Review of J2ME and J2ME-based Mobile Applications

Anna Isakow and **Hao Shi**

School of Computer Science and Mathematics
Victoria University, Melbourne, Australia

Summary
The wireless revolution has transformed mobile communication from only voice-oriented communication with relatively static, hard-coded functionality into extensible, Internet-enabled communication with advanced data and software support. Nowadays, users expect wireless connections to be able to transfer live video and high-quality audio, as well as download significantly larger applications and services comparing to those in the past. There is a great demand for multi-functional mobile devices capable of hosting a broad range of applications for both business and consumer use.
Java 2 Platform, Micro Edition (J2ME) offers the wireless community a standard solution on different platforms without significant changes to the system. Almost all mobile phones available on the market support the programming language Java for J2ME. In this paper, Java 2 platform and J2ME for development of mobile applications are reviewed. Current applications for mobile devices using J2ME are present and possible future mobile applications are addressed for developer to take advantages of J2ME for future mobile applications.

Key words:
J2ME, mobile applications, wireless communication.

1. Introduction

The wireless communication industry has seen explosive growth over the last decade. This made wireless communication one of the fastest growing technology areas in the world. At the same time, the rapid emergence of the Internet has changed the landscape of modern Information Technology (IT). People have become more and more dependent on the information available on the Internet and they increasingly want to get access to the Internet services, not only from their home and office computers but also from mobile, wireless devices. As a consequence of that development, the rapid and efficient deployment of new wireless data and mobile Internet services has become a high priority for communication equipment manufacturers and telecommunication providers.

The wireless revolution has transformed wireless devices from only voice-oriented communication devices with relatively static, hard-coded functionality into extensible, Internet-enabled devices with advanced data and software support. These devices need to support dynamic downloading of new software, and need to be capable of running software developed not only by the device manufacturers themselves, but also written by other software developers. This makes the devices much more dependent on software and requires handling of such topics like software interoperability, security and reliability. Java is the programming language which ideally suites to develop software for mobile devices and provides many benefits.

As Java is a modern object-oriented language and has far better features and higher-level programming constructs than other languages and tools that are used for wireless software development, it allows software to be developed more efficiently. Nowadays, almost all mobile phones available on the market support the programming language Java for Java 2 Micro Edition (J2ME). J2ME allows developers to implement platform independent applications for mobile devices. Java has become the major object-oriented programming language for developers to implement new mobile applications, which benefit from Java's well-known features for design of graphical interfaces.

2. Overview of Java 2 Platform

Java platform consists of the Java language, Java Virtual Machine (JVM), and Java Application Programming Interfaces (APIs). The Java platform is designed to cover a wide range of computer hardware, everything from smart cards through enterprise servers. Therefore, Sun Microsystems has grouped Java technologies into three editions. Each of them aims at a specific area of today's huge computing industry:

- Java 2 Platform, Enterprise Edition (J2EE) is for companies which need to provide server solutions to their customers, suppliers and employees. J2EE is usually used for servers and enterprise computers. It is based on J2SE and adds APIs for server-side computing.
- Java 2 Platform, Standard Edition (J2SE) is designed for the desktop and personal computer

Manuscript received February 5, 2008
Manuscript revised February 20, 2008

market. Most often it runs on top of OS X, Linux, Solaris or Microsoft Windows (Sun Developer Network (SDN) n.d.b).

- Java 2 Platform, Micro Edition (J2ME) is a set of runtime environments and APIs developed for small devices like PDAs, TV set-top boxes, mobile phones and other devices. Those devices cannot support a full J2SE or J2EE implementation because they lack necessary resources. J2ME uses subsets of J2SE components, such as smaller virtual machines and leaner APIs.

J2ME was designed for needs of:

- Consumers and embedded device manufacturers who build many different information devices,
- Service providers who use those devices to deliver content to their customers and
- Content creators who want to make compelling content for small devices with stringent restrictions on computational power, battery life, memory and network bandwidth.

The development of specifications for J2SE, J2EE and J2ME is controlled by the Java Community Process (JCP). JCP is responsible for the development of Java technology and approval of Java technical specifications. A specification begins its life as a Java Specification Request (JSR). JSRs are formal documents that describe proposed specifications and technologies to be added to the Java platform. Before the JSR becomes final and is voted on by the JCP Executive Committee, formal public reviews of JSRs have to be conducted. The specification would be created by an expert group consisting of representatives from interested companies. Before JSR is finished and can be used, it has to pass through different stages of the JCP. Every J2ME specification has an unique JSR number and can be commonly referred to by that number.

