Native Apps I (Android)

Chamilo

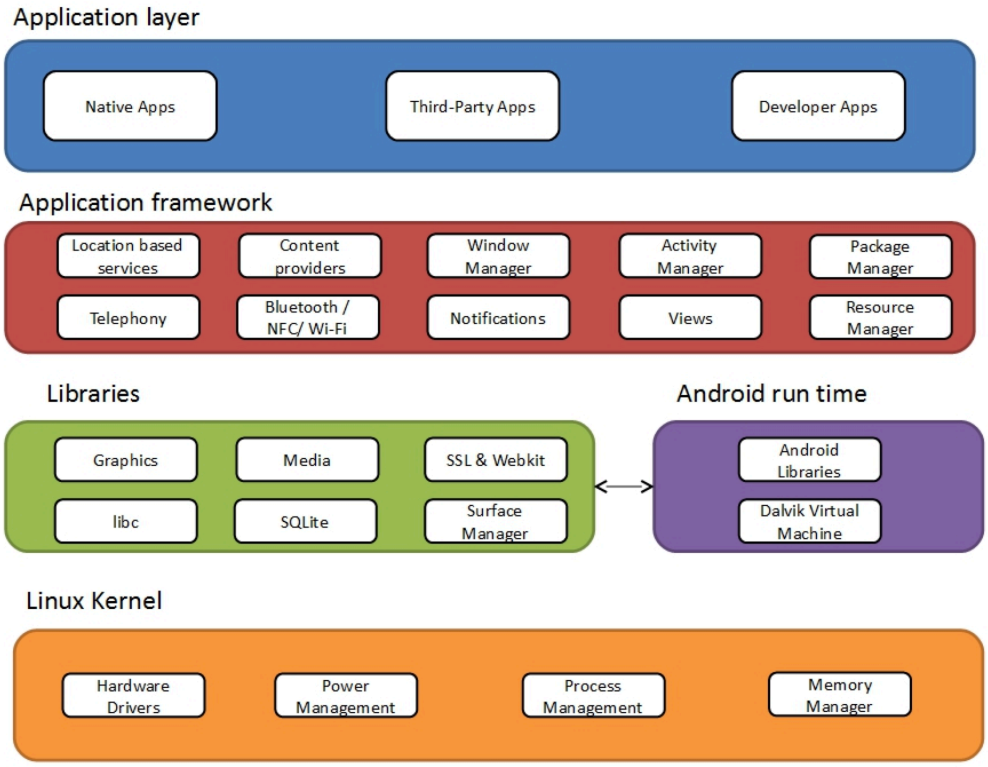
Intro

- geen thread in thread starten
- extra dingen -> bonus punten

Examen vragen:

- open source -> vanaf google iets uitbrengt dan pas mogen wij er mee werken/uitbreiden
- Wat is een embedded device
- Nougat vs Marshmallow (vorige transities zijn niet belangrijk)
- Software Stack

Software Stack



- **Linux kernel** — Core services (including hardware drivers, process and memory management, security, network, and power management) are handled by a Linux 2.6 kernel. The kernel also provides an abstraction layer between the hardware and the remainder of the stack.
- **Libraries** — Running on top of the kernel, Android includes various C/C++ core libraries such as libc and SSL
- **Android run time (ART)** — The run time is what makes an Android phone an Android phone rather than a mobile Linux implementation. Including the core libraries and the Dalvik VM, the Android run time is the engine that powers your applications and, along with the libraries, forms the basis for the application framework.
- **Core libraries** — Although most Android application development is written using the Java language, Dalvik is not a Java VM. The core Android libraries provide most of the functionality available in the core Java libraries, as well as the Androidspecific libraries.
- **Dalvik VM** — Dalvik is a register-based Virtual Machine that's been optimized to

ensure that a device can run multiple instances efficiently. It relies on the Linux kernel for threading and low-level memory management.
- **Application framework** — The application framework provides the classes used to create Android applications. It also provides a generic abstraction for hardware access and manages the user interface and application resources.
- **Application layer** — All applications, both native and third-party, are built on the application layer by means of the same API libraries. The application layer runs within the Android run time, using the classes and services made available from the application framework.

