Chapter 5.

Introduction to Android Apps Development

2024-2025
COMP7506 Smart Phone Apps Development
Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)
Department of Computer Science, School of Computing and Data Science, Faculty of Engineering, The University of Hong Kong

Agenda

- Overview of XML
- Introduction to Android Studio - "Hello World" Example
- Android Layout
- A Simple HCF Calculator Example
- Design Pattern
- Dynamic Layout

Overview of XML

Markup Languages

- Standard Generalized Markup Language (SGML) is a rich meta language that is useful for defining markup languages.
- HTML and XML are based on SGML.
- HTML is particularly useful for displaying Web pages (i.e., how to display the data).
- XML defines data structures for electronic commerce (i.e., what is the data).

HTML Example

```
<html>
    <head>
        <title>Here goes the title</title>
    </head>
    <body>
        <h1>This is a header</h1>
        Here goes the text of the page
    </body>
</html>
```

Tags mean something specific to the browser.

They are used for display.

XML

- eXtensible Markup Language
- Simple text (Unicode) underneath
- Tags (like in HTML) are used to provide information about the data
- Often used to store and transfer data
- Structured, human readable document format
  - Like html
  - Unlike postscript – structured, but not human readable
  - Unlike Java or C – structured, human readable, but not document
  - General format, not just by Android, e.g., web layout

XML Example

```
<?xml version="1.0"/>
<person>
    <name>
        <first>T.W.</first>
        <last>Chim</last>
    </name>
    <email>twchim@cs.hku.hk</email>
    <phone 28578272 />
</person>
```

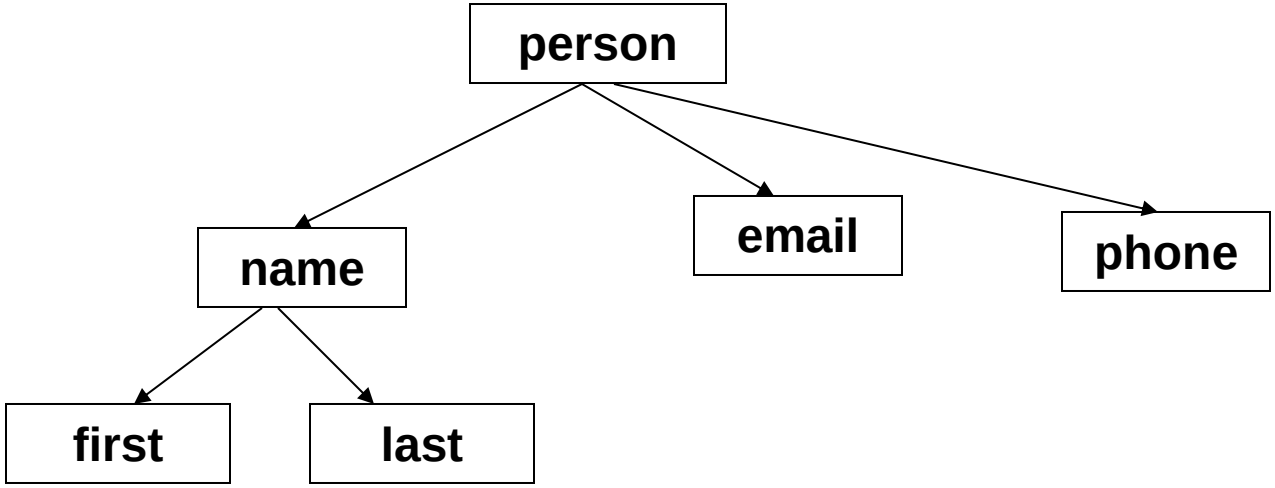
Tags mean whatever the user wants them to mean.

They are used to describe the data.

XML Rules

- Tags are enclosed in angle brackets.
- Tags come in pairs with start-tags and end-tags.
- Tags must be properly nested.
  - **<name><email>…</name></email>** is not allowed.
  - **<name><email>…</email></name>** is a must.
- Tags that do not have end-tags must be terminated by a ‘/’.
- Document has a single root element

