Mobile Phone Programming

Introduction

Dr. Christelle Scharff
Pace University, USA

http://atlantis.seidenberg.pace.edu/wiki/mobile2008

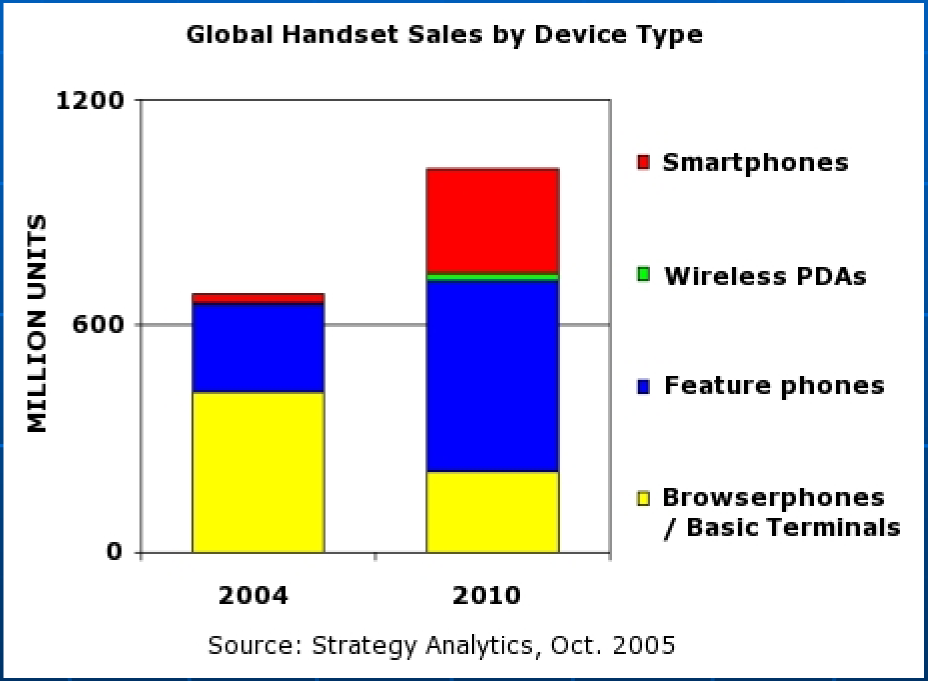
Objectives

- Getting an overall view of the mobile phone market, its possibilities and weaknesses
- Providing an overview of the J2ME architecture and define the buzzwords that accompanies it

Why mobile phones?

- Nowadays mobile phones outnumber desktop computers for Internet connections in the developer world
- A convenient and simpler alternative to the desktop/laptop for all (developed and developing countries)
- Mobile phones are computers!
- Some numbers and important facts:
  - Target of 10 million iphones sales by the end of 2008 (just one year after being launched)
  - Google phone to be launched in 2008
  - 70% of the world’s mobile subscriptions are in developing countries, NY Times April 13, 2008

