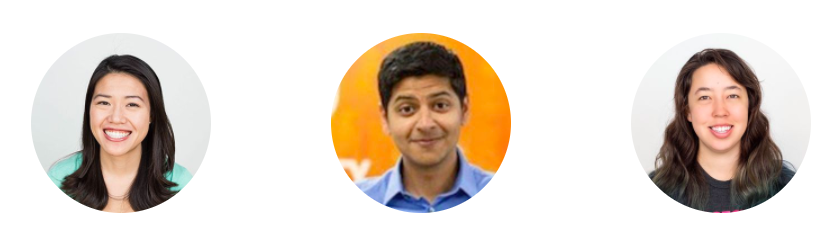
**Android**

**Developers**



Katherine Kuan, Kunal Chawla, and Lyla Fujiwara

**Two Strategies**

1.[**Troubleshooting Document(opens in a new tab)**](https://docs.google.com/document/d/1w1Xn_hnSAODAAtdRDp7haYPBtEwX_l7Htpf8Wpgbu6w/pub?embedded=true) is a growing document that lists solutions to common problems that students run into.

2.[**Google(opens in a new tab)**](https://www.google.com/) is a great tool for troubleshooting. Copy the exact error message you're getting and search for it. Try adding Java (if you're installing the JDK) or Android (if you're installing Android Studio).

nstalling The Java Development Kit: Windows Guide

There are several steps you need to take before installing Android Studio. Some of these steps might seem very technical, and it's OK if you don't fully understand the terminology.

Make sure to follow the instructions carefully.

1. Check If You Have Java on Your Computer

**Open the Terminal**

Before you start, you will need to open a program on your computer called the Terminal. To do this, go to the Task Bar at the bottom of your computer screen, click **Start**, and navigate to **Run...**

Type **cmd** in the box that pops up. (Your box might look slightly different than ours, depending on what version of Windows your computer is running.)

This will open the Terminal window.

**Use the Terminal to find information**

Now you are ready to check if you have the Java Developer Kit, version 7 or greater, already installed on your computer. To check if you have JDK installed (and which version), open a terminal window and type:

java -version

Then hit enter.

The example below shows Java version **8.0\_05** -- the version number comes after the “1.”

If you have Java 7 or greater, you can move on to the next node: Install Android Studio.

If the JDK is **not** available, or the version is lower than 7, go on to Step 2, below.

2. Download the Java Development Kit

Download the Java Development Kit, aka, the JDK, from [**this page(opens in a new tab)**](http://www.oracle.com/technetwork/java/javase/downloads/index.html).

Oracle, the company that maintains Java, has a lot of options and acronyms. We're looking for the plain old JDK. This is the kit you need to start developing your apps with Java.

After you click the download link, you'll see a list of options for download. Go to the **Java SE Development Kit** menu of options. Do not go to the demos and samples (the menus look very similar, so make sure to read the heading at the top).

**Install**

You've got a lot of options here, but the two you care about are the Windows options.

If your computer is only a few years old, you're most likely going to download the **64-bit option**. If your computer is a little older you can [**follow these instructions(opens in a new tab)**](https://support.microsoft.com/en-us/kb/827218) to double check.

I'm going to download the 64-bit option (highlighted below).

Accept the license agreement, and download it. Once you've downloaded it, go ahead and double click it to install.

3: Verify that Java is Installed

Go back to **Step 1** and follow the instructions to open your Terminal and verify that you have Java version 7 or higher installed.

**Do not move on with Android Studio install until after you have installed the JDK**. Without a working copy of Java, the rest of the process will not work. If you can't get the download to work, look for error messages, and try googling to find a solution.

[**Link to Java download site**](http://www.oracle.com/technetwork/java/javase/downloads/index.html)

**Views**

The first thing in Android you need to learn is something called **Views**.

Camel case is a convention that is not limited to programming. If you've ever used FedEx, listened to an iPod, created a PowerPoint, or eaten at McDonalds, you've encountered camel case!

We used a lot of new words in this video!

* Layout
* User Interface
* TextView
* ImageView
* Button
* Camel case

Don't worry, you don't have to memorize them. You can look up their definitions [**by clicking here(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics) and searching for them. Believe it or not, professional developers don't memorize everything -- looking up information is a key part of the job! You can look up their definitions [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

Here are the words we introduced in this video:

* Code
* XML
* Integrated development environment (IDE) - This is the tool in which we will write our code. Similar to how we can use GMail as the tool to write an email, to write code for Android, we will use an IDE an IDE called Android Studio.

