CS 254: Concurrent Object-Oriented Programming with Java & Android

Course Overview & Logistics

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 Lesson

- Understand the course topics & logistics
  - Course philosophy
  - Course contents
  - Structure of the lecture material
  - Overview of the assignments & assessments
- Setting up the Java & Android IDE on Android Studio
- Accessing Android & Java source code
Course Philosophy
There's a growing need for software developers who know how to write concurrent programs for a range of computing platforms, e.g., mobile devices, laptops, desktops, & cloud environments.

The topics covered in this course apply to many platforms, not just Android.
Course Philosophy

- Demand is driven by advances in software & hardware infrastructure
  - e.g., multi-core & many core processors, mass storage, ubiquitous network connectivity, & commodity hardware & software platforms

See [www.gotw.ca/publications/concurrency-ddj.htm](http://www.gotw.ca/publications/concurrency-ddj.htm)
Course Philosophy

- Concurrency is also commonly used to offload work from the user interface (UI) thread to background thread(s) in mobile devices.

See developer.android.com/training/multiple-threads/communicate-ui.html
Course Philosophy

- Effective techniques & practices for designing & programming concurrent (mobile) apps are *not* best learned through generalities & platitudes
Course Philosophy

- Effective techniques & practices for designing & programming concurrent (mobile) apps are *not* best learned through generalities & platitudes

“Sitting & thinking” is not sufficient...
Course Philosophy

• Instead, it’s better to see *by example* how concurrent programs can be made
  • *easier* to write & read,
  • *easier* to maintain & modify,
  • *more* efficient & resilient

by applying time-proven software patterns & object-oriented (& some functional) design & programming techniques

This course involves lots of hands-on software development & testing!
Summary of the Course Contents
Summary of Course Contents

• Coverage of foundational Java & Android concurrency mechanisms

Including Java 8 programming language features & concurrency mechanisms
Summary of Course Contents

- Coverage of foundational Java & Android concurrency mechanisms
- Patterns/frameworks for concurrent programming

See [en.wikipedia.org/wiki/Concurrency_pattern](en.wikipedia.org/wiki/Concurrency_pattern)
Summary of Course Contents

- Coverage of foundational Java & Android concurrency mechanisms
- Patterns/frameworks for concurrent programming
- We assume you know (or can quickly learn) Android, Android Studio, Java 8, & Git

See item #12 at github.com/douglasraigschmidt/CS254/wiki/CS-254-FAQ
Structure of the Lecture Material
This course has three main topics

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java &amp; Android Threading</td>
<td>• Coverage of basic &amp; advanced Java &amp; Android threading mechanisms, e.g.</td>
</tr>
<tr>
<td></td>
<td>• Java Threads, Runnables, &amp; Executor framework</td>
</tr>
<tr>
<td></td>
<td>• Android concurrency frameworks</td>
</tr>
</tbody>
</table>
## Structure of the Lecture Material

- This course has three main topics

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Threading</td>
<td>• Coverage of basic &amp; advanced Java threading mechanisms, e.g.</td>
</tr>
<tr>
<td></td>
<td>• Java Threads, Runnables, &amp; Executor framework</td>
</tr>
<tr>
<td></td>
<td>• Android concurrency frameworks</td>
</tr>
<tr>
<td>Java Synchronization</td>
<td>• Coverage of basic &amp; advanced Java synchronization mechanisms, e.g.,</td>
</tr>
<tr>
<td></td>
<td>• Build-in monitor objects</td>
</tr>
<tr>
<td></td>
<td>• Myriad synchronizer classes in java.util.concurrent</td>
</tr>
</tbody>
</table>
### Structure of the Lecture Material

- This course has three main topics

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Threading</td>
<td>• Coverage of basic &amp; advanced Java threading mechanisms, e.g.</td>
</tr>
<tr>
<td></td>
<td>• Java Threads, Runnable, &amp; Executor framework</td>
</tr>
<tr>
<td></td>
<td>• Android concurrency frameworks</td>
</tr>
<tr>
<td>Java Synchronization</td>
<td>• Coverage of basic &amp; advanced Java synchronization mechanisms, e.g.,</td>
</tr>
<tr>
<td></td>
<td>• Build-in monitor objects</td>
</tr>
<tr>
<td></td>
<td>• Myriad synchronizer classes in java.util.concurrent</td>
</tr>
<tr>
<td>Software Patterns</td>
<td>• Concurrency patterns</td>
</tr>
</tbody>
</table>
## Structure of the Lecture Material

