User Interface Design

EE5415

Mobile Apps Design and Development

Message

- Friendly reminder, please prepare the group project proposal and presentation in week 5.
- Due to the outbreak of the novel coronavirus, the proposal presentation has been changed to video clip and PPT submissions.
  - Each project team must use a PPT file to record a 5-minute video presentation. You can combine multiple video clips of some members of your team as a 5-minute video presentation for satisfying the requirement.

Proposal Submission Requirements

- Submit your project proposal in MS-Words format presentation in PPT format, and video presentation in MPEG-4 (.mp4) format to ee5415@gmail.com with subject
  - EE5415 Project Proposal: Group 1 (Project Name)
- Only attachments are accepted for the submission of MS-Words and PPT files.
- On-line storages are only accepted for video file submissions:
  - For example, Google Drive, Dropbox, etc.
  - Please try to not using China's on-line storages for the video submission, as we may has difficulty to download from these sites.

Lab02 Submission

- For Lab02 submission, students need to scan the Lab02 verification form and short questions as PDF files and send these files to ee5415@gmail.com on or before 11:00pm, Feb 12, 2020.
- The subject of the email submission should be
  - EE5415 Lab02 Submission: Student Name (Student Number)

New Lab Exercise Arrangement

- Students are required to perform lab exercises on your PC or laptop.
- Revised Instructor Verification Sheet in MS-Words format are available in the schedule webpage.
- Students need to take screenshots to capture the results of the AVD (Android Virtual Device) and paste the captured images into a verification sheet to demonstrate the completion of the checkpoint.

EE5415 Lab03: Layouts and Widgets
Instructor Verification Sheet

(Submit this sheet to Lab Technician at the end of the Lab session)

Student Name: _____ Student Number: _____ Date: _____

Lab: (55 marks)

Check Point 1: [15 marks]

1.1 Show your LinearLayout Project with use of different weights.
1.2 Show your ConstraintLayout Project with buttons aligns to the left.
1.3 Show your TableLayout Project.

Capture the AVD screens of the above three results in the follow space:

Lab Submission

- The deadline for the lab submission is changed to next session. Thus, the Lab03 submission deadline is Wednesday of week 4.
- Submit the Lab Verification Sheet with captured AVD results and answers of the short questions in MS-Words or PDF format to ee5415@gmail.com with subject:
  - EE5415 Lab03 Submission: Student Name (Student Number)

Today Content

- **User Interface Design for Mobile Devices**
- **Android Layouts and UI Elements**
- **Lab03: Layouts and Widgets**
  - LinearLayout, ConstraintLayout and TableLayout
  - Widgets:
    - Image button, Edit Text, Check Box, Radio Button, Toggle Button, Rating Bar
  - Enhancement of BMI App
    - Support Landscape Display Mode
    - Add a button for “About BMI”

User Interface Design for Mobile Devices

App Design Process

1. **To define the Story (Vision) of the app**
2. **To define the functions, features and user interfaces to achieve the goals of your app**
3. **Implement the app by the Android Studio**
4. Test the app on virtual device and then real devices
5. User evaluation and may be go back to one of the previous step.
6. Release to the market

Why Humans Dominate the Earth

- Sapiens : A Brief History of Humankind
  - A bestselling book by Yuval Harari
- In this book, Prof. Harari explained why Humans dominate the earth
  - **Sapiens can cooperate in extremely flexible ways with countless numbers of strangers.**
- Why human beings can cooperate on such a large scale because **human can tell "story", especially "fiction".**
  - Many imaginative stories have emerged in human history.
  - Tens of millions or even billions of human beings, because they believe in the same story, work together in countless situations to advance towards the same goal and complete various historical initiatives.
  - Religion is the most obvious example.

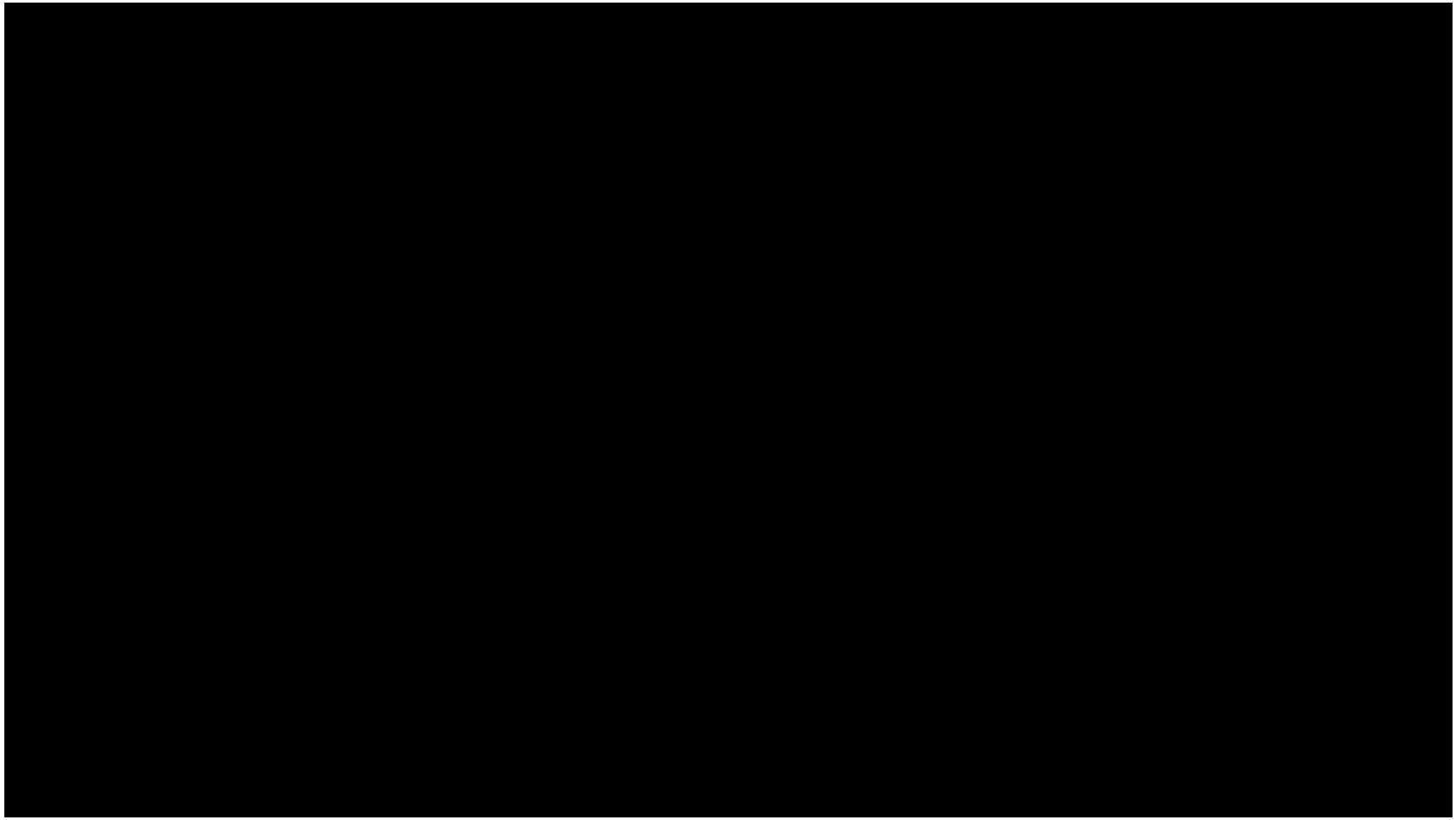
https://fs.blog/2016/01/why-humans-dominate-earth/

Define the Story of your App

- BMI App
  - Provide user-friendly BMI calculation to let users know their health status
- Feng Shui Compass App
  - Use the embedded electronic compass of the smartphone to measure and analyst the House’s Feng Shui as a real Feng Shui master

Feng Shui Compass Stories

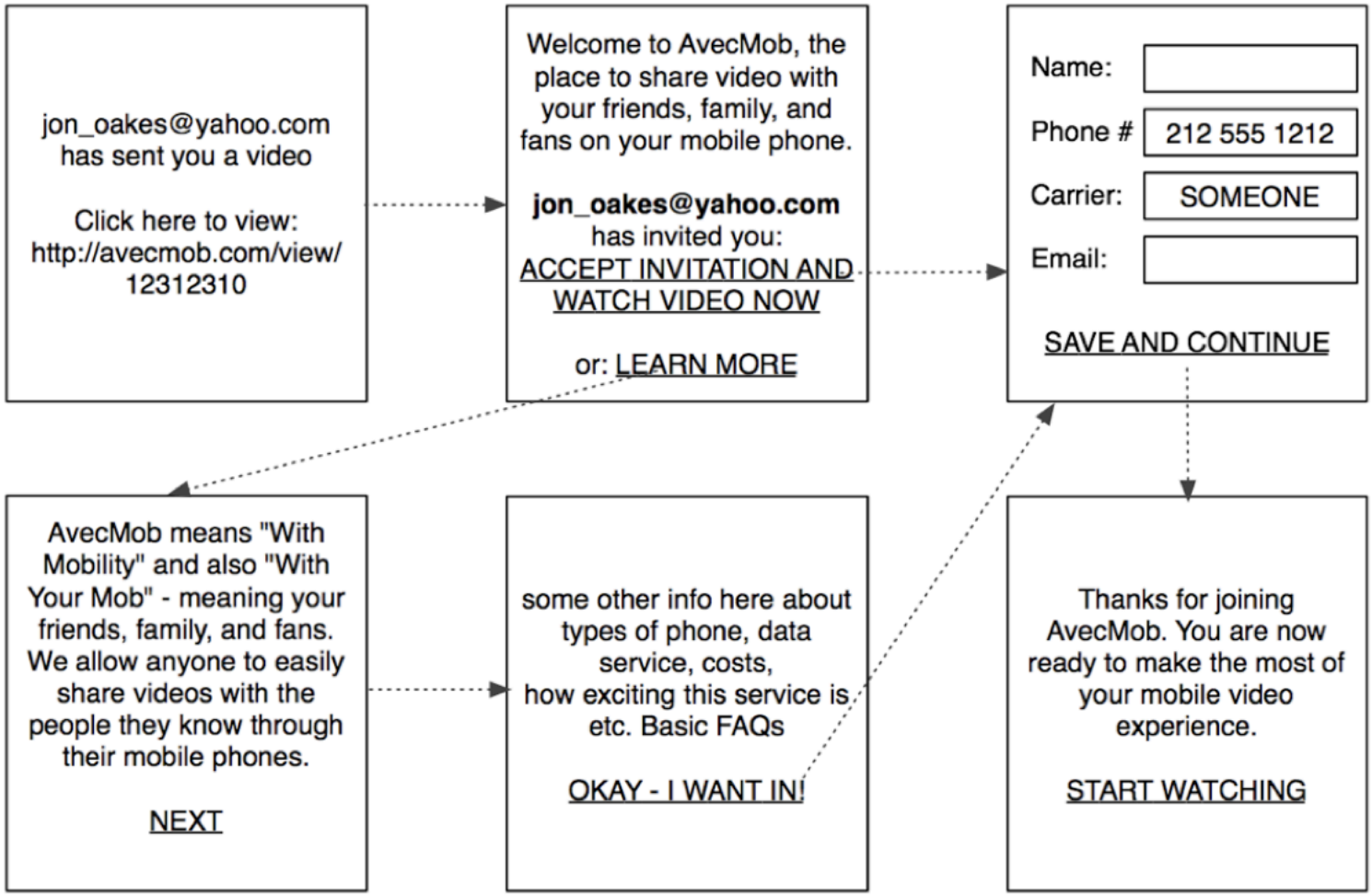
- Measure the door direction by the smartphone electronic compass and enter some basic information, then the app can provide analysis results