You can look up their definitions [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

Here are the words we introduced in this video:

* XML element
* Tag
* Self-closing tags
* Attributes
* Syntax

You can look up their definitions [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

Believe it or not, professional developers don't memorize everything -- looking up information is a key part of the job!

**Correction:** We incorrectly listed one of the attributes to be android:layout\_weight. The correct attribute is android:layout\_width.

Words we used in this video:

* Android Studio
* Device
* Density-Independent Pixels

You can look up their definitions [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

In case you're wondering, 48dp translates to approximately 9mm in physical size. See more info [**here(opens in a new tab)**](http://www.google.com/design/spec/layout/metrics-keylines.html#metrics-keylines-touch-target-size)

**What does Density-Independent Pixel actually mean?** You might be familiar with a pixel, which is a small illuminated area on a screen. Screens are essentially made up of hundreds of thousands of these pixels.

**So what is a "density independent" pixel?** Well, better screens will often have more pixels in the same amount of space. The number of pixels in a fixed space is known as the screen's pixel density. In case you're wondering, 48dp translates to approximately 9mm in physical size. See more info [**here(opens in a new tab)**](http://www.google.com/design/spec/layout/metrics-keylines.html#metrics-keylines-touch-target-size)

**Update: Use Layout Editor instead of XML Visualizer**

In many parts of the program demonstrations, instructors have used an in-house XML Visualizer to write and preview the XML code snippets. Here is the URL to the XML Visualizer used in the demo videos: [**https://udacity.github.io/android-layout/(opens in a new tab)**](https://udacity.github.io/android-layout/) or [**http://labs.udacity.com/android-visualizer/(opens in a new tab)**](http://labs.udacity.com/android-visualizer/)  
However, the updated Android Studio IDE has launched its own [**Layout Editor(opens in a new tab)**](https://developer.android.com/studio/write/layout-editor) (go through the components on this page) that allows you to build layouts by dragging UI elements into a visual design editor instead of writing layout XML by hand. To make it similar to how real developers build a UI, we recommend you to **use the Layout Editor in Android Studio IDE** instead of using the temporary XML Visualizer built by Udacity.  
We are bringing XML Visualizer down intentionally. You will find this tutorial relevant to get started: [**Build a simple user interface using Android Studio Layout Editor(opens in a new tab)**](https://developer.android.com/training/basics/firstapp/building-ui)

[**Material Design color palette(opens in a new tab)**](http://www.google.com/design/spec/style/color.html#color-color-palette)

More info on [**Hex colors(opens in a new tab)**](http://www.w3schools.com/colors/colors_hex.asp)

[**Click here(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics) and search for:

* Hex color

New term we used in this video:

* Documentation

You can look up the definition [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

If you're wondering which link to click, look for the first one that starts with **developer.android.com**. Any website that starts with that web address is part of the official Android developer website and documentation.

**ViewGroup**

* ViewGroups
* Root View
* Parent
* Child
* Sibling

You can look up their definitions [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

[**Link to the code(opens in a new tab)**](https://gist.github.com/anonymous/cd7fda2fa5c4062acb92)

Here are the words we introduced in this video:

* LinearLayout
* RelativeLayout

You can look up their definitions [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

[**Guest list layout**](https://gist.github.com/anonymous/cd7fda2fa5c4062acb92#file-guest-list-layout)

|  |  |
| --- | --- |
|  | <LinearLayout |
|  | android:orientation="vertical" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content"> |
|  |  |
|  | <TextView |
|  | android:text="Guest List" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" /> |
|  |  |
|  | <TextView |
|  | android:text="Kunal" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_height="wrap\_content" /> |
|  |  |
|  | </LinearLayout> |