Global Handset Sales by Device Type

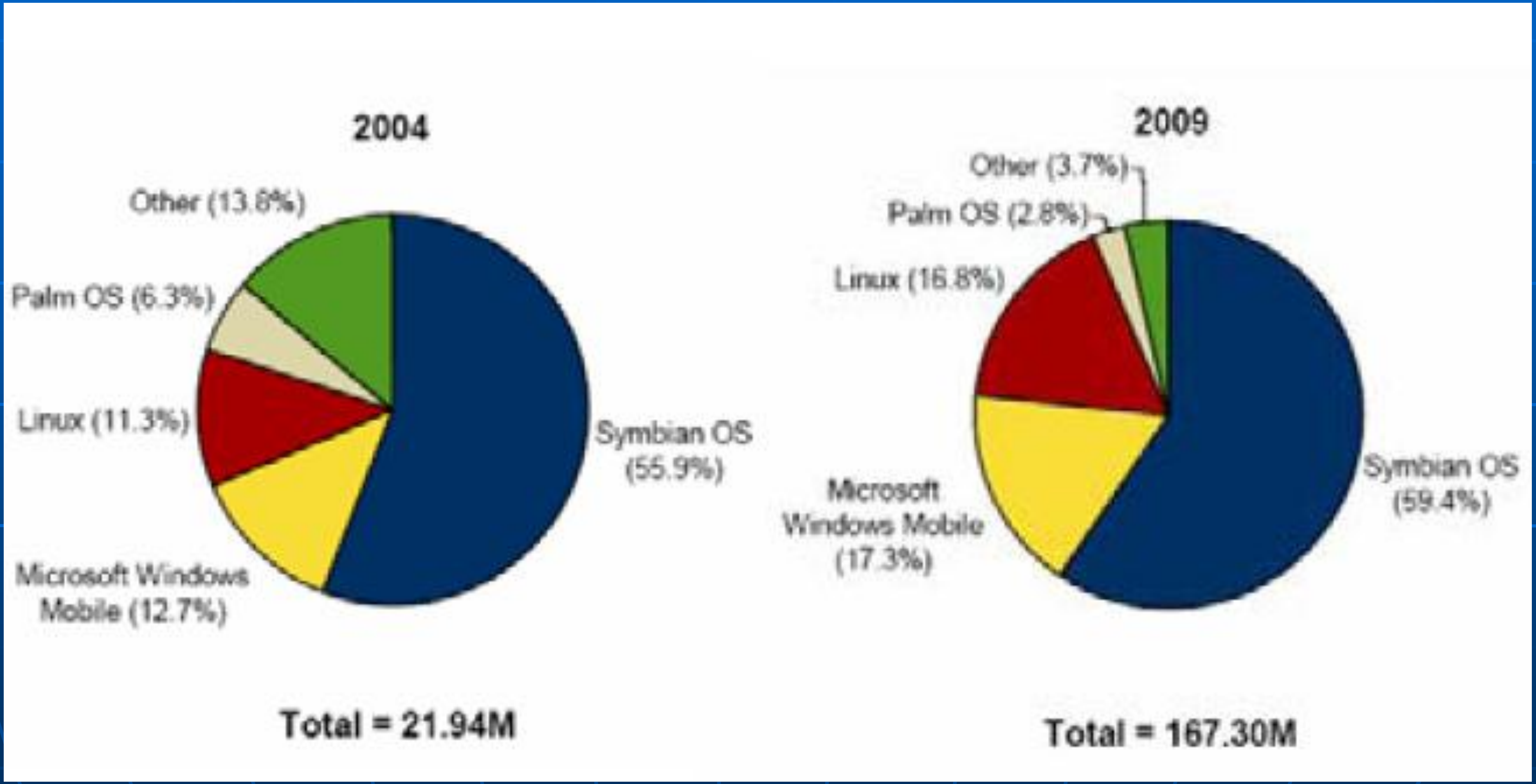


http://linuxdevices.com/files/misc/StrategyAnalytics-mobilephone-segments.jpg

Devices

- A wide variety of devices by the main vendors:
  - E.g, Nokia, Motoral, Sony Ericson
- A wide variety of operating systems
  - E.g., Blackberry, Palm OS, Windows CE/Mobile, Symbian, motomagx, linux
- A wide variety of development environments
  - E.g., Java ME, Qualcomm's BREW, Google' Android, Google App Engine (GAE) for mobile web applications, JavaFX
- Programming languages:
  - Java, Python, Flast-lith, Objective C

Operating Systems



http://mobiledevices.kom.aau.dk

Mobile Web

- Access to wireless data services using a mobile device
- cHTML (Compact HTML) is a subset of HTML that excludes JPEG images, tables, image maps, multiple character fonts and styles, background color and image, frames and style sheets
  - http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/
- WML (Wireless Markup Language) is a standard for content delivered to mobile devices
  - http://openmobilealliance.org
- dotMobi is a top-level domain dedicated to delivering the Internet to mobile devices
  - http://mtld.mobi/

Why Java?

