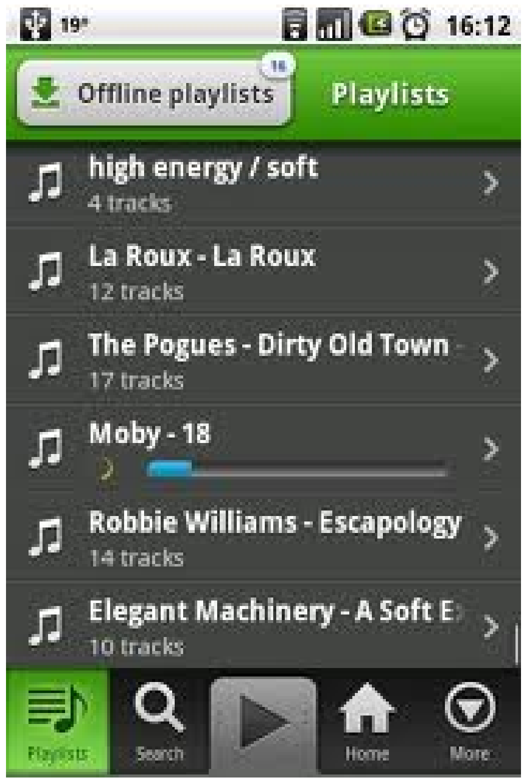
Mobila applikationer och trådlösa nät, HI1033, HT 2014

Today:
- Challengers with mobile services
- Platforms
- Android

What is a Mobile Service?



high energy / soft
4 tracks

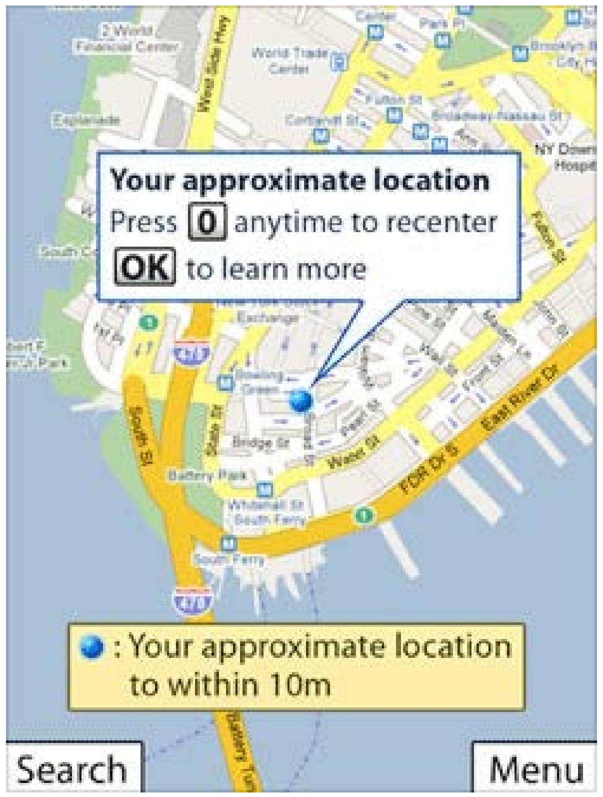
La Roux - La Roux
12 tracks

The Pogues - Dirty Old Town
17 tracks

Moby - 18

Robbie Williams - Escapology
14 tracks

Elegant Machinery - A Soft E
10 tracks

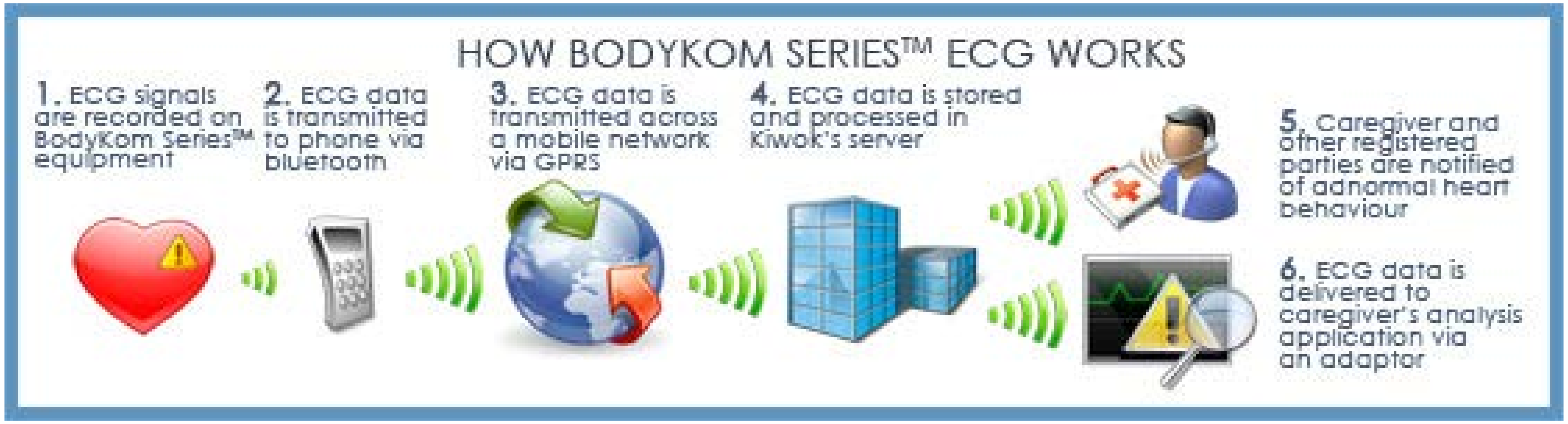


Your approximate location

Press 0 anytime to recenter

OK to learn more

: Your approximate location to within 10m



HOW BODYKOM SERIES™ ECG WORKS

1. ECG signals are recorded on BodyKom Series™ equipment

2. ECG data is transmitted to phone via bluetooth

3. ECG data is transmitted across a mobile network via GPRS

4. ECG data is stored and processed in Kiwok's server

5. Caregiver and other registered parties are notified of abnormal heart behaviour

6. ECG data is delivered to caregiver's analysis application via an adaptor

Mobile devices

Pico	Pocket	Palm	Pad	Lap	Desk
Sensor	*Mobile*	*PDA*	*E-reader*	Laptop	PC
Card	*Smartphone*	Net-book			
Pulsmeter, Glasses, …			*Tablet*		

