리눅스 시스템에서의 자바 클립보드 클래스 라이브러리 설계 및 구현

정명조[0], 성영락[†], 이철훈
충남대학교 컴퓨터공학과, [†] 국민대학교 전자정보통신공학부
(mjjung[0], chlee)@ce.cnu.ac.kr, [†] yeong@mail.kookmin.ac.kr

Design and Implementation of Java Clipboard Class Library Based On the Linux System

Myoung-Jo Jung[0], Yeong Rak Seong[†], and Cheol-Hun Lee
Dept. of Computer Engineering, Chungnam National Univ.
[†] School of Electrical Engineering, Kookmin Univ.

요 약

오늘날 거의 모든 윈도우 시스템들은 어플리케이션간의 데이터 이동을 위한 클립보드 기능을 제공하고 있다. 최근 각광받고 있는 자바기술도 이런 클립보드 기능을 제공하기 위해 표준 클래스 라이브러리에서 클래스를 정의하고 있다. 그러나 정의된 클래스는 주로 자바 인터페이스와 기본적인 기능만을 가진 클래스로 이루어져 있기 때문에 실제 사용하기 위해서는 개발자의 별도의 구현이 있어야 한다. 그리고 어플리케이션간에 데이타를 주고 받기 위해서는 system clipboard 를 이용하여야 하는 데 이것은 시스템에 의존적인 부분이다. 따라서 system clipboard 기능을 가지도록 하려면 타겟시스템에 따라 클래스 라이브러리 계층에서 별도로 구현해야 한다. 본 논문에서는 리눅스의 X 윈도우 시스템 상에서 클립보드 기능을 지원하기 위한 클래스 라이브러리를 구현하고자 한다.

1. 서 론

요즘 대부분의 윈도우 시스템들은 어플리케이션간의 data 를 주고 받을 수 있는 클립보드 기능을 제공한다. 클립보드는 주로 Copy/Cut/Paste 연산에 사용되는 데 자바기술에서도 이런 클립보드 기능을 제공하기 위한 클래스들을 정의하고 있다.

그러나 SUN™ 의 명세서에 명시된 클래스들은 인터페이스로 이루어져 있거나 클립보드를 구현하기 위한 기본적인 메소드만 가진 클래스로 이루어져 있기 때문에 향상된 클립보드 기능을 제공하기 위해서는 별도의 구현이 필요하다. 그리고 자바 어플리케이션은 자신의 클립보드를 생성할 수 있는데 이 클립보드는 자신만이 사용할 수 있다. 만약 다른 어플리케이션과 클립보드를 공유하여 data 를 주고 받기를 원한다면 system clipboard 를 이용하여야 한다[1]. System clipboard 를 사용하기 위한 구현은 플랫폼에 의존적인 부분으로, 자바 플랫폼을 특정 시스템에 탑재하고자 할 때 클래스 라이브러리 계층에서 따로 구현해야 한다. 이런 시스템에 의존적인 부분은 SUN™ 에서 제공하는 명세서에도 제대로 다루어지지 않아서 실제로 클래스 라이브러리를 개발할 경우 어려운 작업을 필요로 한다.

본 논문에서는 리눅스의 X window system 상에서 클립보드 기능을 지원하기 위한 클래스 라이브러리를 구현하고자 한다. 2 장에서는 구현에 사용된 자바 네이티브 인터페이스(java native interface : JNI)와 자바 AWT clipboard 에 대해 설명하고 3 장에서는 본 논문에서 구현한 클립보드 기능에 관한 클래스 라이브러리의 설계와 구현에 대해서 설명한다. 4 장에서는 실험 환경 및 결과를 설명하고, 5 장에서는 결론 및 향후 과제에 대하여 기술한다.

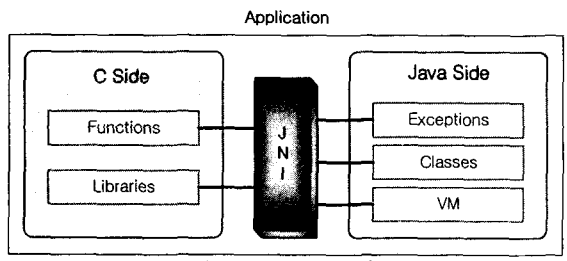
2. 관련 연구

2.1 Java Native Interface

JNI 는 자바 어플리케이션을 작성할 때 자바가 아닌 다른 언어(C, 어셈블리어 등)를 사용할 수 있는 기능을 지원하기 위한 것으로써 자바 언어와 다른 언어(Native Code)와의 인터페이스를 담당한다. Native code 를 사용하는 경우는 다음과 같다.