Relative to Parent

[**activity\_main.xml**](https://gist.github.com/anonymous/e7eb5ceff9dedfc376eb#file-activity_main-xml)

|  |  |
| --- | --- |
|  | <RelativeLayout |
|  | xmlns:android="http://schemas.android.com/apk/res/android" |
|  | android:layout\_width="match\_parent" |
|  | android:layout\_height="match\_parent" |
|  | android:padding="16dp"> |
|  |  |
|  | <TextView |
|  | android:text="I’m in this corner" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_alignParentBottom="true" |
|  | android:layout\_alignParentLeft="true" /> |
|  |  |
|  | <TextView |
|  | android:text="No, up here" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_alignParentLeft="true" /> |
|  |  |
|  | <TextView |
|  | android:text="Wait, I’m here" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_alignParentBottom="true" |
|  | android:layout\_alignParentRight="true" /> |
|  |  |
|  | <TextView |
|  | android:text="Actually, I’m here" |
|  | android:layout\_height="wrap\_content" |
|  | android:layout\_width="wrap\_content" |
|  | android:layout\_alignParentTop="true" |
|  | android:layout\_alignParentRight="true" /> |
|  |  |
|  | </RelativeLayout> |

How to Learn More on Your Own

After this course, if you choose to continue learning Android, an important skill to have is the ability to learn on your own. You might find resources out there that feel too advanced for where you are on your journey, but we want you to become accustomed to how developers speak and share their ideas. You don’t have to understand every word, but you can skim for important ideas. Or you can google search for terms that you aren’t familiar with.

Read your first Android blogpost article

Start by reading [**this post(opens in a new tab)**](http://android-developers.blogspot.com/2014/08/material-design-in-2014-google-io-app.html?utm_source=udacity&utm_medium=course&utm_campaign=android_basics) on the [**Android Developers blog(opens in a new tab)**](http://android-developers.blogspot.com/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics). It's written by Google Design Advocate, Roman Nurik, who was the lead designer on the Google I/O app. Google I/O is an annual conference that Google holds for developers.

Follow official Android Development channels on social media

Aside from the blog, you can get the latest news about Android development via:

* [**Android Developers Twitter page(opens in a new tab)**](https://twitter.com/AndroidDev)
* [**Google Developers YouTube channel(opens in a new tab)**](https://www.youtube.com/user/GoogleDevelopers)

Kirill's Favorite Resources

In addition to the official channels for Android development news, there’s a ton of content online, and a vibrant ecosystem of Android developers who are happy to share their knowledge through blog post articles, social media tips, and conference talks.

Here are some of Kirill’s favorite Android resources:

* [**Chris Banes' blog(opens in a new tab)**](https://chris.banes.me/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics): A blog that gives you a deeper look into Android support libraries.
* [**Fragmented Podcast(opens in a new tab)**](http://fragmentedpodcast.com/): A weekly podcast filled with Android development discussion.

# **Keeping up with the changes**

Google is constantly improving the Android platform and adding new features. This is great for you as a developer, but it makes learning harder sometimes. Recently Google released ConstraintLayout; a tool that makes it super fast to create responsive UIs with many different types of components. ConstraintLayout is a great tool to have on the sleeve, but for this class, we will use RelativeLayout, LinearLayout simpler.

All of this matters to you because the new project templates in Android Studio now use ConstraintLayout as default, which makes the code you see on your computer a bit different from what is on the screen.

# **Current Layout File**

In the new versions of Android Studio, after choosing the Empty Activity template, the layout file app/src/main/res/layout/activity\_main.xml will look like this:

**<?xml version="1.0" encoding="utf-8"?>**

<android.support.constraint.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/activity\_main"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!"

app:layout\_constraintLeft\_toLeftOf="@+id/activity\_main"

app:layout\_constraintTop\_toTopOf="@+id/activity\_main"

app:layout\_constraintRight\_toRightOf="@+id/activity\_main"

app:layout\_constraintBottom\_toBottomOf="@+id/activity\_main" />

</android.support.constraint.ConstraintLayout>

Note the use of ConstraintLayout, and that TextView has a list of limiters that position it within ConstraintLayout.

# **Modify the Layout File**

Unlike the above code, our videos and start code assume that the template looks more like the following, using as the root of the view a RelativeLayout:

**<?xml version="1.0" encoding="utf-8"?>**

<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!" />

</RelativeLayout>

When you create your new project, go to app/src/main/res/layout/activity\_main.xml and copy and paste the above code. Then you're ready to go!

