



Making an Android App with no prior experience

A tutorial, a guide and a resource document

Xavier Tobin

xaviertobintech@gmail.com

There are two ways to approach this document:

1. Be at a computer, follow the explanations and instructions and you'll have an app and some basic skills to branch off by the end of it.
2. Simply read the document and pick up some skills along the way.

Before I begin, a personal message: If at first it seems too hard, google it, then try again. I started making my first app at 15 years old with literally zero prior experience, I just had interest and an idea. My first and only app currently sits at just below 70,000 downloads. There will be times when you are completely stumped: I recall spending several days trying to fix some bugs I came across, but if you are persistent enough you will be able to work it out. Let's begin.

What is Java?

Java is a programming language like C++, Python and Ruby. Most apps on the Android platform are written in Java, although games and some other apps are written in other languages.

Java is an OOP or Objected Oriented-Programming Language

This means that Java is a programming language based on the concept of objects, which are essentially fields of data that can run code and store variables.

For example, a String object is an object that contains any combination of letters, numbers and other characters. A String is formatted in quotation marks, here is an example use

```
String name = "Dennis";  
  
String surname = "Cometti";  
  
String FullName = name + " " + surname;
```



After this runs, the variable `FullName` will equal "Dennis Cometti". A `String` is an example of a basic object, other basic Objects in Java include Integers (any whole number), Textviews and ArrayLists. Booleans (a true or false value) and floating points (decimal values like 3.0) are examples of primitive variables.

Objects can also contain other objects and variables, for example you could define a 'Quote' Object that contains two values: The actual quote and the name of the quoted person.

A lot of the fundamentals in Java are essentially plain English

All of Java is written in English, the structure of the words change but if enough attention is given to it can actually be very easy to understand. For example:

```
String name = "DENNIS";  
  
name = name.toLowerCase();
```

It couldn't be any clearer, this will assign the lower case converted "DENNIS" ("dennis") to the 'name' variable.

After you have typed 'name.' Android Studio will give you a list of possible methods (like `toLowerCase()` or `toUpperCase()`) that can be used, so you get some guidance.

Classes, methods and objects in Java

- A **variable** holds a field of data. A variable might be a surname, your weight or the distance travelled by your car. A `String` is a variable that could contain "Dennis" and an `int` is a primitive variable that could contain the number 89.
- A **method** is a function (like `name.toLowerCase()`). Basically a method does things using variables or any number of lines of code. You can write your own methods, for example in the app we will be making soon we will be making a method called `getQuote()`. A method might display a quote on a screen or change the text in a `TextView`.
- An **object** holds both variables and methods; one object might be the details of your car, a second object might represent my car.
- A **class** is a template that holds variables, methods and other objects.

What is the difference between classes and objects?

A class would be a `Car` (containing no details but rather a template).

An object would be your `Car` (containing all the details about the car).

A class would be a `String` (containing no details but rather a template).

An object would be a 'name' (containing the `String` "Dennis").



If you are confused, don't worry, once you have followed the instructions you'll understand it much clearer.

Here is a Java website which I highly recommend, explaining it in more depth

<http://code.tutsplus.com/tutorials/java-tutorial--mobile-2604>

Some more Java related resources

The syntax of Java: http://www.tutorialspoint.com/java/java_basic_syntax.htm

Let's make an app

Android Studio

Android Studio is the new Android Integrated Development Environment, don't let the words confuse you – it's essentially a program that has all the tools you need to make an app.

Some people come across some issues installing Android Studio, so make sure you are Googling any issues that you come across in this stage, but you should be fine. You'll come across many things you don't understand when making apps but I guarantee you 1000 people have had the same problem before and there will be help or tutorials online (you shouldn't need them for this exercise).

Instruction #1: Download and install the Java JDK

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

Instruction #2: Download and install Android Studio, but don't open it yet <https://developer.android.com/sdk/index.html>

Strings in Android

Strings as mentioned earlier, are used everywhere: App Dialogs, Titles, Messages, Error Logs and literally wherever you see characters. The problem is, when you are making an app with a lot of strings it can get hard to make changes to them all.

So Google created a solution: a single file that stores all of your strings in one place, so you can get that one file translated and refer to those strings in tons of different parts of the code.

Here's a link from Google that can explain it in more detail:

<http://developer.android.com/guide/topics/resources/string-resource.html>



How Android Studio works

Android Studio contains all the tools you need to make an app: for this tutorial you won't be using many. When you create a new 'Project' (App) Android Studio will generate all the files and folders necessary to begin a project.

I'm going to walk through creating a simple Quote app!

What will the app do?

It will show a quote plus the name of the person who made the quote and loop through as many quotes as you like when you tap the screen.

I'll be explaining everything along the way.

Instruction #3: Open Android Studio and click the 'create new project' button, then follow exactly the screenshots over the next few pages to set up the new project.