As Fig. 1 illustrates, there are different Java 2 Platform editions. Each of them defines its own main target market. As already mentioned above, J2EE is used for servers and enterprise computers. J2SE supports desktop and personal computers. J2EE and J2SE are completely separate from J2ME family tree. At a high level J2ME is targeted at two broad categories of products: the high-end and low-end consumer devices.

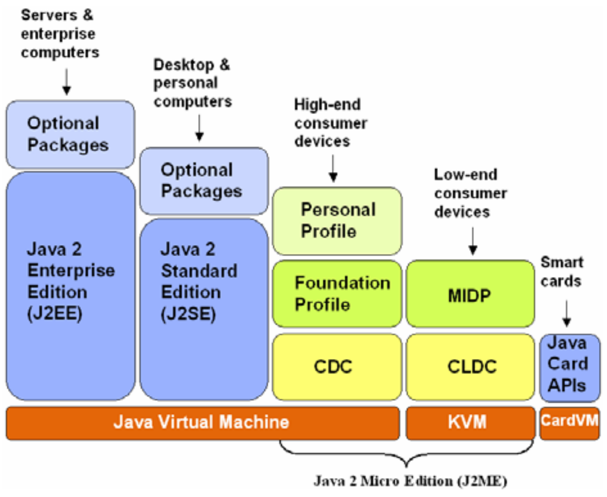

Servers & enterprise computers
Desktop & personal computers
High-end consumer devices
Low-end consumer devices
Smart cards
Optional Packages
Optional Packages
Personal Profile
Java 2 Enterprise Edition (J2EE)
Java 2 Standard Edition (J2SE)
Foundation Profile
MIDP
Java Card APIs
CDC
CLDC
Java Virtual Machine
KVM
CardVM
Java 2 Micro Edition (J2ME)

Fig. 1: Java 2 Platform editions and their target markets

High-end consumer devices are represented by Connected Device Configuration (CDC). Such devices like TV set-top boxes, Internet TVs, Internet-enabled screen phones, high-end PDAs like the Palm Z 22, high-end wireless communicators and navigation systems are good examples of CDC-based devices [1].

Connected Limited Device Configuration (CLDC) represents the low-end consumer devices. Mobile phones, pagers and personal organizers are examples of appliances in this category. Practically, the line between the two categories of J2ME is defined more by the memory budget, bandwidth considerations, battery power consumption and physical screen size of the device rather than by its specific functionality or type of connectivity. This division in two different groups has become more difficult as a result of continuing technological development. Finally, Java Card technology supports development on Java-based smart cards (Ortiz 2004a). Java Card technology complements J2ME. It provides smart cards with a compact Java runtime environment, virtual machine and programming interface. As Java card technology, J2EE and J2SE are beyond the scope of this paper, no further explanations will be made.

3. Java 2 Micro Edition (J2ME)

J2ME (Java 2 platform, Micro Edition) was announced by Sun Microsystems at the JavaOne Conference in June 1999. The purpose of J2ME is to enable Java systems to run on small computing devices. It does not define a new programming language but simply applies existing Java technology to handheld and embedded devices. Unlike J2SE or J2EE, J2ME is neither a piece of software, nor is it a single specification. Instead, J2ME is a platform, a collection of technologies and specifications that are designed for different parts of the small device market. It provides a robust, flexible environment for applications running on mobile and other embedded devices—mobile phones, personal digital assistants (PDAs), TV set-top boxes and printers. It is divided into different and a steadily growing variety of configurations, profiles and optional packages. In the following subsections, each of these key terms will be covered in depth.

3.1 Architecture of J2ME

J2ME has become an organized architecture for electronic devices. It includes sets of Java APIs for high-end PDAs and embedded devices and for more constrained devices such as mobile phones, low-end PDAs and headless devices, those without display or user interface facilities. The JCP has defined, not only the many components of J2ME, but the platform's overall architecture, in the J2ME Platform Specification. Fig. 2 presents the different software bundles of J2ME.