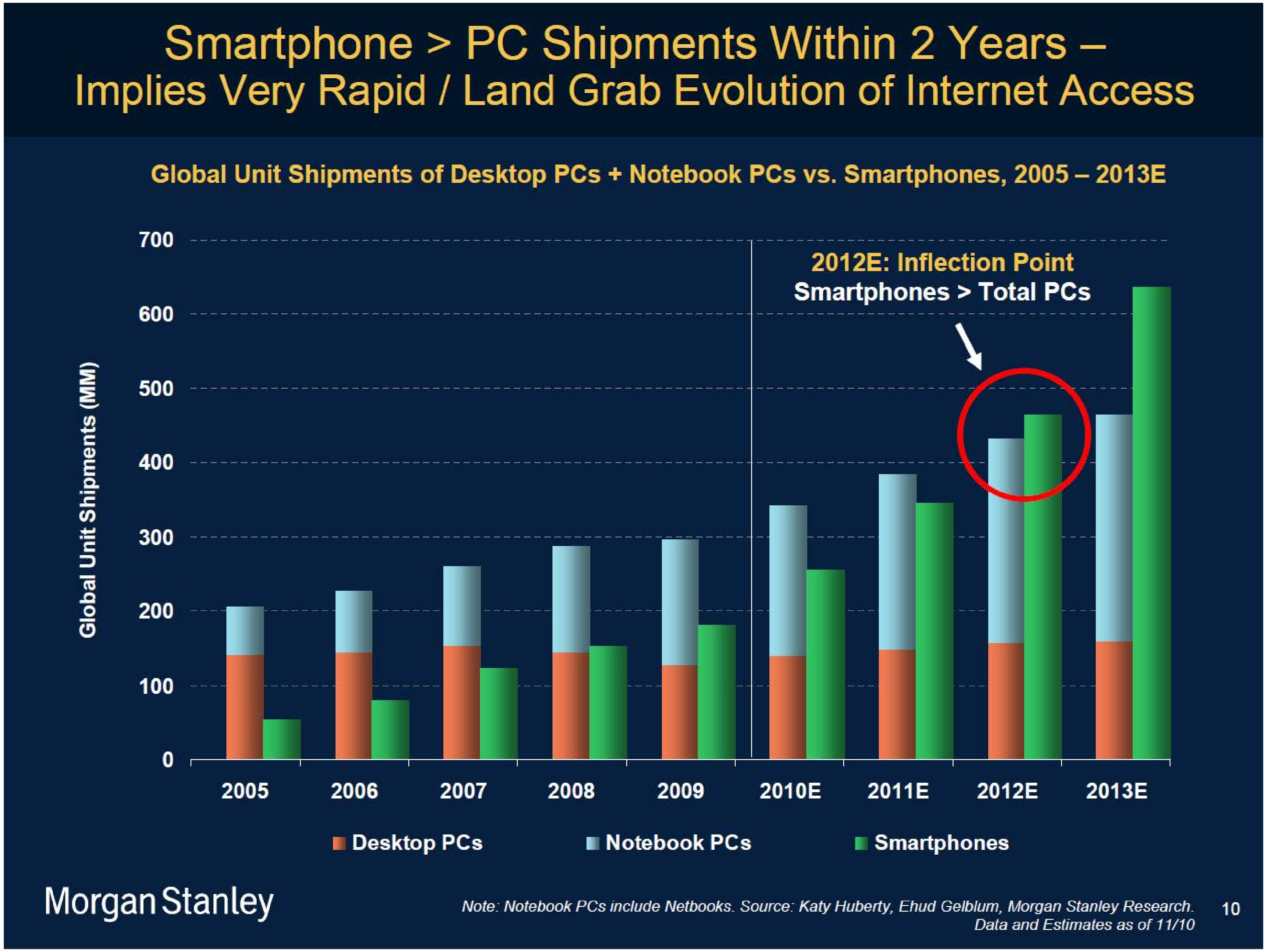
Smartphone vs Feature phone

- Smartphone - “A handheld computer integrated within a mobile telephone”

- Smartphones run complete operating system software providing a platform for application developers

- Common features (italics: usually *not* on a feature phone)
  - Play media
  - Connect to internet
  - *Touch screen*
  - *Hard/Soft keyboard*
  - Run third party software (e.g. J2ME or “apps”)
  - *Run third party software written in a native language*
  - *Additional devices like WiFi, GPS, accelerometer, …*
  - *Access to hardware*

Market



Smartphone > PC Shipments Within 2 Years –
Implies Very Rapid / Land Grab Evolution of Internet Access

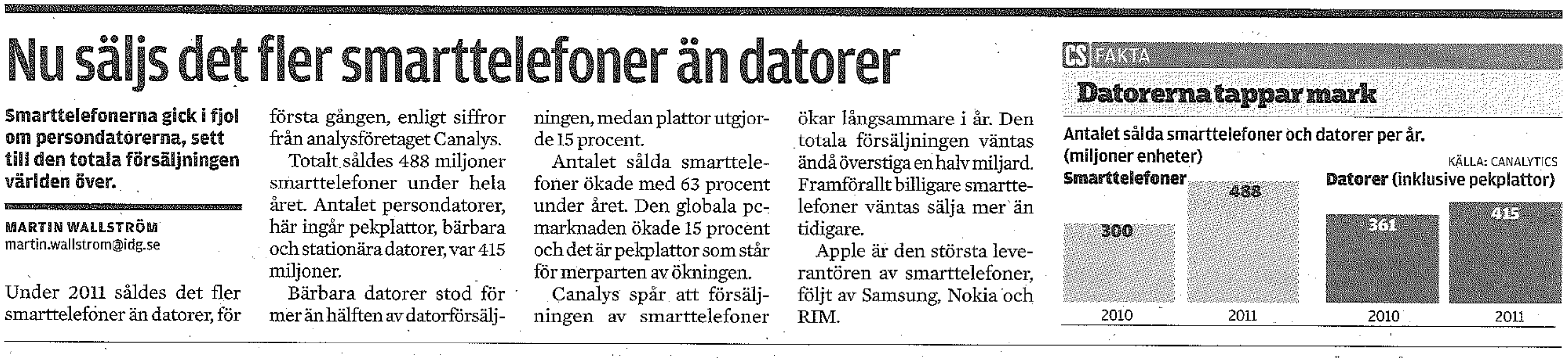
Global Unit Shipments of Desktop PCs + Notebook PCs vs. Smartphones, 2005 – 2013E

Morgan Stanley

Note: Notebook PCs include Netbooks. Source: Katy Huberty, Ehud Gelblum, Morgan Stanley Research. Data and Estimates as of 11/10

10

Market



Nu säljs det fler smarttelefoner än datorer

Smarttelefonerna gick i fjol om persondatorerna, sett till den totala försäljningen världen över.

MARTIN WALLSTRÖM
martin.wallstrom@idg.se

Under 2011 såldes det fler smarttelefoner än datorer, för första gången, enligt siffror från analysföretaget Canalys.

Totalt såldes 488 miljoner smarttelefoner under hela året. Antalet persondatorer, här ingår pekplattor, bärbara och stationära datorer, var 415 miljoner.

