ITSMAP Lesson 1

Introduction to Android

Jesper Rosholm Tørresø

“An Android dogma”

Do not ask what you can do with Android!!!

Ask what Android can do for you!!

The Android Framework controls your App on behalf of your requirements/commands

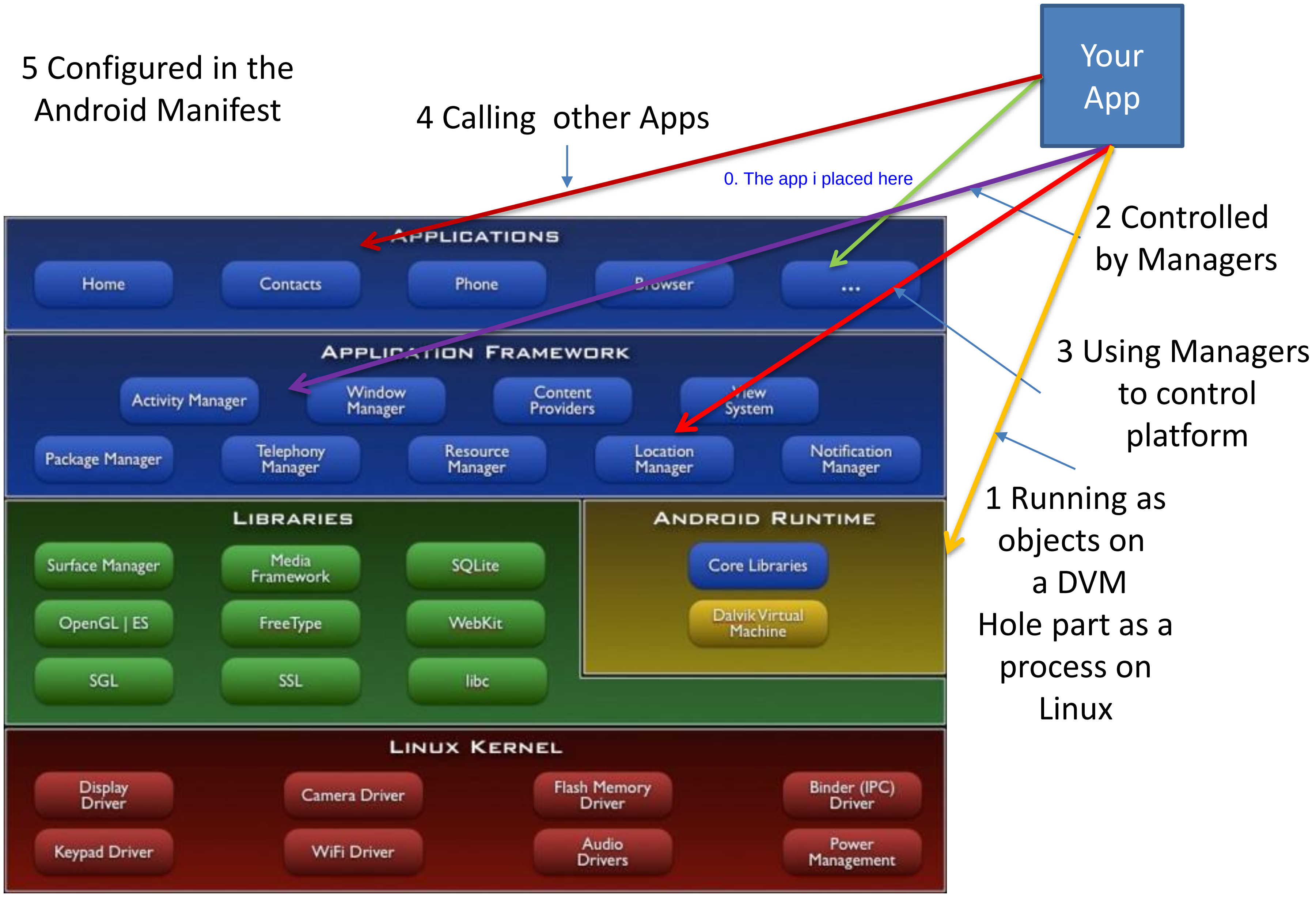
Doing an Android App is

1. Understanding how the Android Framework works
2. Knowing how to write the program/code, the App, that runs in the Android Framework
3. Having focus on the business domain that the App cover
4. Having focus on how the Android Facilities covers business domain in the best way.
5. Use development tools as much as possible.