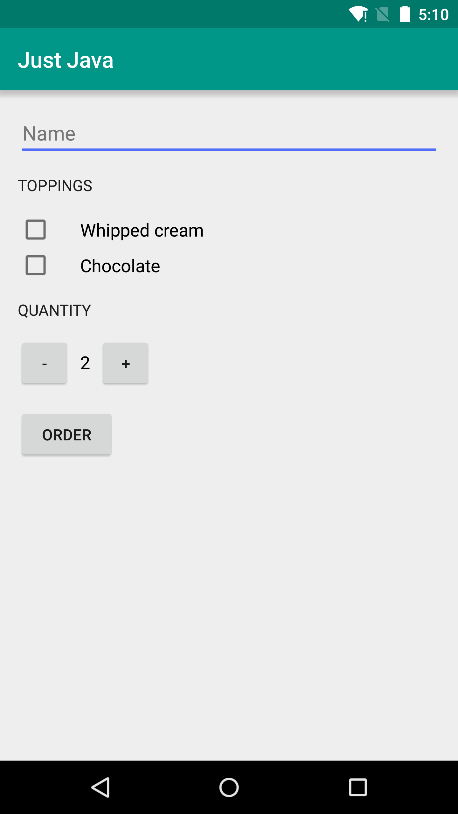
# **Learn More About Constraint Layout**

If you want to understand more about the great features that ConstraintLayout provides, check out the documentation at: [**https://developer.android.com/studio/write/layout-editor.html(opens in a new tab)**](https://developer.android.com/studio/write/layout-editor.html)

Additionally, for those wanting a hands-on demo using Android Studio Layout Editor with ConstraintLayout, here's a [**Code Lab(opens in a new tab)**](https://codelabs.developers.google.com/codelabs/constraint-layout/index.html?index=..%2F..%2Findex#5). Note that this is important information, but is beyond the current scope of this course.

**Welcome to Android Basics: User Input**

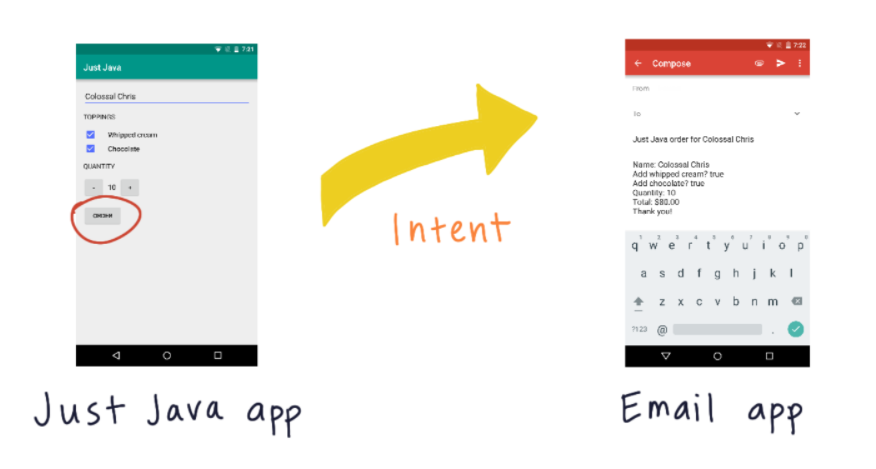
In this course, you'll make an Android app with buttons and text fields that a user can interact with. Together we’ll make the Just Java app, which allows a user to fill out a form to order coffee.



Just Java App

We picked this project because nearly every popular app involves accepting some type of user input and doing something with that data. For example, in a chat app, you type in a message and it gets sent to a friend. Knowing how to accept user input is a critical milestone on the way to becoming an experienced app developer.

To build the Just Java app, we’ll introduce you to the basic fundamentals of Java and computer science, as well as how to debug and fix errors in the code. We’ll explore more of what the Android platform offers, by building more complex user interfaces, adding support for another language, and taking advantage of other apps on the device.



To build the Just Java app, we’ll introduce you to the basic fundamentals of Java such as Intents.

**Course Overview**

You'll be building the Just Java App throughout the course. In the first lesson: **Making an App Interactive,** you'll:

* Create a new project in Android Studio for the Just Java app
* Plan and build a layout for it
* Write code to respond to button clicks
* Test your code and learn how to debug code crashes
* Implement variables in code related to fields and button clicks in the app.

In the second lesson, you'll learn more about **Android Studio and the Android Emulator** to become more familiar with the development and testing environment for the Just Java app.

