



# Priyadarshini College of Engineering Nagpur

## Android Mobile application Development (AMAD)

### Part -2

**Dr. R. V. Bobate**  
**Deptt. of Electronics & Comm. Engg**

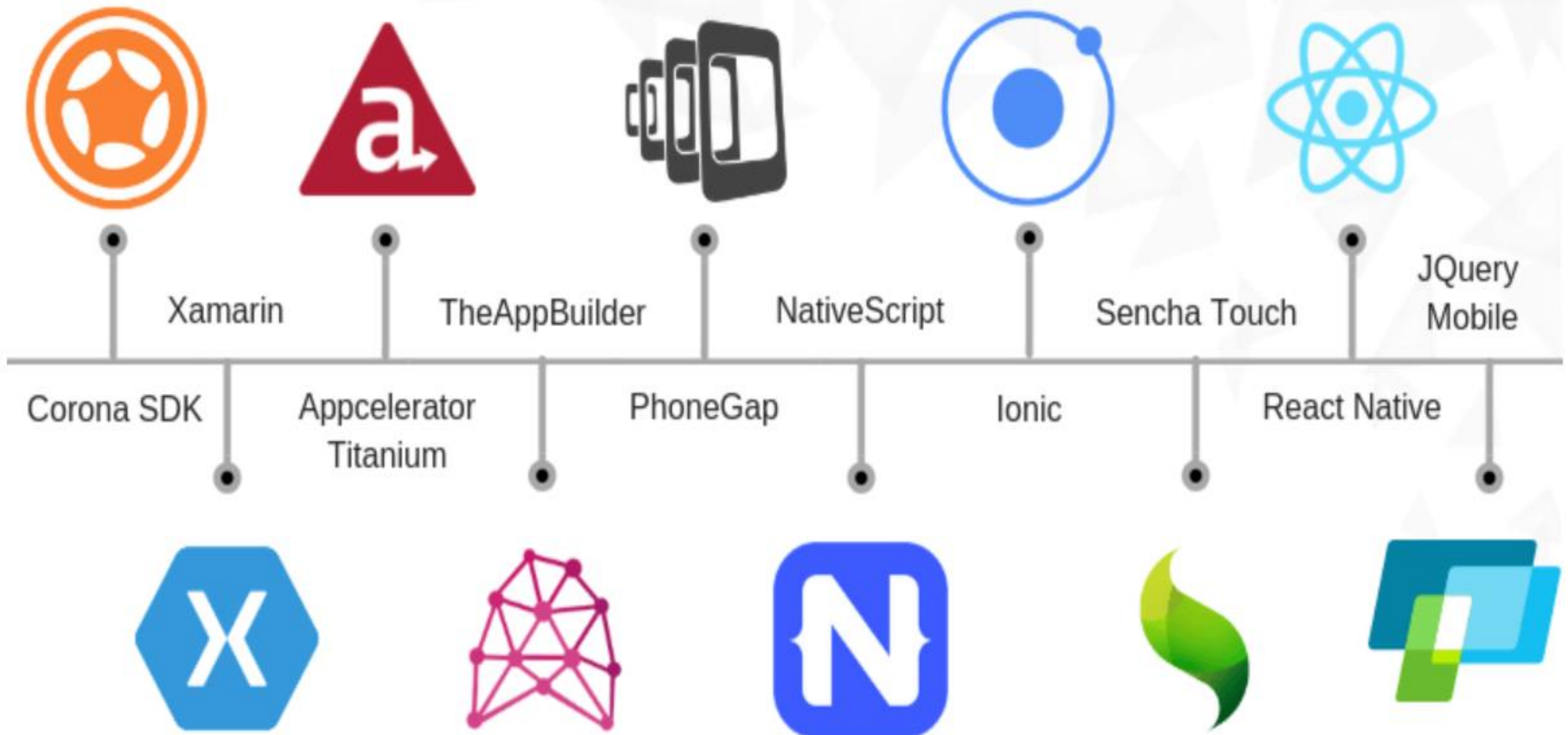


# 10 Excellent Platforms For Building Mobile Apps

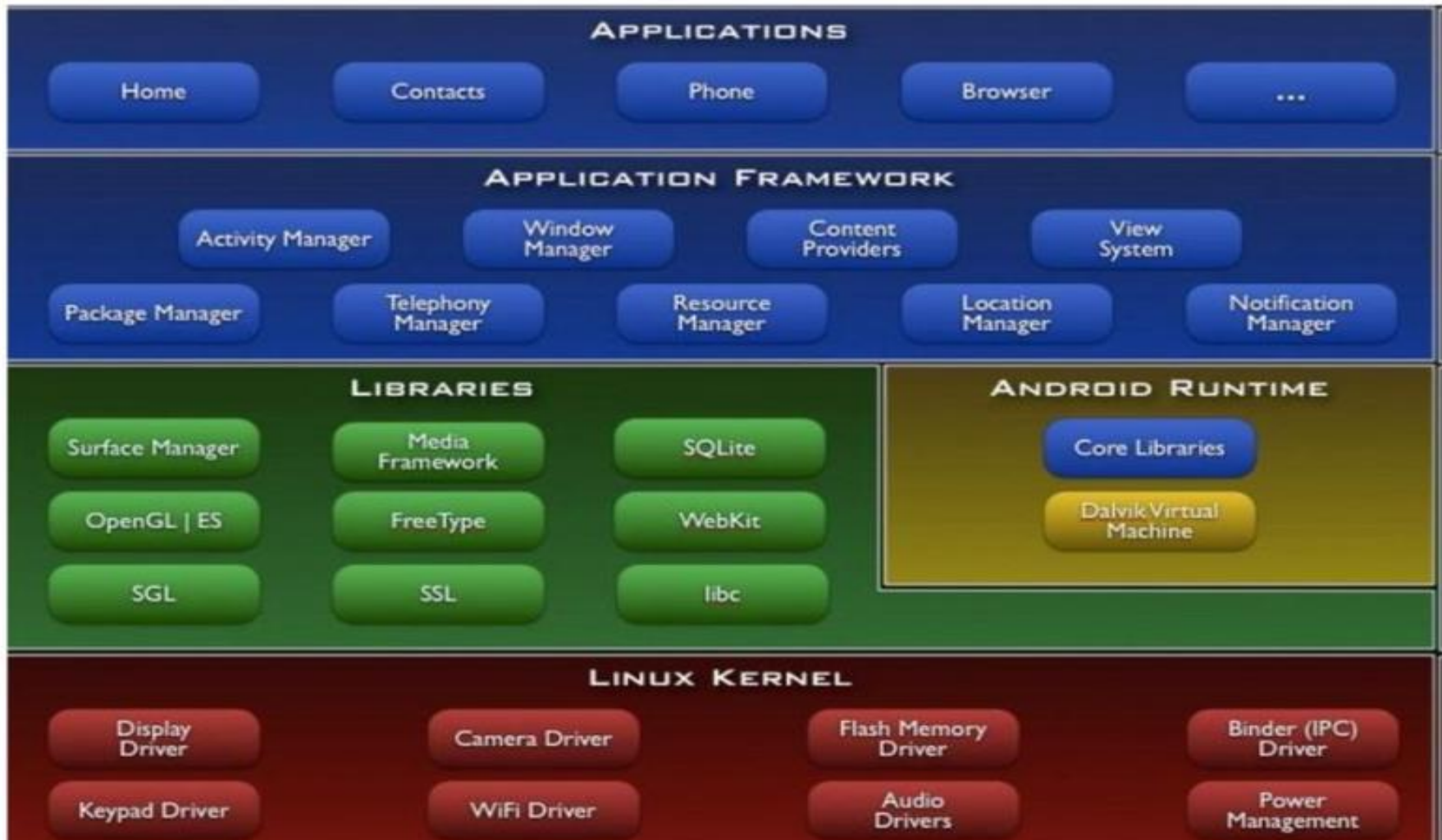


## Android Development Frameworks for Building Superior Mobile Application

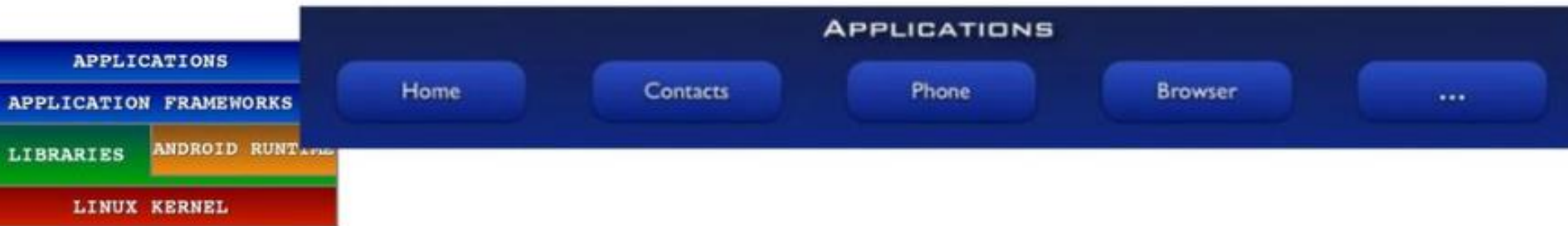
### CROSS - PLATFORM FRAMEWORKS



# Platform - The Android Software Stack



# Android S/W Stack - Application



- Android provides a set of core applications:
  - ✓ Email Client
  - ✓ SMS Program
  - ✓ Calendar
  - ✓ Maps
  - ✓ Browser
  - ✓ Contacts
  - ✓ Etc
- All applications are written using the Java language.

# Android S/W Stack – App Framework



- Most of the application framework accesses these core libraries through the Dalvik VM, the gateway to the Android Platform

