Java 플랫폼의 EmbeddedJava 기술†

김달중* · 하수철*

1. 서 론

무선 인터넷 기술의 발전은 모바일 산업과 접목하여 혁명을 불러일으키고 있다. 이동통신업자 및 ISP 사업자를 중심으로 하는 무선 인터넷 서비스의 컨텐츠들도 다양하게 제공되고 있으며, 이러한 발전은 사용자들로 하여금 정보가전 제품에 인터넷을 적용할 수 있는 기반을 제공하는 계기가 되고 있다. 인터넷 정보가전 산업은 인터넷 인구의 폭발적인 증가와 함께 인터넷을 기반으로 하는 전자상거래, 원격교육, 홈쇼핑 등 다양한 서비스와 함께 경제의 중심이 디지털과 인터넷으로 이동되고 있는 현 시점에서 새로운 산업의 장르로 강조되고 있다[1].

이러한 정보가전 제품들에 사용되는 내장형 장치들은 일상적으로 사용되는 퍼스널 컴퓨터와는 달리 제한된 환경을 가지고 있다.

본 논문에서는 이러한 내장형 환경에 적합한 Java 플랫폼에 대한 기술 동향을 논한다.

이를 위해 먼저 EmbeddedJava 플랫폼 기술을 살펴보고, EmbeddedJava 명세에 관하여 기술하며, 이와 관련된 기술 추이를 살펴보고, Embedded Java를 이용한 어플리케이션 개발 프로세스를 기술한다. 끝으로 EmbeddedJava 사용에 따른 문제점을 살펴보고 결론을 맺는다.

† 본 논문은 2002년도 산 · 학 · 연 공동기술개발 컨소시엄 사업의 지원을 받아 수행되었음.
* 대전대학교 컴퓨터공학과

2. 내장형 Java 플랫폼 기술

Sun사에서 개발한 EmbeddedJava™은 내장형 장치를 위한 소프트웨어 개발을 규격화하는 방법이 있으며 EmbeddedJava 어플리케이션 환경 통합에 의한 개발비용의 절감과 시장에 대한 대응을 향상시키는데 목적이 있다. 또한 EmbeddedJava 툴은 적은 메모리를 사용하여 Java™ 어플리케이션 환경을 최소화시킬 수 있으며 표준화된 개발 툴과 인프라가 구축된 개발자들에게 경제성을 제공한다[2,3].

EmbeddedJava™의 Java 기술은 모바일 폰, 패이저, PDA, 셋탑박스, 프로세스 컨트롤러, 사무용 프린터, 네트워크 라우터나 스위치 등에서 응용되고 있다. 게다가 RTOS(Real Time Operating System) 위에서 사용되며 Java가 가지고 있는 다양한 이점을 그대로 포함하고 있다. 예를 들어 플랫폼 독립적인 부분이 강조되는 이식성, 모듈 재사용성의 증대와 자동 메모리 관리를 통한 메모리 누수 현상을 방지하고, 안정성과 보안이 뛰어나다. 또한 개발자의 가용성이 좋고 Java 기술의 발전에 의한 영속성을 지니고 있다[4].

EmbeddedJava는 특정한 기능을 수행하는 API

만을 내장하게 되므로 Core API의 개념을 제공하지 않는다. 또한, EmbeddedJava 구현은 기능에 따라 모두 다르게 구현한다. 이러한 방식의 장점은 PersonalJava에 비해서 적은 양의 메모리 FootPrint(256K~521K)를 요구한다는 점이다. C 또는 C++ 프로그램에서는 프로그래머의 실수로 인한 포인터 변수의 잘못된 사용으로 인하여 메모리 누수 현상이 발생할 수 있지만, Embedded Java는 이미 만들어진 Garbage Collector에 의해 자동적으로 메모리 누수를 방지하고 있다. 게다가 Java는 C와 C++에서 사용하는 포인터를 사용하지 않기 때문에 메모리를 효율적으로 사용할 수 있다[5].

잘 구성된 네트워크를 통해 내장형 시스템들을 동적으로 업그레이드 하거나, 제어를 해야 할 경우 Java가 적합하다. 제품 공급자들은 시스템을 인도한 후에 제품에 대한 업그레이드를 인터넷을 통하여 쉽게 할 수 있다. 어플리케이션을 위한 Java의 바이트 코드들은 동등한 C 객체 코드들보다 더 작다. 또한 다운 로드에 대한 원격 접근이 가능하며, 프로그램 계층에서 메모리 최적을 위해 이용되기도 한다.

