**Bareeva, Julia**

Q1: Replace the circle login app with a much complicated one.

A1:

Q2: Show like Tinder, Lovoo is working. How about indoor?

A2:

**Belyakov, Alexey**

- !!!!!!!!!!!!

**Derbasov, Maksim**

Q1: Debugging multithreading.

A1: Diagnose memory usage via:

* adb shell procrank
* java.lang.Runtime
* Log messages
* Heap dumps and Eclipse Memory Analyzer (MAT)

More info on <http://de.slideshare.net/tarasleskiv/android-memory-fundamentals>

Q2: 2/3 cores debugging.

A2:

Q3: Native library used in Java (JNI).

A3: In the course materials (slide 200) we have a complete example *on “Exposing our Native Library via Java (i.e. JNI)”*. Additionally in the ANDROID/NDK/SAMPLES folder we have some examples on how the communication is done. The theoretical part into the course is located at slide 54.

Marius is using the Internals samoke:

* Page 53-95 NDK and JNI
* Sample with *FibonacciNative* at page 76
* Check the samples from NDK folder /android/ndk/samples
* The New Circle has a cool sample online at: <https://thenewcircle.com/s/post/49/using_ndk_to_call_c_code_from_android_apps>
* The code is available as ZIP archive: <https://github.com/marakana/FibonacciNative/zipball/master>
* By Git: git clone <https://github.com/marakana/FibonacciNative.git>

**Durnov, Dmitry**

Q1: Full stack profiling.

A1:

Q2: How to arhitect the native/java part (not games).

A2:

Q3: Run new API on older devices.

A3:

**Fedorova, Julia**

Q1: Kiosk app, launcher.

A1:

**Fedotov, Aleksei**

Q1: App to measure the custom room size.

A1:

**Kaleturin, Igor**

Q1: Custom gestures interpretation system-wide and in your app.

A1: A good example with a small app for gesture detection is to be found on the official Android official website : <http://developer.android.com/training/gestures/detector.html> .

For each View (textView, Edittext etc) you can register a listener and do the following

*View myView = findViewById(R.id.my\_view);*

*myView.setOnTouchListener(new OnTouchListener() {*

*public boolean onTouch(View v, MotionEvent event) {*

*// ... Respond to touch events*

*return true;*

*}*

*});*

How to intercept different types of touch gestures (singleplay, long play, zoom etc) – complex?

Check the events in the project **Wheel Menu** from the github. The class where this is visible is located in *customcontrols/WheelMenuSurface.java* in the method *onTouchedEvent*:

*public boolean onTouchEvent(MotionEvent event) {*

*if (event.getAction() == MotionEvent.ACTION\_DOWN)*

*if (event.getAction() == MotionEvent.ACTION\_UP)*

*if (event.getAction() == MotionEvent.ACTION\_MOVE)*

**Larionov, Andrey**

Q1: Face recognition.

A1:

**Leitan, Alex**

Q1: Service communication.

A1:

**Litvinov, Vasilij**

Q1: Pure native applications.

A1: Android framework provides an android.app.NativeActivity.java class to help us

create a "native" activity. In a typical Java activity, we extend android.app.Activity and

overwrite the activity lifecycle methods. NativeActivity is also a subclass of android.

app.Activity and does similar things. At the start of a native activity, NativeActivity.

java will call ANativeActivity\_onCreate, which is declared in native\_activity.h

and implemented by us. In the ANativeActivity\_onCreate method, we can register

our callback methods to handle activity lifecycle events and user inputs.

*At runtime, NativeActivity will invoke these native callback methods when the corresponding events occurred. In a word, NativeActivity is a wrapper that hides the managed Android Java world for our native code, and exposes the native interfaces defined in native\_activity.h.*

NativeActivity - <http://developer.android.com/reference/android/app/NativeActivity.html>

Q2: Python apps for Android.

A2:

Q3: Delphi apps for Android.

A3:

Q4: Analyze an APK file

A4: Marius will discuss free on this elements based on some example. See first the response for Kai**.**

Also check my presentation on SlideShare - <http://de.slideshare.net/fastlink2/droidcon-eastern-europe-2013-how-secure-is-an-androidapp> . A good ideea is to have a look also at the project dexguard - <http://www.saikoa.com/dexguard>

**Mullin, Alexander**

Q1: Custom kernel, custom kernel usage in a custom rom.

A1:

**Potapov, Anton**

Q1: How to force the emulator to use all the available cores.

A1:

Q2: Debugging.

A2:

1. In the Internals course at the page 226-243.
2. Services debugging in Android - <http://stackoverflow.com/questions/9226451/how-to-debug-android-framework-services> and <http://android.opensourceror.org/2010/01/18/android-source/>
3. Step by step debuging NDK - <http://mhandroid.wordpress.com/2011/01/23/using-eclipse-for-android-cc-debugging/>
4. Check also the following article - <http://www.eweek.com/c/a/Linux-and-Open-Source/How-to-Set-Up-Android-Platform-Development-and-Debugging/>
5. A cool alternative article linked tombstone, ndk-stack and addr2line - <http://bytesthink.com/blog/?p=133>

Q3: Have a android 4.4 with an older linux kernel.