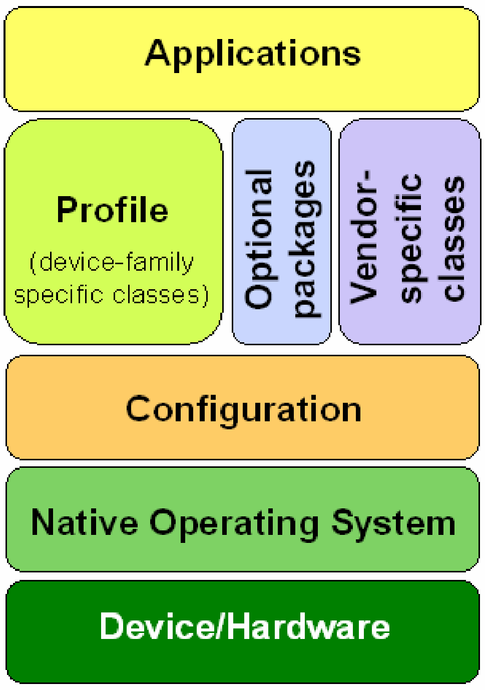


Fig. 2: J2ME software layers

At the heart of J2ME are three core concepts: configurations, profiles and optional packages. Configuration such as CLDC provides core services for a broad category of devices. CLDC was designed for devices which have memory and processor power constraints. A configuration defines the Java language and virtual machine features and minimum class libraries that a device of the same category should have. It specifies a JVM that can be easily ported to devices supporting the configuration. A configuration also defines a minimum platform for a horizontal category of devices, each with similar requirements on total memory budget and processing power.

A profile like Mobile Information Device Profile (MIDP) supports higher-level services common to a more specific class of appliances. It means that profiles are more specific than configurations and consist of class libraries which are more domain-specific than the class libraries provided in a configuration. A profile is based on the top of a configuration and adds APIs for user interface, persistent storage and other classes needed to develop running programs [2]. A profile is for specific needs of a vertical market segment or device family. The main goal of a profile is to guarantee the interoperability within a certain device family or domain by defining a standard Java platform for that market. Each family of devices has its own profile that represents a particular market within a given configuration. For example, the profile for the cell phone vertical market is separate from the profile for the PDA vertical market, but both profiles are based on the same mobile device configuration. One device can support multiple profiles.

Optional packages add special services that are useful on devices of many kinds, but that are not necessarily available on all of them. Usually, applications are based on a configuration appropriate to the desired category of target devices and on a profile that supports the software’s basic functionality and optional packages which support needed specialized functions like messaging or multimedia [3]. A profile is based on top of a configuration and adds APIs for user interface, persistent storage and other classes needed to develop running programs. One device can support multiple profiles.

All that is implemented on a device like configuration, profile and optional APIs is called a stack (see Fig. 3). For instance, a device stack could be CLDC/MIDP + Mobile Media API. This organization supports both reuse and efficiency and enables developers to put together a software stack that fits both the capabilities of target devices and the resource needs of applications.

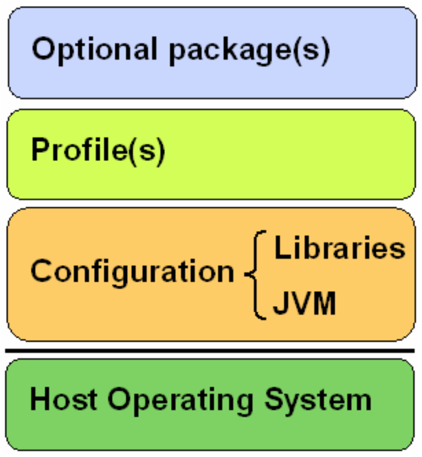


Fig. 3: Organization of the J2ME

In addition to J2ME's configurations, profiles and optional packages as described above, Fig. 2 shows that applications may include vendor-specific classes when required. Vendor-specific APIs are extensions to configurations and profiles. The standard J2ME does not consist of vendor-specific interfaces, but they can be added and extend the functionality that's specific to a given device, for example APIs to control the radio transceivers of a device [2].

3.2 Configurations

J2ME configurations define the basic functionality of the Java platform for a broad family of devices with generally similar capabilities. Sun Developer Network (SDN) defines configurations as "… specifications that detail a virtual machine and a base set of APIs that can be used with a certain class of device." A configuration defines a basic J2ME runtime environment which many devices can implement without changes. This includes the virtual machine and a set of core classes derived primarily from J2SE. Thus, a configuration is a complete Java runtime environment, consisting of three things [4]:

- A JVM to execute Java bytecode
- Native code as an interface to the underlying system
- A set of core Java runtime classes

The configuration's formal specification defines the minimum requirements which a device must meet to be able to use this configuration. Although a configuration does provide a complete Java environment, the set of core classes is normally quite small and must be enhanced with additional classes supplied by J2ME profiles or by configuration implementer. In particular, configurations do not define any user interface classes.

