Java Mobile

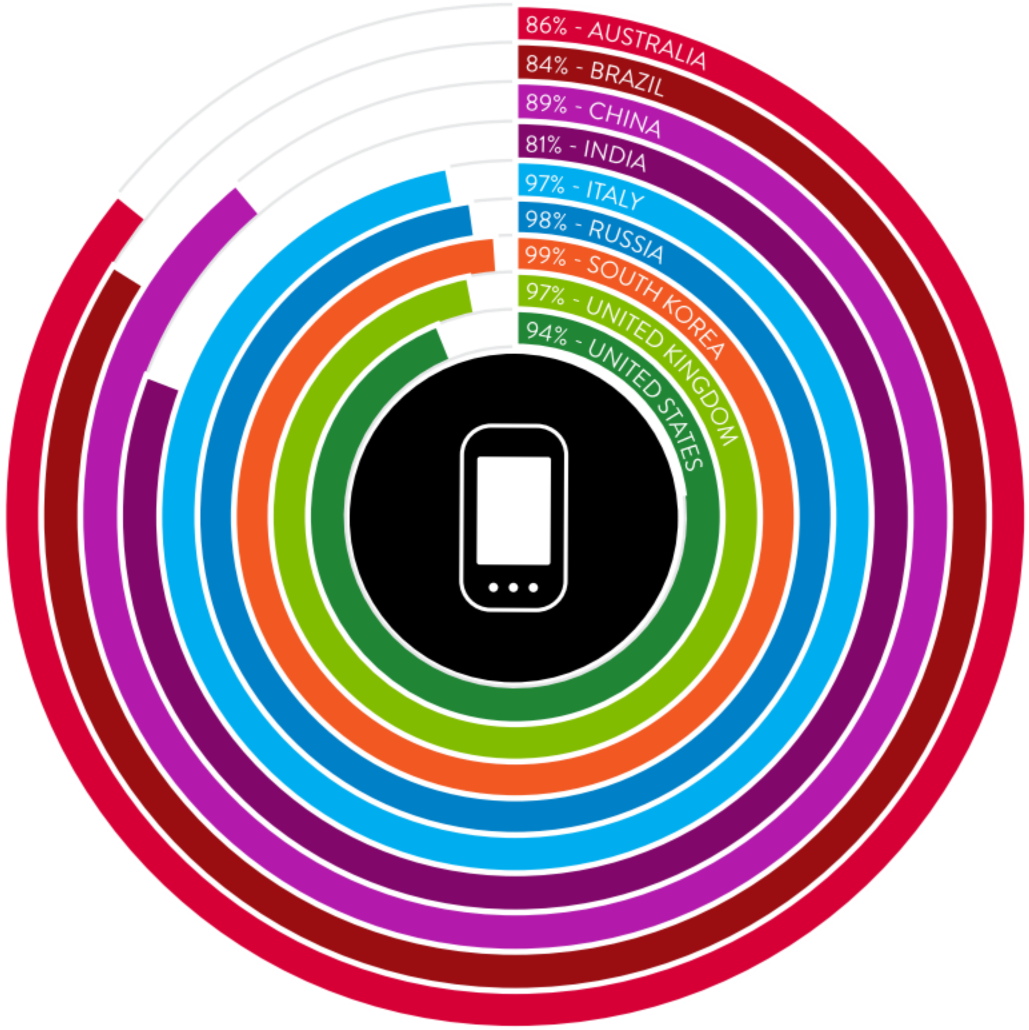
Mobile App Development

- Windows Mobile
  - Developing Languages: C++ or .NET
  - Distribution: free distribution – normal application or market
  - Developing Platform: Windows PC
  - Licence: proprietary
- Android
  - Developing Languages: **Java**
    - Android only reuses the Java language syntax and semantics, but does not provide the full class libraries and APIs bundled with Java SE or ME
  - Distribution: market ($25 one-time fee) or normal applications
  - Developing Platform & Licence: open source

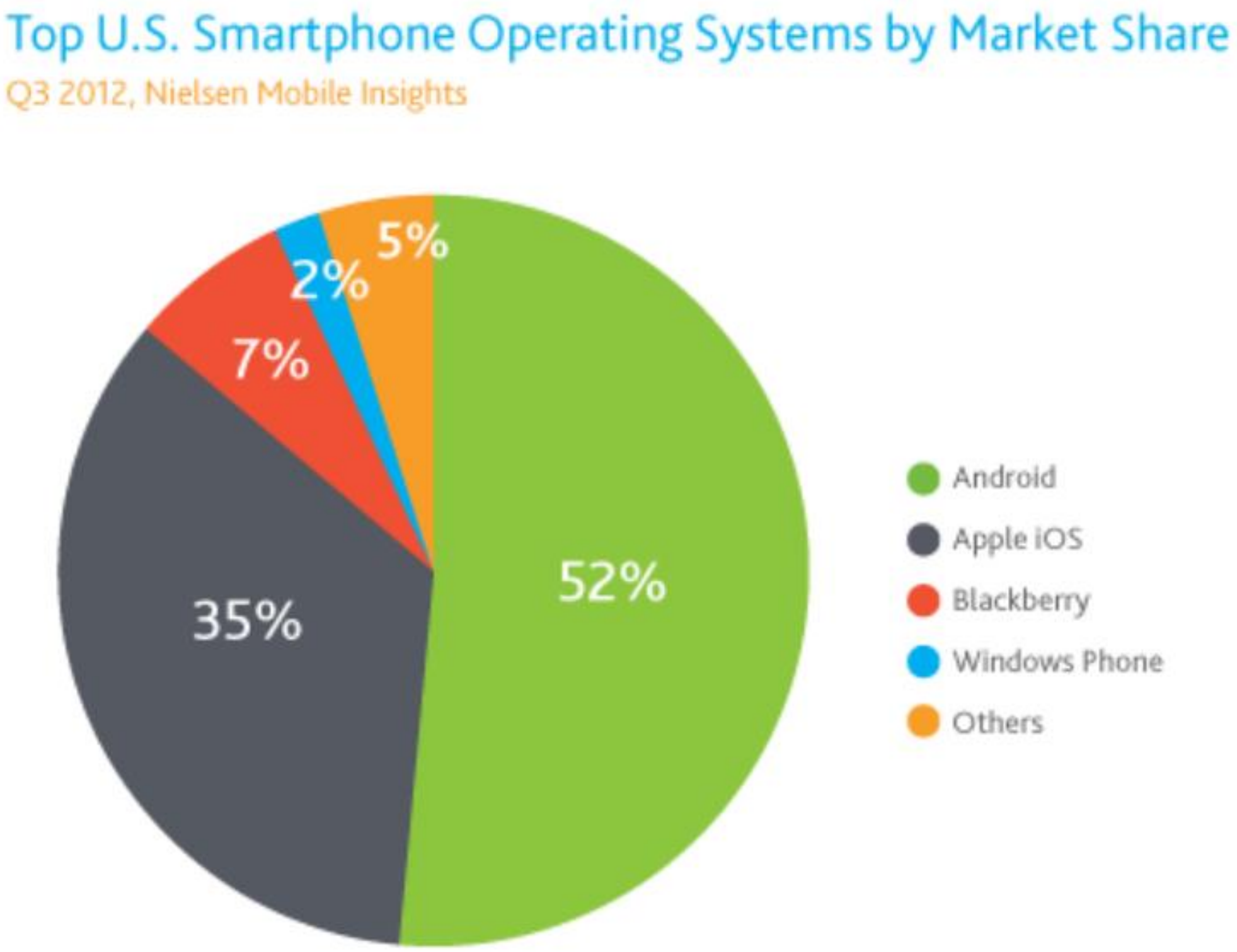
Mobile App Development

- iPhone
  - Developing Languages: Objective-C (& Java possible)
  - Distribution: market ($99/year fee)
  - Developing Platform: Mac OS X
  - Licence: proprietary
- Java ME
  - Developing Languages: **Java**
  - Distribution & Developing Platform & Licence: open source
    - Java ME (or an OS- and device-specific version of Java ME) comes pre-installed with Symbian, BlackBerry, Windows Mobile
- Symbian
- WebOS