In ITSMAP “You are requested to”

- Work with the Android framework
- To do an Android App in the Theme Project
- Take care of the way you cover the five points in the previous slide
- Especially the points one, two four and five in previous slide!

Android Framework Architecture



A Framework?

Source: http://en.wikipedia.org/wiki/Software_framework

Frameworks contain key distinguishing features that separate them from normal libraries:

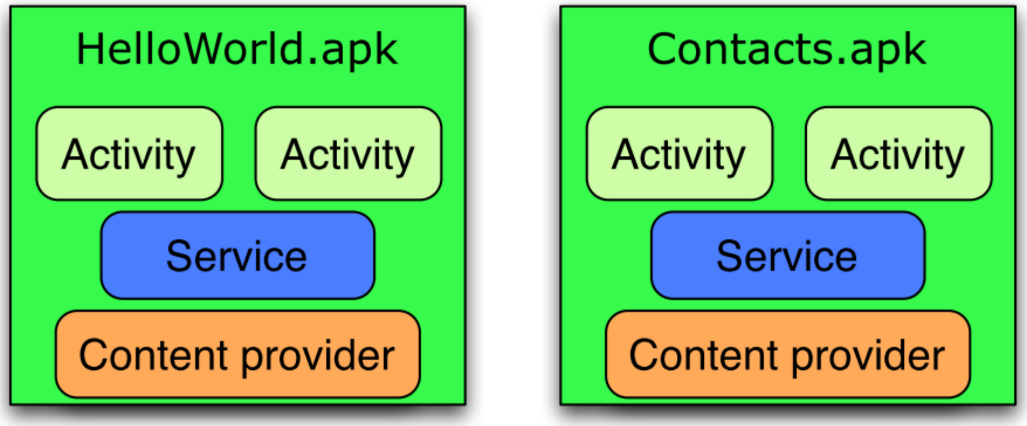
- **inversion of control** - In a framework, unlike in libraries or normal user applications, the overall program's **flow of control** is not dictated by the caller, but by the framework.
- **default behavior** - A framework has a default behavior. This default behavior must actually be some useful behavior and not a series of no operation.
- **extensibility** - A framework can be extended by the user usually by selective overriding or specialized by user code providing specific functionality.
- **non-modifiable framework code** - The framework code, in general, is not allowed to be modified. Users can extend the framework, but not modify its code.

Think SOLID!!!

Initial	Stands for (acronym)	Concept
S	SRP	Single responsibility principle a class should have only a single responsibility.
O	OCP	Open/closed principle “software entities … should be open for extension, but closed for modification”.
L	LSP	Liskov substitution principle “objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program”. See also design by contract.
I	ISP	Interface segregation principle “many client-specific interfaces are better than one general-purpose interface.”
D	DIP	Dependency inversion principle one should “Depend upon Abstractions. Do not depend upon concretions.” Dependency injection is one method of following this principle.

http://en.wikipedia.org/wiki/SOLID_(object-oriented_design) click links for details

Android App. How?