- This course has three main topics

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Threading</td>
<td>• Coverage of basic &amp; advanced Java threading mechanisms, e.g.</td>
</tr>
<tr>
<td></td>
<td>• Java Threads, Runnables, &amp; Executor framework</td>
</tr>
<tr>
<td></td>
<td>• Android concurrency frameworks</td>
</tr>
<tr>
<td>Java Synchronization</td>
<td>• Coverage of basic &amp; advanced Java synchronization mechanisms, e.g.,</td>
</tr>
<tr>
<td></td>
<td>• Build-in monitor objects</td>
</tr>
<tr>
<td></td>
<td>• Myriad synchronizer classes in java.util.concurrent</td>
</tr>
<tr>
<td>Software Patterns</td>
<td>• Concurrency patterns</td>
</tr>
</tbody>
</table>

We’ll bounce around when covering these topics to facilitate assignments
Structure of the Lecture Material

• This course has three main topics

• Each topic is composed of lessons
Structure of the Lecture Material

• This course has three main topics
  • Each topic is composed of lessons
  • Each lesson is composed of parts
Structure of the Lecture Material

- This course has three main topics
  - Each topic is composed of lessons
  - Each lesson is composed of parts
  - Each part is a single lecture

Screencasts of each lesson “part” & PDF versions of the slides will be uploaded to [www.dre.vanderbilt.edu/~schmidt/cs254#lectures](http://www.dre.vanderbilt.edu/~schmidt/cs254#lectures)
Structure of the Lecture Material

• This course has three main topics
  • Each topic is composed of lessons
  • Each lesson is composed of parts
  • Each part is a single lecture
    • Each part is composed of segments
Structure of the Lecture Material

- There will be periodic tests on material covered in the lectures
Structure of the Lecture Material

• There will be periodic tests on material covered in the lectures
• All tests (including the final) are “closed book,” “closed Internet,” “closed electronics,” etc.

1st test quiz will be on Wednesday, February 3rd via Brightspace
Structure of the Lecture Material

- There will be periodic tests on material covered in the lectures
- All tests (including the final) are “closed book,” “closed Internet,” “closed electronics,” etc.
- We’ll try to grade & review the tests by the next class

One of the benefits of a smaller class ;-)
Structure of the Lecture Material

- There will be periodic tests on material covered in the lectures
  - All tests (including the final) are “closed book,” “closed Internet,” “closed electronics,” etc.
- We’ll try to grade & review the tests by the next class

I recommend that you study for tests by reviewing slides & watching screencasts available at www.dre.vanderbilt.edu/~schmidt/cs254#lectures
Structure of the Lecture Material

- There will be periodic tests on material covered in the lectures
  - All tests (including the final) are “closed book,” “closed Internet,” “closed electronics,” etc.
- We’ll try to grade & review the tests by the next class
  - If you don’t attend the next class & don’t get your quiz you will be penalized 50%

See www.dre.vanderbilt.edu/~schmidt/cs254/work-summary.html#quizzes
There will be periodic tests on material covered in the lectures

- All tests (including the final) are “closed book,” “closed Internet,” “closed electronics,” etc.

- We’ll try to grade & review the tests by the next class
  - If you don’t attend the next class & don’t get your quiz you will be penalized 50%
  - Likewise, if you just show up for the test & don’t attend class you’ll be penalized 50%

See [www.dre.vanderbilt.edu/~schmidt/cs254/work-summary.html#quizzes](http://www.dre.vanderbilt.edu/~schmidt/cs254/work-summary.html#quizzes)
Structure of the Lecture Material

- There may be a final exam that can cover all the lectures
- The focus will be on the last week(s) of the semester

The final exam is 2 to 5pm, Friday, May 7th via Brightspace
Overview of the Assignments & Assessments
Overview of Assignments & Assessments

- Programming assignments should be written in Java 8 using Android Studio.

