

## Applications developed with J2ME MIDP 1.0 & 2.0

Yun Guo , University of Stuttgart  
Yunguo909@hotmail.com  
Supervisor: Stephan Rupp

**Abstraction – The Java™ 2 Platform, Micro Edition (J2ME) is the Java platform for consumer and embedded devices. Its architecture defines configurations and profiles. The Mobile Information Device Profile (MIDP) is designed for mobile phones and entry – level PDAs. This paper explains the principles and options for communication services, compares the MIDP 1.0 and MIDP 2.0, shows it’s limits, and summarized the types of applications.**

**Keywords: J2ME, MIDP, CLDC, MIDlet, GCF, KVM**

### 1 . INTRODUCTION

The J2ME MIDP is the Java runtime environment for today’ s mobile information devices, and it makes new applications available, such as interactive games and mobiles commerce. The paper is organized as follows. Chapter 2 gives an overview on J2ME and its architecture. Chapter 3 introduces the Connected, Limited Device Configuration (CLDC). Chapter 4 explains the MIDP and its main concepts, such as MIDlets, the components of MIDP (user interface, networking libraries, and persistent libraries), the APIs of MIDP 2.0 and the limitations of MIDP. Chapter 5 has a look on the standardization process – Java Community Process (JCP), where the MIDP specification and APIs comes from; Chapter 6 summarizes the applications of this technology, such as games and mobile clients for business applications. Finally, chapter 7 makes conclusions.

### 2. Overview of Java™ 2 platform Micro Edition (J2ME) ...

“The Java™ 2 Platform Micro Edition (J2ME) is the Java platform for consumer and embedded devices such as mobile phones, PDAs, TV set-top boxes, in-vehicle telematics systems, and a broad range of embedded

devices “ [1]. The J2ME contains a virtual machine and a set of APIs (Application Programming Interface), which provide a suitable runtime environment for the consumer and embedded devices.

In order to provide a runtime environment for a broad range of devices, the J2ME architecture defines configurations, profiles and optional packages. The two primary concepts of J2ME architecture are configurations and profiles:

- **Configuration:** a J2ME configuration defines a minimum platform for a “horizontal’ category or grouping of devices, each with similar requirements on total memory budget and processing power [2]. Configurations consist of a virtual machine and a minimum set of class libraries, which provide basic functions for a particular range of devices. Currently, only two standard J2ME configurations are available. The first is Connected, Limited Device Configuration (CLDC), which focuses on low-end consumer devices (devices with intermittent network connections, slow processors and limited memory), such as cell phones, two-way pagers and personal organizers. The second one is Connected Device Configuration (CDC), which focuses on high-end consumer devices (devices with greater network bandwidth, faster processors and more memory), such as TV set-top boxes, Internet TVs, and high-end communicators.
- **Profiles:** A J2ME profile is the extension of a configuration and layered on the top of the configuration. Profiles provide a set of higher-level APIs to further define the application life cycle model, the user interface, and access to device specific properties [1]. All of these build a complete runtime environment for

developers to write applications for a specific type of devices.

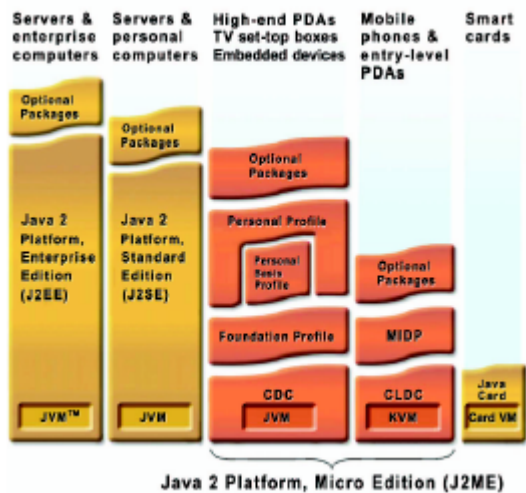


Figure 1 . The family of Java technologies.  
(source: Sun [1])

### 3. CONNECTED LIMITED DEVICE CONFIGURATION (CLDC)

As mentioned above, the Connected, Limited Device Configuration (CLDC) is one of the building blocks of the J2ME. It specifies Java environment by giving a portable, minimum – footprint Java platform for mobile phone, pager and PDA devices, which have the following characters:

- limited user interface (small screen)
- 160 to 512 kilobytes of total memory budget 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, intermittent connection and with limited (often 96000 bps or less) bandwidth. [2]

The CLDC libraries inherited some classes from Java 2 Platform, Standard Edition (J2SE), which are in packages: `java.lang.*`, `java.util.*`, and `java.io.*`. In addition, The CLDC introduced new classes in package: `javax.microedition.*`.

