Module: Mobile Application development (Android)

Session 1: Introduction to Android

1. What is Android?

1.1. Android Operation System

1.2. Google Play (Android Market)

2. Security and permissions

2.1. Security concept in Android

2.2. Permission concept in Android

3. App Framework

3.1 Features

3.2 Android Architecture

3.3 Applications

3.4 Application Framework

3.5 Libraries

3.6 Android Runtime

3.7 Linux Kernel

4. Android applications and tasks

4.1. Application

4.2. Tasks across application borders

## 1. What is Android?

### 1.1. Android Operation System

Android is an operating system based on Linux with a Java programming interface.

The Android Software Development Kit (Android SDK) provides all necessary tools to develop Android applications. This includes a compiler, debugger and a device emulator, as well as its own virtual machine to run Android programs.

Android is currently primarily developed by Google.

Android allows background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL libraries, access to the file system and provides an embedded SQLite database.

Android applications consist of different components and can re-use components of other applications. This leads to the concept of a task in Android; an application can re-use other Android components to archive a task. For example you can trigger from your application another application which has itself registered with the Android system to handle photos. In this other application you select a photo and return to your application to use the selected photo.

### 1.2. Google Play (Android Market)

Google offers the Google Play service in which programmers can offer their Android application to Android users. Google phones include the Google Play application which allows to install applications.

Google Play also offers an update service, e.g. if a programmer uploads a new version of his application to Google Play, this service will notify existing users that an update is available and allow to install it.

Google Play used to be called Android Market.

## 2.Security and permissions

### 2.1. Security concept in Android

During deployment on an Android device, the Android system will create a unique user and group ID for every Android application. Each application file is private to this generated user, e.g. other applications cannot access these files.

In addition each Android application will be started in its own process.

Therefore by means of the underlying Linux operating system, every Android application is isolated from other running applications.

If data should be shared, the application must do this explicitly, e.g. via a service or a content provider.

### 2.2. Permission concept in Android

Android also contains a permission system. Android predefines permissions for certain tasks but every application can define additional permissions.

An Android application declare its required permissions in its AndroidManifest.xml configuration file. For example an application may declare that it requires access to the Internet.

Permissions have different levels. Some permissions are automatically granted by the Android system, some are automatically rejected.

In most cases the requested permissions will be presented to the user before installation of the application. The user needs to decide if these permissions are given to the application.

If the user denies a permission required by the application, this application cannot be installed. The check of the permission is only performed during installation, permissions cannot be denied or granted after the installation.

Not all users pay attention to the required permissions during installation. But some users do and they write negative reviews on Google Play.

## 3. App Framework

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The [Android SDK](http://developer.android.com/sdk/index.html) provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

### 3.1 Features

**Application framework** enabling reuse and replacement of components

* **Dalvik virtual machine** optimized for mobile devices
* **Integrated browser** based on the open source [WebKit](http://webkit.org/) engine
* **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
* **SQLite** for structured data storage
* **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
* **GSM Telephony** (hardware dependent)
* **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
* **Camera, GPS, compass, and accelerometer** (hardware dependent)
* **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

### 3.2 Android Architecture

The following diagram shows the major components of the Android operating system. Each section is described in more detail below



### 3.3 Applications

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

### 3.4 Application Framework

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

* A rich and extensible set of [Views](http://developer.android.com/resources/tutorials/views/index.html) that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
* [Content Providers](http://developer.android.com/guide/topics/providers/content-providers.html) that enable applications to access data from other applications (such as Contacts), or to share their own data
* A [Resource Manager](http://developer.android.com/guide/topics/resources/resources-i18n.html), providing access to non-code resources such as localized strings, graphics, and layout files
* A [Notification Manager](http://developer.android.com/reference/android/app/NotificationManager.html) that enables all applications to display custom alerts in the status bar
* An [Activity Manager](http://developer.android.com/reference/android/app/Activity.html) that manages the lifecycle of applications and provides a common navigation backstack

For more details and a walkthrough of an application, see the [Notepad Tutorial](http://developer.android.com/training/notepad/index.html).

### 3.5 Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

* **System C library** - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
* **Media Libraries** - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
* **Surface Manager** - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
* **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
* **SGL** - the underlying 2D graphics engine
* **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
* **FreeType** - bitmap and vector font rendering
* **SQLite** - a powerful and lightweight relational database engine available to all applications

### 3.6 Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

### 3.7 Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

## 4. Android applications and tasks

### 4.1. Application

An Android application consists out of different Android components and additional resources. The Android system knows activities, services, broadcast receiver and content provider as components.

### 4.2. Tasks across application borders

Android application components can connect to components of other Android applications to create tasks. For example an application which allows you to make a photo can start an email application and instruct this application to create a new email and attach a photo to this email.

## 4. Android user interface components

The following description gives a overview of the most important user interface related component and parts of an Android application.

### 4.1. Activity

An activity represents the visual representation of an Android application. activities use views, i.e. user interface widgets as for example buttons and fragments to create the user interface and to interact with the user.

An Android application can have several activities.

### 4.2. Fragments

Fragments are components which run in the context of an activity. A fragment encapsulates application code so that it is easier to reuse it and to support different sized devices.

Fragments are optional components which allow you to reuse user interface and non user interface components for different devices configurations.

### 4.3. Views and layout manager

Views are user interface widgets, e.g. buttons or text fields. The base class for all views is the android.view.View class. Views have attributes which can be used to configure their appearance and behavior.

A layout manager is responsible for arranging other views. The base class for these layout managers is the android.view.ViewGroup class which extends the View class.

Layout managers can be nestled to create complex layouts. You should avoid nestling them to deeply too deeply as this has a negative impact on the performance.

### 4.4. Device configuration specific layouts

The user interface for Activities is typcally defined via XML files (layout files). It is possible to define defined layout file for different device configuration, e.g. based on the available width of the actual device running the application.

Fragments are designed to support such a setup.

The following pictures shows an activity called MainActivity. On a smaller screen it shows one fragment and allows that the user navigates to another fragment. On a wide screen it shows two fragments.