Bärbara datorer stod för mer än hälften av datorförsäljningen, medan plattor utgjorde 15 procent.

Antalet sålda smarttelefoner ökade med 63 procent under året. Den globala pc-marknaden ökade 15 procent och det är pekplattor som står för merparten av ökningen.

Canalys spår att försäljningen av smarttelefoner ökar långsammare i år. Den totala försäljningen väntas ändå överstiga en halv miljard. Framförallt billigare smarttelefoner väntas sälja mer än tidigare.

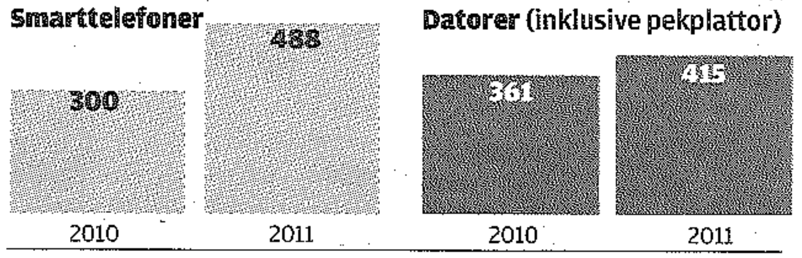
Apple är den största leverantören av smarttelefoner, följt av Samsung, Nokia och RIM.

CS FAKTA

Datorerna tappar mark

Antalet sålda smarttelefoner och datorer per år. (miljoner enheter)

KÄLLA: CANALYTICS



Computer Sweden, februari 2012

Some platforms

- Symbian OS (derived from EPOC). API: C++.
  Open source, today maintained by Nokia (no new models after 2013)

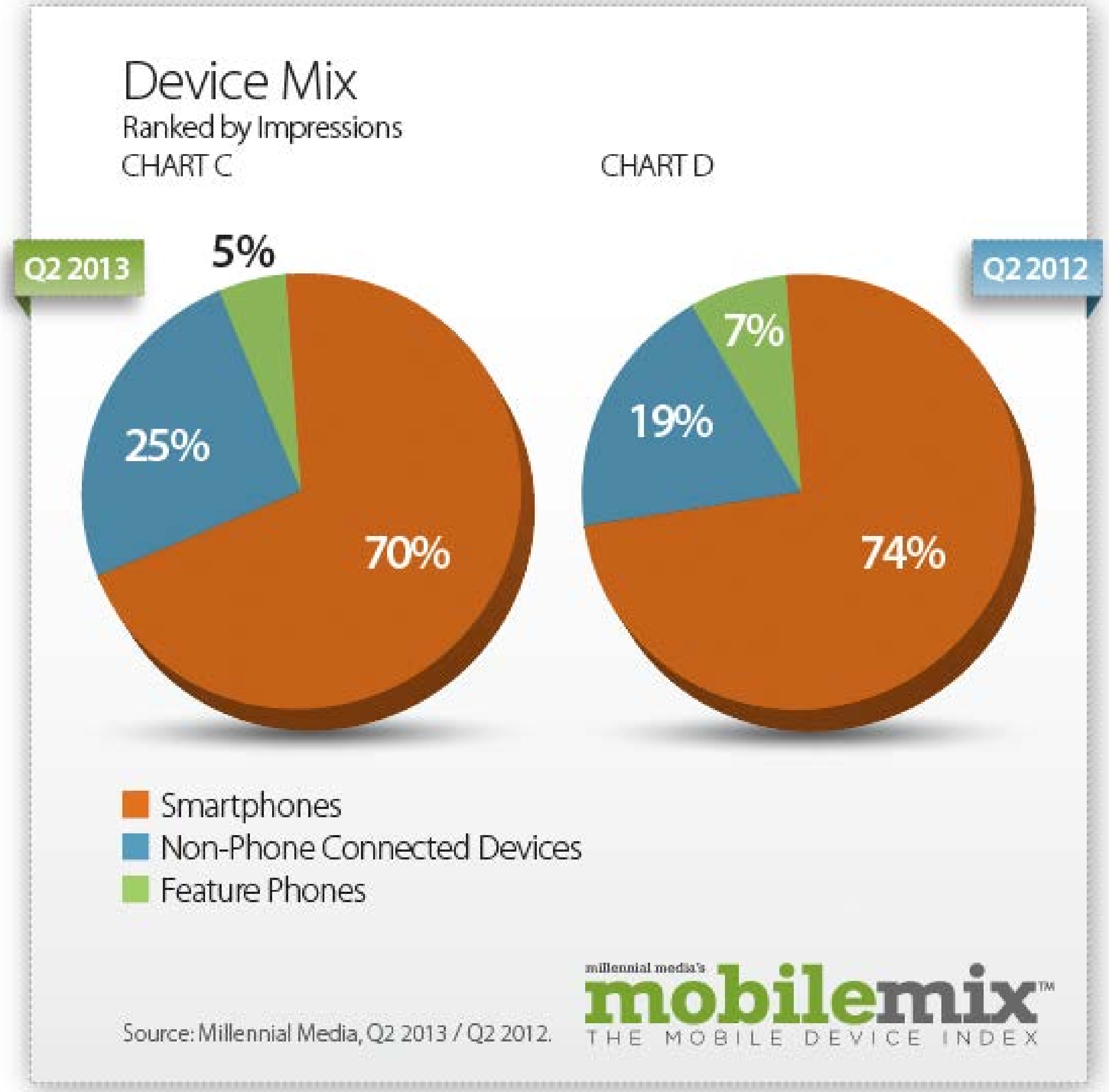
- Java Micro Edition: Cross platform; runs on a virtual machine on top of other OS.
  Designed for embedded systems. Down-scaled Java API.

