

Android Activities and Developing Basic Android App

Hui Chen ^a

^aCUNY Brooklyn College, Brooklyn, NY, USA

February 15, 2022

Outline

- 1 Tools of the Trade
- 2 Android
- 3 Developing Basic Android App
- 4 Preparing Development Environment
- 5 Android Activities

Outline

- 1 Tools of the Trade
- 2 Android
- 3 Developing Basic Android App
- 4 Preparing Development Environment
- 5 Android Activities

Technical Readiness Preparation

Provide a technical readiness for

- ▶ Version Control Systems
- ▶ Develop simple Android Apps with Android Activities and Android Intent
- ▶ Test Android Apps

Outline

- 1 Tools of the Trade
- 2 Android**
- 3 Developing Basic Android App
- 4 Preparing Development Environment
- 5 Android Activities

Lecture Module Outline

- ▶ An introduction to Android
- ▶ The Android OS
- ▶ The IDE and build system
- ▶ Basic App Development
 - ▶ Understand Android Studio projects
 - ▶ Create basic Android app

Note: this is not an Android class!

Outline

- 1 Tools of the Trade
- 2 Android
- 3 Developing Basic Android App**
- 4 Preparing Development Environment
- 5 Android Activities

Android Basics

Recall what we discussed,

- ▶ What is Android – OS, libraries, utility programs
- ▶ What goes into an Android App – Java/Kotlin + XML definitions + some other stuff
- ▶ Create an Android Studio project – (gradle; manifest, resources, src)

What to do today? Learn the following by creating a basic app

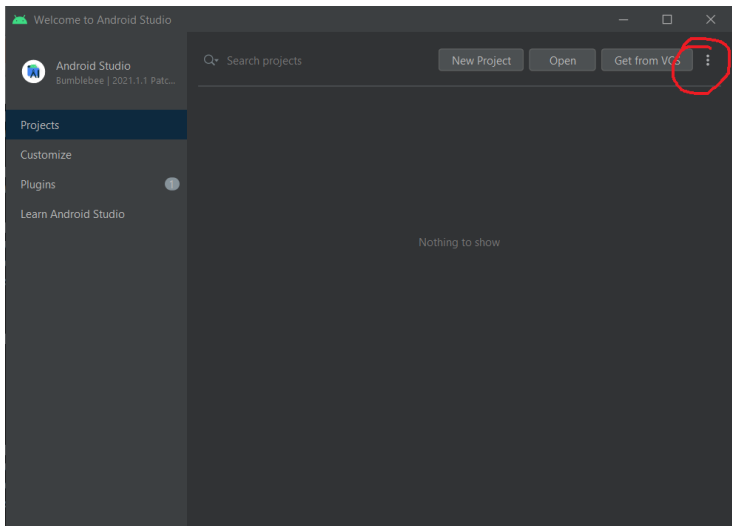
- ▶ Putting together a basic Android App
 - ▶ Activities
 - ▶ Widgets
- ▶ Getting a closer look at SDK
- ▶ Creating an AVD and running app in the AVD

Outline

- 1 Tools of the Trade
- 2 Android
- 3 Developing Basic Android App
- 4 Preparing Development Environment**
- 5 Android Activities

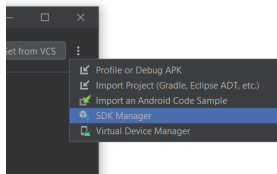
Preparing Development Environment

Take a closer look at Android Studio Start Screen,



Android SDK

Ensure that the desired release/version of Android SDK is installed

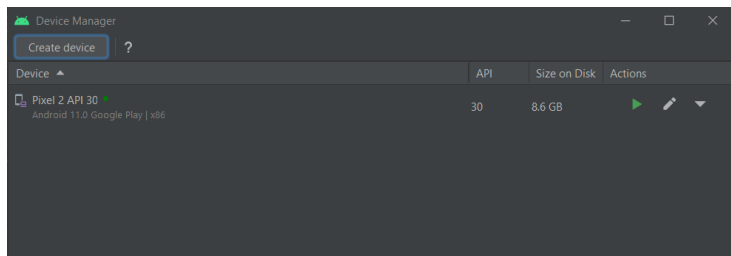
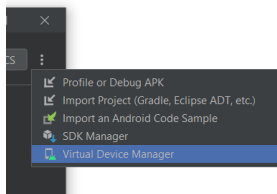


