Content

- Mobile Ecosystem
- Mobile application frameworks
- Java 2 Micro Edition
- Configurations and profiles
- Optional packages
- Generic connection framework
- Application manager and MIDP applications
- Sun Java ME SDK 3.0
- Two examples of Midlets
The Mobile Ecosystem

- Operating Systems - Platforms
  - Devices
  - Networks
  - Operators
- Application Frameworks
- Applications
- Services
Operators

- Also called **Mobile Network Operators** (MNOs) or wireless **carriers**

- **Tasks**
  - Install cellular towers (and related infrastructure)
  - Operate the cellular network
  - Offer services for mobile subscribers
  - Maintain relations with subscribers
  - Handle billing and support
<table>
<thead>
<tr>
<th>Rank</th>
<th>Operator</th>
<th>Markets</th>
<th>Technology</th>
<th>Subscribers (in millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>China Mobile</td>
<td>China (including Hong Kong) and Pakistan</td>
<td>GSM, GPRS, EDGE, TD-SCDMA</td>
<td>436.12</td>
</tr>
<tr>
<td>2.</td>
<td>Vodafone</td>
<td>United Kingdom, Germany, Italy, France, Spain...</td>
<td>GSM, GPRS, EDGE, UMTS, HSDPA</td>
<td>260.5</td>
</tr>
<tr>
<td>3.</td>
<td>Telefónica</td>
<td>Spain, Argentina, Brazil, Chile, Colombia, ...</td>
<td>CDMA, CDMA2000 1x, EV-DO, GSM, GPRS, EDGE, UMTS, HSDPA</td>
<td>188.9</td>
</tr>
<tr>
<td>4.</td>
<td>Américas Móvil</td>
<td>United States, Argentina, Chile, Colombia, Paraguay, ...</td>
<td>CDMA, CDMA2000 1x, EV-DO, GSM, GPRS, EDGE, UMTS, HSDPA</td>
<td>172.5</td>
</tr>
<tr>
<td>5.</td>
<td>Telenor</td>
<td>Norway, Sweden, Denmark, Hungary...</td>
<td>GSM, GPRS, EDGE, UMTS, HSDPA</td>
<td>143.0</td>
</tr>
<tr>
<td>6.</td>
<td>China Unicom</td>
<td>China</td>
<td>GSM, GPRS</td>
<td>127.6</td>
</tr>
<tr>
<td>7.</td>
<td>T-Mobile</td>
<td>Germany, United States, United Kingdom, Poland...</td>
<td>GSM, GPRS, EDGE, UMTS, HSDPA</td>
<td>126.6</td>
</tr>
<tr>
<td>8.</td>
<td>TeliaSonera</td>
<td>Norway, Sweden, Denmark, Finland...</td>
<td>GSM, GPRS, EDGE, UMTS, HSDPA</td>
<td>115.0</td>
</tr>
<tr>
<td>9.</td>
<td>Orange</td>
<td>France, United Kingdom, Switzerland, Poland, Spain...</td>
<td>GSM, GPRS, EDGE, UMTS, HSDPA</td>
<td>111.8</td>
</tr>
<tr>
<td>10.</td>
<td>MTS</td>
<td>Russia, Ukraine, Belarus, Uzbekistan, Turkmenistan,..</td>
<td>GSM, GPRS, EDGE, UMTS</td>
<td>91.7</td>
</tr>
<tr>
<td>11.</td>
<td>MTN Group</td>
<td>Afghanistan, Benin, Botswana, Cameroon, Republic of Congo, Côte d’Ivoire, ...</td>
<td>GSM, GPRS, EDGE, UMTS, HSDPA, HSUPA</td>
<td>80.7</td>
</tr>
</tbody>
</table>
## Networks

<table>
<thead>
<tr>
<th></th>
<th><strong>Second generation of mobile phone standards and technology</strong></th>
<th><strong>Theoretical max data speed</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2G</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile communications</td>
<td>12.2 KB/sec</td>
</tr>
<tr>
<td>GPRS</td>
<td>General Packet Radio Service</td>
<td>Max 60 KB/sec</td>
</tr>
<tr>
<td>EDGE</td>
<td>Enhanced Data rates for GSM Evolution</td>
<td>59.2 KB/sec</td>
</tr>
<tr>
<td>HSCSD</td>
<td>High-Speed Circuit-Switched Data</td>
<td>57.6 KB/sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>3G</strong></th>
<th><strong>Third generation of mobile phone standards and technology</strong></th>
<th><strong>Theoretical max data speed</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>W-CDMA</td>
<td>Wideband Code Division Multiple Access</td>
<td>14.4 MB/sec</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
<td>3.6 MB/sec</td>
</tr>
<tr>
<td>UMTS-TDD</td>
<td>UMTS +Time Division Duplexing</td>
<td>16 MB/sec</td>
</tr>
<tr>
<td>TD-CDMA</td>
<td>Time Divided Code Division Multiple Access</td>
<td>16 MB/sec</td>
</tr>
<tr>
<td>HSPA</td>
<td>High-Speed Packet Access</td>
<td>14.4 MB/sec</td>
</tr>
<tr>
<td>HSDPA</td>
<td>High-Speed Downlink Packet Access</td>
<td>14.4 MB/sec</td>
</tr>
<tr>
<td>HSUPA</td>
<td>High-Speed Uplink Packet Access</td>
<td>5.76 MB/sec</td>
</tr>
</tbody>
</table>
What is a Phone?