In the next lesson on **Making an App Interactive**, you'll learn more about

* interface components for the Just Java app
* Implementing more Java code to respond to user input.

The **Practice Set lesson on Making an App Interactive** will take you through several exercises and examples of handling user input in Android apps.

Then you'll complete a **project** to practice what you've learned about making an interactive Android app.

In the second half of the course, you'll learn more about **Object-oriented programming in Java** including:

* variables, data types, methods, and classes
* You'll complete a **practice set** on object-oriented programming

In the **second course project**, you'll create a more complex interactive Android app using the object-oriented programming skills you developed in this part of the course.

By the end of this course, you'll have 3 Android apps for your Android developer portfolio

We’ll be your instructors guiding you through the course, so let’s get started!

**The Need for Variables**

* Variables
* Robust
* Data type
* int
* Variable name
* Assignment operator
* Initializing

DDMS stands for Dalvik Debug Monitor Server, and is a tool in Android to help you debug your app. For more info, click this [**link(opens in a new tab)**](https://developer.android.com/studio/profile/android-profiler).

Here are the words we introduced in this video:

* Debug
* Crashes
* Compile time error
* Runtime error
* System log
* Stacktrace

You can look up their definitions [**in the Vocab Glossary(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics).

[**Link to MainActivity gist(opens in a new tab)**](https://gist.github.com/anonymous/6dde8cc012ac94aaf9ea)

[**Check out the vocabulary word list(opens in a new tab)**](https://developers.google.com/android/for-all/vocab-words/?utm_source=udacity&utm_medium=course&utm_campaign=android_basics) and search for:

* Variable scope
* Local variables
* Global variable
* Variable declaration

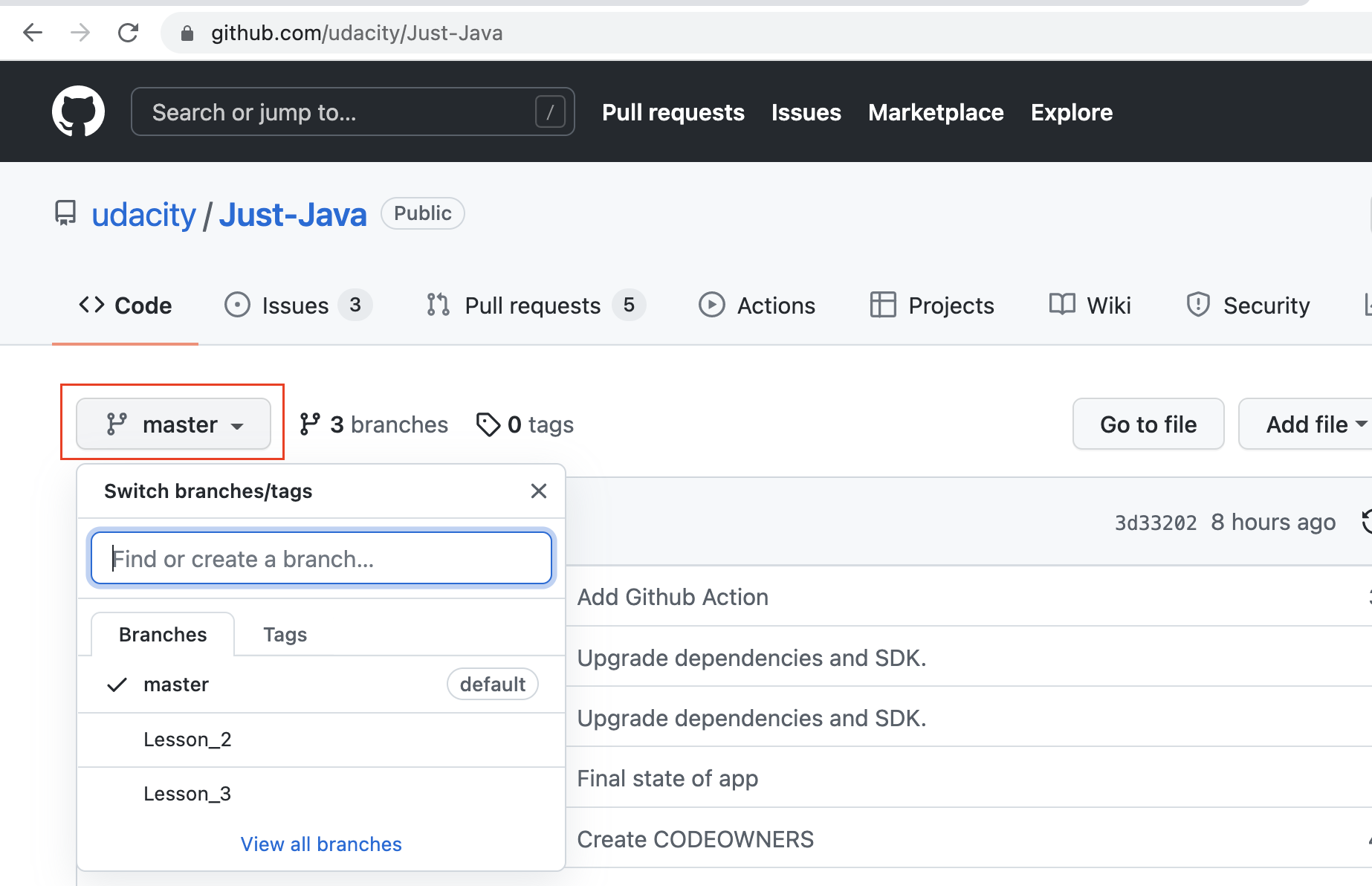
**Lesson Review**

Let's take a moment and reflect on what you learned in this lesson to build the Just Java app

* You created a new project in Android Studio
* You planned and built the XML layout for the app
* Responded to button clicks and input in the app
* You learned a bit about responding to app crashes, and that these are normal!

The Just Java app now contains some core interface elements, and you'll continue to expand it's functionality in the next lessons.

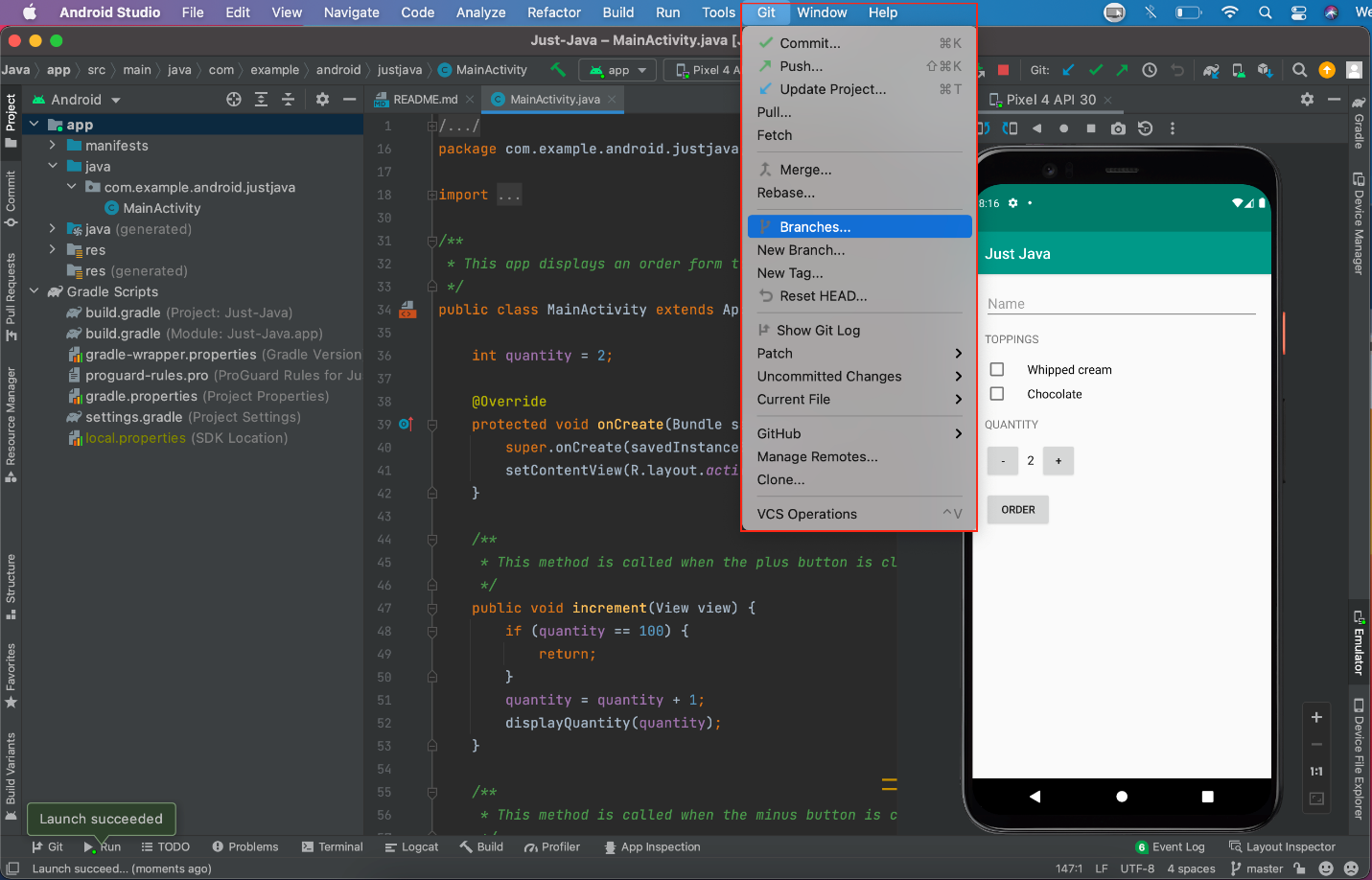
[**Just-Java Github repo(opens in a new tab)**](https://github.com/udacity/Just-Java) - We will develop our app gradually in each subsequent lesson. This repository has three branches, each corresponding to a particular lesson. The solution to the current lesson is present in the "master" branch.