Mobile Usage

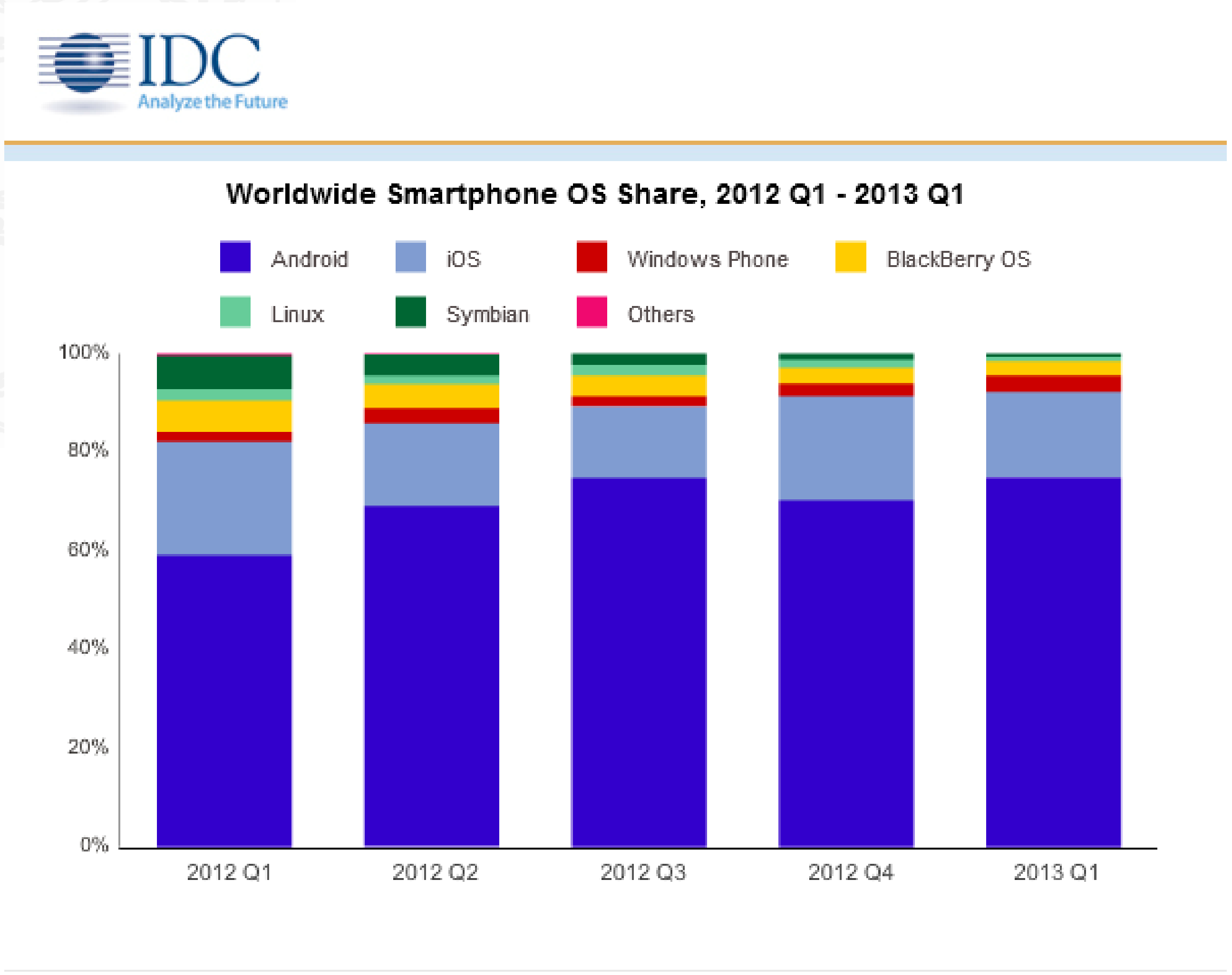


Smartphone Share

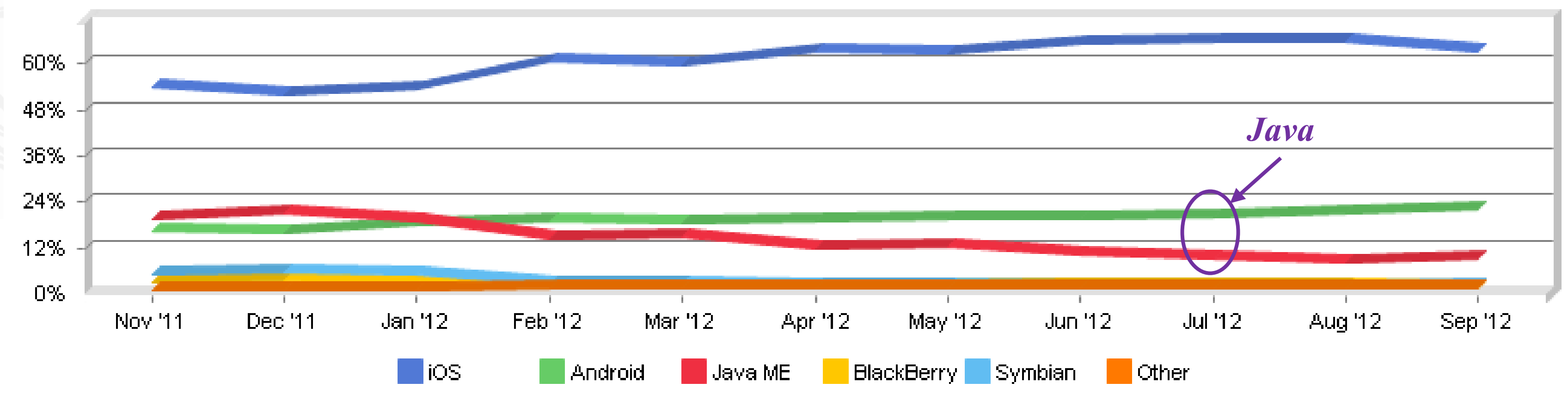


UNIVERSITY OF WOLLONGONG

Worldwide Smartphone OS Share



Mobile/Tablet Development Platform Trend



Source: http://www.netmarketshare.com

UNIVERSITY OF WOLLONGONG

Java Micro Edition

- J2ME (Java 2 Platform, Micro Edition) combines a *resource constrained JVM* and a set of APIs for developing applications for mobile devices and embedded systems
  - Mobile phones, PDAs, TV set-top boxes, printers ...
- A collection of technologies and specifications that can be combined to construct a complete Java runtime environment specifically to fit the requirements of a particular device or market
  - Java for Mobile Devices
  - Java Embedded
  - Java TV
  - Java Card

