Distributed Systems HS2015 – Android Tutorial

Android Basics

Start a new Android Project	
• Configure your new project	Application Name: `Android Tutorial` (will be the name when managing applications) Company Domain: `vs.inf.ethz.ch` Package name: `ch.ethz.inf.vs.<nethz-login>.tutorial`
• Target Android Devices	Phone/Tablet > Minimum SDK: `API 18: Android 4.3 (Jelly Bean)`
• Add am activity to Mobile > Blank Activity	Activity Name: `MainActivity` Layout Name: `activity_main`
Project Structure > Modules > app (Alternatively, edit build.gradle (Module app)	
• Properties	• Compile Sdk Version: API 22: Android 5.1 (Lollipop) • Build Tools Version: 22.0.1
• Flavors	• Min Sdk Version: API 18: Android 4.3 (Jelly Bean) • Target Sdk Version: API 18: Android 4.3 (Jelly Bean)
• Dependencies	• com.android.support:appcompat-v7:22.2.0
Create virtual device: Nexus 5	
• Configure an AVD • Start emulator • Run as > Android Application	System Image: API 18. ABI: x86 Emulated Performance: Use Host GPU RAM: 768 MB SD Card: Studio-managed 20 MB
res/layout/activity_main.xml	
• Check frontend to add elements • Play with drop down menus	Screen sizes, orientation, API version
• Look at corresponding XML	Strings are referenced via identifiers `@string/<name>`
res/values/strings.xml	

• Use frontend to add new strings or edit XML
• *app_name* from "New Project"
	`strings.xml`
	`<string name="app_name">Android Tutorial</string>`

src/…/MainActivity.java	
• onCreate() • setContentView() • onCreateOptionsMenu()	State change handlers are @Override → **always remember to call super first**! The layout in `activity_main.xml` is set via constant in generated resource class R We do not need a menu now, let `onCreateOptionsMenu()` return **`false`**
AndroidManifest.xml	
• Look at XML	Intent-filter: defines first activity upon start (“main”) and that it shall appear in the apps launcher

Play with strings	

• Change `hello_world` in XML	`strings.xml`
	`<string name="hello_world">This is VS!</string>`

• Add automatic ID to TextView:
`@+id/text_main`
The + says “create an automatic ID”
• Change text via code in Main.java
	`layout/activity_main.xml`
	`android:id="@+id/text_main"`
	
	`MainActivity.java` `onCreate():`
	`TextView text = (TextView) findViewById(R.id.text_main);`
	`text.setText("I should not do it this way!");`

• Add new string to XML
• Update setText() to use string ID from R class
	`strings.xml`
	`<string name="welcome">That is the official way!</string>`
	
	`MainActivity.java`
	`text.setText(R.string.welcome);`

Debugging with "printf()"	

• Set breakpoint at different setText()
• Run debug
• Step through with F8 → no output
	`MainActivity.java`
	`text.setText(R.string.hello_world);`
	`text.setText(R.string.app_name);`
	`text.setText(R.string.welcome);`

Debugging with logcat	

• Use `android.util.Log` instead
VERBOSE > DEBUG > INFO > WARN > ERROR > ASSERT
• Put Log call after each `setText()`
• Create a LogCat filter on tag
• Replug phone and restart Eclipse if no output
	`MainActivity.java`
	`public static final String ACTIVITY_TAG = "### Main ###";`
	