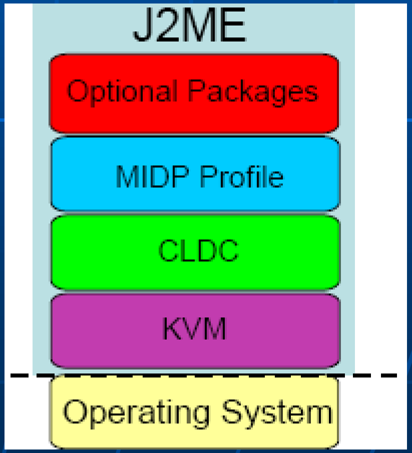
- The Java platform is
  - Safe – the code executes within the JVM
  - Robustness – automated garbage collection prevents memory management
  - Portability – a single executable can run on several devices
  - Rich set of APIs
- Market trends
  - 80% of the mobile devices are Java compliant
  - Lots of Java applications on the market
  - Operators are developing Java services

Java 2 Platform

- Composed of 3 elements:
  - Java programming language specification
  - Virtual machine
  - APIs (Application Programming Interfaces)
- Supports a wide range of hardware:
  - J2SE (Java Standard Edition)
  - J2EE (Java Enterprise Edition)
  - J2ME (Java Micro Edition)
  - Java Card

J2ME

- J2ME is not a piece of software like J2SE
- J2ME is a platform, a collection of technologies and specifications for small devices
- J2ME is divided into 3 components:
  - Configurations
  - Profiles
  - Optional packages



JCP

- Java Community Process
- http://jcp.org/
- JCP is a consortium of experts (companies and individuals) who develop and evolve Java technology specifications
- A *specification* is based on the description of the language, virtual machine, platform editions, profiles, and application programming interfaces
- JCP stages: New Java Specification Request (JSR) review, Early draft review, Public review, Proposed final draft, Maintenance review, Rejected, Removed

JSR

- Java Specification Request
- List of all the JSR: http://jcp.org/en/jsr/all
- JSRs are descriptions of proposed and final specifications for the Java technology
- Examples:
  - JSR 82 – Bluetooth
  - JSR 120 – SMS Messaging
  - JSR 184 – 3D Graphics

Configuration

- A *configuration* is a specification that defines the minimum virtual machine and base set of APIs to develop applications for a family of devices
  - The target may be devices with intermittent access to the Internet, small memory size and processing capabilities
- Examples:
  - CLDC 1.0 / 1.1 – Connected Limited Device Configuration – JSR 30 / 139 – KVM – small memory and intermittent access to Internet
    - CLDC 1.1 supports floating-point math capabilities
  - CDC / CDC 1.1 – Connected Device Configuration – JSR 36 / 218 – CVM – larger memory and always on network connection

CLDC 1.0 APIs

- List of packages:
  - java.lang – data types, basic system and threads (Boolean, Byte, Character, Integer, Long, Short, String, StringBuffer, Math, Object, Runtime, System, Thread, Throwable)
  - java.io – to manage I/O data streams
  - java.util – utility classes (Calendar, Date, Hastable, Random, Stack, Timer, TimerTask, Vector)
  - javax.microedition.io – for generic connections
- Library specification library
  - http://java.sun.com/javame/reference/apis.jsp

Profile

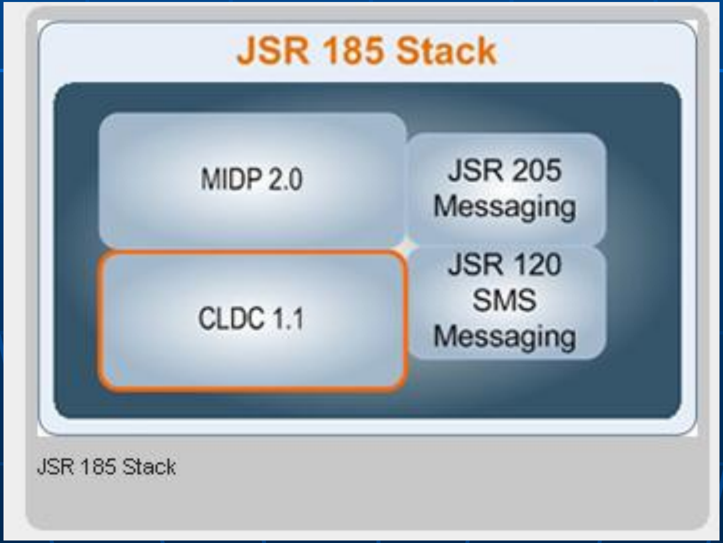
- A *profile* extends a specification and add more specific APIs to provide a more complete environment to develop applications
- Profiles can include APIs for user interface and persistence storage
- Examples:
  - MIDP 1.0 / 2.0 – Mobile Information Device Profile – JSR 37 / 138
    - MIDP 2.0 offers advanced networking, security, gaming, and media features
  - Foundation Profile – JSR 46