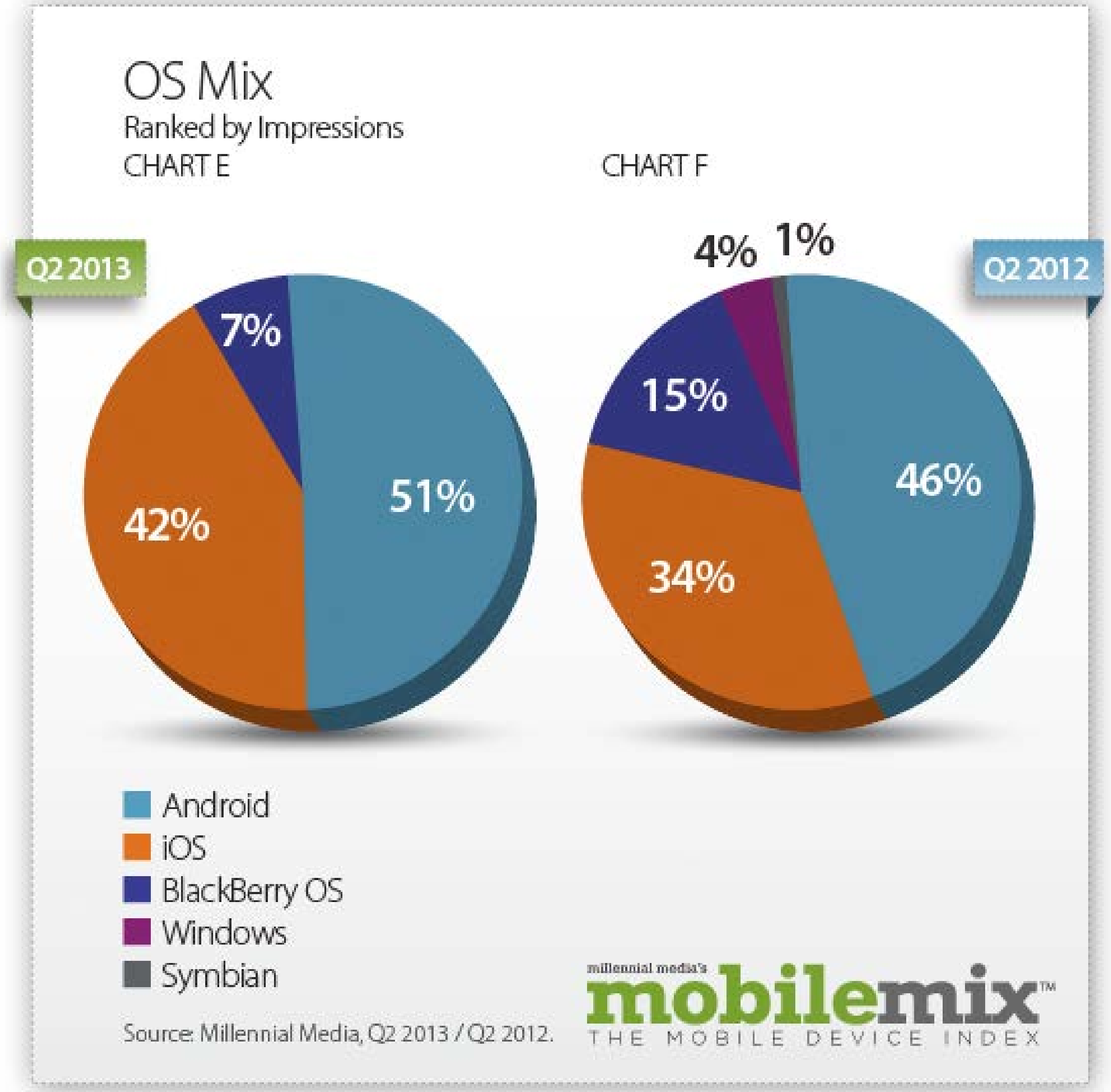
- iPhone and iPad running on iOS (derived from Mac OS X, Unix-like).
  Application programming: Objective C.

- Android. Linux kernel + Dalvik Virtual Machine running applications.
  Application programming : Java dialect.
  Open source, maintained by Open Handset Alliance

- Windows Phone - operating system with 3rd party and Microsoft services.
  Application programming : C#, C++, Visual Basic

Smartphone platforms , 2013





Source: http://www.millennialmedia.com/mobile-intelligence/mobile-mix/

Market; App Stores

- Revolution in distribution of mobile applications.

- Applications available for download “over the air” (June 2011)
  - App Store: 400 000 (from approximately 30 000 developers)
  [2012: over 1.1 million apps]
  - Google Play (former Android Market): 400 000
  [2012: 1.3 million apps, over 50 billion downloads]
  - Windows Phone Marketplace: > 20 000
  [2012: 100 000 apps]

- App Store 2009:
  Every app store user spends an average of €4.37 every month.
  There is over 58 million app store users.

- Advertising…

Typical Smartphone specs (as of Jan 2013)

	iPhone 5	**Samsung Galaxy S III**	**Typical PC**
Mass storage	16-64 GB	16-64 GB (microSD, up to 64 GB)	1 TB
RAM	1 GB	1 GB	8-16 GB
Processor	Dual-core 1.2 GHz	Quad-core Cortex-A9 1.4 GHz	3-3.5 GHz*
Battery Stand by/Talk	*300 hours/420 minutes*	*220 hours/480 minutes*	*-*

Expect this when developing software for limited devices (such as smartphones)

- Limited memory capacity and processor speed
  Limited battery capacity

- Network: High latency, low speeds, possibly interrupted
  Communication (might) be associated with a cost!

- Small screens, of different sizes and different densities

- Application might get interrupted at any time!

- Hardware-imposed design considerations
  Fragmentation of platforms

- *Design with this in mind:*
  Be efficient and be responsive

What’s consuming memory, processor resources and battery capacity?

- Memory
  - Unnecessary allocation of objects
  - Inefficient data structures
  - Size of application code(!)
  - Multiple processes

- Processor recources
  - Inefficient algorithms
  - Garbage Collection(!)
  - Multiple processes and threads
  - Rendering of GUI
  - Unnessecary polling

- Battery
  - Processor working
  - Network communication, especially when using WiFi and Bluetooth

