<JSTORM>

자바 네트워킹

Revision <1.0>

중앙대학교 컴퓨터공학과
자바 동호회
JSTORM
http://www.jstorm.pe.kr

Document Information

Document title:	자바 네트워킹
Document file name:	자바네트워킹.doc
Revision number:	<1.0>
Issued by:	박지훈(pjh@jstorm.pe.kr)
	편집 : 윤준호(junoyoon@jstorm.pe.kr)
Issue Date:	<2000/8/2>
Status:	Final

Content Information

Audience	초보자
Abstract	자바의 소개와 객체 지향 기법에 대한 설명 자바에서의 그래픽과 사운드에 대한 에제 간단한 네트웍 게임을 코딩함으로써 자바의 기초를 이해 - 마이크로 소프트웨어 원고 입니다.
Reference	
Benchmark information	

Document Approvals

고문	Signature	date
지위	Signature	date

.

Revision History

Revision	Date	Author	Description of change

Table of Contents

자바 네트워킹

자바는 초창기 때 생소하다는 반응을 넘어서, 현재는 많은 사람들에 유용한 프로그래밍언어라는 평판을 얻으며 널리 보급되고 있다. 시중에 자바관련 서적들이 넘치고 있는 것도 그런 문맥에서 이해할 수 있을 것이다.

그렇지만 현재 자바와 관련된 많은 책들은 자바언어가 "무엇(what)"인가에 대한 내용을 주로 담고 있다. 즉 자바 문법의 설명과 관련예제에만 치중하고 있는 것이다. 그러나 문법을 특정 어플리케이션에 구체적으로 어떤 원리하에 어떻게 활용하는지에 대한 자세한 설명과 예제는, 중요함에도 불구하고 지면관계상 강조되지 않고 있는 상황이다.

그래서 본 책은 자바언어가 강력한 개발 수단으로 적용되는 멀티미디어와 네트워크 프로그래밍에 많은 지면을 할당하고 있다. 즉 독자가 자바언어의 문법은 어느정도 익힌 상태라고 가정하며, 그 문법들을 구체적으로 이런 어플리케이션에 어떻게 응용하는 지에 대해 각종 예제를 들어 설명할 것이다. 또한 자바언어의 프로그래밍환경과 객체지향프로그래밍의 원리에 대해서도 설명하였다.

따라서 만약 자바를 처음 접하는 초심자는 이 책을 주교재로 삼을 것이 아니라, 부교재로 삼는 것이 바람직할 것이다. 주교재로 삼은 책을 통해 문법을 익히며, 본 책을 통해서는 그 문법을 어플리케이션에 활용하는 방법을 습득하여 좀 더 자바활용의 폭을 넓힐 수 있을 것이다. 또한 이미 자바를 살짝 맛보신 독자는 이 책을 통하여, 자바의 구체적인 활용 방법을 익힐 수 있을 것이다.

본 책에서 설명하는 자바언어는 JDK 1.0.1을 기준으로 삼고, 모든 예제는 Window 95하에서 컴파일 및 실행을 하였다. 웹 브라우저는 네트스케이프사의 Netscape 3.0 및 마이크로소프트사의 Explorer 3.0을 사용하였다.

현재 자바언어는 JDK1.1 베타버전까지 개발된 상태이지만 본 책은 한 단계 낮은 버전인 JDK 1.0.X버전 위주로 쓰여졌다. 그 이유는 JDK 1.0.X버전에서 컴파일된 프로그램은 JDK1.1에서도 그대로 동작하고, 현재 JDK1.1은 베타버전이라서 정식버전이 발표될 때 내용이 바뀔 수도 있고, 또한 실제로 JDK1.1하에서 컴파일된 자바프로그램을 실행할 수 있는 브라우저가 아직 현재로선 없기 때문이다.

1부 자바언어 기본

자바언어의 기본적인 문법과 관련된 활용사례를 알아본다.

1. 자바프로그래밍 환경

자바언어는 문법을 확실히 익히는 것도 중요하지만, 자바언어가 실행되는 컴퓨팅 환경에 대한 이해가 더욱 중요하다. 그것은 자바언어가 일반언어와는 달리 인터넷이라는 네트웍크환경에서 동작하도록 의도되어졌기 때문이다.

자바언어의 특성

자바언어의 특성은 대략 다음과 같다.

* 기존언어의 복잡성을 단순화시켰다.

  널리 사용되는 객체지향언어인 C++의 복잡한 특성(포인터, 다중상속, 연산자 오버로딩 등)을 제거하여 보다 단순화하였다. 그렇지만 멀티쓰레드 등 기존언어에 없던 특성들을 첨가시켰기 때문에 거시적으로 보았을 때, 자바언어는 그렇게 간단하지는 않다.(쉽게 생각하면 다친다!)

* 객체지향적이다.

  기존 객체지향언어보다 더 순수한 객체지향언어를 목표로 만들어졌다.

* 네트웍을 이동하여 실행되며, 분산환경을 지원한다.

  프로그램코드 자체가 네트웍을 이동하여 실행될 수 있다. 또한 특히 인터넷 환경에서 동작하도록 각종 클래스 라이브러리 등을 제공한다.

* 아키텍처 중립적이다.

  컴퓨터의 종류에 상관없이 실행될 수 있도록, 인터프리터방식을 취하고 있다. 즉 특정 컴퓨터에서 컴파일된 실행화일은 다른 컴퓨터에서 소스를 컴파일할 필요없이 그대로 실행가능하다.

* 견고성이 있다.

  C/C++보다 컴파일시에 보다 엄격한 타입 체킹(type checking)을 한다. 그리고 포인터 타입을 제거했기 때문에, 잘못된 포인터의 사용으로 말미암은 오류가 제거되었다.

* 인터프리터 방식이다.

  자바소스화일을 컴파일하여, 머신 코드(machine code)가 아닌 중간 코드인, 바이트 코드(Byte Code)를 생성한다. 바이트코드는 자바 인터프리터나 자바프로그램을 실행할 수 있는 브라우저가 있는 어떤 컴퓨터에서도 그대로 실행될 수 있다.

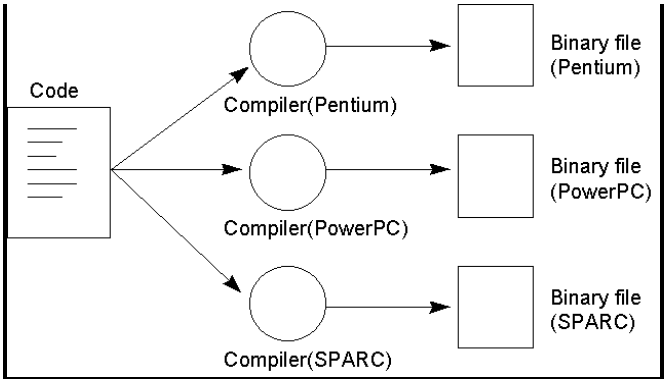
* 멀티 쓰레트(multithread)를 지원한다.

  한 프로그램내에서 동시에 여러 코드를 실행할 수 있다.

이 중에서 가장 핵심은 인터프리터방식과 아키텍처 중립적, 네트워크 지원 등의 특성일 것이다. 이런 특성들은 인터넷에 연결된 컴퓨터들이 다양한 종류이기 때문에, 그런 컴퓨터들에서 실행될 수 있기 위한 수단이다.

컴파일 방법의 장점

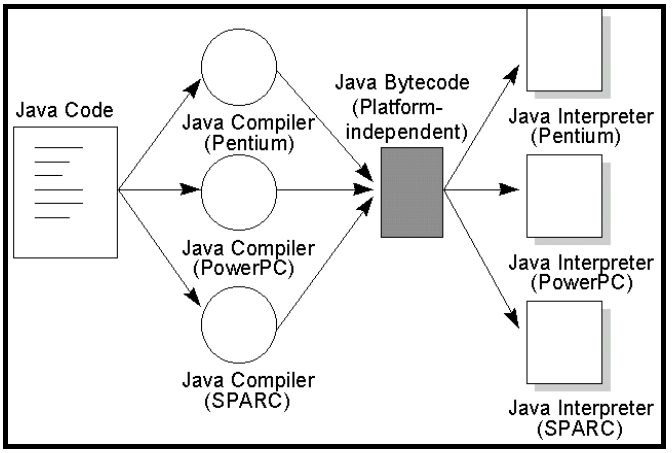
인터프리터방식이 갖는 장점을 알기 위해 다음 그림을 보자.



<그림 1> 기존 컴파일 과정

위 그림은 인터프리터방식이 아닌, 일반 프로그래밍언어(C로 생각해보자)의 컴파일과정이다. 특정 컴퓨터에서 컴파일된 코드는 그 컴퓨터의 CPU에 맞는 머신 코드(실행화일)이 생성되기 때문에 그 코드는 다른 컴퓨터에서 실행될 수 없다. 다른 컴퓨터에서 실행되게 하려면, 소스코드를 그 컴퓨터의 컴파일러를 이용해 컴파일하여 실행해야 한다.

인터프리터방식의 컴파일과정은 다음 그림과 같다.

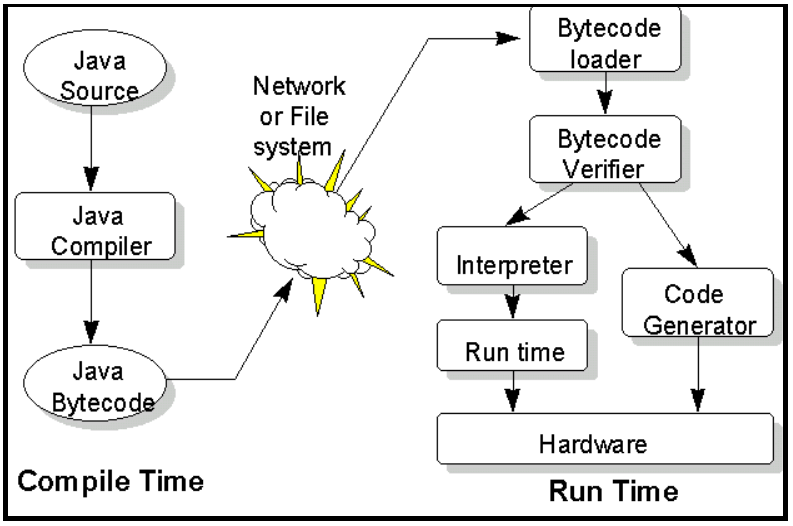


<그림 2> 인터프리터방식의 컴파일과정

자바는 특정 컴퓨터에서 컴파일되어 질 때, 그 컴퓨터의 머신 코드가 생성되는 것이 아니라, 자바 가상 기계(Java Virtual Machine)이라는 가상기계에 맞는 기계에 독립적인 코드가 생성된다. 이런 코드는 자바인터프리터가 탑재된 어떤 컴퓨터에서도 그 코드를 인터프리트하여 실행할 수 가 있게 된다. 이런 방법의 장점은 일단 한 번 자바언어로 된 프로그램소스를 컴파일하면, 그 컴파일된 크드는 어떤 수정도 없이 다른 컴퓨터에서도 실행될 수 있다는 것이다. 즉 이식성에 대한 문제가 해결된다는 것이다. 단점은 인터프리트방식이기 때문에, 속도가 상대적으로 약간 느리다는 것이다. 그러나 요즘은 하드웨어의 발달로 프로그램의 속도문제가 그리 중요한 것이 아니고, 네트웍 시대에는 여러 이기종의 기계에서 실행될 수 있는 프로그램의 특성이 더욱 중요시 되기에, 자바가 미래지향적인 언어라는 것을 알 수 있다.

강력한 네트웍 기능

또한 자바는 네트웍을 이동해서 실행될 수 있는 장점이 있는데, 다음 그림을 보자.



<그림 3> 자바 프로그램 실행 환경

컴파일된 코드를 바이트 코드라고 지칭하는 데, 이런 코드는 네트웍을 통해 이동될 수 있다. 웹 브라우저가 네트웍을 통해 이동된 바이트 코드를 실행해 주는 컨테이너(container) 역할을 한다. 물론 웹브라우저안에는 자바 인터프리터가 내장이 되어 있어서, HTML문서 안에 삽입되어 있는 바이트코드를 실행할 수 있게 된다.

위의 특징들을 볼 때, 자바언어는 인터넷의 웹환경에서 동작하는 어플리케이션을 만들 때 적합하다.

프로그래밍 개발 환경

자바언어로 프로그래밍을 할려면 다음 두 소프트웨어 중 한가지는 가지고 있어야 한다.

1) JDK(Java Development Kit)

자바언어를 발표한 선사가 제공하는 툴로서, 자바언어로 작성된 소스를 컴파일하고 실행시킬 수 있는 각종 실행화일을 담고 있다. 자바언어는 현재 완성된 단계가 아니고, 계속 새로운 특성과 클래스 라이브러리 등을 첨가하며 변화해 나가고 있다. 이런 변화된 자바언어로 프로그래밍을 하려면, 그때마다 새롭게 발표된 JDK를 다음 URL을 통해 구해야 한다. 현재 JDK는 버전 1.1 베타까지 나온 상태이다.

. 외국

http://www.javasoft.com/

. 국내

ftp://ftp.javasoft.com/pub/www/java
ftp://ftp.javasoft.com/pub/java

JDK는 다음과 같은 실행화일들로 구성이 되어 있다.

* java : 자바 인터프리터로서, 바이트코드를 인터프리트하여 실행시킨다.
* javac : 자바 컴파일러로서, 자바소스코드를 컴파일하여 바이트 코드를 생성시킨다 . 자바소스코드는 확장자가 java로 끝나야 하고, 바이트코드 화일은 확장자가 class로 생성이 된다.
* appletviewer : HTML내에 삽입되어 있는 바이트코드를 실행시켜 준다.
* jdb : 자바프로그램을 디버깅한다.
* javap : 바이트코드를 통해 소스코드의 속성과 메쏘드를 유추한다.
* javah : 자바프로그램과 C언어 코드를 연결시켜 준다.
* javadoc : 자바소스코드로부터 직접 HTML형태의 매뉴얼을 생성시킨다.

2) 통합개발 환경

JDK는 운영체제의 프람프트(prompt)에서 명령어를 입력하여 동작하는, 아주 기초단계의 개발환경이다. 크고 복잡한 프로그램을 개발할 때는 이런 개발환경보다, 비주얼한 통합개발환경이 훨씬 중요하다. 많은 통합개발환경중에서 다음 제품들이 널리 사용되고 있다.

. 마이크로소프트사의 Visual J++
. 시멘텍사의 Visual Cafe
. 볼랜드사의 Latte
. 자바소프트사의 Java Workshop

JDK 설치

1) Windows 95

관련 FTP사이트에서 실행화일을 가져온다. 확장자가 exe로 끝나는 파일을 하드디스크의 특정디렉토리에 위치시킨다음, MSDOS프람프트상에서 실행시킨다. 그러면 특정디렉토리 바로 아래에 java라는 하위 디렉토리가 생성된다.

2) Unix

관련 FTP사이트에서 gz이나 tar로 묶여 있는 파일을 가져온다. tar를 풀면 java라는 디렉토리가 생성이 된다.