Java BASICS

```
   (systeem onafhankelijk)

code -> bytecode -> JVM
                     | (interpreteren)
                Machine Code (systeem afhankelijk)
```

ART vs Dalvik

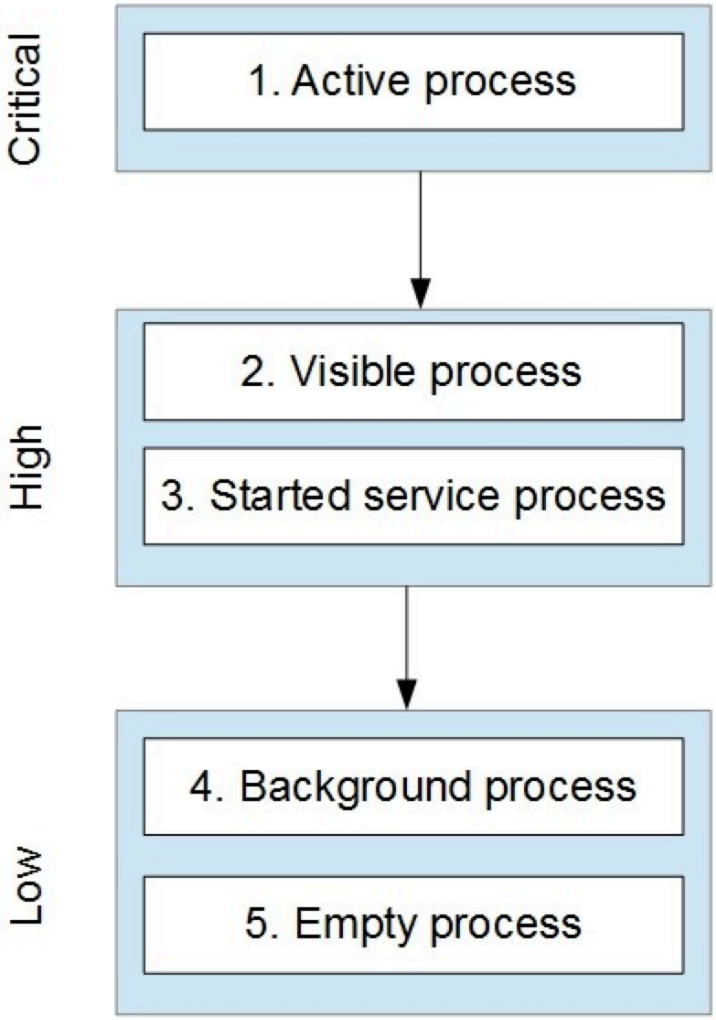
- ART: alles compileren
  - voordeel: kan veel sneller runner / performanter
  - nadeel: meer geheugen gebruik
- Dalvik: Compileren wat je nodig hebt (JIT - Just In Time)

Activities

The android life cycle

> Android heeft verschillende toestanden, en Android gaat heel streng om met je geheugen en kan deze processen stoppen

- **Active processes** — Active (foreground) processes have application components the user is interacting with. These are the processes Android tries to keep responsive by reclaiming resources from other applications.
- **Visible processes** — Visible but inactive processes are those hosting “visible” Activities
- **Started Service processes** — Processes hosting Services that have been started. Because these Services don’t interact directly with the user, they receive a slightly lower priority than visible.
- **Background processes** — Processes hosting Activities that aren’t visible and that don’t have any running Services.
- **Empty processes** — To improve overall system performance, Android will often retain an application in memory after it has reached the end of its lifetime. Android maintains this cache to improve the start-up time of applications when they’re relaunched. These processes are routinely killed, as required.