Switch to the "master" branch to see the application code at this stage.

How to use Just-Java Github repository

* Fork [**this repository(opens in a new tab)**](https://github.com/udacity/Just-Java) to your Github account.
* Open Android Studio in your local machine.
* We recommend you to use the [**Version Control System(opens in a new tab)**](https://developer.android.com/studio/intro#version_control_basics)(VCS) in your Android Studio 3.2.x+ IDE to clone the repository directly in your IDE.
* You may have to use the File > Settings > Version Control (VCS) menu option to set up Git settings.
* Refer to this [**IntelliJ document(opens in a new tab)**](https://www.jetbrains.com/help/idea/version-control-integration.html) for details.
* Working with VCS will make it easy for you to push your updates or switch between branches to your remote repository in Github.



Using the version control in Android Studio to switch between branches

* [**Android Developers Blog Announcing Kotlin(opens in a new tab)**](https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html)
* [**Install JDK(opens in a new tab)**](http://www.oracle.com/technetwork/java/javase/downloads/index.html) if you don't have it
* [**Install IntelliJ**](https://www.jetbrains.com/help/idea/install-and-set-up-product.html)

Installing the Java Development Kit (JDK)

If you don't have the latest JDK already installed on your computer, follow the steps below. You will need to have the JDK installed to run Kotlin programs.

The JDK is freely available, and you can download it here: [**http://www.oracle.com/technetwork/java/javase/overview/index.html(opens in a new tab)**](http://www.oracle.com/technetwork/java/javase/overview/index.html).

The JDK or the JRE?

The JRE (Java Runtime Environment) is needed for running Java and Kotlin programs. The JDK (Java Development Kit), on the other hand, includes the JRE, plus the development tools you'll need for writing and running Java programs. You need the JDK for writing Kotlin Programs.

Steps to install the JDK

**1. Uninstall any older versions of the JDK/JRE**

We recommend that you install only the latest JDK.

**2. Download the JDK**

You can download the JDK for free here: [**http://www.oracle.com/technetwork/java/javase/downloads/index.html(opens in a new tab)**](http://www.oracle.com/technetwork/java/javase/downloads/index.html).

1. Click the "Download" button under the JDK for the latest Java SE version.
2. Check "Accept License Agreement".
3. Choose the JDK for your operating system.

**3. Install the JDK (for Mac)**

From either the Downloads window of the browser, or from the file browser, double-click the .dmg file to launch the install file.

1. A Finder window appears with an icon of an open box and the name of the .pkg file.
2. Double-click the package icon to launch the Install app, and follow the prompts as they appear.
3. You might need to enter the administrator password to continue.
4. Feel free to delete the .dmg file to save space after the installation is complete.