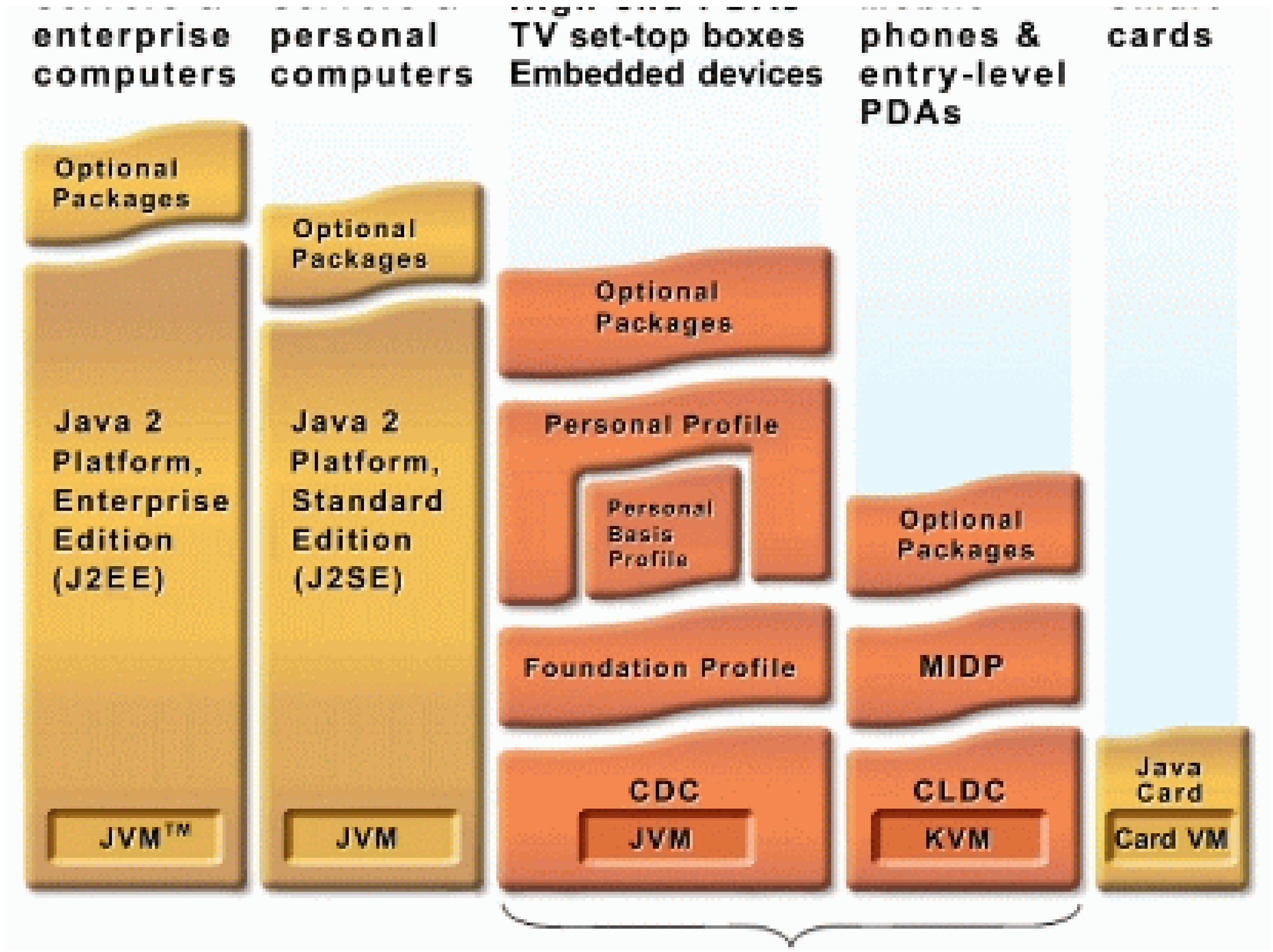
Challenges with mobile data

- Low bandwidth, Frequency vs. Bandwidth
    - GSM, GPRS, EDGE, 3G/4G, WLAN, LAN
    - Wireless connection using different networks
- Datacom vs. Telecom - Best effort vs. Quality of Service
    - Cost and distance
    - Push vs. Pull
- Question regarding benefit, design and standards

Java Micro Edition

- In the middle of the 90s OAK was developed (Java predecessor)
  1999 Palm included KVM (Kilobyte Virtual Machine)

- Supposed to work on:
  - Feature phones and PDA
  - set-top boxes, TV
   and other embedded devices
  - smart cards



CLDC and CDC

- Two different Java ME configurations:
  - CLDC (Connected Limited Device Configuration) Focus on the most limited devices
  - CDC (Connected Device Configuration) Devices that almost handle a complete Java environment
- Why:
  - One common ground for similar devices
  - Keep “core” API’s between different devices
  - Define requirements on virtual machines

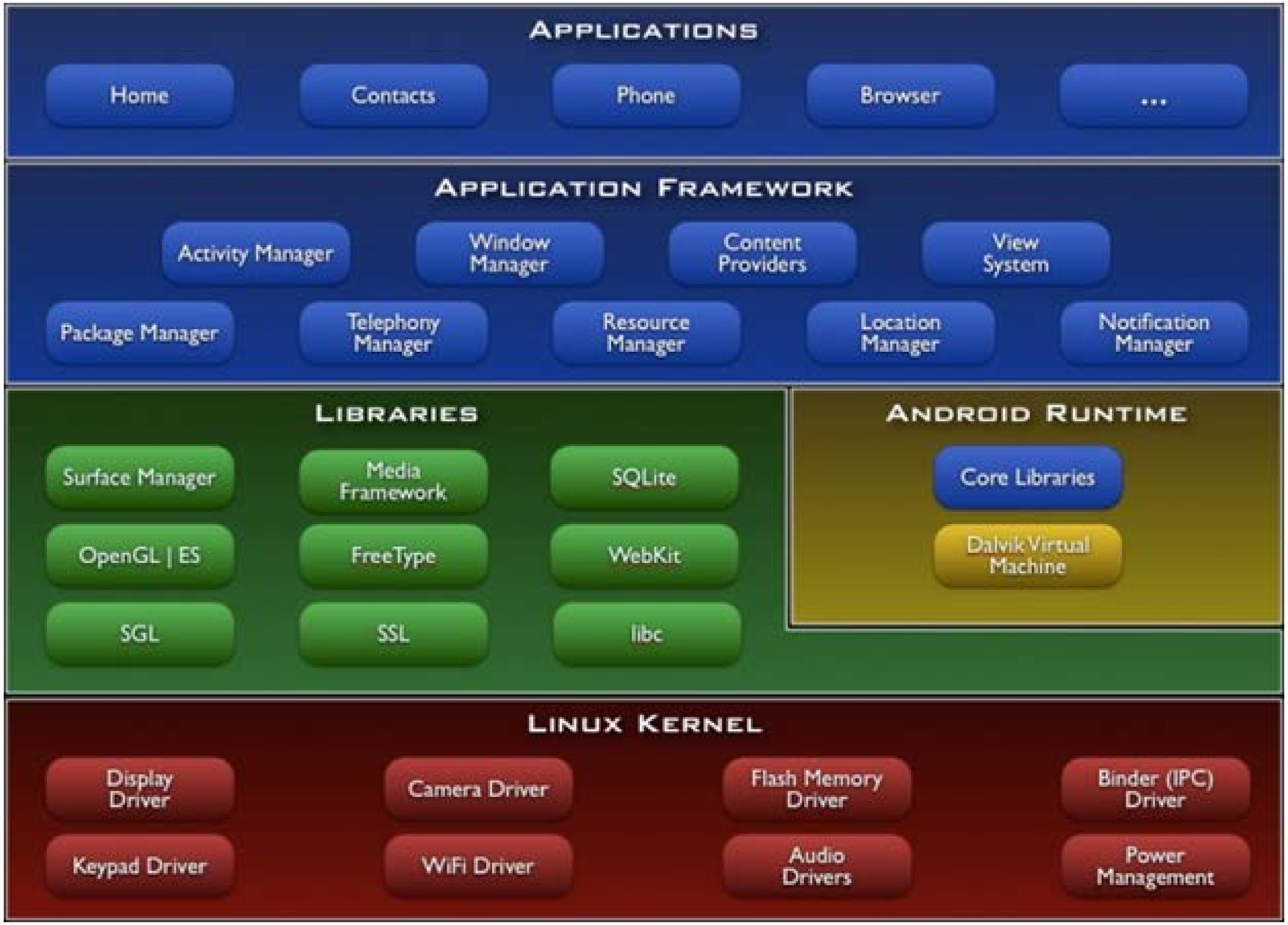
Java Micro Edition

- There are billions of Java ME enabled mobile phones and PDAs
- Java ME might become an old technology, as it is not used on any of today's newest mobile platforms;
  e.g. iPhone, Android, Windows Phone 7, BlackBerry's new QNX
- http://www.oracle.com/technetwork/java/javame/overview/index.html