## 4 . MOBILE INFORMATION DEVICE PROFILE (MIDP)

### 4 . 1 Introduction of MIDlet

Based on CLDC, the Mobile Information Device Profile (MIDP) is designed for mobile phones and entry-level PDAs. All applications running in devices with MIDP profile are called MIDlets [3], which are derived from a special class, `MIDlet`. The life cycle of MIDlets is managed by an application environment. So each MIDlet needs to implement some given methods, which make it change its state, as indicated in Figure 2. A MIDlet can exist in three different state: active, paused, and destroyed. When a MIDlet is downloaded to a device, it is installed on the system. If the MIDlet is activated, it starts from the paused state. After `startApp()` method is called, it changes to active state and stay in this state until the `pauseApp()` or `destroyApp()` method is called, then it will be paused or destroyed accordingly. In this context, destroying a MIDlet means that it is removed from the runtime memory (heap). The MIDlet still stays on the device and can be activated again.

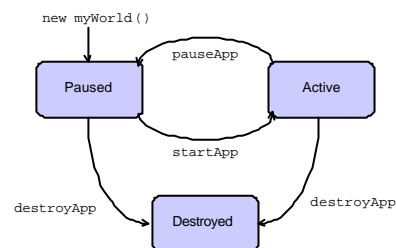


Figure 2. The lifecycle of MIDlet

### 4 . 2 Components of MIDP

The MIDP has the following components: user interface through display and keypad, the networking libraries, the persistent data storage and some advanced system functions, such as timer. MIDP extended the packets of CLDC on the following function: the `java.lang` - state exception for Midlets, `java.util` - advanced timer function for tasks with a given time or a given rate to run repeatedly, as well as `javax.microedition.io`-extending the CLDC

Generic Connection Frameworks (detailed information on GCF will be given later) for communications with HTTP-protocols.

#### 4.2.1 MIDP user interface

The user interface of MIDP is an abstraction of display and keypad of a mobile telephone. These abstract models cover the minimum available functions and must execute on all devices conforming to MIDP. The display of the device is mapped as a permitted and realized component, on which the abstract keypad can be displayed (such as select-key, ok-key etc.). The graphical implementation of the display and the keypad are exhibited as events on the user interface. Thus, to some degree, the device-specific user interface keeps its "Look & Feel".

The user interface of MIDP is covered in the packet javax.microedition.lcdui. The class Display is based on these packets. A display is to present the displayable interface, if there is something to display.

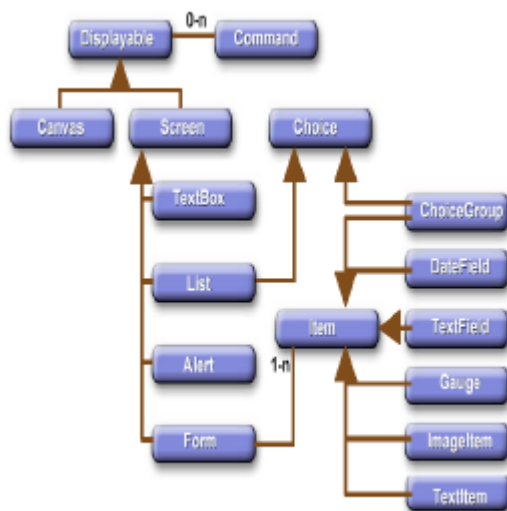


Figure 3. User interface of MIDP (source: Nokia [4])

Shown in the Figure 3, there are basically two types of user interfaces. The low-level objects for canvas include all the applications of graphics and low-level input events; the high-level objects for screen is textual, with a visual display unit. A screen is based on a series of predetermined classes, on which the user interface is relatively easy to implement. Each screen can have the access to the keypad commands.

In a screen, the following elements can exist:

- Alert: display message in text and image for a certain period of time. ALARM, CONFIRMATION, ERROR, INFO or WARNING display a proper textual message to inform users about errors or other exceptional conditions. The type of message is presented as the headline of the screen.
- List: a screen containing a list of choices. Each choice is an access to a string or an icon, which is implemented by Display. Users can scroll the list of choices. The Midlet is indeed informed, as soon as the user chooses one or more elements of the list. There are different types of choicemenu or choicelist, such as the implicit choice, exclusive choice and the multiple choices.

