Department of Electronic Engineering
City University of Hong Kong

EE5415 Mobile Applications Design and Development

Lab 02: BMI Calculator App (Java)

Certain steps in the lab section must be checked by a lab instructor. Students are required to demonstrate their implemented BMI app to obtain the marks. When you have completed one of these steps, simply raise your hand and demonstrate the step to the instructor who will then initial the **Instructor Verification** line for that step.

- 30 marks points for running your app on AVD with basic features (no Data Validation)
- 10 marks for running your app on AVD and your app with feature of Data Validation.
- 10 marks for running the BMI app with Data Validation feature on a real Android device.
- 50 marks for short questions in the verification sheet.

I. Objectives

- To go thought a complete Android app design process
- To build a simple BMI calculator app
- Run the developed app (.apk) on a physical device
- Import sample Android project to Android Studio

II. Introduction

A central feature of Android is that one application can make use of elements of other applications (provided those applications permit it). For example, if your application needs to display a scrolling list of images and another application has developed a suitable scroller and made it available to others, you can call upon that scroller to do the work, rather than develop your own. Your application doesn't incorporate the code of the other application or link to it. Rather, it simply starts up that piece of the other application when the need arises.

For this to work, the system must be able to start an application process when any part of it is needed, and instantiate the Java objects for that part. Therefore, unlike applications on most other systems, Android applications don't have a single entry point for everything in the application (no main() function, for example). Rather, they have essential components that the system can instantiate and run as needed. There are four types of components:

1. **Activity** – A component presents a visual user interface for one focused endeavor the user can undertake. For example, show a list of menu items users can choose from.

2. **Service** – A background component (no visual user interface) runs for an indefinite period of time. For example, play music in the background, or fetch data over the network as the user attends to other matters.

3. **Broadcast Receiver** - A component receives and reacts to broadcast announcements. For example, react on the change of time zone or battery level.

4. **Content Provider** – A component makes a specific set of the application's data available to other applications. For example, look up phone numbers from the system Contacts application.

Apart from "content provider", the other three types of components are activated by asynchronous messages called **Intent**, which is a description of an operation to be performed.

In this lab, students will develop a simple Android app, which is called BMI (Body Mass Index) Calculator, for go thought a complete app design process with two Activities and making use of Intent. The basic concept of the BMI will first introduce in section III and then the Android code implementation will be presented in section IV. Finally, students can run the developed app on the real device by transfer the Android package (.apk) file to the hardware.

III. BMI Calculator

The body mass index (BMI) is a statistical measure of body weight based on a person's weight and height. It is used to estimate a healthy body weight based on a person's height. The BMI is defined as

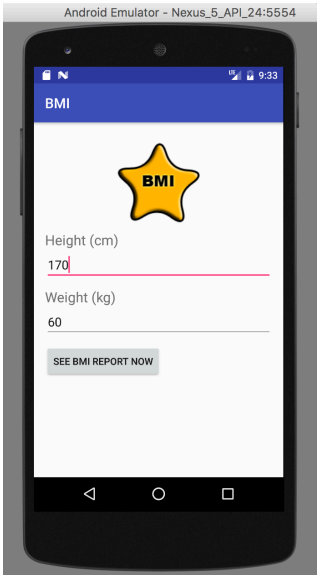
$$BMI = \frac{Mass(Kg)}{\left[Height(m)\right]^2}$$

where Mass is the body weight in Kg and Height the person's height in meter (m). Based on this index, we identify the person's health condition as follows:

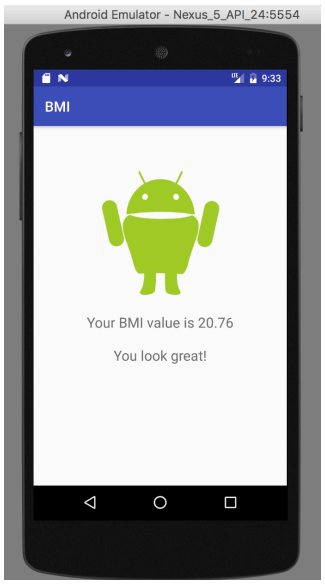
- Fat : BMI > 25
- Fit : 20 ≤ BMI ≤ 25
- Thin : BMI < 20

In this lab, students will implement a simple BMI calculator for identify the fitness of a person based the BMI formula.

Firstly, students need to design the user interfaces (UIs). For such simple app, we can just have two interfaces (Activities) in the Android app as shown below:



Front Page (Main Activity)



Report Page (2nd Activity)

The first UI is for letting user to input the Height (cm) and Weight (Kg), while the second UI is used to report the fitness based on the calculated BMI. The Front Page UI is the main activity of the app and the Report Page UI is another activity. They can be link up by intent in the Android app. In the next section, we will learn how to implement this app using Java and XML.

