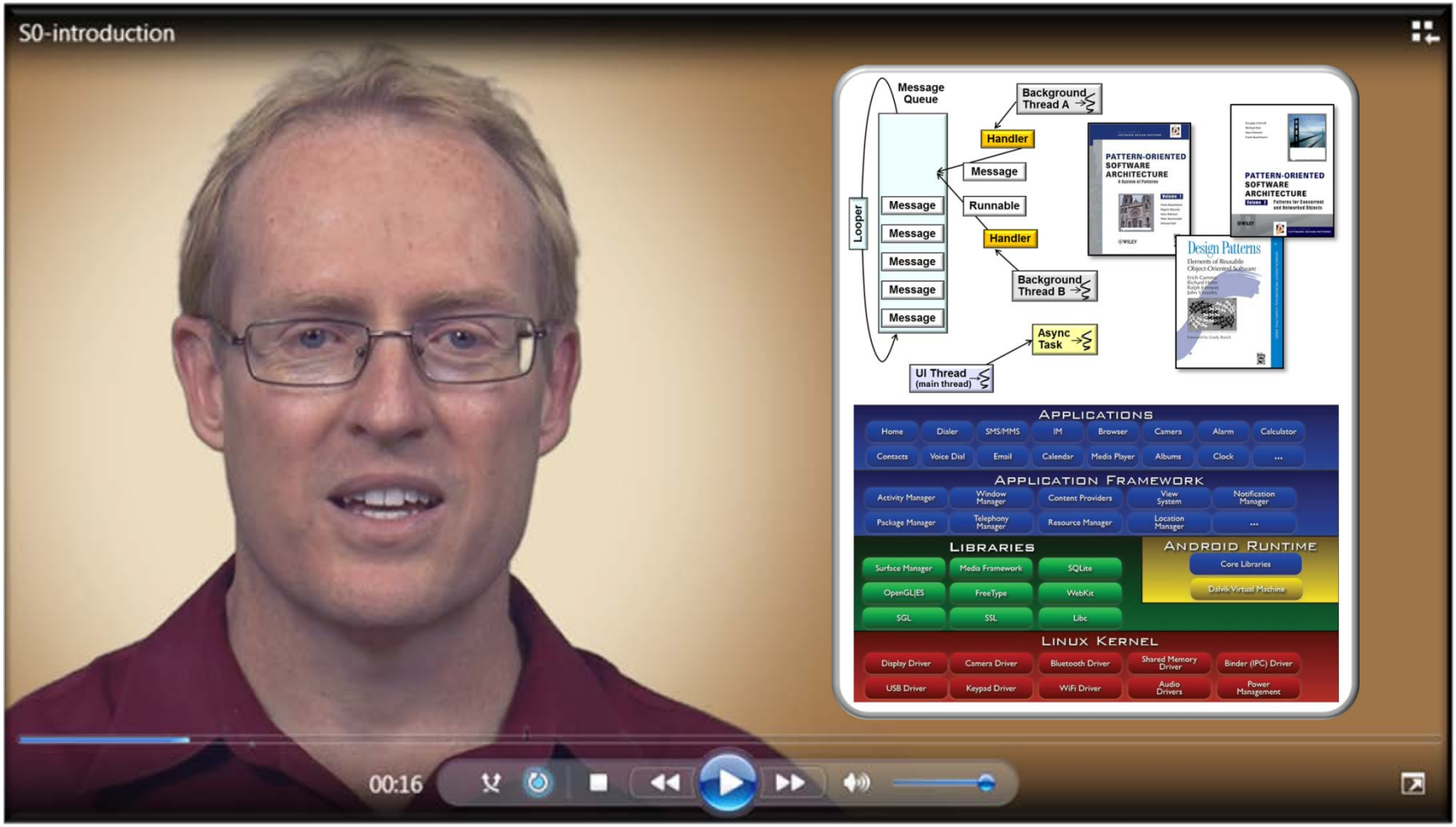
Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times

You may want to watch the videos at multiple speeds!