An implicit list displays the list element without checkboxes or radio buttons. The choice is exclusive and a list element is selected as soon as the choice is made. Radio Button is used to display exclusive list. The selection of one Radio Button deselects the other buttons. The selection of a list element is confirmed by the command "OK". The multiple-choice list allows more options. It is also confirmed by the command "OK".

- TextBox: a TextBox allows users to enter and edit text or digits. There are limitations on how many characters altogether can be contained, and some input constrains. It allows different display formats, such as any textformat (ANY), only digit (NUMERIC), password (PASSWORD), telephonenumber (PHONENUMBER), a URL and E-mail address (EMAILADDR).
- Form: A form contains a collection of items. It is composed in a similar way, as a HTML form in the web. The items on a Form may be Strings, Images, editable TextFields, editable DateFields, Gauges, and ChoiceGroups. The title of the form is displayed as the headline of the screen.

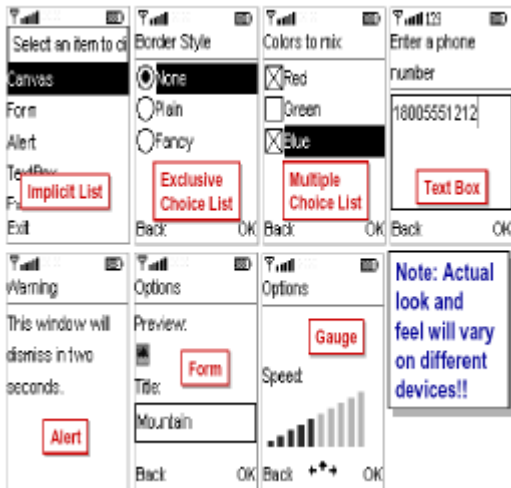


Figure 4 Different user interface of Mobile (source: Sun [1])

The screen is the central abstraction of the MIDP user interface and MIDP works on a class of menu slides. Because there are different kinds of input mechanism for MIDP devices, MIDP defines commands to make the applications unaware of specific devices. Figure 5 displays the mapping of commands and a CommandListener for a screen.

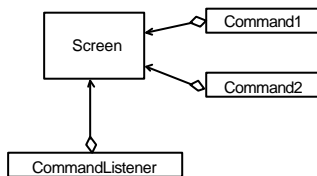


Figure 5 Commands and a CommandListener for a screen

An example of screens with standard components: Alert, List, Textbox and Form are displayed in Figure 6.



Figure 6 Example of the user interface

The picture comes from Wireless Toolkits (WTK), which are sets of tools giving an emulation environment for application programmers, and its target is to design the Java technology applications for the CLDC/MIDP mobile phones and entry - level PDAs. The WTK may be downloaded from <http://java.sun.com/j2me>, the official web site of Sun Microsystems. After downloading the Toolkits and installing them, it can run itself immediately. In order to build an application, a developer activates the Ktoolbar and opens the project “UIDemo”. After the compilation and preverification, it allows the activation (button “Run”) of emulator. Using this emulator, people can emulate different kinds of telephone domainstypes. Figure 7 shows some interfaces of the Toolkit.



Figure 7. Wireless Toolkits Interface (source: Nokia [4])

#### 4.2.2 MIDP Networking libraries

The only well-defined connection in MIDP 1.0 is the connection through HTTP, which is

an extension of the connectivity of CLDC. For this purpose, enhanced MIDP packet `javax.microedition.io` with additional interface `HttpConnection` is used. The methods of these interfaces serve fundamentally the definition and the enquiry of contexts of HTTP-requests. The connection itself is through a URL in the class connector of the CLDC with its compounded method `open`. Basically a HTTP-connection has the following element body:

```
HttpConnection c = (HttpConnection)
Connector. Open (Target_URL);
InputStream is = c.openInputStream ( );
```

The MIDP's support for the HTTP protocol is implemented by the IP protocols, such as TCP/IP, or by non-IP protocols such as WAP or i-Mode. A gateway is used for the Non-IP transport. Figure 8 shows the HTTP network connection.