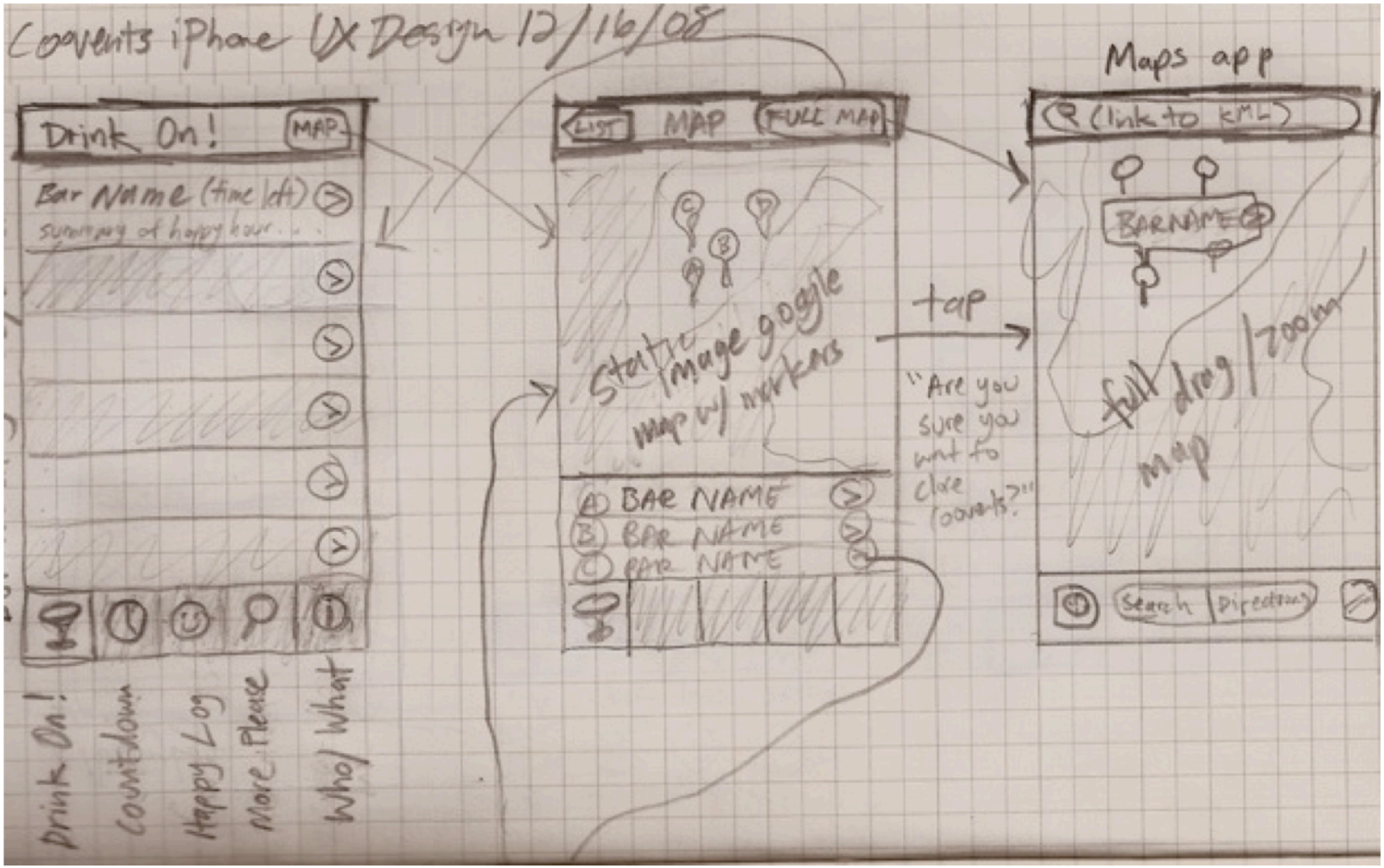
How to Start Your App Design

- **Create Your User Stories**
  - I want to be able to find all the open happy hours nearby so I can save money
  - While I am shopping at a store, I want to be able to find out if I can get it cheaper online
  - I want to be able to poll my friends for advice at any moment
  - … etc

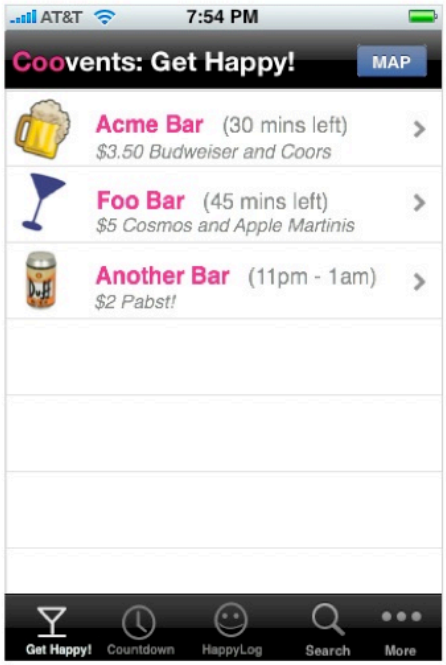
Transform the functionality to UIs

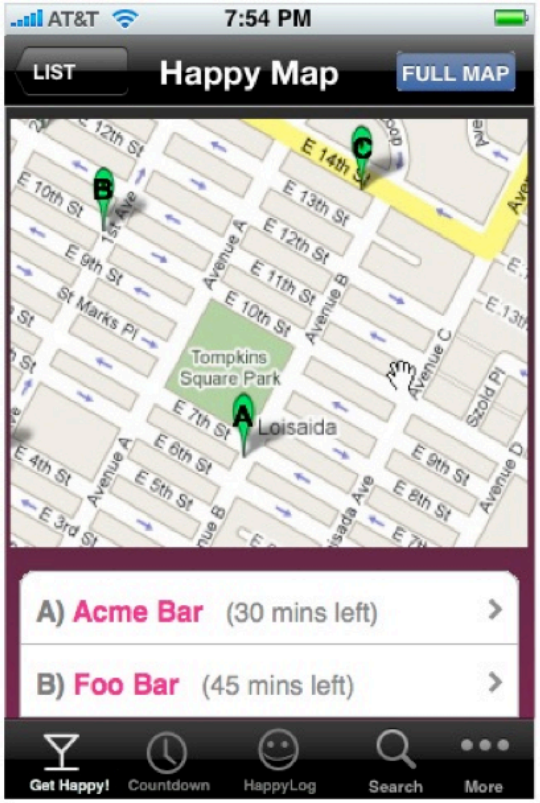


Iterating with Paper



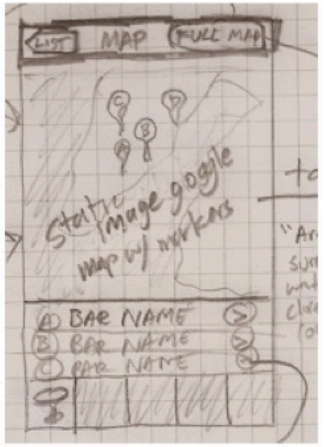
At some point, ink turns into pixels





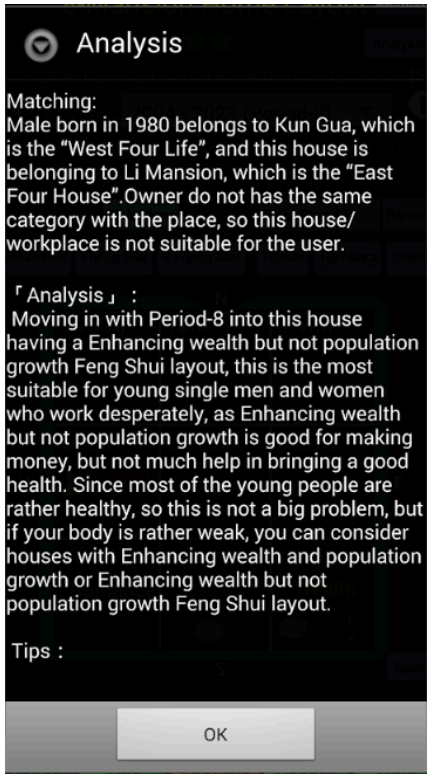
Loads full iPhone Maps application for zoom, directions, etc.

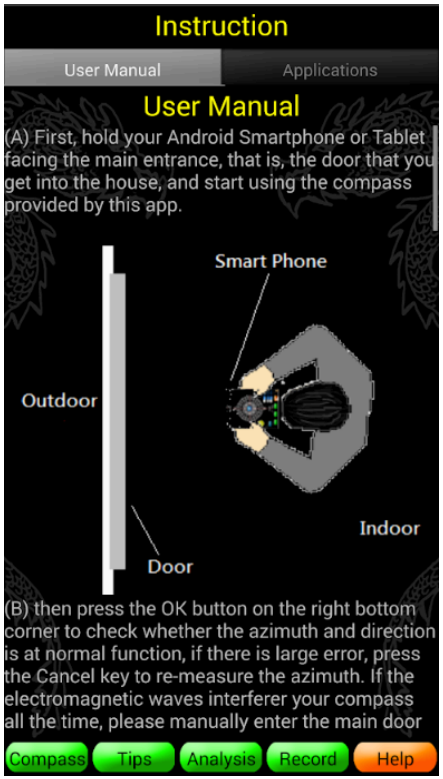
Static image map showing current list of items




Face North(Ren)
347.2
Manual Input
Accuracy:±44.9
CrossLine:
OFF
ON
Enter
167.2
Sit South(Bing)
Compass
Tips
Analysis
Record
Help


8-Mansion Home Layout
Camera
Feng Shui Index:
Analysis
Door Direction: Sit South Face North (352.4°) [Set]
Move-in Date: 2004 - 2023 (Period-8)
Birthday: 01-01-1980 (Male) (West 4 House) (Kun)
3-Cycle: Sit Bing Face Ren (Good Wealth)
Li Mansion
Selections
Floor Plan
Set
Remove
8-Mansion
Flying Star
9 Flying Star
Roman
Romance
Health
N
W
E
S
Jue Ming
Yan Nian
Huo Hai
Wu Gui
Ominous
Life Gua
Sheng Qi
Liu Sha
Romance
Fu Wei
Stable
Tian Yi
Wealth
Save
Compass
Tips
Analysis
Record
Help


Analysis
Matching:
Male born in 1980 belongs to Kun Gua, which is the "West Four Life", and this house is belonging to Li Mansion, which is the "East Four House".Owner do not has the same category with the place, so this house/ workplace is not suitable for the user.
「Analysis」:
Moving in with Period-8 into this house having a Enhancing wealth but not population growth Feng Shui layout, this is the most suitable for young single men and women who work desperately, as Enhancing wealth but not population growth is good for making money, but not much help in bringing a good health. Since most of the young people are rather healthy, so this is not a big problem, but if your body is rather weak, you can consider houses with Enhancing wealth and population growth or Enhancing wealth but not population growth Feng Shui layout.
Tips :
OK


Instruction
User Manual
Applications
User Manual
(A) First, hold your Android Smartphone or Tablet facing the main entrance, that is, the door that you get into the house, and start using the compass provided by this app.
Smart Phone
Outdoor
Indoor
Door
(B) then press the OK button on the right bottom corner to check whether the azimuth and direction is at normal function, if there is large error, press the Cancel key to re-measure the azimuth. If the electromagnetic waves interferer your compass all the time, please manually enter the main door
Compass
Tips
Analysis
Record
Help

User Experience, Desires & Dreams

- Think about the 60-second use case …
- And then think about the 60-minute use cases.
- **60-second = quick reference, lookup or message**
  - *Stocks, scores, weather, calculator, …*
- **60-minute = long, in-depth interaction while relaxing, wasting time or on a train or plane**
  - Game, news, reading, web browsing, music, video, ….