Create New Project

New Project
Android Studio

Configure your new project

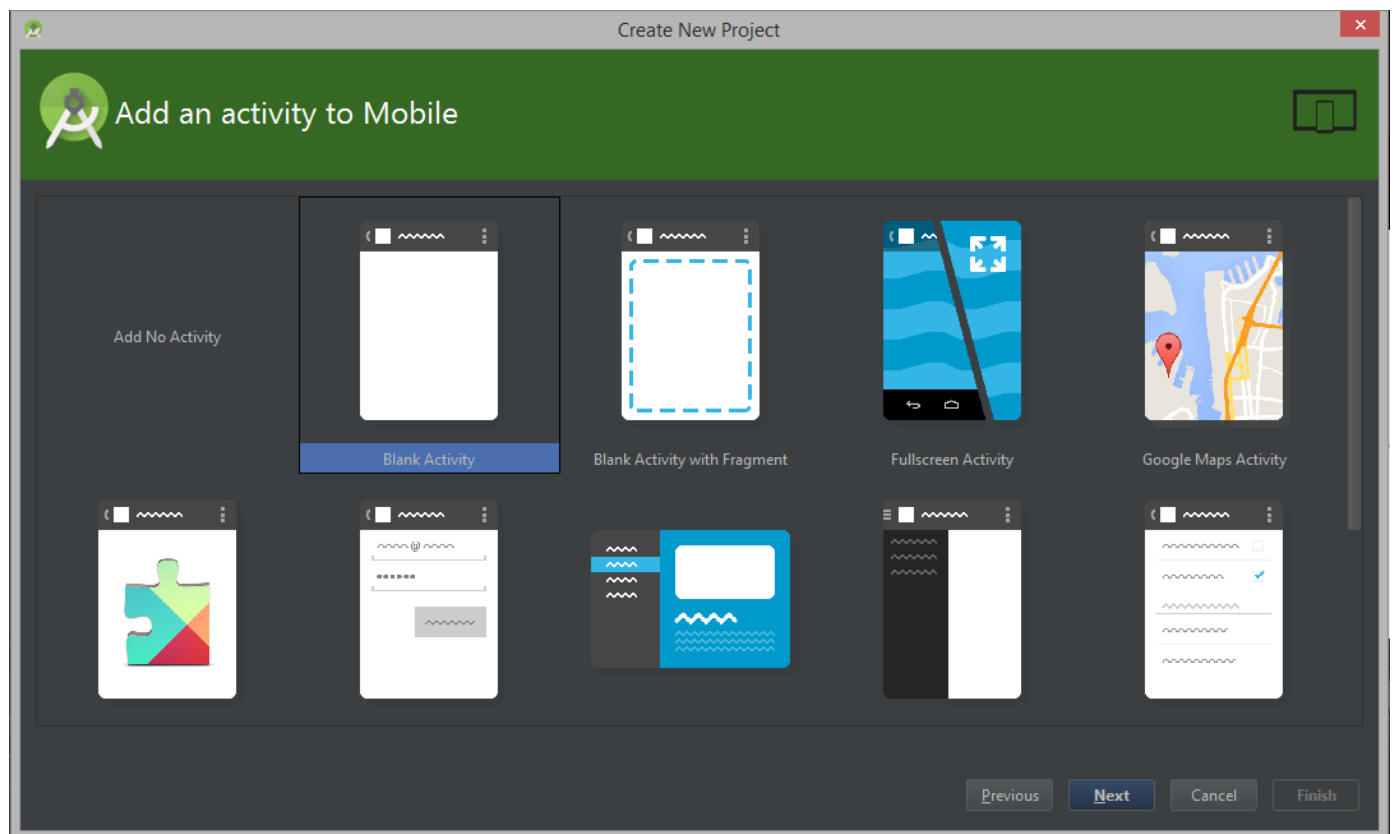
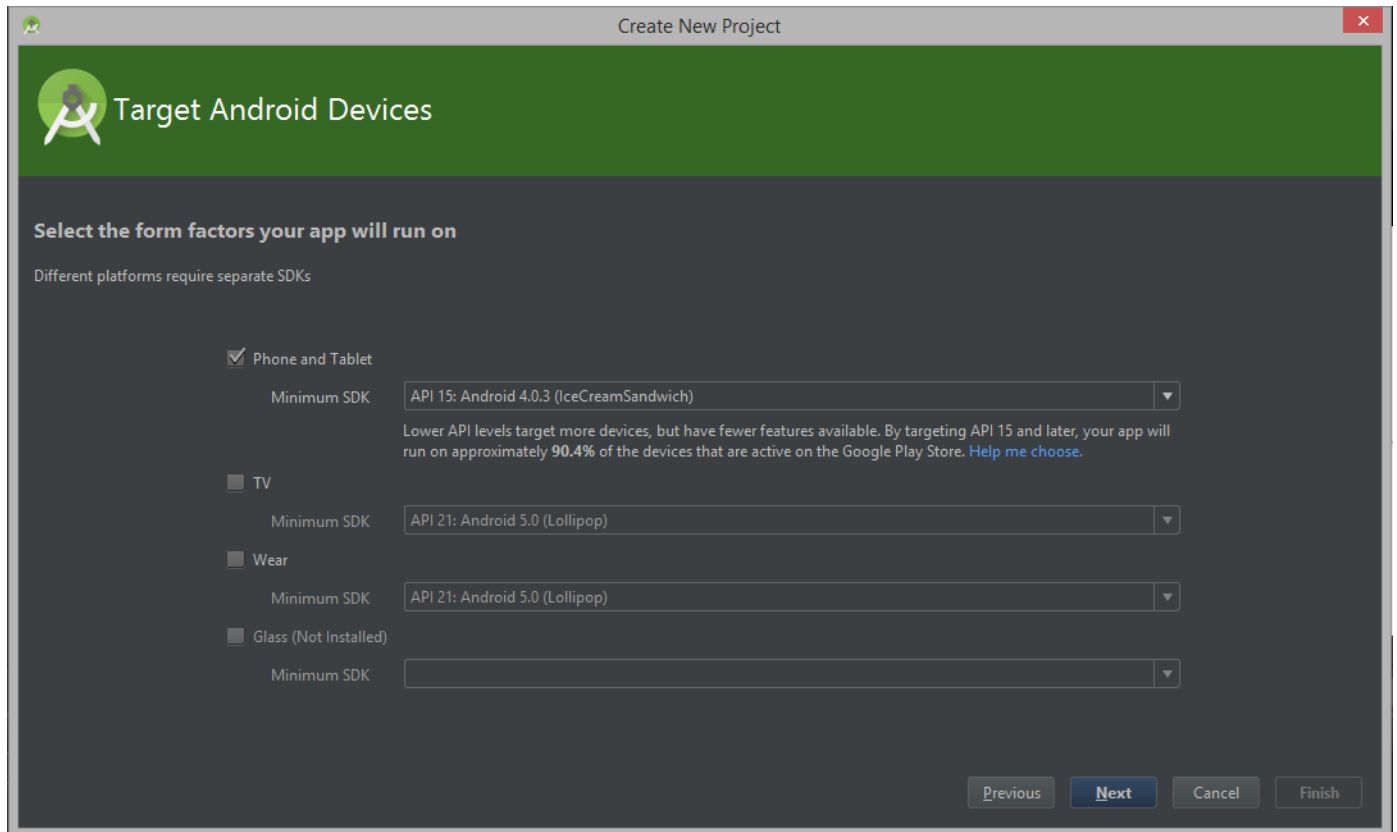
Application name: The Quotebook