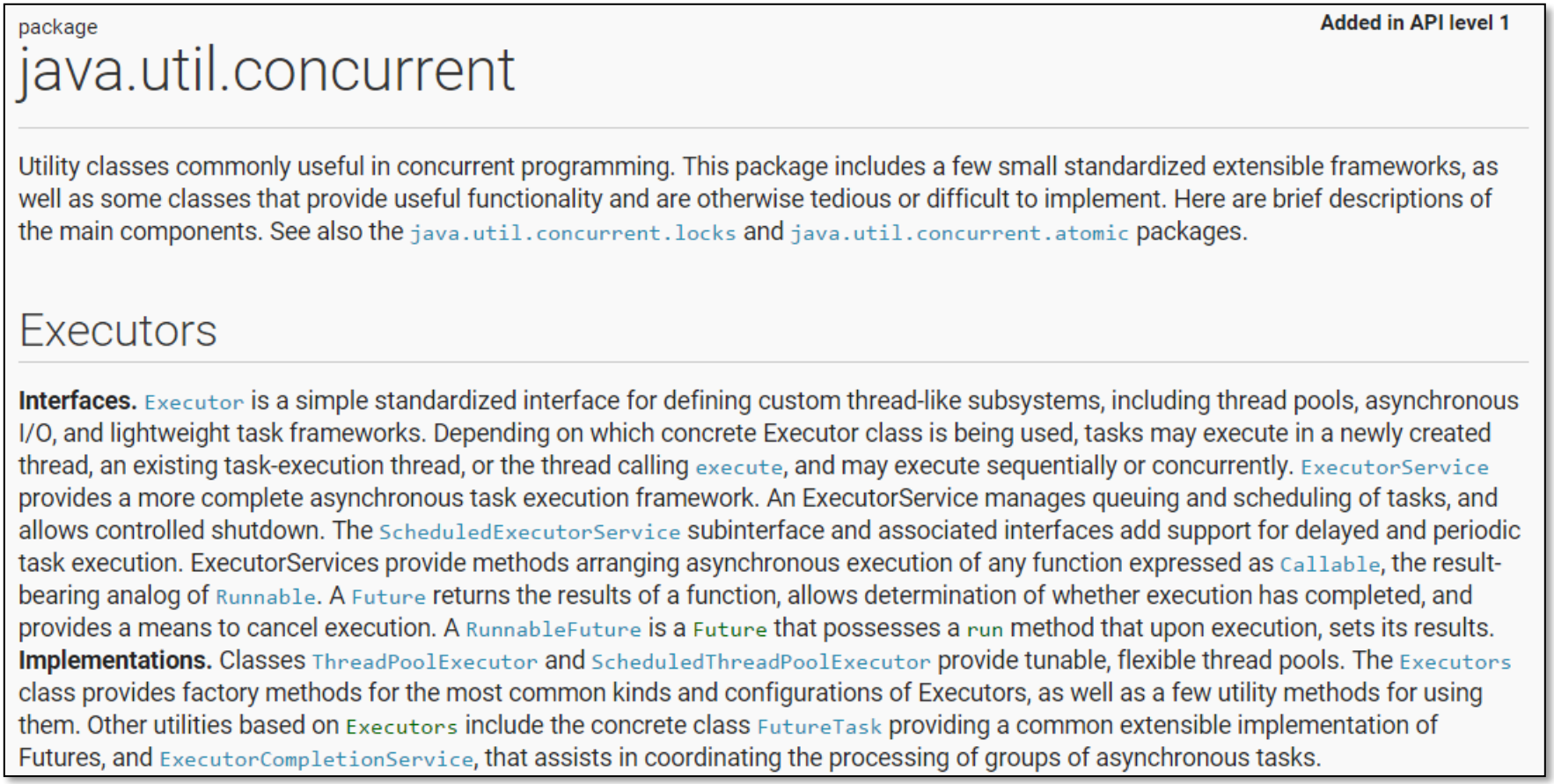
Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times



Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources



package

Added in API level 1

java.util.concurrent

Utility classes commonly useful in concurrent programming. This package includes a few small standardized extensible frameworks, as well as some classes that provide useful functionality and are otherwise tedious or difficult to implement. Here are brief descriptions of the main components. See also the `java.util.concurrent.locks` and `java.util.concurrent.atomic` packages.