III. Implementation of BMI Calculator App

1. In Android Studio, create a new project named BMI with following project setting:
   - Choose your project : **Empty Activity**
   - Application Name : **BMI**
   - Package name : **com.xample.bmi**
   - Project location : use the default setting
   - Language : **Java**
   - Minimum API level: **API 19: Android 4.4 (KitKat)**
   - Click **Finish**
   - This configuration will create an Android Activity with following files
     - Activity Name : **MainActivity**
     - Layout Name : **activity_main**
     - Title : **MainActivity**

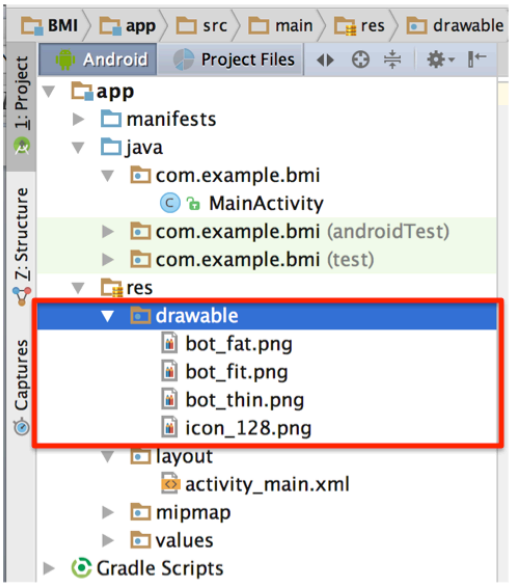
2. Download the sample image files from the course website with URL address

   - http://www.ee.cityu.edu.hk/~lmpo/ee5415/pdf/images.zip

Save this zip file in your computer and decompress it. Move or make copy of the following image files to the project's **res/drawable/** folder.

- bot_*.png
- icon_128.png

To perform this task, you can drag and drop these image files to the **res/drawable** folder in the Project Files view of the Android Studio's Project Window as shown below:



- ../app/src/main/res/drawable/
- bot_fat.png 
- bot_fit.png 
- bot_thin.png 
- icon_128.png 

3. Define string values in **res/values/strings.xml**

```
<resources>
    <string name="app_name">BMI</string>

    <string name="bmi_height">Height (cm)</string>
    <string name="bmi_weight">Weight (kg)</string>
    <string name="bmi_btn">See BMI Report Now</string>
    <string name="bmi_result">Your BMI value is</string>
    <string name="bmi_warning">Height/Weight cannot be empty.</string>
    <string name="advice_light">You need more calories!</string>
    <string name="advice_average">You look great!</string>
    <string name="advice_heavy">You should be on diet!</string>
    <string name="menu_settings">Settings</string>
</resources>
```

4. Open the **res/layout/activity_main.xml** file and insert the following:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="140dp"
        android:layout_height="140dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/icon_128" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:text="@string/bmi_height"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />

    <EditText
        android:id="@+id/heightET"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="16dp"
        android:ems="10"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="8dp"
        android:text="@string/bmi_weight"
        android:textSize="20sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/heightET" />

    <EditText
        android:id="@+id/weightET"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="16dp"
        android:ems="10"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <Button
        android:id="@+id/reportBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:text="@string/bmi_btn"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/weightET" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

This XML defines the first UI in your application. It mainly displays two text boxes and a button for collecting user's height and weight.

5. Open **`MainActivity.java`** and insert the following code for the MainActivity class:

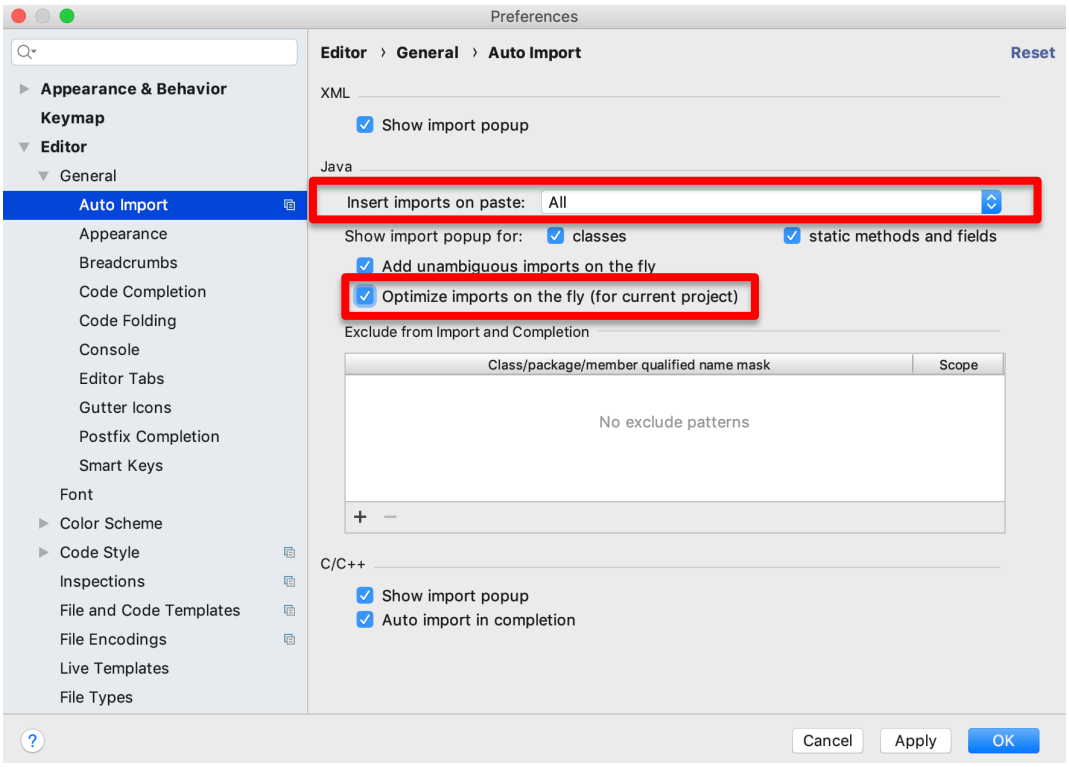
```
public class MainActivity extends AppCompatActivity {

    EditText vHeight, vWeight;
    Button submitButton;

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

        //-- get views
        vHeight = (EditText) findViewById(R.id.heightET);
        vWeight = (EditText) findViewById(R.id.weightET);
        submitButton = (Button) findViewById(R.id.reportBtn);

        submitButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String height = vHeight.getText().toString();
                String weight = vWeight.getText().toString();
                Intent intent = new Intent(getApplicationContext(), ReportActivity.class);

                Bundle bundle = new Bundle();
                bundle.putString("height", height);
                bundle.putString("weight", weight);
                intent.putExtras(bundle);
                startActivity(intent);
            }
        });
    }
}
```

