Android Development Fundamentals Curriculum Outline

Draft Version 4

Last updated Wed Aug 10 2016
By Jocelyn Becker

SHARED EXTERNALLY

Lecture hours: 50, Practical hours: 54

This document is shared under NDA **for review purposes ONLY**. **No guarantee is made about this content.**

Course Objective

This course teaches final-year Computer Science students how to develop Android apps.

Pre-requisites

- Java programming
- Object-oriented programming

Material provided:

- For instructors:
  - Reference slides
- For students:
  - **Concepts**: Textbook (in github) providing high level information for each topic, with links to more learning resources
  - **Practicals**: Detailed instructions for coding exercise, plus description of "do it yourself" challenge to consolidate learning
  - **Solution code**: Code for all apps that students build while working through the practicals

Course Contents

Lecture hours: 50, Practical hours: 60

Unit 1: The Basics

Setting up the Android Application Development Environment, and Creating, Testing and Debugging Applications

Lecture hours: 11
Practical hours: 12

Lesson 1. Hello World

Lecture hours: 5, Practical hours: 5

Scope:

Installing Android Studio, creating an Android app project, and deploying the app to the emulator and a device. Building a layout with UI elements including a scrolling list. Learning where and how to get help with building applications.

1.0 Concept: Intro to Android (1 hours)

- What is Android?
- Why develop apps for Android?
- Flavors of Android operating systems
- Challenges of developing for Android (multiple OS, need backwards compatibility, need to consider performance and offline capability)

1.1 Concept: Create Your First Android App (1 hours)

- Overview of the development process -- Java, Android Studio
- Project layout in Android Studio.

- Target and minimum SDKs.
- Android Virtual Device (AVD) Monitor.
- Viewing logs in logcat and AVD.
- Android manifest file
- App Architecture: An app consists of one or more activities. For an activity, write Java code and layout xml, and hook them together, and register the activity in the manifest file.

1.1 Practical: Install Android Studio, Hello World, Logging (2 hours)

1. Install Android Studio.
2. Create a virtual device.
3. Create and Run Hello World on emulator and device.
4. Explore project layout.
5. Generate and view log statements.
6. Explore manifest file.

1.2 Concept: Layouts, Views and Resources (1 hours)

- Layout elements can be viewed and edited in Layout Editor and XML.
- Introduction to the range of UI elements.
- Resources (layouts, strings, styles, themes).
- Identifying resources with IDs.
- Programmatically referencing resources using resource IDs.
- onClick attribute.
- Getting user input from a view
- Programmatically changing UI elements.
- Layout Managers
- Defining layouts for activities, inflating the layout.

1.2 Practical: Make Your First Interactive UI (1 hours)

1. Add Views and UI elements in Layout Editor to the app's home screen.
2. Edit layout XML.
3. Add click behavior to a button (show a toast).
4. Change the UI through a button click.
5. Write a method to use string resource to define a message to appear in the UI.
6. Experiment with using different layouts.
7. Explore other UI Elements in the Layout Manager.

1.3 Concept: Scrolling Views (1 hours)

- How to make activities scrollable: compare ScrollView, ListView, RecyclerView
- Getting the resource ID for a UI element by inflating a layout (needed for RecyclerView)
- How to implement RecyclerView (requires layout managers and ViewHolders)
- Performance implications of different kinds of scrolling UI elements

1.3 Practical: Working with TextView Elements (1 hours)

1. Use a scroll view for text with minor HTML formatting

1.4 Concept: Resources to Help You Learn (1 hours)

- Resources to help you learn:
- Samples that ship with the SDK.
- Templates for projects.
- developer.android.com.
- Android developer blog.
- Android developer YouTube channel.
- Source code and samples in github.
- Stackoverflow.
- Google search!

1.4 Practical: Learning Resources (1 hours)

1. Get answers from android.developer.com.
2. Create new projects with different templates.
3. Create a new project based on a sample in the SDK.
4. Find out how to add a launcher icon for your app.
5. Find out the most popular Android OS in India.

Lesson 2. Activities and Intents

Lecture hours: 3, Practical hours: 3

Scope:

Creating apps with multiple activities. Starting activities with both explicit and implicit intents. Sending data between activities. Understanding activity lifecycle.