Executors

Interfaces. `Executor` is a simple standardized interface for defining custom thread-like subsystems, including thread pools, asynchronous I/O, and lightweight task frameworks. Depending on which concrete Executor class is being used, tasks may execute in a newly created thread, an existing task-execution thread, or the thread calling `execute`, and may execute sequentially or concurrently. `ExecutorService` provides a more complete asynchronous task execution framework. An ExecutorService manages queuing and scheduling of tasks, and allows controlled shutdown. The `ScheduledExecutorService` subinterface and associated interfaces add support for delayed and periodic task execution. ExecutorServices provide methods arranging asynchronous execution of any function expressed as `Callable`, the result-bearing analog of `Runnable`. A `Future` returns the results of a function, allows determination of whether execution has completed, and provides a means to cancel execution. A `RunnableFuture` is a `Future` that possesses a `run` method that upon execution, sets its results.
Implementations. Classes `ThreadPoolExecutor` and `ScheduledThreadPoolExecutor` provide tunable, flexible thread pools. The `Executors` class provides factory methods for the most common kinds and configurations of Executors, as well as a few utility methods for using them. Other utilities based on `Executors` include the concrete class `FutureTask` providing a common extensible implementation of Futures, and `ExecutorCompletionService`, that assists in coordinating the processing of groups of asynchronous tasks.

See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #11

Strategies for Learning All this Material (Part 2)

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books

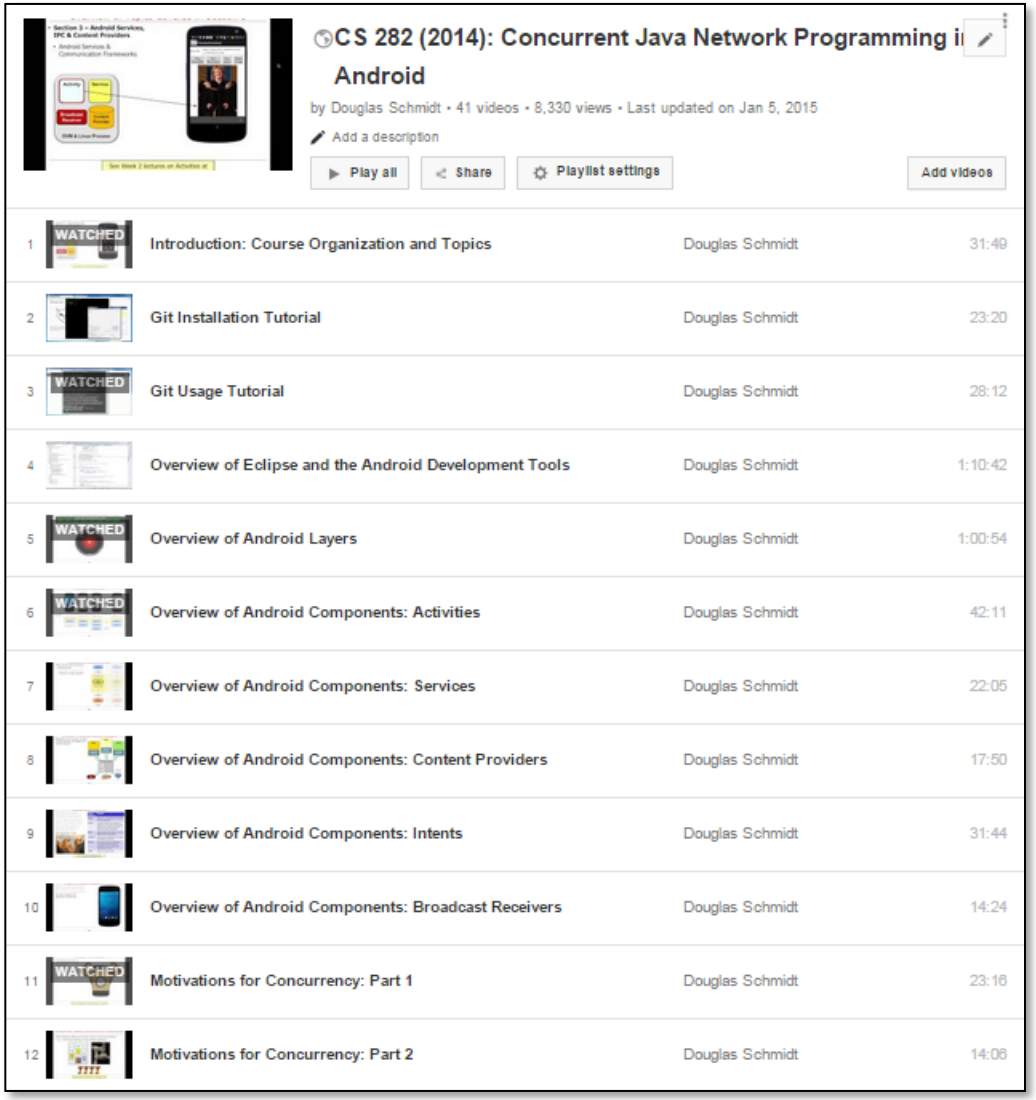
See "Suggested Reading" available at
www.coursera.org/course/posaconcurrency