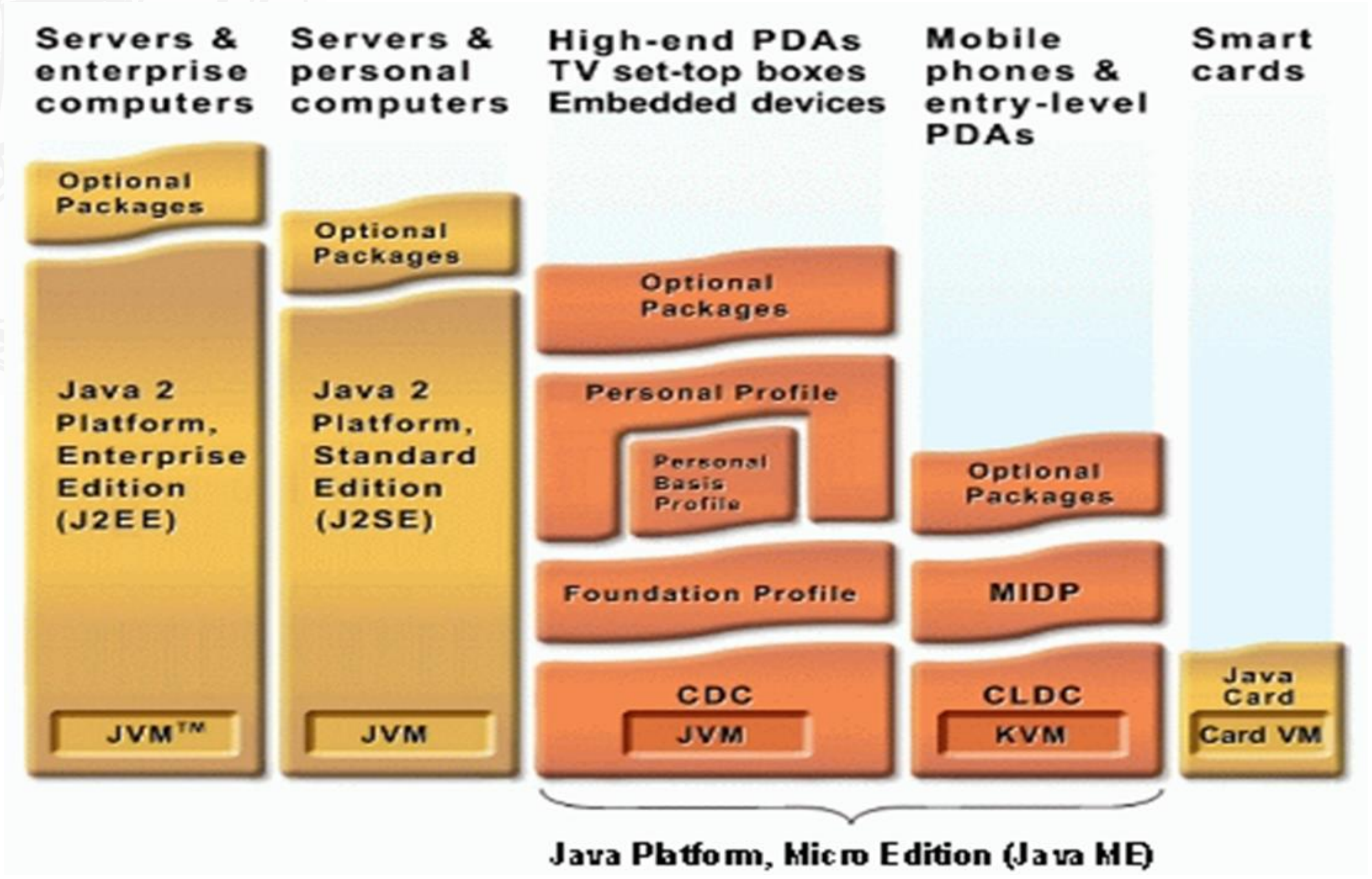
Three Elements of Java ME

- A ***configuration*** provides the most basic set of libraries and virtual machine capabilities for a broad rang of devices

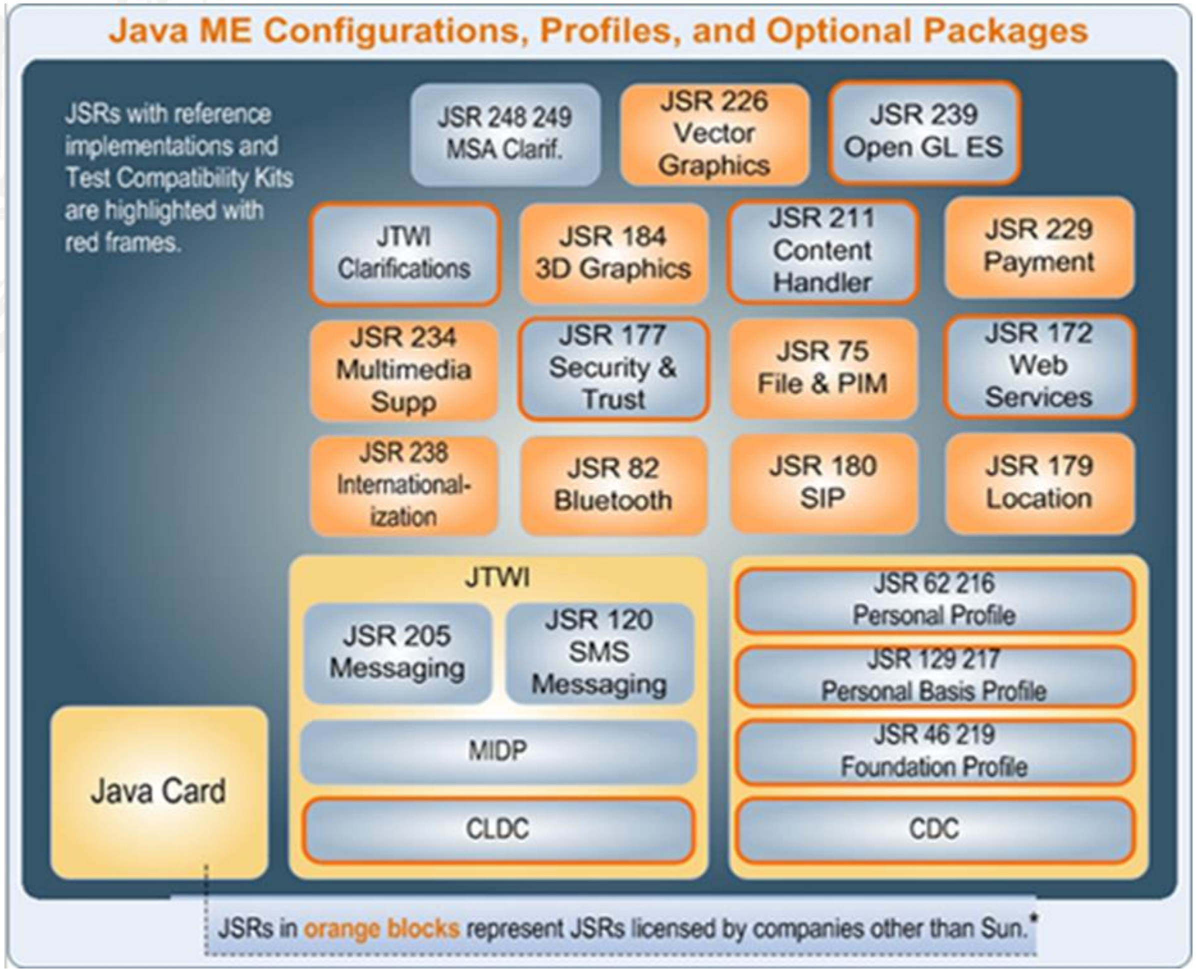
- A ***profile*** is a set of APIs that support a narrower range of devices

- An ***optional package*** is a set of technology-specific APIs

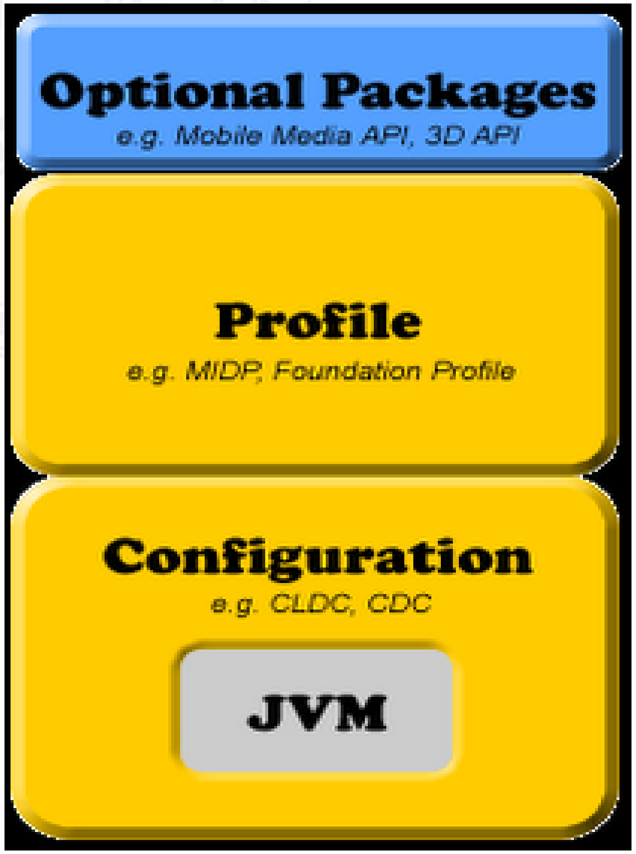
Java Technology

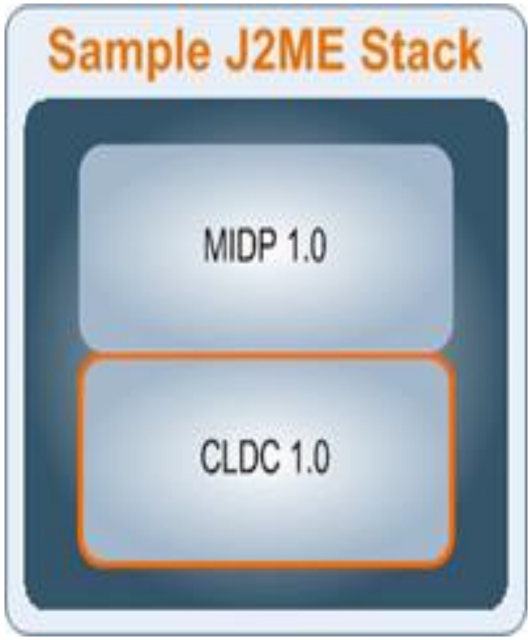

Servers & enterprise computers
Servers & personal computers
High-end PDAs TV set-top boxes Embedded devices
Mobile phones & entry-level PDAs
Smart cards
Optional Packages
Java 2 Platform, Enterprise Edition (J2EE)
JVM™
Optional Packages
Java 2 Platform, Standard Edition (J2SE)
JVM
Optional Packages
Personal Profile
Personal Basis Profile
Foundation Profile
CDC
JVM
Optional Packages
MIDP
CLDC
KVM
Java Card
Card VM
Java Platform, Micro Edition (Java ME)