# Android S/W Stack – App Framework (Cont)

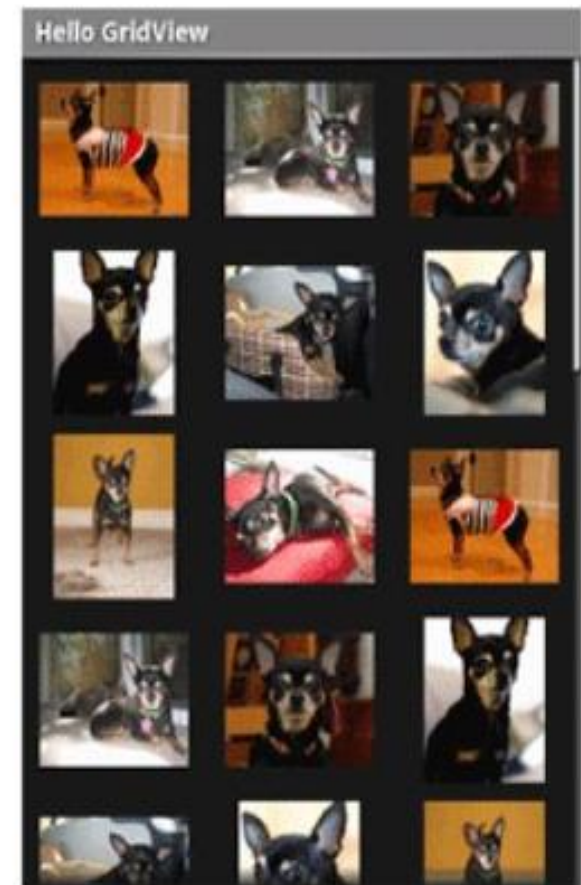
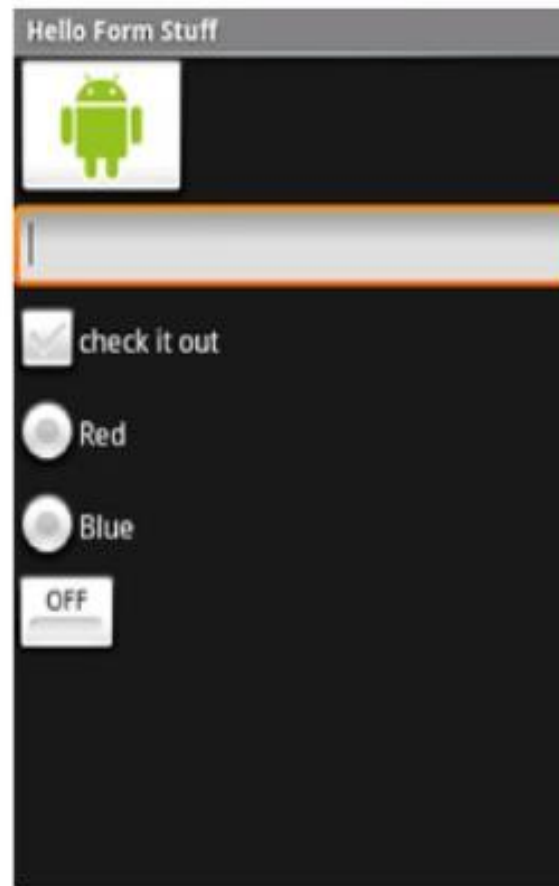
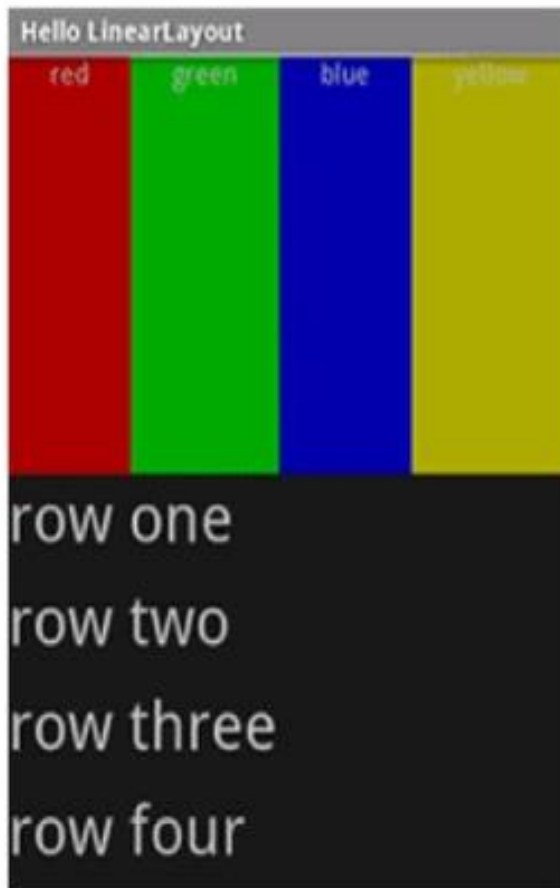
<b>Feature</b>	<b>Role</b>
View System	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser
Content Provider	Enabling applications to access data from other applications or to share their own data
Resource Manager	Providing access to non-code resources (localized string , graphics, and layout files)
Notification Manager	Enabling all applications to display customer alerts in the status bar
Activity Manager	Managing the lifecycle of applications and providing a common navigation backstack



