Android

User Interfaces

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Randy J. Fortier
randy.fortier@uoit.ca
@randy_fortier

Outline

- Configuration
  - The application manifest
  - Internationalization
- User interfaces
  - Activities
  - Views (buttons, etc.)
  - Layouts and view groups
  - Event handling

Android

Configuration

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Android Documentation

- To learn more about any Android component, check out the online documentation:

https://developer.android.com/reference/classes.html

- Layouts, views, classes (e.g. Activity), methods (e.g. onCreate())

The Application Manifest

- A manifest describes your mobile application, including:
  - The name of your main activity
  - Your application's version
  - Features required on the device (min. resolution, GPS)
  - The permissions required by your application
  - The minimum Android version required
  - The target Android version
  - Any additional libraries used
  - A list of messages to which your application will respond

Warning: XML Ahead!

- An Android manifest is an XML file
- One of many XML file formats used by Android
- We'll examine other such formats in this chapter

XML Primer

- XML has a syntax very similar to HTML
- Tags that are on their own:

```
<br />
```

- Tags that contain text:

```
<div></div>
```

- Tags that contain other tags:

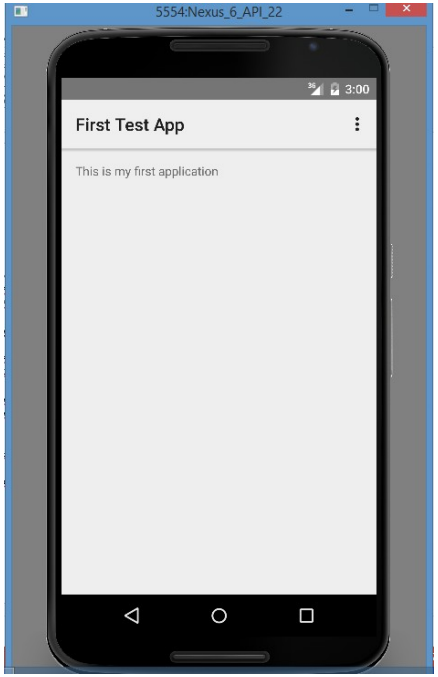
```
<table>
    <tbody>...</tbody>
</table>
```

- Attributes:

```
<img src="images/cat.jpg" alt="Some cute cat picture" />
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

A Sample Manifest




```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="ca.uoit.csci4100"
          android:versionCode="1"
          android:versionName="1.0">

    <uses-sdk android:minSdkVersion="22" />

    <application android:icon="@drawable/ic_launcher"
                 android:label="@string/app_name">

        <activity android:name=".SampleActivity"
                  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>
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Requiring Hardware Capabilities

```
<uses-feature android:name="android.hardware.nfc" />
```

- Other features:

–android.hardware.bluetooth
–android.hardware.camera
–android.hardware.camera.flash, android.hardware.camera.autofocus
–android.hardware.location
–android.hardware.location.gps, android.hardware.location.network
–android.hardware.sensor.accelerometer
–android.hardware.sensor.compass
–android.hardware.telephony
–android.hardware.type.television
–android.hardware.touchscreen, android.hardware.touchscreen.multitouch
–android.hardware.wifi

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Permissions

```
<uses-permission android:name="android.permission.CAMERA" />
```

- Used by Google Play to limit applications downloaded
- Other permissions:

–android.permission.READ_OWNER_DATA
–android.permission.CALL_EMERGENCY_NUMBERS
–android.permission.DEVICE_POWER
–android.permission.BLUETOOTH
–android.permission.ACCESS_COARSE_LOCATION
–android.permission.ACCESS_FINE_LOCATION
–android.permission.RECORD_AUDIO
–android.permission.CALL_PHONE
–android.permission.MODIFY_PHONE_STATE
–android.permission.PROCESS_OUTGOING_CALLS
–android.permission.READ_SMS, android.permission.WRITE_SMS
–android.permission.ACCESS_WIFI_STATE, android.permission.CHANGE_WIFI_STATE

i18n: Internationalization

- i18n involves making your application available in other languages
- Android has many useful features for i18n
  - Strings and other values are kept in separate (XML) files
  - Images can also be language-specific
  - Numbers, currency, dates, and times can easily be localized
    - i.e. drawn in the correct format for the user's locale

res/values/strings.xml

- A place where you can put all of your language-specific strings

– In small applications, it is acceptable to add arrays, and other values to this file