You can use any IDE, but your final submission must build/run with the latest Android Studio & Android 11 "R" (API level 30).
Overview of Assignments & Assessments

• Programming assignments should be written in Java 8 using Android Studio
• The Java 8 runtime environment (JRE) comes pre-installed with Android now

You can use cool Java 11 features starting with Android Studio “Arctic Fox”
Overview of Assignments & Assessments

• Android programming assignments must be submitted using Android Studio

- A wizard for creating new apps
- A visual editor for creating GUIs
- An editor for manipulating Android XML descriptors needed for your app
- An emulator for testing your apps on your PC
- A debugger for finding errors in the emulator or on a device

See developer.android.com/sdk
Overview of Assignments & Assessments

- Android programming assignments must be submitted using Android Studio
- Please install Android 11 “R” (API level 30)

See en.wikipedia.org/wiki/Android_11
Overview of Assignments & Assessments

• All source code for assignments & examples available at GitHub

Go to GitHub at github.com/douglascraigschmidt/CS254
Overview of Assignments & Assessments

• All source code for assignments & examples available at GitHub
• You will need to learn how to use GitLab et al.

See item #13 at github.com/douglascraigschmidt/CS254/wiki/CS-254-FAQ
Overview of Assignments & Assessments

• All source code for assignments & exams
  • You will need to learn how to use GitLab et al.

• Be prepared to update your repositories multiple times
  • i.e., you need to understand Git!

“If you don’t like change, you’re going to like irrelevance even less.”
Overview of Assignments & Assessments

- Assignments will provide a range of experience with Java 8 & Android concurrent programs

See [github.com/douglascraigschmidt/CS254](https://github.com/douglascraigschmidt/CS254)
Overview of Assignments & Assessments

• In particular, you’ll implement multiple variants of a Java concurrent resource manager & an associated Android app

See en.wikipedia.org/wiki/Palantir
Overview of Assignments & Assessments

• In particular, you’ll implement multiple variants of a Java concurrent resource manager & an associated Android app, e.g.
  • Java Thread, Runnable, & Executor frameworks
  • Java Semaphore, built-in monitor objects, ReentrantLock, ConcurrentHashMap, etc.
  • Android Concurrency frameworks

The topics covered by the assignments generalize to more than Android!
Overview of Assignments & Assessments

- Assignment assessments will be done via reviews by course staff
Overview of Assignments & Assessments

- Assignment assessments will be done via reviews by course staff
- Assignments *must* be submitted on time or you’ll get a 0
Overview of Assignments & Assessments

- Assignment assessments will be done via reviews by course staff
  - Assignments *must* be submitted on time or you’ll get a 0
- Your initial submission must compile & be largely complete or you won’t get a review or a final grade
Overview of Assignments & Assessments

• Assignment assessments will be done via reviews by course staff
  • Assignments *must* be submitted on time or you’ll get a 0
  • Your initial submission must compile & be largely complete or you won’t get a review or a final grade
• You will not receive a grade for assignments if you do not attend class regularly

See www.dre.vanderbilt.edu/~schmidt/cs254/assignments.html
Overview of Assignments & Assessments

- Assignment assessments will be done via reviews by course staff
  - Assignments *must* be submitted on time or you’ll get a 0
  - Your initial submission must compile & be largely complete or you won’t get a review or a final grade
  - You will not receive a grade for assignments if you do not attend class regularly
- Work *must* be your own
  - This applies for quizzes & programming assignments

www.vanderbilt.edu/student_handbook/the-honor-system#statement-of-the-honor-code
The bulk of your grade is based on the results of the automated unit tests.