Java ME Overview


Java ME Configurations, Profiles, and Optional Packages
JSRs with reference implementations and Test Compatibility Kits are highlighted with red frames.
JSR 248 249 MSA Clarif.
JSR 226 Vector Graphics
JSR 239 Open GL ES
JTWI Clarifications
JSR 184 3D Graphics
JSR 211 Content Handler
JSR 229 Payment
JSR 234 Multimedia Supp
JSR 177 Security & Trust
JSR 75 File & PIM
JSR 172 Web Services
JSR 238 International-ization
JSR 82 Bluetooth
JSR 180 SIP
JSR 179 Location
JTWI
JSR 205 Messaging
JSR 120 SMS Messaging
MIDP
CLDC
JSR 62 216 Personal Profile
JSR 129 217 Personal Basis Profile
JSR 46 219 Foundation Profile
CDC
Java Card
JSRs in orange blocks represent JSRs licensed by companies other than Sun.*

Java ME Stack

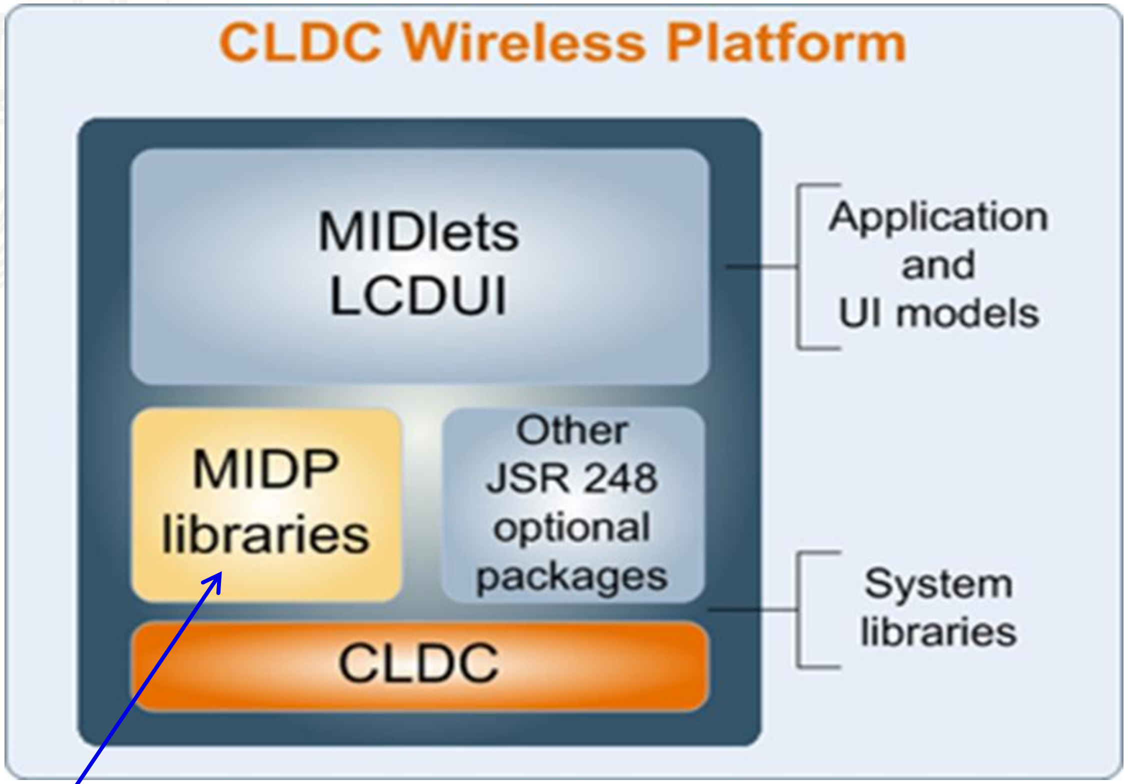




Base Configurations

- Connected Limited Device Configuration (CLDC)
- Connected Device Configuration (CDC)
- JSR – Java Specification Request
  - CLDC 1.1 – JSR 139
  - CDC 1.1.2 – JSR 218

CLDC for Small Devices

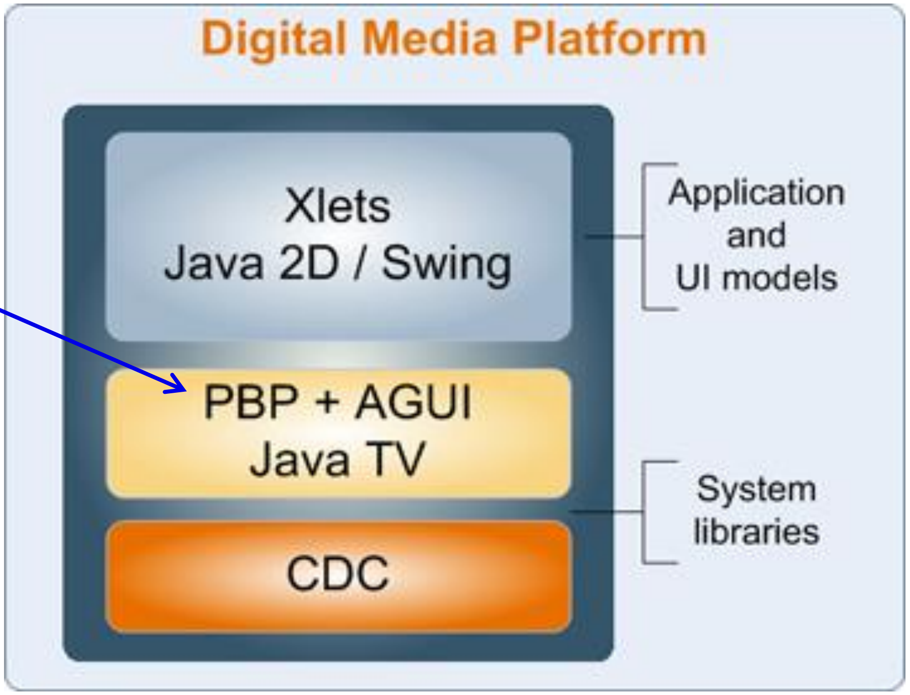


Mobile Information Device Profile

CDC for Smart Phones

CDC Profiles:

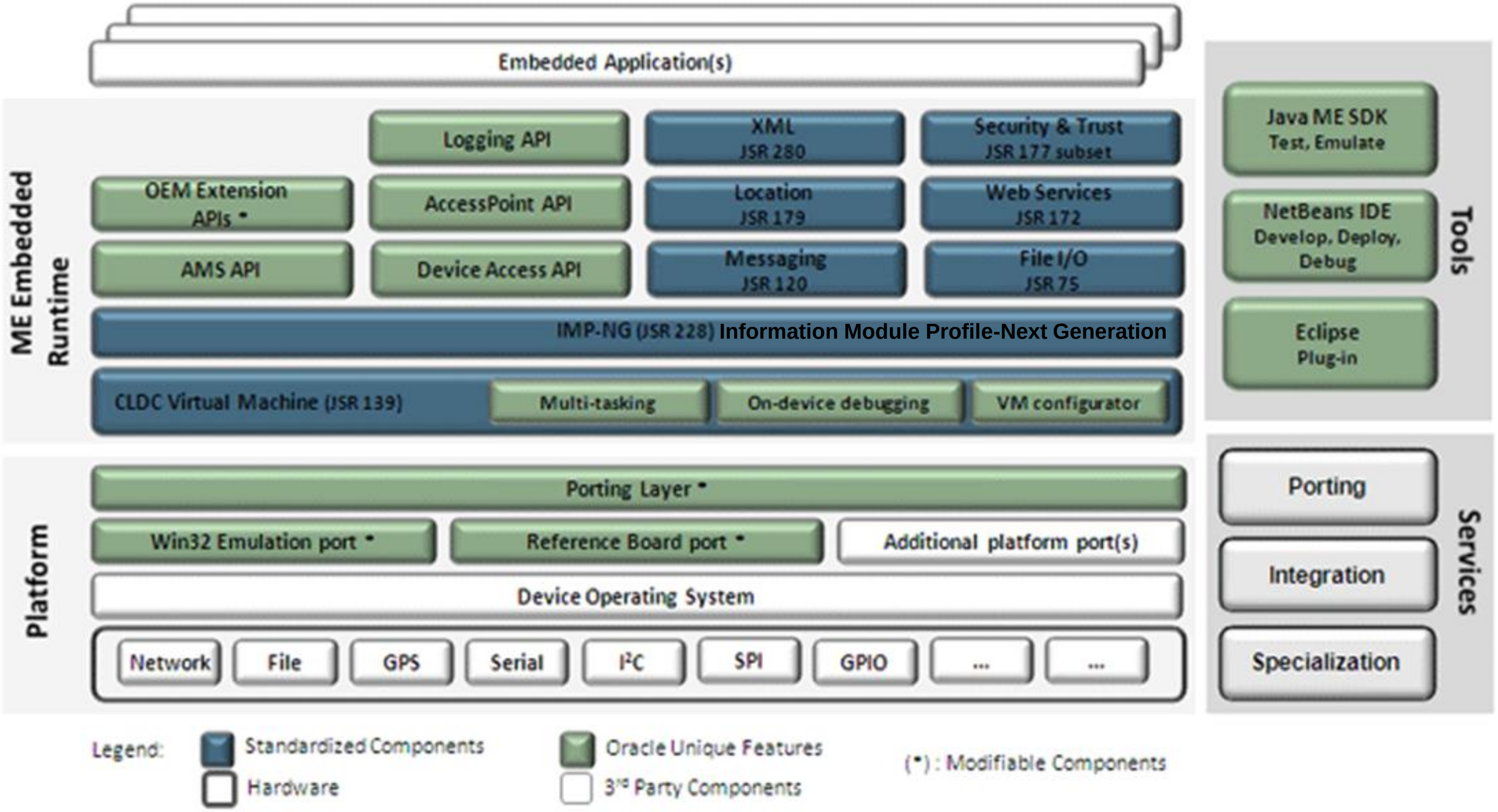
- Personal Basis Profile (PBP)
- Foundation Profile
- Personal Profile