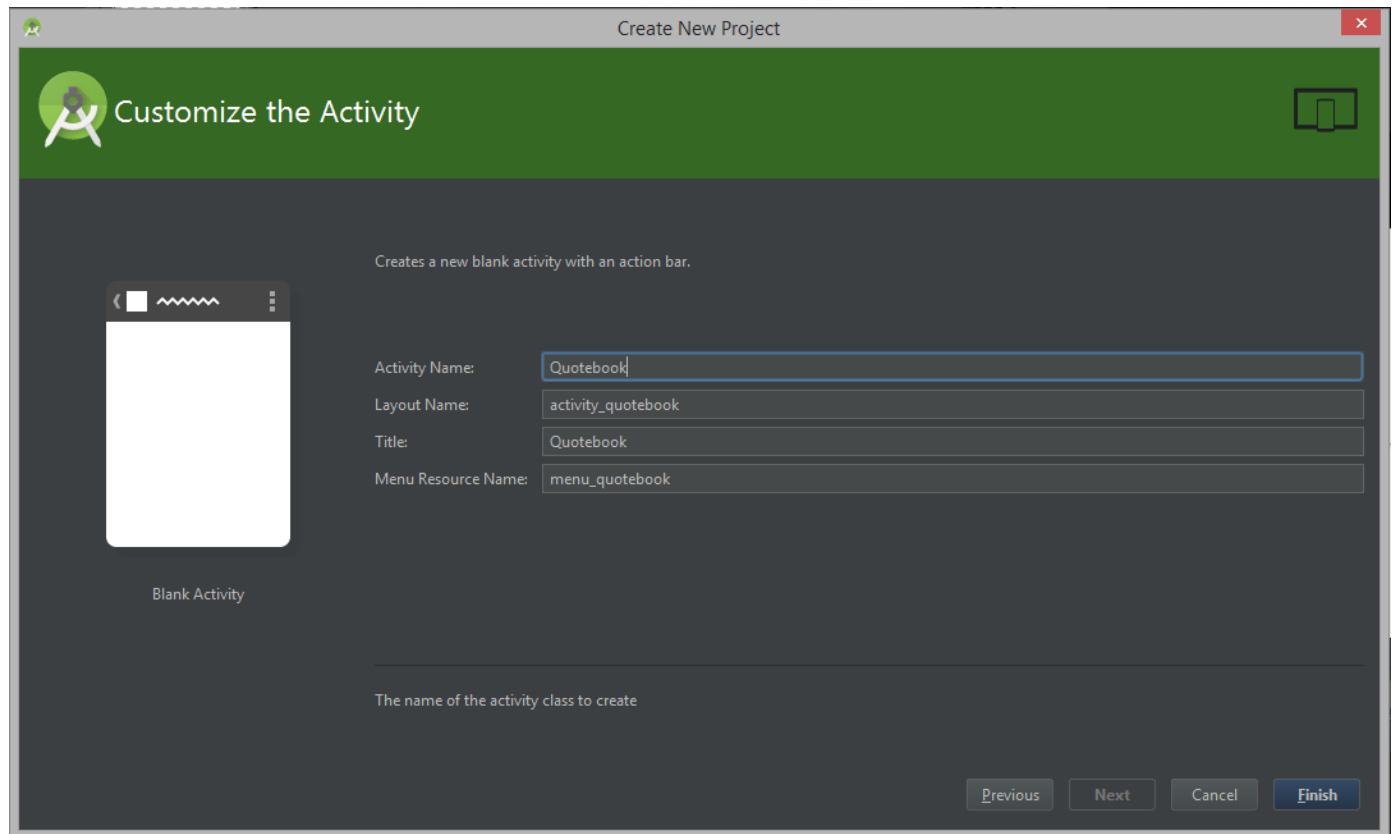
Company Domain: com.theoneandonly

Package name: theoneandonly.com.thequotebook [Edit](#)

