CS 891: Concurrent Object-Oriented Programming with Java in 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
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
Course Philosophy

• Demand is driven by software/hardware infrastructure advances
  • 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)
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

“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 & 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
Summary of Course Contents

- Coverage of foundational Java & Android concurrency mechanisms
- Patterns/frameworks for concurrent programming
- Structured concurrency in Kotlin (maybe)

See kotlinexpertise.com/kotlin-coroutines-guide
Summary of Course Contents

• Coverage of foundational Java & Android concurrency mechanisms
• Patterns/frameworks for concurrent programming
• Structured concurrency in Kotlin (maybe)
• We assume you know (or can quickly learn) Android, Java 8, Kotlin, & Git

See [www.coursera.org/specializations/android-app-development](http://www.coursera.org/specializations/android-app-development)
Structure of the Lecture Material
This course has four main modules

<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 HaMeR &amp; AsyncTask frameworks</td>
</tr>
</tbody>
</table>
## Structure of the Lecture Material

- This course has four main modules

<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 HaMeR &amp; AsyncTask 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 four main modules

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
</table>
| Java Threading              | • Coverage of basic & advanced Java threading mechanisms, e.g.  
                              |   • Java Threads, Runnables, & Executor framework                    |
|                             |   • Android HaMeR & AsyncTask frameworks                               |
| Java Synchronization        | • Coverage of basic & advanced Java synchronization mechanisms, e.g.,  |
|                             |   • Build-in monitor objects                                          |
|                             |   • Myriad synchronizer classes in java.util.concurrent               |
| Kotlin Concurrency          | • Structured concurrency with Kotlin coroutines                       |
Structure of the Lecture Material

- This course has four main modules

<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 HaMeR &amp; AsyncTask 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>Kotlin Concurrency</td>
<td>• Structured concurrency with Kotlin coroutines</td>
</tr>
<tr>
<td>Software Patterns</td>
<td>• Concurrency patterns</td>
</tr>
</tbody>
</table>
# Structure of the Lecture Material

- This course has four main modules

<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 HaMeR &amp; AsyncTask 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>Kotlin Concurrency</td>
<td>• Structured concurrency with Kotlin coroutines</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 four main modules
- Each module is composed of lessons
Structure of the Lecture Material

- This course has four main modules
  - Each module is composed of lessons
  - Each lesson is composed of parts
Structure of the Lecture Material

• This course has four main modules
  • Each module 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/cs891s#lectures](http://www.dre.vanderbilt.edu/~schmidt/cs891s#lectures)
Structure of the Lecture Material

• This course has four main modules
  • Each module is composed of lessons
  • Each lesson is composed of parts
  • Each part is a single lecture
  • Each part is composed of segments

We’ll interactively cover discussion questions at the end of each part
Structure of the Lecture Material

- There will be bi-weekly quizzes on material covered in the lectures
Structure of the Lecture Material

• There will be bi-weekly quizzes on material covered in the lectures
  • 1st quiz will be on Wednesday, January 15th

All quizzes (& the final) are “closed book”
Structure of the Lecture Material

- There will be bi-weekly quizzes on material covered in the lectures
  - 1st quiz will be on Wednesday, January 15th
- We’ll try to hand back & review the quizzes by the next class

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

• There will be bi-weekly quizzes on material covered in the lectures
  • 1st quiz will be on Wednesday, January 15th
  • We’ll try to hand back & review the quizzes by the next class

I recommend that you study for quizzes by reviewing slides & watching screencasts available at www.dre.vanderbilt.edu/~schmidt/cs891s#lectures
Structure of the Lecture Material

• There will be bi-weekly quizzes on material covered in the lectures
  • 1st quiz will be on Wednesday, January 15th

• We’ll try to hand back & review the quizzes 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/cs891s/work-summary.html#quizzes](http://www.dre.vanderbilt.edu/~schmidt/cs891s/work-summary.html#quizzes)
Structure of the Lecture Material

- There will be bi-weekly quizzes on material covered in the lectures
  - 1st quiz will be on Wednesday, January 15th
- We’ll try to hand back & review the quizzes 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 quiz & don’t attend class you’ll be penalized 50%

See www.dre.vanderbilt.edu/~schmidt/cs891s/work-summary.html#quizzes
Structure of the Lecture Material

- There will be a cumulative final exam that covers all the lectures
- The focus will be on the last weeks of the semester

The final exam is 9am to noon, Friday, April 24th in this room
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 Pie 9 (API 28)
Overview of Assignments & Assessments

• Programming assignments should be written in Java 8 using Android Studio
• You’ll need to install the Java 8 runtime environment (JRE)

See github.com/douglasraicschmidt/CS891/wiki/Installing-Software
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 9.x Pie (API level 28)

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

- All source code for assignments & examples available at GitHub

Go to GitHub at [github.com/douglascraigschmidt/CS891](http://github.com/douglascraigschmidt/CS891)
Overview of Assignments & Assessments

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

“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/communication programs

See [github.com/douglasraigschmidt/CS891/tree/master/assignments](https://github.com/douglasraigschmidt/CS891/tree/master/assignments)
Overview of Assignments & Assessments

• In particular, you’ll implement multiple variants of a Java concurrent resource manager & an associated Android app, e.g.
  • Java Threads, Runnables, & Executor frameworks
  • Android HaMeR & AsyncTask frameworks
  • Java Semaphores, built-in monitor objects, & ConcurrentHashMaps
  • Kotlin co-routines (maybe)