Java ME Embedded

- Based on Java ME CLDC with dedicated embedded functionality
  - Wireless modules
  - Smart meters/smart sensors
  - Industrial controllers
  - Telehealth devices
  - Environmental remote monitors
  - Tracking systems
  - Home automation devices
  - Connected vending machines
  - And more, including the general M2M (machine-to-machine) space

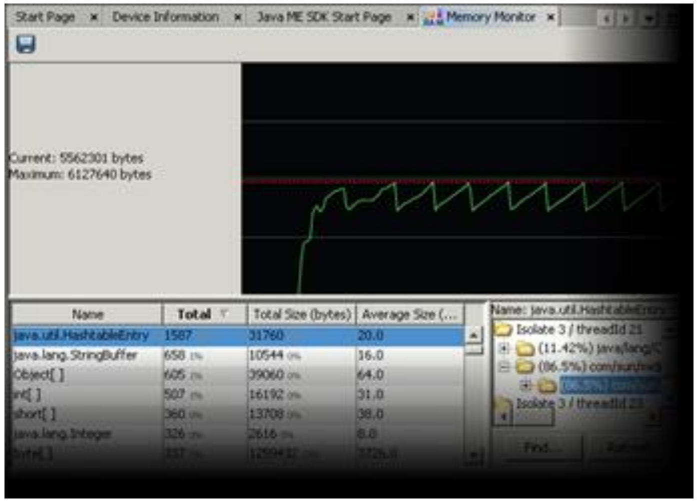
Java ME Embedded Stack



Information Module Profile-Next Generation

Java ME Development Kit

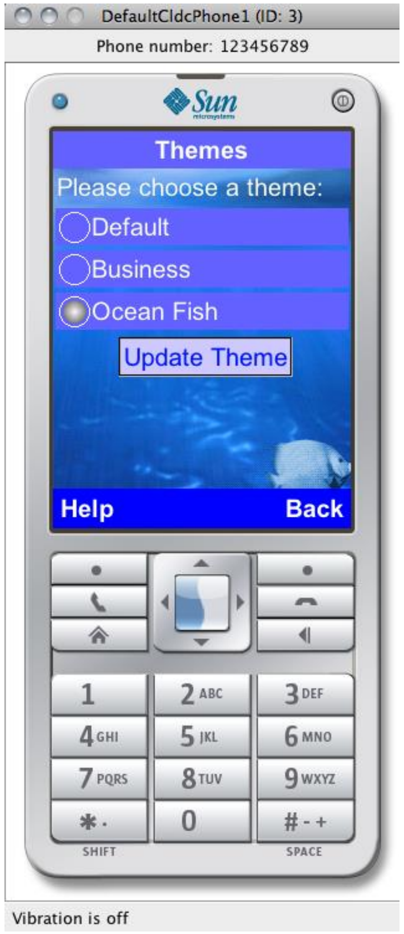
- Java ME SDK 3.2
  - Integrated CLDC, CDC and Blu-ray Disc Java (BD-J)
  - Embedded Platform support, Eclipse/NetBeans support, Memory Monitor



Current: 5562301 bytes
Maximum: 6127640 bytes

Name	Total	Total Size (bytes)	Average Size (...
java.util.HashtableEntry	1587	31760	20.0
java.lang.StringBuffer	658	10544	16.0
Object[ ]	605	39060	64.0
int[ ]	507	16192	31.0
short[ ]	360	13708	38.0
java.lang.Integer	326	2616	8.0
byte[ ]	[illegible]	[illegible]	[illegible]

Name: java.util.HashtableEntry
Isolate 3 / threadId 21
(11.42%) java/lang/
(86.5%) com/sun/
Isolate 3 / threadId 23



UNIVERSITY OF WOLLONGONG

Process of MIDlet Creation

1. Design
   - Different from other Java application, running in a very different environment
2. Code
   - Each code must extend the abstract MIDlet class in javax.microedition.midlet
3. Compile
   - Change boot CLASSPATH:

```
javac -bootclasspath %CLDC_PATH%\common\api\classes yourcode.java
```

4. Preverify
   - To ensure the class file is structurally and conceptually correct as per the JVM specification

```
preverify -classpath %CLDC_PATH%\common\api\classes;tmpclasses
          -d your.full.class.name
```

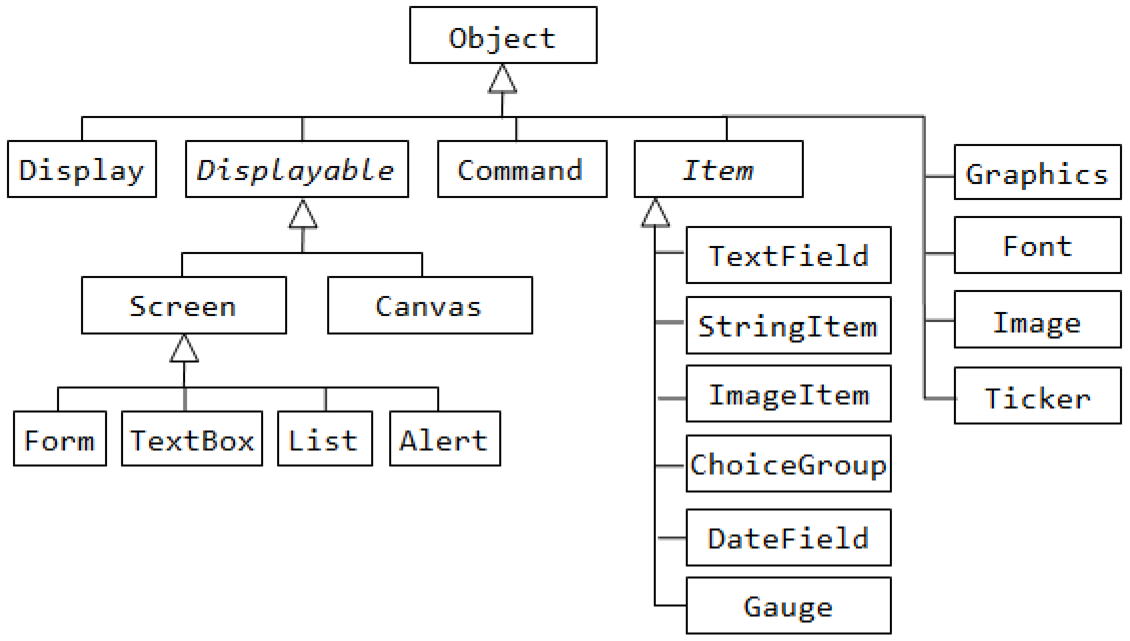
5. Package
   - Create a Manifest file: `Manifest.mf`
   - Create JAR file: `jar cvfm yourJarfile.jar Manifest.mf`
   - Create Java Application Descriptor (JAD) file: `yourJadFile.jad`
6. Test

```
Emulator -Xdescriptor yourJadFile.jad
```