- ◆ 플랫폼 의존적인 특성을 이용하고자 할 경우
- ◆ 다른 언어로 작성된 기존의 라이브러리를 이용하고자 할 경우
- ◆ 어셈블리 같은 저수준의 언어의 사용이 필요한 경우

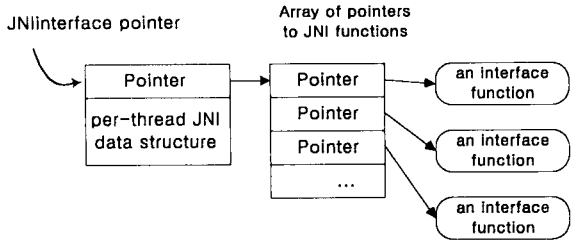
Native code 는 JNI 를 통하여 자바 메소드를 호출하거나 자바 객체를 생성하고 갱신할 수 있으며 런타임 타입 체크를 수행할 수 있다.



[그림 1] JNI 의 역할

Native code 는 JNI 함수를 호출함으로써 자바가상머신의 특성들에 접근할 수 있으며, JNI 함수들은 인터페이스 포인터를 통하여 이용할 수 있다. 인터페이스 포인터는 이중 포인

터이며, 이 포인터는 배열안에서 지정된 오프셋에 있는 인터페이스 함수를 가리킨다[2].



[그림 2] Interface Pointer

본 논문에서는 플랫폼에 의존적인 부분인 system clipboard 기능을 구현하고 X window system 의 x 라이브러리를 이용하기 위해 native code 로 C 언어를 사용하였다.

2.2 자바 AWT Clipboard

클립보드는 어플리케이션간에 전달할 data 를 일시적으로 저장하는 역할을 한다. 어플리케이션은 Clipboard 객체를 생성함으로써 다수의 클립보드를 사용할 수 있지만 이때 생성된 클립보드는 로컬 클립보드이며 생성한 어플리케이션 자신만 사용할 수 있다. 다른 어플리케이션과 data 를 주고 받을 수 있는 기능을 제공하기 위해서는 기반이 되는 window system 의 system clipboard 를 사용하여야 한다[3].

클립보드 기능을 제공하기 위한 SUN 사의 표준 클래스 라이브러리는 다음과 같으며, 여기서는 간단한 기능만 설명하고 3 장에서 자세히 살펴보도록 한다[4].

[표 1] 클립보드 기능과 관련된 자바 클래스

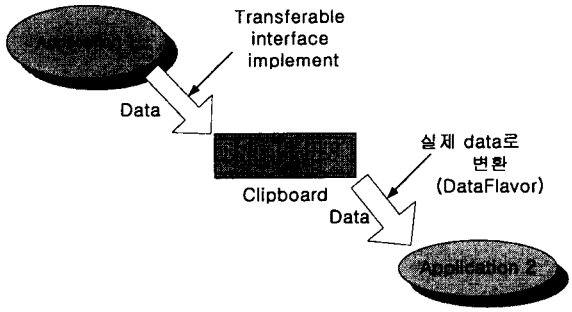
이름	구분	기능
Clipboard	Class	Data 를 저장하기 위한 객체
Clipboard Owner	Interface	클립보드에 data 를 저장하는 객체를 위한 interface
DataFlavor	Class	캡슐화된 data 로부터 실제 data 를 얻는 기능 제공
String Selection	Class	Textual data 를 캡슐화하는 기능 제공
Transferable	Interface	Data 를 캡슐화하는 클래스를 위한 interface
UnsupportedFlavorException	Class	요구되는 data flavor 가 제공되지 않는 경우 발생하는 exception

3. 자바 클립보드 클래스 라이브러리 구현

클립보드 기능을 제공하기 위한 data 전달 메커니즘에 대하여 설명하고 본 연구에서 구현한 클래스에 대해서 기술하며 system clipboard 를 사용하기 위한 native code 에 대해 살펴본다.