	`Log.d(ACTIVITY_TAG, "1");`

Buttons and OnClick Listeners

Extend layout

- Change layout to LinerLayout (vertical)
- Add button `@+id/btn_test` "Click me"
- ID and string naming convention: [a-z0-9_] (general for Android-XML identifiers)

layout/activity_main.xml
```
<LinearLayout
    ...
    android:orientation="vertical"

<Button android:id="@+id/btn_test"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:text="@string/btn_click" />
```

strings.xml
```
<string name="btn_click">Click me</string>
```

Listener

- Add string `@string/btn_clicked` "Clicked"
- Implement onClickListener
- Quick & dirty
- Register OnClickListener

strings.xml
```
<string name="btn_clicked">Clicked</string>
```

MainActivity.java
```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener{
...
    private Button btn_test;
...
    protected void onCreate(Bundle savedInstanceState) {
    ...
        btn_test = (Button) findViewById(R.id.btn_test);
        btn_test.setOnClickListener(this);
    }
...
     @Override
     public void onClick(View v) {
          ((Button) v).setText(R.string.btn_clicked);
     }
}
```

- Add button `@+id/btn_action` “Action”
- Register OnClickListener

```
layout/activity_main.xml
<Button
        android:id="@+id/btn_action"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_click"/>

MainActivity.java
private Button btn_action;
...
protected void onCreate(Bundle savedInstanceState) {
    btn_action = (Button) findViewById(R.id.btn_action);
}
```

- Add branching with switch-case for individual actions

```
strings.xml
<string name="btn_running">Running</string>

MainActivity.java
@Override
public void onClick(View v) {
      switch (v.getId()) {
      case R.id.btn_test:
            ((Button)v).setText(R.string.btn_clicked);
            break;
      case R.id.btn_action:
            ((Button)v).setText(R.string.btn_running);
            break;
      }
}
```

XML linked Listener

- Add `android:onClick` to XML (since 1.6)
- Implement functions (depending on the name specified in `android:onClick`
- Remember to remove `setOnClickListener()`

`layout/activity_main.xml`

```
<Button android:onClick="onClickTest".../>
<Button android:onClick="onClickAction".../>
```

`MainActivity.java`

```
public void onClickTest(View v) {
      ((Button)v).setText(R.string.btn_clicked);
}

public void onClickAction(View v) {
      ((Button)v).setText(R.string.btn_running);
}
```

Other buttons

- Add ToggleButton *`@+id/btn_toggle`* “Stopped”
- Add string *`btn_stopped`*

`layout/activity_main.xml`

```
<ToggleButton android:id="@+id/btn_toggle"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/btn_stopped"
      android:onClick="onClickToggle" />
```

`strings.xml`

```
<string name="btn_stopped">Stopped</string>
```

`MainActivity.java`

```
public void onClickToggle(View v) {
      ToggleButton tb = (ToggleButton) v;
      if (tb.isChecked())
            tb.setText(R.string.btn_running);
      else
            tb.setText(R.string.btn_stopped);
}
```

- Initialize in onCreate()
- Note that some state is lost/overwritten when changing the orientation!
  → onResume() after orientation change

`MainActivity.java` onCreate():

```
((Button)findViewById(R.id.btn_toggle)).setText(R.string.btn_stopped);
```

Actuation and Permissions

New Activity, Intents

- Create new Activity: File > New > Activity > Blank Activity
  Name: ActuatorsActivity
  Layout: <automatic>
  Title: Actuators
  Hierarchical Parent: MainActivity
- Manifest entries are added by Eclipse
- Add string with HTML formatting
- Add Intent to launch new Activity

layout/activity_actuators.xml
```
<TextView
      android:id="@+id/txt_actuators"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:gravity="center_horizontal"
      android:text="@string/actuators" />
```

strings.xml
```
<string name="actuators">Actuators <tt>Activity</tt><br /><tt>TextView</tt>s
<i>understand</i> HTML <b>formatting</b>!</string>
```

MainActivity.java onClickTest():
```
Intent myIntent = new Intent(this, ActuatorsActivity.class);
this.startActivity(myIntent);
```

- Notice: no `<br />`, text style only
- Fix break with \n
- Play with back and home buttons
- Notice: App resumes last activity when launched from phone menu after home button was used

strings.xml
```
<string name="txt_actuators">Actuators <tt>Activity</tt><br /><tt>TextView</tt>s
<i>understand</i> HTML <b>formatting</b>!\n\nBut no HTML breaks</string>
```