Such a configuration might be for devices with less than 512 KB of memory and an intermittent network connection. So far, two configurations have been defined: the Connected Device Configuration (CDC) for programming larger, more powerful handheld devices like PDAs and the Connected Limited Device Configuration (CLDC) for the smallest devices that have more limited resources, like mobile phones [2]. The devices which use the configurations must have some type of network connectivity either it is a high-speed fixed link or a slow-speed wireless link.

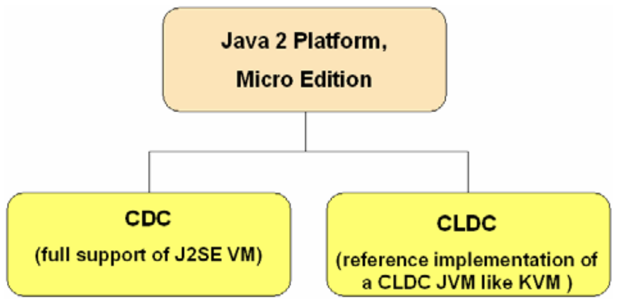

Java 2 Platform, Micro Edition
CDC
(full support of J2SE VM)
CLDC
(reference implementation of a CLDC JVM like KVM)

Fig. 4: Two different configurations of J2ME

The majority of functionality in CLDC and CDC has been inherited from J2SE. Each class inherited from the J2SE environment must be precisely the same or a subset of the corresponding class in the J2SE environment. Additionally, CLDC and CDC may implement a number of functions, not taken from the J2SE, designed specifically to fit the needs of small portable wireless devices. The relationship between J2ME configurations and J2SE is presented in Fig. 5.

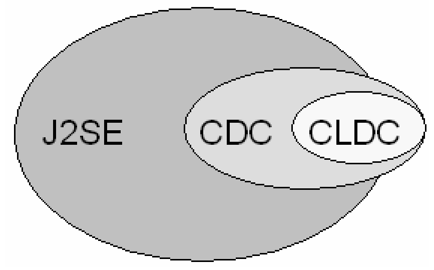


Fig. 5: Relationship between J2ME configurations and J2SE

A particular device might meet the basic hardware requirements of either CDC or CLDC. It can support just one of them because the devices J2ME is aimed at are constrained devices. The device manufacturers who most likely provide Java runtime environments must decide which configuration to support [5].

Core Java libraries are normally intimately tied up with the implementation of a Java virtual machine. This is the most important reason for the configuration layer in the J2ME environment. If small differences in the specification of a configuration are made, it can require a number of significant changes to the internal design of a Java virtual machine. Thus, a substantial amount of additional memory would be required and it could become very expensive and time-consuming to maintain such modifications. The smaller the number of configurations is, the better the few virtual machine implementations can serve the needs of both a large number of different profiles and a large number of different hardware types of devices. Consequently, J2ME environment is absolutely essential for devices which have to be successful and cost-effective in the consumer and embedded industry.

As CLDC and CDC have been developed for usage on different platforms, there are many differences between them. Firstly, CLDC is targeted towards small devices with limited resources like mobile phones. CDC targets a larger device with more capabilities like newer smartphones and PDAs. Secondly, many CDC applications are written like J2SE systems but with a subset of APIs available in J2SE. As Figure 5 illustrates, CDC includes the entire CLDC and contains more J2SE classes in the core runtime library than CLDC. The CLDC supports only a minimal set of classes.

3.3 J2ME Profiles

Although, the configurations provide the foundation for Java programming on small computing devices, they do not provide enough functionality for the development of software. A profile extends a configuration, adding domain-specific classes to the core set of classes. It means that profiles provide classes for specific uses of devices and define functionality missing from the base configuration functions like user interface classes, persistence mechanisms, control of application life cycle, network connections. While they provide important and necessary functionality, not every device will support every profile. In Japan, for example, NTT DoCoMo has released a number of Java-enabled mobile phones based on the CLDC but with their own proprietary profile [6]. Applications written for these devices will not work on mobile devices that support different profiles like Mobile Information Device Profile (MIDP). Profiles are specific to the size of the device (amount of memory) on which a system runs and are associated with certain configurations.