See [www.dre.vanderbilt.edu/~schmidt/cs254/assignments.html](http://www.dre.vanderbilt.edu/~schmidt/cs254/assignments.html)
Overview of Assignments & Assessments

- The bulk of your grade is based on the results of the automated unit tests

It’s important that your current assignment also passes all the unit tests for previous assignments!
Overview of Assignments & Assessments

- The bulk of your grade is based on the results of the automated unit tests

See item #17 at [github.com/douglasraigschmidt/CS254/wiki/CS-254-FAQ](https://github.com/douglasraigschmidt/CS254/wiki/CS-254-FAQ)

Please don’t upgrade to a different (i.e., newer) version of Gradle!
Overview of Assignments & Assessments

• The relative weighting of each portion of the course is:
  • 45% Quizzes
  • 40% Programming projects
  • 10% Final exam
  • 05% Participation

These weightings may change, depending on various factors.
Overview of Assignments & Assessments

• The relative weighting of each portion of the course is:
  • 45% Quizzes
  • 40% Programming projects
  • 10% Final exam
  • 05% Participation
    • Participation includes attendance, involvement, & “following directions”
Overview of Assignments & Assessments

- The relative weighting of each portion of the course is:
  - 45% Quizzes
  - 40% Programming projects
  - 10% Final exam
  - 05% Participation

  Participation includes attendance, involvement, & “following directions”

Attendance also affects other aspects of your quiz & assignment grades

See www.dre.vanderbilt.edu/~schmidt/cs254/work-summary.html#quizzes & www.dre.vanderbilt.edu/~schmidt/cs254/assignments.html
Overview of Assignments & Assessments

- The relative weighting of each portion of the course is:
  - 45% Quizzes
  - 40% Programming projects
  - 10% Final exam
  - 05% Participation

- Participation includes attendance, involvement, & “following directions”

Don’t expect to get an A in this class if you do not actively participate!!!!
Setting Up the Android & Java IDE on Android Studio
Installing Java/Android Developer Tools

• To use Android, you need to install the latest release of Android Studio

See developer.android.com/studio/preview
Installing Java/Android Developer Tools

• Installation steps
Installing Java/Android Developer Tools

- Installation steps
  - Download & install the latest version of Android Studio

See developer.android.com/studio/preview
Add Components to the SDK

- Launch the Android Studio SDK Manager
- Select “R” version of Android (11, API 30)

See developer.android.com/studio/intro/update.html
Add Components to the SDK

- Launch the Android Studio Virtual Device Manager
- Create an Android API 30 emulator

[Link to Android Studio Virtual Device Manager guide]
developer.android.com/tools/devices/managing-avds.html
Intel HAXM Virtualization Driver

- **Requirements**
  - Intel virtualization extensions (VT, VT-x, vmx)
  - AMD virtualization extensions (AMD-v, SVM) [only supported on Linux]
  - Download an x86 emulator image

- **Windows & Mac OSX**
  - `<sdk>/extras/intel/Hardware_Accelerated_Execution_Manager/IntelHAXM.exe/dmg`

- **Linux**
  - Install KVM & pass “-enable-kvm” flag to emulator when starting

developer.android.com/studio/run/emulator-acceleration
Accessing Java & Android Source Code
Accessing Java & Android Source Code

- Android source code is available
  - For browsing android.googlesource.com

android Git repositories

To clone one of these repositories, install git, and run:

```
git clone https://android.googlesource.com/name
```

Name
- accessories/manifest
- device/asus/deb
- device/asus/flo
- device/asus/flo-kernel
- device/asus/grouper
- device/asus/tilapia
- device/common
- device/generic/armv7-a
- device/generic/armv7-a-neon
- device/generic/art
- device/generic/common
- device/generic/goldfish
- device/generic/mini-emulator-armv7-a-neon
- device/generic/mini-emulator-mips
- device/generic/mini-emulator-x86
Accessing Java & Android Source Code

- Android source code is available
  - For browsing android.googlesource.com
  - For downloading source.android.com

The Android Source Code

Android is an open-source software stack created for a wide array of devices with different form factors. The primary purposes of Android are to create an open software platform available for carriers, OEMs, and developers to make their innovative ideas a reality and to introduce a successful, real-world product that improves the mobile experience for users. We also wanted to make sure there was no central point of failure, where one industry player could restrict or control the innovations of any other. The result is a full, production-quality consumer product with source code open for customization and porting.

Governance Philosophy

Android was originated by a group of companies known as the Open Handset Alliance, led by Google. Today, many companies – both original members of the OHA and others – have invested heavily in Android. These companies have allocated significant engineering resources to improve Android and bring Android devices to market.