Vibrator

- Add button `@+id/btn_vibrate` "Vibrate"
- Add and link onClickVibrate() method

ActuatorsActivity.java
```
public void onClickVibrate(View v) {
      Vibrator vib = (Vibrator) getSystemService(VIBRATOR_SERVICE);
      long[] pattern = { 0, 100, 100, 200, 100, 100 };
      vib.vibrate(pattern, -1);
}
```

- Run → crash → why?
- Add uses-permission to Manifest

AndroidManifest.xml
```
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>
```

Seekbar

- Add SeekBar to XML
- Make vib a member
- Anonymous inline implementation of OnSeekBarChangeListener (use anonymous classes only with care!)
- Keep pattern in onClickVibrate
- Add duration vibrate() to onStopSeek()
- Notice: setContentView() before findViewById()

```
layout/activity_actuators.xml
<SeekBar
      android:id="@+id/seek_duration"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:max="100"
      android:progress="50" />

ActuatorsActivity.java Members:
private Vibrator vib = null;
private int duration = 50;

ActuatorsActivity.java onCreate():
vib = (Vibrator) getSystemService(VIBRATOR_SERVICE);

SeekBar seekDuration = (SeekBar) findViewById(R.id.seek_duration);
seekDuration.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(...) {
        duration = progress;
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {}
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        vib.vibrate(duration*10);
    }
});
```

Media/Sound

- Add title TextViews “Sound” (paddingTop)
- Look up unit *dip*
- Add button *@+id/btn_sound* “Play”
- Implement and link onClickSound()
  Use `MediaPlayer`
- Add file sound.mp3 to `res/raw/` directory

```
layout/activity_actuators.xml
<TextView
      ...
      android:text="@string/sound"
      android:paddingTop="30dip" />

ActuatorsActivity.java
public void onClickSound(View v) {
      MediaPlayer mp = MediaPlayer.create(this, R.raw.sound);
      mp.setVolume(1.0f, 1.0f);
      mp.start();
}
```

- Change to looping player
- Make `mp` a member
- Add file loop.mp3 to `res/raw/` directory
- Check `isPlaying()` for action
- Reset player after stopping: prepareAsync()

```
ActuatorsActivity.java onCreate():
initPlayer();

ActuatorsActivity.java
private MediaPlayer mp = null;

private void initPlayer() {
      mp = MediaPlayer.create(this, R.raw.loop);
      mp.setLooping(true);
}

public void onClickSound(View v) {
      if (!mp.isPlaying()) {
            mp.start();
            if (mp.isLooping()) {
                  ((Button)v).setText(R.string.btn_running);
            }
      } else {
            mp.stop();
            try {
                  mp.prepareAsync();
            } catch (IllegalStateException e) {
                  // This is a demo. See Android policy on try/catch!
            }
            ((Button)v).setText(R.string.btn_sound);
      }
}
```

Menu button

- Replace/add items in actuators menu XML
  Options: looping, once, and back
- Add loop argument to `initPlayer()`
- Implement `onCreateOptionsMenu()`
- Implement `onOptionsItemSelected()`
  `finish()` ends Activity

menu/menu_actuators.xml

```xml
<item android:id="@+id/menu_looping"
      android:title="@string/menu_looping"
      android:orderInCategory="1" />
<item android:id="@+id/menu_once"
      android:title="@string/menu_once"
      android:orderInCategory="2" />
<item android:id="@+id/menu_back"
      android:title="@string/menu_back"
      android:orderInCategory="3" />
```

ActuatorsActivity.java

```java
private void initPlayer(boolean loop) {
      mp = MediaPlayer.create(this, loop ? R.raw.loop : R.raw.sound);
      mp.setVolume(1.0f, 1.0f);
      mp.setLooping(loop);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.actuators, menu);
      super.onCreateOptionsMenu(menu);
      if (mp.isPlaying()) return false; else return true; // saving space on paper
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
            case R.id.menu_looping:
                  initPlayer(true);
                  return true;
            case R.id.menu_once:
                  initPlayer(false);
                  return true;
            case R.id.menu_back:
                  finish();
                  return true;
            default:
                  return super.onOptionsItemSelected(item);
      }
}
```