위와 같은 특징들은 정보가전 제품에 적용하기 쉽도록 EmbeddedJava 응용 환경을 만들고 응용에 대한 요구들과 밀접한 관계를 갖는 API들을 포용한다. 이러한 EmbeddedJava는 산업 제어기, 프로세스 컨트롤러, 과학 기계 등을 포함한 내장형 장치의 매우 제한적인 메모리를 최적화 할 수 있다.

3. EmbeddedJava API 명세

EmbeddedJava™ API는 JDK 1.1로부터 파생되며 내장형 장치에 필요하도록 만들어진 소프트웨어이다. 또한 EmbeddedJava는 JDK 1.1 명세에 기초를 두기 때문에 독립형 명세는 아니다[4].

EmbeddedJava API는 JDK 1.1이 가지고 있는 패키지의 모든 방법과 모든 필드를 Embedded Java에서 응용으로 이용할 수 있다. Java 필터는 EmbeddedJava API에 의해 요구되는 필드와 리스트를 결정한다. EmbeddedJava 개발 환경은 EmbeddedJava API가 필요하게 되지만 불필요한 Java 클래스나 패키지 등은 EmbeddedJava API를 구현할 때 생략할 수 있다. EmbeddedJava 플랫폼의 구성은 그림 1과 같다.

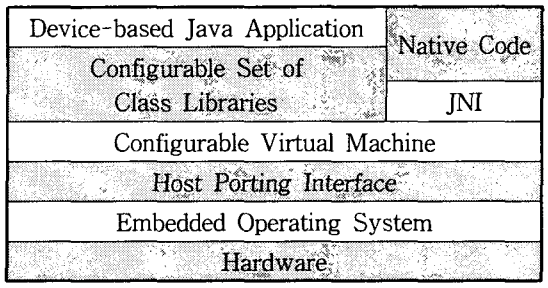


그림 1. 장치 기반 자바 어플리케이션

3.1 EmbeddedJava API

표 1은 EmbeddedJava 플랫폼의 JDK 1.1 API 프로그램 인터페이스 안에서 제공되는 패키지들에 대한 리스트이다[5]. 표 1에서 보는 것과 같이 Java 표준에서 제공하고 있는 java.applet 패키지

표 1. EmbeddedJava 제공 패키지

패키지	제공	패키지	제공
Java.applet	Unsupported	Java.rmi	Configurable
Java.awt	Configurable	Java.rmi.dgc	Configurable
Java.awt.datatransfer	Configurable	Java.rmi.registry	Configurable
Java.awt.event	Configurable	Java.rmi.server	Configurable
Java.awt.image	Configurable	Java.security	Configurable
Java.awt.peer	Configurable	Java.security.acl	Configurable
Java.beans	Configurable	Java.sql	Configurable
Java.io	Configurable	Java.text	Configurable
Java.lang	Configurable	Java.text.resources	Configurable
Java.lang.reflect	Configurable	Java.util	Configurable
Java.math	Configurable	Java.util.zip	Configurable
Java.net	Configurable		

를 제외하면 다른 패키지들은 Configurable로 사용할 수 있다.

3.2 구현 요구 사항들 (Implementation Requirements)

3.2.1 RTOS(Real Time Operating System) 요구

EmbeddedJava 플랫폼에서는 RTOS 위에서 다음과 같은 몇 가지 요구 사항이 필요하다.

- 메모리 관리
- 쓰레드 지원
- 파일 시스템 지원
- 그래픽/윈도우 시스템
- 네트워크 시스템

내장형 장치들은 한정된 메모리를 사용하기 때문에 메모리 관리는 반드시 필요하다. 그리고 RTOS의 특성에서 멀티타스킹(multitasking)을 해결하기 위해서는 Java의 쓰레드(thread)가 제공되어야 한다. 내장형 시스템 자체의 파일 시스템을 가지고 있어야 하며, 최근의 사용자는 그래픽을 중요시 하기 때문에 RTOS 환경에서 그래픽 처리가 필요하게 된다. 또한 네트웍을 통한 장치들의 제어는 필요에 따라 사용될 수 있다[6].