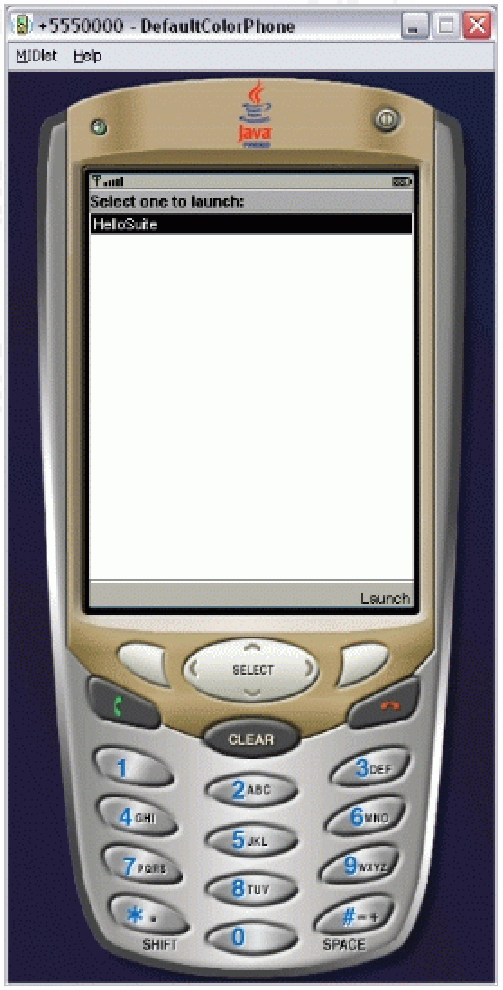
7. Deploy
   - USB or a Bluetooth
   - Internet

```
<HTML>
   Click <a href="yourJadFile.jad" here</a> to download the MIDlet
</HTML>
```

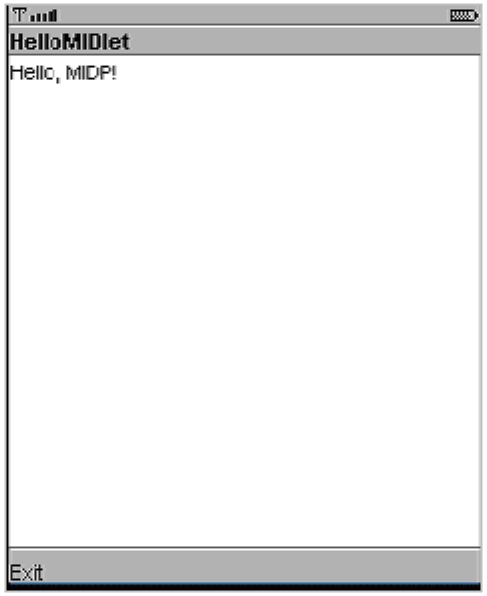
MIDP UI Classes



Example: HelloMIDlet



Select one to launch:
HelloSuite
Launch



HelloMIDlet
Hello, MIDP!
Exit

Running on the emulator

Running on ColorPhone emulator

UNIVERSITY OF WOLLONGONG

Example: HelloMIDlet

```
// Uses Java ME APIs (instead of Java SE)
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class HelloMIDlet extends MIDlet implements CommandListener {
  private Form mMainForm;
  public HelloMIDlet() {
    // allocate a Form to hold the UI components
    mMainForm = new Form("HelloMIDlet");
    // add a String component
    mMainForm.append(new StringItem(null, "Hello, MIDP!"));
    // add the command
    mMainForm.addCommand(new Command("Exit", Command.EXIT, 0));
    // register "this" to handle command
    mMainForm.setCommandListener(this);
  }
```

Continuing

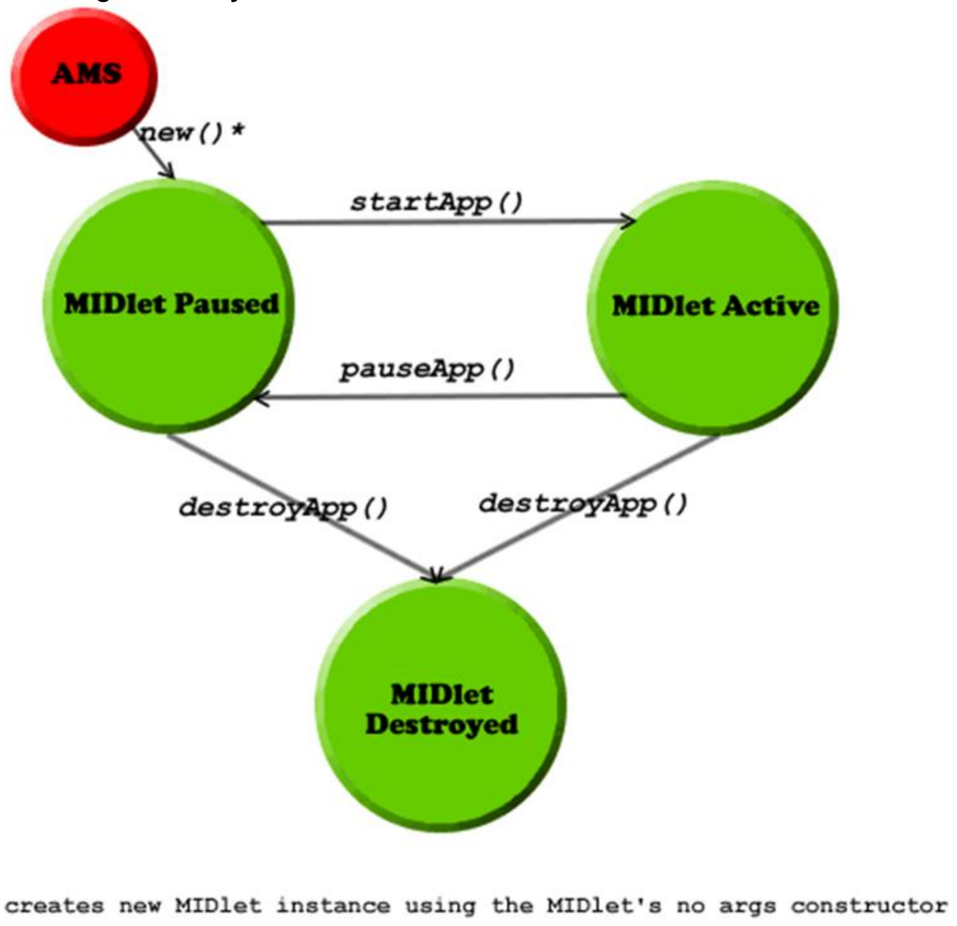
Example: HelloMIDlet

Continued

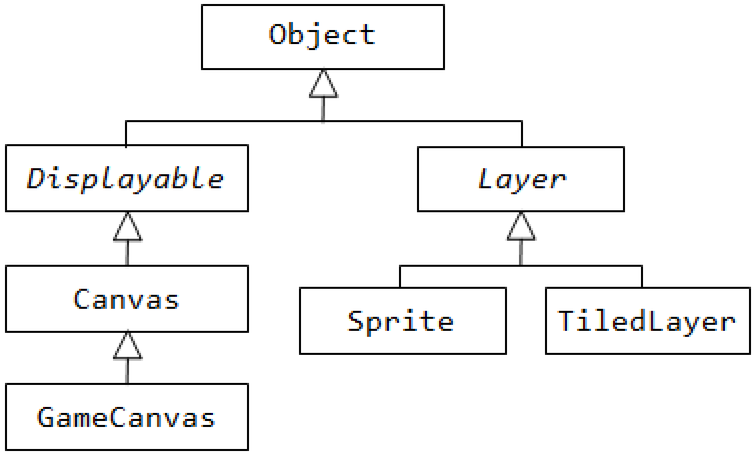
```
  // Called back by the Runtime to start or resume the MIDlet
  public void startApp() {
    Display.getDisplay(this).setCurrent(mMainForm);
  }
  // Called back by the Runtime to pause the MIDlet
  public void pauseApp() {}
  // Called back by the Runtime before the MIDlet is destroyed
  public void destroyApp(boolean unconditional) {}
  // Handler for the Exit command
  public void commandAction(Command c, Displayable s) {
    // put the midlet into destroy state
    notifyDestroyed();
  }
}
```

