CSCI3310 Mobile Computing & Application Development 2021/22 Term 2

Department of CSE, CUHK

Assignment 1 (First Android App)

Fab Passcode

Due : Feb 7, 2022 11:59pm

Introduction

In this assignment, you learn how to create and run your first Android app, **Fab Passcode**, on an emulator and/or on a physical device. You shall need a PC running Windows or Linux, or a Mac running macOS. See the Android Studio download page for up-to-date system requirements for self-installation preparation. **Android Studio Arctic Fox 2020.3.1** is assumed, please also try to implement the program on **Android 10.0 (API 29)**. The default testing simulator is **Pixel 2** Portrait mode.

Objectives

- To create an Android project from a template: the development process for building Android apps.
- To implement basic GUI functions of a simple Android app.

Setting up the Android Project

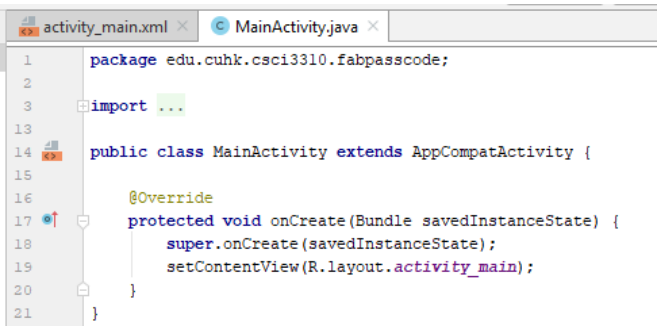
1. Install the Android Studio development environment, and create an emulator (virtual device) to run your app on your computer (Windows, macOS or Linux).

2. 1) Start Android Studio and create a new.**Fab Passcode** project.

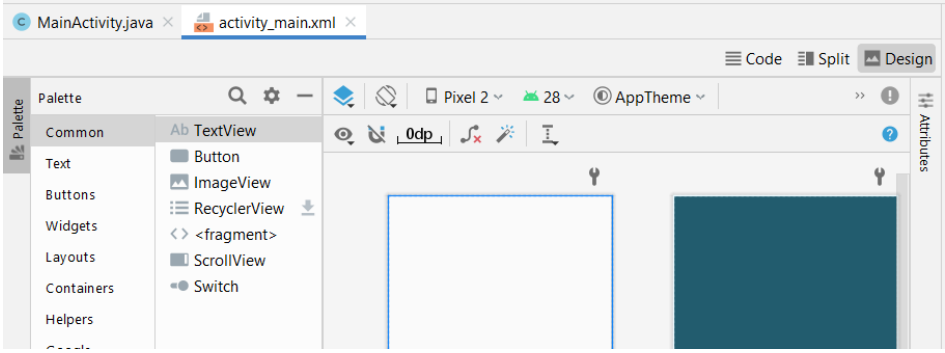
Attribute	**Value**
Application Name	**Fab Passcode**
Company/Domain Name	**edu.cuhk.csci3310**
Phone and Tablet Minimum SDK	**API28: Android 10.0 Q**
Template	**Empty Activity**

3. Explore the project layout in the newly launched Android Studio editor. Follow these steps to

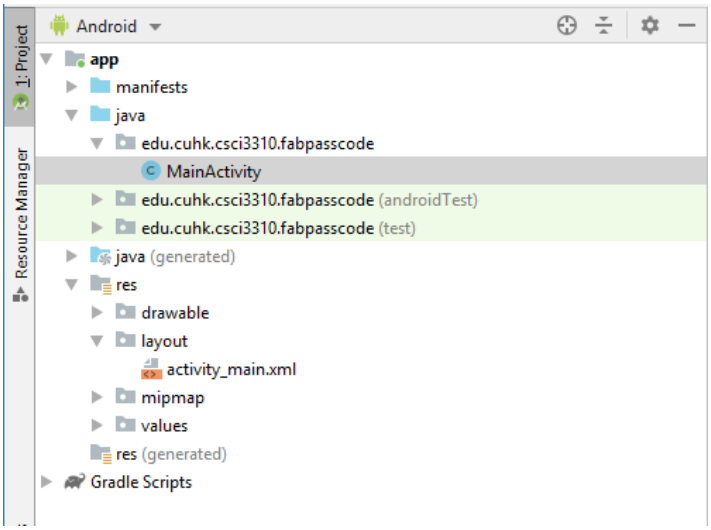
   3.1. Click the **MainActivity.java** tab, if not selected, to see the code editor as shown below.