3.2.2 JNI(Java Native Interface)

EmbeddedJava 플랫폼은 고유 방법인 JNI를 사용한다[7]. JNI는 프로그램 인터페이스로서 C, C++, 어셈블리 등과 같은 프로그램 언어로 작성된 라이브러리나 어플리케이션을 가지고 내부 처리를 위해 JVM(Java Virtual Machine)이 실행되는 Java 코드이다. JNI의 가장 중요한 이점은 JVM에 기초를 두고 실행되어야 한다는 제한이 없다는 점이다. 따라서 JVM 판매자는 VM의 어느 부분에 영향을 가지지 않고 JNI를 추가적으로 제공할 수 있다. 프로그래머는 고유의 어플리케이션과 라이브러리를 사용할 수 있고 JNI에서 제공되는 모든 Java VM을 가지고 작업을 할 수 있다.

다음과 같은 경우에 JNI 프로그램이 요구된다.

- 표준 자바 클래스 라이브러리가 어플리케이션에서 필요로 하는 특징들이 플랫폼 의존적이어서 제공되지 않을 때
- 또 다른 언어로 라이브러리를 가지고 있을 때나 JNI를 통해 자바 코드로 받아들이기를 원할 때
- 어셈블리와 같은 낮은 단계에서 시간 임계 코드의 작은 부분에서 실행되기를 바랄 때

JNI 프로그램으로 다음과 같은 고유 방법을 사용할 수 있다.

- 생성, 점검, 자바 객체 업데이트
- 자바 메소드 호출
- 예외처리
- 클래스와 클래스 정보 적재
- 런타임 타입 체크 수행

여기서 자바 VM이 내장되기 위해 임의의 고유 어플리케이션이 이를 가능토록 하는 Invocation API를 가지고 JNI를 이용할 수 있다. 따라서 JNI를 이용하면 프로그래머가 VM 소스 코드 없이 어플리케이션 자바를 쉽게 만들 수 있다.

4. PJAE(Personal Java Application Environment)

Personal Java는 J2ME(Java2 Micro Edition)의 모체적 성격을 지니고 있는 Java 명세이다[8]. 그리고 PJAE는 Java 프로그램 언어로 작성된 소프트웨어가 실행되기 위한 Java 어플리케이션 환경이 된다. 이것은 네트워크에 연결할 수 있는 소형 기기에 적합한 소규모의 Java 실행 환경을 정의한 것이며, 기본적으로 Personal Java는 JDK1.1.8

API의 대부분의 기능을 수용하고 있으며 PDA (Personal Digital Assistants), PPC(Palm Personal Computer) 그리고 HPC(Hand-held Personal Computer)와 같은 소형 휴대용 컴퓨터에 적합한 명세이다.

PJAE의 명세에 의한 패키지는 6가지의 경우로 나누어진다.

1) PJAE에서 완전하게 지원되는 기능으로 JDK 1.1.8이나 JDK 1.2에서 대응하는 기능을 그대로 유지해야 하는 Required API가 있다.

2) PJAE에서는 지원되지 않는 기능들의 집합을 모은 것이 Unsupported API이며, 이 패키지에 포함된 임의의 클래스, 메소드나 필드를 참조하면 JVM에서 java.lang.NoClass- DefFound Error 예외가 발생한다.

3) PJAE에서 반드시 구현하도록 요구되지 않는 기능들의 집합으로 Optional API가 있고, 이것은 개발자의 선택에 달려 있다.

4) JDK 1.1.8이나 JDK 1.2에는 없으면서 PJAE API에서만 제공하는 Specific API가 있다.

5) PJAE에 의해서 완전하게 지원되지는 않는 기능임을 의미하는 Modified API가 있다. Modified 패키지는 그 패키지에 포함된 일부 클래스가 선택적이거나 PJAE Specific 혹은 Modified 메소드 클래스이며 Modified 클래스는 일부 메소드가 Optional 메소드이거나 PJAE Specific 혹은 Modified 메소드로 그 의미가 JDK에서와 다를 수 있는 API이다.

6) Minimum PJAE인데 이것은 Optional 패키지, Optional 클래스, Optional 메소드를 지원하지 않는 PJAE의 구현을 의미한다.

이와 같은 내용과 JDK를 기반으로 제공되는 패키지를 도표로 나타내면 표 2와 같다[8].