Optional Packages

- An *optional package* provides functionalities that may not be associated with a particular configuration and profile
- Examples:
  - JSR 82 - Bluetooth API
  - JSR 120 - Wireless messaging API WMA
  - JSR 172 - J2ME web services

Stack

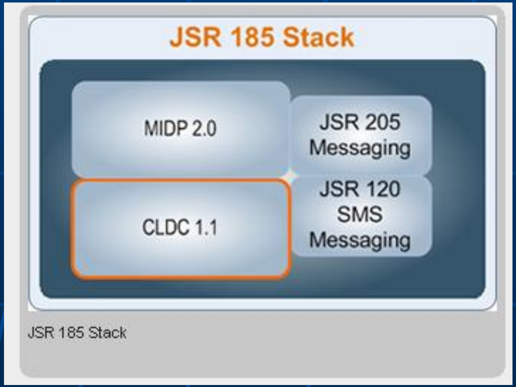
- A device implements a complete software stack that consists of a configuration, a profile and optional packages to make it clear to the developer on what to expect from the device
- Example: JSR 185 Stack - JTWI (Java Technology for the Wireless Industry)



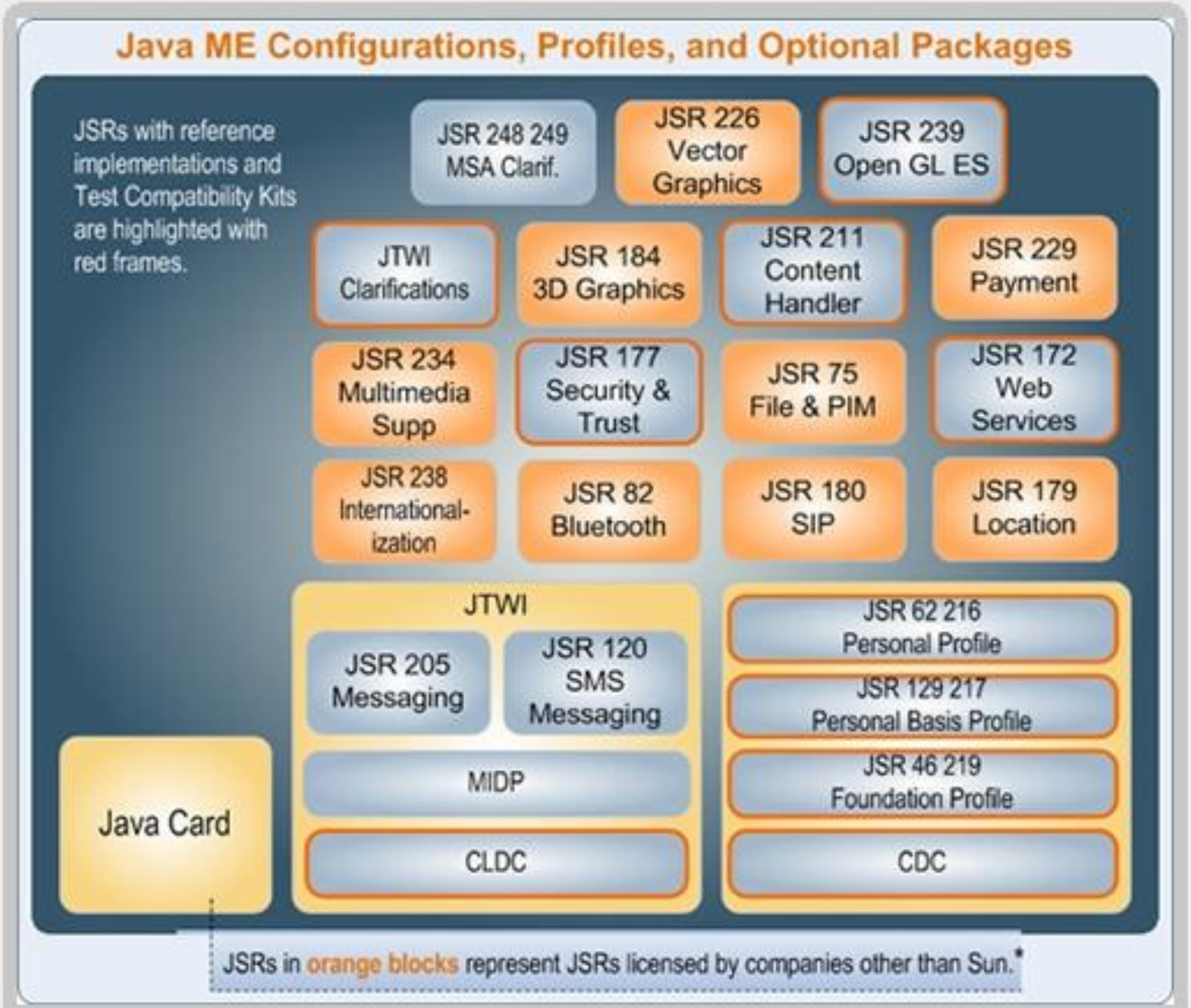
JSR 185 Stack

Fragmentation

- *Fragmentation* is the inability to "write once and run anywhere" due to the multitude of vendor-specific and optional APIs
- Developing an application targeting n different devices required it to be tested on the n devices
- JTWI JSR 185 is one step to provide a comprehensive set of functionalities in a standard application development by clarifying and combining vendor-specific and optional APIs