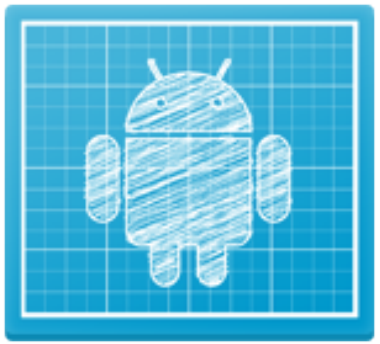
User Interface for Mobile Devices

- When developing software for the desktop computers, you only need to consider the mouse and the keyboard.
- For mobile devices, you must take into account all the possible input sources for mobile devices:

  - Keyboard (hard and soft)
  - “Click” Tap
  - Multitouch and Gestures
  - GPS or Network Location
  - Accelerometer Motion
  - Vibration
  - Sound / Music
  - Wireless Network Coverage
  - Environment Lighting
  - Device Security
  - Orientation / Compass / Altitude

Android UI Guidelines

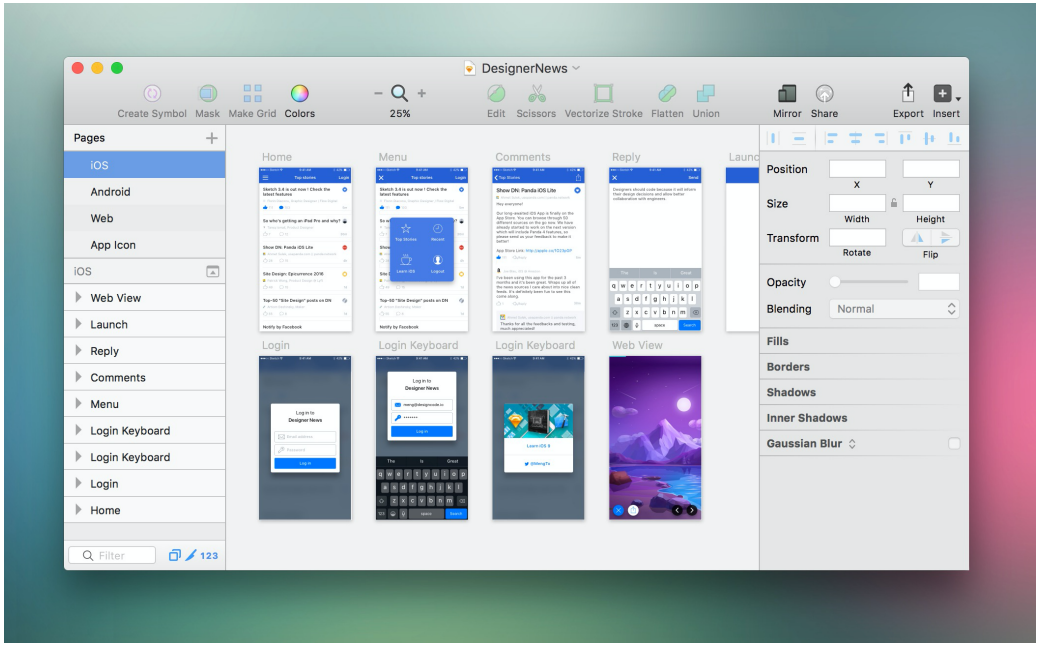
- New Guides for App Designers!
  - The Android UI team has put together a set of guidelines for the interaction and visual design of Android applications.
  - The new collection provides an overview of Android styles, design patterns, building blocks for exceptional Android designs, and more.



- http://developer.android.com/guide/practices/ui_guidelines/index.html

UI Design Tools

- Photoshop: Raster Graphics and expensive
- GMIP : Raster Graphics and Free
- Adobe Illustrator : Vector Graphic and quite expensive
- **Sketch 3 :** Vector Graphic and not very expensive
  - Support Mobile App UI Design



Android Layouts and UI Elements

Android Project Structure

- **app**
  - **manifest**
    - AndroidManifest.xml (App Configuration)
  - **java**
    - pkg name >> **MainActivity.java**
  - **res**
    - drawable >> *.png (graphic images)
    - **layout >> activity_main.xml (UI XML)**
    - mipmap >> app icon
    - values >> string.xml, styles.xml, dimensions.xml, etc
    - menu >> menu.xml
    - raw >> *.mp3, *.mp4
  - **build>>generate>>source>pkg name>r>debug>R.java** (Providing references for Java programs to access the resources in res)
  - **build>>outputs>>apk>>*.apk** (apk is termed as Android Application Package it is the installation file in Android platform, which include .dex, res, assets, lib, AndriodManifest.xml)
- **gradle** (A build tool for Android application)
  - local.properties (locations of SDK, NDK, etc)
  - gradle.properties (buildToolsversion, minSdkVersion, targetSdkVersion, versioncode, versionName, dependenices)

What is XML?

- XML is termed as **Extensible Markup Language**.
  - You may know what is HTML (Hypertext Markup Language) for webpage
- Markup Language: enclosing the data within tags is called as Markup Language. For example:

```
<employees>
  <employee>
    <id>12345</id>
    <name>Peter Lee</name>
    <position>Android Developer</position>
    <dept>Mobile App</dept>
  </employee>
</employees>
```

Why XML?

- XML is interoperable
  - Platform independent and Technology independent
  - For any platform (Android, Linux, Windows) and technology (Java, php, C/C++, .net) can read the XML
- XML is used to transfer the data between multiple technologies
  - XML is used to as textual database
    - No database software is required
  - XML is used as deployment descriptor (Manifest.xml)

Basic Rules for XML

1. Every element must be properly nested
   - **<name>**Peter Chan**</name>**
2. Every XML file should have only one root element
3. Should not contain spaces between the tag names
   - <first name> is not allowed => <first_name>
4. Should not start with a number and special characters
   - <1name> is not allowed => <name1>
   - <@name> is not allowed => <name@>

Define Your Own Rules for XML

- DTD and XSD are used to define your own rules for XML files.
  - **DTD : Document Type Definition** => Old Approach
  - **XSD : XML Schema Definition**
- Android uses XSD to define the rules for the XML files such as AndroidManifest.xml and layout XML files

XML Layout Files

- Files contain instructions for placing objects on the screen.
- They are built with XML and implemented in the background with Java
- Each XML element describes an instance of a Java class.

A Simple Layout File

Name spaces:
android => Core SDK
app => External Libraries
tools => Android Studio

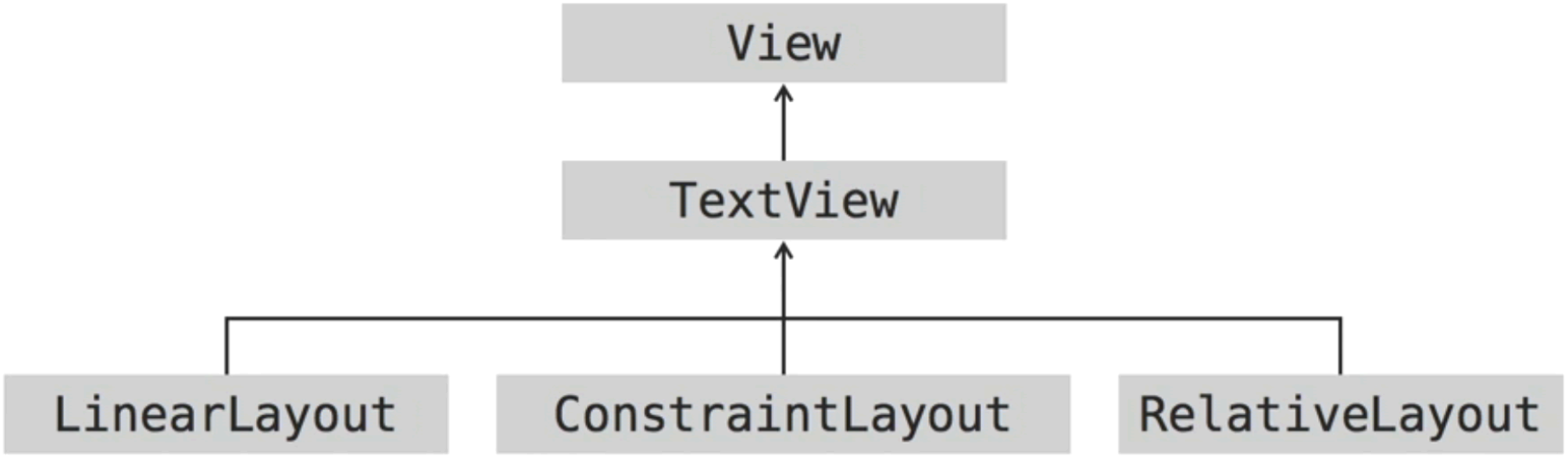
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
  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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Namespaces

Dimension

A Visual Control is a View

A ViewGroup is also a View



A **ViewGroup** is a container. It is used to organize other objects on the screen.

Traditional ViewGroup Components

- **FrameLayout :** contains a single child view
- **LinearLayout :** lays out horizontally or vertically
- **RelativeLayout :** relative to container or other objects
- **GridView :** a scrollable grid
- **ListView :** a vertical list of scrollable items

LinearLayout

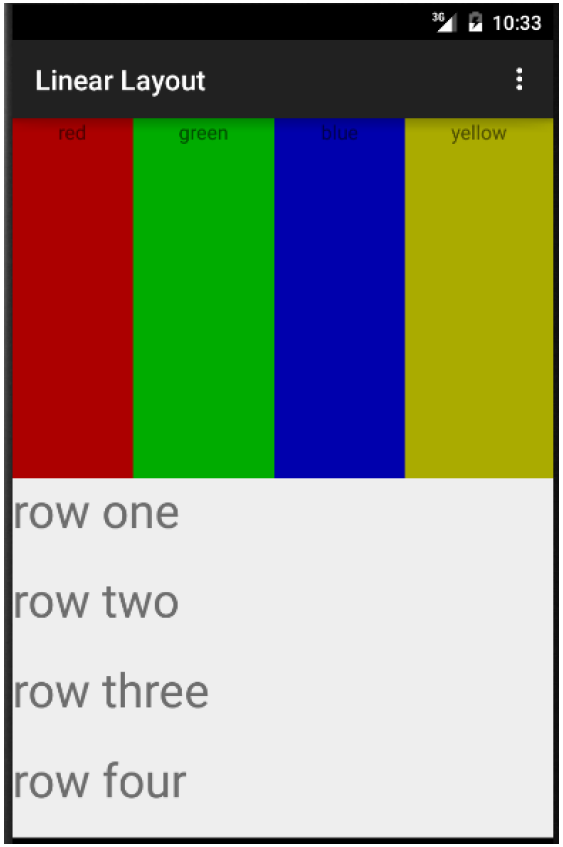
- LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the android:orientation attribute.