XML Documents are Trees



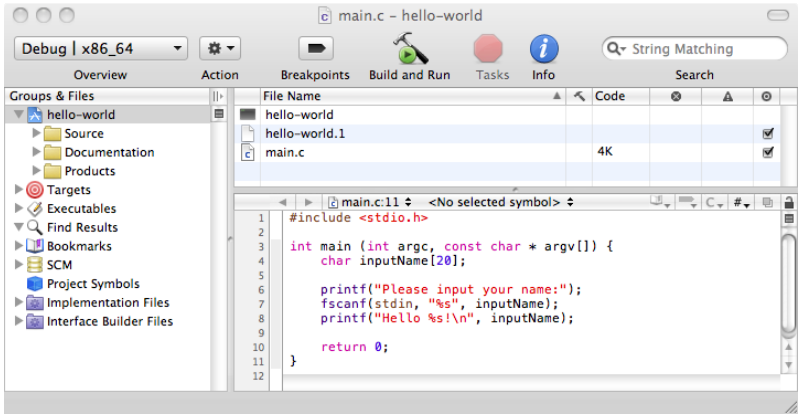
Android Manifest

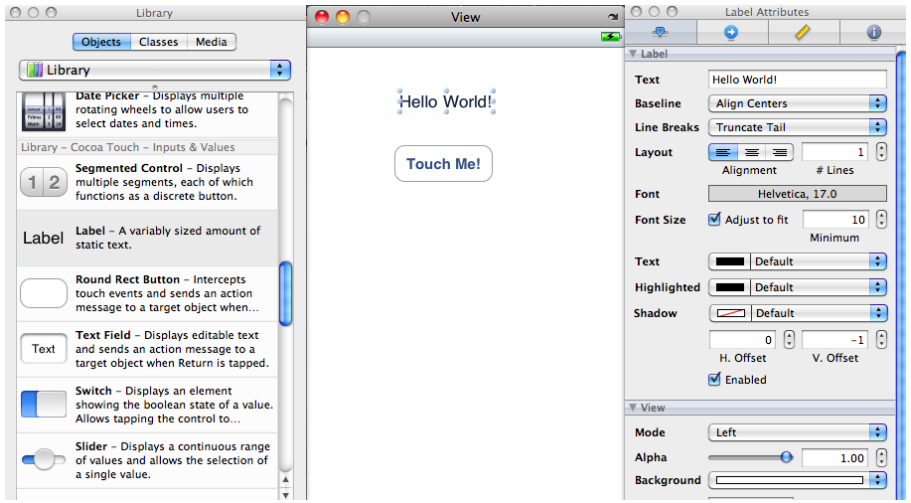
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.helloandroid"
      android:versionCode="1"
      android:versionName="1.0">
      <application android:icon="@drawable/icon" android:label="@string/app_name">
            <activity android:name=".HelloAndroid"
                        android:label="@string/app_name">
                        <intent-filter>
                              <action android:name="android.intent.action.MAIN" />
                              <category android:name="android.intent.category.LAUNCHER" />
                        </intent-filter>
            </activity>
      </application>
</manifest>
```

Attributes

Introduction to Android Studio – “Hello World” Example

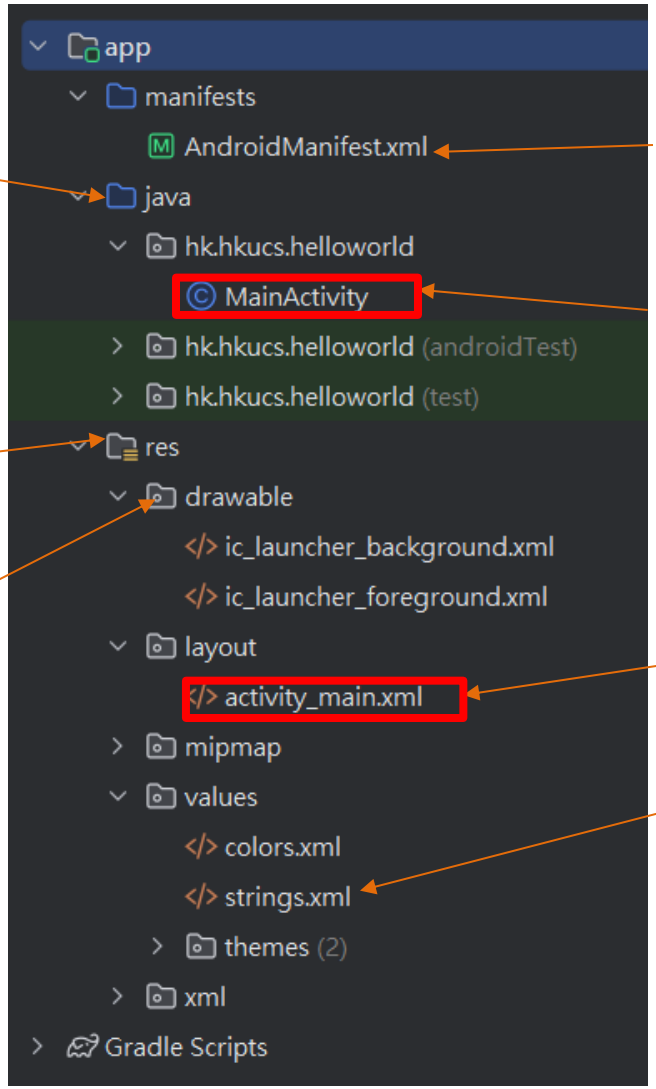
Console Programming vs. Mobile App Programming





- Console programming:
  - write the code
  - compile it
  - run and debug it
- Mobile app programming:
  - design the interface (i.e., screen layout of the application) (by code or with the help of interface builder)
  - work on code and interface alternately (similar to building web pages), define their interaction

Package Contents



All source code here

Android Manifest

Java / Kotlin code for our activity

All non-code resources

Images

Layout of the activity

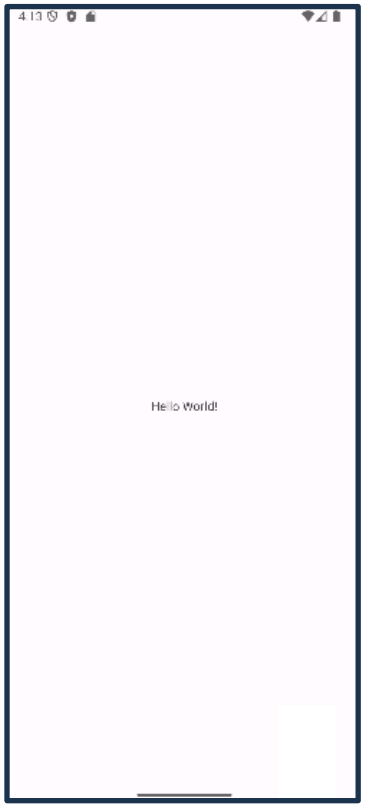
Strings used in the program

"Hello World" Example (*activity_main.xml*)

```
<?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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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



"Hello World" Example (*MainActivity.java*)

```
package hk.hkucs.helloworld;

import android.os.Bundle;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```

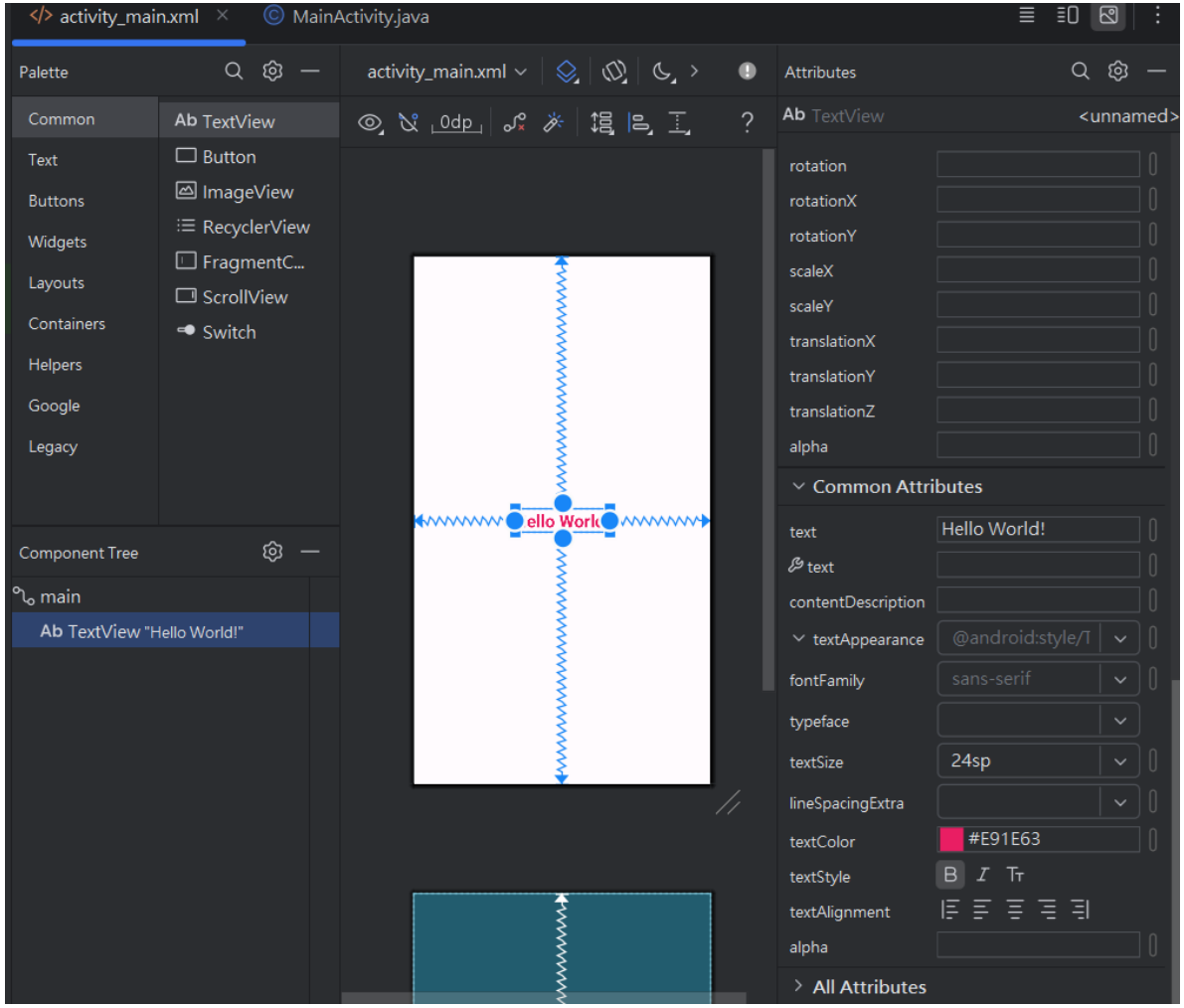
// R = resources of the project ➔
"R.layout.activity_main" =
"res/layout/activity_main.xml"

Display views behind the status and navigation bar of the device

Set the layout of the view as described in the activity_main.xml layout

Configure additional elements (system bars here) that are added to the window by the system

Configuring *textAppearance* in *Attributes*



Android Layout

XML Attributes

- ID
  - Needed to identify each view referenced in the Java code
  - e.g., under layout xml
    - android:id="@+id/my_button“
  - @: parse and expand
  - +: add to resource (i.e. R.java)
  - Under onCreate()
    - Button myButton = (Button) findViewById(R.id.my_button);
- Layout parameters
  - layout_*, e.g.
    - <Button android:id="@+id/my_button"
      android:layout_width=“fill_parent"
      android:layout_height="wrap_content"
      android:text="@string/my_button_text"/>
  - View class dependent

Common Layout

- FrameLayout
- LinearLayout
- RelativeLayout
- TableLayout
- ConstraintLayout

Although Constraint Layout is the default and the recommended layout, we will still introduce other layouts because:

1) **Lots of existing projects are still adopting other layouts.**
2) **Some layouts (e.g., Linear Layout) are easier to understand than Constraint Layout.**

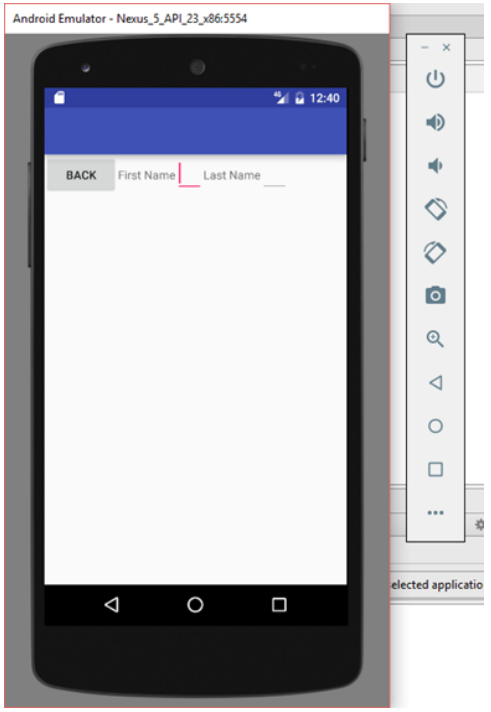
FrameLayout

- Display a single item at a time
- Elements are displayed overlapping