 A screenshot of the 'Settings' dialog box, specifically the 'Android SDK' section. The 'Android SDK Location' is set to 'C:\Users\hui\AppData\Local\Android\Sdk'. The 'SDK Platforms' tab is selected, showing a list of SDK platforms. The 'Android API 32' platform is checked and highlighted with a red arrow. The 'Status' column shows that 'Android API 32' is 'Installed', while all other platforms are 'Not installed'.

Name	API Level	Revision	Status
<input type="checkbox"/> Android Tiramisu Preview	Tiramisu	1	Not installed
<input checked="" type="checkbox"/> Android API 32	32	1	<u>Installed</u>
<input type="checkbox"/> Android 12.0 (S)	31	1	Not installed
<input type="checkbox"/> Android 11.0 (R)	30	3	Not installed
<input type="checkbox"/> Android 10.0 (Q)	29	5	Not installed
<input type="checkbox"/> Android 9.0 (Pie)	28	6	Not installed
<input type="checkbox"/> Android 8.1 (Oreo)	27	3	Not installed
<input type="checkbox"/> Android 8.0 (Oreo)	26	2	Not installed
<input type="checkbox"/> Android 7.1.1 (Nougat)	25	3	Not installed

Android AVD

Ensure that the desired Android Virtual Device (AVD) supporting the the release/version of SDK is set up.



Tips ...

- ▶ Don't shut off or restart AVD unless you have to – AVD runs slowly.
- ▶ Android Studio, Android SDK, and AVDs are in different directors, it can useful to know where they are, such as, using `adb` to list devices.

Outline

- 1 Tools of the Trade
- 2 Android
- 3 Developing Basic Android App
- 4 Preparing Development Environment
- 5 Android Activities**

Android Apps Lifecycle

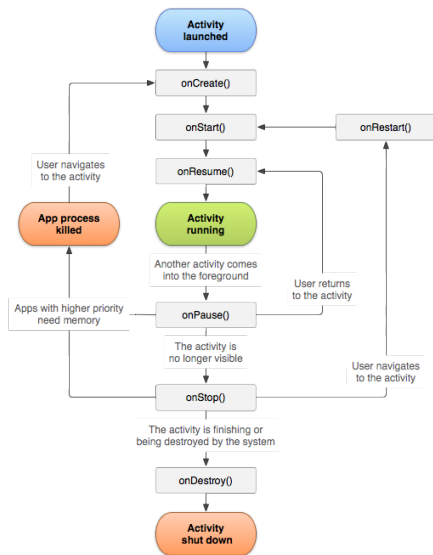
- ▶ Run in a separate OS process
- ▶ Apps don't typically close
- ▶ Can be killed by the Android OS if resources are needed
- ▶ Applications that aren't in use are killed
- ▶ Should design your Android apps with this knowledge in mind

Activities and Intents

Creating basic apps using Activities and Intents

- ▶ Activity roughly corresponds to a screen in an app
 - ▶ A typical app has many of these
 - ▶ Reusable screen parts are encapsulated in a Fragment
- ▶ Intents allow activities to communicate with each other
 - ▶ Including passing data to one another

Android Activity Lifecycle



Creating Activity Class

To develop a basic Android apps, we begin with creating our own Activity class by extending an [Android Activity](#) class.

```
1 public class MainActivity extends Activity {
2     // ...
3 }
```

or

```
1 public class MainActivity extends AppCompatActivity {
2     // ...
3 }
```

and then implementing some of the lifecycle methods, e.g.,

```
1     @Override
2     protected void onCreate(Bundle savedInstanceState) {
3         super.onCreate(savedInstanceState);
4         //add some initialization code here
5         // ...
6     }
```