LinearLayout Exmaple

- Child views arranged in a single horizontal or vertical row

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="horizontal" >
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="3"
    android:orientation="vertical" >
```



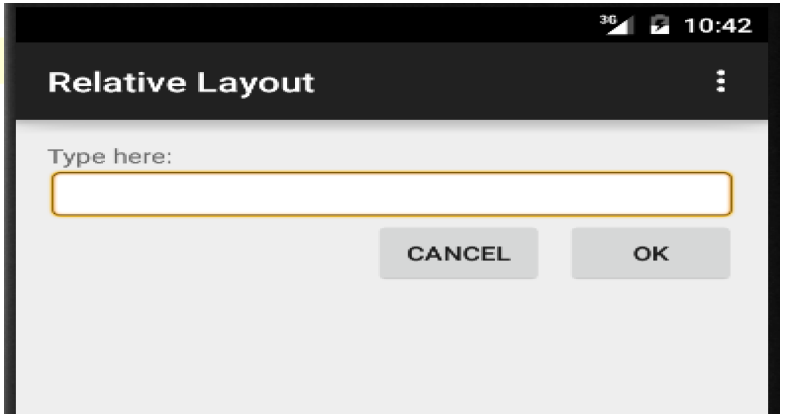
RelativeLayout

- RelativeLayout is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements or in positions relative to the parent RelativeLayout area.

RelativeLayout Example

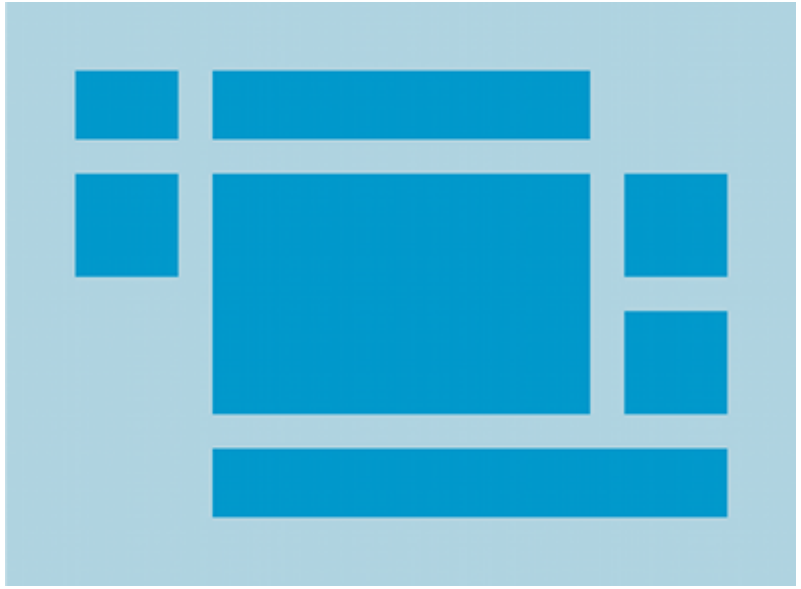
- Child views are positioned relative to each other and to parent view

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp" >

    <TextView
        android:id="@+id/label"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Type here:" />

    <EditText
        android:id="@+id/entry"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/label"
        android:background="@android:drawable/editbox_background" />
```



TableLayout

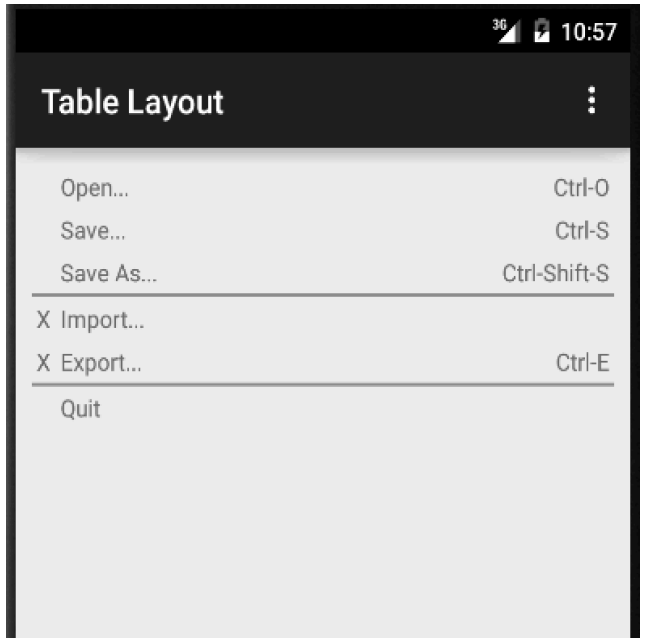
- **TableLayout is a ViewGroup that displays child View elements in rows and columns.**
- TableLayout positions its children into rows and columns. The table will have as many columns as the row with the most cells. A table can leave cells empty, but cells cannot span columns, as they can in HTML.



TableLayout Example

- Child views arranged into rows & columns

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1" >

    <!-- First row -->
    <TableRow>

        <!-- start in column 1 -->
        <TextView
            android:layout_column="1"
            android:padding="3dip"
            android:text="@string/open_string"
            android:textSize="24sp" />

        <TextView
            android:gravity="right"
            android:padding="3dip"
            android:text="@string/ctrl_o_string"
            android:textSize="24sp" />
    </TableRow>
```

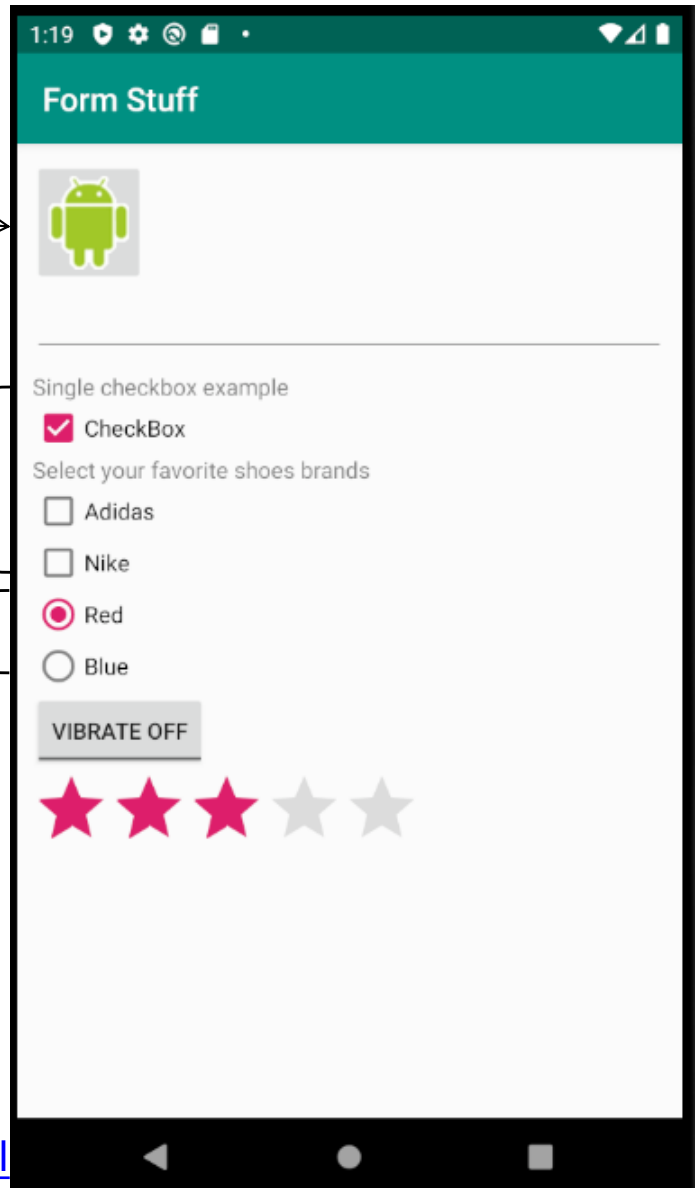


Newer ViewGroup Components

- **CoordinatorLayout :** a super-powered FrameLayout
- **ConstraintLayout :** replaces RelativeLayout
- **RecyclerView :** replaces ListView and GridView
- **AppBarLayout :** manages the app bar
- **DrawerLayout :** adds a pullout navigation bar

Common UI Elements

- Button (Image Button)
- EditText
- Checkbox
- Radio button
- Toggle button
- Rating Bar
- … etc



http://developer.android.com/guide/topics/ui/controls.html

Common View Properties

Layout Width	FILL_PARENT, WRAP_CONTENT or specific size in units
Layout Height	FILL_PARENT, WRAP_CONTENT or specific size in units
Layout Weight	How greedy or generous a widget is. Value is 0 to 1.
Layout Gravity	How to align widget within parent
Gravity	How to align content of widget within itself
Text	Text for the content of the component, usually @string/mytext
id	Unique id of this widget, usually @+id/someUniqueID

Units of Measurement

- Using Dimension Values
  - Dimension values are set for visual components
  - Set in XML-layout files or programmatically

Device-Independent Pixels (dp)

- They are used to set position and size of containers and widgets
- On a 160 dpi screen, each dp is mapped to one physical pixel
- Ratio of dp-to-pixel changes with screen density

Scale-Independent Pixels (sp)

- Similar to dp but used to specify font size

Fixed Units of Measurement

- px (pixels): regardless of device resolution or density
- pt (points): 1/72 of an inch
- mm (millimeters): based on physical size of screen
- in (inch): based on physical size of screen

Fixed units don’t adjust to different devices!

Dimension Resources

- Dimension resource files define reusable dimensions
- Place in res/values folder
- A simple dimens.xml file:

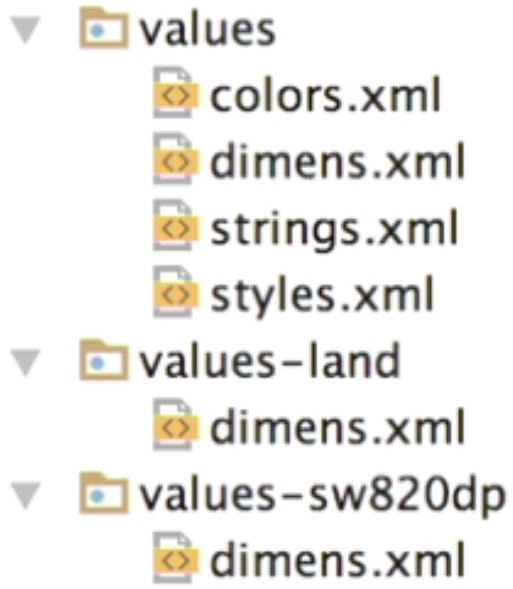
```
<resources>
  <dimen name="horizontal_margin">16dp</dimen>
  <dimen name="vertical_margin">16dp</dimen>
</resources>
```

Apply Dimension Resources

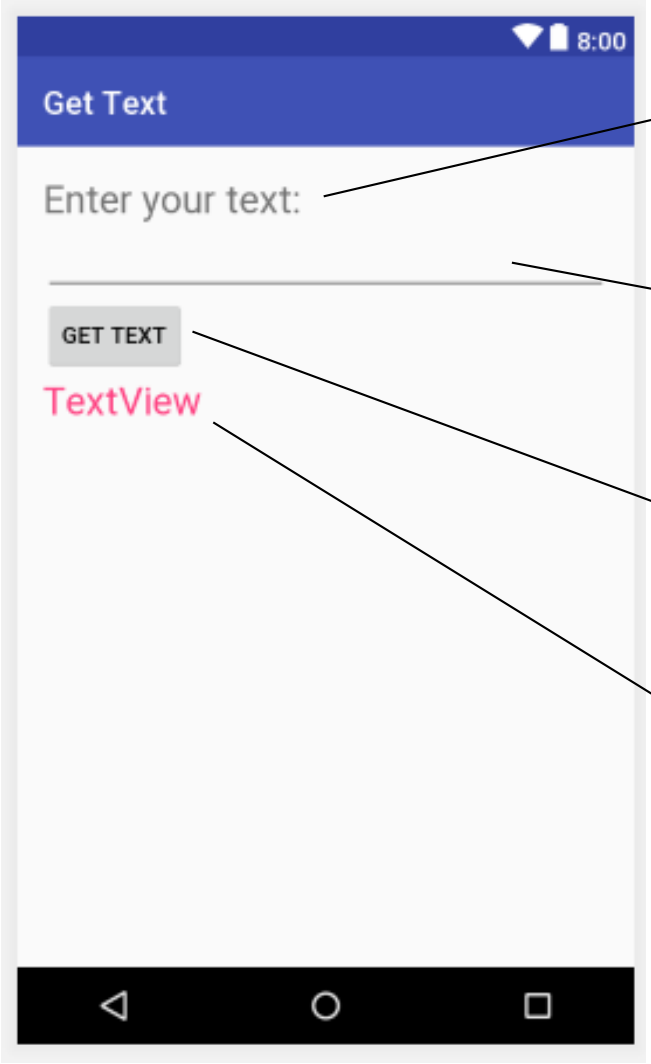
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/vertical_margin"
    android:paddingLeft="@dimen/horizontal_margin"
    android:paddingRight="@dimen/horizontal_margin"
    android:paddingTop="@dimen/vertical_margin">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</LinearLayout>
```