```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android">
        <ImageView
                android:src="@drawable/kpchow"
                android:scaleType="fitCenter"
                android:layout_height="fill_parent"
                android:layout_width="fill_parent"/>
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="24sp"
                android:textColor="#000000"
                android:gravity="center"
                android:text="Dr. Chow began his academic career in this department upon completion of his doctoral degree in the United States. His first contribution was on establishing HARNET, the first academic and research network for tertiary institutions in Hong Kong."/>
</FrameLayout>
```

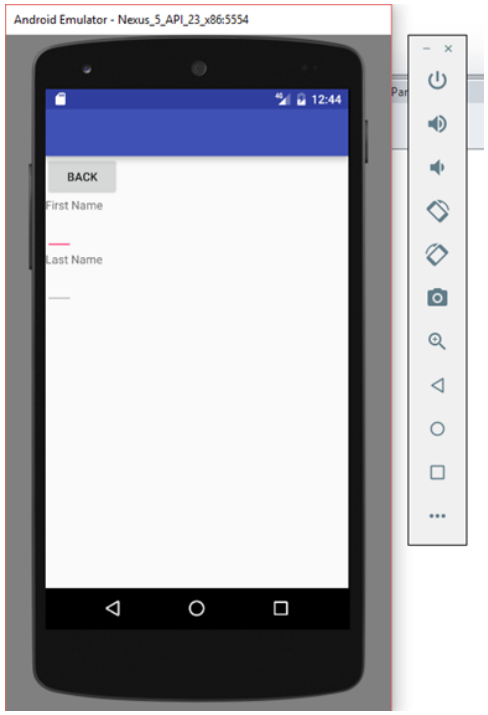
LinearLayout

- Organize elements along a single line
- Can be horizontal or vertical (by setting android:orientation)




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="hkucs.test3.MainActivity">
    <Button
        android:id="@+id/backbutton"
        android:text="Back"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:text="First Name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:width="100px"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:text="Last Name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:width="100px"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

LinearLayout




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:orientation="vertical"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="hkucs.test3.MainActivity">
   <Button
      android:id="@+id/backbutton"
      android:text="Back"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
   <TextView
      android:text="First Name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
      <EditText
         android:width="100px"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content" />
      <TextView
         android:text="Last Name"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content" />
      <EditText
         android:width="100px"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content" />
</LinearLayout>
```

RelativeLayout

- Lay out elements based on their relationships with one another, and with the parent container

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk
/res/android"
xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="hkucs.test5.MainActivity">
   <Button
      android:id="@+id/backbutton"
      android:text="Back"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />
   <TextView
      android:id="@+id/firstName"
      android:text="First Name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/backbutton"/>
   <EditText
      android:id="@+id/editFirstName"
      android:width="500px"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_toRightOf="@id/firstName"
      android:layout_below="@id/backbutton"/>
   <EditText
      android:id="@+id/editLastName"
      android:width="500px"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/editFirstName"
      android:layout_alignLeft="@id/editFirstName"/>
   <TextView
      android:id="@+id/lastName"
      android:text="Last Name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_toLeftOf="@id/editLastName"
      android:layout_below="@id/editFirstName"/>
</RelativeLayout>
```

RelativeLayout

- These properties will layout elements relative to the parent container.
  - android:layout_alignParentBottom – Places the bottom of the element on the bottom of the container
  - android:layout_alignParentLeft – Places the left of the element on the left side of the container
  - android:layout_alignParentRight – Places the right of the element on the right side of the container
  - android:layout_alignParentTop – Places the element at the top of the container
- These properties will layout elements relative to the parent container.
  - android:layout_centerHorizontal – Centers the element horizontally within its parent container
  - android:layout_centerInParent – Centers the element both horizontally and vertically within its container
  - android:layout_centerVertical – Centers the element vertically within its parent container

RelativeLayout

- These properties allow you to specify how elements are aligned in relation to other elements.
  - android:layout_above – Places the element above the specified element
  - android:layout_below – Places the element below the specified element
  - android:layout_toLeftOf – Places the element to the left of the specified element
  - android:layout_toRightOf – Places the element to the right of the specified element
- android:layout_alignBaseline – Aligns baseline of the new element with the baseline of the specified element
- android:layout_alignBottom – Aligns the bottom of new element in with the bottom of the specified element
- android:layout_alignLeft – Aligns left edge of the new element with the left edge of the specified element
- android:layout_alignRight – Aligns right edge of the new element with the right edge of the specified element
- android:layout_alignTop – Places top of the new element in alignment with the top of the specified element

TableLayout

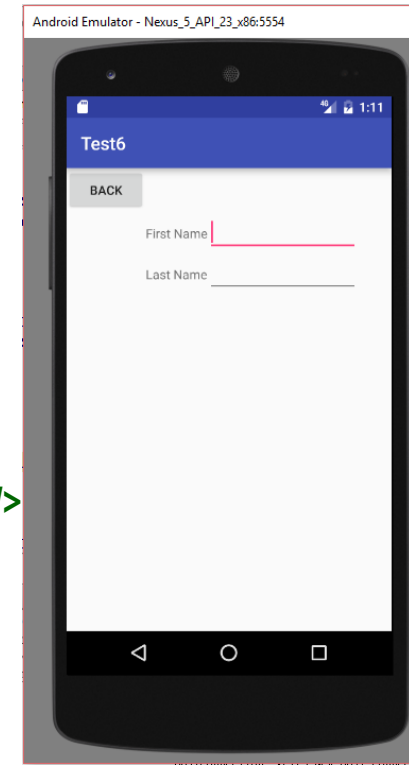
- TableLayout organizes content into rows and columns. The rows are defined in the layout XML, and the columns are determined automatically by Android. This is done by creating at least one column for each element. So, for example, if you had a row with two elements and a row with five elements then you would have a layout with two rows and five columns.
- You can specify that an element should occupy more than one column using android:layout_span. This can increase the total column count as well, so if we have a row with two elements and each element has android:layout_span=”3” then you will have at least six columns in your table.
- By default, Android places each element in the first unused column in the row. You can, however, specify the column an element should occupy using android:layout_column.

TableLayout

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context="hkucs.test6.MainActivity">
   <TableRow>
      <Button
         android:id="@+id/backbutton"
         android:text="Back"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content" />
   </TableRow>
```