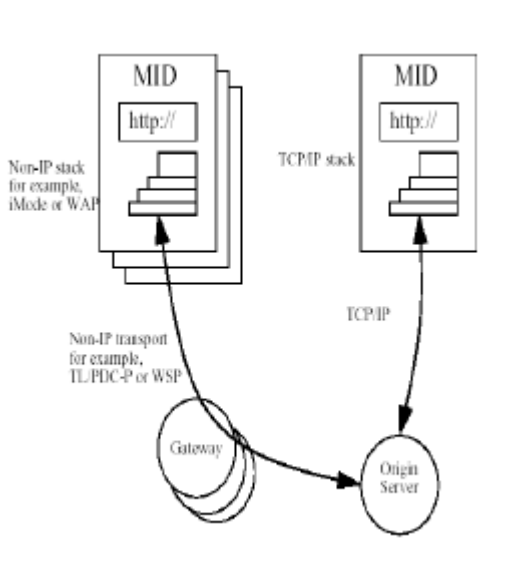


Figure 8. HTTP network connection

J2ME networking was designed for different needs of a large range of devices. However, devices from different companies have different device operating systems and networking programming functions, such as Socket and Http connection, i. e. the networking system is device specific. So an important concept of CLDC Generic Connection Framework (GCF) was introduced. Generic is so called, because CLDC specifies neither certain protocol nor the implementation of certain protocol, only allowing general settings of the connection. Moreover, the CLDC GCF lays a generalization of the

networking and I/O related class for J2ME. A summary of the class hierarchy of CLDC GCF is shown by Figure 9. The MIDP networking library is layed on it.

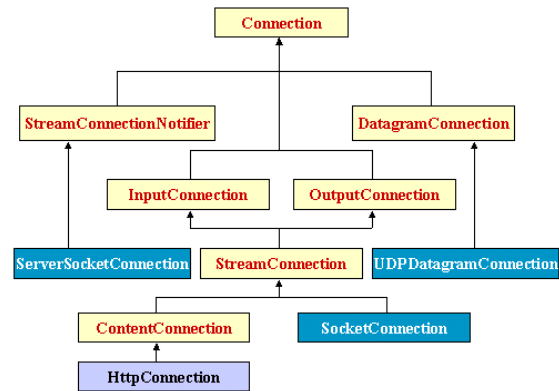


Figure 9 The Interface Hierarchy of the Generic Connection Framework (source: Sun [1])

Based on the CLDC GCF is the interface connection with its class connector from the packet `javax.microedition.io`. The CLDC defines different types of connection. For example, a connection with continuous dataflow (`InputConnection`, `OutputConnection`); a connection through `DatagramConnection`; a connection with constant dataflow in both direction (`StreamConnection`); a connection with a Web-Server (`ContentConnection`), as well as a mechanism notification over a current connection between client and server (`StreamConnectionNotifier`). In the Figure 9 `HttpConnection` interface was added by MIDP 1.0, while `ServerSocketConnection`, `UDPDatagramConnection`, and `SocketConnection` interfaces were added by MIDP 2.0.

#### 4.2.3 MIDP persistent libraries

The MIDP introduced a very elementary concept for data storage, which is record-based. The memory is defined as record store, consisting of persistent data that a Midlet Suite (a collection of Midlets for one application) can store in run-time and retrieve late. Each record is identified by its own `RecordId`. The records may have different length of capacity. New data is written at the end of the data sequence. It is also allowed that new data rewrite existing old ones. Figure 10 shows the

structure of the MIDP record stores. These concepts describe the programming user interface for persistent data in MIDP. To use the records and record stores, a developer needs classes from the package `javax.microedition.rms` of MIDP.

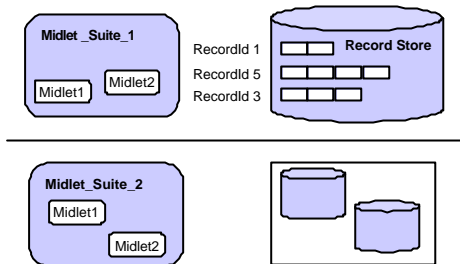


Figure 10 The structure of MIDP record stores

In the MIDP, all the persistent storage mechanism is called Record Management System (RMS). Records are represented by Byte Arrays. RecordIds exist only in its own record stores. Midlets within a Midlet Suite are allowed to produce multiple record stores, given that each one has a unique name. If a Midlet Suite is removed from the platform, all the record stores related to that Midlet Suite must also be removed. Thus, there is no record sharing between Midlets from different Midlet Suites. Midlets in the same Midlet Suite can access their common record stores. However, since there is no locking mechanism for the access in the system, the application program must organize by itself the sharing of common resource. For safety reasons, if one Midlet is accessing record stores, other Midlet should not be allowed to access the same record stores. Record stores are earmarked by names, which is maximum 32 characters long. When working on a record store, a Midlet must use the names to open a record store. After opening it, There may be many kinds of manipulations, such as, to locate numbers and RecordIds, to obtain free RecordIds, to add or deduce Records, to detect the time of last modification and so on. Finally, the process applies closing method to close the record store. In addition, RMS has a Record List Interface to supervise the modification of records, cancel of records and the generation of new records.

