**Create a project, variable, and toast**

The first step we need to do is create a new project in Android Studio. You can skip this step by using the exercise files for this movie. To create a project, open Android Studio and select Start a new Android Studio project. Here, we need to give our project a name and define some properties of the project. I'm going to set our application name to DiceOut. Then, I'll change the company domain, known as a package, to just be example.com. Then, I'm going to set the project location to be the desktop.

Click Next. We are creating our project for phones and tablets, so I'll keep the first option checked, and we will set this to API level 15, which is for Android 4.0.3, known is Ice Cream Sandwich or ICS. Click Next. Next, I need to define the activity for the app. This'll be the initial layout for the app we will use. Select Basic Activity. This'll add a menu and a button to the lower right corner of the app. Click Next.

I'll keep the defaults for the first two items here. Activity Name is the name that we will use for the Java file. Layout Name is the name of the XML file that will define the widgets of our activity. I'm going to change our Title, however. Let's change that to DiceOut so it'll appear as DiceOut on the launcher as well as in the menu bar for the app. I'll keep the Menu Resource Name the same. We won't use the menu in this project, but you can modify or add it on your own when we finish the project. Click Finish. The project will then be generated and assembled.

We will see Android Studio do an initial build of the project, and a message will appear at the bottom saying that the build finished. In Android Studio, we will see a few key areas. The left shows the project and the folder structure of the project. All the files we will use in our project, including Java coding files, XML layout files, and any assets that add to our project like images, we will manage on the left. When we are working with activity layouts, the right side will be split into two sections.

At the top, you'll see that we have a content\_main.xml tab open. The XML code for the layout appears on the left if we go into text view. A visual preview of the app appears on the right. The activity already has a single text view widget that is created for us. If you click it in the preview window, the IDE will highlight the corresponding code in XML. Next, we can open up the Java code for our project. Click the MainActivity.java tab at the top. Since we're working with Java, the layout preview disappears.

We are now within the Java code for our activity. You'll see a few methods already created for us. The first one is called onCreate. This'll execute when the activity is created within the app when it runs. We will use this to set values of variables and create links between fields we create in Java to widgets laid out in XML. After this are two additional methods: onCreateOptionsMenu and onOptionsItemSelected. We aren't going to use these methods, but we need to keep them here for the program to work. They're related to the menu and action bar of our app.

When you're finished with the project, you can modify and extend these with methods of your own. Let's create our first variable, or field as it's called in Java. We'll need a field to hold the score of our game. At the top, after our class definition, but before the onCreate method, we need to create an integer to hold our score. We'll start off with a comment. Field to hold the score. Then, we'll create an integer called score. We don't need to set an initial value right now.

We'll do that in the onCreate method. So let's do that now. At the bottom of the method, add some blank lines and set the score to zero when the activity is created. Add a comment, set initial score. We'll set the score to be equal to zero. The score field is now created and has been set with a initial value of zero. Let's also add a quick message to confirm that everything is running correctly. We can use a feature to send a brief message to the display, called a toast, and we can put a message inside of it.

Start by accessing the Toast class, and we'll use the makeText method. Create greeting. Toast.makeText. Inside, we need to provide the context for the message, which we can use the getApplicationContext method for. Next, we can define the length, which we will use the constant LENGTH\_SHORT, which is part of the Toast class. Then, we append the show method to the end that will display the toast message. You'll see you'll get a notification that says android.widget.Toast with a question mark.

This is because the Toast class is not part of the default application package. We need to add this with import statements. Android Studio provides a shortcut by using the Alt and Enter keys. If you press both of those, it'll add this to the import statements at the top. I pressed those two keys, so let's go ahead and go to the top. If we expand the import section, you'll see that it's now added. That's about it for now, so let's test the app and see how it works. Open the Run menu and select Run 'app'.

The deployment target will display, and we will use the virtual device that we created when we installed for Android Studio. We're going to use the Nexus 5. Go ahead and select that and click OK. The Android emulator will then boot and then launch Android. The app we built will be sent to the emulator to run. You'll see the default layout of our app, and then you'll see the toast message appear at the bottom and then fade away. That's it. We've taken the first step in building our app.



