3.1 클립보드를 통한 data 전달 메커니즘

Data 는 클립보드에 직접 저장될 수 없고 캡슐화 공정을 거쳐 일정한 형태로 저장되며 저장된 데이터를 가져와서 사용하기 위해서 캡슐화된 데이터를 실제 데이터로 변환하는 과정이 필요하다. 자바에서는 캡슐화를 위해 transferable interface 를 implement 하도록 구성되어 있으며 캡슐화된 data 에서 실제 데이터를 얻기 위해 data flavor 클래스를 제공한다.



[그림 3] Clipboard 를 통한 data 전달과정

data 를 캡슐화하기 위해 transferable 객체는 data type 에 대한 정보와 실제 데이터로 변환하기 위한 제공되는 data flavor 의 list 를 가진다. 그리고 data flavor 는 가져오고자 하는 data type(image, text 등)에 대한 정보와 data 전송방식(byte stream, Image object, array of pixel 등)에 대한 메소드를 제공한다.

어느 data type 에 대하여 클립보드 기능을 제공하기 위하여 data type 별로 transferable interface 를 implement 한 클래스가 만들어져야 하며, data type 에 따라 실제 data 를 얻을 수 있는 메소드가 data flavor 에서 제공되어야 한다.

3.2 클립보드 기능을 제공하기 위해 구현된 클래스들

본 연구를 위해 구현한 내용은 SUN™ 의 리눅스용 JDK1.2.2 에서 기본적으로 제공하는 클래스의 수정과 각 data type 을 위해 Transferable interface 를 implement 한 클래스들의 구현, system clipboard 를 사용하기 위한 클래스들의 구현으로 요약할 수 있다. 제공되는 data type 은 string, image, 자바 Object 이다.

◆ JDK 의 표준 클래스 라이브러리
- DataFlavor : 캡슐화된 string, image, 자바 object 형태의 data 를 실제 데이터로 변환할 수 있는 메소드들을 추가

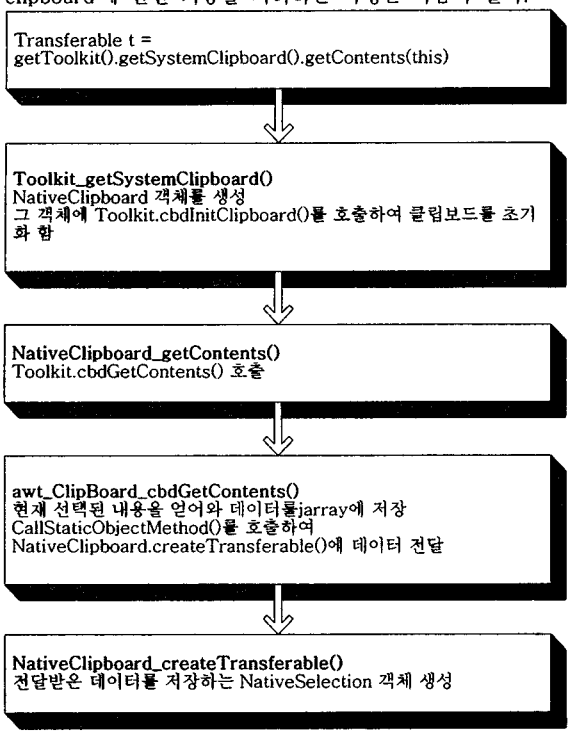
◆ Transferable interface 를 implement 한 클래스들
- StringSelection : string 형태의 data 를 캡슐화하는 기능 제공
- ImageSelection : image 형태의 data 를 캡슐화하는 기능 제공
- ObjectSelection : 자바 Object 형태의 data 를 캡슐화하는 기능 제공

◆ System Clipboard 를 사용하기 위한 클래스들
- NativeClipboard : Clipboard 클래스를 상속하며 system clipboard 의 생성과 제어에 관한 기능 제공
- NativeSelection : 자바 DataFlavor 의 data name 과 X window 의 Data name 의 format 를 번역해주는 기능

3.3 구현 native code

X window system 의 system clipboard 를 제공하기 위한 구현은 X library 와 C 언어를 사용하였다. 구현된 native code 는 system clipboard 를 초기화하고 선택된 영역의 content 를 ByteArray 를 가지는 새로운 데이터로 변환하여 Java level 로 전달하는 기능을 한다. Native code 계층에서

clipboard 에 관한 기능을 처리하는 과정은 다음과 같다.