6. Enable **Auto Import** of your Android Studio for removing the errors in the MainActivity.java as there are two TextView and one Button objects are added in this class file. They need to import their corresponding class library in order to support them.

   You can go to **File -> Settings -> Editor -> General -> Auto Import** and make the following changes:

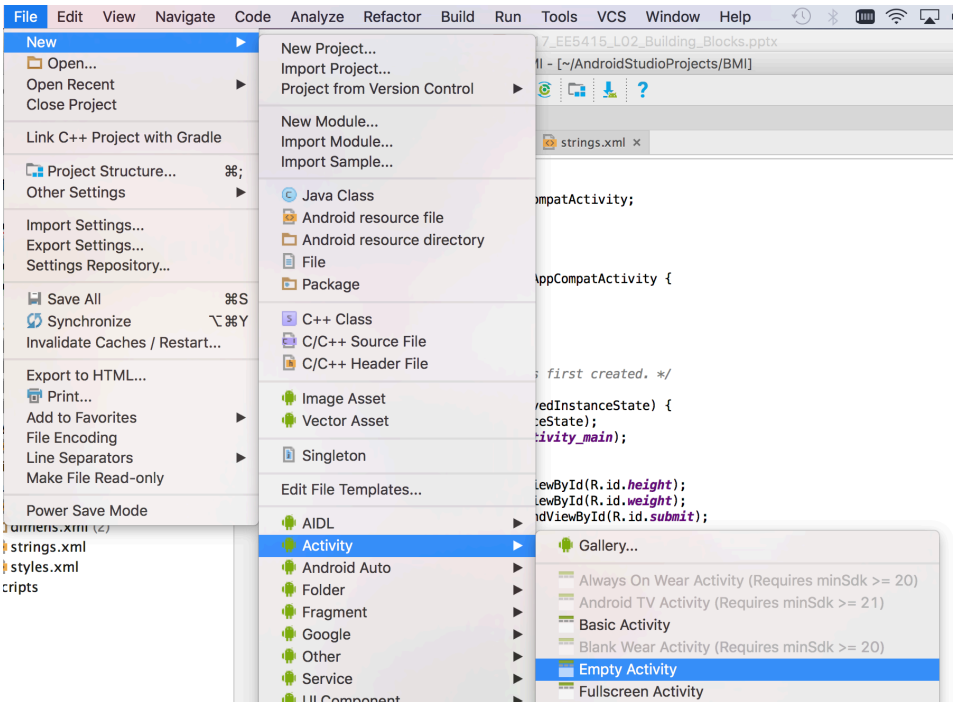
   - Insert imports on paste value : **All**
   - Add unambigious imports on the fly option : **Checked**

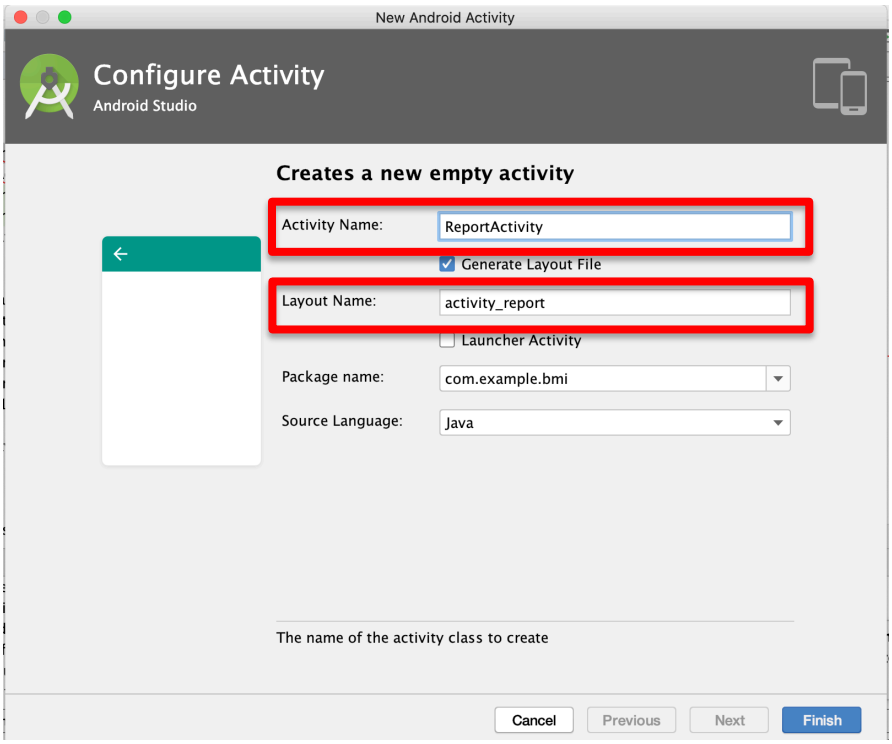
(Or, on a Mac, do the same thing in Preferences)