Manage Multiple Screens

- Alternative settings for different sizes and orientations
- Override settings in directories with name qualifiers



Android UI Example




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Enter your text:"
        android:textSize="24sp" />

    <EditText
        android:id="@+id/et1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPersonName" />

    <Button
        android:id="@+id/showButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="clickHandler"
        android:text="Get Text" />

    <TextView
        android:id="@+id/tv1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:textColor="@color/colorAccent"
        android:textSize="24sp" />
</LinearLayout>
```

"onClick" attitude and findViewById()

Java

```
public class MainActivity extends AppCompatActivity {

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

    }

    public void clickHandler(View view) {

        EditText myET = (EditText) findViewById(R.id.et1);
        TextView myTV = (TextView) findViewById(R.id.tv1);

        myTV.setText(myET.getText());
    }
}
```

XML

```
<EditText
    android:id="@+id/et1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName" />

<Button
    android:id="@+id/showButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="clickHandler"
    android:text="Get Text" />

<TextView
    android:id="@+id/tv1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:textColor="@color/colorAccent"
    android:textSize="24sp" />
```

Button Events Handling in Java Code

Java

```
public class MainActivity extends AppCompatActivity {

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

        Button myButton = (Button) findViewById(R.id.showButton);

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

                EditText myET = (EditText) findViewById(R.id.et1);
                TextView myTV = (TextView) findViewById(R.id.tv1);

                myTV.setText(myET.getText());
            }
        });


    }
}
```

XML

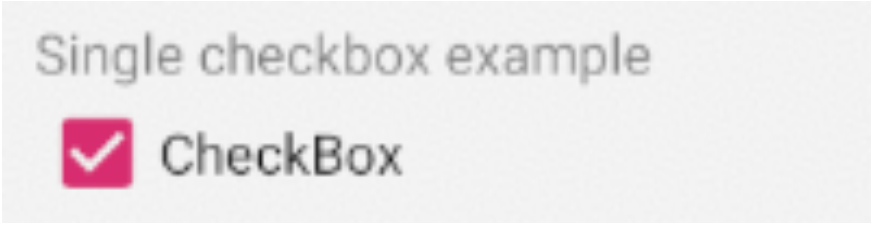
```
<EditText
    android:id="@+id/et1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="textPersonName" />

<Button
    android:id="@+id/showButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="clickHandler"
    android:text="Get Text" />

<TextView
    android:id="@+id/tv1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:textColor="@color/colorAccent"
    android:textSize="24sp" />
```

Handling View Events

- Numerous listener interfaces defined by the View class
- View Listener Interfaces
  - **OnClickListener.onClick()**
    - View has been clicked
  - **OnLongClickListener.onLongClick()**
    - View has been pressed & held
  - **OnFocusChangeListener.onFocusChange()**
    - View has received or lost focus
  - **OnKeyListener.onKey()**
    - View about to receive a hardware key press

Single Checkbox

- A 2-State Button
- Checked/Not Checked



activity_main.xml

```
<CheckBox
    android:id="@+id/cb_single"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="CheckBox" />
```

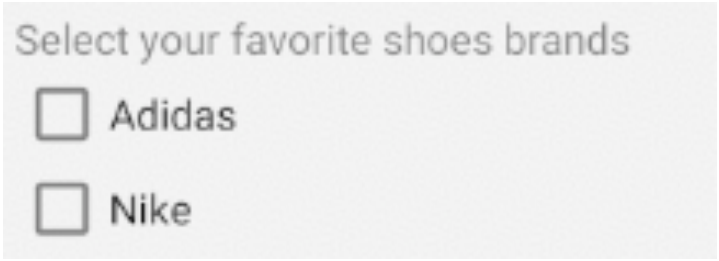
MainActivity.java

```
final CheckBox cb_single = (CheckBox) findViewById(R.id.cb_single);

cb_single.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Perform action on clicks
        if (cb_single.isChecked()) {
            Toast.makeText( context: MainActivity.this,  text: "Ckecker", Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText( context: MainActivity.this,  text: "UnCkecker", Toast.LENGTH_LONG).show();
        }
    }
});
```

Multiple Checkboxs

activity_main.xml

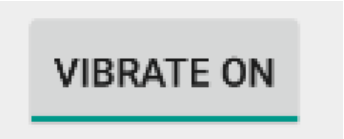
```
<CheckBox
    android:id="@+id/cb_addidas"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onCheckboxClicked"
    android:text="Adidas" />

<CheckBox
    android:id="@+id/cb_nike"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onCheckboxClicked"
    android:text="Nike" />
```



MainActivity.java

```
public void onCheckboxClicked(View view) {
    // Is the button now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.cb_addidas:
            if (checked)
                Toast.makeText( context: MainActivity.this,  text: "Addida Selected",
                        Toast.LENGTH_SHORT).show();
            else
                Toast.makeText( context: MainActivity.this,  text: "Addida Unselected",
                        Toast.LENGTH_SHORT).show();
            break;
        case R.id.cb_nike:
            if (checked)
                Toast.makeText( context: MainActivity.this,  text: "Nike Selected",
                        Toast.LENGTH_SHORT).show();
            else
                Toast.makeText( context: MainActivity.this,  text: "Nike Unselected",
                        Toast.LENGTH_SHORT).show();
            break;
    }
}
```

Toggle Button



- Another kind of 2-state Button
- Checked/Not Checked State
- Light Indicator Showing State

```
<ToggleButton android:id="@+id/togglebutton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Vibrate on"
    android:textOff="Vibrate off"/>
```

```
//**************************************************
// The Toggle Button Code
final ToggleButton togglebutton = (ToggleButton) findViewById(R.id.togglebutton);
togglebutton.setOnClickListener((v) -> {
        // Perform action on clicks
        if (togglebutton.isChecked()) {
            Toast.makeText(MainActivity.this, "Vibrate On", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(MainActivity.this, "Vibrate Off", Toast.LENGTH_SHORT).show();
        }
});
```

Radio Group and Button

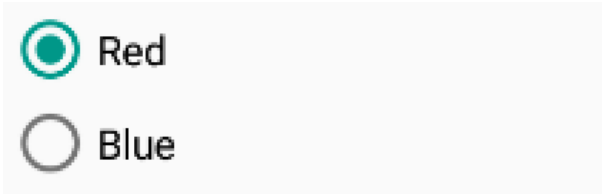
Radio Group

- A ViewGroup containing a set of Radio Buttons (CheckBoxes)
- Only one button can be selected at any one instant

Radio button

- Mutually-exclusive radio buttons (enabling one disables the other)

Radio Button




```
<RadioGroup
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton
        android:id="@+id/radio_red"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Red"
        android:onClick="onRadioButtonClicked"/>

    <RadioButton
        android:id="@+id/radio_blue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Blue"
        android:onClick="onRadioButtonClicked" />
</RadioGroup>
```

Events Handling of Radio button

```
//************************************************
// The Radio Button Code
public void onRadioButtonClicked(View view) {
    // Is the button now checked?
    boolean checked = ((RadioButton) view).isChecked();

    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.radio_red:
            if (checked)
                Toast.makeText(MainActivity.this, "Red Selected", Toast.LENGTH_SHORT).show();
                break;
        case R.id.radio_blue:
            if (checked)
                Toast.makeText(MainActivity.this, "Blue Selected", Toast.LENGTH_SHORT).show();
                break;
    }
}
```

Rating Bar

- A view comprising a row of stars
- The user can click or drag the stars to highlight some number of them

```
<RatingBar android:id="@+id/ratingbar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="5"
    android:stepSize="1.0"/>
```

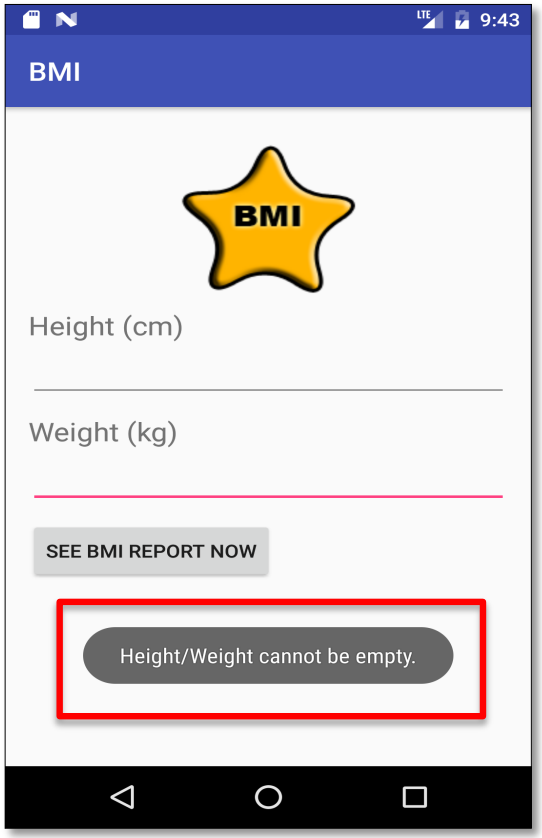
```
//**************************************************
// The Rating Bar Code
final RatingBar ratingbar = (RatingBar) findViewById(R.id.ratingbar);
ratingbar.setOnRatingBarChangeListener((ratingBar, rating, fromUser) -> {
        Toast.makeText(MainActivity.this, "New Rating: " + rating,
                Toast.LENGTH_SHORT).show();
});
```

Android Notifications

- Inform user of events that occur during application execution
  - Long running behaviors
    - Toast Notification
  - Saving / deleting files
    - Dialog Notification

Android Toast Notifications

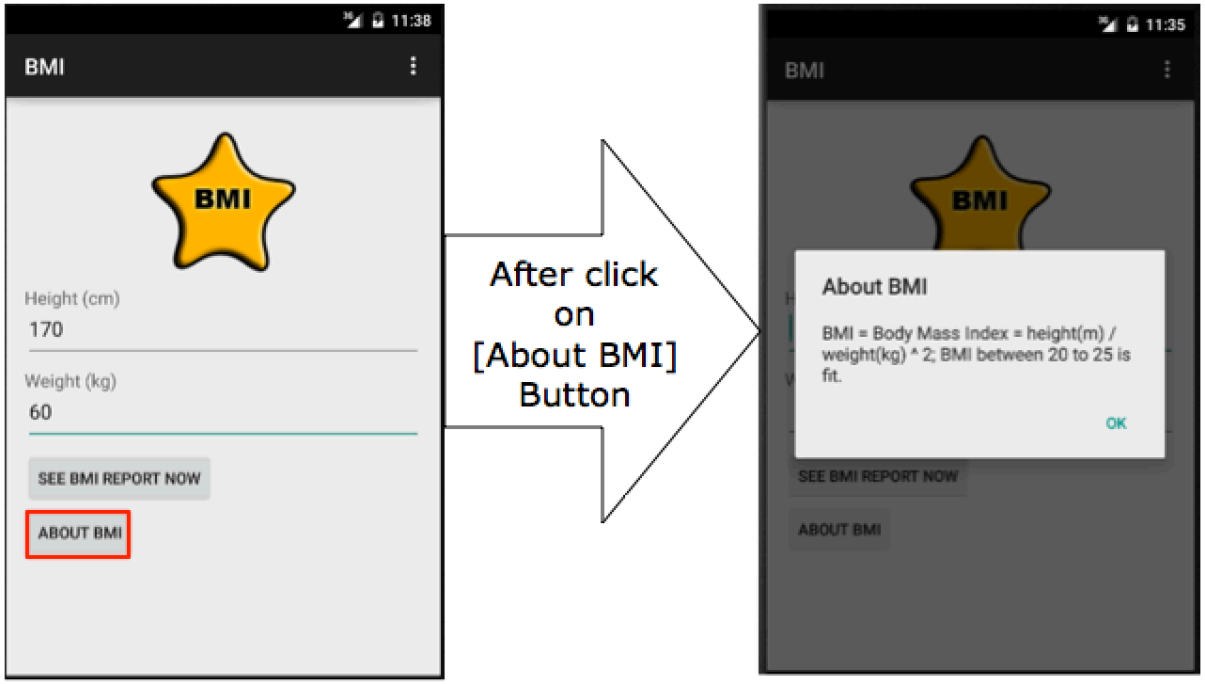
- Floating message
- Pops up from above application
- Fades in and out
- Available from background Service
- Cannot respond to user interactions