기본환경 셋팅

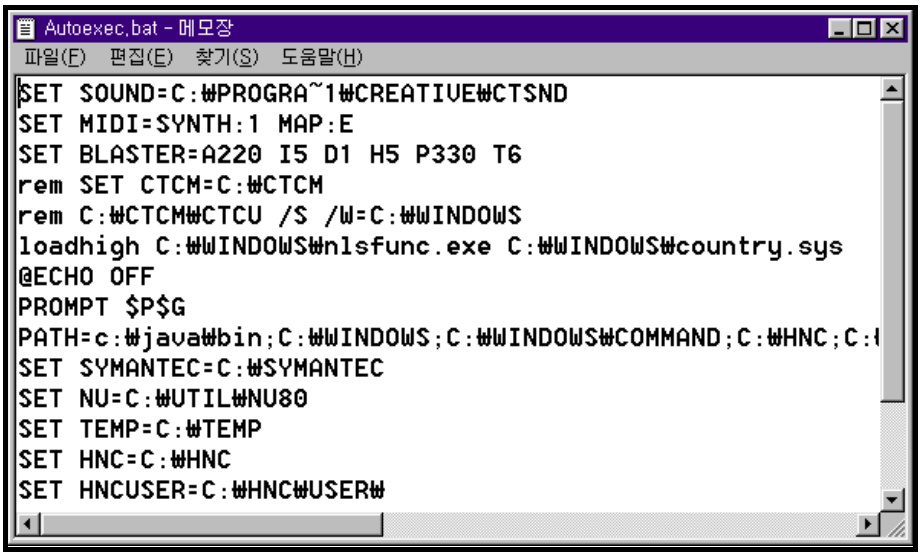
JDK를 설치했으면, 환경변수를 셋팅한다. 가장 기본적인 셋팅은 path라는 환경변수에 java의 하위디렉토리인 bin을 첨가하는 것이다. 이 디렉토리에는 JDK의 실행화일들이 들어 있다.

1) Windows 965

autoexec.bat의 path를 "java의 절대 패스\bin"을 첨가한다

예) path=c:\java\bin;c:\windows.

다음 그림을 보자.



<그림 4> autoexec.bat의 샘플 내용

2) Unix

".cshrc"파일의 path환경변수에 "java의 절대 패스/bin"을 첨가시킨다.

예) set PATH /usr/local/java/bin:/usr/local/bin

2. 기본 문법

애플릿과 어플리케이션

모든 자바프로그램은 다음과 같은 2가지 종류로 분류할 수 있다. 다음 용어가 계속 나올것이니 확실히 기억하길 바란다.

* 어플리케이션(Application)

브라우저와 상관없이 JDK안에 있는 자바인터프리터를 통해 실행되는,
독립적인(standalone) 자바프로그램

* 애플릿(Applet)

HTML문서안에 포함되어서, 인터넷을 통해 자바를 실행시킬 수 있는 브라우저(현재 netscape 2.0이상, Internet Explorer 3.0이상)안에서 실행되는 자바프로그램

웹상에서 동작하는 자바프로그램을 만들기 위해서는 애플릿을 만들어야 한다. 그렇지만 클라이언트/서버 프로그램일 경우, 클라이언트쪽엔 애플릿, 서버쪽엔 어플리케이션이 서로 통신하며 동작하는 경우가 많다.

애플릿과 어플리케이션을 만드는 방법은 소스 프로그램코드에서 차이가 난다.

1) 어플리케이션

어플리케이션은 클래스 안에 반드시 "public static void main(String argv[]) { } "라는 메쏘드를 정의하고 그 안에 원하는 코드를 써서 작성하면 된다.

다음은 어플리케이션을 만들 때 자주 사용되는 프로그램 구조이다.

```
[import 사용하길 원하는 패키지 또는 클래스이름;]

/* 메인_클래스.java라는 파일로 소스코드를 작성한다 */
public class 메인_클래스 {
        public static void main(String argv[]) {
        ...
        }
}
```

<어플리케이션 코드 구조>

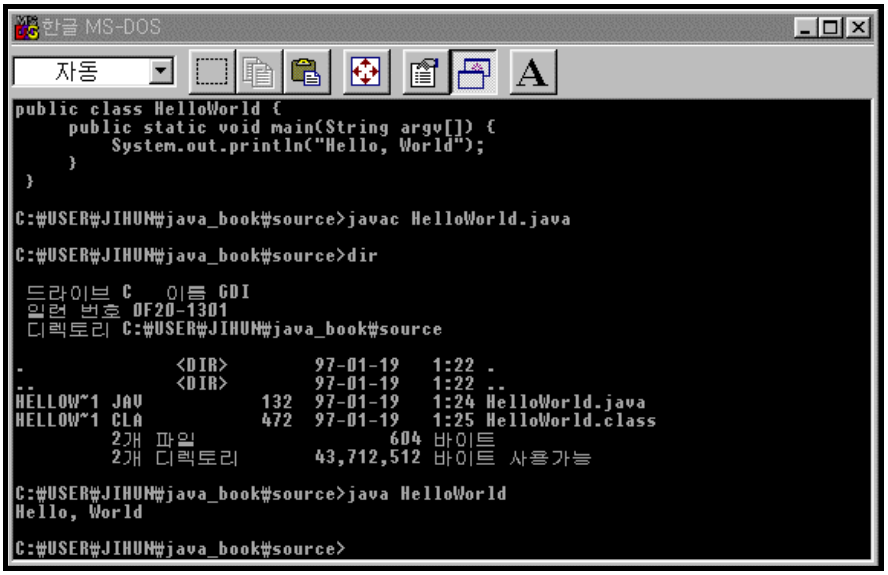
화면에 "Hello, World"를 출력하는 어플리케이션을 작성하여 보자.

```
public class HelloWorld {
    public static void main(String argv[]) {
      System.out.println("Hello, World");
    }
 }
```

그 과정은 다음과 같다.

1) 먼저 main()이 있는 클래스의 이름으로 소스크드를 작성하여, HelloWorld.java라는 파일을 생성한다.
2) "javac HelloWorld.java"명령을 내려서 컴파일한다. 에러가 없으면 HelloWorld.class라는 바이트 코드가 생성된다.

3) "java HelloWorld"라는 명령으로, 바이트코드를 실행한다. 그러면 <그림 5> 처럼 화면에 "Hello, World"라는 문자열이 출력된다.



<그림 5> 어플리케이션 실행모습

2) 애플릿

애플릿은 반드시 java.applet.Applet 클래스를 상속받아서 클래스를 작성하여야 한다. 또한 init(), start(), stop(), paint() 등의 메쏘드에 원하는 코드를 작성해야 한다.

프로그램 코드 구조는 다음과 같다.

```
import java.applet.Applet;
public class Example extends Applet{
  public void start(){... }
  ...
}
```

<애플릿 코드 구조>

화면에 "Hello, World"를 출력하는 어플리케이션과 같은 기능을 하지만, 웹 브라우저에 문자열을 출력하는 애플릿을 만들어 보자.

```
import java.awt.Graphics;
        public class HelloWorld2 extends java.applet.Applet {
        public void init() {
                resize(150, 25);
        }
        public void paint(Graphics g) {
                g.drawString("Hello world!", 50, 25);
        }
```

그 과정은 다음과 같다.

1) 먼저 클래스의 이름과 같은 HelloWorld2.java라는 파일을 작성한다.
2) "javac HelloWorld2.java"명령을 내려서 컴파일한다. 에러가 없으면 HelloWorld2.class라는 바이트 코드가 생성된다.(그 바이트 코드를 애플릿이라고 한다.)
3) 다음과 같은 내용의 "example.html" 파일을 만든다.

```
<html><head><title>A simple program</title></head>
<body><applet code="HelloWorld.class" width=150 height=25></applet>
</body></html>
```

4) 웹 브라우저가 없을 때는 "appletviewer example.html" 라는 명령으로 실행하거나, 있을 경우는 브라우저로 example.html을 읽어 들인다. 그 결과는 <그림 6>과 같다.

<그림 6> 애플릿 실행 모습

3) 애플릿 & 어플리케이션

한 프로그램이 애플릿과 어플리케이션의 역할을 동시에 할 수 있을까?

물론이다. 어플리케이션은 "public static void main()"함수만 정의하면 되기 때문에, 애플릿 내에 이 함수를 정의하면 웹 브라우저와 일반 명령라인에서 동시에 실행될 수

있게 된다.

위의 어플리케이션과 애플릿의 예제를 한꺼번에 합친 다음 예제를 보자.

```
import java.awt.Graphics;

public class HelloMerge extends java.applet.Applet {
        public static void main(String argv[]) {
        System.out.println("Hello, World");
    }

        public void init() {
                resize(150, 25);
        }
        public void paint(Graphics g) {
                g.drawString("Hello world!", 50, 25);
        }
}
```

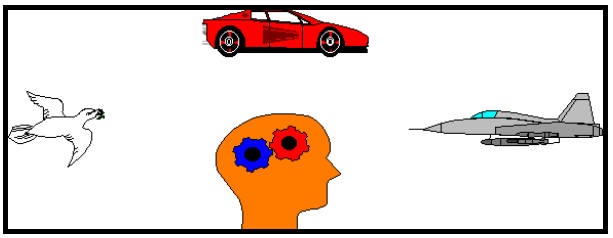
위의 소스를 보면, 애플릿 코드에 "public static void main()" 을 첨가한 것 밖에 다른 예제 코드와 별로 차이가 없음을 알 수 있다. 이런 코드는 자신이 만든 자바프로그램이 웹 브라우저와 일반 명령라인, 두 환경에서 동시에 동작하는 프로그램을 만들 때 자주 사용된다.

2부 객체지향 프로그래밍

1. 객체지향 기본 개념

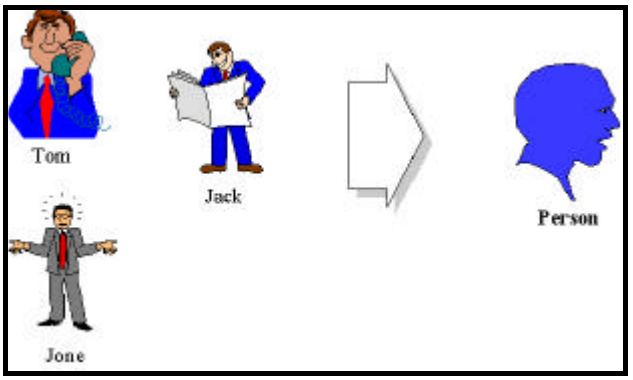
객체지향언어로 프로그래밍할 때는, 프로그램을 객체들이 서로 메시지를 전달하여 서비스를 실행하도록 구성해야 한다.

객체들은 <그림 1>처럼 문제영역에서 의미있는 사물(thing), 장소(place), 개념(concept)등이다. 예를 들어 자동차, 비행기, 인간, 새, 자전거, 호텔 등이 그것이다.



<그림 1> 객체들

이런 객체들을 생성시키기 위한 틀이 <그림 2>과 같은 클래스이다. 쉽게 생각하면 클래스는 붕어빵을 만드는 틀, 객체는 붕어빵이다.



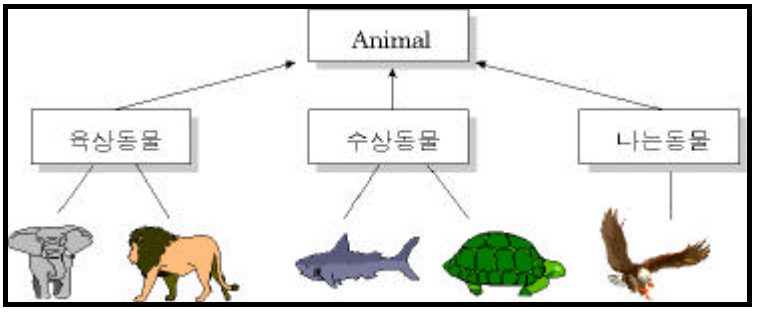
<그림 2> 사람 클래스

클래스는 자료(속성)와 그 자료를 처리하는 연산(메쏘드)을 하나의 단위로 구성한 것이다.

클래스는 자료보다 오히려 메쏘드가 더 중요한데, 그것은 메쏘드가 객체가 할 수 있는 서비스를 의미하기 때문이다. 프로그래머 입장에서는 객체가 어떤 서비스를 하느냐가 중요하지, 어떤 데이터를 가지고 있는지는 별로 중요하지 않기 때문이다.

메쏘드는 주로 객체의 자료를 캡슐화하여 외부로부터 보호하며, 메쏘드를 통해야만 객체의 자료를 변경할 수 있도록 해야 한다. 즉 외부에서 직접 객체의 속성을 변경하는 것은 바람직하지 않다.

객체지향언어에서는 기존언어에서 제공하지 않은 <그림 3>과 같은 "상속"이라는 개념을 제공하고 있다. 만약 특정 클래스를 상속하면, 그 클래스안에 있는 속성과 메쏘드를 그대로 물려 받아 사용할 수 있어서, 코드 재사용에 큰 도움이 된다.



<그림 3> 상속

2. 폴리모피즘과 다이나믹 바인딩

객체지향언어로 제대로 프로그래밍을 하려면, 폴리모피즘과 다이나믹 바인딩을 최대한 이용해야 한다. 이것을 제대로 하면, 프로그램의 확장과 유지보수를 쉽게 할 수 있다.

폴리모피즘은 특정 객체들에게 똑같은 이름의 메쏘드를 호출시킬 때, 각자 자신만의 코드를 실행되는 메커니즘이다.

다이나믹 바인딩은 객체가 컴파일시에 바인딩되는 것이 아니라, 실행시에 바인딩이 된다는 것이다.

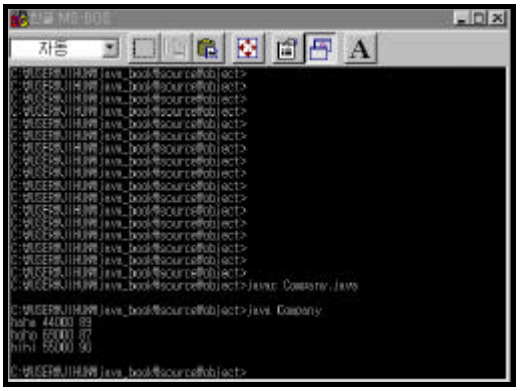
다음 예를 보면서 이해해 보도록 하자. 이 어플리케이션은 회사원의 봉급을 올릴 때, 직책에 따라서 봉급인상정도를 다르게 하고, 화면에 이름과 봉급을 출력한다. 그리고 회사원은 일반회사원과 매니저로 분류가 된다.

```
import java.util.*;

public class Company {
        public static void main(String[] args) {
                Employee[] worker = new Employee[3];

                worker[0] = new Employee("haha", 40000, new Date(89,9,1));
                /* Manager클래스 타입객체가 Employee클래스 타입으로 생성된다.-> 다이나
                worker[1] = new Manager("hoho", 60000, new Date(87,11,1));
                worker[2] = new Employee("hihi", 50000, new Date(90,9,10));

                int i;
                /* worker[1].raiseSalary(10)에서 실제로 호출되는 코드는 Manager클래스에 정                의
                for ( i=0; i<worker.length;i++) worker[i].raiseSalary(10);
                for ( i=0; i<worker.length;i++) worker[i].print();
        }
}
```

```
class Employee {
        private String name;
        private double salary;
        private Date hireDate;

        public Employee(String n, double s, Date d)
        {
                name = n;
                salary = s;
                hireDate = d;
        }

        public void print()
        {
                System.out.println(name + " " + salary + " " + hireYear());
        }

        public void raiseSalary(double byPercent) {
                salary *= 1 + byPercent/100;
        }

        public int hireYear()
        {
                return hireDate.getYear();
        }

}

/* Employee로부터 상속받았기 때문에, Manager는 Employee라고 말할 수 있다. */
class Manager extends Employee
{
        public Manager(String n, double s, Date d)
        {
                super(n,s,d);
        }

        /* Employee에 있는 raiseSalary를 오버라이딩하였다 */
        public void raiseSalary(double byPercent)
        {
                Date toDate = new Date();
                double bonus = 0.5*(toDate.getYear() - hireYear());
                super.raiseSalary(byPercent+bonus);
        }
}
```

<예제> Company.java

이 예제를 실행시키면, 다음 <그림 >과 같은 결과를 얻게 된다.