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books
  - Watch online videos

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books
  - Watch online videos, e.g.
    - Material from earlier MOOCs & courses at Vanderbilt



See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ items #12 & #27

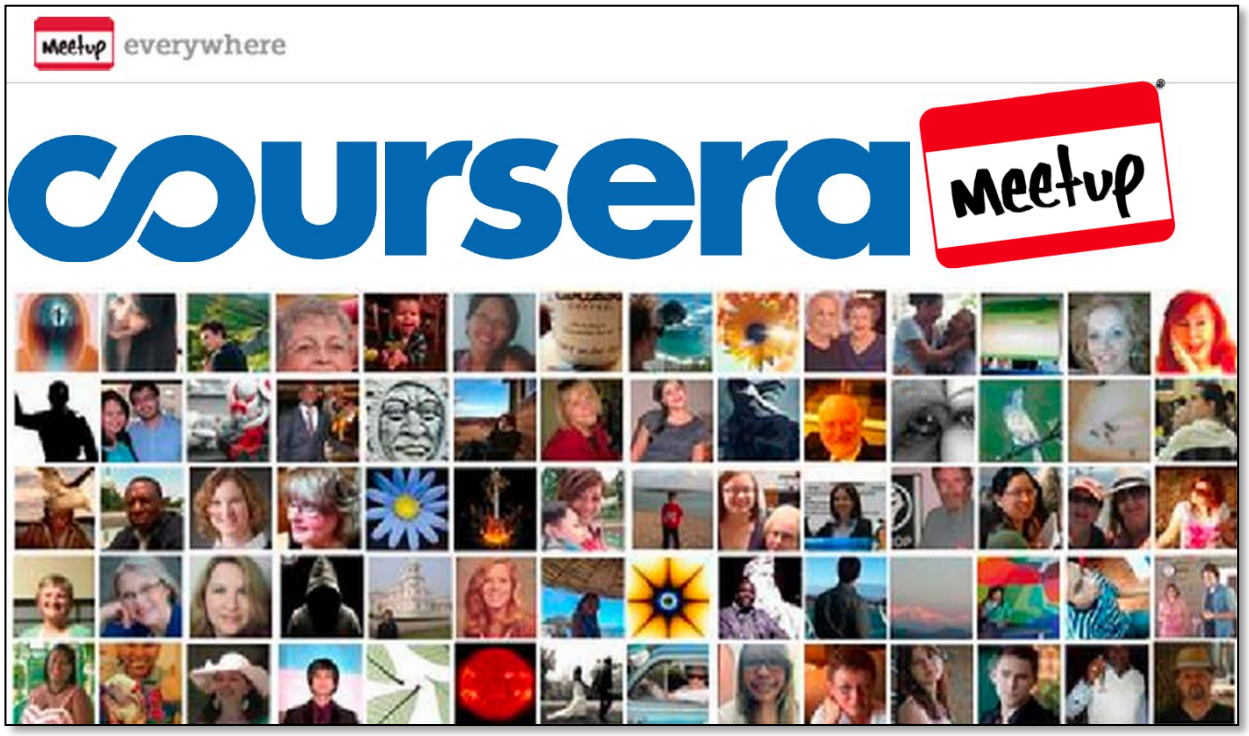
Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books
  - Watch online videos, e.g.
    - Material from earlier MOOCs & courses at Vanderbilt
    - LiveLesson videos on Java Concurrency & Design Patterns in Java