Example onCreate() Method

```
1 @Override // what is this?
2 protected void onCreate(Bundle savedInstanceState)
3     // 1. Reinitialize the Activity with the saved data if
4     //    available
5     super.onCreate(savedInstanceState);
6     // 2. inflate the activity's UI,
7     setContentView(R.layout.activity_main);
8
9     // 3. Getting UI object
10    mDateTimeTextView = findViewById(R.id.dateTimeTextView);
11    final Button addTaskBtn = findViewById(R.id.addTaskBtn);
12    final ListView listView = findViewById(R.id.taskListview);
13    mList = new ArrayList<String>();
14
15    // 4. Setting up listener for clicking on ListView item
16    listView.setOnItemClickListener(new AdapterView.
17        onItemClickListener() {
18        @Override
19        public void onItemClick(AdapterView<?> adapterView, View
20            view, int i, long l) {
21            // 5. Not implemented for now
22        }})
```

Elements in onCreate() Example

- ▶ what is savedInstanceState?
- ▶ what does findViewById do?
- ▶ What is this new AdapterView.OnItemClickListener() ...?
- ▶ What to do with the UI widgets obtained by calling findViewById?
- ▶ What are R.id....?

Connecting UI and Activity

MainActivity.java → activity_main.xml

```
1 <!-- ....-->
2 <Button
3     android:id = "@+id/addTaskBtn",
4     android:layout_height="wrap_content",
5     android:layout_width="wrap_content",
6     android:layout_below="@+id/dateTimeTextView",
7     android:layout_centerHorizontal="true",
8     android:padding="20dp",
9     android:text="@string/add_task",
10    android:onClick="addTaskClicked" />
11 <ListView
12     android:id="@+id/taskListview",
13     android:layout_width="wrap_content",
14     android:layout_height="wrap_content",
15     android:layout_below="@+id/addTaskBtn"/>
16 <!-- ....-->
```

Let's play with several UI widgets and related methods

- ▶ Layouts of UI widgets, e.g., `ConstraintLayout`
- ▶ UI controls, e.g., `Button`, `EditText`, `TextView`
- ▶ UI widget callback methods, e.g., `onClick`
- ▶ Activity methods, e.g., `findViewById`

Event-Driven Programming

The platform puts events in an event queue and runs an event loop (in pseudo code)

```
1  do {
2      e = getNextEvent()
3      processEvent(e)
4  } while (e != EXIT_EVENT)
5
6
7  processEvent(e) {
8      for handler in e.handler_list {
9          handler.invoke()
10     }
11 }
12
```

- ▶ This event loop often implemented by the platform.
- ▶ Users write event handler routines, register them with the platform, and use the handler routines to process events

Event-Driven versus Algorithm-Driven

Application-driven or algorithm-driven programs

- ▶ A program expects inputs in a pre-determined order and timing

Event-driven programming – a type of reactive programming

- ▶ Program waits for input events when it loads
- ▶ The programs runs particular code to response to an event
- ▶ The overall flow of the execution is determined by the events that occur
- ▶ The overall flow of what code is executed is determined by events in non-deterministic order and timing

GUI Event-Driven Programming

GUI programming are typically event-driven – applies to Android

- ▶ GUI Event – An object that represents a user's interaction with a GUI components (e.g., a button, a menu item)
- ▶ Event Listener – An object that waits for events and responds to them.
- ▶ Event Handler – An object that calls by the Event Listener to handle an event as a part of the response

GUI Event Handling

- ▶ Programmer attaches a listener to a component for an event (e.g., a button, a menu)
- ▶ Platform notifies the listener when the event occur (e.g., a button click)
- ▶ The listener calls the Event Handler's methods as a part of reponse

Questions?

- ▶ An introduction to Android
- ▶ The Android OS
- ▶ The IDE and build system
- ▶ Basic App Development
 - ▶ Understand Android Studio projects
 - ▶ Create basic Android app
 - ▶ **Model-View-Controller**
 - ▶ **Event-driven Programming**