그림 2에서 보는 것과 같이 EmbeddedJava와

표 2. PJAE 제공 패키지

패키지	제 공
java.applet	Required API
java.awt	Modified API
java.awt.datatransfer	Required API
java.awt.event	Required API
java.awt.image	Required API
java.beans	Required API
java.io	Modified API
java.lang	Modified API
java.lang.reflect	Modified API
java.math	Optional API
java.rmi 관련	Optional API
java.security 관련	Modified API
java.sql	Optional API
java.text	Required API
java.util	Modified API
java.util.zip	Modified API

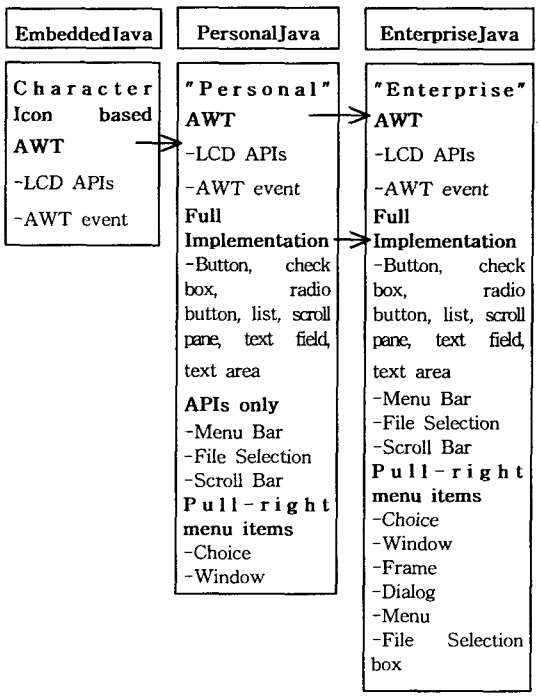


그림 2. JAE(Java Application Environment) 사이에서 AWT 차이점

PJAE와 EnterpriseJava는 Java 패키지를 사용하는 것은 비슷하지만 AWT 패키지에서는 현저하게 차이가 난다[9]. EmbeddedJava는 경량의 시스템에 적합하여 문자나 혹은 아이콘 기반의 AWT를 사용하기 때문에 Character-Based AWT라 부른다. 이것은 EmbeddedJava의 목적이 내장형 시스템의 제어에 그 목표를 두고 있기 때문이다.

5. EmbeddedJava 개발 프로세스

5.1 개발 프로세스

Sun사에서는 EmbeddedJava 기술과 함께 개발 프로세스를 제공하고 있다. 그 예로 JavaFilter™는 자바 어플리케이션을 실행하기 위해 자바 플랫폼에 의해서 사용될 수 있도록 명세 필드 리스트를 만든다. 또한 JavaFilter 툴에 의해 제공된 리스트를 사용하면서 어플리케이션에 의해 사용되는 필드를 선택할 때 사용되지 않는 부분을 제거함으로써 코드를 최적화할 수 있는 JavaCodeCompact™가 있다. 또, HTML, 이미지, 사운드 파일과 같은 보조 데이터 파일의 링크를 제공하는 JavaData Compact™가 있다[4].

EmbeddedJava 어플리케이션 개발의 첫번째 단계는 EmbeddedJava API를 이용하여 어플리케이션을 개발하는 것이다. 다음으로 어플리케이션과 프로그램 클래스들을 JavaFilter 툴로 변형하는 것이다. JavaFilter 툴은 JVM의 오퍼레이션과 어플리케이션을 위한 프로그램 요소와 유사하며 정적으로 내부 의존도를 분석한다. JavaFilter는 요구되는 클래스, 필드, 메소드 등을 리스트로 만들고 이것을 JavaCodeCompact가 제한된 RAM과 ROM으로 적합하도록 최적화 시킨다. 이들 개발 프로세스는 그림 3과 같다.