At last…

- Android is: A mobile device platform including an OS based on the Linux kernel, middleware and key applications
- Designed to support many different hardware devices
- Applications run on the Dalvik Virtual Machine
- An extensive API, including most Java SE classes, for 3rd party application development
- Available under a free software / open source license (no license cost) Standard maintained by Open Handset Alliance, a consortium including Texas Instruments, Google, HTC, Intel, Motorola, SonyEricsson, Samsung, ...

The Android Software Stack



The Dalvik VM

- Every Android application runs in its own process, with its own instance of the Dalvik virtual machine.
  - Android launches a process when any of the application's code needs to be executed (if not already running).
  - And, yes, Dalvik is compact and efficient - a device can run multiple VMs in parallel.
  - *By default no “exit application”.*
    The process is shut down when *it's no longer needed and system resources are required by other applications – unpredictable!*
- The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint
- JIT, Just-In-Time compilation enhance performance (since Android 2.2)

ART

- A new Android run-time, introduced experimentally in the version 4.4 (not the default run time)

- Features
  - Ahead-of-time (AOT) compilation - at install time, ART compiles apps using the on-device dex2oat tool (.dex to .oat)
  - Improved garbage collection – e.g. parallelized processing during part of the GC pause
  - Development and debugging improvements

Android applications

- Android applications don't have a single entry point (no main method)
  Instead: The application consists of one or more essential *components* which the system can instantiate and run as needed

- **Activities** holding View components and references to the model; also entry point for user actions

- **Services** doesn't have a visual user interface, but rather runs tasks in the background

- **Broadcast receivers** receive and react to broadcast announcements, e.g. battery is low, e-mail received, …

- **Content providers** makes a specific set of the application's data available to other applications

Android applications, Activities

- When the first of an application's components needs to be run, Android starts a Linux process for it with a *single thread of execution*.
  By default, all components of the application run in that process and thread.

An activity has essentially three states:

- *active* or *running* when it is in the foreground
- *paused* if it has lost focus but is still visible to the user
- *stopped* if it is completely obscured by another activity

Android applications, Activities

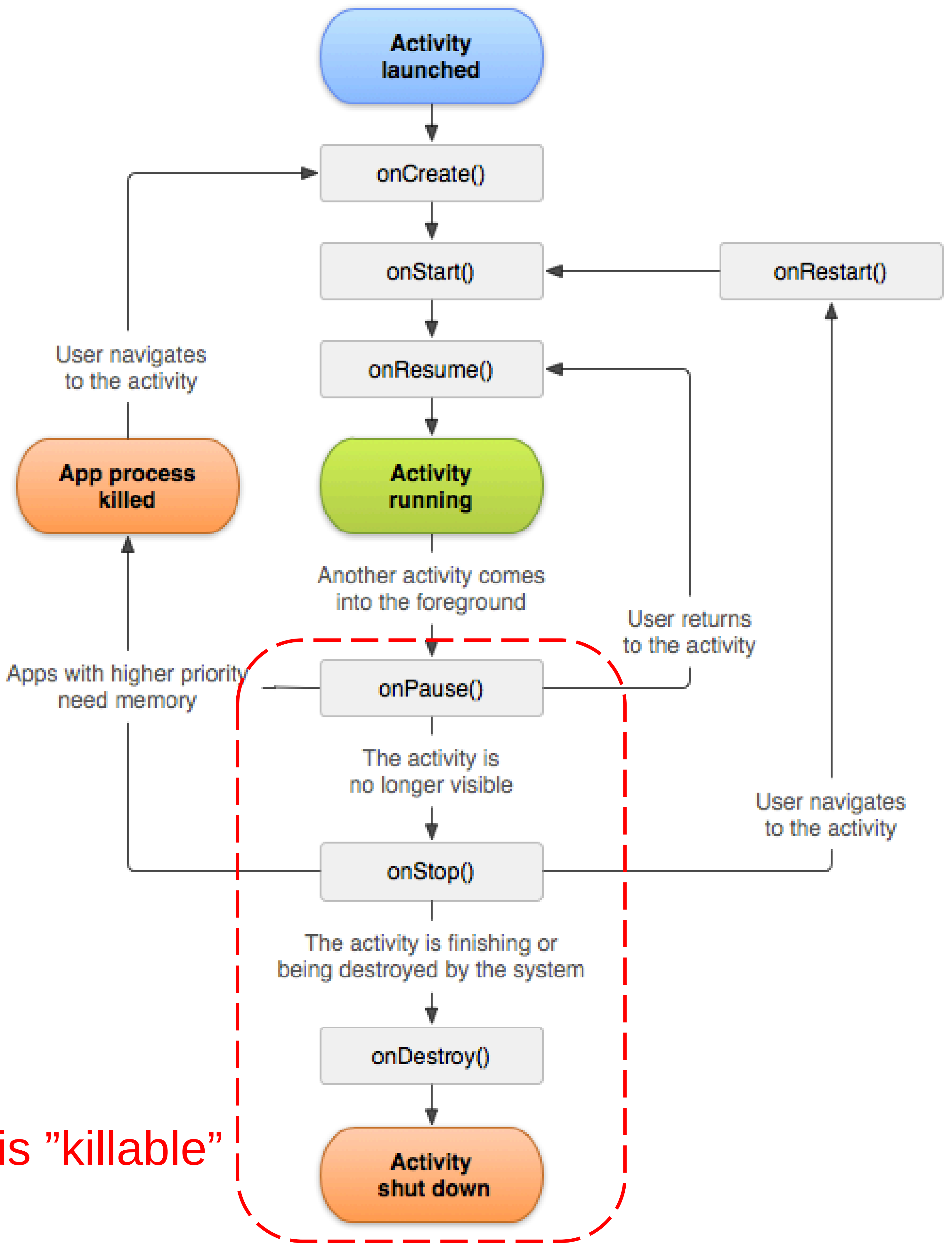
- A paused or stopped activity retains all state and member information, however…

