CS 253: Parallel Functional Programming w/ Java & Android: 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 parallel 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

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

• Effective techniques & practices for developing parallel programs & 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 these 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

- Key Java parallelism frameworks

**Parallel Streams**
- `filter(not(this::urlCached))`
- `map(this::downloadImage)`
- `flatMap(this::applyFilters)`
- `collect(toList())`

**Completable Futures**
- `/page\ = \ 8`
  - `supplyAsync (getPage())`

- `/imgNum\ = /page\ 8`
  - `thenApplyAsync (countImages(page))`
  - `.thenApply(List::size)`
  - `thenComposeAsync (crawlHyperLinks (page))`

**Reactive Streams**
- `observeOn( )`
- `subscribeOn( )`
- `observeOn( )`

Also covers Java object-oriented & functional programming language features
Summary of Course Contents

- Key Java parallelism frameworks
- Patterns for parallel programming

See www.dre.Vanderbilt.edu/~Schmidt/POSA
Summary of Course Contents

• Key Java parallelism frameworks
• Patterns for parallel programming
• We assume you know (or can quickly learn) Java, Android, & Git

See www.coursera.org/specializations/android-app-development
Structure of the Lecture Material
### Structure of the Lecture Material

- This course has three main modules

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java functional programming features</td>
<td>Coverage of Java functional programming features, such as lambda expressions, method references, &amp; functional interfaces</td>
</tr>
<tr>
<td></td>
<td>• Assume you know Java’s support for abstraction, inheritance, &amp; polymorphism</td>
</tr>
</tbody>
</table>
# Structure of the Lecture Material

- This course has three main modules

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java functional programming features</td>
<td>• Coverage of Java functional programming features, such as lambda expressions, method references, &amp; functional interfaces</td>
</tr>
<tr>
<td></td>
<td>• Assume you know Java’s support for abstraction, inheritance, &amp; polymorphism</td>
</tr>
<tr>
<td>Java Parallelism</td>
<td>• Coverage of Java 8 parallelism frameworks, e.g.</td>
</tr>
<tr>
<td></td>
<td>• Java sequential &amp; parallel streams</td>
</tr>
<tr>
<td></td>
<td>• Java completable futures</td>
</tr>
<tr>
<td></td>
<td>• Reactive streams (e.g., RxJava &amp; Project Reactor)</td>
</tr>
</tbody>
</table>
**Structure of the Lecture Material**

- This course has three main modules

<table>
<thead>
<tr>
<th>Section</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java functional programming features</td>
<td>• Coverage of Java functional programming features, such as lambda expressions, method references, &amp; functional interfaces</td>
</tr>
<tr>
<td></td>
<td>• Assume you know Java’s support for abstraction, inheritance, &amp; polymorphism</td>
</tr>
<tr>
<td>Java Parallelism</td>
<td>• Coverage of Java 8 parallelism frameworks, e.g.</td>
</tr>
<tr>
<td></td>
<td>• Java sequential &amp; parallel streams</td>
</tr>
<tr>
<td></td>
<td>• Java completable futures</td>
</tr>
<tr>
<td></td>
<td>• Reactive streams (e.g., RxJava &amp; Project Reactor)</td>
</tr>
<tr>
<td>Software Patterns</td>
<td>• Parallel programming &amp; communication patterns</td>
</tr>
</tbody>
</table>
Structure of the Lecture Material

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

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

• This course has three 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/cs253#lectures
Structure of the Lecture Material

• This course has three 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 leverage YouTube’s “progress bar” feature to demarcate segments!
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, September 2nd

All quizzes are “closed book” & are given on Brightspace
Structure of the Lecture Material

- There will be bi-weekly quizzes on material covered in the lectures
  - 1st quiz will be on Wednesday, September 2nd
- We strive to hand back & review quizzes at the start of 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, September 2nd
  - We strive to hand back & review quizzes at the start of next class

I recommend that you study for quizzes by reviewing slides & watching screencasts available at [www.dre.vanderbilt.edu/~schmidt/cs253#lectures](http://www.dre.vanderbilt.edu/~schmidt/cs253#lectures)
Structure of the Lecture Material

• There *may* 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 2 to 5pm, Tuesday, December 10th via Brightspace
Overview of 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 11/10+ (API 30).
Overview of Assignments & Assessments