After this all unambiguous imports will be added automatically.

7. The `MainActivity` displays the `activity_main.xml` layout file. When the submit button is clicked, the height and weight are obtained from the text boxes and are packed as a Bundle object, which can be used to hold any number of key-value pairs. It is then passed together with the intent to invoke the `ReportActivity` class. Thus, we need to create another class named **ReportActivity.java** that extends `ActionBarActivity` under the folder of java/com/example/bim by right click this folder and select >New >Activity >**Empty Activity**





This action will also create a layout XML with filename of `activity_report.xml`.

8. Open **activity_report.xml** in `res/layout/` folder. Insert the following:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ReportActivity">

    <ImageView
        android:id="@+id/resultImage"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="24dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/bot_fit" />

    <TextView
        android:id="@+id/resultTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="24dp"
        android:text="@string/bmi_result"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/resultImage" />

    <TextView
        android:id="@+id/adviceTV"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
```

```
        android:layout_marginTop="24dp"
        android:text="@string/advice_average"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/resultTV" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

This XML is the second UI in your application. It mainly displays the calculated BMI value based on user's input and show an advice.

9. Open **ReportActivity.java** and insert the following code for the `ReportActivity` class:

```
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;
import java.text.DecimalFormat;
import androidx.appcompat.app.AppCompatActivity;

public class ReportActivity extends AppCompatActivity {

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

        Bundle bundle = getIntent().getExtras();
        double height = Double.parseDouble(bundle.getString("height"))/ 100;
        double weight = Double.parseDouble(bundle.getString("weight"));
        double bmi = weight / (height*height);

        DecimalFormat nf = new DecimalFormat("0.00");

        TextView result = (TextView) findViewById(R.id.resultTV);
        result.setText(getString(R.string.bmi_result) + " " + nf.format(bmi));

        // Give health advice
        ImageView image = (ImageView) findViewById(R.id.resultImage);
        TextView advice = (TextView) findViewById(R.id.adviceTV);

        if (bmi > 25) {
            image.setImageResource(R.drawable.bot_fat);
            advice.setText(R.string.advice_heavy);
        } else if (bmi < 20) {
            image.setImageResource(R.drawable.bot_thin);
            advice.setText(R.string.advice_light);
        } else {
            image.setImageResource(R.drawable.bot_fit);
            advice.setText(R.string.advice_average);
        }
    }
}
```

The intent used to invoke this class can be retrieved by the `getIntend()` method, and so you can obtain the bundled key-value pairs (height and weight). It then calculates the BMI values and set the corresponding symbolic image and advice. The activity displays the activity_report.xml layout file.

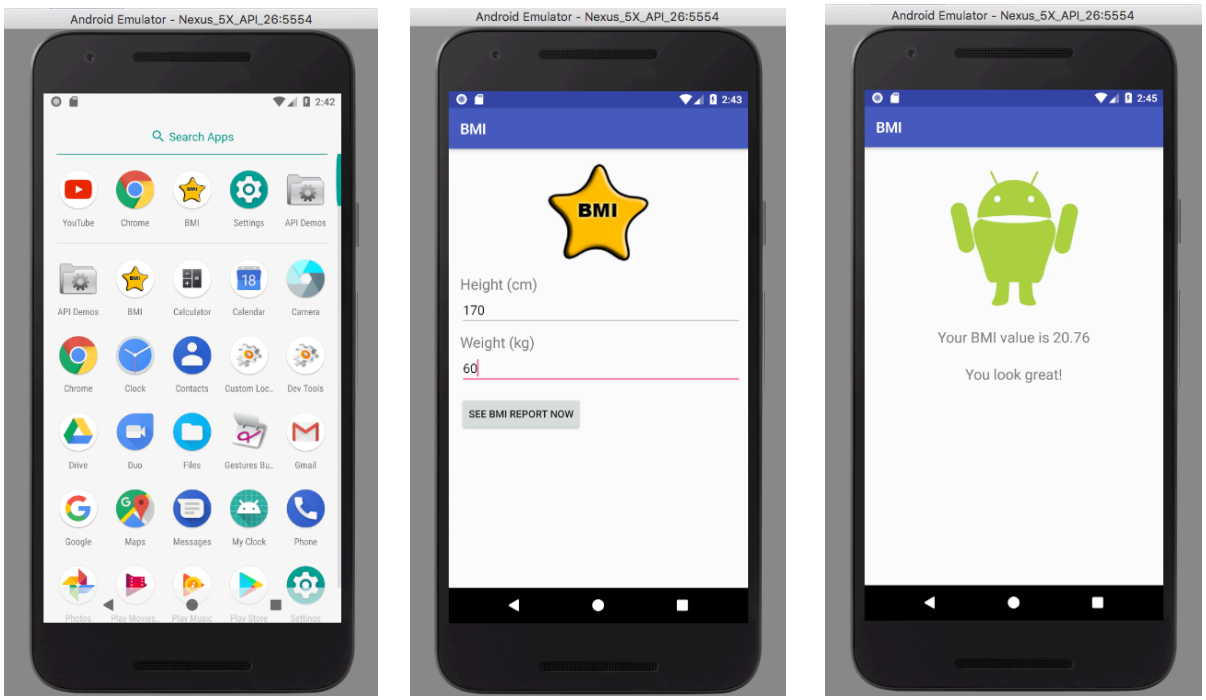
10. To use a custom icon for your application, open the **AndroidManifest.xml** file and update the drawable resource of the `<application>` element:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bmi">
```