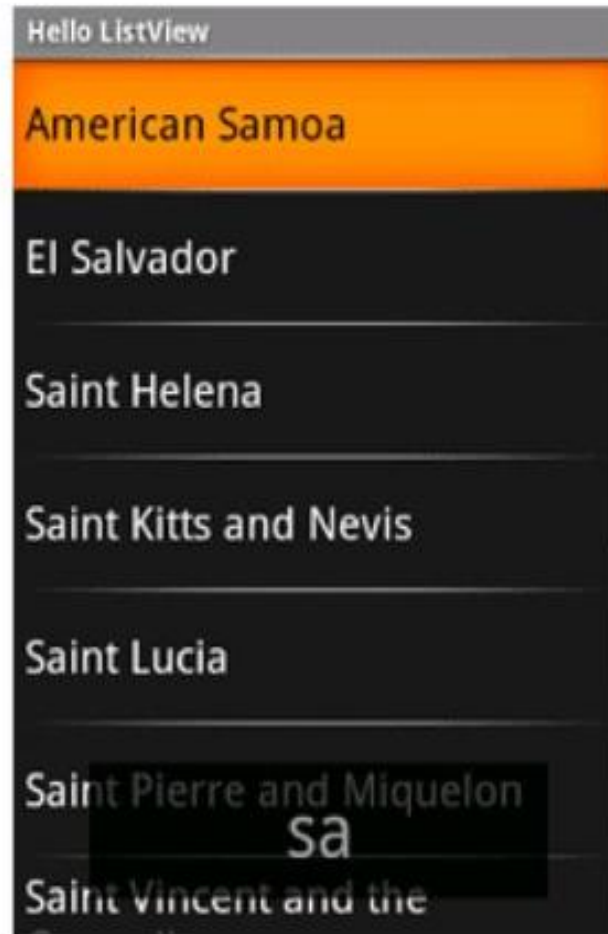
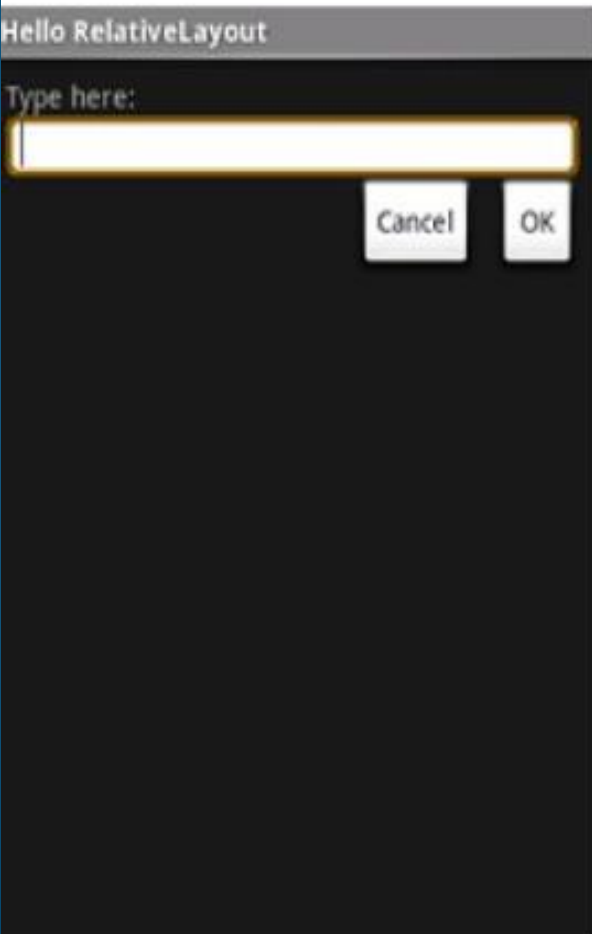
# Notification Manager



# View System



# View System



# Android S/W Stack - Libraries



- Including a set of C/C++ libraries used by components of the Android system
- Exposed to developers through the Android application framework

# Android S/W Stack - Libraries

- The media libraries are based on PacketVideo's (<http://www.packetvideo.com/>) OpenCORE. These libraries are responsible for recording and playback of audio and video formats.

A library called Surface Manager controls access to the display system and supports 2D and 3D.

- The WebKit library is responsible for browser support; it is the same library that supports Google Chrome and Apple Inc.'s Safari.

The FreeType library is responsible for font support.

SQLite (<http://www.sqlite.org/>) is a relational database that is available on the device itself. SQLite is also an independent open source effort for relational databases and not directly tied to Android. You can acquire and use tools meant for SQLite for Android databases as well.

# Android S/W Stack - Runtime

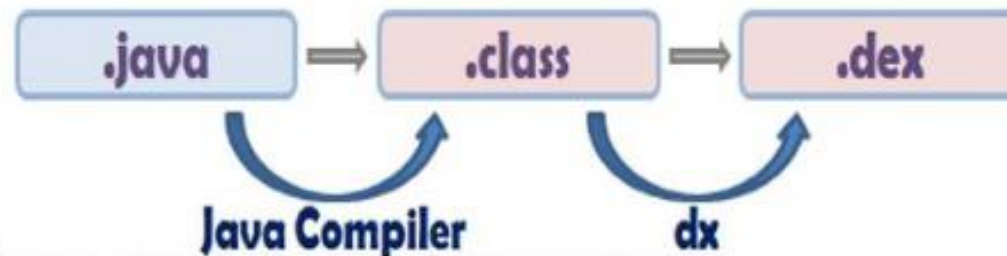


- Core Libraries

- ✓ Providing most of the functionality available in the core libraries of the Java language
- ✓ APIs
  - Data Structures
  - Utilities
  - File Access
  - Network Access
  - Graphics
  - Etc