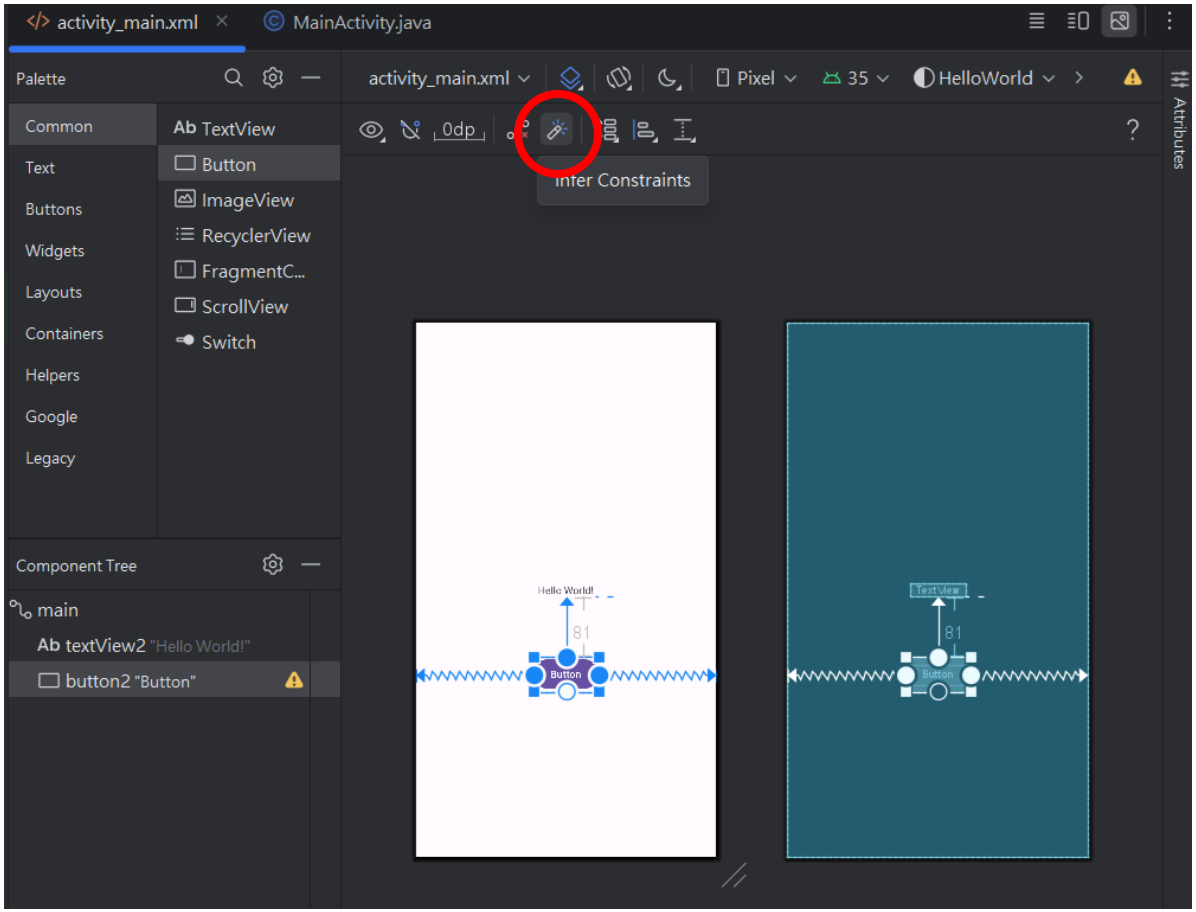
Challenge:
How many rows and columns are there in this table?
Answer:
3 rows and 3 columns

```
   <TableRow>
      <TextView
         android:text="First Name"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_column="1"/>
      <EditText
         android:width="500px"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"/>
   </TableRow>
   <TableRow>
      <TextView
         android:text="Last Name"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_column="1"/>
      <EditText
         android:width="500px"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"/>
   </TableRow>
</TableLayout>
```



ConstraintLayout

- Latest addition to Android's collection of layouts. Default layout in Android Studio 2.2 or above.
- You can drag the components you want one by one.
- Select the components, click the circled button ("Infer Constraints") to define their positions relative to the parent frame or to other components.



ConstraintLayout

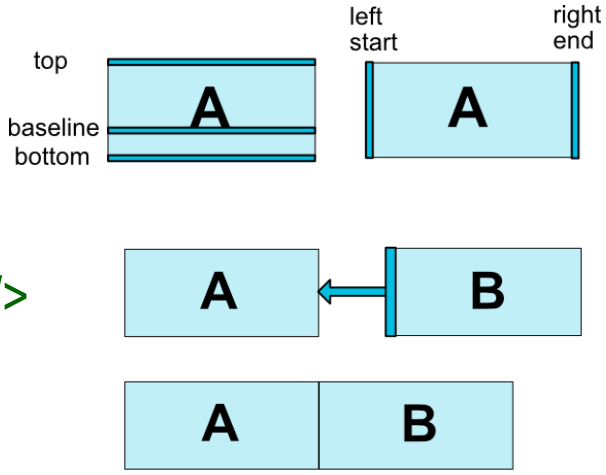
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="81dp"
        android:text="Button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

ConstraintLayout

- Relative positioning is one of the basic building block of creating layouts in ConstraintLayout. Those constraints allow you to position a given widget relative to another one.
- You can constrain a component on the horizontal and vertical axis:
  - Horizontal Axis: Left / Start, Right / End sides
  - Vertical Axis: top, bottom sides and text baseline

- Example:

```
<Button android:id="@+id/buttonA" ... />
<Button android:id="@+id/buttonB" ...
app:layout_constraintLeft_toRightOf="@+id/buttonA" />
```



ConstraintLayout

- List of available constraints:
  - layout_constraintLeft_toLeftOf
  - layout_constraintLeft_toRightOf
  - layout_constraintRight_toLeftOf
  - layout_constraintRight_toRightOf
  - layout_constraintTop_toTopOf
  - layout_constraintTop_toBottomOf
  - layout_constraintBottom_toTopOf
  - layout_constraintBottom_toBottomOf
  - layout_constraintBaseline_toBaselineOf
  - layout_constraintStart_toEndOf
  - layout_constraintStart_toStartOf
  - layout_constraintEnd_toStartOf
  - layout_constraintEnd_toEndOf

Think about when we will use layout_constraintLeft_toLeftOf and layout_constraintTop_toTopOf?