3.3.1 The purpose of a profile

While a configuration defines a minimum level of Java support across a family of devices, a profile defines the application programming interfaces (APIs) for devices with similar uses. A device can use only one configuration but many different profiles can be based on the same configuration. Applications are built on top of profiles. Fig. 6 illustrates the architecture of configurations and profiles.

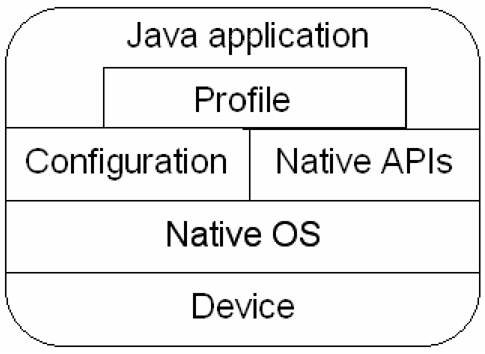


Fig. 6: Profile architecture

Applications can access three different sets of APIs: the API of the configuration, those of the profile or several profiles and native APIs. Native APIs depend on the device and are provided by the implementer of the runtime environment, which is typically the manufacturer of the device.

3.3.2 Types of profiles

As already mentioned, configurations do not provide any classes for managing the application life cycle, for driving the user interface, for maintaining and updating persistent data locally on the device or for accessing information that is stored on a network server [2]. Therefore, that type of functionality is provided by profiles or optional packages. A profile adds domain specific classes to the core set of classes provided by the configuration. Those classes are for specific uses of devices and provide functionality which is not defined by the underlying configuration. There are many different kinds of profiles, which have been defined or are still in development through the Java Community Process (JCP). The following profiles are available:

- The Mobile Information Device Profile (MIDP) which is based on the CLDC and provides a standard Java runtime environment for today's most popular mobile information devices, such as mobile phones and mainstream PDAs.

- The Information Module Profile (IMP) is also based on CLDC and is a subset of MIDP.
- The Foundation Profile (FP) that adds additional J2SE classes to the CDC but no user interface classes. It acts as a foundation for building other profiles.
- The Personal Profile (PP) is based on CDC and provides the J2ME specification for devices that need a high degree of Internet connectivity and web fidelity.
- The Personal Basis Profile (PBP) supports graphics, images and widgets on devices that require a simple user interface, such as automotive devices, consumer devices and simple appliances [2]. PBP is also located on top of CDC.

Fig. 7 illustrates how profiles relate to their underlying configurations and each other.

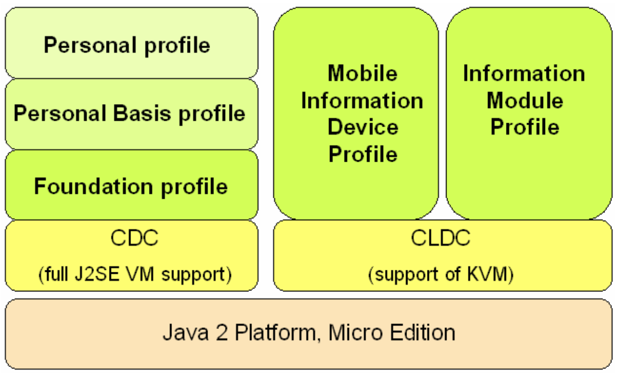


Fig. 7: J2ME Profiles

There are two standard CLDC-based profiles like MIDP and IMP and three CDC-based profiles such as FP, PBP and PP. In order to be able to develop software, the configurations, profiles and perhaps device-specific classes are required. User interface classes are a good example for device-specific classes. User interface classes are not included in the configuration because not every device supports a user interface. Multiple profiles can exist on top of the same configuration, as in case of MIDP and IMP. They can also depend or build on each other like the profiles of CDC. The PP extends the PBP, which in turn is on top of and depends on the FP. Fig. 8 illustrates the relationship between IMP and MIDP.