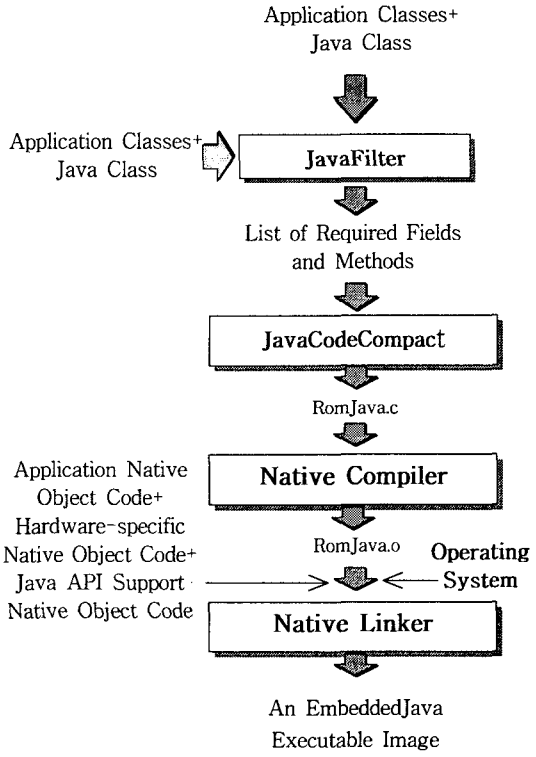


그림 3. EmbeddedJava 개발 프로세스

5.2. J2ME의 CLDC

Java 개발 환경은 크게 3가지로 구분된다. 우선 가장 규모가 큰 시스템 개발에 적합한 J2EE(Java 2 Enterprise Environment)와 Java 어플리케이션 개발을 목표로 하는 J2SE(Java 2 Standard Edition)가 있으며 마지막으로 한정된 자원과 낮은 성능의 프로세서를 내장한 휴대폰, PDA, 양방향 페이저 등의 모바일 장치와 셋탑 박스, 스크린 폰 등의 어플리케이션 개발에 적합하게 설계된 소프트웨어 개발 형식 J2ME(Java 2 Micro Edition)가 있다. J2ME에서 정의하는 Configuration에는 CLDC (Connected Limited Device Configuration)와 CDC(Connected Device Configuration)로 나누어진다[10]. 이중에서 CLDC는 핸드폰과 같이 제한된 환경을 가진 장치를 위한 Configuration이

고, CDC는 이보다 덜 제한된 환경을 가지는 장치에 적합하도록 설계되었다.

CLDC Configuration에 적용되는 장치들의 특징은 16~32비트의 CPU, 그리고 ROM, RAM 등의 총 메모리 공간은 160~512K이며 배터리를 주로 이용하도록 제한된 장치와 느린 네트워크 속도를 가지고 있는 장치들에 적용된다. 예를 들어 핸드폰, 양방향 페이저, PDA, 가전제품, 자동차 항법시스템 등이 이에 속한다[11].

CLDC는 Configuration을 통해 Java 언어 특징을 정의하고, 가상머신(VM: virtual machine)의 특징, 자바 클래스 라이브러리를 정의한다. 실제 구현이나 확장은 Profiles에서 담당하며 이것은 응용프로그램의 설치 및 관리, 사용자 인터페이스, 이벤트 처리, 데이터 저장 방법과 같은 역할을 가지고 있다. 이런 Profiles은 MIDP(Mobil Information Device Profile)가 맡고 있다. MIDP는 모바일 정보 기기에서 응용 개발 환경을 제공한다.

CLDC에 사용되는 VM은 KVM(Kilobyte Virtual Machine)이다. KVM 기술의 디자인 목표는 자바언어의 핵심 기능들을 유지하면서도 단지 수백 킬로 바이트의 메모리를 가진 한정된 성능의 장치에서도 동작하며 가능한 작고 완벽한 VM을 만드는 것이다. KVM은 수백 킬로바이트 이하의 메모리를 가진 16비트 혹은 32비트 RISC/CISC 마이크로 프로세서에서 최적의 성능을 낼 수 있도록 디자인 되었다. KVM은 VM과 클래스 라이브러리 크기를 줄이며, VM이 실행되는 동안 VM의 메모리 사용을 줄이고, VM의 구성 요소들이 장치에 적합하게 수정될 수 있다[12].

CLDC 프로그래밍은 기본적인 클래스 지원과 Applet과 같은 플랫폼이 없기 때문에 Sun사의 CLDC 참조 플랫폼인 kjava GUI 툴킷을 이용하게 된다. 현재 Sun사에서는 CLDC 1.0.3 버전을 제공하고 있다. 그림 4는 Sun사의 j2me_cldc에 포

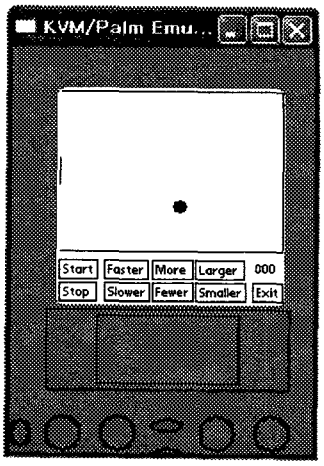


그림 4. Pong.class의 실행화면

함된 예제 Pong.class 파일을 KVM으로 실행한 결과이다. 이 예제에서는 화면에 메뉴를 작성하고 키보드 이벤트 처리를 통하여 게임을 할 수 있도록 작성된 프로그램이다.