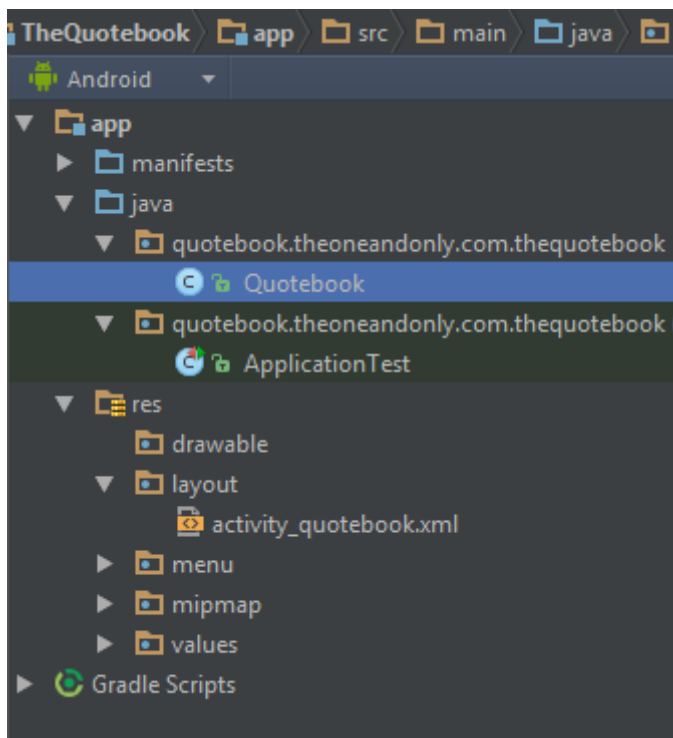
Project location: C:\Users\Family\AndroidStudioProjects\TheQuotebook ...

Previous Next Cancel Finish





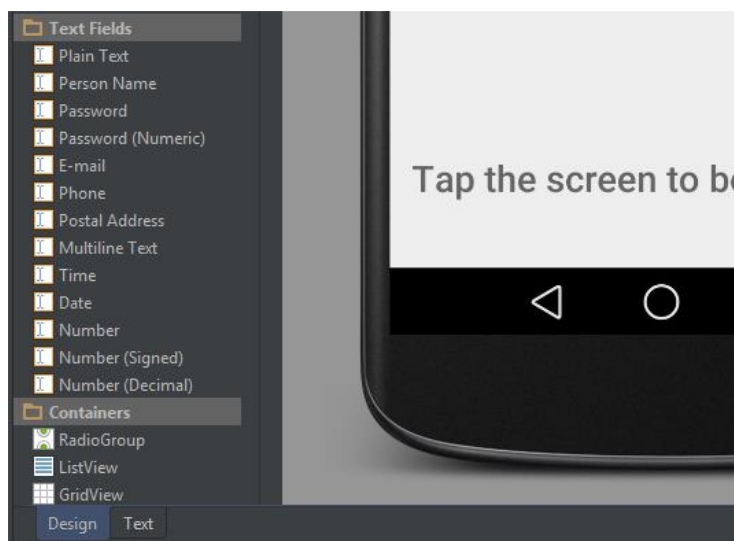
After clicking finish, Android Studio will do its thing, creating all the necessary folders and files:



The 'java' folder is where all the code will be stored, while the 'res' folder is for all the visual elements (icons, images, strings, colours) which you can reference and use in your code.

The layout folder contains the file for the first step in making the app. You should already see this layout file open but double click 'activity_quotebook.xml' just in case.

This is actually a drag and drop editor, but for this example we'll be copying and pasting the layout configuration. Feel free to change sizes, colours and some elements yourself, this part is quite simple to use.



Instruction #4:

Just to the bottom left of the phone and a little further are two tabs – ‘Design’ and ‘Text’ – the design tab is a drag and drop interface, while the Text tab is the actual UI XML data. For now, click on the text tab and replace all of the text inside with the following:

```
<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:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:id="@+id/touch">

    <TextView
        android:text="Tap the screen to begin"
        android:id="@+id/quote"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif-medium"
        android:textSize="26sp"
        android:layout_gravity="center"
        android:layout_above="@+id/person"
        android:textColor="#ff656565"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:text=""
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:fontFamily="sans-serif-medium"
        android:textSize="26sp"
        android:gravity="end"
        android:textColor="#ff767676"
        android:layout_gravity="center"
        android:id="@+id/person"
        android:layout_alignParentBottom="true"
        android:layout_alignLeft="@+id/quote"
        android:layout_alignStart="@+id/quote" />

</RelativeLayout>
```