<그림 4> 봉급 인상 결과

위의 예제에서 중요한 코드는

```
worker[1] = new Manager("hoho", 60000, new Date(87,11,1));
```

이다. worker가 Employee클래스 타입의 객체인데, Manager타입객체가 할당될 수 있는가? 될 수 있다. 일반적으로 상위 클래스객체는 상위클래스를 상속받은 하위클래스 객체를 할당받을 수 있다. Manager가 Employee 클래스로부터 상속을 받았기 때문에, 당연히 할당이 된다. 이것이 폴리모피즘의 출발점이다. 그럼 다음 코드를 유심히 보자.

```
for ( i=0; i<worker.length;i++) worker[i].raiseSalary(10);
```

에서 worker[1].raiseSalary(10)은 Employee 의 raiseSalary()를 호출할 것인가? Manager의 raiseSalary를 호출할 것인가?

언듯보면, worker가 Employee클래스의 객체이기 때문에 Employee의 그것을 호출할 것이라고 생각하기 쉽다. 하지만 worker[1]객체에 실제로 할당된 객체의 타입은 Manager클래스 타입이기 때문에 Manager의 그것이 호출된다.

바로 이것이 폴리모피즘이다. worker[0], woker[1], worker[2] 모두에게 똑같이 raiseSalary()를 호출했는데, worker[0],worker[2]는 Employee의 그것을, worker[1]은 Manager의 그것을 호출하였다. 즉 각자 자신의 타입에 정의되어 있는 메쏘드의 내용을 실행한 것이다.

이런 폴리모피즘은 프로그램을 확장할 때, 아주 편리하다.

만약에 회사에 임시직이란 새로운 회사원이 등장하고, 봉급인상률이 다른 사람들과 다르다고 하자. 그러면 임시직이란 클래스를 생성하고, worker배열의 원소를 더 늘려서 그 객체를 생성해야 할 것이다. 위의 코드를 수정하면 대략 다음과 같이 될 것이다.

```
        Employee[] worker = new Employee[4];
        ........
        worker[3] = new PartTimer("kiki", 10000, new Date(96,1,1));
.........
        for ( i=0; i<worker.length;i++) worker[i].raiseSalary(10);
        for ( i=0; i<worker.length;i++) worker[i].print();

        public class PartTimer extends Employee {
        ....
                public void raiseSalary(double byPercent) {
                .... // 임시직만의 봉급인상 코드
                }
        }
```

C나 Pascal처럼 구조적 언어는 이렇게 깔끔하게 할 수 없다.

3부 멀티미디어 프로그래밍

멀티미디어 프로그램을 자바애플릿으로 구현하는 사례를 원리와 예제로 살펴볼 것이다. 현재 자바는 멀티미디어를 다룰수 있는 클래스 라이브러리를 충분히 가지고 있지 못한 형편이다. au형식의 사운드 파일과 정지 이미지(gif, jpeg등)를 위한 클래스 라이브러리밖에 없기 때문에, 그것들을 조합하거나 응용하는 수준에서 만족하여야 한다. 그렇지만 조만간 멀티미디어를 위한 클래스라이브러리(동화상, 3차원 영상등)들을 본격적으로 제공한다고 하니, 기대해 볼만한다.

현재로선 자바애플릿을 이용한 멀티미디어 구현은, au형식의 간단한 사운드 및 정지이미지를 이용한 2차원적인 애니메이션, 간단한 오락 정도이다.

이번 부에서는 애니메이션과 사운드를 구현하는 방법, 그 둘을 이용한 오락 프로그래밍 등을 간단한 사례를 들어 살펴볼 것이다.

1. 애니메이션

원리

애니메이션 애플릿의 원리는 영화상영의 그것과 같다. 움직이는 영상을 한 장면(프레임)씩 저장한 후, 그런 프레임들을 연속적으로 빨리 보여주면, 실제로 움직이는 효과를 낼 수 있는 것이다. 일반적으로 1초당 20-30프레임을 화면에 연속적으로 보여주면 된다. 이런 프로그램코드를 애니메이션 루프(loop)라고 말하는데, 다음과 같은 형식을 갖고 있다.

```
while( moreFrameToPaint) {
        paintCurrentFrame();
        sleep for some time; // 초당 프레임 수를 결정
        advanceToNextFrame();

}
```

애플릿내에 이 애니메이션 루프를 구현해야 하는데, 반드시 애니메이션 루프를 포함하는 쓰레드를 생성시켜야 한다. 왜냐하면 애니메이션 루프가 실행되고 있을 때, 사용자나 시스템으로부터의 이벤트를 처리할 수 있어야 되기 때문에 쓰레드를 만들어야 된다.

쉽게 말해서 애플릿이 동시에 2가지 일(애니메이션 루프를 돌려서 애니메이션을 사용자에게 보여주는 일, 외부의 이벤트를 처리하는 일)을 처리하기 위해서 Runnable이라는 인터페이스를 구현해야 한다. 즉 애플릿안의 run()메쏘드안에 애니메이션 루프를 집어 넣어서, 쓰레드를 생성하여 실행시켜야 한다.

그렇지않고 만약 애니메이션 루프를 paint()메쏘드안에 집어넣으면, 제대로 동작하지 못하

는 결과를 초래하니까 이 점 주의해야 한다.

일반적으로 애니메이션 애플릿을 만들기 위해 사용되는 자주 사용되는 프로그램 코드 구조는 다음과 같다.

```
import java.awt.*;

/* implements키워드를 사용하여, 본 애플릿이 쓰레드화될 것임을 선언한다*/
public class Example1Applet extends java.applet.Applet implements Runnable {
  int frame; // 현재 프레임을 상징한다.
  int delay; // 프레임간의 휴식시간
  Thread animator; // 애니메이션 루프를 실행하는 쓰레드

  /**
   * 애플릿이 웹 브라우저에 처음 도착했을 때 실행되는 메쏘드로서,
   * 주로 애플릿을 초기화하며 프레임간의 시간을 결정한다.
   */
  public void init() {
    String str = getParameter("fps"); // 초당 프레임 개수를 결정하기 위해 HTML문서로 부터, "fps“
라는 이름의 패러미터의 값을 읽어 들인다.
    int fps = (str != null) ? Integer.parseInt(str) : // 스트링값을 정수값으로 변환
    delay = (fps > 0) ?(1000 / fps) : 100; // milisecond로 바꾸기 위해, 1000/fps를 실행한다.
  }

  /**
   * 애니메이션 루프가 담겨져 있는 쓰레드를 생성하여, 실행시킨다.
  */
  public void start() {
    animator = new Thread(this); // 애플릿 자신을 독립적인 쓰레드로 생성한다.
    animator.start(); //쓰레드가 실행이 되는데, 애플릿안의 run()안의 메쏘드를 실행시킨다.
  }

  /**
   * 애니메이션 루프가 담겨져 있는 메쏘드로서, 가장 중요한 임무를 한다.
   * 프레임을 연속적으로 사용자에게 보여주는 역할을 한다.
   */
  public void run() {
    while (Thread.currentThread() == animator) { // 만약 쓰레드가 여러개일 때,
      // 이전 프레임을 지우고, 현재 프레임을 화면에 보여준다.
      repaint();

      // 다음 프레임까지 휴식한다.
      try {
        Thread.sleep(delay); // delay만큼 휴식한다.
      } catch (InterruptedException e) {
```

```
            break;
          }

          /* 다음 프레임이 무엇이 될지 결정한다. 여기선 frame++이지만, 실제로는 다음 프
               레임을 결정하는 코드를 작성해야 한다. */
          frame++;
        }
      }

      /**
       * 현재 HTML페이지를 떠날 때나 브라우저를 빠져나올 때, 호출된다.
       * 생성한 쓰레드를 중지시키는 코드를 삽입한다.
       */
      public void stop() {
        animator = null; // 애니메이션 루프를 실행하는 쓰레드를 중지시킨다.
      }

      /* public void paint(Graphics g), public void update(Graphics g) 등을 정의하여, 화면에 실질적으로
      무엇이, 어떻게 나타날지를 코딩한다. */

    }
```

<예제> 일반적인 애니메이션 애플릿 소스코드 구조

위의 예제는 실제로 실행가능한 형태가 아니라, 뼈대만을 가지고 있다. 그렇기 때문에 실제로 구현할 때는 뼈대위에 살을 붙여야 한다.

그럼 다음 예제를 보면서, 살을 붙여보자.

2. 이미지 파일 없이, 그래픽 기능을 이용한 애니메이션

사인 곡선들의 결합을 애니메이션 형태로 보여주는 애플릿을 통하여, 애니메이션의 기본테크닉을 연마한다. 여기에 나오는 예제들은 꼭 애니메이션뿐만 아니라 그래픽관련 프로그램을 작성할 때 필요한 기법들을 설명하기 때문에, 주의 깊게 원리를 이해해야 한다.

3가지 예를 들어, 최상의 테크닉을 단계적으로 밟아보자.

1) 깜박거리는 애니메이션

다음 자바 소스코드를 살펴보자. 위의 뼈대코드와 거의 차이가 없지만, 화면으로 내용을 보여주는 paint()메쏘드가 정의가 되어 있음을 알 수 있다. paint()메쏘드 안에는 사인곡선을 그리는 코드가 정의되어 있다.

```
import java.awt.*;
```

```
public class Example4Applet extends java.applet.Applet implements Runnable {
  int frame;
  int delay;
  Thread animator;

  /**
   * Initialize the applet and compute the delay between frames.
   */
  public void init() {
        String str = getParameter("fps");
        int fps = (str != null) ? Integer.parseInt(str) : 10;
        delay = (fps > 0) ? (1000 / fps) : 100;
  }

    public void start() {
        animator = new Thread(this);
        animator.start();
  }


  public void run() {
        long tm = System.currentTimeMillis(); // 현재 시간을 기록
        while (Thread.currentThread() == animator) {
          repaint();

          // delay만큼 정확한 시간을 쉬기 위해, 시간을 추적한다.
          try {
                tm += delay;
                Thread.sleep(Math.max(0, tm - System.currentTimeMillis()));
          } catch (InterruptedException e) {
                break;
          }

          // Advance the frame
          frame++;
        }
  }

  public void stop() {
        animator = null;
  }

  /**
   * 한 프레임을 화면에 출력하는 코드가 정의되어 있다. frame변수의 값에 따라 다른 형태의 *
사인곡선을 화면에 뿌려준다..
```

```
    */
   public void paint(Graphics g) {
            Dimension d = size(); // 애플릿의 사이즈 결정
            int h = d.height / 2;  // 애플릿의 높이를 2로 나눈다. 즉 가운데 위치 파악
            for (int x = 0 ; x < d.width ; x++) {  // 애플릿의 가로 만큼, 연달아 선을 그린다.
              int y1 = (int)((1.0 + Math.sin((x - frame) * 0.05)) * h);
              int y2 = (int)((1.0 + Math.sin((x + frame) * 0.07)) * h);
              g.drawLine(x, y1, x, y2);
            }
   }
}
```

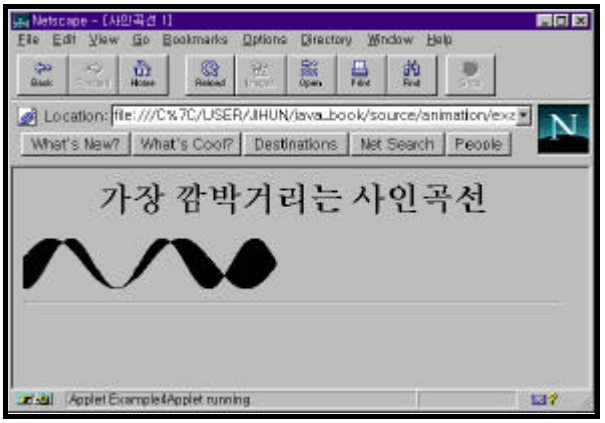
<예제> Example4Applet.java

위의 애플릿을 다음과 같은 HTML 문서를 작성하여 끼워 넣는다. 주의할 것은 HTML문서의 APPLET태그안에서 width, height의 값을, 애플릿내에서 애니메이션을 할 프레임의 가로너비와 높이값과 일치시켜야 한다는 것이다.

```
<TITLE>사인곡선 1</TITLE>

<center>
<H1> 가장 깜박거리는 사인곡선</H1>
</center>

<APPLET code=Example4Applet.class width=500 height=50>
<param name=fps value=10>
</applet>

<HR>
```

<예제> Example4Applet.html

위의 HTML문서를 웹브라우저에 읽어 들이면 <그림 1>과 같은 결과를 얻는다.



<그림 1> 사인곡선 1

실제로 독자가 실행시켜보면, 굉장히 깜박거림을 느낄 수가 있을 것이다. 그렇다면 왜 이렇게 깜박거리는가?

그 이유는 repaint()의 기능때문이다.

자! run()메쏘드 안의 while루프를 주시하자. 이 안에는 repaint()가 있는데, 이것은 애플릿의 전체 배경화면을 지우고 paint()를 호출한다. 즉 화면에 표현된 현재 프레임을 완전히 지우고, 다음 프레임을 paint()의 실행을 통해 화면에 표현하는 것이다. 그 다음 약간의 휴식시간(Thread.sleep(..)) 후에, 다음 프레임을 결정(frame++)하고 또 다시 repaint()를 호출하는, 이런 단계를 계속 반복함으로서 사인곡선이 움직이는 것처럼 보이는 것이다. 잘 이해가 가지 않는다면, 위의 run()메쏘드안을 머리속으로 차례차례 실행시켜보기 바란다.

목적인 애니메이션은 구현했지만 ,배경화면이 완전히 지워지는 과정에서 어쩔 수 없는 깜박거림 현상이 나타나게 된다.

이런 현상을 없애기 위한 아이디어는 무엇일까? 첫 번째는 현재 프레임을 모두 지우지 말고 필요한 부분만 지운 후에 다음 프레임을 표현하자는 거다.

2) 덜 깜박거리는 애니메이션

깜박거림을 없애기 위한 단서는 repaint()에 있다. repaint()의 동작과정을 더욱 자세히 보자.

repaint()는 먼저 update()를 호출한다. update()는 애플릿의 배경을 지운뒤에 paint() 를 호출한다. update()가 실제로 애플릿의 배경을 지우기 때문에, 이 update()를 애플릿내에서 재정의, 즉 오버라이트 함으로써 애플릿의 배경을 다 지우지 않게 할 수 있다.