– However, for a large (production-ready) application, you should create a file for each type

res/values/strings.xml (English)

```
<resources>
  <string name="app_name">Contact Mate</string>
  <string name="first_name">First name</string>
  <string name="last_name">Last name</string>
  <string name="phone_num">Phone #</string>
  <string name="ok">Ok</string>
  <string name="cancel">Cancel</string>
</resources>
```

res/values/strings.xml (Spanish)

```
<resources>
  <string name="app_name">Contact Mate</string>
  <string name="first_name">Nombre</string>
  <string name="last_name">Apellido</string>
  <string name="phone_num">Numero de telefono</string>
  <string name="ok">Aceptar</string>
  <string name="cancel">Cancelar</string>
</resources>
```

res/values/arrays.xml

```
<resources>
  <string-array name="operations">
    <item>Search</item>
    <item>Edit</item>
    <item>Delete</item>
  </string-array>
</resources>
```

Android

User Interfaces

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

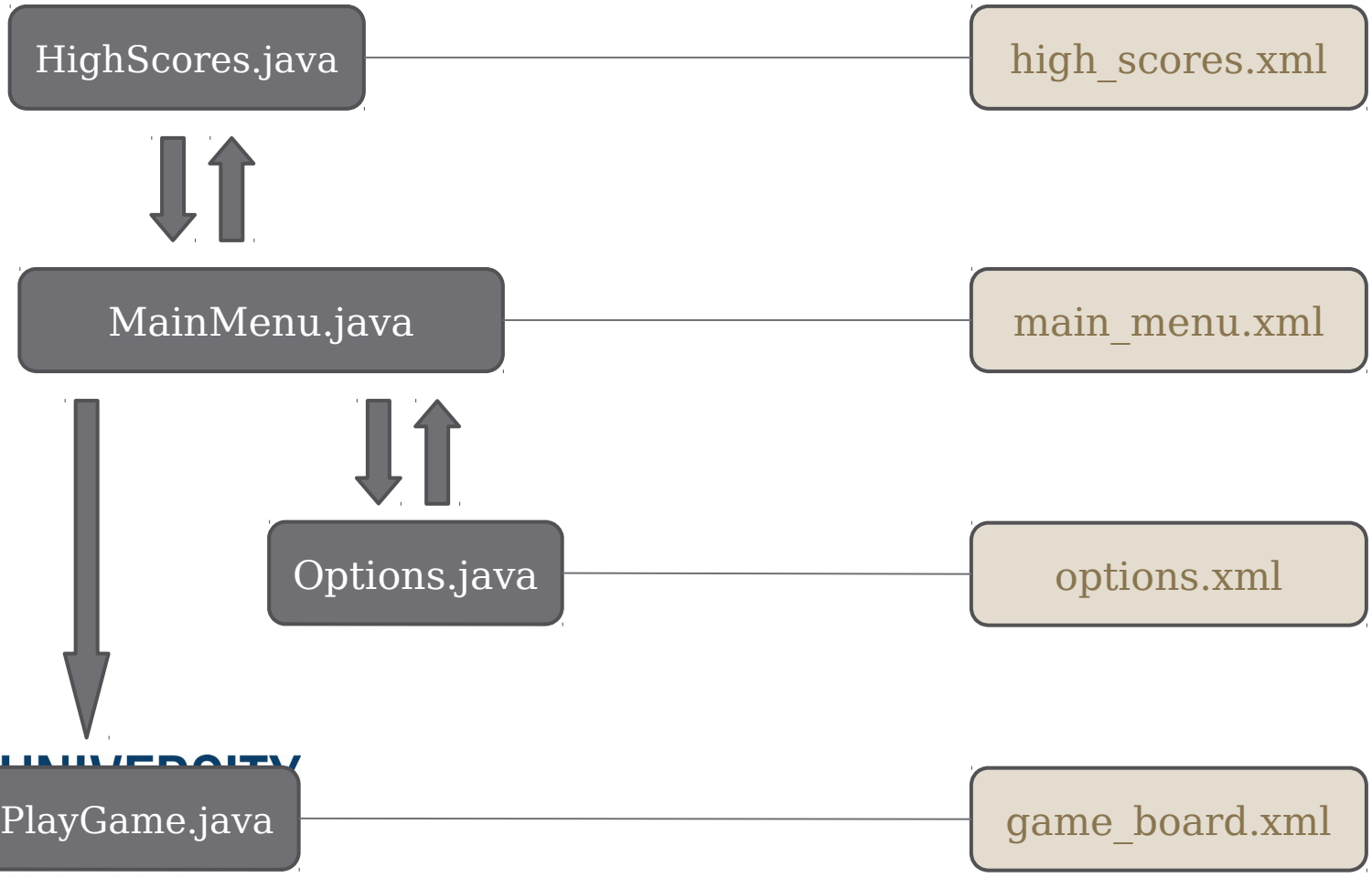
Activities

- In MVC terms, activities are controllers
  - They display the proper views (layouts)
  - They are not UIs
  - They handling user input (events)
  - They typically do not implement high-level application logic

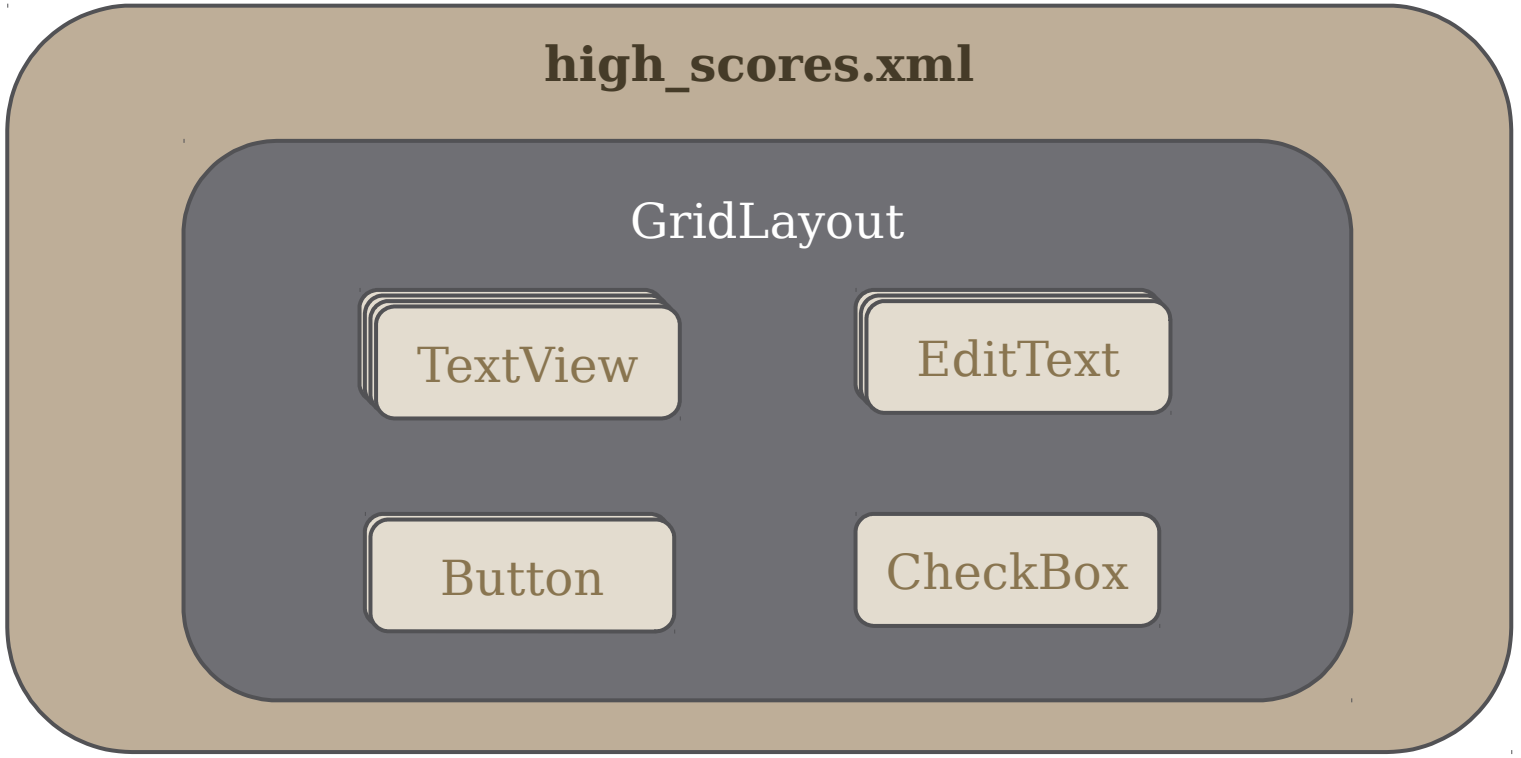
Activities and Layouts

- Generally, activities correspond (one-to-one) with layouts
  - … but not necessarily
- The activity often contains the code to:
  - Initialize the user interface
  - Handle events (e.g. button presses)
- The layout describes:
  - The arrangement of various UI components (views)
  - A list of views

