Distributed Systems HS2012 – Android live hacking

General hints

- Uninstall Application when switching to a different development computer
- Often no connection to debugger on emulator → restart emulator
- Change emulator screen orientation with Ctrl+F11

File > New > Other…	
• Create "Android Application Project"	Application Name: `Live Hacking Demo` (the name when managing applications) Project Name: `LiveHacking` Package Name: `ch.ethz.inf.vs.android.<nethz-login>.livehacking` Build SDK: `Android 2.3.3 (API 10)` Minimum Required SDK: same (lower requires extensive testing, as unchecked by compiler)
• Create Blank Activity	Activity Name: `MainActivity` Title: `Live Hacking Demo` (also the name under the icon) Navigation Type: `none` (uniform types only introduced in API 14)
res/layout/activity_main.xml	
• Frontend (nice by now) • 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	
• Frontend • `app_name` from "Create project"	`strings.xml` `<string name="app_name">Live Hacking Demo</string>`
src/Main.java	
• onCreate() • setContentView() • onCreateOptionsMenu()	State change handlers are @Override → always remember to call super first! The layout in `activity_main.xml` is set via generated resource class R We do not need a menu, let `onCreateOptionsMenu()` return `false`
gen/R.java	
• Classes for ids, layouts, strings	Content of `res` folder is represented as integer handles
Manifest	
• Change uses-sdk versions to 10 • See other XML nodes	Wizard does not handle target/minimum API version correctly Intent-filter: define first activity upon start and it shall appear in the launcher
Create virtual device	
• Configure an AVD • Start emulator • Run app	SD Card: 16 RAM: 576 Configure camera facing back: emulated

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

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

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

Debugging with "printf()"	

• Set breakpoint before several setText()
• Run debug
• Step through with F6 → no output
	`MainActivity.java`
	`text.setText(R.string.hello_world); // <Ctrl+Shift+B>`
	`text.setText(R.string.title_activity_main);`
	`text.setText(R.string.welcome);`

Debugging	

• Use `android.util.Log` instead
• Create a Log Cat filter (green +)
	Levels: VERBOSE > DEBUG > INFO > WARN > ERROR > ASSERT
	
	`MainActivity.java`
	`Log.d("### Main ###", "1");`

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>`

Inline Listener	

• Add string *@string/btn_clicked* "Clicked"
• Add inline on-the-fly Listener
• Quick & dirty
• Multiple per class possible
	MainActivity.java
	`findViewById(R.id.btn_test).setOnClickListener( new OnClickListener() {`
	`  @Override`
	`  public void onClick(View v) {`
	`  ((Button)v).setText(R.string.btn_clicked);`
	`  }`
	`} );`

• Add button *@+id/btn_action* "Action"
• Store Listener in variable
• Assign to both buttons
• For reuse
	MainActivity.java
	`OnClickListener btnListener = new OnClickListener() {`
	`  @Override`
	`  public void onClick(View v) {`
	`  ((Button)v).setText(R.string.btn_clicked);`
	`  }`
	`};`
	`findViewById(R.id.btn_test).setOnClickListener(btnListener);`
	`findViewById(R.id.btn_action).setOnClickListener(btnListener);`

• Add string *@string/btn_running*"Running"
• Add branching with switch()-case for individual action
	MainActivity.java onClick():
	`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;`
	`}`

• Use `implements` Listener (with branching)
• Change `setOnClickListener(this);`
• Reusable
• Compact
• Centralized
• Only one listener per class
	MainActivity.java
	`public class Main extends Activity implements OnClickListener {`
	
	`@Override`
	`public void onClick(View v) {`
	`  switch (v.getId()) {`
	`  case R.id.btn_test:`
	`  ((Button)v).setText(R.string.btn_clicked);`
	`  ((Button)v).append(" (this)");`
	`  break;`
	`  case R.id.btn_action:`
	`  ((Button)v).setText R.string.btn_running);`
	`  ((Button)v).append(" (this)");`
	`  break;`
	`  }`
	`}`

XML linked Listener

- Add `android:onClick` to XML (since 1.6)
- Implement functions
- Remember to change `setOnClickListener()`
- Convenient

layout/activity_main.xml

```
android:onClick="onClickButton"
android:onClick="onClickAction"
```