다음 소스코드를 보자.

```
import java.awt.*;

public class Example5Applet extends java.applet.Applet implements Runnable {
   int frame;
   int delay;
   Thread animator;

   public void init() {
           String str = getParameter("fps");
           int fps = (str != null) ? Integer.parseInt(str) : 10;
           delay = (fps > 0) ? (1000 / fps) : 100;
   }

   public void start() {
           animator = new Thread(this);
           animator.start();
```

```
    }

    public void run() {
        long tm = System.currentTimeMillis();
        while (Thread.currentThread() == animator) {
          repaint();

          try {
              tm += delay;
              Thread.sleep(Math.max(0, tm - System.currentTimeMillis()));
          } catch (InterruptedException e) {
              break;
          }

          frame++;
        }
    }

    public void stop() {
        animator = null;
    }

    /**
     * 실제로 프레임을 화면에 출력하는 코드는 update()에 있으므로, update()를 호출한다.
     */
    public void paint(Graphics g) {
        update(g);
    }

    /**
     * update()를 재정의한다. 이 안에 현재 프레임을 부분적으로 지우고, 새로운 프레임을 화면
     * 에 표현하는 코드를 집어 넣는다. 즉 현재 사인곡선을 부분적으로 지우고, 동시에 새로운
     * 사인곡선을 그린다.
     */
    public void update(Graphics g) {
        Color bg = getBackground(); // 애플릿의 배경색을 얻는다.
        Dimension d = size();
        int h = d.height / 2;
        for (int x = 0 ; x < d.width ; x++) {
          int y1 = (int)((1.0 + Math.sin((x - frame) * 0.05)) * h);
          int y2 = (int)((1.0 + Math.sin((x + frame) * 0.07)) * h);

          if (y1 > y2) {  // 사인곡선의 지울부분을 결정한다.
              int t = y1;
              y1 = y2;
              y2 = t;
```

```
            }
            g.setColor(bg); // 애플릿의 배경색으로 셋팅
            g.drawLine(x, 0, x, y1); // 애플릿의 배경색으로 선을 긋는다.
            g.drawLine(x, y2, x, d.height); // 이것은 기존의 사인곡선을 지우는 기능을 한다.
            g.setColor(Color.black); // 검정색으로 셋팅
            g.drawLine(x, y1, x, y2); // 새로운 사인곡선을 그리기 위해, 선을 긋는다.
        }
    }
}
```

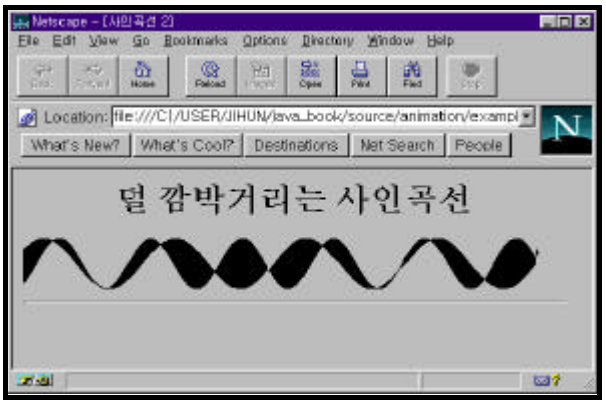
<예제 > Example5Applet.java

위의 애플릿을 다음과 같은 HTML 문서를 작성하여 끼워 넣는다.

```
<TITLE>사인곡선 2</TITLE>

<center>
<H1> 덜 깜박거리는 사인곡선</H1>
</center>

<APPLET code=Example5Applet.class width=500 height=50>
<param name=fps value=10>
</applet>

<HR>
```

<예제 > example5applet.html

위의 HTML문서를 웹브라우저에 읽어 들이면 <그림 2>와 같은 결과를 얻는다.



<그림 2> 사인곡선 2

사인 곡선 1과 비교하면 이것은 덜 깜박거리면서, 왼쪽에서 오른쪽으로 물결치듯이 사인 곡선이 움직이는 것을 보게 될 것이다. 왜 그런가?

while문의 루프를 차례차례 실행시켜보자. 먼저 처음에 repaint()를 실행시킨다. 이 때 repaint()는 update()를 호출시킨다. 만약 애플릿안에 update()가 없으면, 이미 정해져 있는 애플릿의 상위 클래스인 Component클래스안에 있는 update()를 호출하여 애플릿안의 배경을 다 지웠을 것이다. 그렇지만 update()를 새롭게 오버라이드하여 정의했기 때문에, 정의한 update()를 실행시킨다. update()는 기존의 사인곡선 프레임의 일부분을 왼쪽부터 차례차례 지워나가면서, 새로운 사인곡선 프레임을 화면에 표현한다. 그 후에는 약간의 휴식시간(Thread.sleep(..))후에 다음 프레임(frame++)을 결정하고 다시 repaint()를 실행하며, 계속 while문을 반복하게 된다.

사인곡선 1보다는 부드러워졌지만, 여전히 불만스럽다. 더 부드럽게 할 수 없을까?

마지막 해법이 하나 있다.

3) 가장 부드러운 애니메이션

더블 버퍼링기법을 이용해서, 이전 방법과는 비교도 되지 않을 정도로 부드러운 애니메이션을 만들 수 있다. 이 방법은 다음 순서를 반복하여, 애니메이션을 가능케 한다.

1) 오프 스크린(off screen) 이미지를 생성하고, 그 안에 프레임을 그리기 위한 그래픽 객체를 얻는다.
2) 그래픽 객체를 이용하여, 오프 스크린 이미지안에 한 프레임을 그린다.
3) 오프스크린 이미지, 즉 프레임을 화면에 표현한다.

오프 스크린 이미지는 화면에 표현된 이미지가 아니라, 화면에 표현되기 전에 메모리안에 저장되어 있는 이미지를 의미한다.

이전 방법들은 프레임들을 메모리안에 저장하지 않고, 곧장 paint()나 update() 등을 이용하여 화면에 출력했다. 그래서 프레임을 만들 때 걸리는 시간과 메모리 캐쉬의 기능을 살리지 못했기에 깜박거리는 현상을 가져왔다.

그러나 프레임들을 곧장 화면에 표현하지 않고, 먼저 메모리안에 저장한 후에 화면에 표현하는 더블 버퍼링 방법은 다음 이유 때문에 더욱 브드러운 효과를 낼 수 있다.

1) 대부분의 컴퓨터들은 메모리안의 이미지를 효율적으로 화면에 표현하는 방법을 가지고 있다.
2) CPU에서 비디오 메모리는 캐쉬(cache)가 될 수 없지만, 메모리안에 있는 이미지는 캐쉬가 될 수 있다.

더블 버퍼링 기법을 사용한 사인곡선의 예제 코드를 보자.

```
import java.awt.*;

public class Example6Applet extends java.applet.Applet implements Runnable {
   int frame;
   int delay;
   Thread animator;
```

```
Dimension offDimension; // 오프 스크린 이미지의 너비, 높이 등의 정보를 저장한 객체
Image offImage; // 오프스크린 이미지를 저장할 Image객체
Graphics offGraphics; // 오프스크린 이미지에 프레임을 그리기 위한 Graphics 객체

public void init() {
        String str = getParameter("fps");
        int fps = (str != null) ? Integer.parseInt(str) : 10;
        delay = (fps > 0) ? (1000 / fps) : 100;
}

public void start() {
        animator = new Thread(this);
        animator.start();
}

public void run() {
        long tm = System.currentTimeMillis();
        while (Thread.currentThread() == animator) {
          repaint();

          try {
                tm += delay;
                Thread.sleep(Math.max(0, tm - System.currentTimeMillis()));
          } catch (InterruptedException e) {
                break;
          }

                frame++;
        }
}

public void stop() {
        animator = null;
        offImage = null;
        offGraphics = null;
}

/**
 * 더블버퍼링을 사용할 때, update()안의 코드는 다음과 같은 일을 한다.
 * 1. 오프스크린을 상징하는 Image객체와 관련 Graphics 객체를 생성한다.
 * 2. 오프스크린 내의 이전 프레임을 지운다.
 * 3. 오프스크린 내에 새로운 프레임을 그린다.
 * 4. 오프스크린을 화면에 표현한다.
 */
public void update(Graphics g) {
```

```
        Dimension d = size();

        // 오프 스크린 Image객체와 관련 Graphics객체를 생성한다.
        if ((offGraphics == null)
        || (d.width != offDimension.width)
        || (d.height != offDimension.height)) {
          offDimension = d;
          offImage = createImage(d.width, d.height); // 애플릿의 크기와 같은 이미지 생성
          offGraphics = offImage.getGraphics();
        }

        // 오프스크린 이미지안에 있는 이전 프레임을 지운다.
        offGraphics.setColor(getBackground()); // 애플릿의 배경색으로 오프스크린 색깔 셋팅
        offGraphics.fillRect(0, 0, d.width, d.height);// 오프스크린 이미지안의 이전프레임을지운다.
        offGraphics.setColor(Color.black); // 다시 검정색으로 오프스크린 색깔 셋팅

        // 오프스크린 이미지안으로 프레임을 그린다.
        paintFrame(offGraphics);

        // 오프스크린 이미지를 화면에 표현하다.
        g.drawImage(offImage, 0, 0, null);
    }

    /**
     * 만약 paint()를 호출하는 이벤트가 발생하면(처음 애플릿 도착, 브라우저의 크기 변화, 애 *
플릿 위에 다른 윈도우가 겹칠 시), 현재 오프스크린 이미지를 화면에 표현한다.
     */
    public void paint(Graphics g) {
        if (offImage != null) {
          g.drawImage(offImage, 0, 0, null);
        }
    }

    /**
     * 프레임을 오프스크린 이미지안에 그린다. 즉 사인곡선을 계산하여 그려넣는다.
     */
    public void paintFrame(Graphics g) {
        Dimension d = size();
        int h = d.height / 2;
        for (int x = 0 ; x < d.width ; x++) {
          int y1 = (int)((1.0 + Math.sin((x - frame) * 0.05)) * h);
          int y2 = (int)((1.0 + Math.sin((x + frame) * 0.07)) * h);
          g.drawLine(x, y1, x, y2);
        }
    }
}
```

<예제> Example6Applet.java

위의 애플릿을 다음과 같은 HTML 문서를 작성하여 끼워 넣는다.

```
<TITLE>사인곡선 3</TITLE>

<center>
<H1> 가장 부드러운 사인곡선</H1>
</center>

<APPLET code=Example6Applet.class width=500 height=50>
<param name=fps value=10>
</applet>

<HR>
```

<예제 > Example6Applet.html

위의 Example5Applet.java와 비교하며 소스를 분석하면, 차이가 있는 것은 update()의 내용이 바뀌었다는 것과 paintFrame()이 추가되었다는 것이다.

update()의 코드를 자세히 보자.

offImage = createImage(d.width, d.height); 는 애플릿의 너비와 높이가 똑같은 오프스크린 Image객체를 생성시키는 문장이다. 메모리안에 프레임이 담겨질 공간을 확보하는 의미가 된다. 중요한 것은 이 안에 프레임을 그려 넣기 위해서는 Image객체와 관련있는 Graphics객체를 생성해야 한다.

offGraphics = offImage.getGraphics(); 는 offImage라는 오프스크린 Image객체에 프레임을 그려넣기 위해, offGraphics 라는 Graphics 객체를 생성하는 문장이다. Image클래스의 getGraphics()메쏘드는 그 이미지안에 도형 및 이미지 등을 그려넣기 위한 Graphics 객체를 생성하는 기능을 한다.

offGraphics는 오프스크린 Image객체인 offImage와 연관된 Graphics클래스의 객체이기 때문에, 그 클래스의 각종 메쏘드를 통해 도형 및 이미지 등을 오프스크린 이미지인 offImage 안에 그려넣을 수 있다.

```
offGraphics.setColor(getBackground());
 offGraphics.fillRect(0, 0, d.width, d.height);
 offGraphics.setColor(Color.black);
```

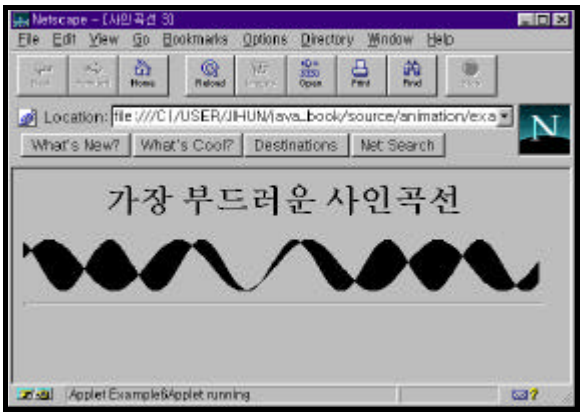
이들 문장들은 오프스크린 이미지 offImage안을 애플릿의 배경색깔로 채워넣는 기능을 한다.

그 다음 paintFrame()이라는 사용자가 정의한 메쏘드를 호출하여, 오프스크린 이미지인 offImage안에 사인곡선을 그려넣는다. 패러미터로 offImage와 연관된 offGraphics 객체가 넘어

간다는 사실을 주의해야 한다.

g.drawImage(offImage, 0, 0, null); 는 메모리안에 있는 오프스크린 이미지인 offImage의 탑 코너(top corner, 사각형의 왼쪽 맨 위 좌표)를 애플릿의 (0, 0)좌표에 표현하는 문장이다. 여기서 g는 애플릿의 화면을 나타내는 Graphics 객체이기 때문에, drawImage(Image img, int x, int y, ImageObserver observer)는 웹 브라우저 화면에 이미지를 표현하게 된다.

위의 HTML문서를 웹브라우저에 읽어 들이면 <그림 3>과 같은 결과를 얻는다.



<그림 3> 사인곡선 3

독자가 애니메이션 애플릿을 만들때는, 더블 버퍼링을 기법을 사용할 것을 권한다. 그리고 위의 소스코드를 틀로 삼아서 update()나 paintFrame()의 내용만을 약간 수정하여 사용하면 자신만의 애플릿을 쉽게 구현할 수 있을 것이다.

앞으로 나오는 모든 예제들은 더블 버퍼링 기법을 사용하고 있으니, 참고 바란다.

배경이 고정된 채, 특정 이미지만 움직이는 애니메이션

이전 예제들은 gif나 jpeg형식의 이미지 파일을 이용하지 않고, 도형이나 선을 이용한 애니메이션 애플릿이다. 하지만 실제로는 이미지 파일을 이용한 애니메이션 애플릿이 많이 사용되는 편이다.

다음 예제는 지구가 배경화면으로 있고, 자동차가 그 위를 지나가는 애니메이션 애플릿이다. 본 애플릿을 위해 <그림 4>, <그림 5>와 같은 두 개의 그림 파일이 필요하다.