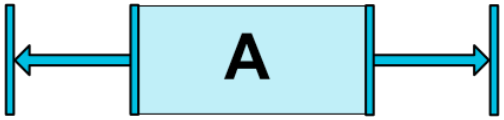
ConstraintLayout

- You can also set a margin between objects:
  - android:layout_marginStart
  - android:layout_marginEnd
  - android:layout_marginLeft
  - android:layout_marginTop
  - android:layout_marginRight
  - android:layout_marginBottom
- Note that a margin can only be positive or equals to zero, and takes a Dimension.
- In our previous example, we use margin to set a component to a certain position:
  - android:layout_marginStart="150dp“
  - android:layout_marginTop="64dp"

ConstraintLayout

- More examples:

```
<android.support.constraint.ConstraintLayout ...>
        <Button android:id="@+id/button" ...
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent/>
</>
```



Contradiction???

```
<android.support.constraint.ConstraintLayout ...>
        <Button android:id="@+id/button" ...
        app:layout_constraintHorizontal_bias="0.3"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent/>
</>
```

Please refer to the following site for more information:
https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html

ScrollView

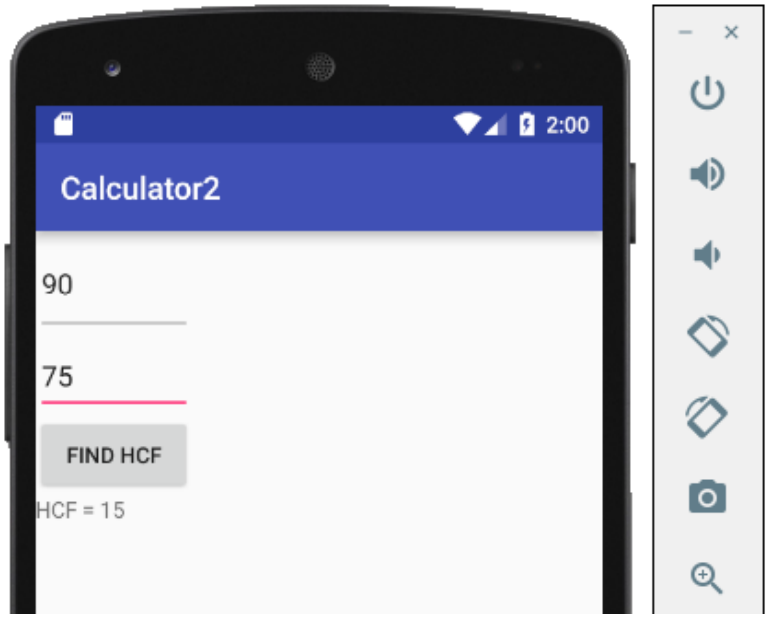
- Layout container for a view hierarchy that can be scrolled by the user, allowing it to be larger than the physical display.
- A ScrollView is a FrameLayout, meaning you should place one child in it containing the entire contents to scroll. This child may itself be a layout manager with a complex hierarchy of objects.

- Example:

```
<ScrollView
xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="fill_parent“
android:layout_height="fill_parent“
android:fillViewport="true">
        <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content“
        android:background="#ffffff">
        …
        </RelativeLayout>
</ScrollView>
```

A Simple HCF Calculator Example

Sample Output



Given 2 input numbers, the HCF of the 2 numbers will be calculated.

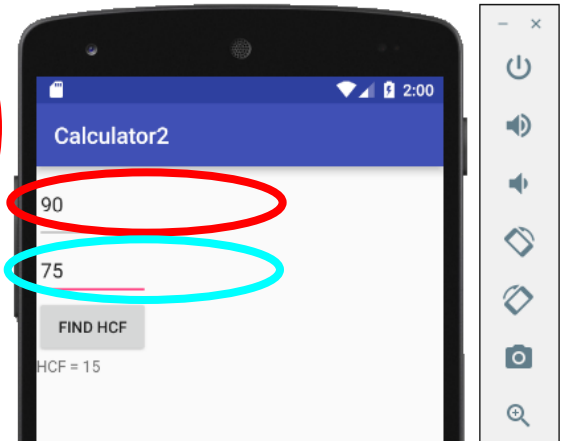
activity_main.xml

XML for the above UI

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context="hk.hkucs.hcfcalculator.MainActivity">
```

```
<EditText
android:id="@+id/txtbox2"
android:layout_width="300px"
android:layout_height="200px"
android:text=""
android:textSize="18sp">
</EditText>
```

```
<EditText
android:id="@+id/txtbox1"
android:layout_width="300px"
android:layout_height="200px"
android:text=""
android:textSize="18sp">
</EditText>
```

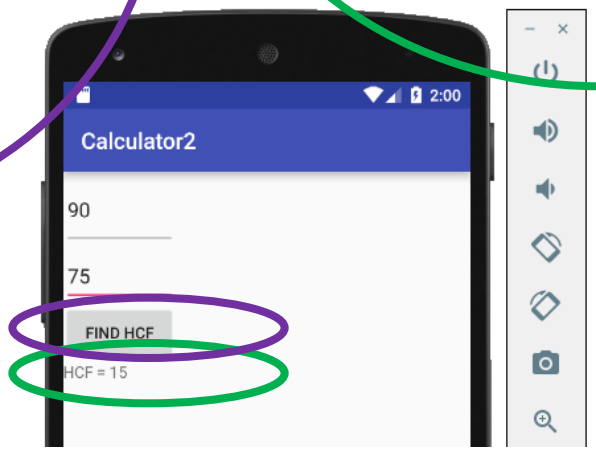


activity_main.xml

XML for the above UI

```
<Button
android:id="@+id/button1"
android:layout_width="300px"
android:layout_height="200px"
android:text="FIND HCF">
</Button>
```

```
<TextView
android:id="@+id/lbl1"
android:layout_width="300px"
android:layout_height="wrap_content"
android:text="">
</TextView>
</LinearLayout>
```



MainActivity.java (Design 1)

Java source code

```
package hk.hkucs.hcfcalculator;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button button1;
    EditText txtbox1,txtbox2;
    TextView tv;
```

- All Android classes are inherited from Activity (old) or AppCompatActivity (new) superclass.
- You must override the onCreate() methods in Activity / AppCompatActivity superclass.

MainActivity.java (Design 1)

Java source code

```
/** Called when the activity is first created. */
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);   ← inflate the layout xml
        txtbox1= (EditText) findViewById(R.id.txtbox1);
        button1 = (Button) findViewById(R.id.button1);
        tv = (TextView) findViewById(R.id.lbl1);
        txtbox2= (EditText) findViewById(R.id.txtbox2);
        button1.setOnClickListener(new clicker());
}
```

MainActivity.java (Design 1)

```
class clicker implements Button.OnClickListener
{
        public void onClick(View v)
        {
                String a,b;
                int x, y, hcf = 0;
                a = txtbox1.getText().toString();
                b = txtbox2.getText().toString();
                x = Integer.parseInt(a);
                y = Integer.parseInt(b);
                for(int i = 1; i <= x || i <= y; i++) {
                        if( x % i == 0 && y % i == 0 )
                                hcf = i;
                }
                tv.setText("HCF = " + String.valueOf(hcf));

        }
    }
}
```

Inner class

Integer.parseInt() → cast a text into an integer.

It seems this program is not very OO. Everything is included into one single class.

String.valueOf() → convert an integer into a string

MainActivity.java (Design 1)

```
class clicker implements Button.OnClickListener
{
        public void onClick(View v)
        {
                String a,b;
                int x, y, hcf = 0;
                a = txtbox1.getText().toString();
                b = txtbox2.getText().toString();
                x = Integer.parseInt(a);
                y = Integer.parseInt(b);
                for(int i = 1; i <= x || i <= y; i++) {
                        if( x % i == 0 && y % i == 0 )
                                hcf = i;
                }
                tv.setText("HCF = " + String.valueOf(hcf));