```
    <application
        android:allowBackup="true"
        android:icon="@drawable/icon_128"
        android:label="@string/app_name"
        android:roundIcon="@drawable/icon_128"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ReportActivity"></activity>
    </application>

</manifest>
```

If you create an activity together with your project, Android Studio automatically adds this activity to your manifest. Activities that are not declared in the AndroidManifest.xml are invisible to the android system, and so cannot be instantiated. Therefore, you are required to add any newly created activities manually. In addition, the icon and roundIcon are set to "@drawable/cion_128" for using the icon_128.png image file as the app icon.

11. Run the application. It should look like this:



Check Point 1:

Running your BMI app on your AVD with basic features (no Data Validation and User Preferences)

Instructor Verification (separate page)

IV. Data Validation

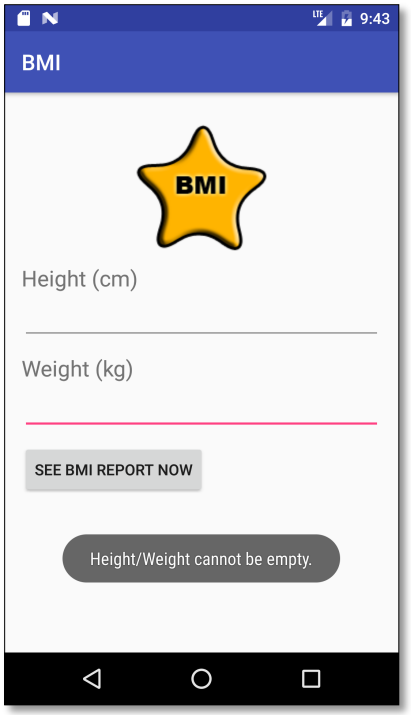
You now have a simple BMI calculator that can measure your body weight. However a user may be able to submit empty values for calculation and it will cause an exception (force close) in your program.

1. Open the **MainActivity.java** and insert the following code for the **submitButton.setOnClickListener()** method:

```
submitButton.setOnClickListener(new View.OnClickListener() {
     @Override
     public void onClick(View v) {

         String height = vHeight.getText().toString();
         String weight = vWeight.getText().toString();

         if (height.equals("") || weight.equals("")) {
             Toast.makeText(MainActivity.this, R.string.bmi_warning,
Toast.LENGTH_LONG).show();
         } else {
             Intent intent = new Intent(getApplicationContext(), ReportActivity.class);

             Bundle bundle = new Bundle();
             bundle.putString("height", height);
             bundle.putString("weight", weight);
             intent.putExtras(bundle);
             startActivity(intent);
         }
     }
});
```

The handler now will check on the inputs and use the handy Toast class to display a warning message.

2. Run the application again. It should work properly now even no input and providing Toast warning message as shown below:



V. User Preference

Being a user, you may not want to enter your height and weight every time, especially your height and weight won't change very dramatically. You can actually save user's data in a persistent storage so that you can get it back when the application is started next time.

1. Open **MainActivity.java** and insert the following two methods to the class:

```
public void savePreferences(String h, String w) {
    SharedPreferences pref = getSharedPreferences("BMI", MODE_PRIVATE);
    pref.edit().putString("height", h).apply();
    pref.edit().putString("weight", w).apply();
}

public void loadPreferences() {
    SharedPreferences pref = getSharedPreferences("BMI", MODE_PRIVATE);
    vHeight.setText(pref.getString("height", "0"));
    vWeight.setText(pref.getString("weight", "0"));
}
```

The "BMI" string is a name used to identify the save. You may use any other name as you like. The MODE_PRIVATE value is used to restrict access from any other applications of different user id (user id of linux).

2. Insert this code at the end of the **submitButton.setOnClickListener()** method such that it will save the inputs when the button is clicked:

```
  savePreferences(height, weight);
```

3. Add the `onStart()` method with in the `MainActivity` class as shown below that calling the `loadPreferences()` method with the app start again every time.