2.1 Concept: Activities and Intents (1 hours)

- About activities
- Defining Activities
- Activity Lifecycle
- Activity navigation
- About intents
- Explicit vs Implicit intents
- Passing info to new activity
- Returning data from activity

- Activity State

2.1 Practical: Create and Start Activities (1 hours)

1. Create a new activity and layout
2. Start the new activity from an existing activity with an explicit intent
3. Pass user-entered information from one activity to the other
4. Pass information back to the main activity

2.2 Concept: The Activity Lifecycle and Managing State (1 hours)

- Activity lifecycle
- Activity lifecycle callback methods
- Activity instance state

2.2 Practical: Lifecycle and State Callbacks (1 hours)

1. Add Lifecycle callbacks
2. Save and restore instance state

2.3 Concept: Starting Activities with Implicit Intents (1 hours)

- Starting activities by sending implicit intents
- Intent filters and enabling your activities to recieve intents
- ShareCompat

2.3 Practical: Start Activities with Implicit Intents (1 hours)

1. Send an implicit intent to start an activity (open web site)
2. Send an implicit intent to start an activity (open location)
3. Use an intent filter to allow other apps to start an activity in your app
4. Use ShareCompat.IntentBuilder

Lesson 3. Testing and Debugging, and Backwards Compatibility

Lecture hours: 3, Practical hours: 5

Scope:

Using the debugger, testing your code, and learning about support libraries for backwards compatibility with previous versions of Android.

3.1 Concept: Debugging your apps (1 hours)

3.1 Practical: Using the Debugger (2 hours)

3.2 Concept: Testing your app (1 hours)

3.2 Practical: Testing your code (2 hours)

3.3 Concept: Support libraries (1 hours)

3.3 Practical: Use support library (1 hours)

Unit 2: User Interface

Create responsive, adaptive user interfaces that work across different devices

Lecture hours: 6
Practical hours: 8

Lesson 4. User Interaction and intuitive navigation

Lecture hours: 3, Practical hours: 4

Scope:

Receiving and responding to user input. Understanding and implementing different ways for users to navigate your application.

4.1 Concept: User Input Controls (1 hours)

- Getting user input
- Changing keyboards
- Buttons
- Dialogs and pickers
- Spinners, checkboxes, and radio buttons
- Gestures
- Speech recognition (not done)
- Sensors (not done)

4.1 Practical: Use Keyboards, Input Controls, Alerts, and Pickers (1 hours)

1. Experiment in your app with different keyboards for user input, spelling suggestions, and auto-capitalization.
2. Add a spinner input control for selecting one value out of a set of values.

Lecture hours: , Practical hours:

1. Create new app to show an alert, and record the user's selection (OK or Cancel). MOVE TO CONCEPT.
2. Update app to show date and time pickers and record the user's selections.

4.2 Concept: Menus (1 hours)

- Options menu, contextual menus (floating and action bar), and popup menu
- Adding menu items.
- Handling onClicks from menus.

4.2 Practical: Use an Options Menu and Radio Buttons (2 hours)

1. Set up an options menu and overflow menu
2. Add items to the option (overflow) menu.
3. Add radio buttons for user selection.
4. Add Up navigation to the app bar.

4.3 Concept: Screen Navigation (1 hours)

- Terminology
- Different ways a user can navigate through an app.
- Action bar
- Settings menu
- Navigation drawer
- Directed workflow (funnels).
- Best practices for navigation

4.3 Practical: Create a Recycler View (1 hours)

1. Create an activity that displays data in a RecyclerView.
2. Make the items in the list clickable
3. Add a floating action button to add items to the list

4.3 Practical Challenge: Tab Navigation (hours)

1. Create new app with tab navigation to 3 views.

Lesson 6. Delightful User Experience

Lecture hours: 5, Practical hours: 5

Scope:

Using themes and styles. Creating responsive user interfaces that use material design principles. Creating layouts that work on different screen sizes and orientations. Creating accessible and localizable apps.