- For the majority of people the perception of what is a phone has not changed.
- Actually the modern mobile phone is something totally different: is a communication and information device.
The Evolution of Devices

- **The Brick Era (1973-1988):** large, heavy, professional usage or for rich people
- **The Feature Phone Era (1998-2008):** better data connection GPRS and numerous features - 85% of current phones (2009)
- **The Smartphone Era (2002-now):** QWERTY keyboard, PDA like, push email, various Oss – 13% of current phones (2009)
- **The Touch Era (2007-now):** iPhone and followers
Nokia N96

- **Operating Frequency:** WCDMA2100/900 (HSDPA) / EGSM900, GSM850/1800/1900 MHz (EGPRS)
- **Memory:** 128MB RAM, 256MB system memory; 16GB internal flash memory; memory card slot
- **Display:** 2.8" QVGA (240 x 320 pixels)
- **Data Transfer:**
  - WCDMA HSDPA with simultaneous voice and packet data (PS max speed DL/UL= 3.6Mbps/384kbps, CS max speed 64kbps)
  - Dual Transfer Mode (DTM) support for simultaneous voice and packet data connection in GSM/EDGE networks. Simple class A, multi slot class 11, max speed DL/UL: 177.6/118.4kbps
  - EGPRS class B, multi slot class 32, max speed DL/UL= 296/177.6kbps
  - GPRS class B, multi slot class 32, max speed DL/UL= 107/64.2kbps
Nokia N96

- **Connectivity**
  - WLAN - IEEE802.11 g/b with UPnP support
  - Hi-Speed USB 2.0 with Micro USB type B interface
  - 3.5mm stereo headset plug, TV-out support (PAL/NTSC)
  - Bluetooth 2.0 with A2DP stereo audio and Enhanced Data Rates (EDR)
  - Nokia Nseries PC Suite connectivity with USB and Bluetooth wireless technology
  - Local synchronization of contacts and calendar to a compatible PC using compatible connection
  - Remote over-the-air synchronization
  - Send and receive images, video clips, graphics, and business cards via Bluetooth wireless technology.
Nokia N96

- **Applications**
  - Java MIDP 2.1, CLDC 1.1 (Connected Limited Device Configuration (J2ME))
  - Over-the-air download of Java-based applications and games
  - Flash Lite 3.0

- **Imaging and Video**
  - Up to 5 megapixel (2592 x 1944 pixels) camera - MPEG-4 Part 2 (H.263/SP), up to VGA 30 fps
  - Geotagging: automatic insertion of GPS-based location tags into images
  - Video call and video sharing support (WCDMA network services)
  - Online album/blog: photo/video uploading from gallery
  - Broadcast Television (DVB-H) capable
Nokia N96

- **Music Features**
  - Digital music player - supports MP3/AAC/AAC+/eAAC+/WMA/M4A with playlists and equalizer
  - Integrated handsfree speaker
  - OMA DRM 2.0 & WMDRM support for music
  - Stereo FM radio (87.5-108MHz /76-90MHz) with Visual Radio support and RDS

- **Navigation:** Built-in GPS

- **E-mail:** e-mail client with attachment support for images, videos, music and documents

- **Browsing:** Nokia Web Browser with Mini Map, visual history, HTML and JavaScript support, Flash Lite 3.0 and Flash video support, RSS reader.
Modu

- Slightly larger than a domino
- Capable of sending and receiving calls and text messages
- Can store contacts and MP3s with up to 16 gigabytes of storage capacity
- Small but usable screen and a sparse keypad that lacks numbers
- It can be slipped into a variety of "jackets," such as in-car MP3 players, GPS, and larger cell phones, that expand the Modu's functions and change its look.

http://www.technologyreview.com
4G - Fourth Generation

- **4G**: Fourth-Generation describes the next complete evolution in wireless communications.
- **4G networks will come in 2012-2015**
- 4G will be a fully IP-based integrated system.
- 4G will be capable of providing between **100 Mbit/s and 1 Gbit/s speeds** both indoors and outdoors, with premium quality and high security.
Effects of device portability

- **Power consumption**
  - limited computing power, low quality displays, small disks due to limited battery capacity
  - CPU: power consumption $\sim CV^2f$
    - $C$: internal capacitance, reduced by integration
    - $V$: supply voltage, can be reduced to a certain limit
    - $f$: clock frequency, can be reduced temporally

- **Loss of data**
  - higher probability, has to be included in advance into the design (e.g., defects, theft)

- **Limited user interfaces**
  - compromise between size of fingers and portability
  - integration of character/voice recognition, abstract symbols

- **Limited memory and computing power**
  - limited RAM, and CPU
Computers for the next decades?