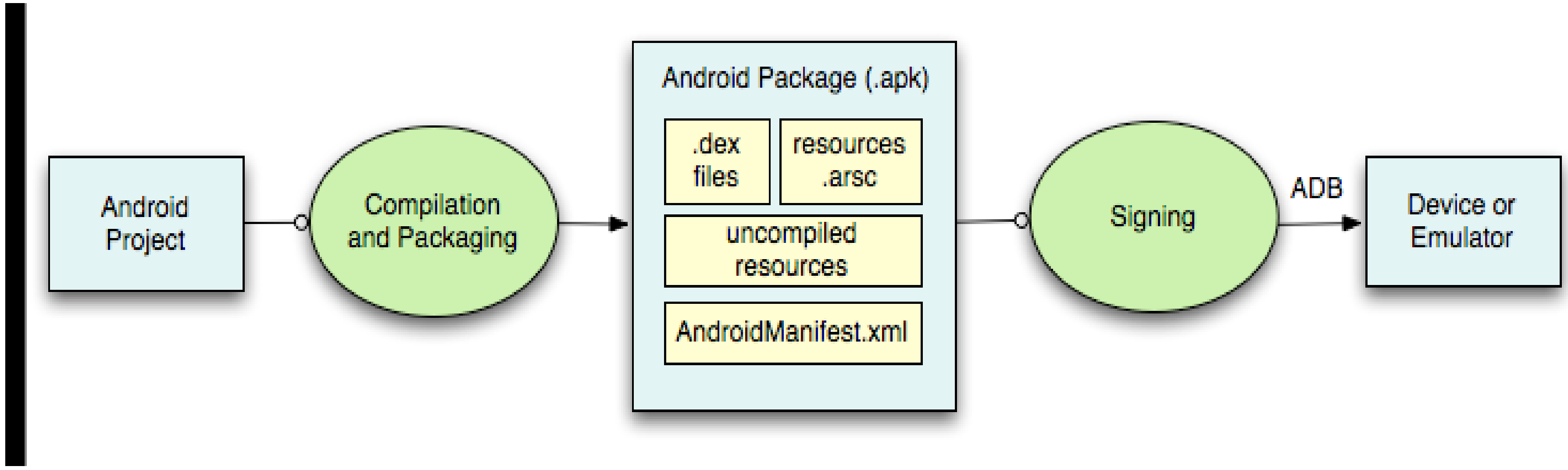


- An Android App is a collection of different components put into a jar file, all bound together through the package and file structure and a XML manifest
- The APK file is installed on the smartphone or tablet, the target.

Which components?
The anatomy of an Android App

- Activity + Fragments + Views (Foreground task coupled to the User and the User Interface)
- Service (Background task not coupled to the UI)
- Content Provider (Provides data in a structured way across FW/App)
- Broadcast Receiver (Receiver of messages from other components across FW / App)
- Widget (little part of the App embedded in another App)
- Use of other Framework Resources (notifications, option menu, action bar…)
- All build up of a mix of Java Code + Java Libraries, XML Files, images, sounds, movies, text and **external resources** in given a structure given from the development tools and the Android framework.

Building and Running Android App



.dex = dalvic executable

.arsc = compiled application resources

ADB = Android debugging bridge (to communicate with emulator or android devices)

AndroidManifest.xml = android application manifest file

Externalization

That is:

- Putting a substantial part of the App' data, UI, configuration etc. in XML files
- Allowing you to customize the App'
  - Appearance
  - Behavior
  - Localization (i.e. Language)
  - Target control (i.e. Screen and API Level)
- Without recoding it.!

But Recompiling is needed!

- This App Externalization and following defragmentation requires an overview from you!

Your code interact with Android FW 1

- Inheritance of generic component properties

```
public class MainActivity extends Activity {
/** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
      super.onRestoreInstanceState(savedInstanceState);
     }
    @Override
    public void onRestart(){
      super.onRestart();
      ………… etc for 213 methods to override
```

Activity Inheritance

```
package dk.iha.itsmap.lesson1;

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

public class MainActivity extends Activity {
    /** Called when the activity is first created.
     * Called at the start of the full lifetime.
     *   */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    // Called after onCreate has finished, use to restore UI state
    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
      super.onRestoreInstanceState(savedInstanceState);
      // Restore UI state from the savedInstanceState.
      // This bundle has also been passed to onCreate.
    }
```

Service

```java
public class Lesson1Service extends Service {
@Override
public IBinder onBind(Intent intent) {
// TODO Put your code here
return null;
}
@Override
public void onCreate() {
// TODO Put your code here
}
@Override
public void onStart(Intent intent, int startId) {
// TODO Put your code here
}
}
```

Content Provider

```
public class Lesson1ContentProvider extends ContentProvider {
public static final Uri CONTENT_URI = Uri
.parse("content://dk.iha.itsmap.lesson1.lesson1contentprovider")
;
@Override
public int delete(Uri uri, String selection, String[]
selectionArgs) {
// TODO Put your code here
return 0;
}
@Override
public String getType(Uri uri) {
// TODO Put your code here
return null;
}………. + other methods
```