KVM은 EmbeddedJava 플랫폼 개발 프로세스의 Java Code Compact(JCC) 유틸리티를 지원한다. 이 유틸리티는 자바 클래스가 VM에 직접 링크되는 것을 이용하여 VM의 시작시간을 단축한다. Sun사에서는 CLDC 환경에서 GUI 라이브러리를 Palm PDA에 맞추어 개발했기 때문에 기본적으로 Palm PDA 형태로 실행된다.

5.3 MIDP(Mobile Information Device Profile)

비슷한 특성을 가진 장치들이 가져야할 최소 요구 사항을 정의한 Configuration은 임베디드 프로그래밍을 위한 최소한의 API를 제공한다. 하지만 Configuration은 세련된 프로그램을 위한 응용과 다양한 기능들을 제공하지는 못한다. 반면 Profile은 Configuration 위에서 Java 응용 프로그램 수행을 위한 구체적인 기능을 제공하며 그 중의 하나가 MIDP다[11].

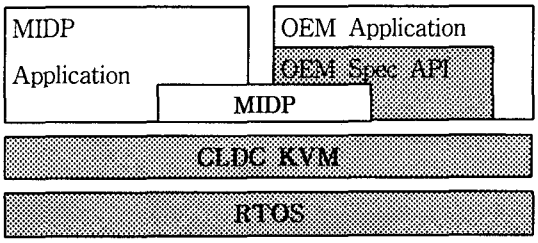


그림 5. CLDC 상에서의 MIDP

MIDP의 대상은 휴대폰, PDA, 양방향 페이저가 된다. 이러한 장치들은 96*54 픽셀 이상의 액정과 입력도구, 128K나 그 이상의 메모리, 응용프로그램 저장을 위한 8K 이상과 양방향으로 연결가능한 무선 네트워킹으로 구성한다. 따라서 MIDP는 Java API의 일부분이며 CLDC의 위에서 실행되는 J2ME 응용 프로그램 실행을 위한 완전한 실행환경을 제공하는 MIDP다. MIDP의 역할은 응용 프로그램의 정의 및 제어와 GUI 환경 제공과 HTTP를 이용한 네트워킹, 그리고 Timer 기능을 제공한다.

볼랜드사의 Jbuilder6.0 Enterprise 버전에서는 Mobileset2.0을 설치하여 MIDP 프로그램을 작성할 수 있다[13]. MIDP 프로그램은 반드시 javax.microedtion.midlet.MIDlet라는 추상클래스를 상속받은 클래스를 작성해야 한다. MIDlet 클래스는 애플릿을 작성하기 위해 무조건 Applet 클래스를 상속받아 작성되는 J2SE에서와 마찬가지로 상속되어 사용된다. MIDlet는 5가지의 기본 메소드를 표 3과 같이 제공한다.

표 3. MIDlet의 메소드

MIDlet 기본 메서드의 역할	
DestoryApp()	프로그램의 자원 해제와 상태저장
StartApp()	실행될 때 호출하여 자원 획득
PauseApp()	일시적인 자원 해제
NotifyPaused()	대시 상태 알림
NotifyDestroyed()	종료됨을 통보

프로그램은 Jbuilder6을 설치한 후 볼랜드사의 홈페이지에서 Mobilset2.0을 다운 받아 설치한다. 그리고 프로젝트를 생성한 후 Jbuilder6의 프로젝트 환경을 맞추고 나서 위저드를 통해 생성한 MIDlet 파일을 컴파일을 하면 된다. 예제 실행의 결과와 소스코드는 각각 그림 6과 그림 7에서 보인다.

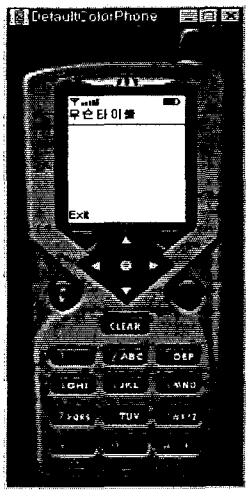


그림 6. JBuilder6.0에서의 MIDP 예제 실행

6. EmbeddedJava의 문제점

Java의 장점만을 보고 사용한다면 내장형 시스템에 자바를 적용할 때 짧은 개발 기간과 Java의 기술적 접근의 용의성 안에서 시스템 개발에 대한 해결책을 찾아내려고 할 것이다. 이것은 Java의 문제점인 많은 메모리를 필요로 하게 된다. 예를 들면 어플리케이션을 실시간으로 수행하기 위해서는 느린 실행 속도나 예측하지 못한 일들의 발생과 하드웨어 성능이 부족할 때 등을 대비하기 위해 일정한 메모리를 남겨두게 된다. 또한 JVM에 숨겨진 작업과 여러 새로운 프로세서의 클래스 라이브러리를 포팅하는 것에도 원인이 있다[14].