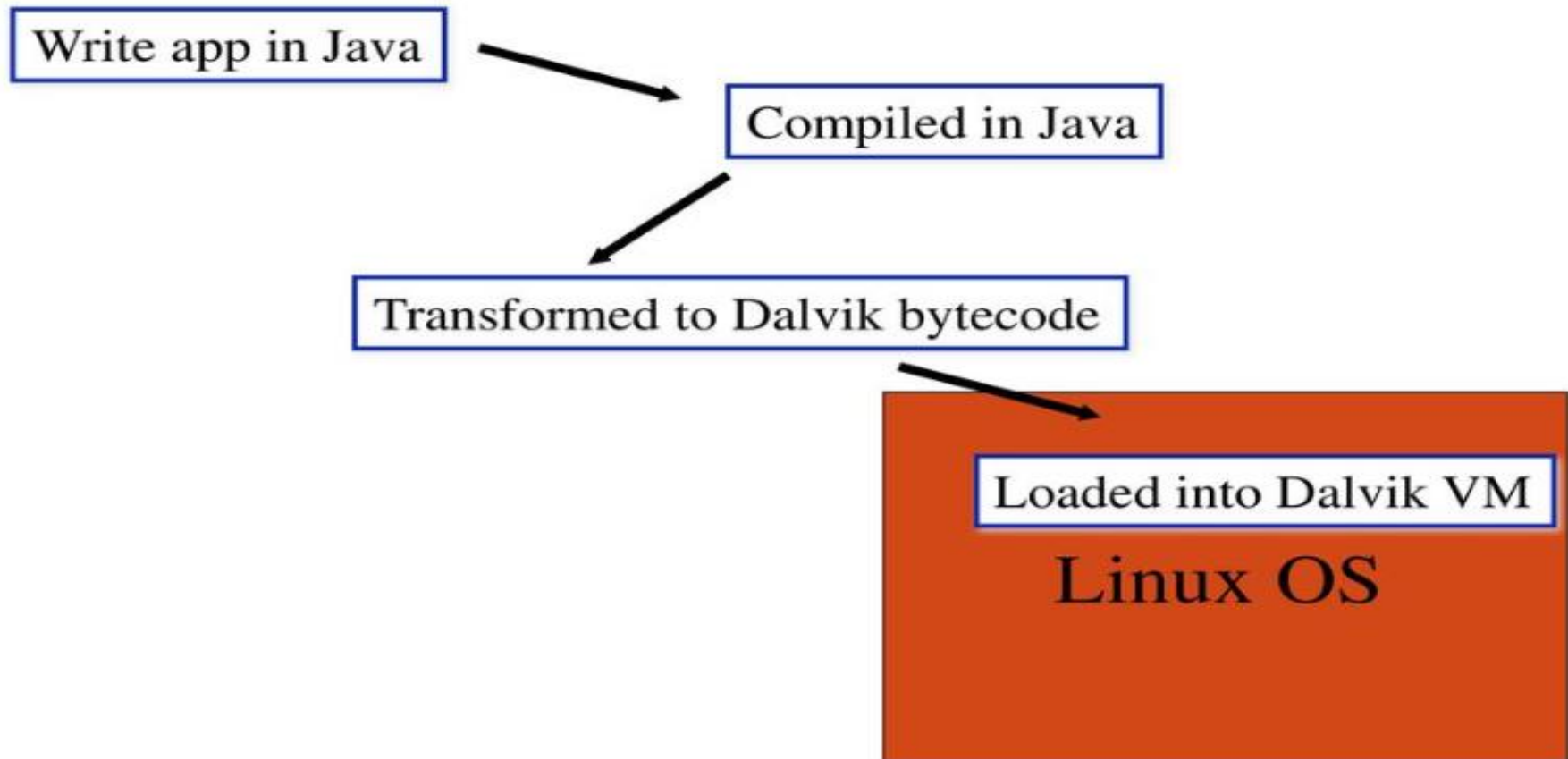
# Android S/W Stack – Runtime (Cont)

- Dalvik Virtual Machine (Cont)
  - ✓ Executing the Dalvik Executable (.dex) format
    - .dex format is optimized for minimal memory footprint.
    - Compilation



- ✓ Relying on the LINUX kernel for:
  - Threading
  - Low-level memory management

# Android applications are compiled to Dalvik bytecode



# Android S/W Stack – Linux Kernel



- Relying on Linux Kernel 2.6 for core system services
  - ✓ Memory and Process Management
  - ✓ Network Stack
  - ✓ Driver Model
  - ✓ Security
- The supplied device drivers include Display, Camera, Keypad, WiFi, Flash Memory, Audio, and IPC (interprocess communication).
- Providing an abstraction layer between the H/W and the rest of the S/W stack

# Platform

## Network Connectivity

It supports wireless communications using:

- GSM mobile-phone technology
- 3G
- Edge
- 802.11 Wi-Fi networks

## Software development

### Development requirements

- Java
- Android SDK

# Software development

## IDE and Tools

### Android SDK

- Class Library
- Developer Tools
- Emulator and System Images
- Documentation and Sample Code

### Eclipse IDE + ADT (Android Development Tools)

- Reduces Development and Testing Time
- Makes User Interface-Creation easier
- Makes Application Description Easier

# Advantages

Here are a few other advantages Android offers you as a developer:

- The Android SDK is available for Windows, Mac and Linux, so you don't need to pay for new hardware to start writing applications.
- An SDK built on Java. If you're familiar with the Java programming language, you're already halfway there.
- By distributing your application on Android Market, it's available to hundreds of thousands of users instantly. You're not just limited to one store, because there are alternatives, too. For instance, you can release your application on your own blog. Amazon have recently been rumoured to be preparing their own Android app store also.
- As well as the technical SDK documentation, new resources are being published for Android developers as the platform gains popularity among both users and developers.