MIDLet Life Cycle

Application Management System

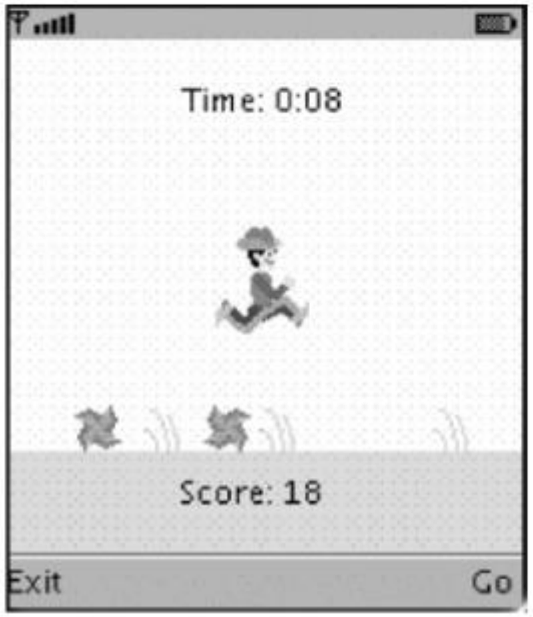


MIDP Game API



Example: MIDP Game

- Game API in Mobile Internet Device Profile (MIDP) 2.0
  - javax.microedition.lcdui.game.*



Tumbleweed

A cowboy walking through a prairie jumping over tumbleweeds

Example: MIDP Game

- MIDlet Class
  `Jump.java`
- Thread Class – animation loop
  `GameThread.java`
- GameCanas Class – area of the screen
  `JumpCanvas.java`
- LayerManager Class – organizing layers that represent graphical objects (background and sprite)
  `JumpManager.java`
- Sprite Class – graphical objects
  `Cowboy.java; Tumbleweed.java`
- TileLayer Class – background objects
  `Grass.java`

Example: MIDP Game

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
Import javax.microedition.lcdui.game.*;

public class Jump extends MIDlet implements CommandListener {
  private Command myExitCommand = new Command("Exit", Command.EXIT, 99);
  private Command myGoCommand = new Command("Go", Command.SCREEN, 1);
  private Command myPauseCommand = new Command("Pause", Command.SCREEN, 1);
  private Command myNewCommand = new Command("Play Again", Command.SCREEN, 1);
  private JumpCanvas myCanvas;
  private GameThread myGameThread;
  ...
  public void startApp() throws MIDletStateChangeException {
    try {
      if(myCanvas == null) {
        myCanvas = new JumpCanvas(this);
        myCanvas.addCommand(myExitCommand); ...
      }
    } catch(Exception e) {
      errorMsg(e);
    }
  }
```

Example: MIDP Game

```
public class JumpCanvas extends GameCanvas {
  ...
  public JumpCanvas(Jump midlet) throws Exception {
    super(false);
    myDisplay = Display.getDisplay(midlet);
    myJump = midlet;
    // calculate the dimensions
    DISP_WIDTH = getWidth();
    DISP_HEIGHT = getHeight();
    Display disp = Display.getDisplay(myJump);
    ...
    }
  }
  void start() {
    myGameOver = false;
    myDisplay.setCurrent(this);
    repaint();
  }
```

Example: MIDP 2 Game

```
 public void paint(Graphics g) {
    // clear the screen:
    g.setColor(WHITE);
    g.fillRect(CORNER_X, CORNER_Y, DISP_WIDTH, DISP_HEIGHT);
    // color the grass green
    g.setColor(0, 255, 0);
    g.fillRect(CORNER_X, CORNER_Y + DISP_HEIGHT - GROUND_HEIGHT, ...);
    try {
      myManager.paint(g);
    } catch(Exception e) {
      myJump.errorMsg(e);
    }
    ...
    if(myGameOver) {
      myJump.setNewCommand();
      // clear the top region:
      ...
    }
  }
  ...
}
```

Example: MIDP 2 Game