        }
}
}
```

Inner class

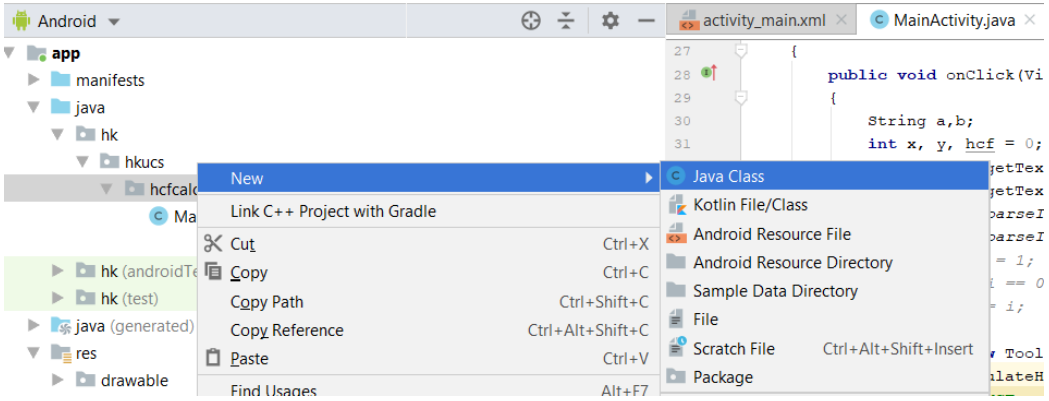
Integer.parseInt() → cast a text into an integer.

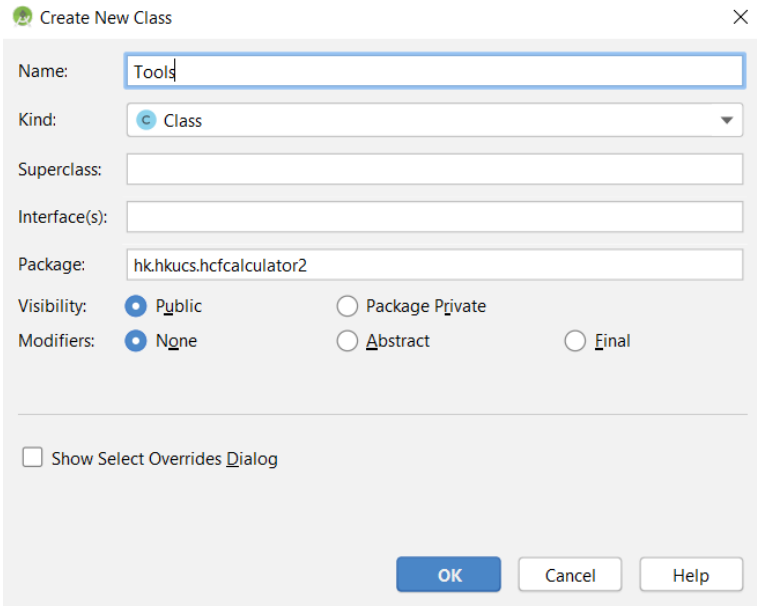
What if taking away this part and form a new class?

String.valueOf() → convert an integer into a string

Creating New Class in Android Studio Project

Right-click the folder containing MainActivity.java and choose “New → Java Class”.





Tools.java (Design 2)

All files in the project must have the same package name.

```
package hk.hkucs.hcfcalculator2;

public class Tools {
    private int num1, num2;
    public Tools(int n1, int n2) {
        num1 = n1;
        num2 = n2;
    }
    public int calculateHCF() {
        int hcf = 0;
        for(int i = 1; i <= num1 || i <= num2; i++) {
            if( num1 % i == 0 && num2 % i == 0 )
                hcf = i;
        }
        return hcf;
    }
}
```

MainActivity.java (Design 2)

Java source code

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button button1;
    EditText txtbox1,txtbox2;
    TextView tv;
```

- All Android classes are inherited from Activity (old) or AppCompatActivity (new) superclass.
- You must override the onCreate() methods in Activity / AppCompatActivity superclass.

MainActivity.java (Design 2)

Java source code

```
/** Called when the activity is first created. */
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtbox1= (EditText) findViewById(R.id.txtbox1);
        button1 = (Button) findViewById(R.id.button1);
        tv = (TextView) findViewById(R.id.lbl1);
        txtbox2= (EditText) findViewById(R.id.txtbox2);
        button1.setOnClickListener(new clicker());
}
```

MainActivity.java (Design 2)

```
class clicker implements Button.OnClickListener
{
        public void onClick(View v)
        {
                String a,b;
                int x, y, hcf = 0;
                a = txtbox1.getText().toString();
                b = txtbox2.getText().toString();
                x = Integer.parseInt(a);
                y = Integer.parseInt(b);
                Tools t = new Tools(x, y);
                hcf = t.calculateHCF();
                tv.setText("HCF = " + String.valueOf(hcf));
        }
}
}
```

Inner class

Integer.parseInt() → cast a text into an integer.

Tools.java handles the HCF calculation (based on x and y).

String.valueOf() → convert an integer into a string

Importance of Design 2

- The second design is more preferred because:
  - It is more object-oriented. Functions are implemented in different modules and can be tested separately.
  - It follows the Model-View-Controller (MVC) pattern (will be described soon).

Separate Testing of Tools.java

- We can actually take out Tools.java and use a tester program to test it separately.

Tools.java:

```
public class Tools {
    private int num1, num2;
    public Tools(int n1, int n2) {
        num1 = n1;
        num2 = n2;
    }
    public int calculateHCF() {
        int hcf = 0;
        for(int i = 1; i <= num1 || i <= num2; i++) {
            if( num1 % i == 0 && num2 % i == 0 )
                hcf = i;
        }
        return hcf;
    }
}
```

ToolsTest.java:

```
public class ToolsTest {
    public static void main(String[] args) {
        int x = 75;
        int y = 6;
        Tools t = new Tools(x, y);
        int hcf = t.calculateHCF();
        System.out.println(hcf);
    }
}
```

```
C:\Users\twchim\Desktop>javac *.java

C:\Users\twchim\Desktop>java ToolsTest
3

C:\Users\twchim\Desktop>
```

Design Pattern

What is Design Pattern?

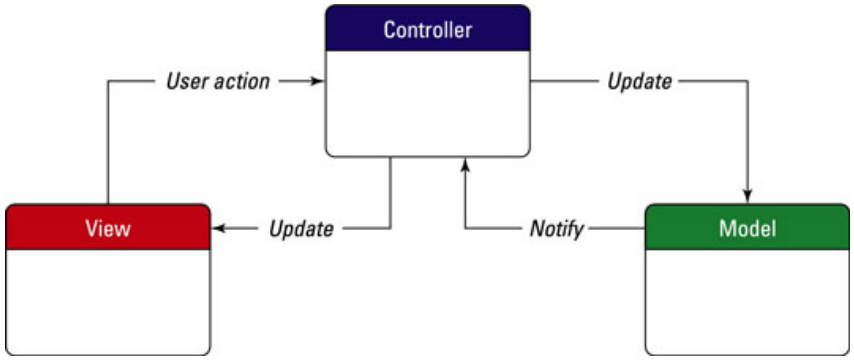
- A well-established way to group application functions into objects
- Variations of it have been around at least since the early days of Smalltalk, one of the very first object-oriented languages.
- A high-level pattern:
  - It addresses the architecture of an application and classifies objects according to the general roles they play in an application, rather than drilling down into specifics.
- Creates a miniature universe for the application, populated with distinct kinds of objects
- Specifies roles and responsibilities for each type of objects and specifies the way they're supposed to interact with each other.