# Application Building Blocks

- Activity
- IntentReceiver
- Service
- ContentProvider

## Activities

- Typically correspond to one UI screen
- But, they can:
  - Be faceless
  - Be in a floating window
  - Return a value

# Android Development Framework

- Android software Development Kit (SDK) includes :
  - Android APIs
  - Development Tools
  - The Android Virtual Device Manager and Emulator
  - Full Documentation
  - Sample Code
  - Online Support-  
<http://developer.android.com/support.html>

# DVM

Dalvik is open-source software. Dan Bornstein originally wrote Dalvik VM which is responsible for running apps on Android devices.

- It is a Register based Virtual Machine.
- It is optimized for low memory requirements.
- It has been designed to allow multiple VM instances to run at once.
- Relies on the underlying OS for process isolation, memory management and threading support.
- Operates on DEX files.
- All hardware and system service access is managed using Dalvik (middle tier)

# Development Environment

- JDK
- Software Development Kit (SDK)
- IDE – Eclipse
- Eclipse plug-in - ADT
- Android Emulator, Virtual Device Manager
- Debugger – Dalvik Debug Monitoring Service(DDMS)
- Android Asset Packaging Tool (AAPT) --- .apk
- Android Debug Bridge( ADB)

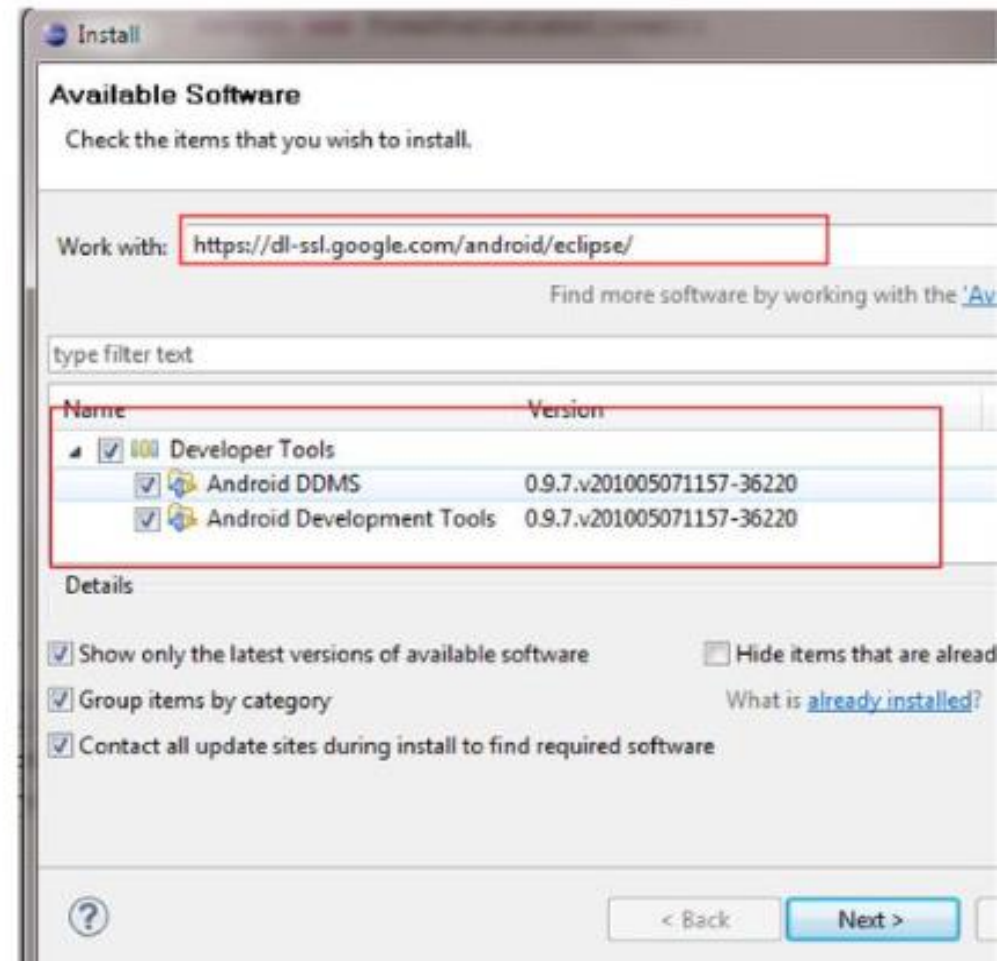
# Setup Android SDK

- Download Android SDK and extract the zip file to an arbitrary folder
  - <http://androidappdocs.appspot.com/sdk/index.html>
  - E.g.: extract to C:\
  - The SDK will be used by ADT in eclipse(located in adt-bundle../eclipse

Platform	Package	Size	MD5 Checksum
Windows	<a href="#">android-sdk_r06-windows.zip</a>	23293160 bytes	7c7fcec3c6b5c7c3df6ae654b27effb5
Mac OS X (intel)	<a href="#">android-sdk_r06-mac_86.zip</a>	19108077 bytes	c92abf66a82c7a3f2b8493ebe025dd22
Linux (i386)	<a href="#">android-sdk_r06-linux_86.tgz</a>	16971139 bytes	848371e4bf068dbb582b709f4e56d903