#### 4.3 APIs of MIDP 2.0

The MIDP 2.0 is an extension of the MIDP 1.0, so the structure of MIDP 2.0 is fundamentally remains the same, and nothing is removed from MIDP 1.0. Therefore, application programmers need few changes, and the already available Midlets that can run on the MIDP 1.0 can also run on the MIDP 2.0, i.e., the MIDP 2.0 is downwardly compatible. On the other hand, the MIDP 2.0 enhanced the function of MIDP 1.0 on the following fields:

- Enhancement of the user interface: a user-defined class `CustomItem` upgrades the form of class screen. These enhancements of screens and items make it easier and more interactive for users to operate the device. For instance, a new `PopupChoiceGroup` shows a current selection with a visual image; more interactive `StringItem` and `ImageItem`; Gauges can be added to Alert screens.
- Support for game and audio: To facilitate the programming of games, the class `Canvas` is extended for game (`GameCanvas`). The `GameCanvas` support the display with more layers for background and for animated objects. Also the background can be composed of animated blocks (tiles, i.e. text tiles). Such a background is identified as `TiledLayer`. For each loop, the graphic of game can initially create the outlying display on the screen as graphic object, then combine them to complete the display of actual screen. In addition to the support of the `GameCanvas`, it also supports other abstract keys, such as `FIRE`, `UP`, `DOWN`, `LEFT`, `RIGHT`, and other freely configurable buttons. For the audible background music of games, MIDP further contains a subset of Mobile Media API (JSR-135). This API supports tone generation, tone sequence and WAV files (if the device support sampled audios).
- Networking extensions: MIDP 1.0 supports only HTTP connection [5]. In the MIDP 2.0 more connectivity

standards are used. Besides HTTP, there are HTTPS, Sockets, Secure Sockets and UDP datagram. Using HTTPS and secure sockets, it can support secured connection and signed Midlets, which are needed for the reliable establishment of networking connection. For the encoding and signature, the MIDP 2.0 supports public key and personal key.

- Push Registry: A new and important enhancement. With this Push model, MIDlets can be registered and activated later when there are messages received from servers, without self-activation. After getting an entry in the application descriptor file or through PushRegistry class, a MIDlet can register an inbound connection listener. When a message arrives, the system activates the proper Midlet with startApp() method, without the user's intervention. In this way, MIDlets can provide services even when it is not active, and improved the event – driven ability of devices. For example, a MIDlet subscribed to the stock alert service. When the price is equal or higher than the idea price of the customer, the service will send a notification message to the MIDlet, which does not need to keep active all the time. Figure 11 shows the basic principle of the Push Registry.

Figure 11. Push Registry Service (source : Nokia [6])

- Over the air (OTA) provisioning: OTA definition was introduced, in order to establish a concrete standard for the security and interoperability between service provider and the different kinds of Java-enabled devices. OTA provisioning gives users the opportunity to find an application, download it to their own devices and install it. In the MIDP 1.0, the download of Midlets through the air interface is not provided, while it is now a required part in the MIDP 2.0 specification. Thus the MIDP 2.0 requires simplified procedures of downloading and buying Midlets from service provider or from the so-called vending machine.

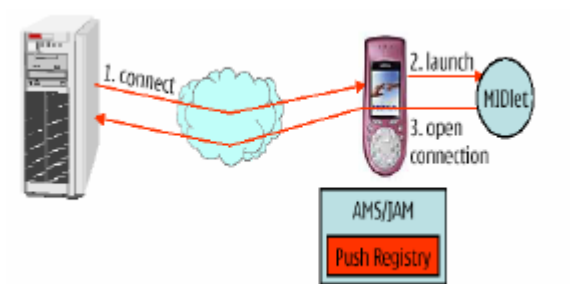


Figure 12 Over the air (OTA) provisioning (source: Nokia [6])

Concerning the above-mentioned enhancement, the demands of MIDP on the hardware are also increased correspondingly. Based on CLDC, MIDP 2.0 now needs at least 256 kbytes memory for the class library, 8 kbytes memory for persistent data (like in version 1.0), as well as 128 kbytes for process memory. In order to play sounds, a simple drive for voice sequence is supported by the MIDP.