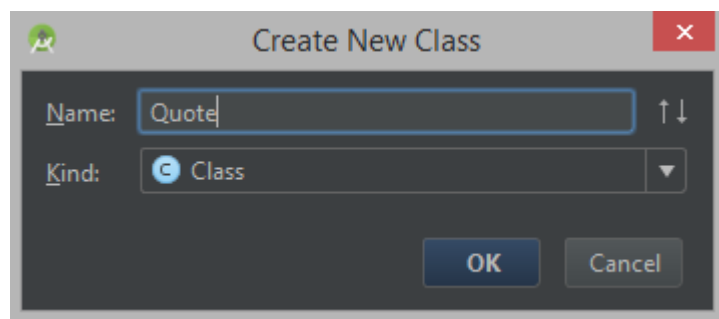
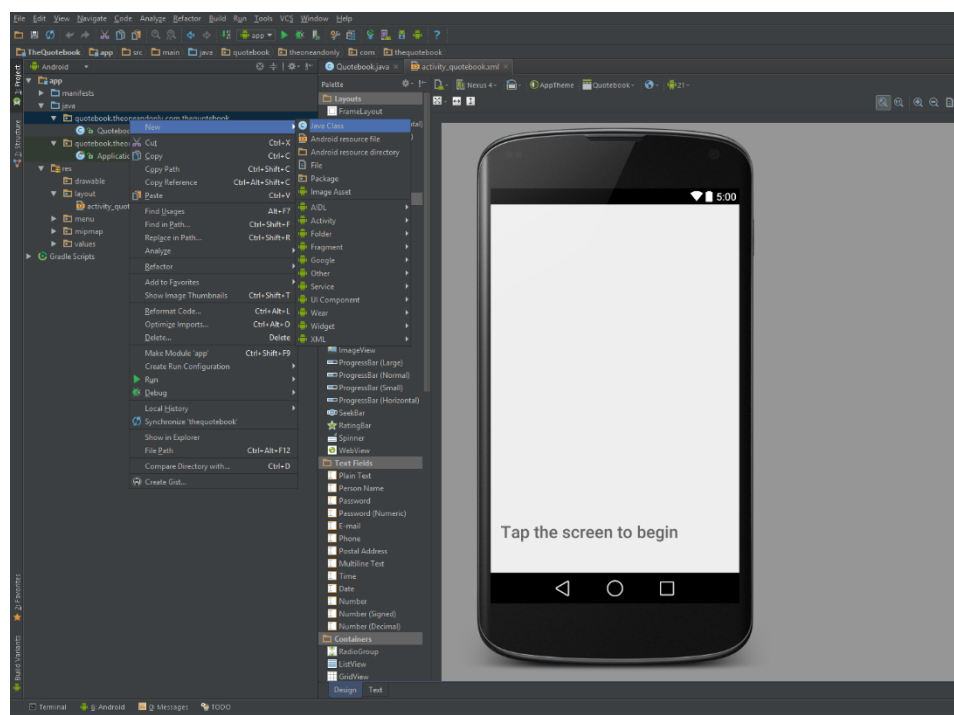


This has formatted the layout of the main app Activity, but you can change some things around. Try changing the text from “Tap the screen to begin” to something else. Extra points if who can change the font colour.

Instruction #5:

This is where things get a bit trickier, now we have to make a new class and the Quote Object we spoke about earlier.

These screenshots show how to make a new class:





After creating the Quote class Android Studio should open it, and we'll be filling it with some code now. This is what should be in your quote class:

```
package quotebook.theoneandonly.com.thequotebook;

public class Quote {

}
```

You should see 'public class Quote{ }', in between these two squiggly brackets paste this code:

```
public String quote;

    public String person;

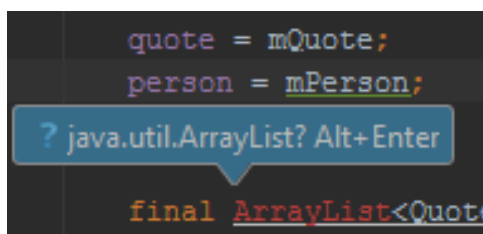
    public Quote(String mQuote, String mPerson){
        super();

        quote = mQuote;
        person = mPerson;

    }

    public String getPerson() {
        return person;
    }

    public String getQuote() {
        return quote;
    }
}
```



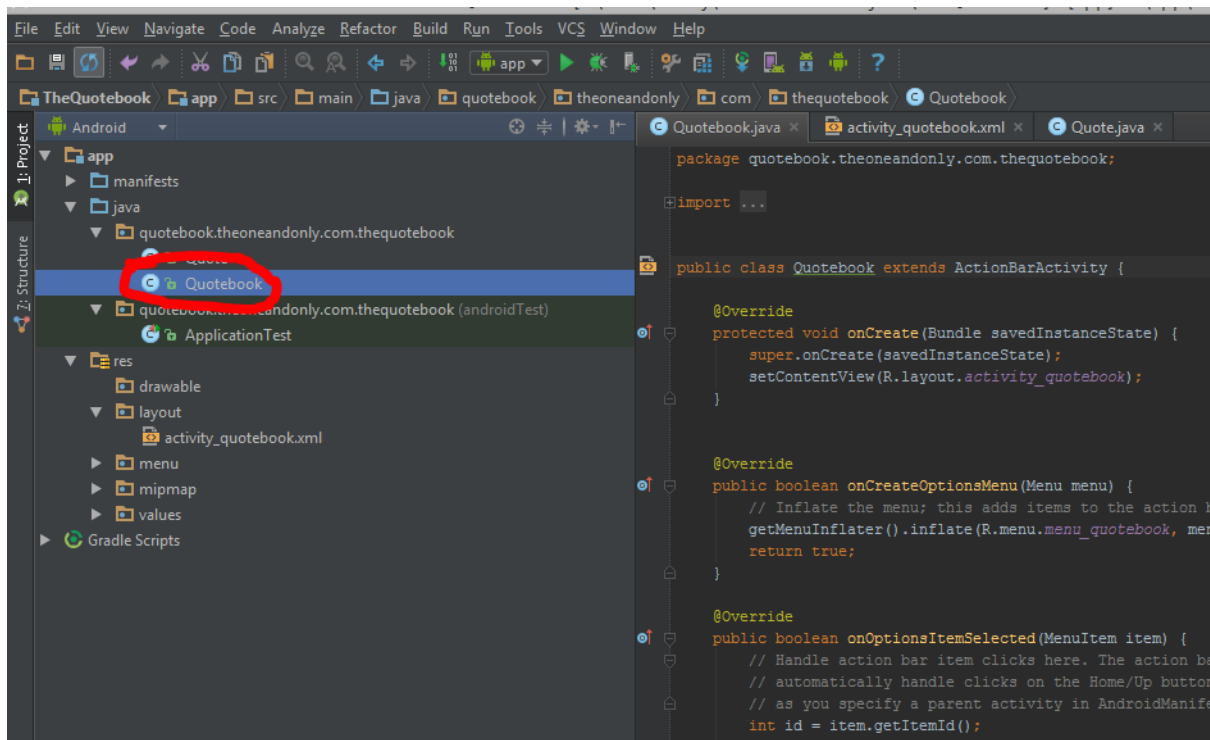
Click OK to any pop up boxes, and if you ever see something like this just click 'Alt+Enter':

