Mobile Phone Programming

Free Study Activity
Day 3

Part 1
Java 2 Micro Edition (J2ME) overview

- Introduction
- J2ME architecture
Introduction

J2ME key factors

- Portability:
  - Write once run anywhere
- Security:
  - Code runs within the confines of its JVM
- Rich set of APIs
- Very quick learning time
- Huge number of possible users
**Market trends**

- 708 million mobile Java devices shipped by June 2005
- 635 mobile Java device models on the market
- 32 mobile device vendors using Java
- 45000 mobile Java applications on the market
- 180 operators worldwide that have developed Java services
- 23 million mobile Java downloads globally per month in 2005

---

**J2ME: Games and...much more**

- At the beginning J2ME was intended as a platform for game developing
- Now, with the set of APIs provided by J2ME it is possible to easily develop different kinds of applications (enterprise, multimedia, music applications)
• Java 2, Enterprise Edition (J2EE)
  – Multiuser, enterprise-wide applications
  – Based on J2SE + several APIs for server-side computing

http://mobilephones.kom.aau.dk
Java Family

- Java 2, Micro Edition (J2ME)
  - Designed for small capabilities devices that cannot support a full J2SE implementation

Java Card
  - Provides a secure environment for applications that run on smart cards and other devices with very limited memory and processing capabilities.
J2ME architecture

- **KVM (Kilo Virtual Machine):**
  - A compact, portable Java virtual machine intended for resource-constrained devices.

- **CLDC (Connected Limited Device Configuration):**
  - Provides the most basic set of libraries, virtual machine features and APIs for resource-constrained devices.

- **MIDP (Mobile Information Device Profile):**
  - A collection of different APIs and libraries offering the core functionality required by mobile applications, such as user interface, networking support and persistent storage.

- **Optional Packages:**
  - Additional APIs for more advanced features.

**Operating System:**
- J2ME works on the top of the OS of the device.
KVM

- The KVM is a layer of software that runs on top of the host operating system.
- It is implemented in C and it is responsible for the interpretation of the Java byte-code and translating this into native system calls.
  - This makes J2ME programs highly portable between different computing architectures and different types of embedded devices.

CLDC

- CLDC provides the lowest common denominator for small and resource-constrained devices characterized as follows:
  - 160 kB to 512 kB of totally memory budget available for the Java platform.
  - A 16-bit or 32-bit processor.
  - Low power consumption, often operating with battery power.
  - Connectivity to some kind of network, often with a wireless connection and with limited (often 9600 bps or less) bandwidth.
- There are 2 versions of CLDC:
  - CLDC 1.0
  - CLDC 1.1

http://mobilephones.kom.aau.dk
MIDP

- MIDP adds an additional layer on the top of the CLDC providing APIs for a specific class of devices:
  - 128KB of non-volatile memory for the MIDP implementation
  - 32KB of volatile memory for the runtime heap
  - 8KB of non-volatile memory for persistent data
  - Input capacity (screen, keyboard, touch)
  - Screen size of at least 96x54 pixels
- Currently there are two versions of MIDP:
  - MIDP 1.0
  - MIDP 2.0

Some Optional Packages

- Information Module Profile (IMP), JSR 195
- Wireless Messaging API (WMA); JSR 120, JSR 205
- Mobile Media API (MMAPI), JSR 135
- Location API for J2ME, JSR 179
- SIP API for J2ME, JSR 180
- Security and Trust Services API for J2ME, JSR 177
- Mobile 3D Graphics, JSR 184
- J2ME Web Services APIs (WSA), JSR 172
- Bluetooth API, JSR 82
- J2ME RMI, JSR 66
- JDBC for CDC/Foundation Profile API, JSR 169
Java Standardization Process

• Java architecture and editions are specified by the Java Community Process (JCP)
• The JCP is an open forum where all interested parties can participate
• The JCP is guided by the Executive Committee
• Members of the EC are chosen by election
• Requests for new functionality in specifications is called a Java Specification Request (JSR)

MIDlet

• MIDP applications are called MIDlets
• Every application must extend `javax.microedition.midlet.MIDlet` class
• The MIDlet class defines abstract methods that the main class implements (for example: `startApp()`, `destroyApp()`, `notifyDestroyed()`).
MIDlet

- One or more MIDlets are packaged together into a *MIDlet suite*, composed of:
  - JAR (Java archive) file
  - JAD (Java Application Descriptor) file
- All the user-defined classes and resources required by the suite's MIDlets must be in the JAR file.

AMS

- The Application Management Software (AMS) manages the MIDlets.
- The AMS is a part of the device's operating environment and guides the MIDlets through their various states during the execution process.
MIDlet states

- AMS invokes MIDlet constructor
- AMS invokes startApp()
- AMS invokes pauseApp()
- AMS invokes destroyApp()
- AMS reclaims MIDlet

- AMS calls startApp() to place MIDlet in an active state
- AMS calls pauseApp() to place MIDlet in a paused state
- AMS calls destroyApp() to place MIDlet in a destroyed state

The user launches the MIDlet:
- AMS creates a new instance of the MIDlet class (call constructor)
- MIDlet is placed in PAUSE state

MIDlet skeleton example

```java
import javax.microedition.midlet.*;
public class Example extends MIDlet {
    public void startApp() {
    }
    public void destroyApp(boolean unconditional) {
    }
    public void pauseApp() {
    }
}
```

http://mobilephones.kom.aau.dk
Data Storage: RMS

- Several applications need to store some data in a permanent way (even when the device is switched off).
- This kind of data is called “persistent data” and they are stored in a dedicated memory part called “persistent storage.”
- MIDP defines a set of classes for storing and retrieving persistent data called Record Management System (RMS).

Storage mechanism

- The basic storage mechanism used by the RMS is a “Record Store”.
- A Record Store can be shared among different MIDlets (only MIDP 2.0).
- When a MIDlet suite is removed from the device then all the record stores associated with it will also be removed.
Storage mechanism

- A Record store is a collection of records:
  - A record is uniquely identified by its Record ID
  - A record is an array of bytes
- The value of RecordId for the first record created is 1
- RecordId for new records will monotonically increase
- A RecordId is never reused:
  - If a record is deleted from the Record Store, its RecordId will never be assigned to any other record