See www.dre.vanderbilt.edu/~schmidt/LiveLessons

Strategies for Learning All this Material

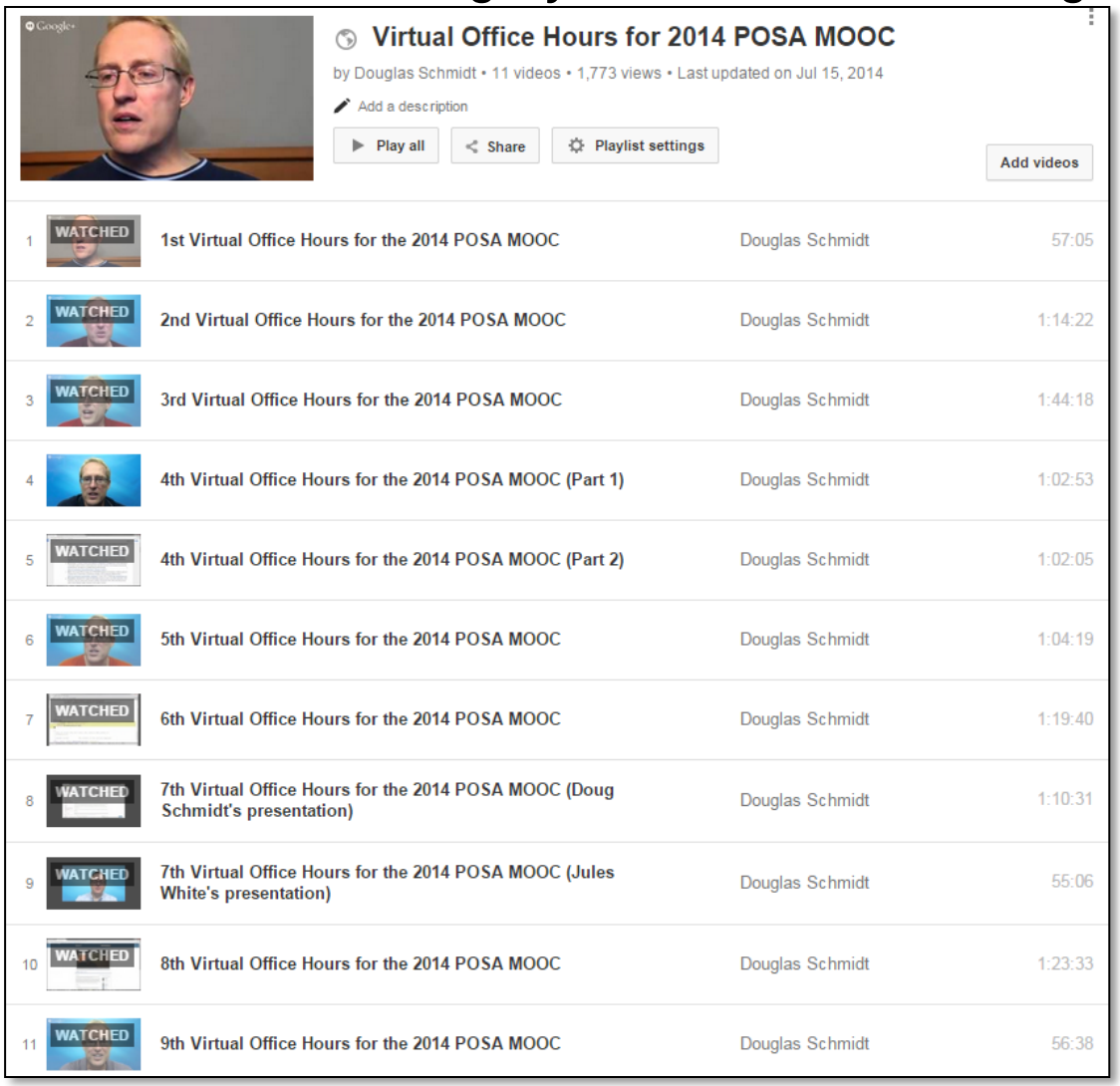
- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books
  - Watch online videos
  - Join a “meetup group”