<그림 4> car.gif

<그림 5> world.gif

소스는 다음과 같다.

```
import java.awt.*;

public class Example7Applet extends java.applet.Applet implements Runnable {
   int frame;
   int delay;
   Thread animator;

   Dimension offDimension;
   Image offImage;
   Graphics offGraphics;

   Image world; // 지구 그림을 위한 Image객체 선언
   Image car; // 차 그림을 위한 Image객체 선언

   /* 이미지 파일을 서버로부터 가져온다 */
   public void init() {
           String str = getParameter("fps");
           int fps = (str != null) ? Integer.parseInt(str) : 10;
           delay = (fps > 0) ? (1000 / fps) : 100;

           world = getImage(getCodeBase(), "world.gif"); // 서버로부터, world.gif를 가져옴
           car = getImage(getCodeBase(), "car.gif"); // 서버로부터, car.gif를 가져옴
   }

   public void start() {
           animator = new Thread(this);
           animator.start();
   }

   public void run() {
           long tm = System.currentTimeMillis();
           while (Thread.currentThread() == animator) {
             repaint();

             try {
                     tm += delay;
                     Thread.sleep(Math.max(0, tm - System.currentTimeMillis()));
             } catch (InterruptedException e) {
```

```
                    break;
                }

                frame++;
            }
        }

        public void stop() {
            animator = null;
            offImage = null;
            offGraphics = null;
        }

        public void update(Graphics g) {
            Dimension d = size();
            if ((offGraphics == null)
             || (d.width != offDimension.width)
             || (d.height != offDimension.height)) {
                offDimension = d;
                offImage = createImage(d.width, d.height);
                offGraphics = offImage.getGraphics();
            }

            offGraphics.setColor(getBackground());
            offGraphics.fillRect(0, 0, d.width, d.height);
            offGraphics.setColor(Color.black);

            paintFrame(offGraphics);

            g.drawImage(offImage, 0, 0, null);
        }

        public void paint(Graphics g) {
            update(g);
        }

        /**
        * 오프스크린 이미지안에 지구위에 차 2대가 있는 한 프레임을 그린다. 여기서 패러미터로
        * 넘어가는 Graphics객체 g는 오프스크린 이미지와 연관된 객체이다.
        */
        public void paintFrame(Graphics g) {
            Dimension d = size();
            int w = world.getWidth(this); // 지구그림의 너비
            int h = world.getHeight(this); // 지구그림의 높이

            if ((w > 0) && (h > 0)) { // 만약 지구그림을 서버로부터 가져왔으면,
```

```
            g.drawImage(world, (d.width- w)/2, (d.height - h)/2, this);
          }

          w = car.getWidth(this);
          h = car.getHeight(this);

          if ((w > 0) && (h > 0)) { // 만약 차 그림을 서버로부터 가져 왔으면,
            w += d.width;
            // 첫 번째 차를 지구위에 그린다.
            g.drawImage(car, d.width - ((frame * 5) % w), (d.height - h)/3, this);
            // 두 번째 차를 다른 위치로 지구위에 그린다.
            g.drawImage(car, d.width - ((frame * 7) % w), (d.height - h)/2, this);
          }
    }
}
```

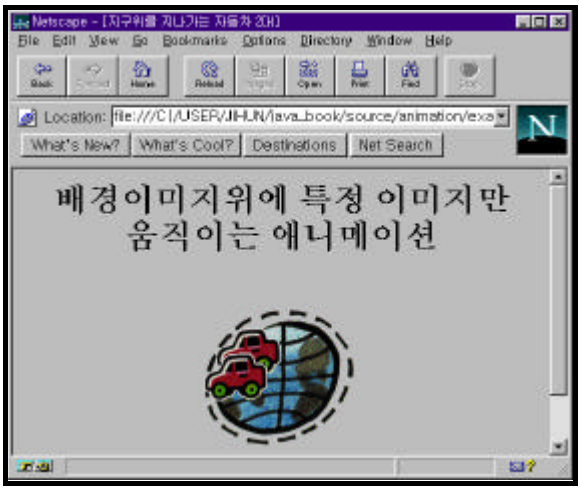
<예제> Example7Applet.java

위의 애플릿을 다음과 같은 HTML 문서를 작성하여 끼워 넣는다. world.gif의 너비와 높이가 각각 200이기 때문에 애플릿의 너비와 높이를 200으로 정한다.

```
<TITLE>지구위를 지나가는 자동차 2대</TITLE>

<center>
<H1> 배경이미지위에 특정 이미지만 움직이는 애니메이션</H1>

<APPLET code=Example7Applet.class width=200 height=200>
<param name=fps value=10>
</applet>

</center>

<HR>
```

<예제 > Example7Applet.html

위의 HTML문서를 웹브라우저에 읽어 들이면 <그림 6>과 같은 결과를 얻는다.



<그림 6> 움직이는 자동차

Example6Applet.java와 비교해보면, init()와 paintFrame()이 좀 다르다.

init()안의 소스코드를 살펴보자.

```
world = getImage(getCodeBase(), "world.gif");

car = getImage(getCodeBase(), "car.gif");
```

은 애플릿이 있는 서버로부터, world.gif와 car.gif를 가져와서 각각 world, car라는 Image객체로 할당시키는 문장들이다.여기서 world.gif와 car.gif는 애플릿이 있는 디렉토리와 같은 디렉토리에 있어야 한다. Image getImage(URL url, String name)는 Applet클래스에 정의되어 있는 메쏘드로서, URL객체가 지정한 위치로부터 name에 해당하는 이미지 파일을 서버로부터 가져오는 기능을 한다.

paintFrame()은 오프스크린 이미지인 offImage안에 지구위에 자동차 2대가 위치한 한 프레임을그린다.

```
g.drawImage(world, (d.width - w)/2, (d.height - h)/2, this);
```

는 배경화면인 지구그림을 먼저 오프스크린 이미지의 중앙에 그리는 문장이다.

```
g.drawImage(car, d.width - ((frame * 5) % w), (d.height - h)/3, this);

g.drawImage(car, d.width - ((frame * 7) % w), (d.height - h)/2, this);
```

는 이미 그려져 있는 지구그림위에 자동차 2대를 겹쳐서 한 프레임을 만드는 문장이다. 여기서 자동차의 x축위치가 frame의 값에 의해서 결정됨을 알 수 있다. 이것은 run()안의 while 루프안에서 repaint()를 한 후, frame의 값이 계속 증가하기 때문에 매 프레임마다 그 값이 달라져서 이전 프레임과는 약간 다른 위치에 자동차 2대를 그릴 수 있는 이유가 된다. 그래서 계속 프레임을 연속적으로 보여주면, 자동차가 움직이는 것처럼 보이는 것이다.

여러개의 이미지 파일을 연속적으로 보여주는 애니메이션

연속적인 프레임들을 각각의 이미지 파일로 만든 후, 그것들을 연속적으로 보여줌으로서 애니메이션 효과를 내는 예제를 살펴보자.

본 예제는 duke라는 마스코트가 손을 움직이는 애플릿인데, <그림 7>처럼 손을 움직이는 각 프레임들이 연속적인 10개의 이미지화일로 서버에 설치가 되어야 한다.

<그림 7> 손을 흔드는 duke의 10 이미지

소스를 살펴보자.

```
import java.awt.*;
import java.applet.*;
public class Example8Applet extends java.applet.Applet implements Runnable {
   int frame;
   int delay;
   Thread animator;

   Dimension offDimension;
   Image offImage;
   Graphics offGraphics;

   Image frames[]; // 각 이미지 파일을 나타낼 Image객체 배열을 선언

   public void init() {
           String str = getParameter("fps");
           int fps = (str != null) ? Integer.parseInt(str) : 10;
           delay = (fps > 0) ? (1000 / fps) : 100;

           frames = new Image[10]; // 10개의 Image객체 배열 생성
           for (int i = 1 ; i <= 10 ; i++) { // 각 이미지 파일을 Image객체배열에 할당한다.
             frames[i-1] = getImage(getCodeBase(), "duke/T" + i + ".gif");
           }
   }

   public void start() {
           animator = new Thread(this);
           animator.start();
   }

   public void run() {
           long tm = System.currentTimeMillis();
           while (Thread.currentThread() == animator) {
```

```
            repaint();

            try {
                tm += delay;
                Thread.sleep(Math.max(0, tm - System.currentTimeMillis()));
            } catch (InterruptedException e) {
                break;
            }

            frame++;
        }
    }

    public void stop() {
        animator = null;
        offImage = null;
        offGraphics = null;

    }

    public void update(Graphics g) {
        Dimension d = size();

        if ((offGraphics == null)
        || (d.width != offDimension.width)
        || (d.height != offDimension.height)) {
            offDimension = d;
            offImage = createImage(d.width, d.height);
            offGraphics = offImage.getGraphics();
        }

        offGraphics.setColor(getBackground());
        offGraphics.fillRect(0, 0, d.width, d.height);
        offGraphics.setColor(Color.black);

        paintFrame(offGraphics);

        g.drawImage(offImage, 0, 0, null);
    }

    public void paint(Graphics g) {
        update(g);
    }

    /**
     * 각 프레임을 저장하고 있는 frames 배열중에서 적당한 프레임을 선택하여, 오프스크린 이      *
```

미지안에 그린다.

```
   */
  public void paintFrame(Graphics g) {
          g.drawImage(frames[frame % 10], 0, 0, null);
  }
}
```

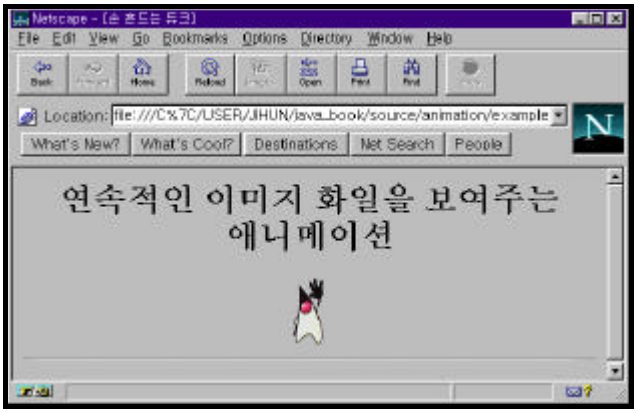
<예제> Example8Applet.java

위의 애플릿을 다음과 같은 HTML 문서를 작성하여 끼워 넣는다. T1.gif~T10.gif들의 너비와 높이가 각각 55, 68 픽셀(pixel)이기 때문에 애플릿의 너비와 높이를 각각 55, 68으로 정한다.

```
<TITLE> 손 흔드는 듀크 </TITLE>

<center>
<H1> 연속적인 이미지 화일을 보여주는 애니메이션</H1>

<APPLET code=Example8Applet.class width=55 height=68>
<param name=fps value=10>
</applet>

</center>

<HR>
```

<예제> Example8Applet.html

위의 HTML문서를 웹브라우저에 읽어 들이면 <그림 8>과 같은 결과를 얻는다.



<그림 8> 손 흔드는 듀크

애플릿에서 init()는 초기화 역할을 하기 때문에, 애니메이션 애플릿에서도 그림 파일을 가져와서 초기화하는 역할을 한다.

```
frames = new Image[10];
for (int i = 1 ; i <= 10 ; i++) {
    frames[i-1] = getImage(getCodeBase(), "duke/T" + i + ".gif");
}
```

는 10개의 이미지 파일을 frames배열에 할당하는 코드인데, getImage의 두 번째 패러미터는 서버의 애플릿이 있는 디렉토리의 하위 디렉토리인 duke에서 T1.gif부터T10.gif까지의 그림파일을 가져오게 된다.

paintFrame()에서 오프스크린 이미지에 프레임을 그려 넣어야 되는데, 그 프레임들은 frames배열안에 있는 이미지들이다.

```
g.drawImage(frames[frame % 10], 0, 0, null);
```

는 frames 배열안의 한 이미지를 오프스크린 이미지안에 집어 넣는 문장이다. 주의 깊게 볼 것은 frame%10이다. frames 의 배열원소가 총 10개이기 때문에, 배열 인덱스는 0부터 9의 값을 가져야 한다. 그래서 frame의 값이 어떤 값일지라도 그것을 만족시키기 위해, frame을 10으로 나눈 나머지(0부터 9)를 배열 인덱스로 한 것이다. 즉 frame의 값이 repaint()후에 1씩 증가하기 때문에, 연속적인 이미지화일을 화면에 보여줄 수 있게 된다.

이미지가 다 도착한 후에 보여주는 애니메이션

그런데, Example7Applet.java나 Example8Applet.java를 실제로 실행시켜보면, 이미지들이 한꺼번에 보이지 않고, 애플릿의 위에서부터 아래로 단계적으로 화면에 나타나는 것을 볼 수 있다. 그것은 getImage()가 실제로 서버로부터 웹브라우저로 이미지화일을 완전히 가져오지 않기 때문이다. 그래서 이미지 파일이 서버로부터 도착하는 양만큼 화면에 보여주기 때문에 이런 현상이 나타난 것이다. 이런 현상을 없애고, 즉 서버로부터 그림화일들이 전부다 도착한 후 화면에 표현하도록 할려면 MediaTracker라는 클래스를 이용해야 한다.

Example8Applet.java와 거의 비슷하지만, 모든 이미지가 도착한 후 애니메이션을 하는 애플릿은 Example8Applet.java에 다음과 같은 코드를 추가해야 한다.

```
public class Example9Applet implements Runnable {
...
MediaTracker tracker;

public void init() {
    ...
    tracker = new MediaTracker(this);
    frames = new Image[10];
    for (int i = 1 ; i <= 10 ; i++) {
      frames[i-1] = getImage(getCodeBase(), "duke/T" + i + ".gif");
      tracker.addImage(frames[i-1], 0); // 추적자가 추적할 이미지를 더한다.
    }
```

```
    }
  public void paintFrame(Graphics g) {
        // 모든 이미지가 도착했을 경우에만, 오프스크린에 프레임을 그린다.
        if (tracker.statusID(0, true) == MediaTracker.COMPLETE) {
          g.drawImage(frames[frame % 10], 0, 0, null);
        }
    }
}
```

<예제> Example9Applet.java

init()에서

```
tracker = new MediaTracker(this);
```

를 통하여 그림화일의 도착여부를 추적할 객체를 생성시킨 후에,

```
tracker.addImage(frames[i-1], 0);
```

로 추적할 각 이미지화일들을 0이라는 그룹에 등록시킨다. MediaTracker클래스의 addImage(Image img, int id)메쏘드는 id로 지정한 그룹에 img를 등록시키는 기능을 한다.

paintFrame()에서

```
if (tracker.statusID(0, true) == MediaTracker.COMPLETE) {
  g.drawImage(frames[frame % 10], 0, 0, null);
}
```

의 tracker.statusID(0, true)는 0이란 그룹에 등록되어 있는 이미지가 웹브라우저로 완전히 도착이 되었는지의 상태를 반환하는 MediaTracker의 메쏘드이다. 완전히 도착했을 경우는 MediaTracker클래스안에 정의되어있는 static상수인 COMPLETE값을 반환한다. 그러므로 위 소스는 모든 이미지화일이 웹브라우저로 전부 도착이 되었을 경우만, 오프스크린 이미지안으로 프레임을 그릴 수 있도록 제한한다.

Example9Applet은 모든 이미지 파일이 다 도착한 후, 화면에 프레임을 표현하기 때문에 Example8Applet보다는 처음 실행속도가 상대적으로 느리다. 그렇지만 깔끔한 느낌을 줄 것이다.

2. 사운드

자바에서 사운드는 현재로선 선(sun)사가 제안한 오디오 형식인 au형식밖에는 지원이 되지 않는다. 애플릿에서 서버의 오디오 파일을 가져와서 그 오디오에 대해서 다음과 같은 콘트롤을 할 수 있다. 오디오를 위한 메쏘드들이 있는 클래스와 인터페이스는 다음과 같다.

1) java.applet.Applet클래스

- void play(URL sound), void play(URL sound, String location) 메쏘드
  사운드 파일의 위치를 URL로 가리키게 하고, 해당되는 사운드화일이 있으면 연주한다. 만약 패러미터로 location이 추가되면 URL위치를 기준, 상대적인 사운드 화일의

위치를 찾아 연주한다.

- AudioClip getAudioClip(URL url, String location)
  사운드 파일의 기준 위치를 url, 상대위치가 location인 사운드화일을 가져와서 AudioClip객체로 반환한다.

2) java.applet.AudioClip 인터페이스

사운드 파일에 대해서 여러 콘트롤을 할 수 있기 때문에 가장 많이 사용된다.
애플릿안에 정의되어 있는 getAudioClip을 이용하여, AudioClip객체를 생성한 후 play(), stop(), loop()를 실행하여 사운드 파일을 연주한다.

애플릿내에서 사용되는 간단한 예를 보자.

```
AudioClip clip = getAudioClip(getCodeBase(), "audio/loop.au");
        clip.play(); // loop.au라는 사운드 화일을 한번만 연주한다.
        clip.stop();  // 사운드를 중단한다.
        clip.loop();  // 사운드를 무한히 반복한다
```

stop()메쏘드는 loop()를 실행해서 사운드가 무한히 연주중일 때, 중단시키는 기능을 한다.