A3:

Q4: GCJ -

**Savonichev, Andrei**

Q1: Show the Notifly; Push concept .

A1:

Q2: Native code to Java, Java code to Native.

How do I share raw data with native code?

You can store the data in a *byte[*]. This allows very fast access from managed code. On the native side, however, you're not guaranteed to be able to access the data without having to copy it. In some implementations, GetByteArrayElements and GetPrimitiveArrayCritical will return actual pointers to the raw data in the managed heap, but in others it will allocate a buffer on the native heap and copy the data over.

The alternative is to store the data in a direct byte buffer. These can be created with *java.nio.ByteBuffer.allocateDirect*, or the *JNI NewDirectByteBuffer* function. Unlike regular byte buffers, the storage is not allocated on the managed heap, and can always be accessed directly from native code (get the address with *GetDirectBufferAddress*). Depending on how direct byte buffer access is implemented, accessing the data from managed code can be very slow (more on this on <http://developer.android.com/training/articles/perf-jni.html> ).

**How to call Java from native?**

**C code**

#include <string.h>

#include <stdio.h>

#include <jni.h>

jstring Java\_the\_package\_MainActivity\_getJniString( JNIEnv\* env, jobject obj){

jstring jstr = (\*env)->NewStringUTF(env, "This comes from jni.");

jclass clazz = (\*env)->FindClass(env, "com/intel/MainActivity");

jmethodID messageMe = (\*env)->GetMethodID(env, clazz, "messageMe", "(Ljava/lang/String;)Ljava/lang/String;");

jobject result = (\*env)->CallObjectMethod(env, obj, messageMe, jstr);

const char\* str = (\*env)->GetStringUTFChars(env,(jstring) result, NULL);

printf("%s\n", str);

return (\*env)->NewStringUTF(env, str);

}

**Java code:**

public class MainActivity extends Activity {

private static String LIB\_NAME = "thelib";

static {

System.loadLibrary(LIB\_NAME);

}

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

TextView tv = (TextView) findViewById(R.id.textview);

tv.setText(this.getJniString());

}

public String messageMe(String text) {

System.out.println(text);

return text;

}

public native String getJniString();

}

Q1: How I can subscribe for some events from native code ?

A1: Android Native Development Kit Cookbook page 180, chapter *Detecting and handling input events at Android NDK*.

**Smelov, Vladimir**

Q1: Location reminder .

A1: Based on the sample located on <http://developer.android.com/training/location/receive-location-updates.html> and <http://developer.android.com/training/location/index.html> I will demo the /Samples/LocationUpdates

**Smoryakova, Valya**

Q1: App with scripting capabilities aka monkeyrunner.

A1:

**Solovyeva, Daria**

Q1: Overlay window information.

A1: Hard to find the solution but here you go <http://stackoverflow.com/questions/4481226/creating-a-system-overlay-always-on-top-button-in-android> and working example at <http://goo.gl/2qX50G>

Check also the settings->gpu equivalent source code <https://github.com/android/platform_packages_apps_settings/tree/master/src/com/android/settings>

**Surmin, Igor**

Q1: How to debug the native part of the platform.

A1:

**Vinogradov, Sergey**

Q1: Real custom rom.

A1: Cyangenmod is the best example of combining real drivers over a customized AOSP + own app. For example the Nexus 4 binary releases can be found on <http://download.cyanogenmod.org/?device=mako&type=stable>

If you are interesting on getting a Nexus 5 compile from scratch the guide on <http://wiki.cyanogenmod.org/w/Build_for_hammerhead> will help get the source code repository on git://github.com/CyanogenMod/android.git -b cm-11.0 and compile it in a working ROM.

* Officialflashing documentation can be found: <http://source.android.com/source/building-devices.html>
* How to build Android AOSP for Nexus 4 - <http://nosemaj.org/howto-build-android-nexus-4>
* How to build Android 4.3 for Nexus 4 - <http://nosemaj.org/build-android-4-3-nexus-4>
* Nexus 4 and Nexus 5 drivers - <https://developers.google.com/android/nexus/drivers>
* Howto Build Android KitKat (4.4) for the Google Nexus 5 - <http://nosemaj.org/howto-build-android-kitkat-nexus-5>

**Zhelybalov, Igor**

Q1: Root/restore a rooted HTC device.

A1: Each device is different but HTC works with Revone and using a RUU specific file, more information on step by step instruction on XDA or http://htc-one.wonderhowto.com/how-to/return-your-rooted-unlocked-htc-one-back-factory-settings-for-warranty-repairs-0149204/