Java가 방대한 양의 시스템 메모리를 사용한다는 것은 내장형 장치의 설계에서 고려해야할 문제

```
package mit;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class MIDlet1 extends MIDlet {
  static MIDlet1 instance;
  Displayable1 displayable = new Displayable1();
  /** Constructor */
  public MIDlet1() {
    instance = this;
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  /** Main method */
  public void startApp() {
    Display.getDisplay(this).setCurrent(displayable);
  }
  /** Handle pausing the MIDlet */
  public void pauseApp() {
  }
 /** Handle destroying the MIDlet */
  public void destroyApp(boolean unconditional) {
  }
 /** Quit the MIDlet */
  public static void quitApp() {
    instance.destroyApp(true);
    instance.notifyDestroyed();
    instance = null;
  }
  private void jbInit() throws Exception {
  }
}
```

그림 7. 그림 6의 MIDlet 클래스

점이다. 이것은 Java 클래스 라이브러리, 어플리케이션 소프트웨어, 운영체제, JVM(Java Virtual Machine)을 위해 충분한 메모리가 필요하기 때문이다. 아주 작은 JVM이라도 최소 512Kbyte의 ROM이나 이와 같은 크기의 RAM이 필요하다.

내장형 시스템용 Java의 주요한 단점 중의 하나는 대단히 느린 실행 속도이다. 표준 JVM은 C 프로그램보다도 성능면에서 떨어지는데 이것은 Java 바이트 코드에 대한 해석 시간이 들기 때문이다.

Java에서 실시간에 대한 불만족은 JVM의 자동 메모리 관리 시스템인 Garbage Collection이 항상 VM 안에 포함되는 점이다. Java 프로그램

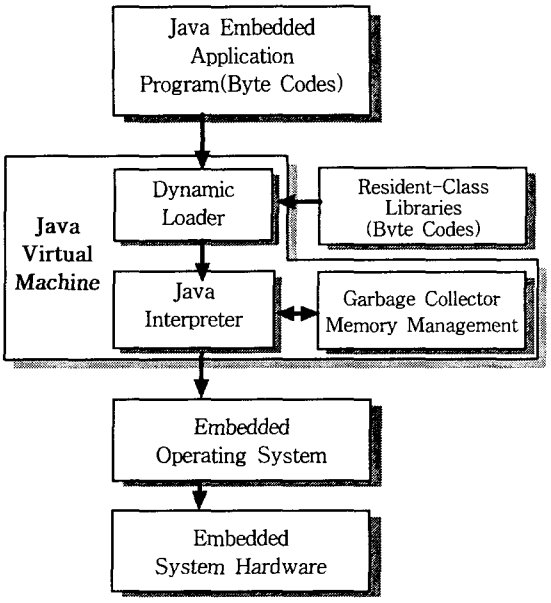


그림 8. JVM에서의 Garbage Collector의 메모리 관리

은 Garbage Collector가 완료될 때까지 멈추고 있다. 이때 Garbage Collection이 진행되는 동안에 사용자는 어떤 컨트롤도 할 수 없게 된다. 따라서 실시간 소프트웨어에서는 이것을 미리 예측해서 프로그램을 디자인해야 한다.

비록 Java가 쓰레드와 멀티타스킹을 제공한다고 하지만 실시간 언어에서는 그렇지 못하다. 실시간 시스템의 우선 순위가 높은 타스크가 완전히 수행될 때까지 낮은 우선 순위의 타스크들은 일시 중지해야 제한된 메모리를 효율적으로 사용할 수 있다. 성능 향상은 Java로 구성된 RTOS를 사용함으로서 해결책을 찾는다.

또 다른 문제점은 내장형 시스템에서 통합된 Java는 하드웨어에서 직접적인 접근이나 통신을 제공할 수 없다는 것이다. C나 C++ 언어는 포인터를 이용하여 하드웨어 접근이 가능하나 Java는 포인터를 사용할 수 없다. 이로 인해 개발자는 하드웨어 드라이버를 위해 C나 C++, 또는 어셈블리 언어로 작성해야 한다.

7. 결론 및 향후 연구과제

본 논문에서는 내장형 시스템을 위한 Sun사의 Java 플랫폼으로 EJAE(EmbeddedJava Application Environment), PJAE(PersonalJava Application Environment), J2ME(Java2 Micro Edition) 등에 대한 논의를 하였다.