The topics covered by the assignments may change during the semester
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

See [github.com/douglascraigschmidt/CS891/wiki/CS-891-FAQ](http://github.com/douglascraigschmidt/CS891/wiki/CS-891-FAQ)
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
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/cs891s/assignments.html](http://www.dre.vanderbilt.edu/~schmidt/cs891s/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
Overview of Assignments & Assessments

- Assessment criteria

<table>
<thead>
<tr>
<th>Assessment Category</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution correctness</td>
<td>40%</td>
</tr>
<tr>
<td>Structure (e.g., modularization, information hiding, etc.)</td>
<td>30%</td>
</tr>
<tr>
<td>Insightful programming (e.g., developing reusable class components, etc.)</td>
<td>10%</td>
</tr>
<tr>
<td>Consistent style (e.g., capitalization, indenting, etc.)</td>
<td>10%</td>
</tr>
<tr>
<td>Appropriate commenting style</td>
<td>10%</td>
</tr>
</tbody>
</table>

See [www.dre.vanderbilt.edu/~schmidt/cs891s/assignments.html](http://www.dre.vanderbilt.edu/~schmidt/cs891s/assignments.html)
Overview of Assignments & Assessments

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

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

- The relative weighting of each portion of the course is:
  - 40% Quizzes
  - 40% Programming projects
  - 10% Final exam
  - 10% Participation
    - Participation is roughly 5% attendance & 5% in-class involvement in discussions
Overview of Assignments & Assessments

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

- Participation is roughly 5% attendance & 5% in-class involvement in discussions

Attendance also affects other aspects of your quiz & assignment grades

See www.dre.vanderbilt.edu/~schmidt/cs891s/work-summary.html#quizzes & www.dre.vanderbilt.edu/~schmidt/cs891s/assignments.html
Overview of Assignments & Assessments

• The relative weighting of each portion of the course is:
  • 40% Quizzes
  • 40% Programming projects
  • 10% Final exam
  • 10% Participation
  • Participation is roughly 5% attendance & 5% in-class involvement in discussions

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 Eclipse Java/Android Developer Tools

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

Android Studio provides the fastest tools for building apps on every type of Android device.

DOWNLOAD ANDROID STUDIO

3.5 for Windows 64-bit (710 MB)

DOWNLOAD OPTIONS  RELEASE NOTES

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

• Installation steps
Installing Eclipse Java/Android Developer Tools

- Installation steps
- Download & install the Java Standard Edition JDK & JRE 8

Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:
- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

Looking for JDK 8 on ARM?
JDK 8 for ARM downloads have moved to the JDK 8 for ARM download page.

Java SE Development Kit 8u25

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

- Accept License Agreement
- Decline License Agreement

<table>
<thead>
<tr>
<th>Product / File Description</th>
<th>File Size</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux x86</td>
<td>135.24 MB</td>
<td>jdk-8u25-linux-i586.rpm</td>
</tr>
<tr>
<td>Linux x86</td>
<td>154.88 MB</td>
<td>jdk-8u25-linux-i586.tar.gz</td>
</tr>
<tr>
<td>Linux x64</td>
<td>135.6 MB</td>
<td>jdk-8u25-linux-x64.rpm</td>
</tr>
</tbody>
</table>

www.oracle.com/technetwork/java/javase/downloads
Installing Eclipse Java/Android Developer Tools

- Installation steps
  - Download & install the Java Standard Edition JDK & JRE 8
  - Download & install the latest version of Android Studio

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

- Launch the Android Studio SDK Manager
- Select “Pie” version of Android (9, API 28)

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

- Launch the Android Studio Virtual Device Manager
- Create an Android API 28 emulator

[Android Virtual Device Manager]

[Developer documentation link]
[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](https://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 [zgrepcodel.com](http://zgrepcodel.com)

**JDK 8 Project**

Building the next generation of the JDK platform

**JDK 8 snapshot builds**

- Download 8u40 early access snapshot builds
- Source code (instructions)
- Official Java SE 8 Reference Implementations
- Early Access Build Test Results (instructions)

**We Want Contributions!**

Frustrated with a bug that never got fixed? Have a great idea for improving the Java SE platform? See how to contribute for information on making contributions to the platform.

**Feedback**

Please use the [Project Feedback](http://Project Feedback) forum if you have suggestions for or encounter issues using JDK 8.

If you find bugs in a release, please submit them using the usual Java SE bug reporting channels, not with the Issue tracker accompanying this project. Be sure to include complete version information from the output of the `java -version` command.
Accessing Java & Android Source Code

- Java 8 source code is available
  - For browsing [zgrepcode.com](http://zgrepcode.com)
  - For downloading [jdk8.java.net/download.html](http://jdk8.java.net/download.html)
Summary
• You will get out of this course what you put into it
• 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...
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

See piazza.com/vanderbilt/spring2020/cs891
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
• No laptops/phones in class unless explicitly allowed

Failure to comply with this rule will cost you participation points.
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
- No laptops/phones in class unless explicitly allowed
- Avail yourself of available resources

See www.dre.vanderbilt.edu/~schmidt/cs891s
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
• No laptops/phones in class unless explicitly allowed
• 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

• 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
• No laptops/phones in class unless explicitly allowed
• Avail yourself of available resources
• There are abundant opportunities!

Summary

• If there’s an emergency, pay attention to the escape route!
• See engineering.vanderbilt.edu/about/evacuationplans.php