Activities and Layouts

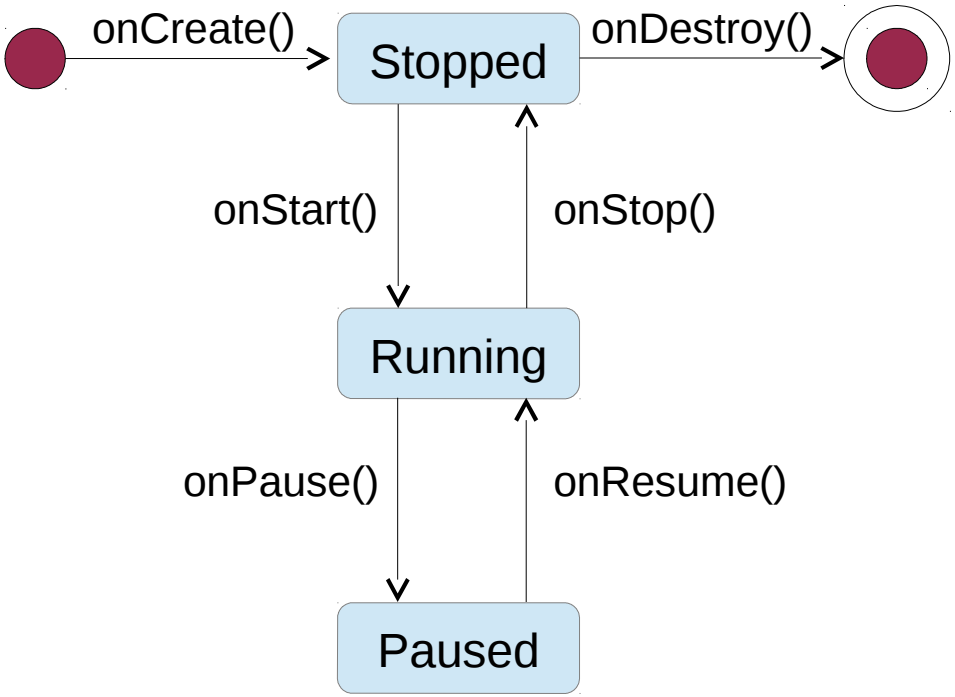


Views and Layouts



UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Simplified Activity Life Cycle



Activity Creation

- Activities are created by the Android OS in response to an intent
  - Intents are messages between activities, or between the OS and an activity
  - We'll discuss more about intents later
- When you click on the icon for an application, it triggers an intent to start an activity
  - The activity to start is specified in the manifest
  - After the activity has been created, the onCreate() method is called

Activity Destruction

- Android OS may decide to destroy your activity to clear some space
  - Write your activities so that they re-read their own previous state (Bundle)
  - This can happen when the activity is backgrounded, or when switching orientation

onCreate()

- Called when your activity is first created
- When created, your activity should:
  - Obtain any necessary resource locks
  - e.g. Database connections
- Although, popular wisdom may be wrong here:
  - It perhaps makes more sense to do this in onStart()
  - However, onCreate() is the only method that has access to the bundle of saved state
  - This bundle is not discussed now

onCreate()

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // show the correct view (layout)
    setContentView(R.layout.main);
}
```

onDestroy()

- Your activity is destroyed when it is unloaded from memory
- You generally should not need to do anything in onDestroy()
  - You have already freed all resources in onStop()(to be discussed later)

onPause()

- Called when another activity's layout partially covers your layout
  - e.g. A notification popup
- When paused, your activity should:
  - Stop consuming CPU (e.g. animations)
  - Release resources that use the battery, sound, etc. (e.g. GPS sensor)
  - Complete any actions (e.g. open files, partially completed database transactions)

onResume()

- Your activity is resumed when its layout becomes visible
  - You undo the precautions you took in onPause in this method
    - Resume animations
    - Re-obtain resources

onStop()

- Your activity is stopped when it is completely invisible
  - As it is in the background, it may get unloaded from memory
  - At this stage, it is important to release all resources that are difficult to obtain
    - e.g. Database connections, open files, network connections

onStart()

- Your activity is started (again) if it becomes visible again
  - At this stage, it is important to restore those resources you freed in onStop()

Example