- Computers are **integrated**
  - small, cheap, portable, replaceable
- Technology is in the **background**
  - computer are aware of their environment and adapt (“location awareness”)
  - computer recognize the location of the user and react appropriately (e.g., call forwarding, fax forwarding, “context awareness”)
- Advances in technology
  - more computing power in smaller devices
  - flat, lightweight displays with low power consumption
  - new user interfaces due to small dimensions
  - more bandwidth per cubic meter
  - multiple wireless interfaces: wireless LANs, wireless WANs, regional wireless telecommunication networks etc. („overlay networks“)
The Mobile Ecosystem

Services

Applications

Application Frameworks

Operating Systems - Platforms

Devices

Networks

Operators
Mobile Operating Systems - Platforms

- **Palm OS**: advanced OS now supporting webOS that is based on the WebKit browser framework
- **Windows Mobile**: compact version of the Windows OS
- **Symbian**: open source OS with libraries, user interface, frameworks, reference implementation of common tools
- **Linux**: used in some phones e.g. Motorola RAZR2 or larger mobile devices as Nokia 900
- **Mac OS X**: iPhone and iPod touch (unix based)
- **Android**: unix based – a fast growing set of devices are using it
Application Frameworks (I)

- **Java**: Java ME can be deployed – purchase through the operator or for free over the air – across the majority of devices (Java-Based)
- **S60**: application framework for devices sunning Symbian OS. Runs mostly on Nokia phones. Applications are created on Java, Symbian C++ or Flash Lite
- **BREW**: Binary Runtime Environment for Wireless
  - Download and run small programs (C or C++) for playing games, sending messages, and sharing photos
  - Applications can be ported among all Qualcomm devices
  - Applications must go through a costly certification process – trough the operator.

Application Frameworks (II)

- **Windows Mobile**: applications written for Win32 API can be deployed on the majority of Mobile-based devices.

- **Cocoa Touch**
  - API for creating native applications for the iPhone and iPod touch.
  - Applications must be submitted and certificated by Apple before they can be included in the AppStore.

- **Android SDK**: based on java – exploit “activities” offered and shared by the framework – e.g. map visualization.
Application Frameworks (III)

- **Web Runtimes (WRTs)**
  - Provided by Nokia, Opera and Yahoo!
  - Miniframeworks for creating mobile widgets based on web standards (XHTML+JS+CSS)
  - They are programmed in a SDK and are distributed as Symbian applications (OVI)
  - Widgets run in WRT (not in the browser), a web-application runtime environment that is part of the S60 Browser – the phone must support it!
  - PC widgets can be ported to WRT
  - Widget can combine information from the internet with data stored on a device

http://www.forum.nokia.com/Develop/Web/
Application Frameworks (IV)

- **The Web**
  - The only application framework that work across all devices and operating systems
  - Applications are built using web standards
    - WML
    - XHTM-MP
    - Java script
    - CSS
  - A variety of web standards support in different devices and the characteristics of devices makes building mobile web applications difficult.
Mobile Applications

- Mobile applications are designed to support services
- They can be pushed to a mobile device or downloaded and installed locally or they may rely on the browser
- Classification – technology based
  - SMS
  - Mobile Websites
  - Mobile Web Widget
  - Mobile Web Applications
  - Native Applications
  - Games
The **simplest** – but more complex applications, e.g., native ones, can use it as a component

**Examples**

- Self service provisioning: “INTERNET YES” to 4033 and you get roaming for one month for 3 euro
- Game and ringtones requesting and paying
- On twitter you can receive SMS alerts from friends or post tweets
- Notifications when making a purchase with the credit card.
Mobile Websites

- A website designed specifically for mobile devices
- Simple architecture, presentation and navigation links
- Mobile websites are typically informational
- **Easy to create** but fail to **display consistently** across multiple web browsers
- Is becoming increasingly used
- **Better browsers** are stimulating the development of new sites.
Mobile Web Widgets

- Introduced in response to the poor experience provided by the mobile web in the past
- Web application that must be downloaded and installed
- The difference between widgets and mobile applications that run in the browser is subtle
- They require a specific widget platform on the device
- They are built with (proprietary) techniques that extend web standards
- They are used to support short, task-based operations
- Examples: Java ODP, Web Runtimes.
Mobile Web Applications

- They do not need to be installed or compiled for the target device
- Developed using XHTM, CSS and JavaScript
- They provide application-like experience – they do not use the drill-down or page metaphors
- The challenge is device fragmentation
- Rendering of CSS2 is inconsistent, support for JavaScript simple
- Only recently some phones (iPhone) are supporting better them.
Native Applications

