We used XML, Java and Android to create applications for Android based devices emulated as Virtual Machines on Eclipse, with either an ARM or Intel microprocessor emulated device. We used the Android Development Kit and Eclipse programming tools that Android software developers use and build complete and useful apps during this course. Fundamental computer science principles and programming ideas that power today’s smartphone and tablet apps were covered too.

Topics Covered: Exploring device files, Working with resources, Reducing large images, Working with ImageView, Supporting Different Screen Sizes, Supporting Different Densities, Working with strings, Scaling images, Using RelativeLayout, Qualified layouts, setAdjustViewBounds (boolean), Relative Layout, ImageView.ScaleType, Supporting Different Screen Sizes, Language-specific strings, Changing the emulator's locale, alternative localizations for your app, Previewing your app under different languages using the layout editor of your phone or emulator, Google translate as starting point for text translation, Localizing with Resources and Introduction to resources and qualifiers, Scrolling, XML start tags and end tags, Working with layout constraints, Creating a keystore, Signing and exporting an app, Deleting apps, Installing an app from an unknown source, Downloading and installing the APK, Removing unnecessary resources, Spacing, Non-translatable resources, Image memory constraints on Android 2.x devices, onCreate when app is rotated, How to export your entire project, Memory Management for Android Apps. Modern Android (3.0+) now uses Java heap memory for bitmaps, Adding mp3 files as an Android raw resource, Creating a MediaPlayer object. Buttons, Calling methods when buttons are clicked, Comments, Web browsing with Intent, MediaPlayer state diagram, Activity lifecycle, Responding to onResume and onPause events, Declaring variables inside and outside of methods, Test your app on the emulator to find out when you see your 3 log messages. Telnet to localhost (port 5554) and test the following conditions: When the emulator receives a phone call. (Hint: gsm call 1234), When the emulator receives an SMS message. (Hint: sms send 1234 Hello), When you rotate the screen from portrait to landscape (and vice versa), When you press the home button or switch to a different app, Debugging, Breakpoints, Log messages, Error messages, While paused notice that you can see the stack trace (methods calling methods) and the value of your variables, State Diagrams as a Computer Science concept, User Interface, supporting different screens, Android Manifest tricks, create a background music player that is independent of an Activity lifecycle you will need to use a media player inside an Android Service, Android Asset Studio: Launcher Icon Generator, Density Support Supporting Multiple Screens, Android Manifest The AndroidManifest.xml File, The Android Manifest's activity tag, Defining Styles: Style Resource, Styles and Themes, Android style definitions, Configuration qualifiers and resource matching: How Android Finds the Best-matching Resource, Using configuration qualifiers, Tim Berners-Lee's World Wide Web, A free Java Book, Android assets, WebView, A better user experience ("UX"), Take me back with onKeyDown, Android Manifest Activity, Adding Internet Permission, Enabling Javascript and zoom controls, Building Web Apps in WebView, Supporting Different Screens in Web Apps, Debugging Web Apps, Migrating to WebView in Android 4.4, Best Practices for Web Apps, Creating a second launcher activity, Starting activities with Intent, Adding an element of surprise, Introducing vertical LinearLayout, Designing a simple survey/feedback form, Introducing text fields inputTypes, How Android uses inputTypes, Using EditText hints, LinearLayout widget reference, Linear Layout Android guide, Text fields design notes, Text controls Android guide, Invisible views, findViewById, Casting types, Debugging tips, Toast, View, Input Events, String vs. integer variables, Assignment vs. comparing variables, String methods: indexOf, equals, substring, contains, Requesting focus, Returning early, Comparing Strings using .equals, Converting integers to and from Strings, Expecting the unexpected in your app, Using try-catch to catch exceptions, boolean variables, An easy 2-line animation, Integer.parseInt() reference, AnimationUtils reference, An implicit Intent to share, Sharing text as SMS (text) and email messages, Tips for using startActivity, ScrollView and LinearLayouts, Gravity and weight, Visibility, TextWatcher interface, Null pointers, Creating your own ScrollView and LinearLayout, and experimenting with fillViewPort, layout\_gravity, and layout\_weight, using getString(...) to pull a string entry from the strings.xml file, refactor/rename and also extracting string constant menu items, LinearLayout, ScrollView, HorizontalScrollView, TextWatcher, Linear Layout, Working with SharedPreference objects, Creating Views in Java, Working without a layout xml, Adding an OnClickListener, Coloring text, Shared Preferences, Responding to UI vents, Hexadecimal color values, TextView reference, Introducing the Android Event Queue, Don't be a hog; work fast; don't sleep, Runnable interfaces, Using postDelayed to delay UI code until later, anonymous interfaces and listeners, SystemClock reference, Android’s Processes and Threads guide, Specifying the Code to Run on a Thread, Creating bitmaps, Bitmap configurations, Wiring up a Canvas, Bitmap, and ImageView, ARGB colors using hexadecimal integers, Who flipped the y-axis? Drawing lines and bitmaps, Bitmap reference, Canvas reference, ImageView reference, Intent to pick a picture, Adding images to the emulator's gallery, startActivityForResult, onActivityResult, Working with URIs, ContentResolver, and streams, Interacting with other apps, Intent reference, InputStream reference, Using BitmapFactory.Options to read just the image size, Finding the default screen size in pixels, Sampling bitmaps to reduce their size, Using a 'while' loop, Displaying Bitmaps Efficiently, Mutable bitmaps, Drawing bitmap on another bitmap, Transparent paint and ARGB color, Drawing text, Saving bitmaps, External storage, Files and directories, Date and time, Sharing bitmaps, File reference, Environment reference, File storage guide, Date and Time format reference and example code, The finally Block, all the keywords of Java, Programmatic views

Inner classes, Pixel perfect experiments, Working with floats, Exploring Android source with grepcode, Invalidating views, Bitmap, Canvas, Paint, Canvas transformations, Single-touch actions, SystemClock, Canvas, Math public methods, Primitive Data Types, Why f is placed after float values, Semi-transparent bitmap trails, Reading xml dimensions, Rescaling bitmaps, MotionEvent, View.OnTouchListener, Tracking Movement, Using Touch Gestures, Refactoring towards readable code, Refactoring to nested and a separate View class, Custom views in xml, inner class, nested class, A completely separate Java class in its own file, Java Methods, Constructor fundamentals, Guide to Custom Views, Preference settings, Developing for new and old, Preferences for pre-Honeycomb (<API 11), API 11+: Using fragments and fragment transactions, PreferenceManager, Responding to the action bar/menu bar events, Listening for changes in preferences, Create your own PreferenceScreen xml file, Adding a checkbox and edit text, You can also group your items using a PreferenceCategory, Creating a new class (and include it in your ApplicationManifest) that extends PreferenceActivity, support Fragments, Read your preference values from Java, Listen for changes in preferences, In main activity you can respond to settings item in the ActionBar/Menu to start your Settings Activity, An in-depth guide to using Preferences, Fragments in Android, String array resources and string arrays, Random integers, Create your own facts file, Use a BroadcastReceiver and an Intent to start the app whenever the phone is powered on, Create a home screen widget to display a Java fact, Downloading Internet text data using AsyncTaskLoader, Parsing JSON data into Java objects,

Displaying lists, Working with NFC to beam data from one device to another, Handling NFC events