See www.meetup.com/coursera

Strategies for Learning All this Material

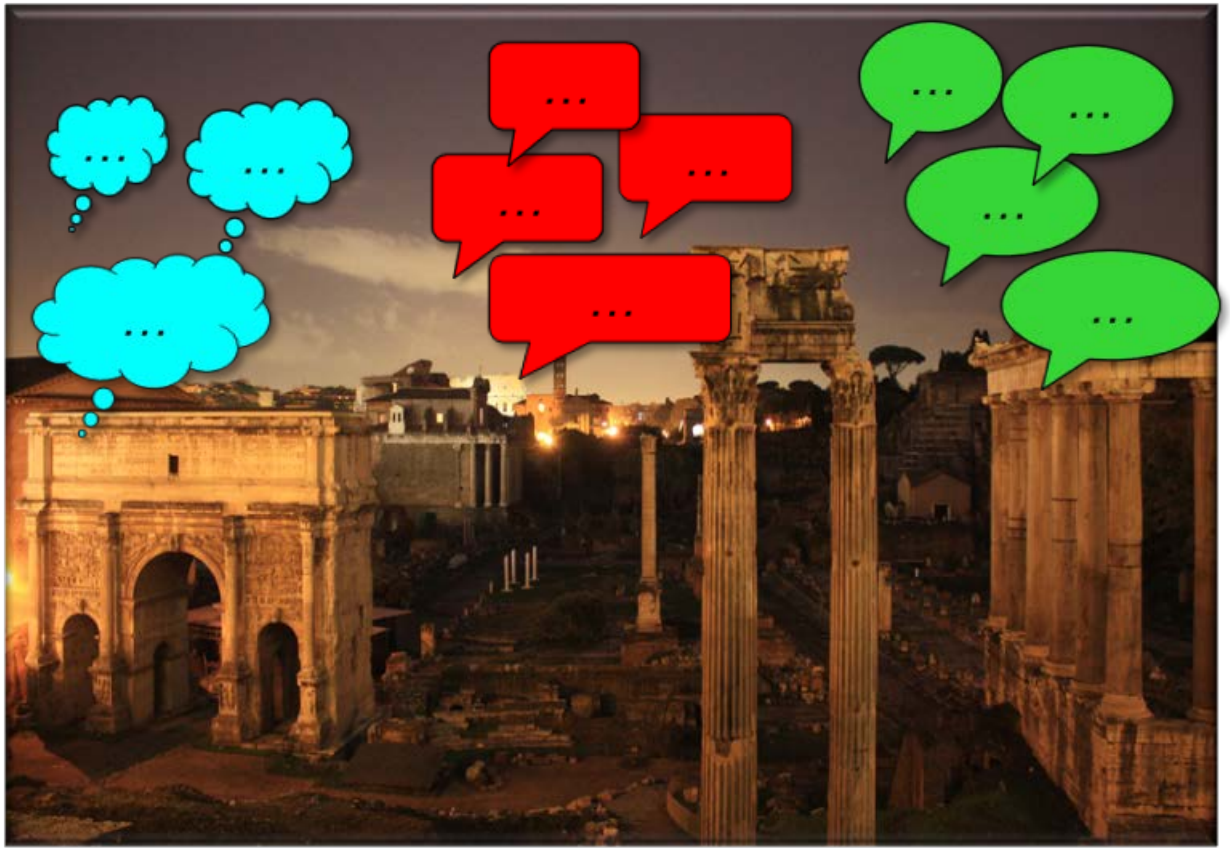
- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books
  - Watch online videos
  - Join a "meetup group"
  - Attend "Virtual Office Hours"