```
public class SongIdentifyActivity extends Activity
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.song_identify);
    }
    public void onDestroy() {
    }
    public void onStart() {
        DbConnection = createNewDatabaseConnection();
    }
    public void onStop() {
        dbConnection.disconnect();
    }
    public void onPause() {
        pauseAudio();
    }
    public void onResume() {
        resumeAudio();
    }
}
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Views

- TextView
- EditText
- Button
- Checkbox
- Spinner
- ... and many more

TextView

- TextView is an uneditable label

```
<TextView
        android:id="@+id/lblFirstName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/firstName" />
```

EditText

- EditText is an editable text field

```
<EditText
        android:id="@+id/txtFirstName"
        android:layout_width="match_parent"
        android:capitalize="words"
        android:layout_height="wrap_content"
        android:hint="@string/firstNameHint" />
```

Button

```
<Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/login" />
```

CheckBox

- Checkboxes are good for boolean properties

```
<CheckBox android:id="@+id/cbxSmoking"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/smoking"
        android:onClick="onSmokingClicked"/>
```

Spinner

- A spinner is a weird name for a dropdown list

```
<Spinner
    android:id="@+id/lstCity"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Spinner Data

- A common way to put data into a spinner is to use an external array:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="cities">
        <item>Mercury</item>
        <item>Venus</item>
        ...
    </string-array>
</resources>

Spinner spinner = (Spinner)findViewById(R.id.lstCity);
ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.cities,
android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(
        android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

ListView

- A ListView is a scrollable list (not dropdown)
  - It works the same way as Spinner

```
<ListView   android:id="@+id/lstCity"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >
</ListView>
```

Spinner

- A spinner is a weird name for a dropdown list

```
<Spinner
    android:id="@+id/lstCity"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

Layouts

- Used in the XML layouts, these layouts (confusingly) are used to combine views together
  - LinearLayout
  - RelativeLayout
  - TableLayout
- Google now often refers to 'layouts' as view groups in documentation
- More view groups:
  - ListView
  - GridView

Layouts

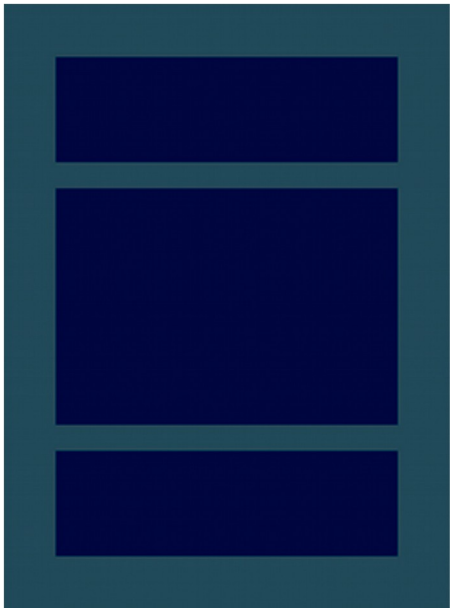
- View width and height:
  - wrap_content: Use only the necessary space
  - match_parent: As above (Android 2.2+)
  - 100dp: Use 100dp of space

Layouts

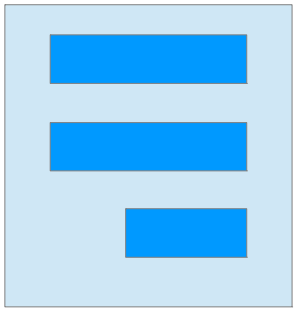
- Units
  - dp: Density independent pixels
    - Use for everything except for fonts
  - sp: scale-independent pixels
    - Use for fonts
- In general, do not use:
  - px: pixels
  - in: inches
  - mm: millimetres

LinearLayout

- Views are arranged either horizontally or vertically



LinearLayout



```
<LinearLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/firstName" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/lastName" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/submit" />
</LinearLayout>
```