사운드를 사용할 때, 프로그램 코드 구조는 일반적으로 init()안에 getAudioClip()을 사용하여 사운드화일을 가져오고, 그 이후에 다른 메쏘드에서 그 파일을 연주하는 것이다.

배경 사운드가 연주되는 중에, 4초간격으로 특정 사운드가 연주되는, 다음 예제화일을 보자.

```
import java.awt.Graphics;
import java.applet.*;

public class AudioExample extends Applet implements Runnable {

    AudioClip bgsound; // 배경 사운드 파일을 할당할 AudioClip객체 선언
    AudioClip beep; // 단발 사운드 파일을 할당할 AudioClip객체 선언
    Thread sound_runner; // 사운드 파일을 연주하는 쓰레드 객체 선언

    /** 사운드 애플릿은 일반적으로 init()에서 사운드 파일을 가져온다. */
        public void init() {
                bgsound = getAudioClip(getCodeBase(), "loop.au");
                beep = getAudioClip(getCodeBase(), "beep.au");
        }

    /** 사운드를 연주할 쓰레드를 생성한다. */
    public void start() {
        if (sound_runner == null) {
            sound_runner = new Thread(this);
            sound_runner.start();
```

```
        }
    }

    /** 사운드가 연주되고 있을 때, stop()이 호출되면 멈춘다 */
    public void stop() {
        if ( sound_runner != null) {
            if (bgsound != null) bgsound.stop();
            sound_runner.stop();
            sound_runner = null;
        }
    }

    public void run() {
        if (bgsound != null) bgsound.loop();
        while( sound_runner != null) {
            try { Thread.sleep(4000); }
            catch (InterruptedException e) {}
            if (bgsound != null) beep.play();
        }
    }

    public void paint(Graphics g) {
        g.drawString("We are playing sounds. Let's Listen!", 10, 10);
    }
}
```

<예제> AudioExample.java

다음 HTML문서를 만들고 웹 브라우저로 읽으면, 사운드가 출력될 것이다.

```
<TITLE>사운드를 연주하는 애플릿</TITLE>

<center>
<H1> 배경사운드위에 특정 사운드를 정기적으로 연주하는 애니메이션</H1>
<APPLET code=AudioExample.class width=100 height=50>
</applet>
</center>
<HR>
```

<예제> AudioExample.html

loop.au와 beep.au를 애플릿이 있는 디렉토리에 같이 놓고 웹브라우저로 AudioExample.html을 읽는다.

위의 예제에서 사운드를 연주할 쓰레드를 생성한 이유는 배경사운드를 무한히 연주해야 하기 때문이다. 애플릿에서 쓰레드를 생성할 때는, 대부분 init(), start(), paint() 등의 메쏘드 안에 무한히 동작하는 루프를 집어 넣을 경우이다. 그럴 때 사용자나 시스템의 이벤트를 받아들일 수 없는 부작용이 생기기 때문에, 무한 루프를 쓰레드를 생성하여 run()안에 집어 넣는다. 그러면 사용자나 시스템의 이벤트를 받아들이는 코드와 무한 루프를 동시에 실행

시킬 수 있다.

3. 게임 프로그래밍

자바의 멀티미디어 특징을 이용하여, 애니메이션을 이용한 게임 애플릿이 많이 등장하고 있다.. 여기서 설명하고 있는 내용은 네트웍크 기능을 이용하지 않은, 애니메이션만을 활용한 게임 애플릿이다. 그렇지만 다음 부에서 설명하는 네트웍크 기능을 이용하며, MUD 같은 네트웍을 활용한 게임 애플릿도 제작할 수 있을 것이다.

전체 구조

게임 애플릿을 작성하기 위해서, 대략 다음 단계를 밟아야 한다.

1. 게임에 등장하는 사물들을 디자인한다.
2. 게임에서 등장하는 사물들의 움직임 및 반응을 조절하는 게임 엔진을 작성한다.
3. 필요한 이미지화일과 사운드 파일을 준비하여, 애플릿을 작성한다.
4. 테스트한다.

1,2번째 단계가 가장 중요하다. 각 단계를 자세히 살펴보자.

1. 게임에 등장하는 사물들을 디자인한다.

모든 게임에는 등장하는 사물들이 있다. 옛날 오락실의 갤러그같은 경우, 무수히 나타나는 적 우주선, 총알, 자신의 우주선 등이 그것들이다.

객체지향언어로 게임 프로그래밍을 할 때는 이런 사물들을 객체화시켜서 하게 된다. 예를 들면 우주선, 총알 같은 것을 클래스로 만들어, 우주선이나 총알의 속성이나 메쏘드 등을 정하게 된다. 사물들을 객체화시키면, 얻을 수 있는 장점은 크게 2가지다.

1) 비객체지향언어로 프로그래밍하는 것보다, 훨씬 직관적이고 이해가 쉽게 프로그래밍 할 수 있다.

2) 클래스의 상속을 이용하여, 게임을 확장할 때 새롭게 나타나는 사물들을 재사용할 수 있다.

예들 들어 “적 우주선”이란 클래스가 있고, 새롭게 “총을 쏘는, 적 우주선”이란 클래스가 필요할 경우 "적 우주선“이란 클래스를 상속받아서 새로운 속성과 ”총을 쏜다“는 메쏘드를 첨가하면 쉽게 클래스를 만들 수 있다.

게임에 등장하는 사물(thing)을 클래스로 디자인할 때, 많이 사용되는 자바 코드구조는 다음과 같다.

```
public class thing {
        int x, y; // 사물이 애플릿 화면내 배치될, (x,y)좌표
        Image picture; // 사물의 이미지가 할당될 Image객체
```

```
        int priority; // 사물의
        int velocity; // 사물의 움직이는 속도, 주로 한 번에 움직이는 픽셀값을 쓴다.
        boolean visible; // 사물이 화면에 보일 것인가를 결정하는 대수값, true면 화면에 출력한다
        .
        ... // 기타 사물의 데이터가 되는 속성들

        /** 생성자에는 클래스가 나타내는 사물의 이미지와 기타 속성들을 초기화하는 역할을
        한다. */
        public thing(Image picture) {
                this.picture = picture;
        }
        ... // 다른 메쏘드들을 정의한다.
}
```

만약 한 사물이 여러 이미지를 가질 때는, Image객체의 배열을 선언해야 할 것이다. 예를 들어 만약 우주선이 총탄에 맞아 부서지는 장면이 있다면 우주선의 멀쩡한 모습과 우주선이 총탄에 맞아 부서진 모습 등 2개의 이미지가 클래스안에 할당되어야 할 것이다.

사물 클래스를 디자인 할 때, 가장 중요한 것은 사물의 행동 및 서비스, 즉 메쏘드를 정의하는 일이다. 우주선 클래스 같은 경우엔, "(x,y)위치로 총을 쏜다", "(x,y)위치로 이동한다" 등의 메쏘드가 필요할 것이다.

2. 게임에서 등장하는 사물들의 움직임 및 반응을 조절하는 게임 엔진을 작성한다.

사물들을 클래스로 지정한 다음, 그런 사물들의 움직임 및 반응을 계속 추적해야 하는 코드를 작성해야 한다. 이런 코드를 게임 엔진이라고 한다. 예를 들어, 적 우주선이 자신의 우주선이 쏜 총알에 맞으면, 폭발하면서 사라져야 되는데, 이런 것들을 코딩하는 부분이다.

게임엔진을 작성할 때, 큰 파트로 나누면 다음과 같다.

* 사물들의 생성

  사물들을 클래스로 정의한 후에, 게임안에 등장하는 사물들의 종류와 수 만큼 객체들을 생성한다. 주로 init()안에서 객체들을 생성하며, 실제로 게임을 진행하면서 필요하면 동적으로 객체를 생성시키거나 삭제한다.

  Enemy라는 우주선을 나타내는 클래스를 다음과 같이 정의할 수 있을 것이다.

```
public class Enemy {
        int x, y; // 우주선이 애플릿 화면내 배치될, (x,y)좌표
        Image picture; // 우주선의 이미지가 할당될 Image객체
        int velocity; // 우주선이 움직이는 속도, 주로 한 번에 움직이는 픽셀값을 쓴다.
        boolean visible; //우주선이 화면에 보일 것인가를 결정하는 대수값, true면 화면에 출력한
        다.
```

```
        /** 생성자에는 클래스가 나타내는 사물의 이미지와 기타 속성들을 초기화하는 역할을
        한다. */
        public Enemy(Image picture) {
                this.picture = picture;
        }
        public void fire(int x, int y) // (x,y)위치로 미사일을 발사
        { .... }
        public void move(int x, int y)  // (x,y)위치로 우주선을 이동한다.
        { .... }
        …
}

Enemy[] ufo;
public void init() {
        ufo = new Enemy[100]; // 적 우주선 100개를 생성한다.
        for ( int i=0; i<100; i++)
                ufo[i] = Enemy(...); // Enemy클래스의 생성자를 호출하여 초기화한다.
....
}
```

사물들의 숫자가 커질수록, 객체를 배열로 생성하기도 하지만 Linked List를 이용하여 체인화시키기도 한다. 왜냐면 게임에 등장하는 사물들이 갑자기 생기기도 하고, 사라지기도 하기 때문에 원소의 수가 정해진 배열만으로는 이런 동적인 변화를 충분히 수용하기가 어렵기 때문이다. Linked List를 사용할 때는 게임에 새롭게 등장하는 사물은 List에 붙이고, 더 이상 필요없게 된 사물은 List에서 제거하는 작업이 이루어져야 한다.

* 게임 루틴

사용자의 키보드나 마우스의 입력을 반영하여, 구체적으로 사물들의 위치와 반응을 조절해야 한다. 주로 쓰레드안의 run()안에 그런 코드를 집어 넣는다.
게임 프로그램을 자세히 보면, 게임을 만든 사람을 소개하거나 게임의 사용법 소개, 실제 게임 등 게임내에 여러 가지 상태가 존재한다는 것을 알 수 있다. run()안의 코드는 이런 상태에 알맞는 메쏘드를 호출하는 루틴으로 이루어져 있는데, 일반적으로 코드구조는 다음과 같다.

```
        public void run()
        {
                while (game !=null) // Thread객체인 game이 현재 동작중이면,
                {
                        try
                        {
                          Thread.sleep(snooze); // 애니메이션효과를 내기 위해 잠시 쉰다.
                        } catch (InterruptedException e) {}
```

```
                switch (gameState) // 게임상태별로 해당 함수를 호출한다.
                {
                        case 게임상태:
                                게임상태에 맞는 함수 호출(예, gameloop() );
                                break;
                        ....
                        default:
                                break;
                }
                repaint(); // 애니메이션 효과를 내기 위해, repaint()를 호출한다.
        }
}
```

이런 게임 루틴은 사용자의 이벤트에 대해서, 즉각적으로 반응해야 한다.
사용자의 입력을 반영하는 형식은 키보드입력과 마우스입력이 있다.
1) 키보드 입력
애플릿에서 다음의 메쏘드를 이용하여, 키보드 값을 입력받아 사물들의 움직임을 조절한다.

```
public boolean keyDown(java.awt.Event e,int key)
{
                switch (key)
                {
                        case 키값 : 사물의 움직임을 조절하는 코드;
                                break;
                        .....
                        default:
                                break;
                }
}
public boolean keyUp(Event e, int key)
```

일반적으로 사물을 상징하는 객체안의 x,y속성값을 바꾸는 코드를 집어넣는다.
예를 들면 다음과 같은 코드들이 사용될 수 있다.

```
    switch(key)
    {
                case 97:  ufo.x -= 2; // A키를 눌렀을 경우, 왼쪽으로 2픽셀만큼 이동
                        break;
                .....
    }
```

x좌표는 좌우, y좌표는 상하를 나타내기 때문에, 왼쪽으로 이동할려면 x좌표값을 줄여야 한다.

2) 마우스 입력

```
public boolean mouseDown(Event evt, int x, int y)
public boolean mouseUp(Event evt, int x, int y)
public boolean mouseMove(Event evt, int x, int y)
public boolean mouseDrag(Event evt, int x, int y)
public boolean mouseEnter(Event evt, int x, int y)
public boolean mo useExit(Event evt, int x, int y )
```

들을 통해서 마우스의 움직임을 게임안의 사물들에게 반영하는 코드를 작성한다. 키보드 입력과 마찬가지로 주로 사물을 상징하는 객체안의 x,y속성값을 바꾸는 코드를 집어넣는다.

* 그래픽 처리

게임에 등장하는 사물들은 부드럽게 움직여야 한다. 자바에서 부드러운 애니메이션에 최상인 기법은 더블 버퍼링 기법이다. 즉 이 기법을 사용하여, 사물들의 움직임을 연출해야 한다.
위의 게임 루틴을 보면, repaint()가 있을 것이다. 더블 버퍼링 기법을 활용할 때는, repaint()가 호출하는 update()를 오버라이드해야 한다. 즉 화면에 사물들을 표현하는 코드는 update()안에 정의를 해야한다. 일반적인 코드구조는 다음과 같다.

```
public void paint(Graphics g)
{
        g.drawImage(offImage,0,0,this); // 오프스크린 이미지를 화면에 표현한다.
}

public void update(Graphics g)
{
        int k;
        switch (gameState) // 게임상태에 따라서 화면에 표현하는 사물들이 달라진다.
        {
                case 상태번호: // 만약 실제 게임이 진행 중인 상태일 때면,
                        // 게임의 배경화면을 먼저 오프스크린에 그린다.
                        offGraphics.drawImage(배경이미지, x위치, y위치, this);
                        // 배경화면위에 각 사물들을 그린다.
                        for (k=0; k<사물의 개수; k++)
        offGraphics.drawImage(사물 객체의 이미지, 사물의 x좌표, 사물의 y좌표, this);
                                        break;
                .....
                default:
                        break;
        }

                paint(g); // 화면에 오프스크린 이미지를 색칠한다.
```

}

offGraphics객체는 오프스크린 이미지에 프레임을 그리기 위한 Graphics 객체이며, offImage는 오프스크린 이미지이다.(애니메이션 파트의 더블버퍼링 기법 참조할 것). 위에서 사물 객체의 이미지는 만약 사물객체안에 저장된 이미지일 것이다. 예를 들면, 위에서 언급한 ufo가 thing클래스와 같은 종류의 클래스 객체 배열이라면, ufo[k].image가 구체적 코드가 될 것이다. x좌표, y좌표는 ufo[k].x, ufo[k].y가 될 것이다. update()는 화면에 표현할 모든 사물을 프레임에 그리는 것이 목적이기 때문에, 각 사물을 나타내는 객체들이 배열로 저장이 되어 있거나 Linked List로 연결되어 있는지에 따라서 코드가 달라지게 된다. 그렇지만 for문을 사용하여 배열이나 Linkded List로 연결된 사물들을 적당한 위치의 프레임으로 그리는 원리는 같다.