**3. Install the JDK (for Windows)**

Run the downloaded installer (e.g., jdk-10.0.x\_windows-x64\_bin.exe), which installs both the JDK and the JRE.

By default, the JDK will be installed in the directory "C:\Program Files\Java\jdk-10.0.x", where x denotes the version number; and the JRE in "C:\Program Files\Java\jre-10.0.x".

Accept the defaults, and follow the screen instructions to install the JDK.

**4. Add the JDK installation path to PATH (Windows only)**

Windows searches the current directory and the directories listed in the PATH environment variable (system variable) for executable programs.

1. Open "Control Panel" -> "System" -> "Advanced system settings" -> "Environment Variables".
2. Under "System variables", scroll down to select "Path" and click "Edit...".
3. Append to the existing Path value a semi-colon ";" then the JDK's "bin" directory (e.g. ";C:\Program Files\Java\jdk-10.0.0\bin").

Download and Install IntelliJ IDEA

1. [**Download IntelliJ IDEA(opens in a new tab)**](https://www.jetbrains.com/idea/download/index.html) for your operating system.
2. Do the following, depending on your operating system:

**Windows:**

1. Run the ideaIC.exe or the ideaIU.exe file you have downloaded.
2. Follow the instructions in the installation wizard.

**Mac:**

1. Double-click the ideaIC.dmg or ideaIU.dmg file you have downloaded to mount the macOS disk image.
2. Copy IntelliJ IDEA to the Applications folder.

More information

Check out this [**link(opens in a new tab)**](https://www.jetbrains.com/help/idea/install-and-set-up-product.html) for more information on how to install and set up IntelliJ.

Verify IntelliJ Installation

Check that your IntelliJ is installed and up-to-date by following these steps below:

1. To check whether you need to install JDK, in a terminal window, type:

java -version

javac -version

1. If you do not have it, or do not have the latest version, download and install JDK.
2. Install IntelliJ. This course uses IntelliJ and we recommend you use the same IDE.
3. Start IntelliJ.
4. Install any updates and additional content you are prompted for.
5. Select IntelliJ IDEA -> Check for updates… until there are no more updates.

Code to enter into REPL:

fun printHello () {

println ("Hello World")

}

**printHello**()

* [**Kotlin Language Documentation**](http://kotlinlang.org/docs/reference/)

Kotlin Koans

The official [**Kotlin documentation(opens in a new tab)**](https://kotlinlang.org/) includes rich tutorials called [**Kotlin Koans(opens in a new tab)**](https://try.kotlinlang.org/#/Examples/Hello,%20world!/Simplest%20version/Simplest%20version.kt), a web-based interpreter and a complete set of reference documentation with examples. If you want more information on any topic in this course, or if you get stuck, [**https://kotlinlang.org/(opens in a new tab)**](https://kotlinlang.org/) is your best starting point.

* Go to the official [**Kotlin language documentation(opens in a new tab)**](https://kotlinlang.org/) and find the code for Hello World.
* If you prefer to work offline, you can download the entire reference documentation as a single [**PDF file(opens in a new tab)**](https://kotlinlang.org/docs/kotlin-docs.pdf).
* Go to the [**Kotlin Koans(opens in a new tab)**](https://try.kotlinlang.org/#/Examples/Hello,%20world!/Simplest%20version/Simplest%20version.kt). Find and run the program to print Hello World.

Aquarium fun (Optional)

When you are building an aquarium, it is good to have some resources in case you have questions. Open a Browser and search for "aquarium tutorial" or "how to set up an aquarium". There are many resources available, check out some of them!



Kotlin Documentation

Throughout this course you might find it useful to check the official Kotlin documentation every time you learn something new:

[**http://kotlinlang.org/docs/reference/**](http://kotlinlang.org/docs/reference/)

* [**Interfaces(opens in a new tab)**](https://kotlinlang.org/docs/reference/interfaces.html)
* [**Delegation(opens in a new tab)**](https://kotlinlang.org/docs/reference/delegation.html)
* [**Kotlin Language Documentation(opens in a new tab)**](http://kotlinlang.org/docs/reference/)
* [**Kotlin Koans**](https://try.kotlinlang.org/#/Examples/Hello,%20world!/Simplest%20version/Simplest%20version.kt)