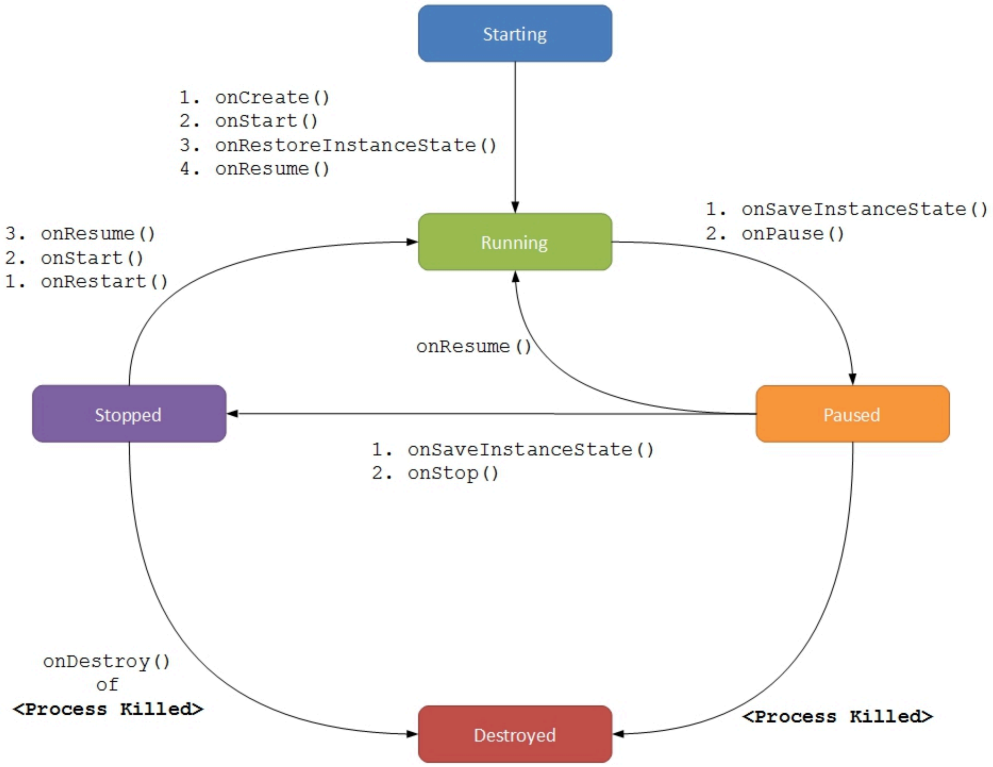


Basic Building blocks: Activities

> **Activities** are screens, each activity has its own states. Each activity can be killed by Android itself

Hooks

Examen vraag:



- **Start -> Running**
  1. onCreate()
  2. onStart()
  3. onRestoreInstanceState()
  4. onResume()
- **Running -> Paused**
  1. onSaveInstanceState()
  2. onPause()
- **Paused -> Running**
  1. onResume()
- **Paused -> Stopped**
  1. onSaveInstanceState()
  2. onStop()
- **Paused -> Destroyed**
  1. \
- **Stopped -> Running**
  1. onRestart()
  2. onStart()
  3. onResume()
- **Stopped -> Destroyed**
  1. onDestroy()
  2. \

Zelf als je van Portrait naar landscape gaat, wordt de activity gekilled, dus we moeten zeker de onSaveInstanceState allemaal goed implementeren.

Example Activity

```
public class ExampleActivity extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            // The activity is being created.
        }
        @Override
        protected void onStart() {
            super.onStart();
            // The activity is about to become visible.
        }
        @Override
        protected void onResume() {
            super.onResume();
            // The activity has become visible (it is now "resumed").
        }
        @Override
        protected void onPause() {
            super.onPause();
            // Another activity is taking focus (this activity is about to
be "paused").
        }
        @Override
        protected void onStop() {
            super.onStop();
            // The activity is no longer visible (it is now "stopped")
        }
        @Override
        protected void onDestroy() {
            super.onDestroy();
            // The activity is about to be destroyed.
        }
}
```

The Manifest

- It includes nodes for each of the Activities, Services, Content Providers, and Broadcast Receivers that make up your application and, using Intent Filters and Permissions, determines how they interact with each other and with other applications.
- The manifest can also specify application metadata (such as its icon, version number, or theme), and additional top-level nodes can specify any required permissions, unit tests, and define hardware, screen, or platform requirements.
- The manifest is made up of a root manifest tag with a package attribute set to the project's package. It should also include an xmlns:android attribute that supplies several system attributes used within the fi le.
- Use the versionCode attribute to define the current application version as an integer

that increases with each version iteration, and use the versionName attribute to specify a public version that will be displayed to users.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="be.hogent.helloandroid">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            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>
```

Basic Application

```
package com.commonsware.android.button;

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

public class ButtonDemoActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main); // res/layout/main.xml

        // R is een klasse die gegenereerd filetje die referenties heeft
naar je xml resources
    }
}
```

res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button"/>

</LinearLayout>
```