- Platform applications – they are developed and compiled for each mobile platform.
- The most common native application platform is **Java ME**.
- Other Smartphone programming languages are C, C++ (Symbian), Objective-C (iPhone), Java (Android).
- You must decide the platform.
- Native applications must be tested, certificated and distributed.
- They can work off-line, access the file system, use the camera, the music player, etc.
Consumer Devices and Embedded Systems

- **One solution does not fit all**: consumer devices are highly specialized for the intended use
- Diverse **range of existing applications** and features
- **Users/developers want flexibility**: they want to choose what they want to use and what they don’t
- *The performance of a consumer device is not just measured by the computing power but how well it serves the intended usage*
- Factors differentiating consumer devices from desktop computers
  - **Small screen size**
  - **Different usage models**: stylus, tiny keypad, small QWERTY keyboards, voice operated
  - **Mobility**: in traffic, while skiing, etc.
  - **Limited network bandwidth with intermittent connections.**
Configuration

- A **configuration** is a complete Java runtime environment:
  - Java virtual machine (VM) to execute Java
  - Set of core Java runtime classes
  - Interface to the underlying system
- Defines a **minimum** platform for a “horizontal” category or grouping of devices with similar requirements on memory and processing power
- A J2ME **application** is written for a particular **profile** and a **profile** is **based upon** or **extends** a particular **configuration**
- The CLDC/MIDP stack is based on the open source project PhoneME™ at https://phoneme.dev.java.net/
**CDC (Connected Device Configuration)**

- **CDC** (Connected Device Configuration): high-end consumer devices (TV set-top boxes, Internet TV)
  - 512KB of read-only-memory (ROM), 256 KB of random access memory (RAM), **minimum**
  - 32-bit processor
  - High bandwidth network connection
  - Full-featured Java2 virtual machine (CVM)
  - 17 packages
  - Use for devices like Palms

- **Most of the core APIs are identical between CDC and J2SE 1.3.1.**
  - The main differences are in java.awt and the omission of javax.swing
Configuration: CLDC

- **CLDC (Connected Limited Device Configuration):** low-end consumer devices - cell phones, two-way pagers, personal digital assistants (PDAs), organizers, home appliances, and point of sale terminals
- 160 - 512 KB of total memory (**160KB ROM** and **32KB RAM**, minimum)
- 16-bit or 32-bit processor
- Low power consumption and often operating with battery power
- Connectivity with limited bandwidth
- Selected classes from: java.lang, java.io, java.util
- Limited VM (called **KVM**):
  - NO Object finalization
  - NO JNI (Java Native Interface) or reflection
  - NO Thread groups or daemon threads
  - NO User Class loaders
Relationships between J2ME conf. and J2SE
Profile and Optional Packages

- The **profile** adds classes to a configuration:
  - To fill in missing functionality
  - To support specific uses of a device
  - To address the specific demands of a vertical market sector, e.g., cellular telephones, washing machines, electronic toys

- The **Optional Packages** are set of APIs that support additional and common behaviors
  - Examples of optional packages:
    - Bluetooth Optional Package
    - JDBC Optional Package
    - File connection
    - Personal Information Management (PIM)
    - Location API
Profiles

Several profiles in various stages of development:

- **Mobile Information Device Profile (MIDP)** - CLDC-based, used for running applications on cell phones and interactive pagers with small screens, wireless HTTP connectivity, and limited memory.

- **Foundation Profile (FP)** – CDC-based, is a set of Java APIs that support resource-constrained devices without a standards-based GUI.

- **Personal Basis Profile (PBP)** – CDC is a set of Java APIs that support resource-constrained devices with a standards-based GUI framework based on lightweight components.

- **Personal Profile (PP)** - extends the PBP lightweight (AWT-derived) user interface classes and a new application model with applet support and heavyweight UI classes (nokia 9300i).

- Check on [http://jcp.org/](http://jcp.org/) the state of these specifications.
Optional Packages for the Wireless Market

- JSR 120: Wireless Messaging API
- JSR 135: Mobile media API
- JSR 172: J2ME Web Services Specification
- JSR 177: Security and Trust Services Specification
- JSR 179: Location API for J2ME (many students used that last year)
- JSR 082: Bluetooth
- JSR 075: PDA optional
- JSR 184: Mobile 3D Graphics for J2ME
- JSR 226: SVG Scalable Vector Graphics
- JSR 190: Event Tracking API for J2ME – monitoring and tracking MIDlets
JTWI

- JSR-185: Java Technology for Wireless Industry - umbrella specification defined in 2003
MSA Mobile Service Architecture JSR248