5.1 Concept: Themes and Styles (1 hours)

- Best practices for themes and styles
- Performance benefits for themes.
- When and how to use drawables, best practices for drawable
- When and how to use nine-patches, best practices for nine-patches
- Tools for creating drawables

5.1 Practical: Theme, Custom Styles, Drawables (1 hours)

1. Define and use a theme
2. Define and use a custom style that uses a drawable

5.2 Concept: Material Design (1 hours)

- What is material design? Material design best practices. Material Design guidelines.
- Implementing Material Design look and feel, with compatibility with previous versions
- Support library for Material Design design
- Transitions and Animations

5.2 Practical: Add a FAB and Cards (1 hours)

1. Create an app that uses a Floating Action Button (FAB)
2. Add an activity that uses cards. Optionally, style the cards.
3. Customize your app's theme and styles to use Material Design styles and colors.

5.3 Practical Challenge: Transitions and Animations (optional) (hours)

1. Add a Material Design Transition and/or Animation in your app.

5.4 Concept: Adapt layouts for multiple devices and orientations (1 hours)

- Why we need to consider different screen sizes and orientations
- Screen density (dip or dp).

- How to create adaptive layouts using resources folders
- Different ways to create images that scale nicely.
- Images and image formats and how they affect performance (download speeds).

5.4 Exercise: Landscape, Multiple Devices (1 hours)

1. Update layout to look good on landscape orientation, and on tablet.
2. Use emulator for different device sizes.
3. Add images that look good on different devices.

5.5 Concept: Accessibility (1 hours)

- Why accessibility matters
- Accessibility considerations: Color blindness, poor vision, poor hearing, physical limitations
- Accessibility guidelines
- Testing for accessibility
- Screenreaders
- Making your app more accessible: Color and Contrast, button size --> Material Design guidelines, considerate layouts and navigation

5.5 Practical: Put yourself in the Users shoes (1 hours)

1. Test your app for accessibility, using Talkback and Explore by Touch. Switch to monochrome color space
2. Put in earplugs, can you still use your app?
3. Wear the darkest glasses you can find, can you still use your gloves?
4. Put on gloves, can you still use your app?
5. How would you make one of the apps you have written so far more accessible?

5.6 Concept: Localization (1 hours)

- How to prep your app for localization.
- LTR and RTL (eg Arabic) text.

5.6 Practical: Implement Localized Strings (1 hours)

1. Create localized strings in your app
2. Test by changing default language

5.7 Practical Challenge: (optional) Find three good apps, that do accessibility right, localization right, workflow and navigation right (hours)

Lesson 6. Testing your UI

Lecture hours: 1, Practical hours: 3

Scope:

Testing your UI

6.1 Concept: Testing the User Interface: (1 hours)

- Automated testing of UIs
- User testing your UI with real users
- Using the Espresso and UI Automator frameworks for testing UIs

8.1 Practical: Use Espresso to test your UI (1 hours)

1. Use Espresso to Test Your UI

Unit 3: Background Tasks

Perform background work and long-running tasks in Android applications

Lecture hours: 5
Practical hours: 5

Lesson 7. Connect to the Internet

Lecture hours: 3, Practical hours: 4

Scope:

Establishing an internet connection, sending an HTTP request and parsing a JSON response. Running work asynchronously in the background.

7.1 Concept: Background Tasks (1 hours)

- Synchronous versus async tasks.
- What is the UI thread and when should you use it?

- Example of a background task -- retrieving data over the internet.
- Creating background tasks. (schedule, send data, etc.)
- Implementing AsyncTask (doInBackground(), callbacks)
- Limitations of AsyncTask
- Passing info to background tasks.
- Initiating background tasks.
- Scheduling background tasks (intro only, more later)

7.1 Practical: Create an AsyncTask (1 hours)

1. Create a simple AsyncTask to do work in the background

7.2 Concept: Connecting to the Internet (1 hours)

- Permissions.
- Building URIs
- Opening and closing Internet connections.
- Parsing JSON in Android. (Because it's common.)
- Sending requests and parsing response.

7.2 Practical: Google APIs Explorer, JSON, Books API (2 hours)

1. Use the Books API in the Google APIs Explorer to investigate request format and JSON response format
2. Create a new app that uses the Books API and AsyncTask to search for the author of a book..
3. Write the code to parse the response and extract and display the relevant information
4. Debug errors when the Internet permission is missing
5. Add the missing permission to the Android Manifest.
6. Verify your fix by running and testing your app.