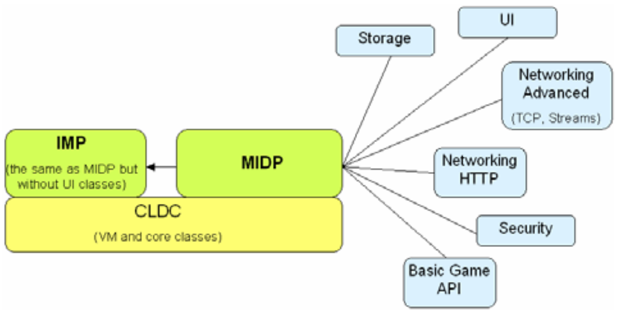


Fig. 8: IMP and MIDP relationship

3.3.3 Optional Packages

Optional packages are important components of the J2ME architecture. They extend profiles in their functionality. As they provide support in relatively narrow areas of functionality some devices and applications need them but other don't, such as messaging, multimedia and location services. With optional packages taking over such burdens, profiles can concentrate on supporting only those capabilities that most or all devices in a category need. They can supply the runtime environment, while optional packages supply specific kinds of functionality. This approach allows to include capabilities that were not even envisioned when the profiles were designed. An optional package provides functionality that may not be associated with a specific configuration or profile. One example of an optional package is the Bluetooth API, which provides a standardized API for using Bluetooth networking. Bluetooth API could be implemented on top of any combination of configurations and profiles.

An optional package is also a set of APIs, but unlike a profile, it does not define a complete application environment. An optional package is always used in conjunction with a configuration or a profile. It extends the runtime environment to support device capabilities that are not universal enough to be defined as part of a profile or that need to be shared by different profiles like the Wireless Messaging API (WMA). WMA is a set of classes for sending and receiving Short Message Service (SMS) messages. Because the WMA is an optional package, it can be included on any J2ME device with SMS capabilities, not just MIDP enabled mobile phones. If

WMA were part of a specific profile, such as MIDP, its use would have been limited to that profile and its supersets [7].

All J2ME optional packages are defined by the JCP, making them standard APIs. As the term implies, inclusion of these packages is optional. Handset manufacturers may choose to include them in a particular product and in extensible environments such as PDAs developers may elect to include them. For example, a particular MIDP handset could add support for Bluetooth connectivity by including the Java APIs for Bluetooth and support for enterprise integration by including the Web Services API.

Optional packages depend on what kind of functionality is necessary. Some of them are specialized in CLDC environments, some in CDC, some in both. There is an increasing number of optional packages:

- Java APIs for Bluetooth (JABWT)
- Wireless Messaging API (WMA)
- Mobile Media API (MMAPI)
- Web Services API for J2ME (WSA)
- Location API for J2ME
- JDBC Optional Package for CDC/Foundation Profile

The complete list of optional packages is available at http://developers.sun.com/mobility/apis/.

4. J2ME-based mobile applications

Nowadays, everybody uses the Short Messaging Service (SMS) to communicate via messages with other telephone subscribers. It is the cheapest, quickest and easiest way to use mobile communication. According to Informa’s World 66 Cellular Data Metrics, worldwide SMS traffic in the first quarter 2007 was up year-on-year growth by around 50 % to more than 620 billion messages. Global mobile data revenues from services other than SMS exceeded US$10 billion in the first quarter 2007. The total of US$11.3 billion compares with US$8.1 billion in the first quarter 2006 [8]. It means that nearly one third of mobile data revenues now come from non-SMS services, suggesting operators to make investments in advanced technologies like mobile payments, cameras, MP3-Players, navigational utilities and games.

-

Fig. 9: Currently very popular mobile phone Sony Ericsson W660

Devices combining these features are called smartphones. As all latest mobile phones offer at least several of functions named above, smartphones are often equated with mobile phones (see Fig. 9). In this paper, both terms are also used synonymously.

4.1 Current J2ME applications

Current mobile phones dispose of a variety of platform dependent software for example organizer, games and address book. Through the invention of programming language Java for mobile phones, it additionally became possible to implement platform independent systems. Nowadays, support of J2ME is a permanent feature of new mobile phones. The majority of J2ME programs are games like puzzles, flight emulators and sport games. Some of the games are already installed on the mobile phone when customers buy them. Games which are not included in the purchase pack, can be directly downloaded from the Internet and installed. Such games enjoy great popularity especially among young people.

4.2 Future J2ME systems

Mobile phone industry is one of the most dynamic areas. In no other economic sector new products come more quickly on the market having such short innovation cycles like mobile devices. By the end of 2006, there were a total of nearly four billion mobile and fixed line subscribers and over one billion Internet users worldwide. This included 1.27 billion fixed line subscribers and 2.68 billion mobile subscribers [9]. The number of mobile subscribers has exceeded the number of fixed line subscribers for many years. Entirely new and interesting systems can be developed using J2ME, especially location-based applications. The economic potential of this kind of software cannot be exactly estimated but the number of software based on GPS (Global Position System) and position determination increases continually.