See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #13

Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books
  - Watch online videos
  - Join a "meetup group"
  - Attend "Virtual Office Hours"
  - Participate in online discussion forums



Strategies for Learning All this Material

- This MOOC covers a lot of material, so we encourage you to do the following
  - Watch videos multiple times
  - Explore online resources
  - Read suggested books
  - Watch online videos
  - Join a "meetup group"
  - Attend "Virtual Office Hours"
  - Participate in online discussion forums
  - Please be respectful of fellow students & staff

See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ item #9

Summary

Summary

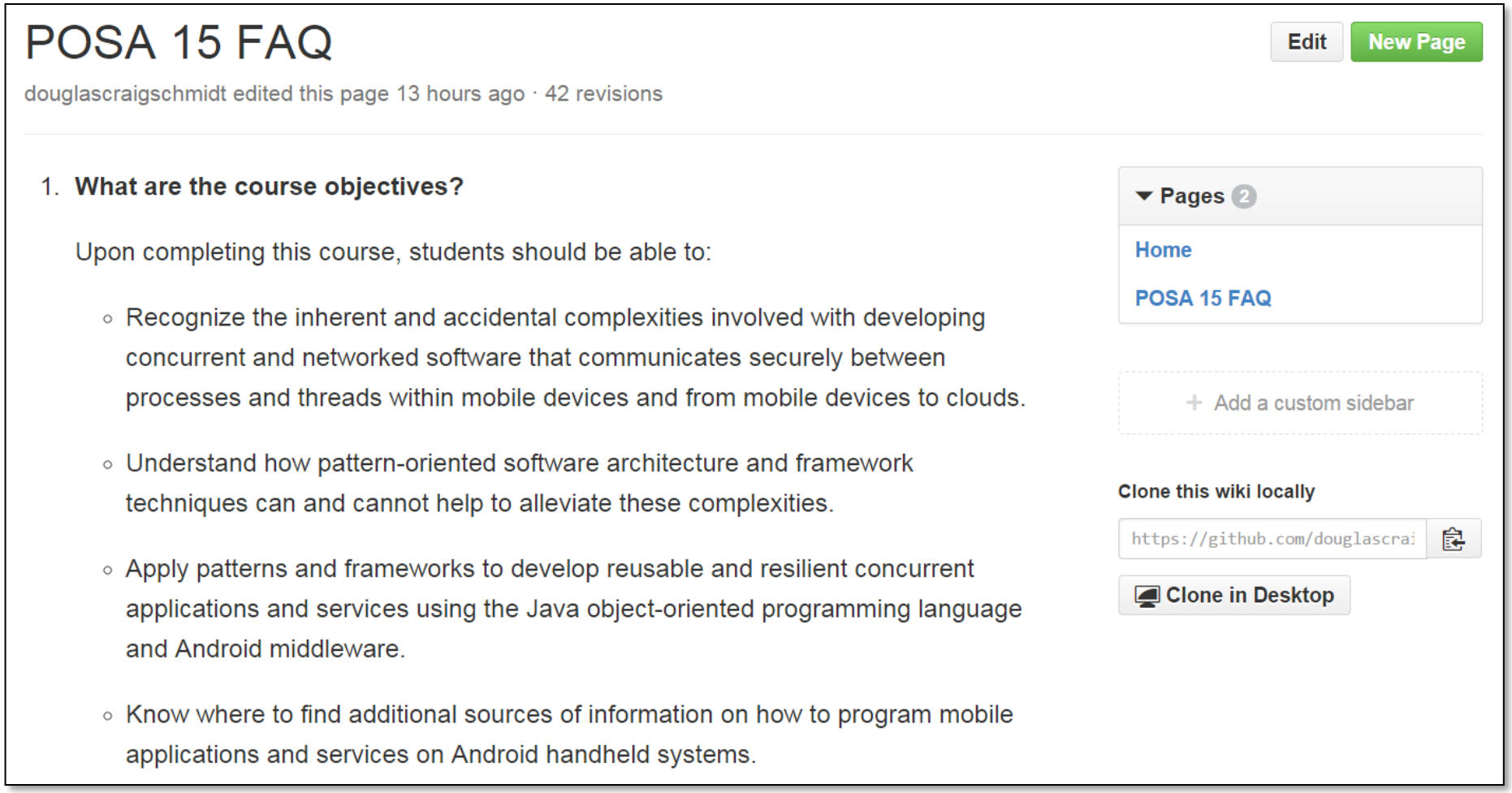
- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android