7.3 Concept:AsyncTaskLoader (1 hours)

- Intro to AsyncTaskLoader
- loadInBackground()
- AsyncTaskLoader callbacks
- Benefits of loaders

7.3 Practical: Use AsyncTaskLoader (1 hours)

1. Use AsyncTaskLoader instead of AsyncTask to show book search results in a RecyclerView

Lesson 8. Notifications and Background Tasks

Lecture hours: 3, Practical hours: 2

Scope:

Implementing long-running tasks that persist beyond an activity's duration. Understanding app priorities.

8.1 Concept: Broadcast Receivers (1 hours)

- What is a Broadcast Receiver and a Broadcast Intent?
- Broadcast Receiver Security and Lifecycle

8.1 Practical: BroadcastReceiver (1 hours)

1. Create an app with a BroadcastReceiver

8.2 Concept: Services (1 hours)

- What is a service? Long running task without a UI.
- Difference between Activity and Service
- Start and stop services.
- Lifecycle methods.
- Foreground services.
- IntentService class.
- App priority (critical, high, low).
- How to create a new Service.

8.3 Concept: Notifications (1 hours)

- What is a Notification?
- Notification Design Guidelines

8.3 Practical: Notifications (1 hours)

1. Trigger a Notification
2. Add Actions to your Notification

Lesson 9. Triggering, Scheduling, and Optimizing Background Tasks

Lecture hours: 2, Practical hours: 3

Scope:

Scheduling and triggering background tasks. Using alarms, Job Scheduler, Broadcast Receivers.

Understanding the impact of data transfer on battery power.

9.1 Concept: Alarm Manager (1 hours)

- Alarm Managers

9.1 Practical: Alarm Manager (1 hours)

1. Implement an alarm manager

9.2 Concept: Transferring Data Efficiently (1 hours)

- Less data, less often!
- Cell radio lifecycle
- JobScheduler. Why to use JobScheduler instead of SyncManager/SyncAdapter.
- Difference between alarms and job schedulers.

9.2 Practical: Job Scheduler (1 hours)

1. Use JobScheduler to do background updates

9.3 Practical: Firebase Job Dispatcher (1 hours)

Unit 4: Data -- Saving, Retrieving, Loading

Storing, sharing and retrieving data in Android applications

Lecture hours: 1
Practical hours: 0

Lesson 10. Storing Data in your app

Lecture hours: 2, Practical hours: 1

Scope:

Understand the different ways to store and retrieve data both in the app and externally. Use Preferences to save key value pairs.

10.1 Concepts: Overview to storing data (1 hours)

- Internal versus external storage
- Privacy, sharing, security, encryption of your data

- Shared Preferences: Store private primitive data in key-value pairs
- SQLite Databases: Store structured data in a private database
- Store data on the web with your own network server
- Firebase for storing and sharing data in the cloud,
- Concept: Preferences
- What are Settings and Preferences?
- Settings best practices (harder to take away settings than to add, for usability reasons
- Storing and retrieving preferences as key/value pairs using SharedPreference
- Different Settings types.
- Settings menu.
- Using Activity and PreferenceFragments to allow users to set preferences

10.1 Practical: Get and Save User Preferences (1 hours)

1. Implement Settings menu to allow users to enter preferences.
2. Implement code to retrieve and user user preferences

Lesson 11. Storing Data using SQLite

Lecture hours: 1, Practical hours: 4

Scope:

Saving data locally to the app in a SQLite database. Allow users to add, edit and delete items.

11.1 Concept: Store data using SQLite database (1 hours)

- Overview of SQLite.
- OpenHelper Android class
- Querying (dev) Searching (user) databases
- Best practices for using databases in Android
- Best practices for testing your database

11.1 Practical: Save user data in a database (2 hours)

1. Create an app that allows users to enter notes
2. Save the notes in a SQLite database
3. Create an app that stores data in an SQL database.

   Display the data in a RecyclerView.

   Allow users to add, delete, and edit data items.

11.2 Practical: Querying and Searching a Database (2 hours)

Lesson 12. Sharing Data: Content Resolvers and Content Providers

Lecture hours: 3, Practical hours: 4

Scope:

Using Content Resolvers and Content Providers to provide an interface to the app's data.