The companies that have invested in Android have done so on its merits because we believe an open platform is necessary. Android is intentionally and explicitly an open-source – as opposed to a free software – effort; a group of organizations with shared needs has pooled resources to collaborate on a single implementation of a shared product. The Android philosophy is pragmatic, first and foremost. The objective is a shared product that each contributor can tailor and customize.

Uncontrolled customization can, of course, lead to incompatible implementations. To prevent this, the Android Open Source Project also maintains the Android Compatibility Program, which spells out what it means to be "Android compatible" and what is required of device builders to achieve that status. Anyone can (and will!) use the Android source code for any purpose, and we welcome all legitimate uses. However, in order to take part in the shared ecosystem of applications we are building around Android, device builders must participate in the Android Compatibility Program.

The Android Open Source Project is led by Google, who maintains and further develops Android. Although Android consists of multiple subprojects, this is strictly a project management technique. We view and manage Android as a single, holistic software product, not a "distribution", specification, or collection of replaceable parts. Our intent is that device builders port Android to a device; they don't implement a specification or curate a distribution.
Accessing Java & Android Source Code

- Java 8 source code is available
- For browsing zgrepcode.com
Accessing Java & Android Source Code

- Java 8 source code is available
  - For browsing zgrepcode.com
  - For downloading jdk8.java.net/download.html
Summary
Summary

- You will get out of this course what you put into it.
Summary

• You will get out of this course what you put into it
• Be prepared to work hard

HARD WORK

“Human Felicity is produc'd not so much by great Pieces of good Fortune that seldom happen, as by little Advantages that occur every Day” - Benjamin Franklin
Summary

• You will get out of this course what you put into it
• Be prepared to work hard
• Do *not* miss deadlines...

See [github.com/douglascraigschmidt/CS254/wiki/Assignment-Deadlines](https://github.com/douglascraigschmidt/CS254/wiki/Assignment-Deadlines)
• You will get out of this course what you put into it
• Be prepared to work hard
• Do not miss deadlines...
• Participate in discussions in class & on piazza

See piazza.com/vanderbilt/spring2021/cs254
Summary

• You will get out of this course what you put into it
• Be prepared to work hard
• Do *not* miss deadlines...
• Participate in discussions in class & on piazza
• Avail yourself of available resources

See [www.dre.vanderbilt.edu/~schmidt/cs254](http://www.dre.vanderbilt.edu/~schmidt/cs254)
Summary

• You will get out of this course what you put into it
• Be prepared to work hard
• Do *not* miss deadlines...
• Participate in discussions in class & on piazza
• Avail yourself of available resources

Please resist the urge to email me directly unless it’s a confidential matter or you’d like to schedule a meeting!
Summary

• There are abundant opportunities!

<table>
<thead>
<tr>
<th>BROAD CATEGORY</th>
<th>2020 SALARY PROJECTION</th>
<th>2019 SALARY PROJECTION</th>
<th>% CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering</td>
<td>$69,961</td>
<td>$69,188</td>
<td>1.1%</td>
</tr>
<tr>
<td>Computer Science</td>
<td>$67,411</td>
<td>$67,539</td>
<td>-0.2%</td>
</tr>
<tr>
<td>Math &amp; Sciences</td>
<td>$62,488</td>
<td>$62,177</td>
<td>0.5%</td>
</tr>
<tr>
<td>Business</td>
<td>$57,939</td>
<td>$57,657</td>
<td>0.5%</td>
</tr>
<tr>
<td>Social Sciences</td>
<td>$57,425</td>
<td>$57,310</td>
<td>0.2%</td>
</tr>
<tr>
<td>Communications</td>
<td>$56,484</td>
<td>$52,056</td>
<td>8.5%</td>
</tr>
<tr>
<td>Humanities</td>
<td>$53,617</td>
<td>$56,651</td>
<td>-5.4%</td>
</tr>
<tr>
<td>Agriculture &amp; Natural Resources</td>
<td>$53,504</td>
<td>$55,750</td>
<td>-4.0%</td>
</tr>
</tbody>
</table>