### MSA:
- JSR 238 (Internationalization)
- JSR 234 (Multimedia Supplements)
- JSR 211 (Content Handler)
- JSR 180 (SIP)
- JSR 179 (Location)
- JSR 177 (Security & Trust)
- JSR 172 (Web Services)
- JSR 226 (Vector Graphics)
- JSR 205 (Messaging)
- JSR 184 (3D Graphics)
- JSR 135 (Mobile Media)
- JSR 82 (Bluetooth)
- JSR 75 (File & PIM)
- **JSR 118 (MIDP)**
- **JSR 139 (CLDC)**

### MSA Subset:
- JSR 226 (Vector Graphics)
- JSR 205 (Messaging)
- JSR 184 (3D Graphics)
- JSR 135 (Mobile Media)
- JSR 82 (Bluetooth)
- JSR 75 (File & PIM)
- **JSR 118 (MIDP)**
- **JSR 139 (CLDC)**

Version 1.1.0b – 18-August-2008
Only a few phones (8) support the full MSA
Hardware Requirements: MIDP

- **Memory:**
  - 256Kb non-volatile for MIDP components (in addition to the requirements of CLDC),
  - 8Kb non-volatile for application created persistent data,
  - 128 Kb volatile for virtual machine run time

- **Display:** 96x54, depth 1-bit, pixel shape 1:1

- **Input:** either keypad, or keyboard, or touch screen

- **Networking:** two-way, intermittent, with limited bandwidth

- **Sound:** play tones.
Software Requirements: MIDP

- **Minimal kernel** to manage the underlying hardware (interrupts, exceptions, and minimal scheduling)
- Mechanism for **reading and writing from non-volatile memory** (to support persistence API)
- **Read and write access to devices' wireless networking** (to support networking API)
- A mechanism to **time-stamping the records** written in the persistence storage
- Support to write a **bit-mapped graphic display**
- Mechanism to **capture user input** from keypad or touch screen.
Security

- **Low-level security** (virtual machine security): ensure that the application running in the JVM follows the semantic of the java prog. language (malicious classes must not harm the device)

- **Class file verifier** ensures that the bytecode:
  - cannot contain illegal instructions,
  - cannot be executed in an illegal order, and
  - cannot contain references to invalid memory locations

- **Application security**: Java application running on the device can access only those libraries, system resources, and components that the device and Java environment allow to access.

*Midlet can be downloaded from Internet and may not be certified or signed*
Malware detection in mobile phones

- Mobile devices lack the processing power to scan for large numbers of signatures.

**Approach:**

- First shutting off non-vital applications, such as an e-mail app or a browser.
- Nothing should be running except the detection software and the operating system itself.
- If malware is present and active, it will need to use some RAM to execute instructions on the device.
- The central server contacts the detection software to check to see if malware is using RAM by measuring how much memory is available.

**CLDC 1.1** and **MIDP 2.0** packages

<table>
<thead>
<tr>
<th>MIDP 2.0</th>
<th>CLDC 1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>javax.microedition.lcdui</code></td>
<td><code>java.lang</code></td>
</tr>
<tr>
<td><code>javax.microedition.lcdui.game</code></td>
<td><code>java.lang.ref</code></td>
</tr>
<tr>
<td><code>javax.microedition.media</code></td>
<td><code>java.io</code></td>
</tr>
<tr>
<td><code>javax.microedition.media.control</code></td>
<td><code>java.util</code></td>
</tr>
<tr>
<td><code>javax.microedition.midlet</code></td>
<td><code>javax.microedition.io</code></td>
</tr>
<tr>
<td><code>javax.microedition.pki</code></td>
<td></td>
</tr>
<tr>
<td><code>javax.microedition.rms</code></td>
<td></td>
</tr>
</tbody>
</table>

Latest MIDP2.1 has minor differences with 2.0: making LCDUI layout directive mandatory, `javax.microedition.io.SocketConnection` and `javax.microedition.io.HTTPConnection` is no longer optional.
Devices Evolution (Nokia)

6600 (2003)
- MIDP 2.0
- CLDC 1.0
- Bluetooth API (JSR-82 No OBEX)
- Mobile Media API (JSR-135)
- Nokia UI API
- Wireless Messaging API (JSR-120)

N70 (2005)
- MIDP 2.0
- CLDC 1.1
- Bluetooth API (JSR-82)
- FileConnection and PIM API (JSR-75)
- JT WI (JSR-185)
- Mobile 3D Graphics API (JSR-184)
- Mobile Media API (JSR-135)
- Nokia UI API
- Web Services API (JSR-172)
- Wireless Messaging API (JSR-120)

N95 (2007)
- MIDP 2.0
- CLDC 1.1
- Advanced Multimedia Supplements (JSR-234)
- Bluetooth API (JSR-82)
- FileConnection and PIM API (JSR-75)
- JT WI (JSR-185)
- Location API (JSR-179)
- Mobile 3D Graphics API (JSR-184)
- Mobile Media API (JSR-135)
- Nokia UI API
- Scalable 2D Vector Graphics API (JSR-226)
- Security and Trust Services API (JSR-177)
- SIP API (JSR-180)
- Web Services API (JSR-172)
- Wireless Messaging API (JSR-205)
What device?