* 사운드 처리

위의 사운드 관련예제처럼 먼저 init()안에 게임 프로그램 내에서 사용하는 사운드 파일을 서버로부터 가져온다. 그리고 게임 루프안에서 특정소리가 나야 될 코드다음에, 사운드 파일을 연주하는 코드를 작성한다. 우주선이 미사일을 발사할 때, 사운드를 내는 다음 예를 보자.

```
AudioClip fire_sound;

public void init() {
        ....
        fire_sound = getAudioClip(getCodeBase(), "sound/fire_sound.au");
        ....
}

public void gameloop() {
        ....
        ufo.fire(120, 30); // ufo라는 우주선이 (120, 30)위치로 미사일을 발사한다.
        fire_sound.play();
        …
}
```

애플릿의 전체 구조

게임 애플릿내에 정의해야 되는 코드의 역할들은 다음과 같다.

```
public class 게임이름 extends Applet implements Runnable {
        게임에 필요한 각종 변수나 객체 선언;
        ....
        public void init() {
          1. 서버로부터 게임에 등장하는 사물그림을 모아놓은 한 이미지화일을 가져온다.
          2. 그 이미지를 사물그림별로 쪼개서, 각각을 Image객체에 할당한다.
          3. 만약 사운드가 사용되면, 사운드화일을 가져온다.
          4. 게임에 필요한 사물들을 나타내는 객체를 생성한다.
```

5. 프레임을 나타내는 오프스크린 이미지와 오프스크린 그래픽 객체를 생성한다.
}

public void start() {
게임을 수행할 쓰레드를 생성한다.
}

public void run() {
게임상태에 따라, 필요한 코드를 호출한다. 게임을 실질적으로 실행하는 코딩을 호출하거나 집어넣는다.
}

public void paint() {
실제로 프레임을 화면에 표현한다.
}

public void update() {
게임에 나타나는 모든 사물들을 오프스크린 이미지안으로 그린다. 즉 한 프레임을 만든다.
}
}

예제 프로그램

이번 예제에서 설명할 애플릿은, 펭귄이 미로속을 다니면서, 자신을 추적하는 불을 향해 얼음을 던져서 파괴하는 게임이다. 사용자가 키보드를 누름으로서, 펭귄은 상하좌우로 움직이며, 얼음 방향으로 계속 밀면 얼음이 날아간다. 불은 펭귄을 추적하며, 펭귄의 몸을 건드려서 죽인다.

본 애플릿은 사운드는 사용하지 않지만, 펭귄이나 불의 특정 행동에 사운드 효과를 넣을 수도 있을 것이다.

이번 예제는 사물들을 클래스로 정의하지 않고 사물의 이미지와 위치만으로 프로그래밍 되었기 때문에, 객체지향적으로 프로그래밍되었다고 말할 수 없다. 따라서 독자가 아래 코드를 바탕으로 객체지향프로그래밍으로 애플릿을 재작성하는 것이 좋은 숙제로 남을 것이다. 본 게임에서 등장하는 사물들, 즉 펭귄, 불, 얼음조각, 돌 등을 클래스화시켜서 적당한 속성과 메쏘드를 정의하면 될 것이다.

본 애플릿에서 사용되는 이미지화일은 <그림 9>처럼 단 1개이다. 잘 보면 이 이미지화일 속에 게임속에 사용되는 모든 사물의 이미지들이 포함되어 있음을 알 수 있다.

<그림 9> 게임에 사용되는 이미지 (iceblox.gif)

애플릿의 예제 코드는 다음과 같다.

```
// Iceblox
// By Karl Harnell, April 8 1996
import java.awt.*;
import java.awt.image.*;
import java.net.*;

public  class iceblox extends java.applet.Applet implements Runnable
{
        int i,j,k;
        final int playX=390,playY=330,mainX=390,mainY=348,smalls=48,blockX=13,blockY=11;
        final int animP[]={7,8,9,8,10,11,12,11,4,5,6,5,1,2,3,2};
        final int animF[]={32,33,34,35,36,35,34,33};
        final int levFlame[]={2,3,4,2,3,4},levRock[]={5,6,7,8,9,10};
        final int levSpeed[]={3,3,3,5,5,5},levIce[]={35,33,31,29,27,25};
        final int effMax=5;
        int playArea[];
        int gameState,counter,dir,inFront,inFront2,level,coins;
        int effLevel,lives=3;
        int x[],y[],dx[],dy[],motion[],look[],creature[],ccount[],actors,flames;
        int sideIX[]={0,-1,1,-15,15},coorDx[]={0,-30,30,0,0},coorDy[]={0,0,0,-30,30};
        Image collection,offImage,playField,small[],title;
        Graphics offGraphics,playGraphics,tempG;
        MediaTracker tracker;
        ImageFilter filter;
        ImageProducer collectionProducer;
        long snooze=100,score;
        Thread game;
        Math m;

        /** 이미지를 사물별로 쪼개서, Image객체에 할당시킨다. */
        public void init()
```

```
{
        setBackground(Color.black);
        offImage=createImage(mainX,mainY);
        offGraphics=offImage.getGraphics();
        offGraphics.setColor(Color.black);
        offGraphics.fillRect(0,0,mainX,mainY);
        playField=createImage(playX,playY);
        playGraphics=playField.getGraphics();
        playGraphics.setColor(Color.black);
        tracker=new MediaTracker(this);
        collection = getImage(getCodeBase(),"iceblox.gif");
        tracker.addImage(collection,0);
        try
        {
                tracker.waitForID(0);
        }
                catch(InterruptedException e) {}
        collectionProducer=collection.getSource();
        small=new Image[smalls];
        k=0;i=0;j=0;
        while (k<smalls)
        {
                filter=new CropImageFilter(j*30,i*30,30,30);
                small[k]=createImage(new FilteredImageSource(
                        collectionProducer,filter));
                tracker.addImage(small[k],1);
                k++;
                j++;
                if (j==8)
                {
                        j=0;
                        i++;
                }
        }
        filter=new CropImageFilter(0,180,224,64);
        title=createImage(new FilteredImageSource(
                        collectionProducer,filter));
        tracker.addImage(title,1);

        playArea=new int[(blockX+2)*(blockY+3)];
        x=new int[20];
        y=new int[20];
        dx=new int[20];
        dy=new int[20];
        look=new int[20];
        motion=new int[20];
        creature=new int[20];
        ccount=new int[20];
```

```
		gameState=7;
		try
		{
			tracker.waitForID(1);
		}
			catch(InterruptedException e) {}
		resize(mainX,mainY);
}

/** 게임상황에 따른 코드 호출을 한다. 중앙 관제탑의 기능을 한다. */
public void run()
{
		while (game !=null)
		{
			try
			{
				game.sleep(snooze);
			} catch (InterruptedException e) {}
			counter=(counter+1)&255;
			switch (gameState)
			{
				case 0:
					prepareField();
					break;
				case 1:
					showField();
					break;
				case 2:
					gameLoop(); // 실제 게임이 진행되는 코드
					break;
				case 3:
					happyPenguin();
					break;
				case 4:
					clearField();
					break;
				case 5:
					fixDeath();
					break;
				case 6:
					gameOver();
					break;
				case 7:
					drawIntro1();
					break;
				case 8:
```

```
                                        waitIntro1();
                                        break;
                                case 9:
                                        drawIntro2();
                                        break;
                                case 10:
                                        waitIntro2();
                                        break;
                                case 11:
                                        drawIntro3();
                                        break;
                                case 12:
                                        waitIntro3();
                                        break;
                                default:
                                        break;
                        }
                        repaint();
                }
        }
        public void start()
        {
                if (game==null)
                {
                        game=new Thread(this);
                        game.start();
                }
        }
        public void stop()
        {
                if ((game!=null)&&(game.isAlive()))
                {
                        game.stop();
                }
                game=null;
        }
        public boolean keyDown(java.awt.Event e,int key)
        {
                if (gameState==2)
                {
                        switch (key)
                        {
                                case 97:
                                        dir=1; // A키보드를 누르면 펭귄이 왼쪽 이동
                                        break;
                                case 100:
                                        dir=2; // D키보드를 누르면 오른쪽 이동
```

```
                                        break;
                                case 107:
                                        dir=3; // K키보드를 누르면 위로 이동
                                        break;
                                case 109:
                                        dir=4; // M키보드를 누르면 아래로 이동
                                        break;
                                default:
                                        break;
                        }
                }
                else if ((gameState>6)&&(key==32))
                        gameState=0;
                return false;
        }

        public boolean keyUp(java.awt.Event e,int key)
        {
                dir=0;
                return false;
        }

        /** 게임의 초기화면을 만든다. 돌, 불꽃, 펭귄의 위치를 확정한다. */
        public void prepareField()
        {
                int i,j,p,q;
                if (level>effMax)
                        effLevel=effMax;
                else
                        effLevel=level;
                offGraphics.setColor(Color.black);
                offGraphics.fillRect(0,0,mainX,mainY);
                playGraphics.setColor(Color.black);
                playGraphics.fillRect(0,0,playX,playY);
                offGraphics.setColor(Color.lightGray);
                offGraphics.fill3DRect(0,mainY-playY-4,mainX,4,true);
                offGraphics.setColor(Color.white);
                offGraphics.drawString("SCORE:",2,12);
                updateScore(0);
                offGraphics.drawString("LEVEL:  "+(level+1),125,12);
                offGraphics.drawString("SPARE LIVES:",220,12);
                for (i=0;i<lives;i++)
                        offGraphics.drawImage(small[13],300+i*15,-16,this);
                for (i=0;i<(blockX+2)*(blockY+3);i++)
                        playArea[i]=1;
                for (i=1;i<=blockY;i++)
                        for (j=1;j<=blockX;j++)
```

```
                    playArea[i*(blockX+2)+j]=0;
playArea[blockX+3]=1;
i=0;
while (i<levIce[effLevel]) // 얼음 조각
{
        p=1+(int)(m.random()*blockX);
        q=1+(int)(m.random()*blockY);
        if (playArea[q*(blockX+2)+p]==0)
        {
                playArea[q*(blockX+2)+p]=2;
                playGraphics.drawImage(small[16],(p-1)*30,(q-1)*30,this);
                i++;
        }
}
i=0;
while (i<levRock[effLevel]) // 돌조각
{
        p=1+(int)(m.random()*blockX);
        q=1+(int)(m.random()*blockY);
        if (playArea[q*(blockX+2)+p]==0)
        {
                playArea[q*(blockX+2)+p]=1;
                playGraphics.drawImage(small[14],(p-1)*30,(q-1)*30,this);
                i++;
        }
}
i=0;
while (i<5) // 동전들
{
        p=1+(int)(m.random()*blockX);
        q=1+(int)(m.random()*blockY);
        if (playArea[q*(blockX+2)+p]==0)
        {
                playArea[q*(blockX+2)+p]=10;
                playGraphics.drawImage(small[24],(p-1)*30,(q-1)*30,this);
                i++;
        }
}
playArea[blockX+3]=0; // Clear start square
gameState=1;
snooze=100;
counter=0;
motion[0]=0;
actors=1;
coins=0;
flames=0;
look[0]=0;
```

```
		x[0]=30;
		y[0]=30;
		dx[0]=6;
		dy[0]=6;
		look[0]=2;
		creature[0]=1;
		for (i=0;i<20;i++)
			ccount[i]=0;
	}
	public void showField()
	{
		Graphics saveGraphics;
		saveGraphics=offGraphics.create();
		offGraphics.clipRect(playX/2-(counter*playX/2/30),mainY-playY+
		playY/2-(counter*playY/2/30),counter*playX/30,counter*playY/30);
		offGraphics.drawImage(playField,0,mainY-playY,this);
		if (counter==30)
		{
			gameState=2;
			snooze=100;
		}
		offGraphics=saveGraphics;
	}

	/** 실질적으로 게임이 진행되는 가장 핵심적인 코드이다. */
	public void gameLoop()
	{
		if (flames<levFlame[effLevel])
		{
			if (x[0]<(playX/2))
				x[actors]=playX+30;
			else
				x[actors]=0;
			y[actors]=30*(1+(int)(m.random()*blockY));
			j=(y[actors]/30)*(blockX+2)+x[actors]/30;
			motion[actors]=0;
			dx[actors]=levSpeed[effLevel];
			dy[actors]=levSpeed[effLevel];
			creature[actors]=4;
			if ((playArea[j+1]==0)||(playArea[j-1]==0))
			{
				actors++;
				flames++;
			}
		}
		for (i=0;i<actors;i++)
		{
```

```
                    ccount[i]++;
                    switch(motion[i])
                    {
                            case 1:
                                    x[i]-=dx[i];
                                    break;
                            case 2:
                                    x[i]+=dx[i];
                                    break;
                            case 3:
                                    y[i]-=dy[i];
                                    break;
                            case 4:
                                    y[i]+=dy[i];
                                    break;
                            default:
                                    break;
                    }
                    j=(y[i]/30)*(blockX+2)+x[i]/30;
                    switch(creature[i])
                    {
                            case 1: // 펭귄일때
                                    if ((x[i]%30 == 0)&&(y[i]%30 == 0))
                                            motion[i]=0;
                                    if (motion[i]==0)
                                    {
                                            inFront=playArea[j+sideIX[dir]];
                                            if ((j+2*sideIX[dir])<0)
                                                    inFront2=1;
                                            else

            inFront2=playArea[j+2*sideIX[dir]];
                                            if (inFront==0)
                                                    motion[i]=dir;
                                            else
                                            {
                                                    if
((inFront2==0)&&((inFront==2)||(inFront==10))) // 펭귄이 얼음조각을 밀면?
                                                    {
                                                            if (inFront==2)
                                                            {

            creature[actors]=2;

            look[actors]=16;
                                                            }
                                                            else
```

```
                                                                {

            creature[actors]=3;

            look[actors]=24;
                                                                }

            x[actors]=x[i]+coorDx[dir];

            y[actors]=y[i]+coorDy[dir];
                                                                dx[actors]=15;
                                                                dy[actors]=15;

            playGraphics.fillRect(x[actors]-30,y[actors]-30,30,30);
                                                                motion[actors]=dir;
                                                                actors++;

            playArea[j+sideIX[dir]]=0;
                                                            }
                                                            else if
((inFront>1)&&(inFront<18)) // Crack ice
                                                            {

            playArea[j+sideIX[dir]]++;
                                                                if (inFront==9) // All
cracked?
                                                                {

            playGraphics.fillRect(x[i]+coorDx[dir]-30,y[i]+coorDy[dir]-30,30,30);

            playArea[j+sideIX[dir]]=0;

            updateScore(5);
                                                                }
                                                                else if (inFront==17)
                                                                {

            playGraphics.fillRect(x[i]+coorDx[dir]-30,y[i]+coorDy[dir]-30,30,30);

            playArea[j+sideIX[dir]]=0;

            updateScore(100);
                                                                    coins++;
                                                                }
                                                                else
```

```
                playGraphics.drawImage(small[inFront+15],x[i]+coorDx[dir]-30,y[i]+coorDy[dir]-30,this);
                                        }
                                }
                        }
                        if (motion[i]!=0)
                                look[i]=animP[(motion[i]-1)*4+counter%4];
                        for (k=1;k<actors;k++)
                                if (creature[k]==4)
                                        if (((x[k]-x[i])<20)&&((x[i]-
x[k])<20)&&((y[k]-y[i])<20)&&((y[i]-y[k])<20))
                                        {
                                                creature[k]=6;
                                                x[k]=0;
                                                y[k]=0;
                                                motion[k]=0;
                                                ccount[i]=0;
                                                dx[i]=0;
                                                dy[i]=0;
                                                creature[i]=7;
                                        }
                        break;

                case 2: // 얼음조각이 이동되고 있을때
                        if ((x[i]%30 == 0)&&(y[i]%30 ==
0)&&(playArea[j+sideIX[motion[i]]]!=0))
                        {
                                playArea[j]=2;
                                playGraphics.drawImage(small[16],x[i]-
30,y[i]-30,this);
                                removeActor(i);
                        }
                        break;
                case 3: // 동전이 든 얼음조각이 이동되고 있을 때
                        if ((x[i]%30 == 0)&&(y[i]%30 ==
0)&&(playArea[j+sideIX[motion[i]]]!=0))
                        {
                                playArea[j]=10;
                                playGraphics.drawImage(small[24],x[i]-
30,y[i]-30,this);
                                removeActor(i);
                        }
                        break;
                case 4: // 불
                        look[i]=animF[counter%8];
                        if (motion[i]==0)
                                motion[i]=(int)(1+m.random()*4);
                        if ((x[i]%30 == 0)&&(y[i]%30 == 0)) // 펭귄을 추적
```

```
                {
                        if (((x[i]-x[0])<3)&&((x[0]-x[i])<3))
                        {
                                if (y[i]>y[0])
                                        motion[i]=3;
                                else
                                        motion[i]=4;
                        }
                        else if (((y[i]-y[0])<3)&&((y[0]-y[i])<3))
                        {
                                if (x[i]>x[0])
                                        motion[i]=1;
                                else
                                        motion[i]=2;
                        }
                        if (playArea[j+sideIX[motion[i]]]!=0)
                                motion[i]=0;
                }
                for (k=1;k<actors;k++) // 이동되는 얼음조각과 충
돌?
                        if ((creature[k]&254)==2)
                                if // 충돌식  (((x[k]-
x[i])<30)&&((x[i]-x[k])<30)&&((y[k]-y[i])<30)&&((y[i]-y[k])<30))
                                {
                                        creature[i]=5;
                                        k=actors;
                                        look[i]=37;
                                        motion[i]=0;
                                        ccount[i]=0;
                                        updateScore(50);
                                }
                        break;
        case 5: // 반짝이는 "50" 이란 숫자
                        look[i]=37+(counter&1);
                        if (ccount[i]>20)
                        {
                                flames--;
                                removeActor(i);
                        }
                        break;
        case 6: // 아무것도 아님
                        break;
        case 7: // 펭귄이 죽을 때, 보여주는 해골
                        if (ccount[i]<8)
                                look[i]=39+ccount[i];
                        else if (ccount[i]<30)
                                look[i]=47;
```

```
                                        else
                                        {
                                                lives--;
                                                if (lives<0)
                                                        gameState=5;
                                                else
                                                {
                                                        actors=1;
                                                        flames=0;
                                                        counter=0;
                                                        dx[i]=6;
                                                        dy[i]=6;
                                                        creature[0]=1;
                                                        look[0]=2;

offGraphics.setColor(Color.black);
                                                        offGraphics.fillRect(300,0,45,14);
                                                        for (k=0;k<lives;k++)

offGraphics.drawImage(small[13],300+k*15,-16,this);
                                                }
                                        }
                                        break;
                                default:
                                        break;
                        }
                }
                if (coins>4)
                {
                        gameState=3;
                        updateScore(1000);
                        counter=0;
                        coins=0;
                        offGraphics.drawImage(playField,0,mainY-playY,this);
                }
}
public void happyPenguin()
{
                if (counter>35)
                {
                        level++;
                        gameState=4;
                        counter=0;
                }
}
public void clearField()
{
```

```
                offGraphics.setColor(Color.black);
                offGraphics.fillRect(playX/2-(playX*counter/30),mainY-playY/2-(playY*counter/30),
                        playX*counter/15,playY*counter/15);
                if (counter>14)
                        gameState=0;
        }
        public void fixDeath()
        {
                offGraphics.setColor(Color.black);
                offGraphics.fillRect(0,0,mainX,mainY);
                offGraphics.setColor(Color.white);
                offGraphics.drawString("GAME OVER",175,100);
                offGraphics.drawString("You scored "+score,160,130);
                offGraphics.drawImage(small[2],190,150,this);
                counter=0;
                gameState=6;
        }
        public void gameOver()
        {
                if (counter>80)
                        gameState=7;
        }
        public void drawIntro1()
        {
                level=0;
                score=0;
                lives=3;
                offGraphics.setColor(Color.black);
                offGraphics.fillRect(0,0,mainX,mainY);
                offGraphics.setColor(Color.white);
                offGraphics.drawImage(title,(mainX-224)/2,10,this);
                offGraphics.drawString("ACTORS AND OBJECTS",145,97);
                offGraphics.drawImage(small[2],140,110,this);
                offGraphics.drawString("Pixel Pete, the penguin",180,130);
                offGraphics.drawImage(small[34],120,150,this);
                offGraphics.drawImage(small[32],140,150,this);
                offGraphics.drawString("Evil flames",180,170);
                offGraphics.drawImage(small[16],140,190,this);
                offGraphics.drawString("Ice cube",180,210);
                offGraphics.drawImage(small[14],140,230,this);
                offGraphics.drawString("Solid rock",180,250);
                offGraphics.drawImage(small[24],140,270,this);
                offGraphics.drawString("Frozen gold coin",180,290);
                offGraphics.drawString("Press SPACE to start",138,330);
                counter=0;
                gameState=8;
        }
```

```
public void waitIntro1()
{
	offGraphics.setColor(Color.black);
	offGraphics.fillRect(120,150,50,30);
	offGraphics.drawImage(small[animF[(counter+2)&7]],120,150,this);
	offGraphics.drawImage(small[animF[counter&7]],140,150,this);
	if (counter>70)
		gameState=9;
}
public void drawIntro2()
{
	offGraphics.setColor(Color.black);
	offGraphics.fillRect(0,75,mainX,230);
	offGraphics.setColor(Color.white);
	offGraphics.drawString("HOW TO PLAY",165,97);
	offGraphics.drawImage(small[2],140,110,this);
	offGraphics.drawString("Move up, down, left and right",180,122);
	offGraphics.drawString("with the K, M, A and D keys",180,137);
	offGraphics.drawImage(small[10],70,150,this);
	offGraphics.drawImage(small[16],140,150,this);
	offGraphics.drawString("Walk against ice cubes",180,162);
	offGraphics.drawString("to move them out of the way",180,177);
	offGraphics.drawLine(110,160,136,160);
	offGraphics.drawLine(116,169,136,169);
	offGraphics.drawImage(small[10],80,190,this);
	offGraphics.drawImage(small[18],110,190,this);
	offGraphics.drawImage(small[16],140,190,this);
	offGraphics.drawString("Walk against blocked",180,202);
	offGraphics.drawString("ice cubes to crack them",180,217);
	offGraphics.drawImage(small[28],110,230,this);
	offGraphics.drawImage(small[9],140,230,this);
	offGraphics.drawString("Free the gold coins by",180,242);
	offGraphics.drawString("crushing the ice around them",180,257);
	offGraphics.drawImage(small[9],80,270,this);
	offGraphics.drawImage(small[32],140,270,this);
	offGraphics.drawLine(110,280,126,280);
	offGraphics.drawLine(110,289,130,289);
	offGraphics.drawString("And watch out",180,282);
	offGraphics.drawString("for the flames",180,297);
	gameState=10;
	counter=0;
}

public void waitIntro2()
{
	offGraphics.setColor(Color.black);
	offGraphics.drawImage(small[1+(counter % 12)],140,110,this);
	offGraphics.fillRect(140,270,30,30);
```

```
                offGraphics.drawImage(small[animF[counter&7]],140,270,this);
                if (counter>80)
                        gameState=11;
        }
        public void drawIntro3()
        {
                offGraphics.setColor(Color.black);
                offGraphics.fillRect(0,75,mainX,230);
                offGraphics.setColor(Color.white);
                offGraphics.drawString("SCORING",180,97);
                offGraphics.drawImage(small[10],110,110,this);
                offGraphics.drawImage(small[18],140,110,this);
                offGraphics.drawString("Breaking ice,",180,122);
                offGraphics.drawString("5 points",180,137);
                offGraphics.drawImage(small[33],60,150,this);
                offGraphics.drawImage(small[16],80,150,this);
                offGraphics.drawImage(small[9],140,150,this);
                offGraphics.drawLine(112,160,126,160);
                offGraphics.drawLine(112,169,130,169);
                offGraphics.drawString("Putting out flame",180,162);
                offGraphics.drawString("with ice, 50 points",180,177);
                offGraphics.drawImage(small[10],110,190,this);
                offGraphics.drawImage(small[27],140,190,this);
                offGraphics.drawString("Freeing coin,",180,202);
                offGraphics.drawString("100 points",180,217);
                for (j=0;j<5;j++)
                        offGraphics.drawImage(small[15],100-9*j,230,this);
                offGraphics.drawImage(small[39],140,230,this);
                offGraphics.drawString("Taking all coins and advancing",180,242);
                offGraphics.drawString("to next level, 1000 points",180,257);
                gameState=12;
                counter=0;
        }

        public void waitIntro3()
        {
                offGraphics.setColor(Color.black);
                offGraphics.fillRect(60,150,20,30);
                offGraphics.drawImage(small[animF[counter&7]],60,150,this);
                offGraphics.drawImage(small[16],80,150,this);
                if (counter>70)
                        gameState=7;
        }
        public void removeActor(int i)
        {
                int j;
                for (j=i;j<actors;j++)
                {
```

```
                        x[j]=x[j+1];
                        y[j]=y[j+1];
                        dx[j]=dx[j+1];
                        dy[j]=dy[j+1];
                        look[j]=look[j+1];
                        motion[j]=motion[j+1];
                        creature[j]=creature[j+1];
                }
                actors--;
        }
        public void updateScore(long i)
        {
                score+=i;
                offGraphics.setColor(Color.black);
                offGraphics.fillRect(50,0,60,12);
                offGraphics.setColor(Color.white);
                offGraphics.drawString(String.valueOf(score),50,12);
        }
        public void paint(Graphics g)
        {
                g.drawImage(offImage,0,0,this);
        }
        public void update(Graphics g)
        {
                int k;
                switch (gameState)
                {
                        case 2: // 실제 게임이 진행될 때
                                offGraphics.drawImage(playField,0,mainY-playY,this);
                                for (k=0;k<actors;k++)
                                        offGraphics.drawImage(small[look[k]],x[k]-30,y[k]-
30+mainY-playY,this);
                                break;
                        case 3:
                                offGraphics.drawImage(small[39*(counter&1)],x[0]-30,y[0]-
30+mainY-playY,this);
                                break;
                        default:
                                break;
                }

                paint(g);
        }
}
```