[그림 4] Native code 계층에서 Clipboard 기능 처리 과정

4. 실험 환경 및 결과

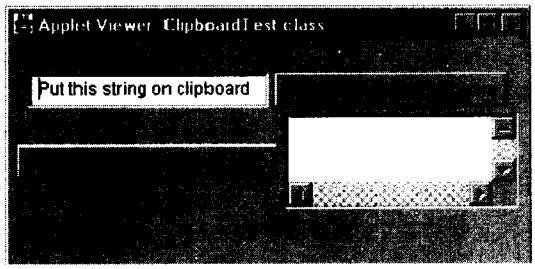
본 연구를 위한 실험환경은 다음과 같다.

[표 2] 실험 환경

Target machine	IBM 호환 PC
운영체제	범용 리눅스(커널 버전 2.4.x)
JVM	Linux 용 JDK1.2.2
클래스 라이브러리	Linux 용 JDK1.2.2
컴파일러	Linux 용 JDK1.2.2 의 컴파일러
Native code 컴파일러	Linux 상의 gcc

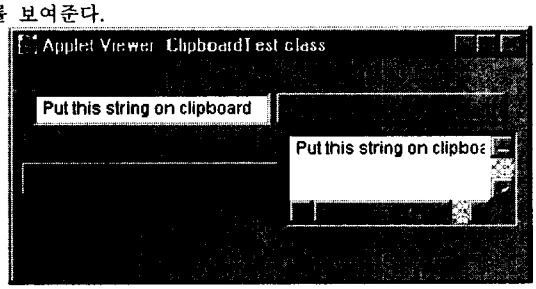
실험 프로그램을 동작시키기 위해 클립보드 관련 클래스를 제외한 다른 클래스들은 JDK1.2.2 의 클래스 라이브러리를 그대로 사용하였고, 구현 클래스들의 native code 부분은 JDK1.2.2 의 JVM 과 인터페이스를 맞추었다. 실험 프로그램은 textfield 상에 text 를 입력받고 button 컴포넌트에서 event 가 발생할 때 입력받은 data 를 system clipboard 로 copy 하거나 system clipboard 로부터 data 를 가져와서 다른 textfield 에 나타내주도록 동작을 한다. [그림 5], [그림 6]은 실험한 결과를 보여준다.

Copy To System Clipboard 버튼은 비활성화 상태를 유지하고 있다가 위에 위치한 textfield 에 text 가 입력되면 활성화되고 클릭이 되면 system clipboard 에 data 를 저장하게 된다. 이 과정까지를 [그림 5]에서 보여주고 있다.



[그림 5] 실험 결과 1

다음에 Paste From System Clipboard 버튼이 눌려지면 위의 system clipboard 에 있던 text data 가 밑에 위치한 textfield 에 나타나게 된다. [그림 6]은 동작이 완료된 상태를 보여준다.



[그림 5] 실험 결과 2

이상의 실험과정을 통하여 본 논문에서 구현한 클립보드 기능에 관련된 클래스들이 올바르게 동작하는 것을 알 수 있다.

5. 결론 및 향후 과제

본 논문에서는 리눅스 시스템을 타겟시스템으로 하여 X window system 의 x library 를 이용하여 clipboard 기능을 지원하기 위한 자바 클래스 라이브러리를 구현하였다. 그러나 3 개(string, image, 자바 Object)의 data type 만 지원하므로 기존의 여러 가지 컴포넌트를 이용한 Copy/Cut/Paste 연산을 수행하는 기능이 부족하다. 향후 연구과제에서는 다양한 data type 을 지원하고 마우스로 블록을 지정하여 copy/cut/paste 연산을 수행할 수 있는 drag&drop 기능과 같은 사용자가 편리하게 사용할 수 있는 기능을 제공하도록 하여야 할 것이다.

참고문헌

[1] Patrick Chan, Rosanna Lee, *The Java Class Libraries,* 2nd Edition.
[2] Java Native Interface Specification http://java.sun.com/products/jdk/1.2/docs/guide/jni
[3] Clipboard and Data Transfer http://developer.java.sun.com/developer/Books/GJ21AWT/ch20.pdf
[4] http://java.sun.com/products/jdk/1.2/docs/api/index.html