Summary

- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android

Summary

- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android



POSA 15 FAQ

Edit | New Page

douglascraigschmidt edited this page 13 hours ago · 42 revisions

1. **What are the course objectives?**

   Upon completing this course, students should be able to:

   - Recognize the inherent and accidental complexities involved with developing concurrent and networked software that communicates securely between processes and threads within mobile devices and from mobile devices to clouds.
   - Understand how pattern-oriented software architecture and framework techniques can and cannot help to alleviate these complexities.
   - Apply patterns and frameworks to develop reusable and resilient concurrent applications and services using the Java object-oriented programming language and Android middleware.
   - Know where to find additional sources of information on how to program mobile applications and services on Android handheld systems.

Pages 2

Home

POSA 15 FAQ

+ Add a custom sidebar

Clone this wiki locally

https://github.com/douglascrai

Clone in Desktop

See github.com/douglascraigschmidt/POSA-15/wiki/POSA-15-FAQ

Summary

- Our goal is to help as many students as possible gain access to a world class education on Mobile Cloud Computing with Android

Digital Learning Offerings

Douglas C. Schmidt (d.schmidt@vanderbilt.edu)
Associate Chair of Computer Science and Engineering,
Professor of Computer Science, and Senior Researcher
in the Institute for Software Integrated Systems (ISIS)
at Vanderbilt University

Coursera MOOCs on Pattern-Oriented Software Architecture (POSA)

- Mobile Cloud Computing with Android Specialization
- Spring 2015 Offering of Programming Mobile Services for Android Handheld Systems: Concurrency
- Spring 2015 Offering of Programming Mobile Services for Android Handheld Systems: Communication
- Spring 2014 Offering of Pattern-Oriented Software Architecture: Programming Mobile Services for Android Handheld Systems
- Spring 2013 Offering of Pattern-Oriented Software Architectures for Concurrent and Networked Software

Vanderbilt University Courses

- Playlist from my YouTube Channel videos from CS 251: Intermediate Software Design with Java
- Playlist from my YouTube Channel videos from CS 282: Concurrent Java Network Programming in Android
- Playlist from my YouTube Channel videos from CS 251: Intermediate Software Design with C++
- Playlist from my YouTube Channel videos from CS 282: Systems Programming for Android

Pearson LiveLessons Courses

- Design Patterns in Java
- Concurrent Programming in Java

See www.dre.vanderbilt.edu/~schmidt/DigitalLearning