- *…the system may kill the process running the activity from memory when memory is needed elsewhere*

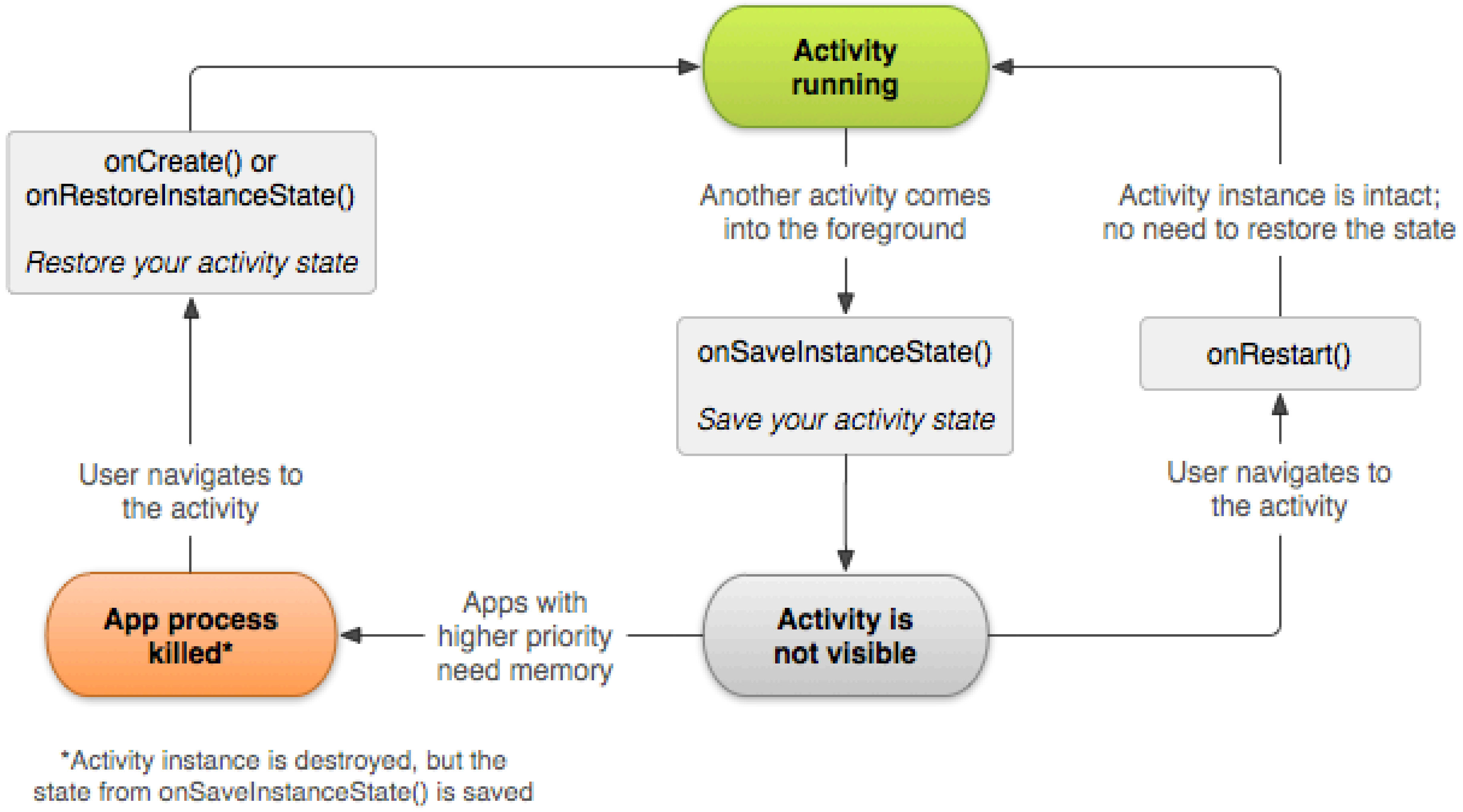
- As an activity transitions from state to state, it is notified of the change by calls to the following protected methods:

-
  ```
  void onCreate(Bundle savedInstanceState)
  void onStart()
  void onRestart()
  void onResume()
  void onPause()
  void onStop()
  void onDestroy()
  ```

Activity lifecycle


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down
process is ”killable”