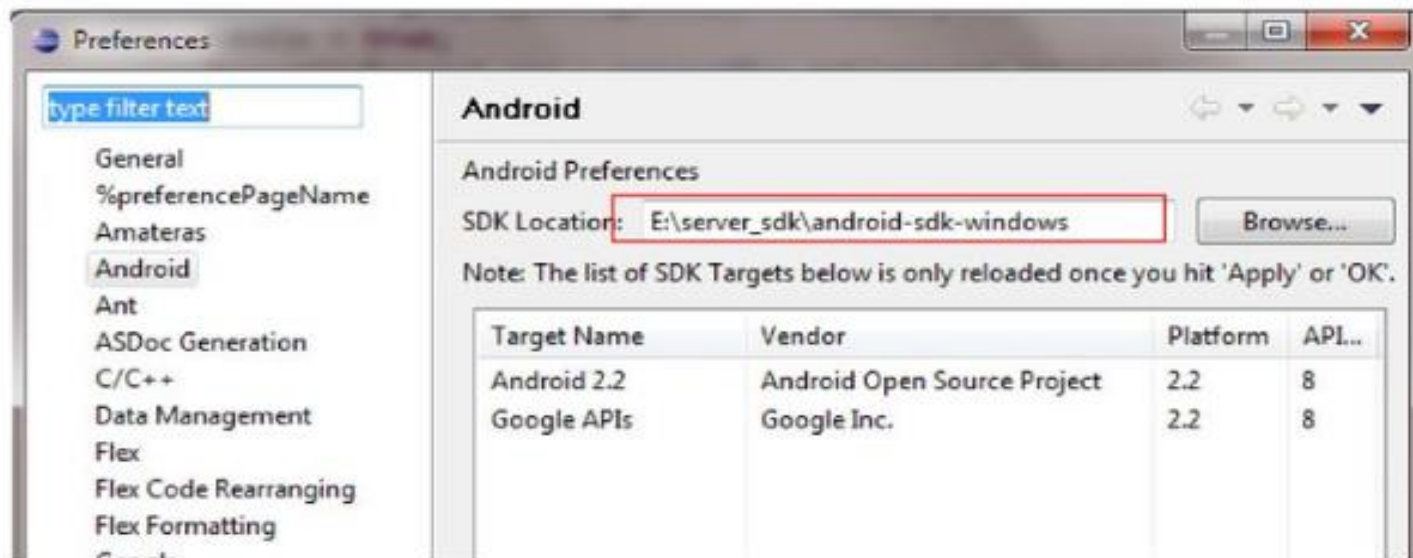
# Setup ADT plugin

- Install Eclipse ADT plugin
  - Eclipse must be J2EE edition
  - Update site: <https://dl-ssl.google.com/android/eclipse/>
- Install all the plugins in the repository
- Restart needed after installation



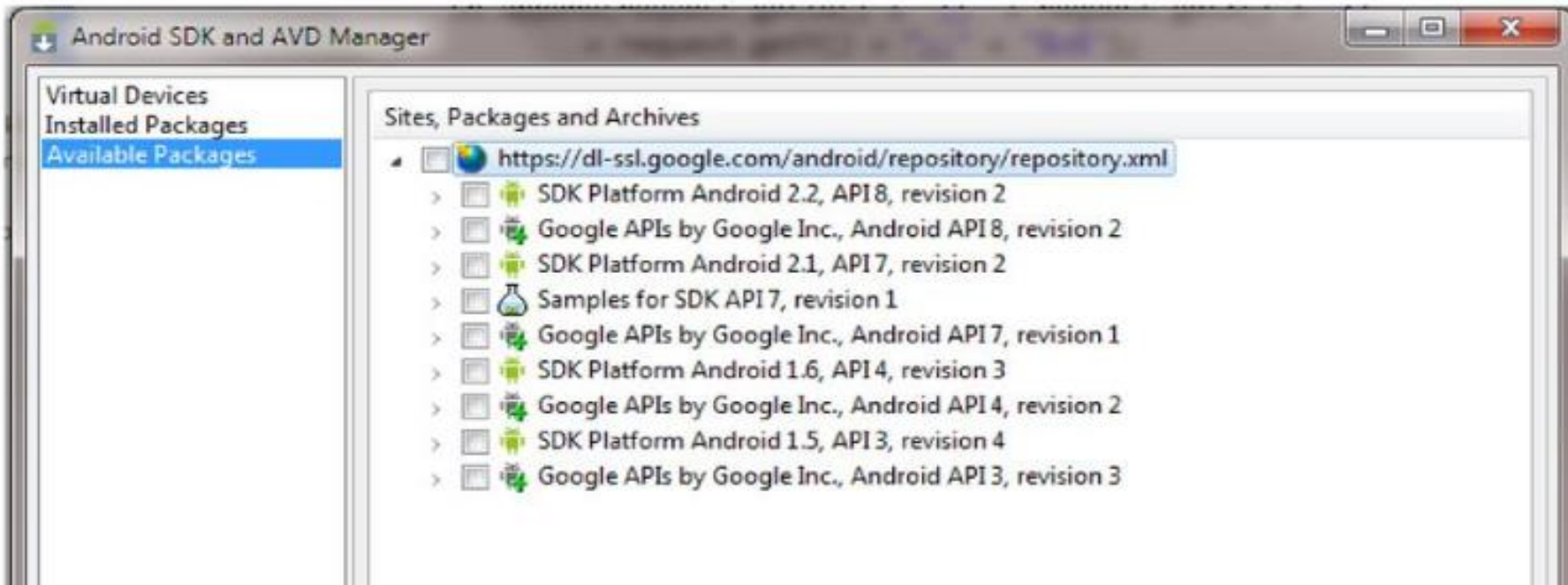
# Configure ADT Plugin

- Open eclipse Window->Preferences, select Android
- Setup the SDK location as the folder where you extracted the downloaded SDK zip file