An Analogy (MVC Design Pattern)

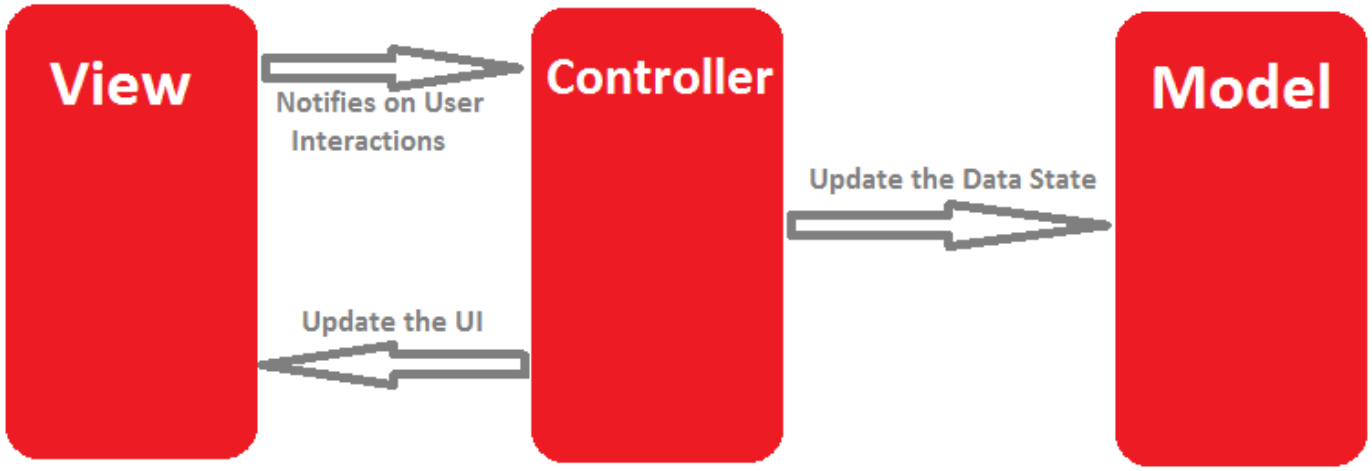
- Consider a big flat-screen television…
  - Model:
    - A particular television program (can in fact be played on different TV set)
  - View:
    - Television screen (can in fact show different television programs)
  - Controller:
    - The circuitry and signal that pull the show off the cable and then sends it to the screen

Model View Controller (MVC)

- Model:
  - Represents the data models
  - Manages the data states
  - Has business logics
- View:
  - The way we represent our data
    - e.g., views / layouts in Android
  - Renders the UI
- Controller:
  - Handles user interactions with our application
  - The communication channel between the model and the view
    - e.g., the fragments / activities in Android.

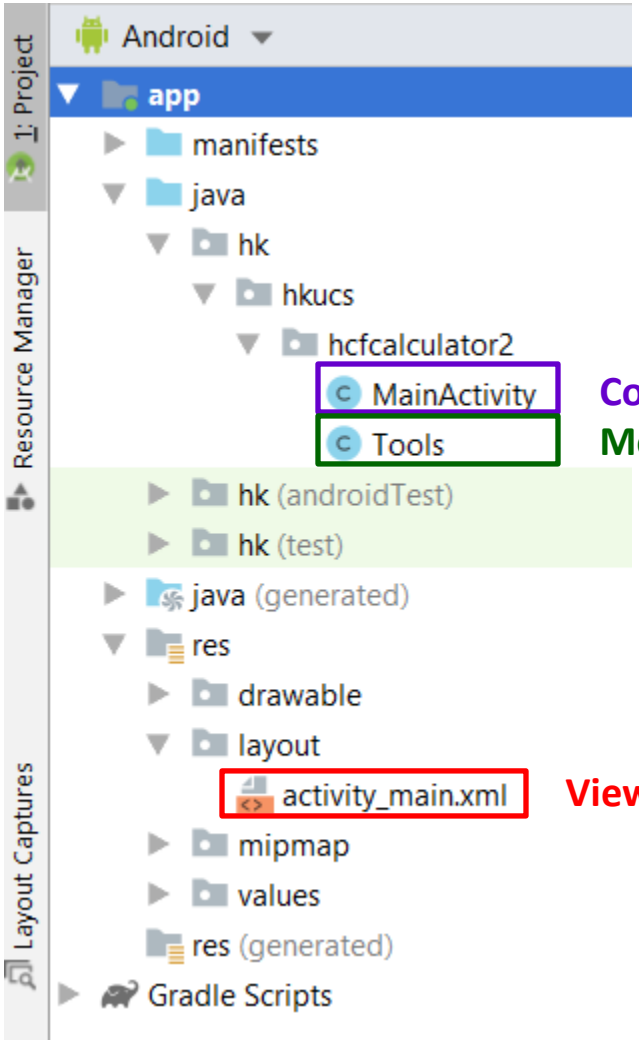


Model View Controller (MVC)



- The user interacts with the UI, and the controller gets notified via the view.
- Based on the User interaction the controller modifies certain Models.
- Models perform some business logic and return the updated model data state to the controller.
- The controller can then update the UI according to the new data state as received from Model.

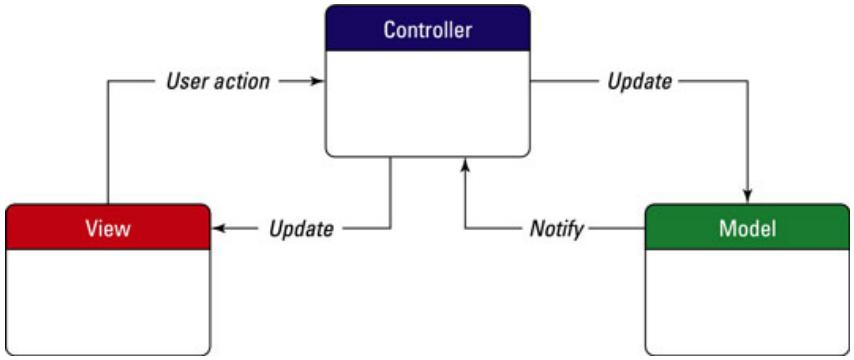
Back to our Design 2



Controller

Model

View



Model: Program logic

View: User Interface

Controller: Interconnection between view and model (components in UI ⇔ variables for calculation)

Other Design Patterns

- Model View Presenter (MVP)
- Model View ViewModel (MVVM)

Please refer to the site below for details:

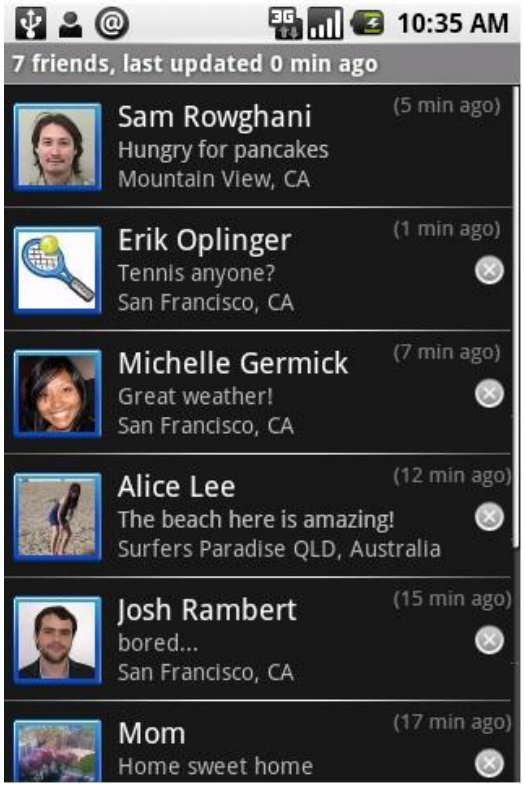
https://medium.com/@mr.anmolsehgal/common-android-architectures-mvc-vs-mvp-vs-mvvm-afd8461e1fee

Dynamic Layout

- The layout structure is first defined. Data is then inserted later dynamically.
- Especially useful for displaying dynamic data from server.