4.2.1 Navigation

Based on GPS information, different services can be developed which should show the current user position on the map and navigate to a certain place. These services would help tourists to orient themselves in a foreign city. Useful applications would be indication of the next free parking lot or nearest cash machine, the position of hotels or restaurants next to the user location like shown in Fig. 10.

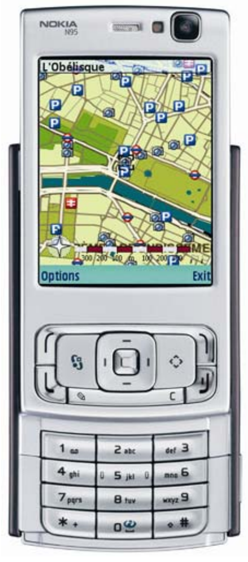


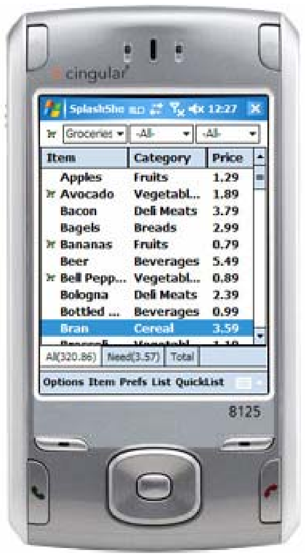
Fig. 10: Navigation software installed on a NOKIA N95

A more complicated service is an early-warning-system for traffic congestion. User would register on a remote server. A GPS receiver could determine the current driver location and the data could be transmitted to the server. The server should recognize the traffic jam on the way to the specified destination, address a warning to the driver and send a suggestion for a new route.

While currently introduced navigation systems are only applicable outside of buildings, it would be also useful to be able to navigate within buildings. Many applications require indoor positioning capabilities such as emergency calls, indoor routing, product tracking or location sensing billing. Indoor Locating System software could help people in orientation in large buildings like museums, galleries, supermarkets, libraries and conferences. If a customer cannot find a certain product in a supermarket, the mobile phone could guide him to the very shelf. A fairgoer would find the necessary booth using a map shown on his mobile device or guided by voice command on his phone.

4.2.2 Shopping

An important area of J2ME applications is also the mobile shopping. The mobile phone could display its owner information about special discounts at supermarkets of the customer's choice. It could also locate the nearest cheap gas station. Another useful service for countries like Germany where shops are not opened around the clock could help in finding an emergency-pharmacy opened at night hours and on the weekends.



Item	Category	Price
Apples	Fruits	1.29
Avocado	Vegetabl...	1.89
Bacon	Deli Meats	3.79
Bagels	Breads	2.99
Bananas	Fruits	0.79
Beer	Beverages	5.49
Bell Pepp...	Vegetabl...	0.89
Bologna	Deli Meats	2.39
Bottled ...	Beverages	0.99
Bran	Cereal	3.59

Fig. 11: Discounts at a supermarket

Mobile advertisements and mobile shopping are related to each other. One possibility would be to display customers current discounts on their mobile phones when they enter a supermarket. Fig. 11 illustrates the prototype for this kind of future J2ME applications.

Another useful functionality is to customize advertising depending on consumer behaviour. A consumer in a bookstore might be able to compare the cost of buying a book at the store with the costs of ordering it online and receiving it in a few days or in an hour.

4.2.3 Entertainment

In our society, entertainment and fun are rated high. There are probably just a few cases when a mobile phone is essential for survival for example for a sick person.

Nevertheless, not many people would want to do without it. Mobile devices make the communication between people more spontaneous and simpler. These features are exactly what customers expect from mobile services. Therefore, it is not remarkable that the majority of J2ME applications have been designed for entertainment for example the Phantom Solitaire game presented in Fig. 12.

Fig. 12: Phantom Solitaire game on the display of a mobile phone

On the one hand, J2ME makes it possible to design and implement simple services like call up movie reviews, check information on events or display opening hours of a restaurant. On the other hand, more complex systems can be developed, e.g. interactive city guides or balance the checking accounts. Although, many J2ME developers have tried to realize this idea, the software they implemented is not platform independent.

Another example for entertainment software is a dating service. In this way, couples could receive a message as soon as they are near each other. The system Buddy Finder should work in similar way. It should display the location of registered friends and make it possible to guide to or invite each other to a spontaneous meeting.