# Setup SDK APIs

- Open Window->Android SDK and AVD Manager
- Click *Available Packages* and then choose proper APIs to install, the latest may be the best



An Android Virtual Device (AVD) is a configuration of emulator options

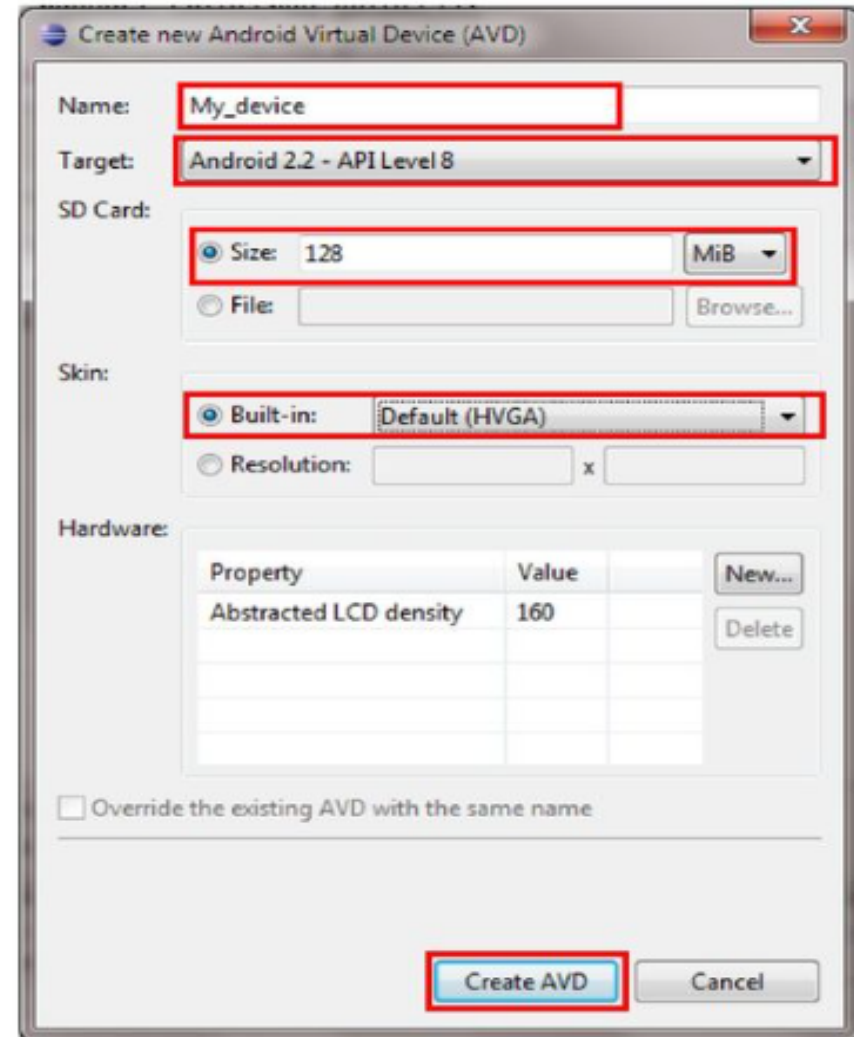
An AVD includes:

- A hardware profile
- A platform
- Other options
  - e.g., emulator skin, screen dimensions, SD card size

A storage area  
Create AVDs using Android SDK and AVD Manager tool

# Setup Emulators

- After SDK APIs installation, click *Virtual Devices*
- Click *new*, there will be a dialog
  - input a name
  - choose a running target and a skin
  - specify the SD card size



# Ready...

- Now you may start the AVD
  - Click start to start the new AVD
  - First start-up may take a **very** long time



## 1. DDMS

- What's happening under the surface
- Debugging tool -- to interrogate active processes, view stack, watch and pause active threads

## 2. ADB

- Client server app – to connect with android emulator.
- 3 components- daemon running in emulator
  - service that runs on your hardware
  - client app that communicate with daemon through service

# Android SDK Documentation

- Home
- SDK --- version and NDK, release notes
- Dev Guide -- Design and development
- Reference – Android APIs
- Resources – Technical articles and tutorials
- Videos
- Blog

# Developing for mobile devices

- Hardware imposed design consideration
- Be Efficient
- Expect Limited Capacity
- Design for small screens
- Expect low speeds, high latency
- At what cost

# Android Component's life cycle

# Android Components

- Broadcast Receivers
- Content Providers
- Services and
- Activities.

- During execution of onReceive() ---- alive
- As soon as *onReceive()* returns ---destroyed.
- Because of this short lifetime there are some restrictions on what you can do inside this function. For example, its invalid to use *bindService()* or *registerReceiver()* functions from inside *onReceive()*.

# Content Provider

- content providers are never destroyed, they exist for the entire lifetime of their process.
- It is considered alive on call to *onCreate()* and until process that contains this component is killed.
- *shutdown()*. ---introduced in API Level 11 and may be implemented by Content Provider. Unit tests may call it in order to guarantee unit test isolation.

## Service

- service is alive between calls to *onCreate()* and *onDestroy()* functions.

# Questions – Answer Session