- If you want to know what devices support what profile/configuration/package go to the WTK3.0 and select "Tools>Device Database Search"

- It is based on WURFL
  - The WURFL is an "ambitious" configuration file that contains info about all known Wireless devices on earth
    - [http://wurfl.sourceforge.net](http://wurfl.sourceforge.net)
CLDC 1.1. Class Library

- Classes that are a subset of standard J2SE:
  - `java.lang.*`, `java.util.*`, `java.io.*`, `java.lang.ref`
  - A class with the same name and package name as a J2SE class must be identical to or a subset of the corresponding J2SE class
  - The classes cannot add any public or protected methods or fields that are not available in J2SE

- Classes that are specific to CLDC
  - `javax.microedition.io`
CLDC Classes (subset of J2SE)

- **System Classes**
  - java.lang.Object
  - java.lang.Class
  - java.lang.Runtime
  - java.lang.System
  - java.lang.Thread
  - java.lang.Runnable (interface)
  - java.lang.String
  - java.lang.StringBuffer
  - java.lang.Throwables

- **Data Types Classes**
  - java.lang.Boolean
  - java.lang.Byte
  - java.lang.Short
  - java.lang.Integer
  - java.lang.Long
  - java.lang.Float
  - java.lang.Double
  - java.lang.Character

- **Collection Classes**
  - java.util.Vector
  - java.util.Stack
  - java.util.Hashtable
  - java.util.Enumeration (interface)

New in CLDC1.1: java.lang.Float, java.lang.Double
CLDC Classes (subset of J2SE)

- **IO Classes**
  - java.io.InputStream
  - java.io.OutputStream
  - java.io.ByteArrayInputStream
  - java.io.ByteArrayOutputStream
  - java.io.DataInput (interface)
  - java.io.DataOutput (interface)
  - java.io.DataInputStream
  - java.io.DataOutputStream
  - java.io.Reader
  - java.io.Writer
  - java.io.InputStreamReader
  - java.io.OutputStreamWriter
  - java.io.PrintStream

- **Calendar and Time Classes**
  - java.util.Calendar
  - java.util.Date
  - java.util.TimeZone

- **Utility classes**
  - java.util.Random
  - java.lang.Math

- **Exception and Error classes**
  - See the specification!
CLDC Specific Classes: javax.microedition.io

- The package java.net of JDK contains 31 classes and interfaces and 8 exception classes
- It is difficult (and not useful) to make all this functionality fit in a small device with only few hundreds Kbs of memory
- There is a plethora of wireless technologies in use with varying levels of sophistication, compatibility and interoperability (GSM, TDMA, CDMA, WCDMA, UMTS, GPRS, EDGE, ...)
- J2ME standardization efforts is to define solutions that can work effectively with all these network technologies and standards
- J2ME (in CLDC) defines a Generic Connection Framework – connect to Internet irrespectively to the wireless communication technology.
MIDlets – The heart of J2ME

- MIDP does not run in the “regular” Java fashion using: `main()`, `System.exit()`
- Instead, we use **MIDlet** applications - which are subclasses of `javax.microedition.midlet.MIDlet`
- The application must extend this class to allow the **application management** software to control the MIDlet:
  - control the MIDlet installation
  - **Inspect** existing Java applications stored on the device
  - be able to retrieve properties from the application descriptor
  - **Select** and **launch** Java applications; respond to a request for state change
  - **Delete** existing applications
- A CLDC system may allow multiple Java applications to execute concurrently (**MIDP2.1**) or restrict to one application at a time (**MIDP2.0**).
MIDP Application Lifecycle

- **MIDlets** move from state to state in the lifecycle – it is the application manager (AM) or the midlet that changes the state

- **Pause**: after the constructor
  - `pauseApp()` called by AM or the midlet: signals the MIDlet to enter the *paused* state
  - `notifyPaused()`: notifies the AM that the MIDlet does not want to be active and has entered the *Paused* state

- **Active**
  - The AM has called `startApp()`
  - The midlet has called `resumeRequest()`: indicate that it is interested in entering the *active* state

- **Destroyed**
  - The AM or the midlet has called `destroyApp()`: signals the MIDlet to terminate and enter the *destroyed* state
  - `notifyDestroyed()`: the midlet notifies the AM that has entered the destroyed state.
MIDlet Suite

- One or more MIDlets are packaged together into a **MIDlet suite**, composed of:
  - **JAR (Java archive) file**
    - Contains **Java classes** for each MIDlet in the suite and Java classes that are shared between MIDlets
    - Contains **resource files** (e.g. an image) used by the MIDlets and a **manifest file**
  - **JAD (Java Application Descriptor) file**
    - Contains a predefined set of attributes that allows the device application management software to identify, retrieve, and install the MIDlets
    - Can be modified after packaging (and signing)
- Eventually the JAR / JAD files are **uploaded to the device** in order to run the application.
Wireless Development Tutorial Part I