Saving Activity (UI) state

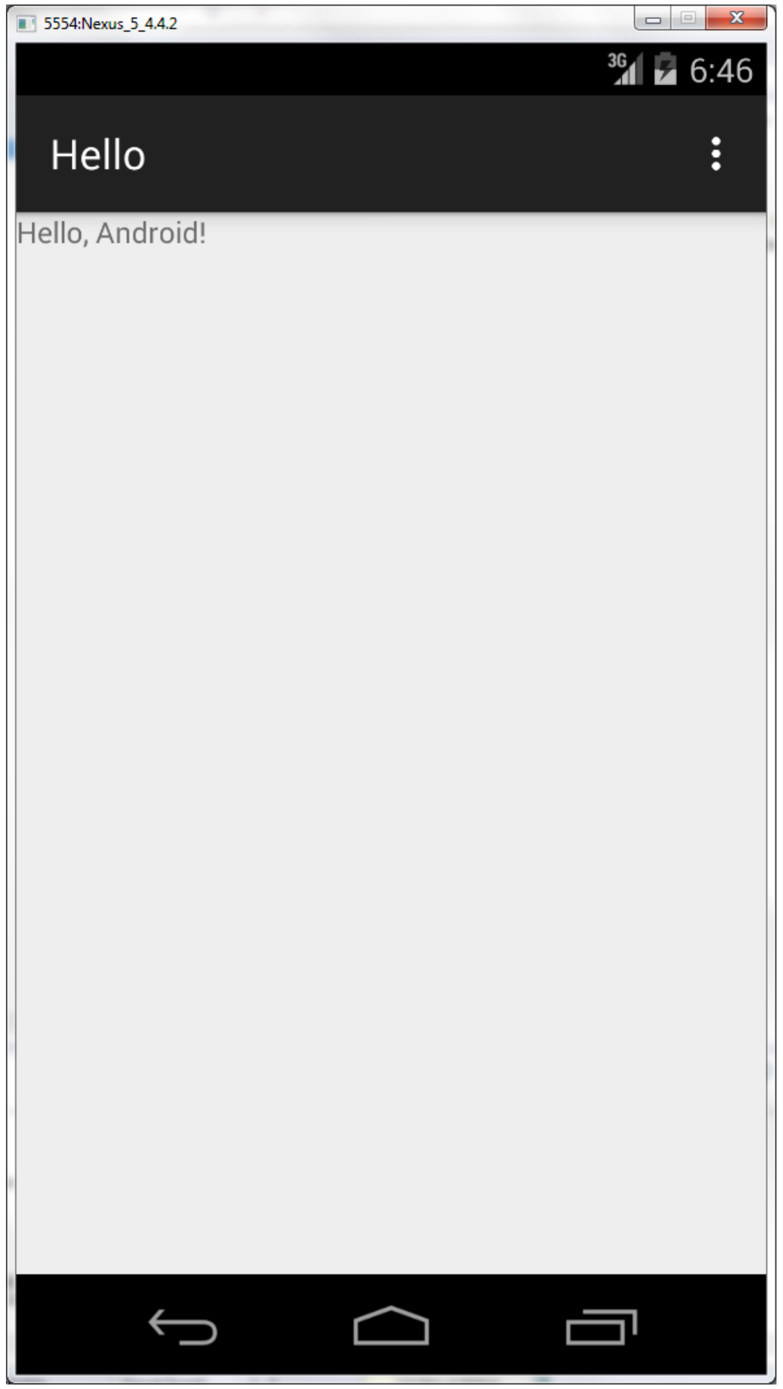

Activity running
onCreate() or onRestoreInstanceState()
Restore your activity state
Another activity comes into the foreground
Activity instance is intact; no need to restore the state
onSaveInstanceState()
Save your activity state
onRestart()
User navigates to the activity
User navigates to the activity
App process killed*
Apps with higher priority need memory
Activity is not visible
*Activity instance is destroyed, but the state from onSaveInstanceState() is saved

Android applications, Activities

```
package se.kth.anderslm.hello;

import android.app.Activity;
import . . .;

public class HelloAndroid
                extends ActionBarActivity {
    // Called when the activity is first created
    @Override
    public void onCreate(
                Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android!");
        setContentView(tv);
    }
    . . .
}
```



Hello, Android!

Preferable: Layout defined in res/layout/activity_main.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft= . . .
    . . .
    tools:context="se.kth.anderslm.hello.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

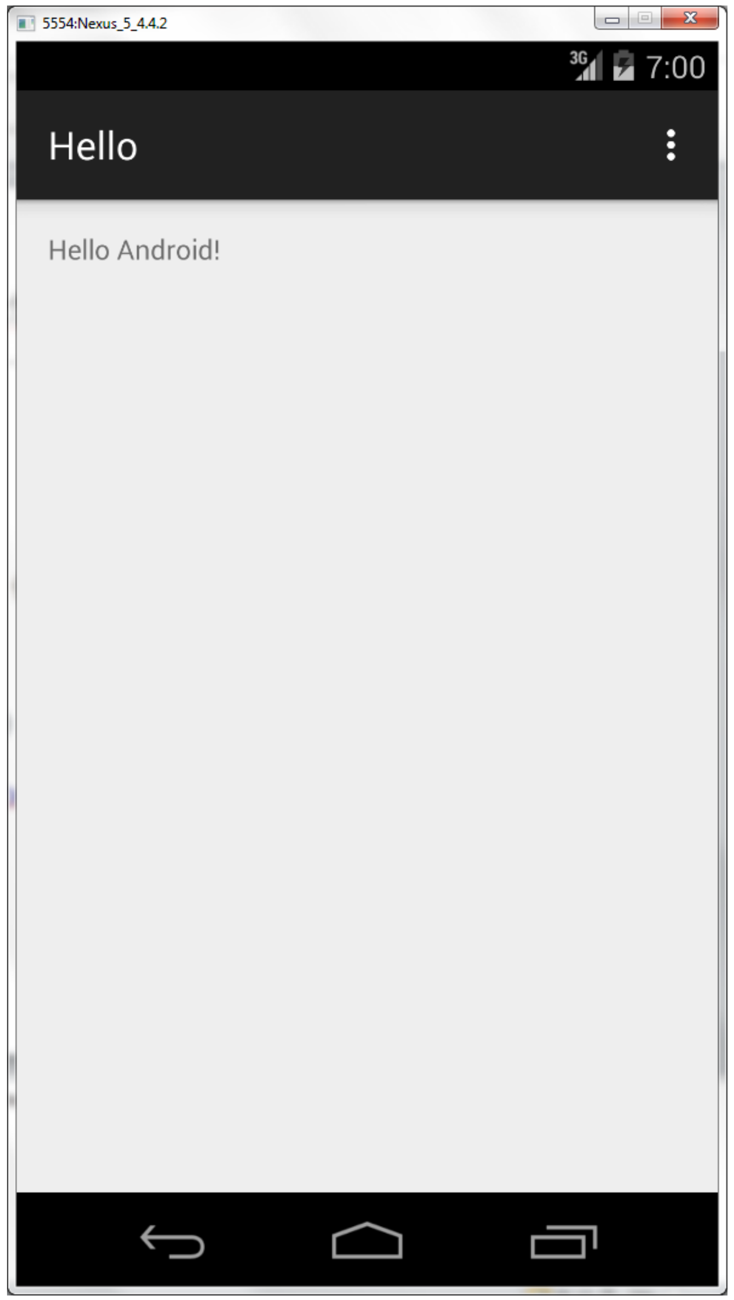
Cont: Load/inflate layout in Actvivity.onCreate

```
package se.kth.anderslm.hello;

import . . .;

public class MainActivity
                extends ActionBarActivity {

   @Override
   protected void onCreate(
                Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.setContentView(R.layout.activity_main);
   }
   . . .
}
```



Android from the perspective of the developer

- High level Java APIs for accessing hardware such as camera, GPS, accelerometer – same interface for different devices

- Native and 3rd party applications are treated equal. You may
  - replace native applications
  - access the same underlying data and hardware
  - use components of native applications

- Reuse of application components (Activities) in other applications possible

- Support for background services

- WebKit, persistent storage using SQLite, OpenGL, …

Android from the perspective of the developer

APIs including

- WiFi hardware access. GSM and 3G for telephony or data transfer
- GPS
- Bluetooth
- HTML 5 WebKit-based browser
- Hardware accelerated graphics (if possible) including OpenGL
- And more…

Some ”Designing For Performance” guide lines

- Memory management
  - Avoid creating unnessecary objects
  - When concatenating text in a loop – use a StringBuffer instead of Strings
- Minimize (virtual) method calls
  - Avoid internal use of getters and setters
  - Declare methods that don’t access member fields as ”static”
- Use the ”for-each” loop except for arrays and ArrayLists
- Know and use the API-libraries – they are probably more efficient than your custom code (e.g. animations)
- Use events +callbacks methods instead of polling for data

Android – where to go from here?

- This is where you find it all:
  http://developer.android.com/index.html
- Introduction to Android the Android platform “Androidology” part 1, 2 and 3 on Youtube
- More on developing for performance:
  Meier, pp 38-47
  http://developer.android.com/guide/practices/design/performance.html