ListView

- A view that shows items in a vertically scrolling list. The items come from the ListAdapter associated with this view.
- Example:



ListView

- Displaying a list needs three elements:
  - ListView
  - Adapter: an interface for mapping the data into the ListView
  - Data: Strings, images, or even button
- There are THREE types of list according to the type of adapters:
  - ArrayAdapter

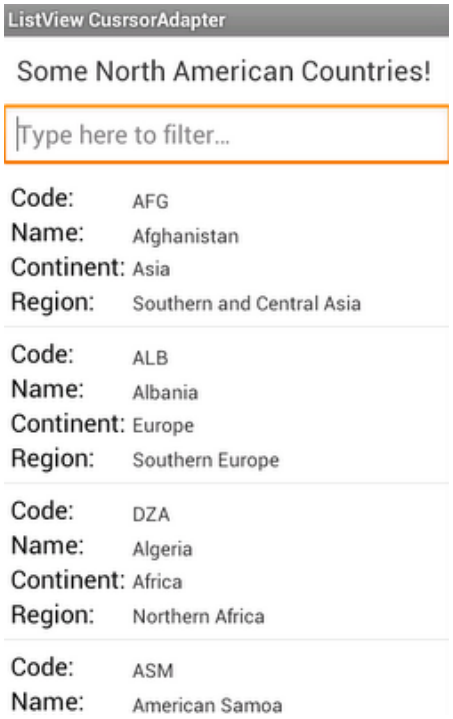
    The simplest list which can only show 1 line of words
  - SimpleAdapter

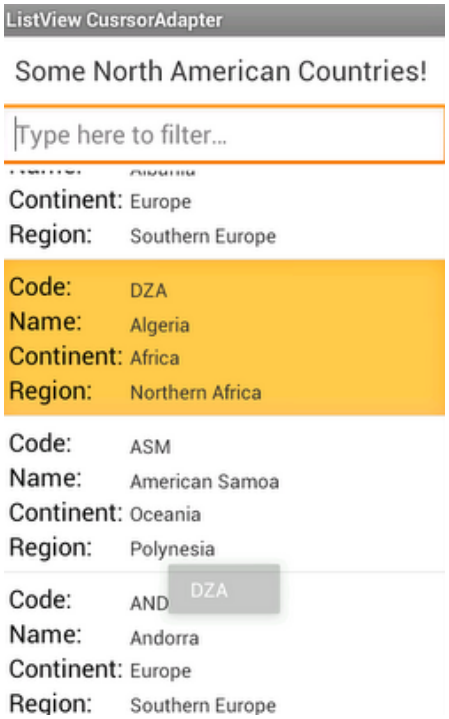
    It is more flexible. We can design various layout inside a row of the list.
  - SimpleCursorAdapter

    An easy adapter to map columns from a cursor to TextViews

ListView

- An example of SimpleCursor Adapter:





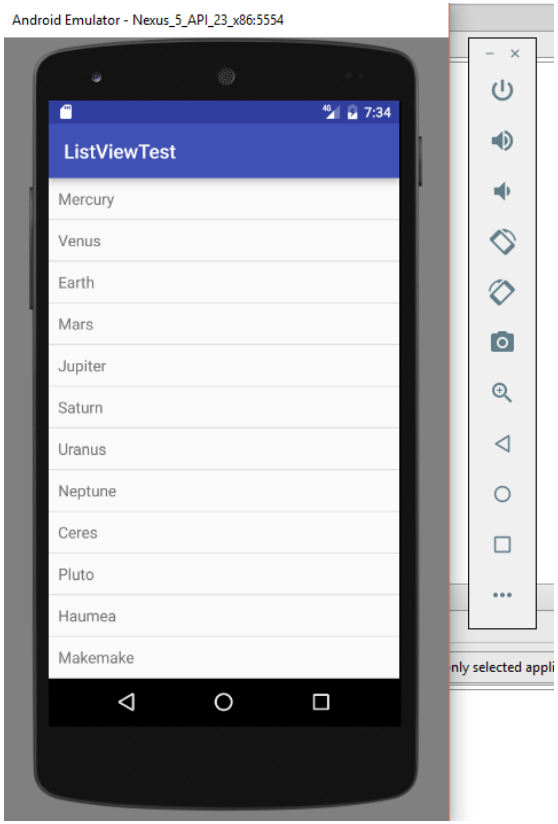
Reference: http://www.mysamplecode.com/2012/07/android-listview-cursoradapter-sqlite.html

ListView

- Example:

  Let's look at how to create a simple ListView and populate it with text data (the names of various planets).

  The following picture shows what the Android program looks like.



Note: ListView will never be used with ScrollView as ListView takes care of its own vertical scrolling.

Example (activity_main.xml)

- ListView is defined in the main layout file (res/layout/activity_main.xml) within a LinearLayout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical“ android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ListView android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/mainListView">
    </ListView>
</LinearLayout>
```

Example (simplerow.xml)

- Each row in the ListView will be a TextView. The TextView is defined in another file (res/layout/simplerow.xml).

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/rowTextView"  android:layout_width="fill_parent"
android:layout_height="wrap_content" android:padding="10dp"
android:textSize="16sp" >
</TextView>
```

- The resource ID of the ListView is mainListView, which we will use to get a reference to the ListView in our Activity class.

```
// Find the ListView resource.
mainListView = (ListView) findViewById( R.id.mainListView );
```

Example (MainActivity.java)

- Our main activity (SimpleListViewActivity) creates an ArrayAdapter, which holds the objects to be displayed in the ListView.
- The ArrayAdapter constructor passes the resource ID of the TextView layout file (R.layout.simplerow).
- The ArrayAdapter will use it to instantiate a TextView for each row.
- We then set the ArrayAdapter as our ListView's adapter.

Example (MainActivity.java)

```
package hkucs.listviewtest;

import java.util.ArrayList;
import java.util.Arrays;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends AppCompatActivity {

    private ListView mainListView ;
    private ArrayAdapter<String> listAdapter ;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // Find the ListView resource.
    mainListView = (ListView)
    findViewById( R.id.mainListView );

    // Create and populate a List of planet names.
    String[] planets = new String[] { "Mercury",
    "Venus", "Earth", "Mars", "Jupiter", "Saturn",
    "Uranus", "Neptune"};
    ArrayList<String> planetList = new
    ArrayList<String>();
    planetList.addAll( Arrays.asList(planets) );
```

Note the addAll() function.

Example (MainActivity.java)

```
        // Create ArrayAdapter using the planet list.
        listAdapter = new ArrayAdapter<String>(this, R.layout.simplerow, planetList);

        // Add more planets. If you passed a String[] instead of a List<String>
        // into the ArrayAdapter constructor, you must not add more items.
        // Otherwise an exception will occur.
        listAdapter.add( "Ceres" );
        listAdapter.add( "Pluto" );
        listAdapter.add( "Haumea" );
        listAdapter.add( "Makemake" );
        listAdapter.add( "Eris" );

        // Set the ArrayAdapter as the ListView's adapter.
        mainListView.setAdapter( listAdapter );
    }
}
```

Summary:
1.Data array ready
2.Data array → ArrayList
3.ArrayList → ArrayAdapter
4.ArrayAdapter → ListView

Chapter 5.

End

2024-2025
COMP7506 Smart Phone Apps Development
Dr. T.W. Chim (E-mail: twchim@cs.hku.hk)
Department of Computer Science, School of Computing and Data Science,
Faculty of Engineering, The University of Hong Kong