What this class does is allows the app to create a Quote object that we can use, you 'instantiate' the class and pass through a quote and name (where it says public Quote(String mQuote, String mPerson)) and then you can retrieve the quote or person name later. More on this soon.



Instruction #6

Click on the Quotebook class here:



You should see something like this in all the pre-generated code in the Quotebook class:

```
@Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_quotebook);

    }
```

Then copy and paste the code on the following page AFTER the 'setContentView()' line, but still inside the squiggly brackets.

It seems like a lot of code, but I'll explain it all.



```
RelativeLayout touch = (RelativeLayout) findViewById(R.id.touch);
final TextView quoteText = (TextView) findViewById(R.id.quote);
final TextView personText = (TextView) findViewById(R.id.person);

final ArrayList<Quote> quoteList = new ArrayList<Quote>();

Quote quote4 = new Quote("You're more of a fun vampire. You don't suck
blood, you just suck.", "Troy Barnes");
quoteList.add(quote4);

Quote quote1 = new Quote("Cool Beans", "Rod Kimble");
quoteList.add(quote1);

Quote quote2 = new Quote("How can mirrors be real if our eyes aren't real",
"Jaden Smith");
quoteList.add(quote2);

Quote quote3 = new Quote("That's like me blaming owls for how bad I suck at
analogies.", "Britta Perry");
quoteList.add(quote3);

Quote quote5 = new Quote("I was gonna be the first person in my family to
graduate from community college. Everyone else graduated from normal
college", "Troy Barnes");
quoteList.add(quote5);

touch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if (count < quoteList.size()) {

            Quote q = quoteList.get(count);

            quoteText.setText(q.getQuote());

            personText.setText(q.getPerson());

            count = count + 1;

        } else{

            count = 0;

        }

    }
});
```



You'll notice some red squiggly lines telling you there is an error, **so change this:**

```
public class Quotebook extends ActionBarActivity {
```

To this:

```
public class Quotebook extends ActionBarActivity {  
    int count = 0;
```

This is what the class should look like after the have been copied and pasted:

<http://pastebin.com/3XuU4bSA>



That's the bulk of the code done so far!

Explanation time!

At this point, assuming you haven't programmed extensively before, you are probably confused. I'll go through each section of code step by step:

```
setContentView(R.layout.activity_quotebook);
```

```
    RelativeLayout touch = (RelativeLayout) findViewById(R.id.touch);
```

```
    final TextView quoteText = (TextView) findViewById(R.id.quote);
```

```
    final TextView personText = (TextView) findViewById(R.id.person);
```

The first line (setContentView) sets the app page (Activity) to be the layout we created earlier, essentially loading the main Quote screen.

The other lines just find the Textviews in the activity layout and declare them as an Object so we can change the text in them.



```
final ArrayList<Quote> quoteList = new ArrayList<Quote>();
```

```
Quote quote1 = new Quote("You're more of a fun vampire. You don't suck blood,  
you just suck.", "Troy Barnes");
```

```
quoteList.add(quote1);
```

```
Quote quote2 = new Quote("Cool Beans", "Rod Kimble");
```

```
quoteList.add(quote2);
```

The first line here creates an Array/List that we can add as many quotes as we like to, note how the List is called 'quoteList'.

The next 4 lines are where the Quote class we created earlier are coming in to play. What we are doing here is passing a quote and a person's name (separated by a comma) through to the Quote class and it becomes a variable, we then add that Variable to the quoteList.

You can add as many quotes by just adding an extra value to the variable name (quote7) add then adding it to the quote list.

The next explanation requires a full page, keep scrolling!



This is where it gets a little tricky:

```
touch.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
  
        if (count < quoteList.size()) {  
  
            Quote q = quoteList.get(count);  
            quoteText.setText(q.getQuote());  
            personText.setText(q.getPerson());  
  
            count = count + 1;  
  
        } else{
```

This looks complex but if you imagine it as a plain English sentence it makes far more sense.

Basically: If every quote has been cycled through, set the count to 0 so it starts again.

Or: If we have not gone through every quote, get the Quote variable in the quoteList at the count we are up to, then set the text on the quote and person textboxes to the quote data we just grabbed*

If you read through the code and the English algorithm above a few times you should be able to understand what this code is doing.

Hopefully my explanations make sense, we are now ready to send the app to your phone!