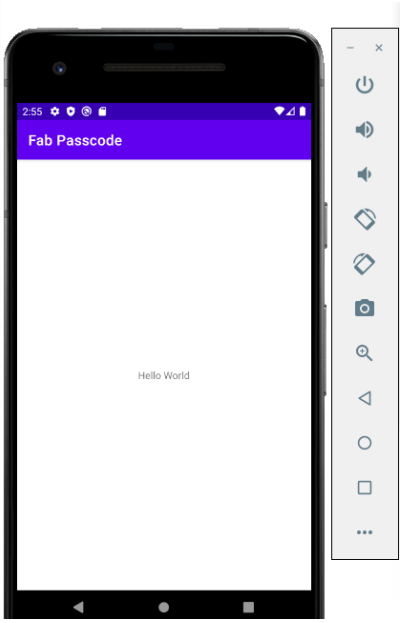
   3.2. Click the **activity_main.xml** tab next to the default **MainActivity.java** tab to see the layout editor. Click the layout editor **Design** tab, if not already selected, to show a graphical rendition of the layout as shown below.



4. Explore the **Project** > **Android** pane > **app** and **res** folders
   4.1. If not already selected, click the **Project** tab in the vertical tab column on the left side of the Android Studio window. The Project pane appears. To view the project in the standard Android project hierarchy, choose **Android** from the popup menu at the top of the Project pane.
   4.2. Expand the **app** folder, the **java** folder, and the **edu.cuhk.csci3310.fabpasscode** folder to see the **MainActivity.java** file, and then expand the **res** folder and the **layout** folder to see the **activity_main.xml** file, as shown below.



5. Run the **Fab Passcode** app on the virtual or physical devices (optional).
   5.1. In the top menu, choose **Run > Run app** or click the **Run** icon  in the toolbar.
   5.2. The **Select Deployment Target** window, under **Available Virtual Devices**, select any created virtual device, or [create a new Android virtual device (AVD)] and click **OK**.



Core Features Implementation (Step-by-Step)

Having done the basic setup, this assignment needs you to write a simple **Android** App which can show a passcode, which is placed on top of the app, entered by the user via number buttons. An unlock button is placed at the bottom of the app for verifying the correctness of the inputted passcode. If the entered passcode **contains** the **last four digit** of your own Student ID number (consecutively and in order), display a hidden image and that is!

1. Implement the Passcode entry function
   1.1. Customizing the `TextView` in the layout. In the **Text** tab of **activity_main.xml**, update the `TextView` to remove the bottom constraint, add a hint text, etc:

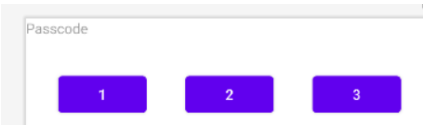
```
<TextView
    android:id="@+id/passcodeView"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Passcode"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

   1.2. Add `Buttons` '1', '2' and '3' to the layout by adding the following extra lines:

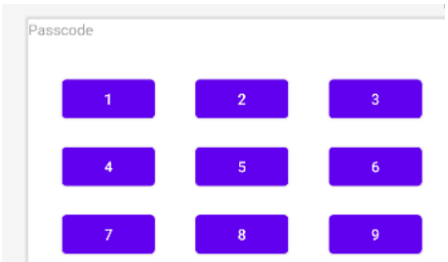
```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="32dp"
    android:layout_marginTop="16dp"
    android:text="1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/passcodeView" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="2"
    app:layout_constraintEnd_toStartOf="@id/button3"
    app:layout_constraintStart_toEndOf="@id/button1"
    app:layout_constraintTop_toBottomOf="@+id/passcodeView" />

<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:text="3"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@id/button2"
    app:layout_constraintTop_toBottomOf="@+id/passcodeView" />
```

   Alternatively, button (or other Views) can be added via "drag-and-drop" actions using the layout editor; this shall be illustrated in our lab.

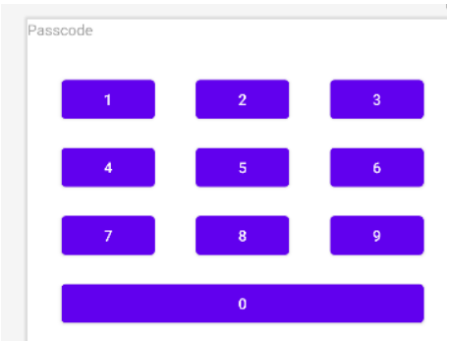
   1.3. Click on the **Design** tab of the **activity_main.xml**, the graphical representation should be updated as:



   1.4. Add `Buttons` for '4', '5', '6', '7', '8', '9' similarly within the same ConstraintLayout:



1.5. Add `Buttons` for '0' similarily to make:



1.6. Add onClick handlers for the buttons.

- Next, we shall add a Java method for each `Button` in **MainActivity** that executes when the user taps the `Button`. A *click handler* is a method that is invoked when the user clicks or taps on a clickable UI element. In Android Studio you can specify the name of the method in the `onClick` field in the **Design** tab's **Attributes** pane. You can also specify the name of the handler method in the XML editor by adding the `android:onClick` property to the `Button`.
- With the XML editor open (the **Text** tab), find the `Button` with the `android:id` set to `button1`. Add the `android:onClick` attribute to before the `android:text` attribute:

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="32dp"
    android:layout_marginTop="16dp"
    android:onClick="updatePasscode"
    android:text="1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/passcodeView" />
```

- Click the red bulb icon that appears next to the attribute. Select **Create click handler**, choose **MainActivity**, and click **OK**. If the red bulb icon doesn't appear, click the method name ("updatePasscode"). Press **Alt-Enter** (**Option-Enter** on the Mac), select **Create 'updatePasscode(view)' in MainActivity**, and click **OK**.