JSR 185 Stack

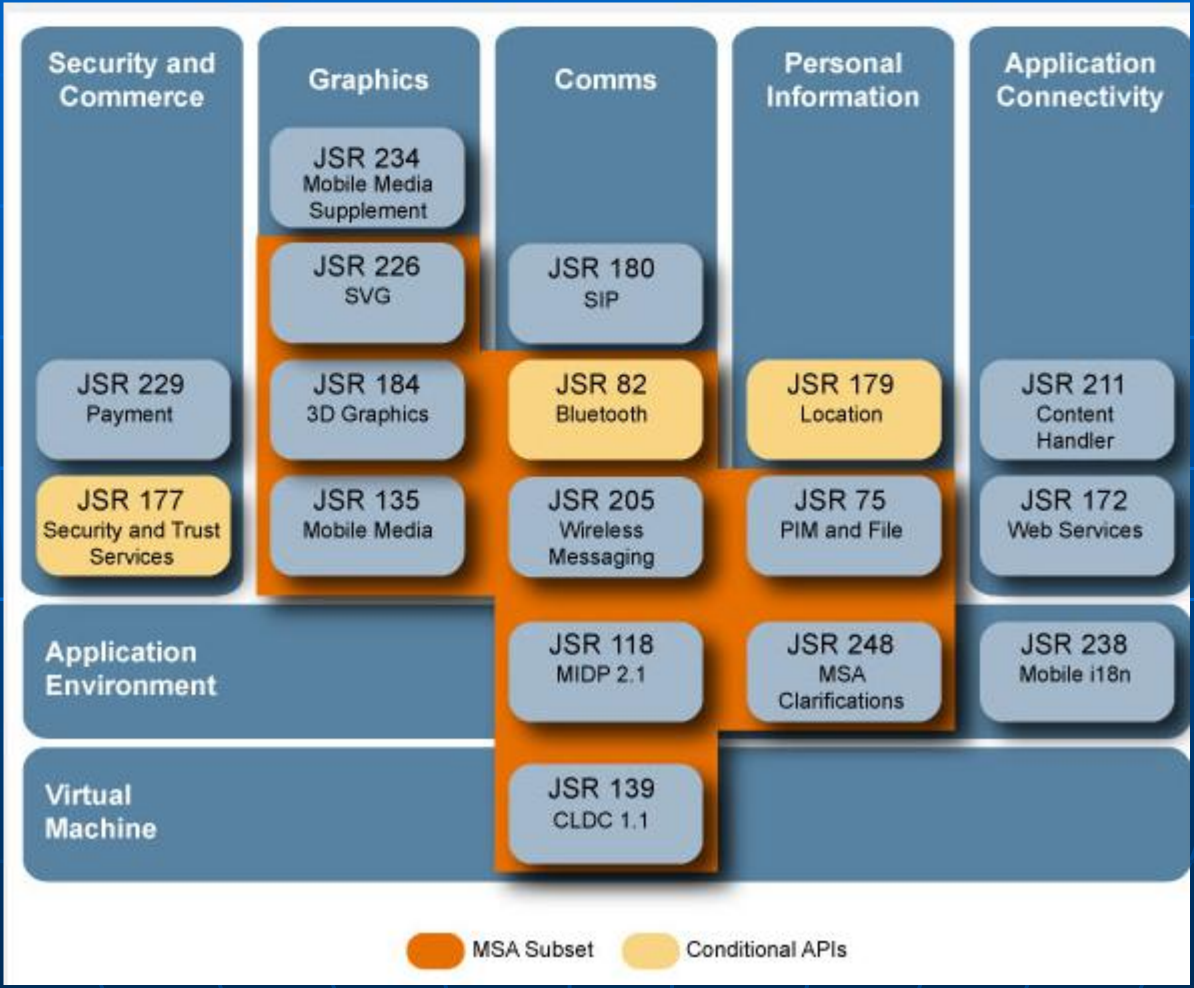

Java ME Configurations, Profiles, and Optional Packages
JSRs with reference implementations and Test Compatibility Kits are highlighted with red frames.
JSR 248 249 MSA Clarif.
JSR 226 Vector Graphics
JSR 239 Open GL ES
JTWI Clarifications
JSR 184 3D Graphics
JSR 211 Content Handler
JSR 229 Payment
JSR 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.*

J2ME Overview

MSA

- Mobile Service Architecture JSR 248
- MSA is a specification built on CLDC 1.1, MIDP 2.0 and JTWI to incorporate new technology and services
- MSA is the new wireless industry-defined standard
- MSA is divided in 2 branches: MSA and MSA subset
- MSA contains a set of mandatory and conditionally mandatory APIs
  - A conditionally mandatory API is an API that is not present on all devices (e.g., JSR 179 Location API)
- Advanced Mobile Service Architecture JSR 249 is next!

MSA


Security and Commerce
Graphics
Comms
Personal Information
Application Connectivity
JSR 234 Mobile Media Supplement
JSR 226 SVG
JSR 180 SIP
JSR 229 Payment
JSR 184 3D Graphics
JSR 82 Bluetooth
JSR 179 Location
JSR 211 Content Handler
JSR 177 Security and Trust Services
JSR 135 Mobile Media
JSR 205 Wireless Messaging
JSR 75 PIM and File
JSR 172 Web Services
Application Environment
JSR 118 MIDP 2.1
JSR 248 MSA Clarifications
JSR 238 Mobile i18n
Virtual Machine
JSR 139 CLDC 1.1
MSA Subset
Conditional APIs