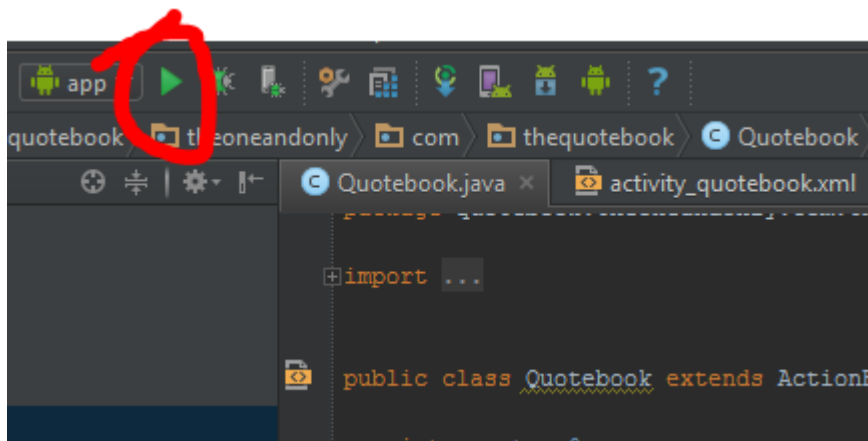


Instruction #7:

To do this next step, you have to:

- Ensure that you have your phones USB drivers properly installed.
- Enable Developer Settings: <http://www.greenbot.com/article/2457986/how-to-enable-developer-options-on-your-android-phone-or-tablet.html>
- Enable USB Debugging in the developer options.
- Have your phone plugged in and accept the popup that checks if you would like to connect to your computer (Android Studio/ADB)

Then, you have to click the green play button <http://i.imgur.com/cOkY2Ep.png>, the app will compile and if you have set it up correctly it should send it to your phone and open the app!



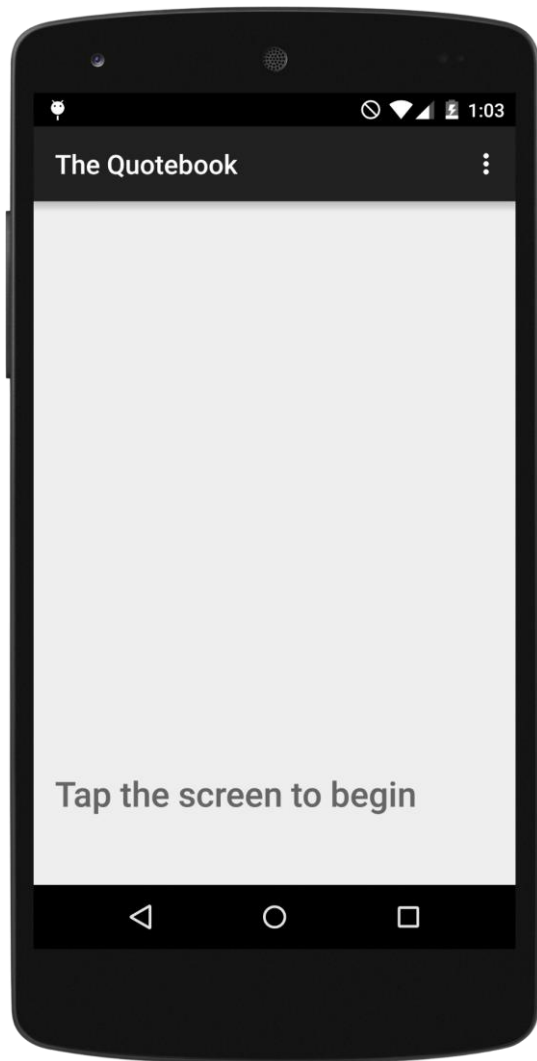
If you have issues here, Google: *your phone* + abd drivers/android studio.

Optional Instruction #8:

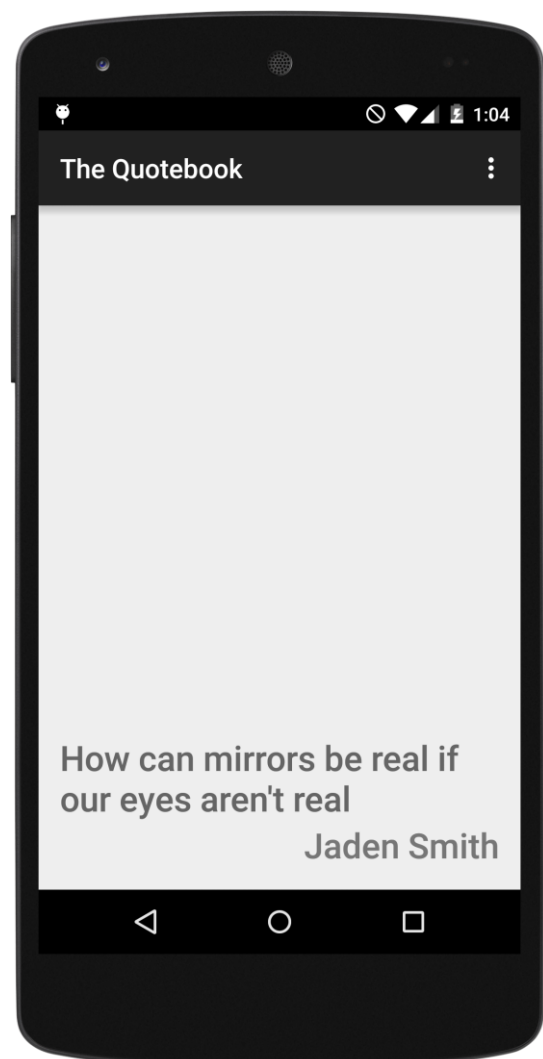
Change the quotes around, try and add more! If you have a particular interest in an area change the quotes and make a targeted app like a movie quotes app that has all your favourite quotes or lines.

Change the font, colours, formatting or use.

Email your own versions to me!



