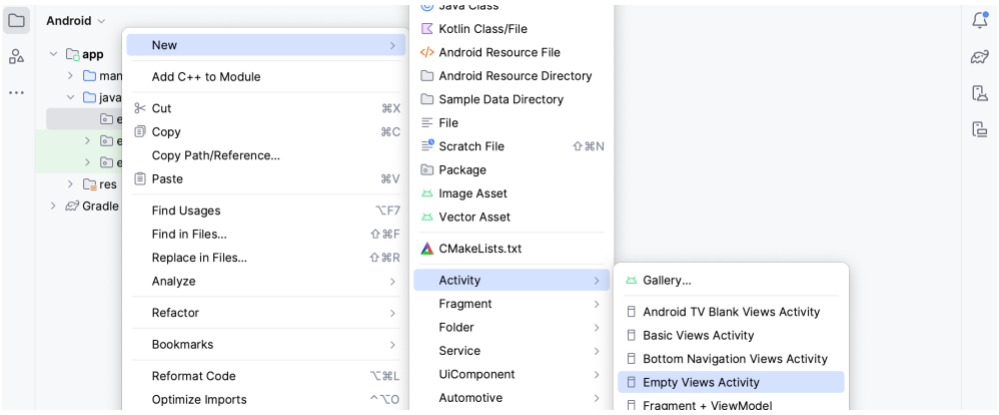
CS 522—Mobile Systems and Applications—Summer 2024
Assignment One—My First App

This assignment involves building and deploying a simple Hello World app. This exercise involves:

- Building an Android app with a user interface (activity).
- Accepting user input in the app.
- Displaying the user input

The Android developer website provides a wealth of information about Android, including instructions for building and running your first Android app[1]. However, you should ignore the Getting Started tutorial, which assumes you are programming in Kotlin (We will be developing in Java) and immediately launches you into using the Jetpack Compose library (We will start from the ground up, learning the basic UI components). You should also avoid using the Android Studio wizard to create activities, since the wizard introduces several complications. Eventually you should be able to use the wizard as an aid, not as a crutch.

Create an empty project (with no activities initially), with package name `edu.stevens.cs522.hello`, and then add an empty activity (`New | Activity | Empty Views Activity`).



The target platform for the app will have been configured to be Android API Level 34 (UpsideDownCake).

In the application manifest, the <application> element will specify the theme that styles the application:

```
<application
    ...
    android:label="@string/app_name"
    android:theme="@style/AppTheme">
    ...
</application>
```

[1] http://developer.android.com/training/basics/firstapp.

The theme itself should be specified in a file `res/values/themes.xml`:

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme"
           parent="android:Theme.Material.Light.DarkActionBar">
    <!-- Customize your theme here. -->
    </style>
</resources>
```

Do not leave the theme as the Google material design theme that the wizard in Android Studio prefers. We will be using that theme later, when we incorporate elements of material design into the app.

You will need to create an `<activity>` element as a child of the `<application>` element in the manifest, and this will need to specify the right action (to make this the activity launched for the app) and category (to make sure it shows on the app launcher):

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

The activity itself displays a layout that includes a prompt, a text box for user input, and a button for signaling when user input is ready. This should be defined as a layout resource in `res/layout/activity_main.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:text="@string/prompt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <EditText
        android:id="@+id/textbox"
        android:hint="@string/hint"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/button"
        android:text="@string/enter"
        android:layout_gravity="center"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"/>

</LinearLayout>
```

The prompt and hint strings are defined as string resources in res/values/strings.xml:

```
<string name="prompt">What is your name?</string>
<string name="hint">Enter your name here.</string>
<string name="enter">ENTER</string>
```

You should also define a name for the app that includes your own name as a string resource, so that it is displayed at the top of the app screen (see the application element above):

```
<string name="app_name">XYZ\'s First App</string>
```

Create a class for the main activity of this app, whose `onCreate` method inflates this layout on the device screen:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```

However, this app will not react to user input. You will need to add a callback for button presses (Don't forget to put a text label on the button in the view). You can make the activity itself be this callback object:

```
public class MainActivity extends Activity implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        button = (Button) findViewById(R.id.button);
        button.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        ...
    }
}
```

On receiving the button press, the callback should get the input text from the text box, and launch another activity using an explicit intent:

```
EditText textbox = (EditText) findViewById(R.id.textbox);
String text = textbox.getText().toString();
```

```
Intent intent = new Intent(this, ShowActivity.class);
intent.putExtra(ShowActivity.MESSAGE_KEY, text);
startActivity(intent);
```

The child activity displays this text on its screen:

```
public class ShowActivity extends Activity {

    public final static String MESSAGE_KEY = "message";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_show);
        String message = getIntent().getStringExtra(MESSAGE_KEY);
        ...
    }
}
```

For the layout for this child activity, make it a linear layout with a single text view:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/message"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

Then programmatically set the string to be displayed in the activity:

```
TextView textview = (TextView) findViewById(R.id.message);
textview.setText(message);
```

You can specify a response message as a string resource with a parameter:

```
<string name="show_name">Hello, %s!</string>
```

Then instantiate this parameter with the name you want to display:

```
String text = getIntent().getStringExtra(MESSAGE_KEY);
String message = getResources().getString(R.string.show_name, text);
textview.setText(message);
```

You should run the app on a virtual Android device of your choice (e.g. a Pixel 4), or on your own physical device if you prefer and if it supports the API. You can watch the processing of

your application (insert log statements in your code for more information) in the logcat window in Android Studio.

Once you have your code working, please follow these instructions for submitting your assignment:

1. Create a zip archive file, named after you, containing a directory with your name. E.g. if your name is Humphrey Bogart, then name the directory Humphrey_Bogart.
2. In that directory you should provide the Android Studio project for your Android app.
3. Also include in the directory a completed rubric where you self-evaluate your submission.
4. In addition, record a short mpeg (MP4) video of a demonstration of your deployment working. Make sure that your name appears in the video. It should appear in the title bar of the app while it is running. *Do not provide private information such as your email or cwid in the video*.

Your solution should be uploaded via the Canvas classroom. Your solution should consist of a **zip archive** with one folder, identified by your name. Within that folder, you should have one Android project, for the app you have built. You should also provide a completed rubric for your solution, as well as a video showing the app working.