Flashlight (optional as device-specific)

- Add title TextView “Flashlight” (paddingTop)
- Add ToggleButton *@+id/btn_flash* (no text)
- Add Camera member
- Implement and link onClickFlash()
- Add uses-permission
- Notice: works only since 2.2
- Some devices require `cam.setPreviewDisplay()` with `SurfaceTexture and/or SurfaceHolder` and `cam.startPreview();` e.g., Nexus 5

`layout/activity_actuators.xml`

```
<TextView
      ...
      android:text="@string/flashlight"
      android:paddingTop="30dip"/>
```

`ActuatorsActivity.java`

```
import android.hardware.Camera;
private Camera cam = null;

public void onClickFlash(View v) {
      ToggleButton tb = (ToggleButton) v;
      if (tb.isChecked()) {
            cam = Camera.open();
            Camera.Parameters parameters = cam.getParameters();
            parameters.setFlashMode(Camera.Parameters.FLASH_MODE_TORCH);
            cam.setParameters(parameters);
            cam.startPreview();
      } else {
            cam.release();
            cam = null;
      }
}
```

- Goes off or crashes when rotating screen: Add release to `onPause()`
- Display a `Toast`
- Also allows other apps to access camera when switching apps
- See transition diagrams from introduction

```
@Override
public void onPause() {
      super.onPause();

      if (cam!=null) {
            cam.release();
            cam = null;
            Toast.makeText(this, "Camera released", Toast.LENGTH_LONG).show();
      }
}

@Override
public void onResume() {
      super.onResume();

      ((ToggleButton)findViewById(R.id.btn_flash)).setChecked(false);
}
```

Async Task

AsyncTask

- Note: Do not do heavy processing in onCreate()
- Never do blocking I/O on UI/main thread
- Create new Activity: `WorkerActivity`
- Add ProgressBar: `@+id/progress_bar`
- Add id to TextView: `@+id/txt_progress`
- Extend AsyncTask<Integer, Integer, Void>
- Add string `@string/done` “Done.”
- Execute it in onCreate()
- Link activity to the action button in MainActivity
- Make sure to call `publishProgress()` when **updating the GUI** in `onProgressUpdate()`

```
MainActivity.java onClickAction():
Intent myIntent = new Intent(this, ActuatorsActivity.class);
this.startActivity(myIntent);

layout/activity_worker.xml
<ProgressBar
      android:id="@+id/progress_bar"
      style="?android:attr/progressBarStyleHorizontal"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="50dp" />

WorkerActivity.java
public class WorkerActivity extends Activity {
    private ProgressBar progress;
    private TextView    textview;

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

        progress = (ProgressBar)findViewById(R.id.progress_bar);
        textview = (TextView)findViewById(R.id.txt_progress);

        new MyWorker().execute(20);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        return false;
    }


...
```

```
      class MyWorker extends AsyncTask<Integer, Integer, Void> {

          private int index;

          @Override
          protected void onPreExecute() {
                  progress.setMax(100);
                  progress.setProgress(0);
          }

          @Override
          protected Void doInBackground(Integer... step) {
                  for (int i = 0; I < 100 / step[0]; ++i) {
                        try {
                              Thread.sleep(500);
                              index += step[0];
                        } catch (InterruptedException e) { }
                        publishProgress(step); // run onProgressUpdate on UI thread
                  }
                  return null;
          }

          @Override
          protected void onProgressUpdate(final Integer... values) {
              textview.setText(Integer.toString(index));
              progress.incrementProgressBy(values[0]);
          }

          @Override
          protected void onPostExecute(final Void result) {
              textview.setText(R.string.done);
          }
      }
}
```