4.2.4 Safety and emergency

Many people see the mobile phone as a perfect companion in emergency cases like car accidents when quick help is required. If the exact location can be determined via GPS or other technologies, ambulance would know the position of injured people and apply first aid more quickly. In this context, an important application would also be for car drivers. It should give the alarm to the nearest hospital, if the car is involved in an accident for example when airbags have been released and the driver does not respond to the request of the system. In the same way, a stolen car could be localized and returned to the owner without much investigating effort of the police.

Furthermore, there is still need for applications which could help in handling with results of natural disasters. Team leaders of rescuers could be informed about the position of action forces and could coordinate it in a better way. Mobile phone of a rescuer could show on its display the position of the next electricity or water connection. However, usage of GPS services requires the intact mobile phone infrastructure. If mobile transmission system does not work, a sophisticated system will not be much help.

The research of current and future J2ME applications has clearly shown that mobile systems based on location arouse a great deal of interest of mobile subscribers. Nevertheless, up to date, there are only a few Location Based Services available on the market. In 2006, the first mobile phones with an integrated GPS receiver came into the market. Conditioned by the availability of this technology on mobile phones, it is expectable that indoor and outdoor Location Based Applications will become more popular in the near future.

5. Conclusion and Future Work

In this paper, Java 2 platform is introduced and Java 2 Platform, Micro Edition (J2ME) for developing mobile applications is extensively reviewed. It aims at developers to take full advantage of J2ME in their mobile applications. J2ME supports persistent storage of data which is enough to save configuration data constantly for current and future mobile applications. J2ME consists of a variety of optional packages which provide special APIs. Those APIs can support the future of mobile applications development. Due to maturity of GPS (Global Position System), location-based mobile applications will become more important in the coming years. The project priority should be given to location-based mobile applications which can display various maps such as road maps and weather maps. It is expected that J2ME and its offspring will offer the most ubiquitous solution for future mobile applications with the continuous improvement in mobile devices.

References

[1] Riggs, Taivalsaari, VandenBrink, 2001, Programming Wireless Devices with the Java 2 Platform, Micro Edition, Addison-Wesley Pearson Education, New York, US.
[2] Ortiz, 2004a, A Survey of J2ME Today, Sun Developer Network (SDN), viewed 13 August 2007, http://developers.sun.com/mobility/getstart/articles/survey/.
[3] Ortiz, 2006, Summary of CLDC-Based Profiles, Sun Developer Network (SDN), viewed 23 August 2007, http://developers.sun.com/ mobility/midp/ttips/cldc/.
[4] Giguere, 2002a, J2ME Core Concepts, viewed 21 August 2007, http://www.ericgiguere.com/articles/j2me-core-concepts.html.
[5] Giguere, 2000, Java 2 Micro Edition: Professional Developer's Guide, Electronic Resource, viewed 20 August 2007, http://library.vu.edu.au.

[6] Giguere, 2001, Java 2 Micro Edition Basics, viewed 22 August 2007, http://www.ericgiguere.com/books/midp/midp-chapter-1.pdf.
[7] Giguere, 2002c, J2ME Optional Packages, Sun Developer Network (SDN), viewed 27 August 2007, http://developers.sun.com/mobility/midp/articles/optional/.
[8] Grenville, 2007, Stats & Research: SMS Traffic Up 50%, viewed 17 October 2007, http://www.160characters.org/news.php?action=view&nid=2325.
[9] ITU, 2007, Next-generation networks set to transform communications, International Telecommunication Union, viewed 18 October 2007, http://www.izmf.de/download/Studien/ITU.pdf.

Ms. Anna Isakow is a final year Business Computer Science student at University of Applied Sciences Regensburg in Regensburg, Germany. She was supervised by Dr. Hao Shi as an international exchange student at Victoria University from August 2007 to November 2007 and successfully completed her minor thesis with topic “J2ME – specification, development and operation using the example of smart devices.”

Dr. Hao Shi is an Associate Professor in School of Computer Science and Mathematics at Victoria University, Australia. She completed her PhD in the area of Computer Engineering at University of Wollongong in 1992 and obtained her Bachelor of Engineering degree at Shanghai Jiao Tong University, China. She has been actively engaged in R&D and external consultancy activities. Her research interests include p2p Network, Location-Based Services, Web Services, Computer/Robotics Vision, Visual Communications, Internet and Multimedia Technologies.