**Android fundamentals**

**1 Chapter Objectives**  
Understand the architecture with MVC model of Android.

**2 Architecture**  
Contains 4 main layers:  
1. Application:  
*•* Description: written in Java, where to make the app  
*•* Example: Contacts, Phone, Browser,...  
2. Application Framework  
*•* Description: in Java, higher level, UI, location service, notification  
*•* Example: Window manager, Resource manager, ...  
3. Librearies:  
*•* Description: mostly in C/C++, low level, render text, play media, local database, ...  
*•* Example: SQLite stores relational database, OpenGL - Open Graph ics Library, ...  
4. Linux Kernel  
*•* Description: well shaped, secured and activity development  
*•* Example: Display driver, Audio driver, ...

**3 Compilation**  
1. Description  
*•* Java source code = Java compiler  
*•* Reason: compile once run everywhere - on many different platform.  
2. Example  
*•* Dalvik VM : used very long time ago  
*•* ART VM: now change to Android Runtime Virtual Machine  
**4 MVC Model**  
Figure 1: Simple MVC model  
*•* Model: store  
*•* View: display  
*•* Controller: process actions in UI  
**4.1 Controller**  
**4.1.1 Context and Application**  
1. Context  
*•* Central command center  
*•* System services  
2*•* Access application-specific data  
Example: setting, private files, resources, assets  
2. Application  
*•* Subclass - child class of context  
Example: Global data, early initialization of libraries  
*•* Android memory management  
Example:  
Garbage collector: collect objects no used  
Upper limit for each application  
”Kill” activities when low on memory

Out-of-memory exception: very popular  
*•* AndroidManifest.xml  
Example:  
Metadata about the app  
Target SDK  
”Entry point” of the app  
Permissions, activities, services, receivers...  
*•* Declare permission

**4.1.2 Activity**  
*•* A kind of controller - mean in the middle of model and view, update model  
to UI  
*•* In Android do not have a main(), all codes are in different activities  
Example: like different webpages in the website, each page is an UI  
and can click button to go to another UI  
*•* Activity:  
**–** Is fundamental building block  
**–** Has a unique task or purpose  
**–** Need at least one per application  
**–** Handle display of single screen  
**–** Controls UI  
Figure 3: Activity Lifecycle  
*•* Activity lifecycle: states different from webpage (all content cleared when  
closed)  
**–** onCreate() : initialization  
@override: polymorphism call parent  
Always choose which view to use/control  
**–** onStart(): visible state  
**–** onPause(): do not have to override (just cases you need)  
Example1: Facebook messenger with small circle icon  
Example2: Camera in Facebook - only when want to push image

**–** onStop()  
Example: Gmail  
Switch activity: pause then stop  
**–** onResume(): continue  
Example: When you need camera start it in onResume()  
**–** Screen orientaion  
onSaveInstanceState()  
onDestroy() - will be called if no memory leak  
**–** Create a new activity instance  
onCreate()  
onRestoreInstanceState()  
**–** Close current activity: finish()  
Example: Dialog share on Facebook  
*•* Intent: pass information from one activity to another  
**–** Asynchronous messaging mechanism  
**–** Message to pass to other activities/services  
**–** Contains data  
Example: In Gmail has a list of email, you can click to show  
details  
**4.1.3 Fragment**  
*•* Description  
**–** Represents a behavior or a portion of user interface  
**–** Is building block of the Fundamental building blocks  
**–** Is officially supported from Honeycomb [API 11]  
**–** Is optional  
Example some apps do not need fragment: games, camera, calculator, ...  
*•* Example: Contact with list on the left and details on the right  
*•* Purpose  
**–** Adapt UI according to devices - explosion in the variety of devices  
**–** Screen size, resolution, density, orientation differs  
*•* Lifecycle: similar to Activity  
*•* Activity with fragments: is simplified, coordinates fragments, uses FragmentManager

*•* Put inside a layout XML  
*•* Dynamically created using codes  
*•* Example popular fragment classes: DialogFragment, ListFragment, Pref  
erenceFragment  
**4.2 View**  
*•* Description: basic building blocks of UI - what user interacts with  
*•* Attributes  
**–** id: findViewById()  
**–** width, height  
**–** padding (distance between border and content) and margin (distance  
of border of the view to another view)  
**–** visibility: visible, invisible, non  
**–** alpha: classic transparent  
**–** rotation  
**–** background  
**–** click  
*•* TextView ( like span in HTML)  
**–** setText()  
**–** can contain one and only one icon  
**–** drawable, font, gravity, style, align  
*•* ImageView  
**–** src: setImageResource()  
**–** scaleType: fitXY, fitStart, fitEnd, centerCrop, centerIn side  
**–** tint, crop, viewBounds  
*•* View Group  
**–** Contain children (other View)  
**–** LayoutParams  
**–** Example important subclasses: FrameLayout, LinearLayout, Rela  
tiveLayout, AbsListView  
*•* Button  
**–** Push-button  
**–** State-list

**–** onClick()  
*•* EditText  
**–** TextBoxes: allow to edit a text  
**–** getText()  
**–** Selection