- **What do we need**
  - **Java Platform, Standard Edition version** 1.5 or higher
  - **Sun Java Micro Edition SDK** This is a package of tools for building and testing MIDlets
  - **Text editor.** This can be something as rudimentary as Notepad (on Windows) or something more elaborate (IDE environment as NetBeans)

- Following example is from
Java ME SDK

- Download the Java ME SDK 3.0 from www.oracle.com/technetwork/java/javame/
- Execute the installation file
- There is a very good user guide
Java ME SDK works with projects, where the end result of each project is one MIDlet suite.
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class HelloMIDlet
    extends MIDlet
    implements CommandListener {
    private Form mMainForm;

    public HelloMIDlet() {
        mMainForm = new Form("HelloMIDlet");
        mMainForm.append(new StringItem(null, "Hello, MIDP!");
        mMainForm.addCommand(new Command("Exit", Command.EXIT, 0));
        mMainForm.setCommandListener(this);
    }

    public void startApp() {
        Display.getDisplay(this).setCurrent(mMainForm);
    }

    public void pauseApp() {}    public void destroyApp(boolean unconditional) {}  public void commandAction(Command c, Displayable s) {
        notifyDestroyed();
    }
}
Hello World

- Right click on the project and select New -> MIDlet
- Input the midlet name (class) and click on finish

*Note: New MIDlets are automatically added to the application descriptor.*
Write the Code and Build

- Copy the code in the generated file

Click here to build - or right click on the project and select “build”
Running

- Click on the **Run** button
- You should see a phone emulator
- The emulator is showing a list of MIDlets if there are more than one – otherwise it starts the unique MIDlet
1) Build

- What happens when you press the **Build** button?
- The toolkit finds all the `.java` files in the `src` directory of your project and **compiles** them.
- Source files must be compiled in a MIDP environment rather than a J2SE 5.0 environment.
- For instance a MIDlet that uses the `java.lang.System` class: this class has different APIs in J2SE 5.0 and MIDP.
- When the toolkit compiles your MIDlet class it uses the MIDP `java.lang.System`, not J2SE 5.0 version of the class.
- You could make this selection yourself (if you installed the MIDP reference implementation), using the command `javac` and the `-bootclasspath` option:
  - `javac -bootclasspath hellomidlet.java`
2) Preverifying Class Files

- The toolkit performs an **initial verification** at build time (preverifying)
  - Certain checks are performed and the class file is modified in such a way that the second-step (runtime) can be easily handled
- The device's runtime system performs a second verification when it loads the classes
  - If a class file has not *preverified* it is rejected
- You could perform the first verification yourself using the command line **preverify** tool.
3) JARing

- Finally, MIDlets are bundled into MIDlet suites for distribution to actual devices **3) package**
- Bundling entails JARing the MIDlet suite class files and the resource files, and putting some extra information in the JAR manifest
- Finally the files are **4) deployed** on the device
- The above steps are not required for running the application in the Wireless Toolkit (*actually WTK3.0 always build a package*)
- But are required if you want to deploy the MIDlet suit on a mobile device.
Deploying MIDlets

- MIDlets can be deployed on a phone in two ways:
  - Transfer the jar and jad files to the phone from the computer via an **external connection**: serial cable, USB cable, IRDA, Bluetooth
  - **Over the Air (OTA) provisioning**: download the midlet suite from a server

- Installation is specific to the device!
- Check the documentation of your device to see how to install a MIDlet suite
- More on these topics in the LABS!
Manifest Information

- Every JAR includes a manifest file `META-INF/MANIFEST.MF`

  MIDlet-1: Hellosuite, Hellosuite.png, HelloMIDlet
  MIDlet-2: HitMIDlet, , HitMIDlet
  MIDlet-Name: Hellosuite
  MIDlet-Vendor: Unknown
  MIDlet-Version: 1.0
  MicroEdition-Configuration: CLDC-1.1
  MicroEdition-Profile: MIDP-2.0

- It describes the content of the archive
- It may contain extra information that is important to the MIDP runtime environment (e.g. a URL to connect).
MIDlet Suite Descriptor

- Before a midlet can be deployed an additional file is required: an application description, a \texttt{.jad} file.
- The \texttt{.jad} file contains a lot of the same information that’s in the \texttt{manifest} file.
- The application descriptors contains information that help the device and/or the user to decide whether or not to load a MIDlet suite.
- It can be downloaded and examined before downloading the \texttt{.jar}.
- Useful in OTA provisioning – the server returned MIME type for the \texttt{.jad} file must be \texttt{text/vnd.sun.j2me.app-descriptor}. 
Hellosuite.jad