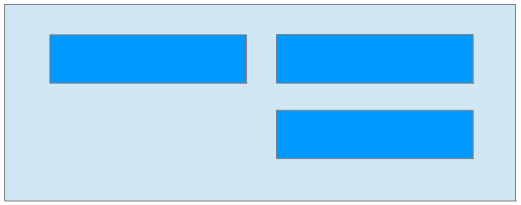
UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

RelativeLayout

- Views are arranged in positions relative to the parent
  - e.g. top, bottom, right, left
- Views are arranged in positions relative to their sibings

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

RelativeLayout

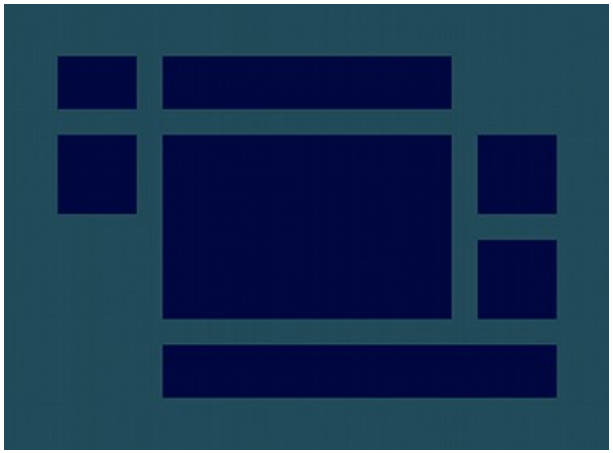


```
<RelativeLayout xmlns:android="..."
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <Spinner android:id="@+id/date"
             android:layout_width="0dp"
             android:layout_height="wrap_content"
             android:layout_below="@id/name"
             android:layout_alignParentLeft="true"
             android:layout_toLeftOf="@+id/time" />
    <Spinner android:id="@id/time"
             android:layout_width="96dp"
             android:layout_height="wrap_content"
             android:layout_below="@id/name"
             android:layout_alignParentRight="true" />
    <Button android:layout_width="96dp"
             android:layout_height="wrap_content"
             android:layout_below="@id/time"
             android:layout_alignParentRight="true"
             android:text="@string/done" />
</RelativeLayout>
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

TableLayout

- Views are positioned in rows and columns of variable size
  - You can specify zero or more cells to grow with extra width or height
- TableLayout acts a lot like an HTML table



TableLayout

```
<TableLayout android:layout_width="match_parent"
             android:layout_height="match_parent"
             xmlns:android="...">
  <TableRow>
    <TextView android:text="@string/firstName"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
    <EditText android:width="100dp"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content" />
  </TableRow>
  <TableRow>
    <TextView android:text="@string/lastName" ... />
    <EditText android:width="100dp" ... />
  </TableRow>
  <TableRow>
    <Button  android:text="@string/login"
             android:layout_column="1" ... />
  </TableRow>
</TableLayout>
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Event Handling: Buttons

```
public class MyActivity extends Activity
                        implements View.OnClickListener {
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);

        setContentView(R.layout.layout_with_button);

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

    public void onClick(View v) {
        // Handle the click somehow
    }
}
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Event Handling: Buttons

- Another way:

```
public class MyActivity extends Activity {
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);

        setContentView(R.layout.layout_with_button);

        Button btn = (Button)findViewById(R.id.btnOk);
        btn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // Handle the click somehow
            }
        });
    }
}
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Event Handling: Buttons

- Yet another way (recommended):

```
<Button android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:text="@string/ok"
        android:onClick="handleOk" />

 public class MyActivity extends Activity {
     ...

     public void handleOk(View v) {
         // Handle the click somehow
     }
 }
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Event Handling: Keys

```
public class MyActivity extends Activity {
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);

        setContentView(R.layout.layout_with_button);
    }

    @override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        switch (keyCode) {
          case KeyEvent.KEYCODE_MENU:
            HandleMenu();
            break;
        }
    }
}
```

Event Handling: List Selection

```
public class MyActivity extends Activity {
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);

        setContentView(R.layout.layout_with_button);

        ListView list = (ListView)findViewById(R.id.lstCity);
        list.setOnItemClickListener(new OnItemClickListener()
        {
           @Override
           public void onItemClick(AdapterView<?> parent,
                     View view, int position, long id) {
              // handleCitySelection();
           }
        });
 }
```

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Wrap-Up

- In this section, we learned about
  - Android configuration
  - Basic view components
  - Layout files
  - Layouts and view groups
  - Handling events