  This action creates a placeholder method stub for the `updatePasscode()` method in **MainActivity**, as shown below:

```
import ...

public class MainActivity extends AppCompatActivity {

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

    public void updatePasscode(View view) {
    }
}
```

1.7. Update passcode `TextView` on clicking the number button

- Add new private member variables for storing the entered passcode and a reference of the `passcodeView`, which you will add to the click handler:

```
public class MainActivity extends AppCompatActivity {
    private int mPasscode;
    private TextView mShowPasscode;
```

Please import the corresponding package as suggested by Android Studio.

- Now that you have `mShowPasscode`, you can get a reference to the `TextView` using the ID you set in the XML layout file. In order to get this reference only once, specify it in the `onCreate()` method:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mShowPasscode = (TextView) findViewById(R.id.passcodeView);
}
```

- Finally, we can update and display the passcode on the `TextView` by implementing the click handler as follows:

```
public void updatePasscode(View view) {
    Button button = (Button) view;
    String buttonText = button.getText().toString();
    mShowPasscode.setText(buttonText);
}
```

Run the code again and see how the passcode is updated when button 1 is clicked. Modify the code to append a digit to the end.

- Repeat the steps for other buttons for adding the click handler so that passcode is updated (with a new digit appended to the end).

2. Check Passcode and show a hidden image

2.1. Add an unlock `Button` to the bottom of the layout (by removing its top constraint) which spans the width of the app.

2.2. Add an `ImageView` and hide it initially

- One image (hidden_bird.png) is provided for this assignment, which you can download from Blackboard.
- To copy the image to your project, first, close the project.
- Copy the image file into your project's **drawable** folder. Find the **drawable** folder in a project by using this path: *project_name* **> app > src > main > res > drawable**.
- Reopen your project, open **activity_main.xml** file, and click the **Design** tab.
- Drag an `ImageView` to the layout, choose the **hidden_bird** image for it, and constrain it to the top of the **Unlock** `Button` and to the bottom of number `Button`s with a margin of **16** (16dp) for all top/bottom/left/right constraints.
- In the **Attributes** pane, change the **layout_width** and **layout_height** in the inspector pane to **match constraint**, and enter the following values for the attributes:

Attribute field	Enter the following:
ID	hidden_bird
contentDescription	Hidden Bird
visibility	invisible

2.3. You should obtain a layout similar to the right figure.

2.4. On clicking the **Unlock** `Button`, implement a feature that checks whether the currently entered passcode contains the last 4 digits of your SID, say **7654**; passcode 1**7654** or **7654**20 are "good" passcode, but not **76154** nor **4567**. If no, toast a message "*Incorrect Passcode*" and reset the passcode for re-entry. Or otherwise, toast "*Bingo!*" instead and then show the **hidden_bird** `ImageView` and disable all buttons from being clicked. You may assume the entered passcode is of 1 to 6 digits.

2.5. You should disable all button clicks at the end.

2.6. Congrats! You've completed your first Android assignment.

Submission

You should pack all your app folders and related files into an archive named "3310_asg1.zip", in Android Studio Artic Fox, select **File** - > **Manage IDE Settings** - > **Export to Zip file**, and submitted it into our assignment collection slot in the Blackboard system before the deadline, Feb 7, 2022, 11:59 pm.

Late submissions will risk a score deduction of range between 10% to 50% if they are being done within 48 hours after the deadline. Submission later than Feb 9, 2022, 11:59 pm won't be considered.

Grading Remarks:

1. Follow the project/package naming stated in the specification.
2. Put down personal information (Name and SID) in the launcher Java (.java) code.
3. The code should be easy to read and contain comments to indicate computational logic.
4. The submitted code should be free of any typing mistakes, compilation errors/warnings.
5. The submitted app should be runnable at least on the virtual device stated in spec.
6. The program has to pass the corresponding test steps stated in the specification. In general, if you've followed the instructions above, you'll earn the vast majority of the points below.
   - **(20%) Basic** – does the project named correctly with personal particulars and properly styled in code?
   - **(20%) Running** – does the app compile and run properly?
   - **(30%) Layout**– does views correctly contained in the layout?
   - **(30%) Program Logic** – is passcode checking correct? are views updated correctly via clicks?

References

- Build a Responsive UI with ConstraintLayout| Android Developers
  https://developer.android.com/training/constraint-layout
- Understand the Activity Lifecycle – OnCreate | Android Developers
  https://developer.android.com/guide/components/activities/activity-lifecycle#oncreate
- Input events overview | Android Developers
  https://developer.android.com/guide/topics/ui/ui-events