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

See [github.com/douglascraigschmidt/CS253/wiki/Installing-Software](https://github.com/douglascraigschmidt/CS253/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 11/10+ (API level 30)

See [en.wikipedia.org/wiki/Android_11](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/CS253
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 & exercises
  • You will need to learn how to use GitLab et al.
  • Be prepared to update your repositories occasionally

“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 parallel programs

Go to GitHub at github.com/douglasraigschmidt/CS253
Overview of Assignments & Assessments

- Assignments will provide a range of experience with Java 8 & Android parallel programs
- Implement an image crawler app on Android using various Java 8 features, e.g.
  - Java lambda expressions, method references, & functional interfaces
  - Java sequential streams
  - Java parallel streams
  - Java completable futures
  - Java reactive streams

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/douglasraigschmidt/CS253/wiki/CS-253-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
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/cs253/assignments.html](http://www.dre.vanderbilt.edu/~schmidt/cs253/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 goes for quizzes & programming assignments
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/cs253/assignments.html](http://www.dre.vanderbilt.edu/~schmidt/cs253/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 ~5% attendance & involvement & ~5% “following directions”
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 ~5% attendance & involvement & ~5% “following directions”

Attendance also affects other aspects of your quiz & assignment grades

See www.dre.vanderbilt.edu/~schmidt/cs253/work-summary.html#quizzes & www.dre.vanderbilt.edu/~schmidt/cs253/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 ~5% attendance & involvement & ~5% “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 Eclipse Java/Android Developer Tools

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

See developer.android.com/studio

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

DOWNLOAD ANDROID STUDIO

4.0.1 for Windows 64-bit (871 MB)
Installing Eclipse Java/Android Developer Tools

- Installation steps
Installing Eclipse Java/Android Developer Tools

• Installation steps
  • 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 the “R” version of Android (11/10+, 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
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
• Android source code is available
• For browsing
  android.googlesource.com

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>accessories/manifest</td>
</tr>
<tr>
<td>device/asus/deb</td>
</tr>
<tr>
<td>device/asus/flo</td>
</tr>
<tr>
<td>device/asus/flo-kernel</td>
</tr>
<tr>
<td>device/asus/grouper</td>
</tr>
<tr>
<td>device/asus/tilapia</td>
</tr>
<tr>
<td>device/common</td>
</tr>
<tr>
<td>device/generic/armv7-a</td>
</tr>
<tr>
<td>device/generic/armv7-a-neon</td>
</tr>
<tr>
<td>device/generic/art</td>
</tr>
<tr>
<td>device/generic/common</td>
</tr>
<tr>
<td>device/generic/goldfish</td>
</tr>
<tr>
<td>device/generic/mini-emulator-armv7-a-neon</td>
</tr>
<tr>
<td>device/generic/mini-emulator-mips</td>
</tr>
<tr>
<td>device/generic/mini-emulator-x86</td>
</tr>
</tbody>
</table>
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

• 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

“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/fall2020/cs253
• 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/cs253
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!
There are abundant opportunities!

<table>
<thead>
<tr>
<th>BROAD CATEGORY</th>
<th>2019 SALARY PROJECTION</th>
<th>2018 SALARY PROJECTION</th>
<th>PERCENT CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering</td>
<td>$69,188</td>
<td>$66,521</td>
<td>4.0%</td>
</tr>
<tr>
<td>Computer Science</td>
<td>$67,539</td>
<td>$66,005</td>
<td>2.3%</td>
</tr>
<tr>
<td>Math &amp; Sciences</td>
<td>$62,177</td>
<td>$61,867</td>
<td>0.5%</td>
</tr>
<tr>
<td>Business</td>
<td>$57,657</td>
<td>$56,720</td>
<td>1.7%</td>
</tr>
<tr>
<td>Social Sciences</td>
<td>$57,310</td>
<td>$56,689</td>
<td>1.1%</td>
</tr>
<tr>
<td>Humanities</td>
<td>$56,651</td>
<td>$56,688</td>
<td>-0.1%</td>
</tr>
<tr>
<td>Agriculture &amp; Natural Resources</td>
<td>$55,750</td>
<td>$53,565</td>
<td>4.1%</td>
</tr>
<tr>
<td>Communications</td>
<td>$52,056</td>
<td>$51,448</td>
<td>1.2%</td>
</tr>
</tbody>
</table>