```
@Override
protected void onStart() {
    super.onStart();
    loadPreferences(); }
```

4. Run the application again. Now your application will load the height and weight when it starts and save them when the submit button is clicked.

Check Point 2:

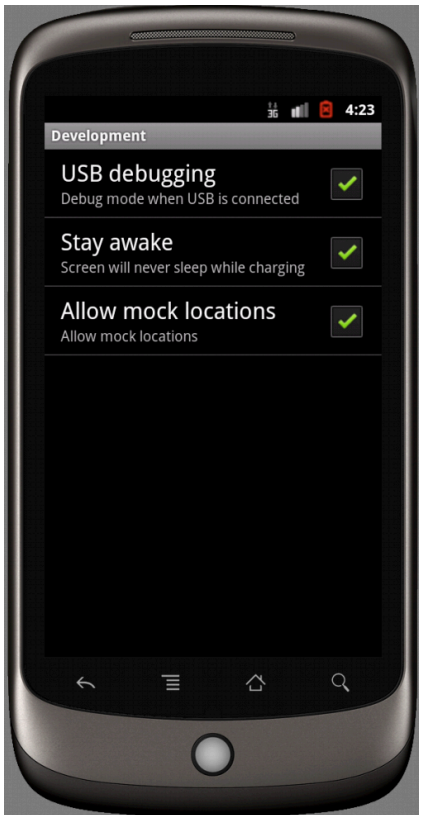
Running your BMI app with features of Data Validation and User Preference on your AVD.
Instructor Verification (separate page)

VI. Run Your App on a Physical Device

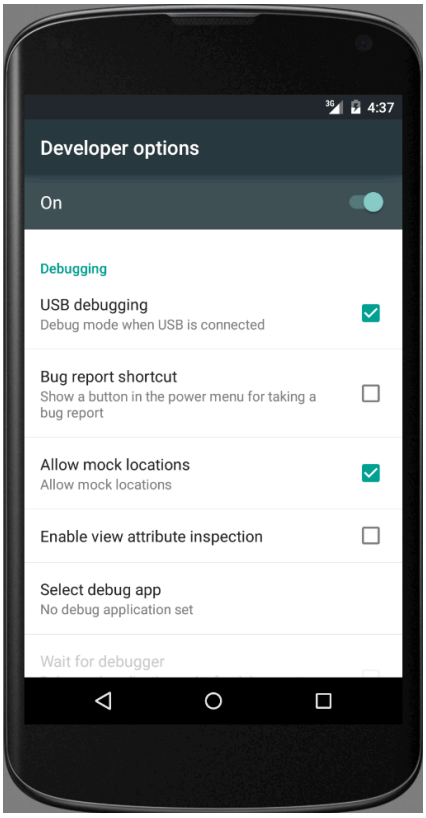
For many activities, it is important not to rely completely on an emulator, but to check your app on an actual device. For instance, this is particularly useful when you are testing advanced graphics rendering, utilizing location services, or making use of advanced sensors.

6.1 Configuring a Physical Device for Development

Almost any Android device can be used for development. In order to configure a device, enable the option in the Settings app on the device. To do this, open the Settings app, then select Applications → Development. Place a checkmark next to USB debugging as shown below for NexusOne smartphone. If your Android device is running Android 4.0 or above, the setting is located in a slightly different place within the Settings app, namely "**Developer options**" as shown below for Nexus 4 smartphone.



NeuxOne (Android 2.3)



Nexus 4 (Android 5.0)

For Samsung Galaxy Devices such as S5 or Note4, you need to use the following steps to enable the Developer Options:

1. Open the App drawer.
2. Launch Settings menu.
3. Find the open the 'About Device' menu.
4. Scroll down to 'Build Number'.
5. Next, tap on the 'build number' section seven times.
6. After the seventh tap you will be told that you are now a developer.
7. Go back to Settings menu and the Developer Options menu will now be displayed.

In order to enable the USB Debugging you will simply need to open Developer Options, scroll down and tick the box that says 'USB Debugging'.

Note: If “device unauthorized” error message is report during connection, you can use the adb to check the problem as follows:

1. Check if authorized:

```
<ANDROID_SDK_HOME>\platform-tools>adb devices
List of devices attached
4df798d76f98cf6d        unauthorized
```

2. Revoke USB Debugging on phone

If the device is shown as **unauthorized**, go to the developer options on the phone and click **"Revoke USB debugging authorization"** (tested with JellyBean & Samsung GalaxyIII).

3. Restart ADB Server: Then restarted adb server

```
adb kill-server
adb start-server
```

4. Reconnect the device: The device will ask if you are agree to connect the computer id. You need to confirm it.

5. Now Check the device: It is now authorized!