```
public class GameThread extends Thread {
  GameThread(JumpCanvas canvas) {
    myJumpCanvas = canvas;
  }
  private long getWaitTime() { ... }
  void pauseGame() { ...  }
  void resumeGame() { ... }
  void requestStop() { ... }
  public void run() {
    ...
    while(true) {
      myLastRefreshTime = System.currentTimeMillis();
      if(myShouldStop) {
          break;
      }
      ...
    }
  }
}
```

Example: MIDP 2 Game

- Download the complete code

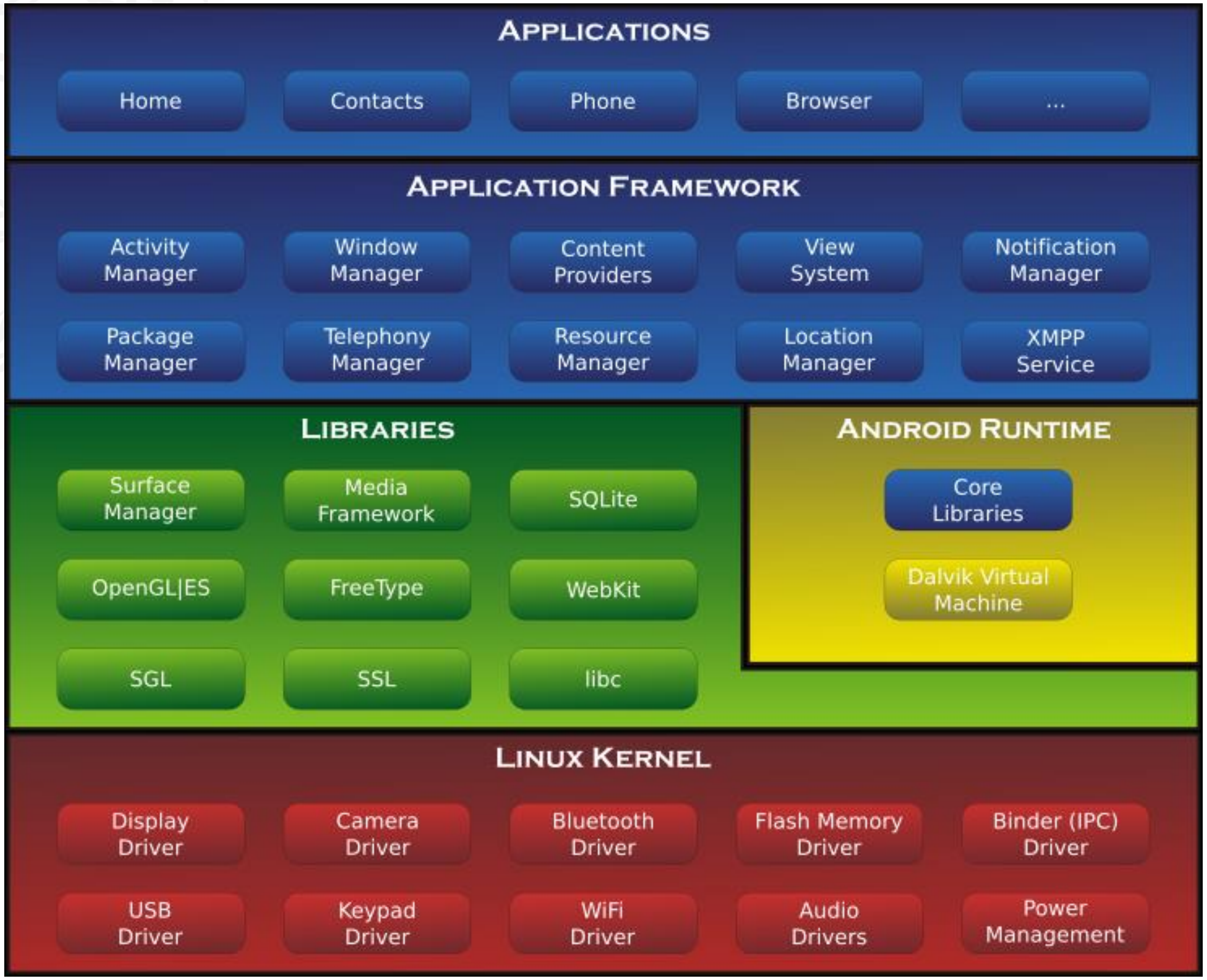
`http://www.apress.com/book/downloadfile/3644`

Android

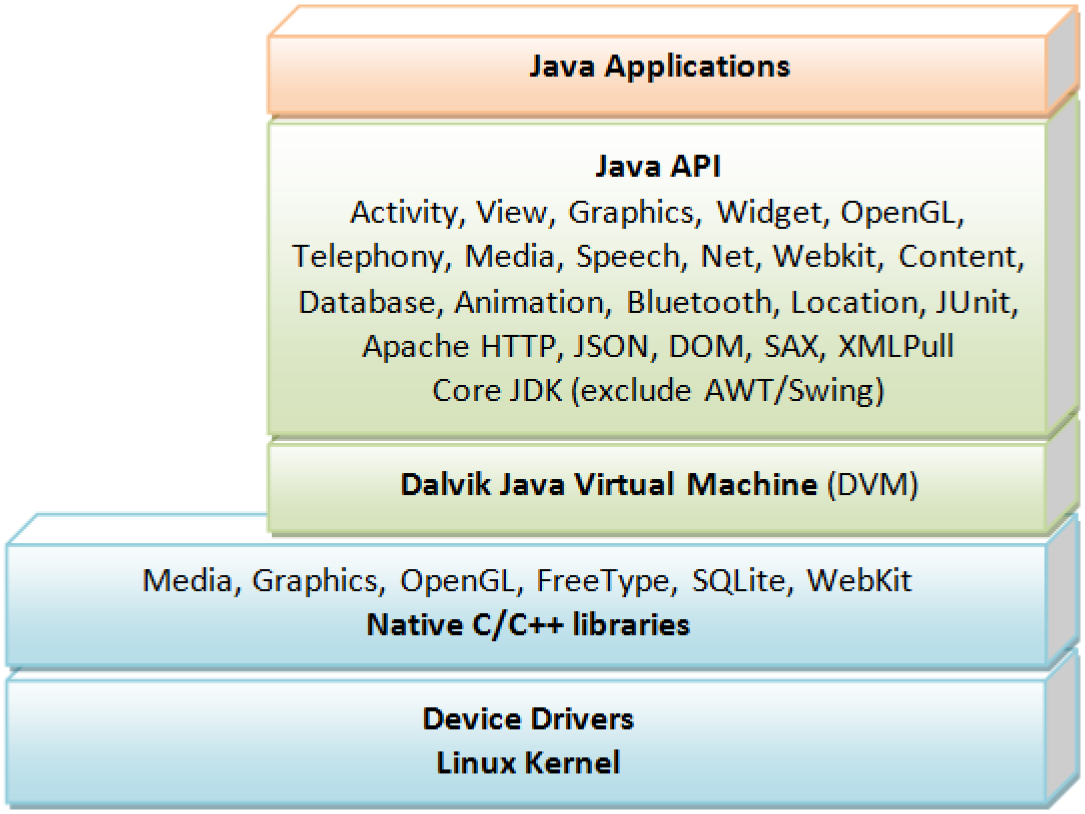
- Software Stack
  - Modified version of Linux kernel
- Middleware
- Apps
  - In Java
    - Not AWT or Swing
    - A small set of UI APIs

UNIVERSITY OF WOLLONGONG

Android System Architecture



Android Java API


Java Applications
Java API
Activity, View, Graphics, Widget, OpenGL,
Telephony, Media, Speech, Net, Webkit, Content,
Database, Animation, Bluetooth, Location, JUnit,
Apache HTTP, JSON, DOM, SAX, XMLPull
Core JDK (exclude AWT/Swing)
Dalvik Java Virtual Machine (DVM)
Media, Graphics, OpenGL, FreeType, SQLite, WebKit
Native C/C++ libraries
Device Drivers
Linux Kernel

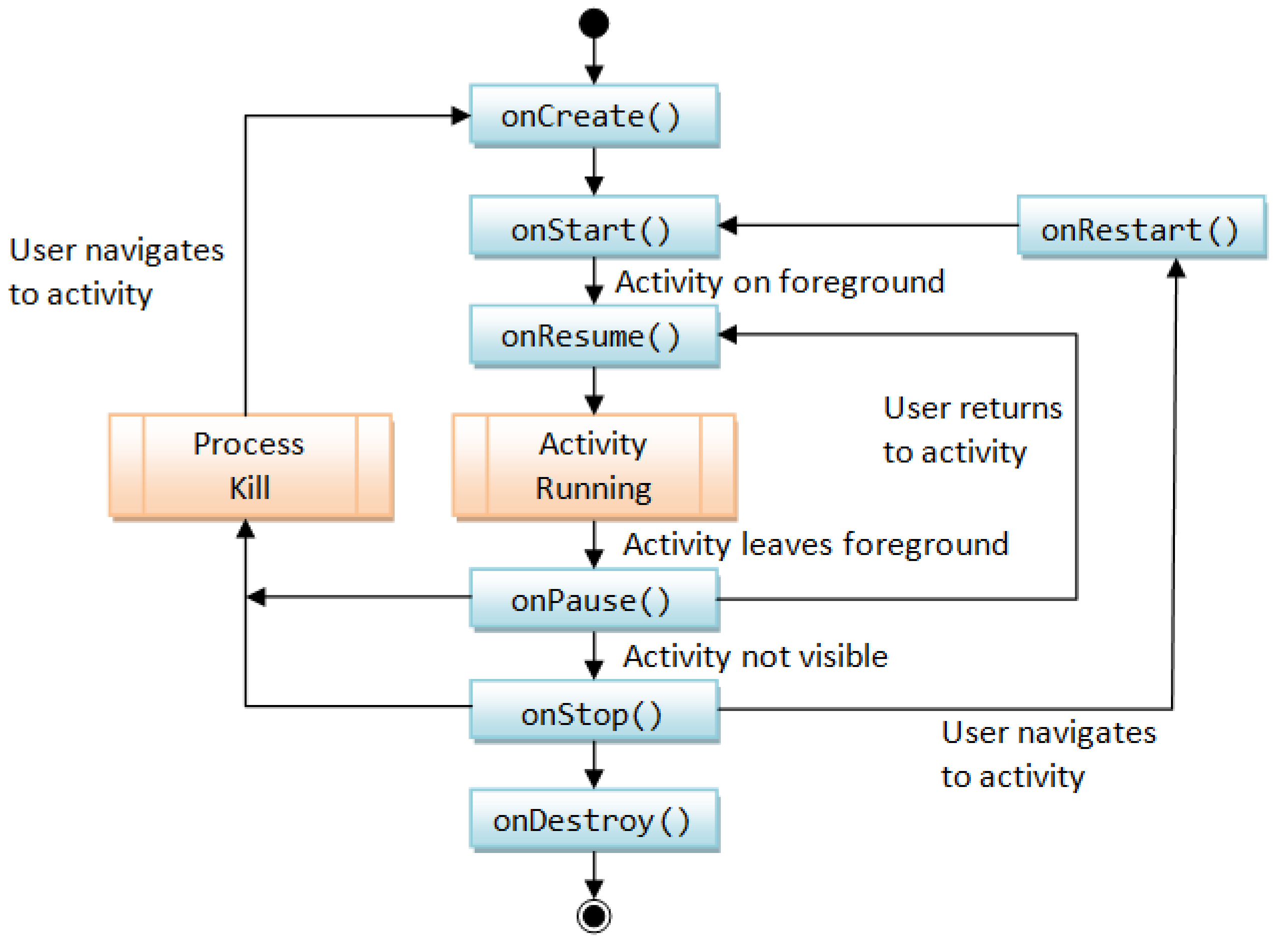
Example: Hello Android

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloActivity extends Activity {

    // Called when the activity is first created.
    public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          // Construct a TextView UI component
          TextView textView = new TextView(this);
          // Set the text message for TextView
          textView.setText("Hello Android");
          // this Activity sets its content to the TextView
          setContentView(textView);
    }
}
```

Android Basics

- Activity
  - An activity has a single screen, which usually composes of one of more views
  - An activity interacts with the user to do one and only one thing, such as viewing data, creating data, or editing data
- View
  - Views are UI components (or widget, or control) (such as button, label, text field) as well as containers of components
  - A framework similar to Swing based around `Views` rather than `Jcomponent`s
    - Android supports many core JDK packages, except graphics packages AWT and Swing.
    - It provides its own 2D graphics supports, via views and widgets. It supports 3D graphics via OpenGL ES
- Fragment
  - An activity can display one or more fragments on the screen at the same time
    - For a smaller screen, an activity is more likely to contain just one fragment
- Intent
  - An intent declares an intention to do something
  - Intents are like "glue" that enable different activities from different applications to work together

Activity Life Cycle



Java Programs

- Java SE
  - Console:

    ```
    public class MyApp
    ```

  - Window:

    ```
    public class MyFrame extends JFrame
    ```

  - Applet:

    ```
    public class MyApplet extends JApplet
    ```

- Java ME
  - MIDlet:

    ```
    public class MyMIDlet extends MIDlet
    ```

- Android Java
  - App:

    ```
    public class MyActivity extends Activity
    ```

References

- Java ME Technology

  `http://www.oracle.com/technetwork/java/javame/java-me-overview-402920.html`

- Wireless Development Tutorials

  `http://www.oracle.com/technetwork/systems/wtoolkit-155632.html`

- Carol Hamer, Creating Mobile Games, Home, Apress, 2007

- Android Developers

  `http://developer.android.com`