```
// When clicked, show a toast with the TextView text
   Toast.makeText(getApplicationContext(), ((TextView) view).getText(),
         Toast.LENGTH_SHORT).show();
```

Android Dialog Notifications

- **Modal window**
  - User cannot continue until dialog dismissed
- Window that pop up in front of current activity



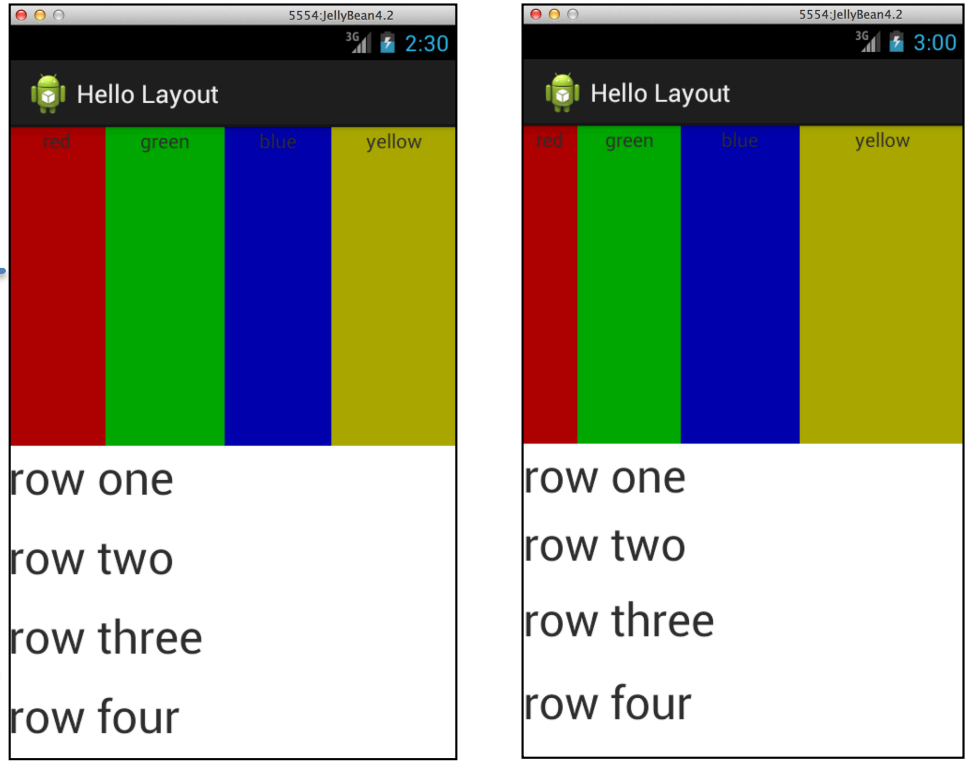
Lab03:
Layouts and Widgets

Lab03: Layouts and Widgets

- LinearLayout, ConstraintLayout, and TableLayout
- Common Andriod Widgets:
  - Image button, Edit Text, Check Box, Radio Button, Toggle Button, Rating Bar
- Enhancement of BMI App
  - Support Landscape Display Mode
  - Add a button for “About BMI” using Dialog Notification for displaying the message

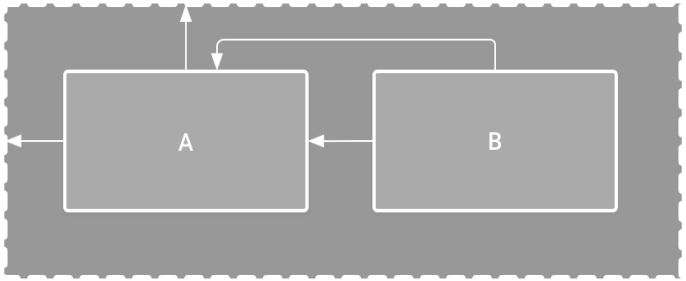
Linear Layout

```
<?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">

     <LinearLayout
       android:orientation="horizontal"
       android:layout_width=”match_parent"
       android:layout_height=”match_parent"
       android:layout_weight="1">
       <TextView
          android:text="red"
          android:gravity="center_horizontal"
      [.....................]
    </LinearLayout>

     <LinearLayout
      android:orientation="vertical"
      android:layout_width=”match_parent"
      android:layout_height=”match_parent"
      android:layout_weight="1">
      <TextView
         android:text="row one"
         android:textSize="15pt"
         android:layout_width=”match_parent"
         android:layout_height="wrap_content"
         android:layout_weight="1"/>
      <TextView
         android:text="row two"
         android:textSize="15pt"
         android:layout_width=”match_parent"
         android:layout_height="wrap_content"
         android:layout_weight="1"/>
    [.....................................]
     </LinearLayout>

    </LinearLayout>
```



http://developer.android.com/resources/tutorials/views/hello-linearlayout.html

Constraint Layout



- ConstraintLayout is much more flexible than RelativeLayout or LinearLayout.
- Basically, it is a flexible layout manager for your app that allows you to create dynamic user interfaces without nesting multiple layouts.
- It is distributed as a support library that is tightly coupled with Android Studio and backwards compatible to API Level 9.

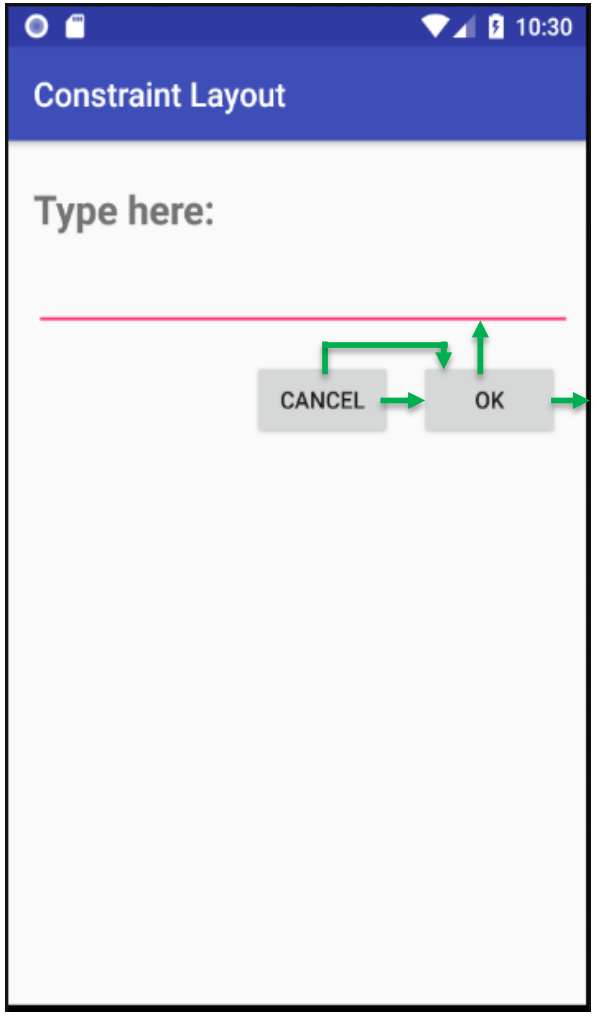
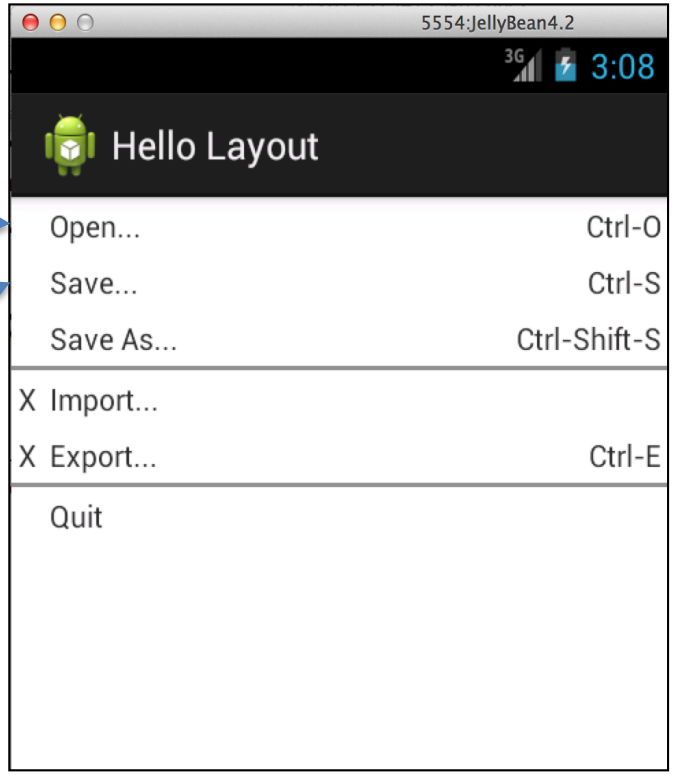


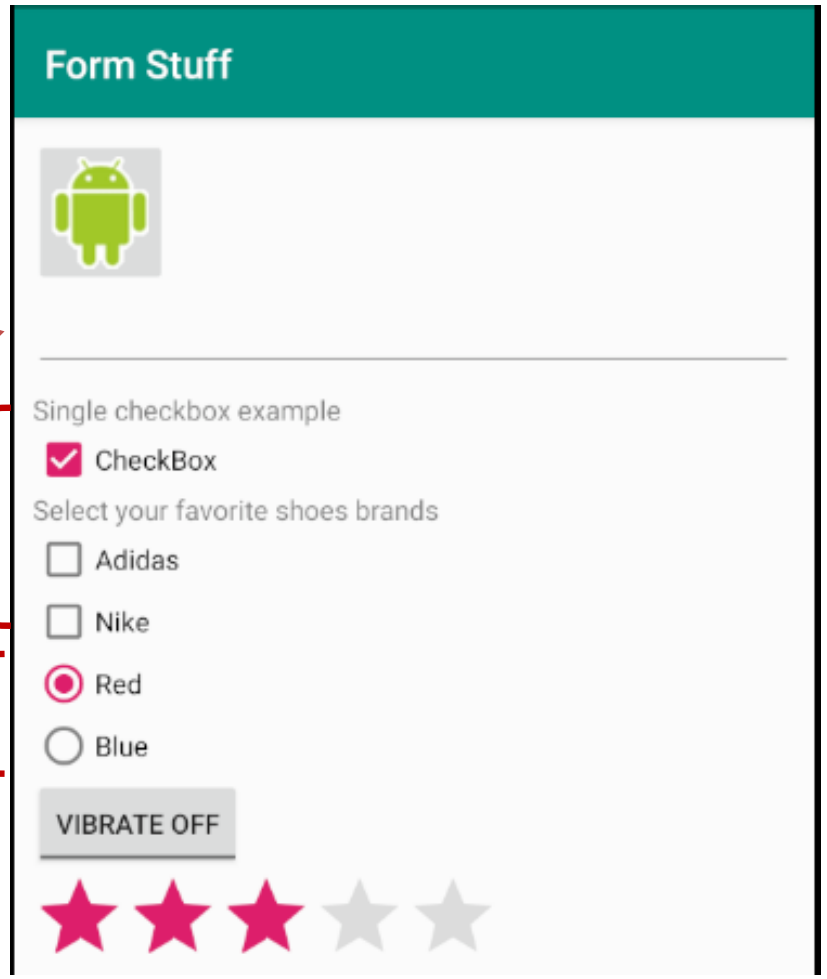
Table Layout