이러한 내장형 JAVA 플랫폼은 기존의 Java가 가지고 있던 객체지향성, 이식성, 소프트웨어 재사용, 단순성, 안정성과 보안, 개발자 가용성, 영속성 등의 특징을 대부분 수용하며 제한된 자원, CPU 능력을 갖는 내장형 장치에 적합하도록 만들어졌다.

다가오는 미래에는 초고속 정보 통신의 완성과 함께 무선 인터넷, 정보가전 제품의 수요가 증가될 것이며 이에 따른 내장형 시스템에 대한 요구로 EmbeddedJava 플랫폼의 기술 수요가 급속도로 증가될 것이다.

따라서 내장형 플랫폼 기술 및 어플리케이션 서비스에 대한 연구가 활성화되어야 하며 내장형 시스템에 대한 표준 확립 등에 대한 연구가 계속되어야 할 것이다.

참 고 문 헌

[1] 하영국, 임신영, 함호상, "정보가전 및 내장형 장치를 위한 Java 기술", 전자통신동향분석, 제16권 제2호, 2001.4.

[2] Sun Microsystems, "http://java.sun.com."

[3] Sun Microsystems, "http://java.sun.com/products/embeddedjava"

[4] Sun Microsystems, "Technical Overview of EmbeddedJava Technology"

[5] Sun Microsystems, "EmbeddedJava™ API Specification", Versioin 1.0.

[6] 김선자, 김홍남, 김채규, "인터넷 정보가전용 RTOS 기술 현황", 정보과학회지, 제19권 제4호, 2001.4.

[7] Sun Microsystems, "http://java.sun.com/products/jdk/1.1/docs/guide/jni/spec/jniTOC.doc.html"

[8] Sun Microsystems, "http://java.sun.com/products/personaljava"

[9] 손종문, "가전제품 및 Mobile NC를 위한 Java 기술", 소프트웨어 기술동향 1997.3.

[10] Sun Microsystems, "http://java.sun.com/j2me/"

[11] 강성윤, 이경범, 홍성인, "클릭하세요! 자바 모바일 프로그래밍", 대림, 2002.1.

[12] 강룡, 김근호, 이재만, 임종관, 조성환, "about Mobile Programming", 영진.com, 2001.6.

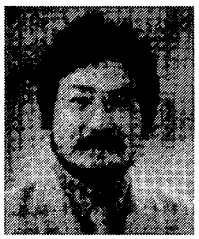
[13] Borland, "http://www.borland.com"

[14] Warren Webb, "EmbeddedJava: an uncertain future", EDN, May 13, 1999.

[15] Bruce R, "JN: OS for an Embedded Java Network Computer", IEEE Micro, 1997.

김 달 중

- 1997 대전대학교 컴퓨터공학과(공학사)
- 1999 대전대학교 컴퓨터공학과(공학석사)
- 1999~현재 대전대학교 멀티미디어 컨텐츠·기술 센터 연구원
- 1999~현재 대전대학교 컴퓨터공학과 박사과정
- 관심분야 : 소프트웨어 공학, 멀티미디어 응용, 프로그래밍 언어, 게임 공학



하 수 철

- 1981 홍익대학교 컴퓨터공학과(학사)
- 1986 홍익대학교 대학원 컴퓨터공학과(석사)
- 1990 홍익대학교 대학원 컴퓨터공학과(박사)
- 1987~현재 대전대학교 컴퓨터공학과 정교수
- 1999~현재 멀티미디어 컨텐츠·기술 센터 소장
- 2002~현재 한국멀티미디어학회 이사, 학회지편집위원
- 2001~현재 한국컴퓨터게임학회 이사
- 2001 워싱턴 주립대학교 방문 연구교수
- 2000~현재 한국디지털컨텐츠학회 이사
- 2001~현재 한국컨텐츠학회 이사
- 1999~2000 한국정보처리학회 논문지 편집위원
- 1998~현재 한국정보처리학회 멀티미디어시스템 연구회 부위원장
- 1996 한국전자통신연구원 초빙연구원
- 1991~1992 플로리다 주립대학교/텍사스 주립대학교 객원교수
- 1981~1984 Army Logistics Command, System Analyst
- 관심분야 : 소프트웨어공학(객체지향화), Java/Java3D, Web Service, 멀티미디어 응용, 게임공학, 시각언어, 프로토콜기술언어, XML, UML