Your code interact with Android FW 2

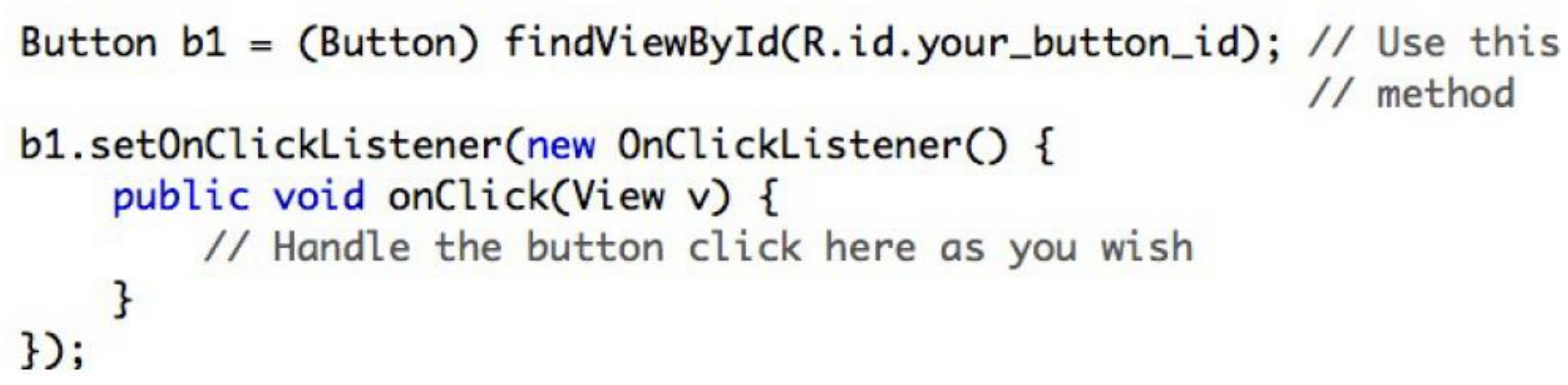
- Through references to Managers and other components
- Using Subscription and Call Back (Publish-Subscribe)

```
LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
locationManager.addGpsStatusListener(new
GpsStatus.Listener(){

public void onGpsStatusChanged(int event) {
switch(event){
// Event sent when the GPS system has started
case GpsStatus.GPS_EVENT_STARTED:
// put your code here
```

Your code interact with Android FW 3

- Subscription UI events



```
Button b1 = (Button) findViewById(R.id.your_button_id); // Use this
                                                        // method
b1.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        // Handle the button click here as you wish
    }
});
```

Your code interact with Android FW 4

- Through Intents Explicit adresses

```
public void manTimer(View view) {
Intent mainIntent = new Intent(this,
FFShakerControl.class);
startService(mainIntent);
}
Or .. Implicit
Intent iImp = new Intent("actionName"); //TODO
Replace 'actionName' as appropriate for your action
(for example, Intent.ACTION_EDIT)
iImp.addCategory("categoryName"); //TODO Replace
'categoryName' as appropriate for your category (for
example, Intent.CATEGORY_DEFAULT)

startActivity(iImp);
```

Your code interact with Android FW 4

- Through Intents Explicit adresses

```
public void manTimer(View view) {
Intent mainIntent = new Intent(this, FFShakerControl.class);
startService(mainIntent);
}
Or .. Implicit
Intent iImp = new Intent("actionName"); //TODO Replace 'actionName' as appropriate for your action (for example, Intent.ACTION_EDIT)
iImp.addCategory("categoryName"); //TODO Replace 'categoryName' as appropriate for your category (for example, Intent.CATEGORY_DEFAULT)

startActivity(iImp);
```

Exercise Lecture 1

- Installation and Setting of the development environment
  - Android Studio IDE
  - Android SDK
  - Emulator, Android Virtual Device (AVD),
  - Emulator Accelerator (HAXM)
- Create an android application with a single activity and view and run it on emulator or physical device
- Get an overview of the different components of Android SDK and Android Studio IDE