```
<?xml version="1.0" encoding="utf-8"?>
        <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:stretchColumns="1">

           <TableRow>
              <TextView
                 android:layout_column="1"
                 android:text="Open..."
                 android:padding="3dip" />
              <TextView
                 android:text="Ctrl-O"
                 android:gravity="right"
                 android:padding="3dip" />
           </TableRow>

           <TableRow>
              <TextView
                 android:layout_column="1"
                 android:text="Save..."
                 android:padding="3dip" />
              <TextView
                 android:text="Ctrl-S"
                 android:gravity="right"
                 android:padding="3dip" />
           </TableRow>

           <TableRow>
              <TextView
                 android:layout_column="1"
                 android:text="Save As..."
                 android:padding="3dip" />
              <TextView
                 android:text="Ctrl-Shift-S"
                 android:gravity="right"
                 android:padding="3dip" />
           </TableRow>

           <View
              android:layout_height="2dip"
              android:background="#FF909090" />

        [.....................]
        </TableLayout>
```



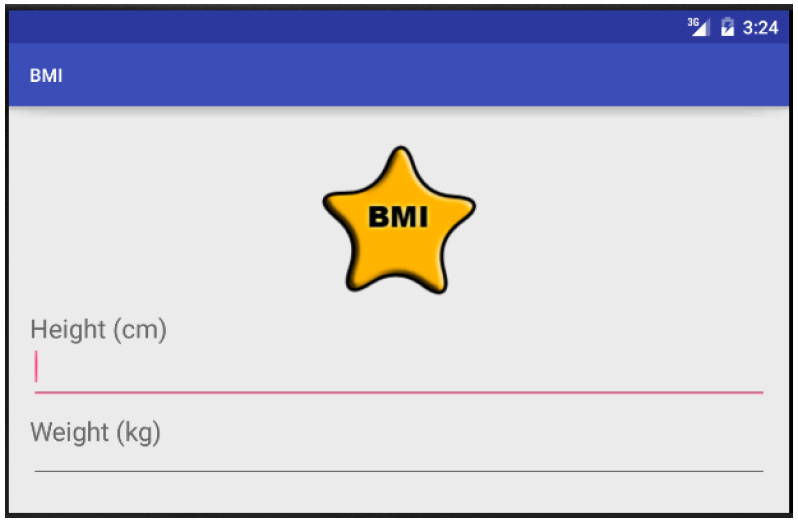
Form Stuff Project

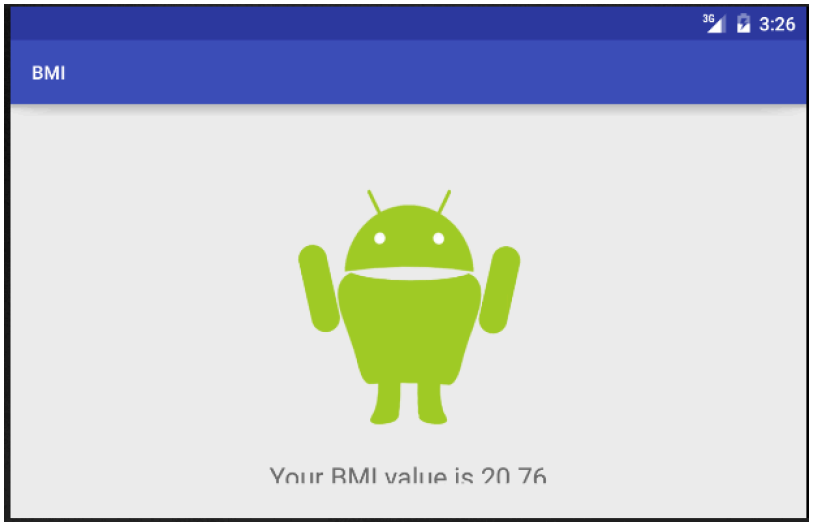
- This project covers most of the common Android widgets such as
  - Image Button
  - Edit Text
  - Checkbox
  - Radio Buttons
  - Toggle Button
  - Rating Bar



BMI App in Landscape Mode

- In the AVD, press “Ctl+F11” keys can switch the device to landscape display mode:

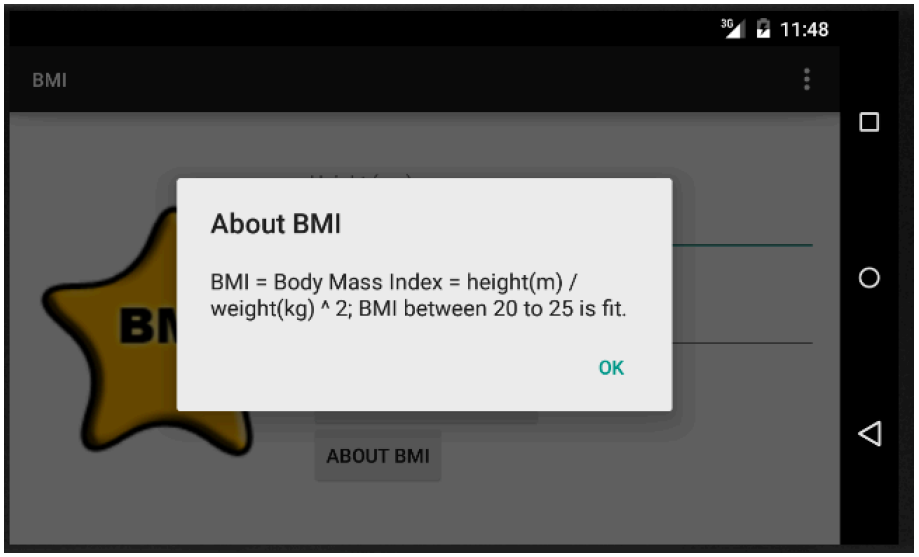




- In the landscape mode, the button of “See BMI Report Now” cannot be displayed on the screen.
- Similarly problem also appears on the Report Activity UI.

Support Landscape Display Mode

- Create two layout XML files in
  - res->layout-land->activity_main.xml
  - res->layout-land->activity_report.xml

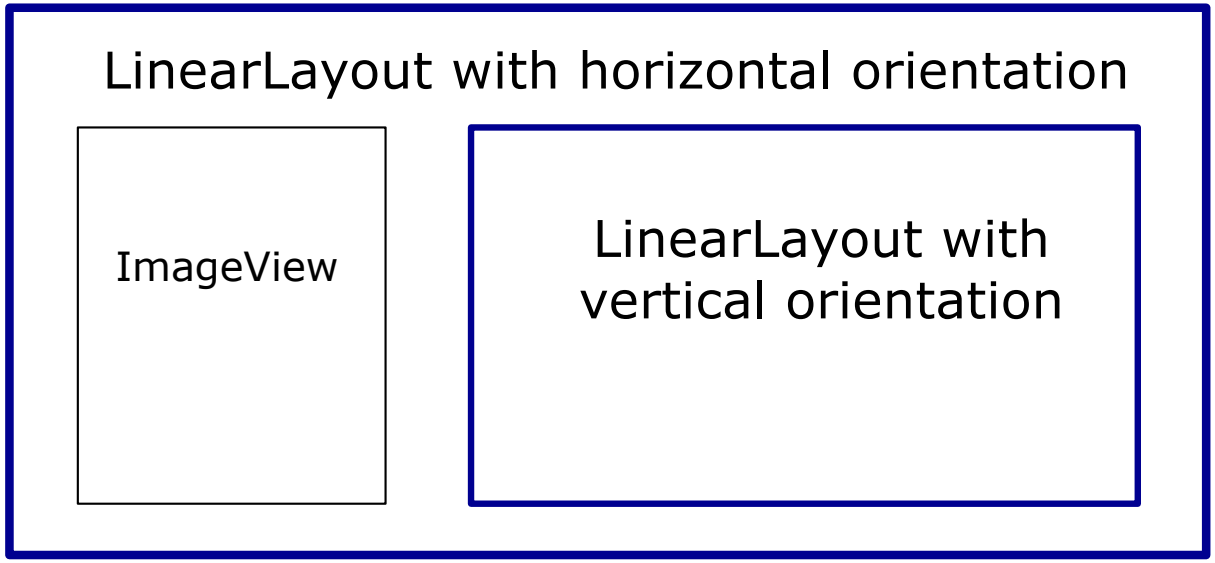


Constriant Layout Approach

- Just re-arrange the UI element positions in the activity_main.xml and activity_report.xml under the new folder of res/layout-land/ for supporting the landscape display mode.
- res/layout-land/activity_main.xml based on Constriant Layout is provided in the lab sheet.
- Students are required to create the activity_report.xml for the landscape display mode

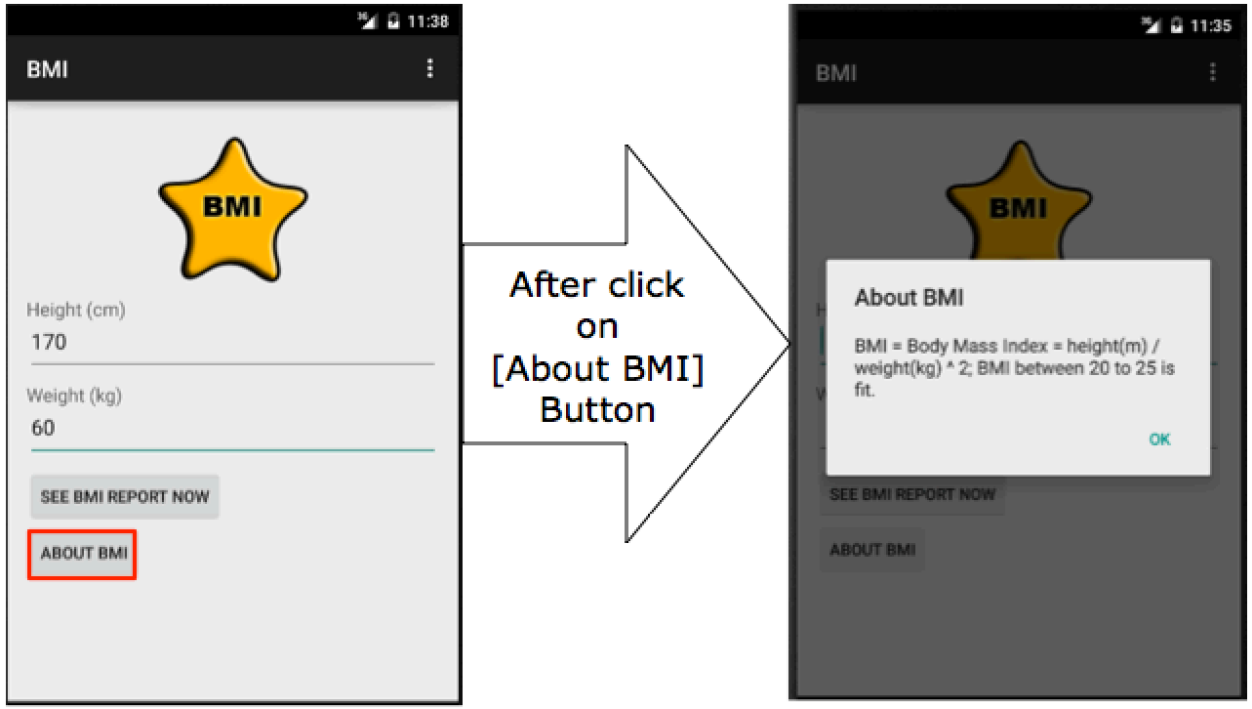
Linear Layout Approach