Beyond MIDP 2.0, there are also some other application areas as following:

- Further functions of the Mobile Media API (JSR-135) for the replay of video, tone and other medias.



- The wireless message API (JSR-120).
- The Bluetooth API (JSR-182) to communicate with other devices.
- Enhancements of the Mobile Games API to support completed and typical attributes of multi - media, such as sound, graphic and so on.
- A API for location based services, which is the support for the personal searching applications, such as navigation-aid, routes-planner, etc. For example, you can know your location automatically through your mobile, and check the nearest railway station or next train to another city.

#### 4.4 Limitations of MIDP

The MIDP gives programmer a basic environment to design applications for different resource-constrained devices. However, there are also some limitations imposed on MIDlets in comparison to the Java Standard Edition.

1. MIDP runs on top of KVM (Kilobyte Virtual Machine), which is a Java Virtual Machine, and is specifically designed for small and resource-constrained devices. Thus, the capabilities of KVM are limited by the nature of the device, such as limited static and runtime memory. These limitations affect the way programmers approach coding and debugging:

- No floating point support: since the devices have no hardware support for the floating point arithmetic, only fixed-point numbers can be used.
- Limited error handling: there are only three error classes in the CLDC: `java.lang.Error`, `java.lang.OutOfMemoryError`, and `java.lang.VirtualMachineError`. So each device has to handle errors in its own way, which may result in terminating applications or resetting of device.
- No user-defined class loaders: For the sake of security when downloading applications, in CLDC and MIDP, there is a closed sandbox. Applications must run in this closed environment, and can only access libraries that have been defined and supported by the device. No user-defined class loaders

are provided, only system class loader provided by KVM itself can be used.

- No reflection features: reflection features allows a Java applications to inspect the number and contents of classes, objects and some other runtime structures in the virtual machine. No support for the features means a CLDC virtual machine can not use the remote method invocation (RMI), object serialization and other features that are depend on the reflection features.
- No thread groups and daemon threads: KVM can implement multithreading, but has no support for thread groups and daemon threads, i.e. the operations (such as starting and stopping of threads) can only be performed on individual thread. If programmers want to perform thread operations on groups of threads, they have to use explicit collection objects to store the thread objects at the application level.

2. A virtual machine based on CLDC must be able to reject invalid class files. However, the standard class verifier needs much more static and dynamic memory than that of the resource-constrained device. Thus, the concepts of off-device preverification and runtime verification are introduced. Firstly, class files are run through a preverifier to get additional attributes in order to speed up runtime verification. Then, the runtime verifier uses these attributes to perform the actual class file verification efficiently. Only after the class file has passed the runtime verifier successfully, the interpretation can start.

3. Limited functionality of MIDP 1.0 and MIDP 2.0, such as no API for Bluetooth, no access to name directory, and no access to calendar or the to do list, and other information or applications which are contained in the mobile device.

#### 5. JAVA COMMUNITY PROCESS (JCP)

All the APIs and specifications of MIDP come from Java Community Process (JCP), which is an international organization, developing and evolving Java™ technology specification. The JCP was originally created

by Sun Microsystems, and then became a formalized process, including representatives from many organizations and companies all over the world [7]. It helps to make sure the standards of Java technology and its compatibility, enabling the technology to work in different kinds of devices, not only the communication devices, but also industrial product and other consumer devices.

The JCP gives anybody a chance to make his own work an official component of Java platform, also share ideas on how to improve the technology. It is open to everyone, individuals or organization can become a JCP member by signing the Java Specification Agreement (JSPA), which is an agreement between an organization or individual and Sun. There are two Executive Committees (EC) in JCP, and they are responsible for different markets of Java platform (one for the desktop/server space and the other for consumer/embedded space). The duties of EC are: select Java Specification Requests (JSRs) for development, approve draft specification for public review, approve final specification, review Technology Compatibility Kit (TCK) appeals, and so on.