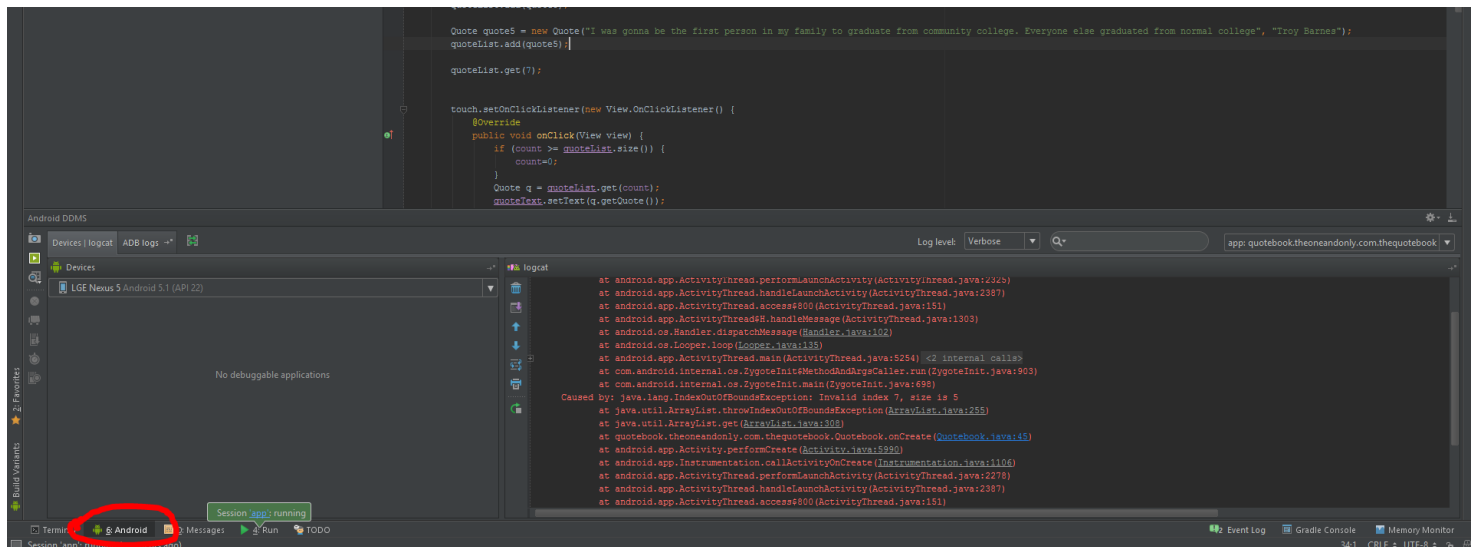
Here's what the app should look like!





Debugging

One of my biggest regrets looking back on my dive into programming was not learning how to debug when there was an error in my code, Android Studio helps a lot here.



If your phone is properly connected to the computer, a constant feed of debugging information is being sent to Android Studio, called logcat. Whenever there are errors or crashes a stacktrace will be printed in red to the feed, which you can read over and work out where your error is. If you click the blue line in the logcat error message (as you can see above) it will actually take you to the exact line of code where the crash occurred and the error message should give you a hint about what the problem may be. Often copying the first few lines of these stacktraces can be very helpful.

Libraries

Libraries are like pre-made bundles of code that you can use instead of coding everything yourself.

For example the IO Commons library contains a huge range of methods that manipulate files in one line, like `copyFile()`, `moveFile()` and `getExtension()` instead of having to do them manually.

There are specifically made Android libraries from Google that allow you to use newer Android features like the navigation drawer on older devices.

Android Arsenal is a great site for finding Android Libraries:

<https://android-arsenal.com/>

And here is how to add them to Android Studio:



<http://stackoverflow.com/questions/16588064/how-do-i-add-a-library-project-to-the-android-studio>

If you enjoyed the guide, here are some more tips:

- Stack Overflow is a fantastic community if you have any development questions - but Google it first.
- Check out www.reddit.com/r/androiddev
- Follow Google Design Guidelines.
- If you don't really understand some code or how to do a particular task, Google it, comment what you are trying to do and ask around.
- Use libraries wherever you can.

And now, every Android development related resource you will ever need...

Android Development Course

<https://www.udemy.com/android-lollipop-complete-development-course/>

A huge resources list:

https://github.com/thecodepath/android_guides/wiki/Beginning-Android-Resources

Activity Lifecycle IMPORTANT

<http://developer.android.com/reference/android/app/Activity.html>

Fragments IMPORTANT

<http://www.vogella.com/tutorials/AndroidFragments/article.html>

More Fragments

<http://developer.android.com/guide/components/fragments.html>

Dialogs

<http://stackoverflow.com/questions/2478517/how-to-display-a-yes-no-dialog-box-in-android>

Navigation Drawer

<http://developer.android.com/training/implementing-navigation/nav-drawer.html>

Toasts popups

<http://www.mkymong.com/android/android-toast-example/>



A list of design related resources

Material Design Guidelines

<http://www.google.com/design/spec/material-design/introduction.html>

Material Design Icon Downloads

<https://github.com/google/material-design-icons>

Material Design Icon Index

<https://google.github.io/material-design-icons/>

Design Inspiration

<http://androidniceties.tumblr.com/>

That's it for now, I'll be updating the document regularly so check back at xaviertobin.com for updates!

I have made this document free of charge, but any support would be more than welcome - I have a PayPal donation link at www.xaviertobin.com

Good luck and enjoy programming!