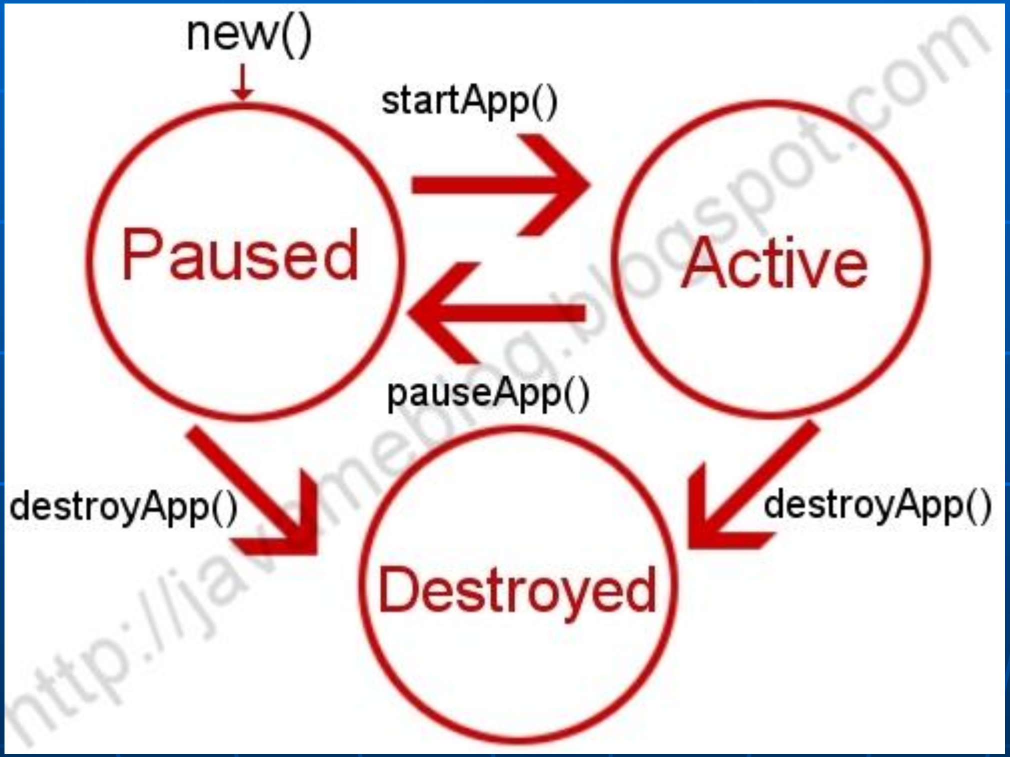
Development Environments

- Sun Java Wireless Toolkit for CLDC
  - http://java.sun.com/products/sjwtoolkit/
- EclipseME plugin
  - http://wlcipseme.org
- NetBeans Mobility Pack
  - http://www.netbeans.org/products/mobility
- Vendor-specific development environments of:
  - Motorola http://developer.motorola.com/
  - Nokia http://forum.nokia.com
  - Sony Ericson http://developer.sonyericsson.com
  - Sprint http://developer.spring.com

MIDlet

- A MIDlet is an application that can run on MIDP devices
- A MIDlet is a class that inherits from javax.microedition.midlet.MIDlet
- A MIDlet has three methods:
  - startApp() – to initialize the MIDlet or resume a paused MIDlet
  - pauseApp() – to pause the application
  - destroyApp() – to clean up the application and release all resources
- These methods are *callback* – the Application Management Software (AMS) calls them whenever necessary
- These methods can also be called in the MIDlet code

MIDlet Application Lifecycle



MIDlet Suites

- One or more MIDlets are packaged together into a MIDlet suite composed of:
  - A Java Archive (JAR) file – containing the user-defined classes, images and sounds that make up the application and the JAR file manifest that describes the attributes of the MIDlet
  - A Java Descriptor (JAD) file – containing the description of the MIDlet suite
    - It permits a device to examine the descriptor before downloading the whole MIDlet suite

Skeleton of a MIDlet

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

public class SkeletonMIDlet extends MIDlet implements CommandListener {

    public SkeletonMIDlet() {}

    protected void destroyApp(boolean arg0) throws MIDletStateChangeException {}

    protected void pauseApp() {}

    protected void startApp() throws MIDletStateChangeException {}

    public void commandAction(Command arg0, Displayable arg1) {}

}
```

References

- Introduction to Java Mobility Technology
  - http://developers.sun.com/mobility/getstart/
- Java Community Process
  - http://jcp.org
- Glossary
  - http://developers.sun.com/mobility/glossary/