HitMIDlet.URL: http://localhost:8080/midp/hits
MIDlet-1: Hellosuite, Hellosuite.png, HelloMIDlet
MIDlet-2: HitMIDlet, , HitMIDlet
MIDlet-Jar-Size: 3016
MIDlet-Jar-URL: Hellosuite.jar
MIDlet-Name: Hellosuite
MIDlet-Vendor: Unknown
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
Connection with a Servlet

- Install **NetBeans**

  - *Remember at the beginning of the installation to choose “customize” installation and deselect GlassFish and select Tomcat*

- We need to develop simple Web applications based on servlets

- Create a new java web project with the servlet shown in the next slide
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;

public class HitServlet extends HttpServlet {
    private int mCount;

    public void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws ServletException, IOException {
        String message = "Hits: " + ++mCount;

        response.setContentType("text/plain");
        response.setContentLength(message.length());
        PrintWriter out = response.getWriter();
        out.println(message);
    }
}
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class HitMIDlet
    extends MIDlet
    implements CommandListener {
private Display mDisplay;
private Form mMainForm;
private StringItem mMessageItem;
private Command mExitCommand, mConnectCommand;

public HitMIDlet() {
    mMainForm = new Form("HitMIDlet");
    mMessageItem = new StringItem(null, "");
    mExitCommand = new Command("Exit", Command.EXIT, 0);
    mConnectCommand = new Command("Connect", Command.SCREEN, 0);
    mMainForm.append(mMessageItem);
    mMainForm.addCommand(mExitCommand);
    mMainForm.addCommand(mConnectCommand);
    mMainForm.setCommandListener(this);
}
public void startApp() {
    mDisplay = Display.getDisplay(this);
    mDisplay.setCurrent(mMainForm);
}

public void pauseApp() {}

public void destroyApp(boolean unconditional) {}

public void commandAction(Command c, Displayable s) {
    if (c == mExitCommand)
        notifyDestroyed();
    else if (c == mConnectCommand) {
        Form waitForm = new Form("Waiting...");
        mDisplay.setCurrent(waitForm);
        Thread t = new Thread() {
            public void run() {
                connect();
            }
        };
        t.start();
    }
}
private void connect() {
    HttpConnection hc = null;
    InputStream in = null;
    String url = getAppProperty("HitMIDlet.URL");

    try {
        hc = (HttpConnection)Connector.open(url);
        in = hc.openInputStream();

        int contentLength = (int)hc.getLength();
        byte[] raw = new byte[contentLength];
        int length = in.read(raw);

        in.close();
        hc.close();

        // Show the response to the user.
        String s = new String(raw, 0, length);
        mMessageItem.setText(s);
    }
    catch (IOException ioe) {
        mMessageItem.setText(ioe.toString());
    }
    mDisplay.setCurrent(mMainForm);
}
MIDlet Properties

- Attributes that have meaning in a MIDlet can be added to the manifest and/or the application descriptor files.
- It is more convenient to add an attribute to the application descriptor – it may be changed without touching the application files (user defined attributes only in JAD).
- If an attribute is listed in both files the value in the application descriptor will be used.
- A MIDlet can retrieve the values of these attributes using `getAppProperty()`.
- Example:
  - `HitMIDlet.URL: http://localhost:8080/midp/hits` in the `Hellosuite.jad`
  - `String url = getAppProperty("HitMIDlet.URL")` // in the code
Add an Attribute to a Suite

- Add this property **only to the JAD**
Running

After 4 clicks on the 'connect' command
MIDP 3.0 (still a JSR -complete)

- JSR 271: Mobile Information Device Profile 3
- **Enable multiple concurrent MIDlets in one VM**
- Specify proper firewalling, runtime behaviors, and lifecycle management issues for MIDlets
- Enable background MIDlets (e.g. UI-less)
- Enable ?auto-launched? MIDlets (e.g. started at platform boot time)
- **Enable inter-MIDlet communications**
- Enable shared libraries for MIDlets
- Improve UI expressability and extensibility
- **Better support for devices with larger displays**
- Enable MIDlets to draw to secondary display(s)
- Enable richer and higher performance games
- Secure RMS stores
- **Removable/remote RMS stores**
- IPv6
- Multiple network interfaces per device
- Specify standard ways for doing MIDlet provisioning through other means (e.g. OMA (SyncML) DM/DS, Bluetooth, removable media, MMS, JSR-232, etc.)
- Extensive device capabilities query
- Localization & Internationalization

http://java.sun.com/developer/technicalArticles/javame/midp3_enhance/
Exercises

- Install Java ME SDK and NetBeans
- Follow the slides and install, compile, and run the two midlets: HelloMIDlet.java, HitMIDlet.java (the code is on the course web site)
- First install the two midlets in Java SDK and then in NetBeans
- Write a MIDlet that displays the current date and time – use the HelloMIDlet.java code and the class Calendar (consult the MIDP API in your J2ME SDK install directory or in Netbeans)
- Write a new MIDlet that asks a servlet to return the IPAddress of the server, the name of the student, and a timestamp (use the Java classes InetAddress and Calendar).