```
adb devices
<ANDROID_SDK_HOME>\platform-tools>adb devices
List of devices attached
4df798d76f98cf6d        device
```

If you want to learn how to enable Developer Options of other Android devices, you can use Google to search for the detail information with use of the keywords such as “Device Name Developer Options”. Here is an example of the webpage link for “**How to Enable Developer Options on Samsung Galaxy S8**”:

https://androidcure.com/enable-developer-options-samsung-galaxy-s8-plus/

6.2 Run your app on the real devices using USB connection

If you have a real Android-powered device, here's how you can install and run your app:

- Plug in your device to the computer with a USB cable. If you're developing on Windows, you might need to install the appropriate USB driver for your device.

- Enable USB debugging on your device.
    - On most devices running Android 3.2 or older, you can find the option under Settings > Applications > Development.
    - On Android 4.0 and newer, it's in Settings > Developer options.

To run the BMI app from Android Studio:

- Open the BMI project's file and click Run from the toolbar.
- In the Run as window that appears, select Android Application and click OK.

Android Studio installs the app on your connected device and starts it. Or you can transfer the BMI.apk file of your app to an Android Device that available in our lab. Install your app into the device and show our demonstrator to obtain the marks of this Lab02.

Check Point 3:

Running your BMI app with Data Validation feature and User Preference on a real Android device.
Instructor Verification (separate page)

VII. Import Android Sample Project (Optional)

7.1 Download the Android Sample Projects

To help you understand some fundamental Android APIs and coding practices, a variety of sample code is available from the Android SDK Manager. Each version of the Android platform available from the SDK Manager offers its own set of sample apps. To download the samples:

1. Launch the Android SDK Manager.
    - On Windows, double-click the SDK Manager.exe file at the root of the Android SDK directory.
    - On Mac or Linux, open a terminal to the `tools/` directory in the Android SDK, then execute `android sdk`.
2. Expand the list of packages for the latest Android platform.
3. Select and download *Samples for SDK*.

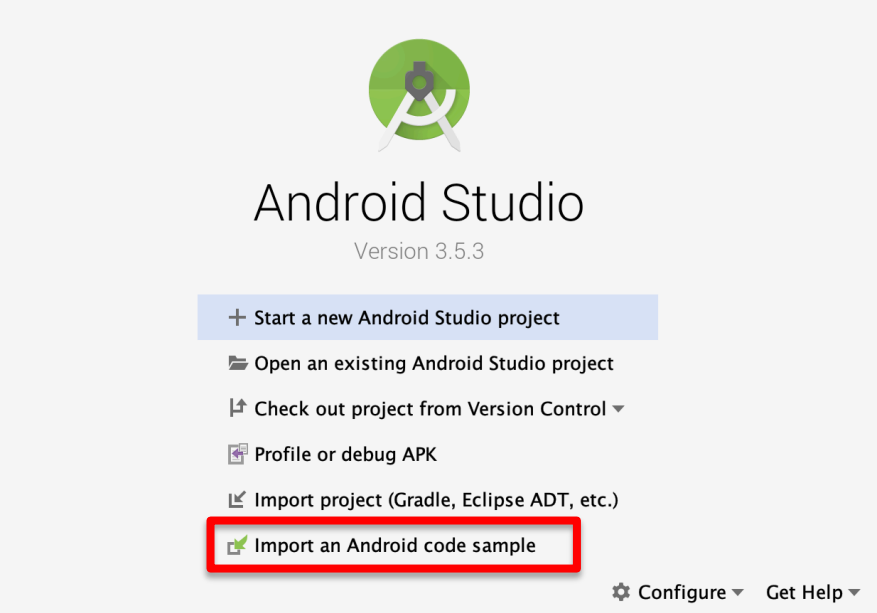
When the download is complete, you can find the source code for all samples at this location:
`<sdk>/samples/android-<version>/`

The `<version>` number corresponds to the platform's API level.

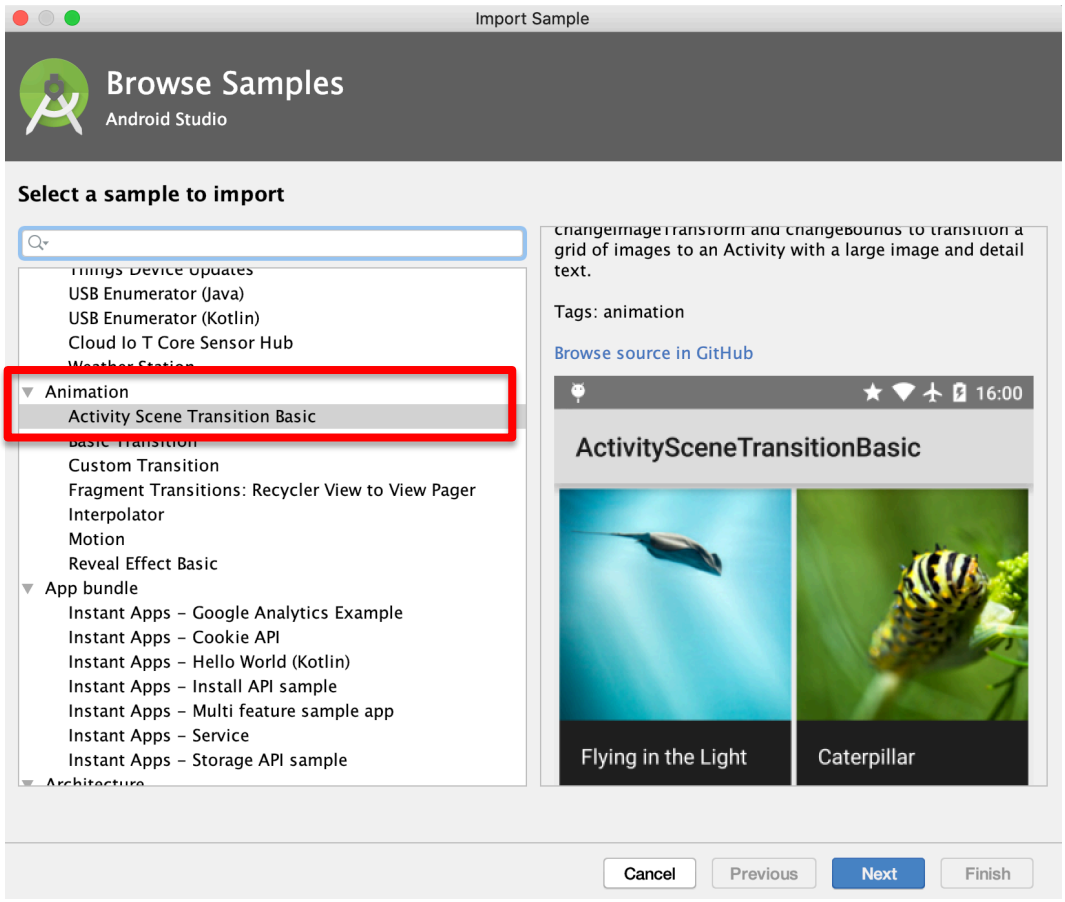
You can easily create new Android projects with the downloaded samples, modify them if you'd like, and then run them on an emulator or device.

7.2 Import the "Display Bitmaps" Sample Project

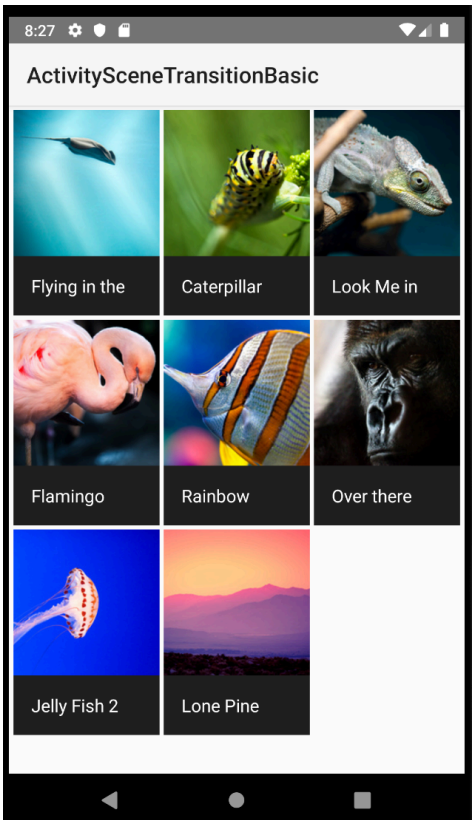
In Android Studio quick start menu, select "Import an Android code sample" for browser the downloaded sample projects:



Browser the available sample project and import the "Activity Scene Transition Basic" Project under the Animation category. This sample project demonstrating how to use scene transitions from one Activity to another in Lollipop. Uses a combination of changeImageTransform and changeBounds to transition a grid of images to an Activity with a large image and detail text.



Run this sample project on your AVD as shown below:



Students are also encouraged to study the structure of this project.

EE5415 Lab 02: BMI Calculator App
Instructor Verification Sheet

(Deadline for this verification sheet is at the end of Lab session on week 3)

Student Name: _____ Student Number: _____ Date: _____

Lab: (50 marks)

Check Point 1: [30 marks]

1. Running your BMI app on your AVD with basic features (no Data Validation and User Preference)

Instructor Verification: _____

Check Point 2: [10 marks]

2. Running your BMI app with features of Data Validation and User Preference on your AVD.

Instructor Verification: _____

Check Point 3: [10 marks]

3. Running your BMI app with Data Validation feature and User Preference on a real Android device.

Instructor Verification: _____

EE5415 Lab02: BMI Calculator App
Short Questions

Student Name: _____ Student Number: _____ Date: _____

Short Questions: (50 marks for 3 marks per question exception question 13 for 5 marks)

1. Android Studio is based on which edition of IntelliJ IDEA?

2. Which update channel of Android Studio could provide you to access to software as soon as it is available?

3. What is the purpose for selecting the minimum SDK version when starting a new Android Studio project?

4. What is the purpose of installing Intel HAXM in Android Studio development environment?

5. What are the four layers in the Android Software Stack?

6. Describe the major reasons of Android XML resources are compiled to a binary form when stored to a .dex file?

7. Where is the icon image file an Android application specified and folder location of the image file?

8. What is the superclass of all activities in Android?

9. State a method of the Activity class for starting another activity.

10. Assume you want to develop an app for devices of Android version 2.2 (Froyo) and later. How do you specify the minimum required API level in the Gradle Scripts of the Android Studio?

11. What is Gradle in Android Studio?

12. In creating your Android app, why we do not set the minSDK to Android API 1 for supporting all Android devices?

13. Why we would like to set the targetSDK of your Android project to the latest Android API release? [5 marks]

14. The Android App Framework revolves around this programming language:

15. Do we need to install the Hardware Acceleration Manager from the SDK manager for creating virtual devices for testing an application?

16. List the three major directories exist under the app module directory in the Project window.