- You can use two Linear Layouts with orientation of horizontal and vertical to make ImageView positioned on the left side and all the TextViews, EditTexts and Button to posited on the right side.



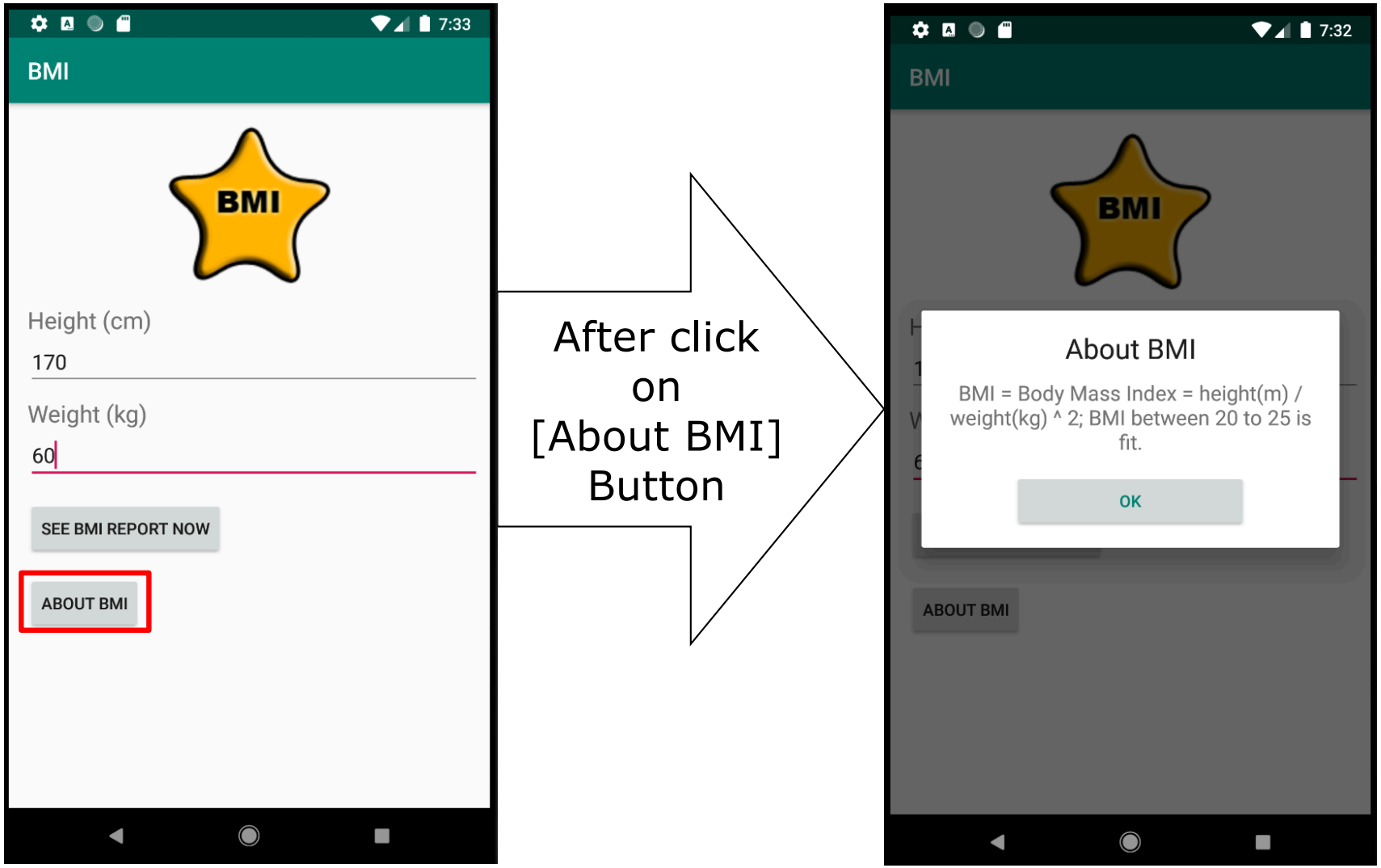
Android Dialog Notifications

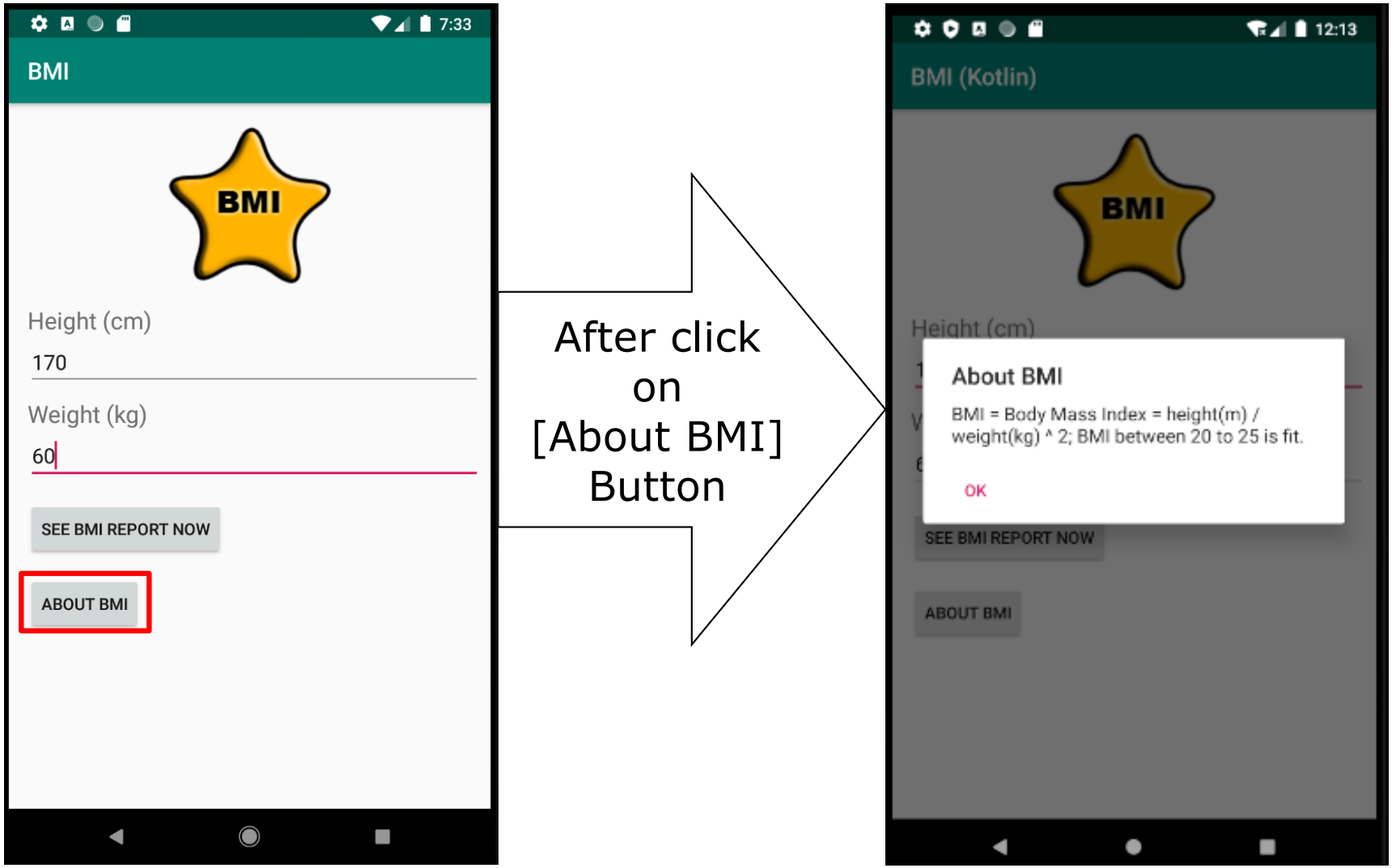
- **Modal** window
  - User cannot continue until dialog dismissed
- Window that pop up in front of current activity



Dialog Notifications Example

```
aboutButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
          AlertDialog.Builder builder =
new AlertDialog.Builder(MainActivity.this);
          builder.setTitle(R.string.about_button);
          builder.setMessage(R.string.about_msg);
          builder.setPositiveButton("OK",
new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which)
                { }
          });
          builder.create();
          builder.show();
        }
    });
```


7:33
BMI
BMI
Height (cm)
170
Weight (kg)
60
SEE BMI REPORT NOW
ABOUT BMI
After click on [About BMI] Button
7:32
BMI
BMI
About BMI
BMI = Body Mass Index = height(m) / weight(kg) ^ 2; BMI between 20 to 25 is fit.
OK
ABOUT BMI


7:33
BMI
BMI
Height (cm)
170
Weight (kg)
60
SEE BMI REPORT NOW
ABOUT BMI
After click on [About BMI] Button
12:13
BMI (Kotlin)
BMI
About BMI
BMI = Body Mass Index = height(m) / weight(kg) ^ 2; BMI between 20 to 25 is fit.
OK
SEE BMI REPORT NOW
ABOUT BMI

New Lab Exercise Arrangement

- Students are required to perform lab exercises on your PC or laptop.
- Revised Instructor Verification Sheet in MS-Words format are available in the schedule webpage.
- Students need to take screenshots to capture the results of the AVD (Android Virtual Device) and paste the captured images into a verification sheet to demonstrate the completion of the checkpoint.

EE5415 Lab03: Layouts and Widgets
Instructor Verification Sheet

(Submit this sheet to Lab Technician at the end of the Lab session)

Student Name: _____ Student Number: _____ Date: _____

Lab: (55 marks)

Check Point 1: [15 marks]

1.1 Show your LinearLayout Project with use of different weights.
1.2 Show your ConstraintLayout Project with buttons aligns to the left.
1.3 Show your TableLayout Project.

Capture the AVD screens of the above three results in the follow space:

Lab Submission

- The deadline for the lab submission is changed to next session. Thus, the Lab03 submission deadline is Wednesday of week 4.
- Submit the Lab Verification Sheet with captured AVD results and answers of the short questions in MS-Words or PDF format to ee5415@gmail.com with subject:
  - EE5415 Lab03 Submission: Student Name (Student Number)