MainActivity.java

```
public void onClickButton(View v) {
	((Button)v).setText(R.string.btn_clicked);
	((Button)v).append(" (XML)");
}

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

Other buttons

- Add ToggleButton *@+id/btn_toggle* "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"
	/>
```

- `android:text` not supported
- 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);
```

MainActivity.java

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

New Activity, Intents

- Create new Activity: `ActuatorsActivity`
  Hierarchical Parent: `MainActivity`
  Title: `Actuators`
- Manifest entries are added by Eclipse
- Remove ActionBar related code
- Add string with HTML formatting
- Add Intent to launch new Activity

`ActuatorsActivity.java`

```
package ch.ethz.inf.vs.android.<nethz-login>.livehacking;

import android.app.Activity;
import android.os.Bundle;

public class ActuatorsActivity extends Activity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.actuators);
	}
}
```

`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`

```
public void onClickButton(View v) {
	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
- Second argument:
  Index from where to start to repeat!
  Not how often.

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

Manifest

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

Seekbar

- Add SeekBar to XML
- Make vib a member
- Add inline Listener
- 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() {
            public void onProgressChanged(SeekBar seekBar,
                                          int progress,
                                          boolean fromUser) {
                  duration = progress;
            }
            public void onStartTrackingTouch(SeekBar seekBar) {}
            public void onStopTrackingTouch(SeekBar seekBar) {
                  vib.vibrate(duration*10);
            }
      });
```

Media/Sound

- Add title TextViews “Flashlight” (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 onPrepareOptionsMenu()
- Implement onOptionsItemSelected()
  `finish()` ends Activity

menu/activity_actuators.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

```
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 onPrepareOptionsMenu(Menu menu) {
      super.onPrepareOptionsMenu(menu);
      if (mp.isPlaying()) return false else return true; // saving space on paper
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
            case LOOPING_MENU_ID:
                  initPlayer(true);
                  return true;
            case ONCE_MENU_ID:
                  initPlayer(false);
                  return true;
            case BACK_MENU_ID:
                  finish();
                  return true;
            default:
                  return super.onOptionsItemSelected(item);
      }
}
```

Using different Streams (optional as device-specific)	

• Use the alarm stream
• Set permissions: *MODIFY_AUDIO_SETTINGS*
	ActuatorsActivity.java initPlayer():
	`AudioManager am = (AudioManager) getSystemService(Context.AUDIO_SERVICE);`
	`am.setMode(AudioManager.MODE_NORMAL);`
	
	`AssetFileDescriptor afd;`
	`if (loop) {`
	`  afd = getResources().openRawResourceFd(R.raw.sound);`
	`} else {`
	`  afd = getResources().openRawResourceFd(R.raw.sound);`
	`}`
	
	`try {`
	`  mp = new MediaPlayer();`
	`  mp.setDataSource(afd.getFileDescriptor(),`
	`  afd.getStartOffset(), afd.getLength());`
	`  mp.setAudioStreamType(AudioManager.STREAM_ALARM);`
	`  mp.prepare();`
	`  mp.setVolume(1.0f, 1.0f);`
	`} catch (IllegalArgumentException e) {`
	`  // See Android policy on try/catch!`
	`} catch (IllegalStateException e) {`
	
	`} catch (IOException e) {`
	
	`}`

Flashlight (optional as very device-specific)

- Add title TextViews “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 `SurfaceView` and `SurfaceHolder` and `cam.startPreview();`
  e.g., Nexus S with Android 4.1

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);
      } else {
            cam.release();
      }
}
```

- Still crashes when rotatiting 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);
}
```

AsyncTask

- Note: Do not do heavy processing in onCreate()
- Create new Activity: `WorkerActivity`
- Add ProgressBar: *`@+id/progress_bar`*
- Add id to TextView: *`@+id/txt_progress`*
- Extend AsyncTask<Input, Progress, Result>
- Execute it in onCreate()

`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 {

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

            private int index;
            private final ProgressBar progress;
            private final TextView textview;

            public MyWorker(final ProgressBar bar,
                        final TextView text) {
                  this.progress = bar;
                  this.textview = text;
            }

            @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);
                  }
                  return null;
            }
```

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

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


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


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

        new MyWorker(this, progress, textview).execute(20);

    }

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