12.1 Concept: Using Content Resolvers to access data (1 hours)

- Content Providers and Content Resolvers work together
- What is a content provider?
- What is a content resolver?
- How do they work together?
- How to implement and use Content Resolvers

12.2 Concept: Content Providers (2 hours)

- When to implement content providers
- How to implement content providers (overview)
- Content URIs
- UriMatcher
- Content Provider authorities
- Required methods on ContentProvider (query, insert, delete, update)
- MIME types
- Contracts
- Making content provider data accessible to other apps by modifying manifest, and protecting data with permissions.

12.2 Practical: Implement a Content Provider (2 hours)

1. Add a content provider for your SQLite database

12.3 Practical: Use a ContentResolver to query your data (2 hours)

1. Use a content resolver to query the database
2. Display the results of the query
3. Use the content resolver to add data to the database

Lesson 13. Loading Data using Loaders

Lecture hours: 2, Practical hours: 2

Scope:

Loading data efficiently using Loaders.

13.1 Concept: Using Loaders to Load and Display Data (2 hours)

- Using loaders to asynchronously load data into an activity or fragment
- Benefits of Loaders -- why use them?
- Loader states (started, stopped, reset)
- LoaderManager
- Methods & callbacks to implement in Loaders: loadInBackground(), deliverResult() onStart/StopLoading(), onReset/Cancelled())
- Registering listeners
- Using CursorLoader with ContentProviders

13.1 Practical: Implement a Loader (2 hours)

1. Implement a loader
2. Register a Listener for the Loader
3. Test the loader by checking that the Items in the UI update when the data generated by the loader changes
4. Use an AsyncTaskLoader to update a scrolling list of notes titles as the user adds more notes
5. Register a Listener for the Loader
6. Test the loader by checking that the Items in the UI update when the underlying data changes

---

Unit 5: Polish and Publish

Publishing Android Applications

Lecture hours: 8
Practical hours: 1

Lesson 14. Permissions and Libraries

Lecture hours: 2, Practical hours: 0

Scope:

Understanding the permissions model in Android. Know how to find and use libraries to make the development process more efficient.

14.1 Concept: Permissions (1 hours)

- The permsissions model

14.2 Concept: Libraries (1 hours)

- Using libraries

Lesson 15. Security best practices

Lecture hours: 1, Practical hours:

Lesson 16. Widgets

Lecture hours: 1, Practical hours: 0

Scope:

Using notifications to send data from your app to the user.

16.2 Concept: Widgets (1 hours)

- What are widgets? When to use them and how to implement them.

16.2 Practical Challenge: Widgets (optional) (hours)

1. Optional -- create a widget for one of your apps.

Lesson 17. Publishing your App

Lecture hours: 2, Practical hours: 1

Scope:

Understanding ways to monetize your app. Packaging and publishing your app.

17.1 Concept: Monetizing your app (1 hours)

- Different ways to monetize your app (overview only)

17.2 Concept: Making and publishing APKs (1 hours)

- Guidelines for publishing in Google Play
- Make and sign the APK.
- Beta test your app
- Publish your app to Google Play

17.2 Practical: Beta testing your app (1 hours)

1. Running a beta test on Google Play

Lesson 18. What's Next?

Lecture hours: 5, Practical hours: 0

Scope:

Understanding the range of form factors that Android runs on. Understanding the range of Google services that your app can use, from Maps, to Location, to Fit and more.

18.1 Concept: Multiple Form Factors (1 hours)

- Wearables
- Auto
- TV

18.2 Concept: Google Services (hours)

- Using Google services

18.3 Concept: Firebase (1 hours)

- Firebase -- (Google Platform)

18.4 Concept: Google Cloud Messaging (1 hours)

- Google Cloud Messaging -- What it is, Why and when to use it.

18.5 Concept: Making your app data searchable (1 hours)

- Making your app data searchable

18.6 Practical Challenge: Wrapup (optional) (hours)

1. Wrap-up Challenge: Go forth and write great apps

Lesson Appendix

Lecture hours: , Practical hours:

0.1 Compare Custom Objects (hours)

0.2 Copy and Rename a Project (hours)

0.3 Extract Resources (hours)

0.4 Save Custom Objects (hours)

That's all folks! THE END