One or more members can initiate a request for a new specification or request the revision of old ones. There are four major steps to follow in JCP:

1. **Initiation:** Community members initiate a specification, which is targeted at desktop/server or consumer/embedded space. Then the responsible Executive Committee (EC) approves the development of the specification.
2. **Community Draft:** In the community, a group of experts are organized to give the first draft of the specification, which will be reviewed by the community and EC. According to their opinions and suggestions, the experts revise the draft. Finally, the EC decides whether this draft can go to next step or not.
3. **Public Draft:** The community draft is pasted on the Internet, and anybody can evaluate it and give their suggestions and comments. Based on these, the expert group further revises the draft. After that, the leader of the expert group makes sure that all the

related references implementation and the needed technology compatibility kit are completed, then send this document to the responsible EC, who will give it a final approval.

4. **Maintenance:** After the completed specification becomes public, there are requests for the interpretation, clarification, and improvement of the specification from community members and public. The components of the specification are updated according to these requests.

## 6. SERVICE APPLICATIONS WITH J2ME MIDP

Wireless Java applications fall into two broad categories:

- **Stand – alone applications:** The APIs has been downloaded and can be operated without access to the networks. Examples are single – player games and some Java services, such as calendar, personal organizers and so on.
- **Network applications:** Some components of the application run on the wireless device, and some run on network. Therefore, these applications need access to the network. Examples are interactive or multi – player games, mobile clients and on line Audios or Videos.

There are many game providers on the Internet, some provide free games, which you can download the free MIDlets and run it on your device, such as Spruce Technology [8]. Others offer charged games, such as Vodafone [9].

Another potentially important area is mobile commerce. People become more and more interested in doing business on the run. For example, MSales application enable a salesperson to have easy access to the sales data base including production information, price list, customer purchase histories and so on, which helps her schedule her visits and take orders [10]. Other examples are wireless banking, on line shopping and so on. The big challenge of mobile commerce is the security, for most enterprise applications are driven by the identity of the person using the system. “Java technology continues to be the leading

platform for advanced mobile devices, ” said Alan Brenner, vice president, Consumer and Mobile System Group, Sun Microsystems. “Updating MIDP and adding new media and messaging capabilities to J2ME will help to ensure that the Java platform continues to provide the most robust environment for delivering new applications and services to mobile devices.” [11].

## 7. CONCLUSIONS

With the development of the wireless data transmission and the popularity of it, the applications based on the J2ME become more and more attractive, and give service providers more opportunities in the market. Nowadays, the devices are available, and everything needed to write Java applications for mobile devices is available. Through the Java-enabled mobile, PDA and other terminals, service providers can offer safe, interactive, and more plentiful applications, such as messages, games, socket services, mobile commerce, mobile office and so on, which let users freely enjoy the internet service anywhere and anytime. Though MIDP 2.0 improved the MIDP 1.0 in many fields, there are still some limitations, for example, without floating point capabilities, the MIDP 2.0 is useless for applications such as popular navigation services. Another problem is the trade – off between security and flexibility, i.e. high security limits the flexibility of applications. Therefore, updating the MIDP and adding new MIDlets are very important to ensure that J2ME can provide robust runtime environment for applications for mobile devices. When the J2ME MIDP technology is becoming more and more mature, the existing applications will be further improved and more new applications will be created.

## REFERENCES

- [1] Introduction to J2Me Development, <http://java.sun.com>
- [2] Roger Riggs, Antero Taivalsaari, Mark VandenBrink: Programming Wireless Devices with the Java™ 2 Platform, Micro Edition, ADDISON-WESLEY, USA 2001

- [3] Sun Microsystems, Inc. MIDP APIs for Wireless Applications, [www.sun.com](http://www.sun.com)
- [4] Nokia official web site, <http://nokia.com>
- [5] Qusay H. Mahmoud. Future Java Technology for the Wireless Services Industry, February 2003 <http://www.wireless.java.sun.com>
- [6] Introduction to MIDP2.0, <http://www.forum.nokia.com>
- [7] Java Community Process on Standardisation and Technology, <http://www.jcp.org>
- [8] Free game provider, <http://www.spruce.jp>
- [9] Charged game provider, <http://www.vodafone.co.uk>
- [10] Imtiyaz Haque and Brian O'Connor, J2ME Enterprise Development, April, 2002, <http://www.ashnasoft.com>
- [11] Java(TM) Technology Enhances Support for Messaging, Graphics, Video and Audio To Mobile Devices, 2002, <http://id.sun.com>