<예제> Iceblox.java

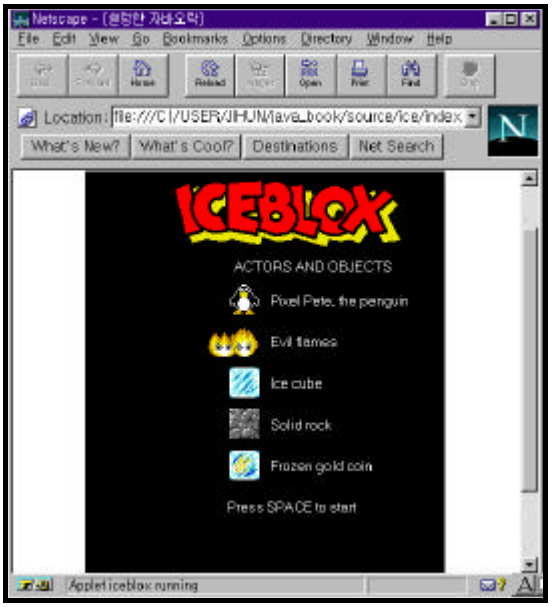
본 애플릿을 다음과 같은 HTML문서에 끼어 넣는다.

<예제>

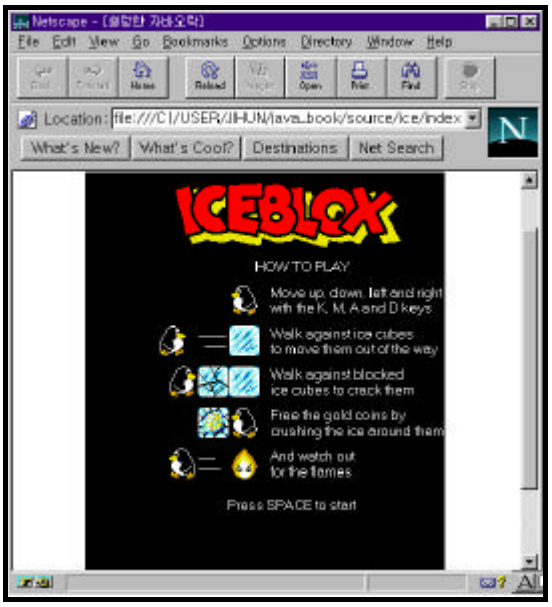
```
<HTML>
<HEAD>
<TITLE>썰렁한 자바오락 </TITLE>
</HEAD>
<body bgcolor=FFFFFF>
<center>
<h2>펭귄은 불을 싫어한다</h2>
<APPLET CODE="iceblox.class" width=350 height=390>
</APPLET>
</center>
</BODY>
</HTML>
```

<예제> Iceblox.html

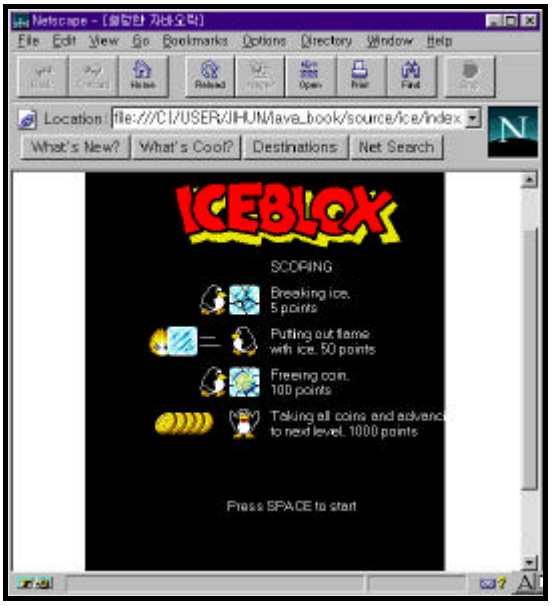
본 애플릿을 실행하면, 게임을 진행하기 전에 다음 3개의 장면이 소개되면서, 게임소개 및 사용법을 설명한다.



<그림 10> 게임에 등장하는 사물 소개

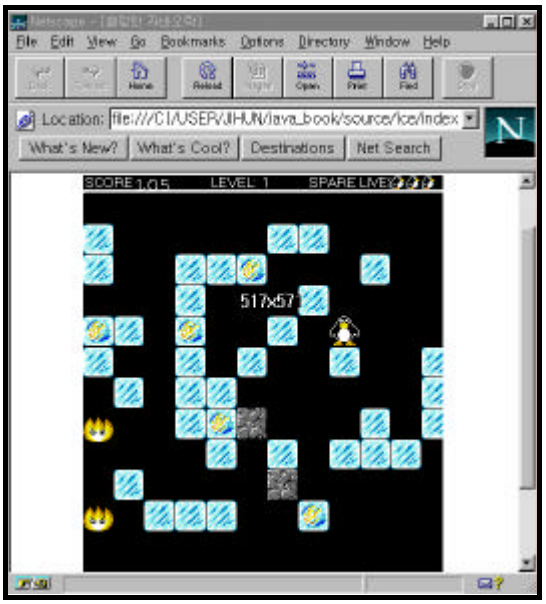


<그림 11> 게임진행 방법소개



<그림 12> 점수계산 소개

소개가 계속될 때, 스페이스바를 누르면 <그림 13>과 같은 본격적인 게임이 시작된다.



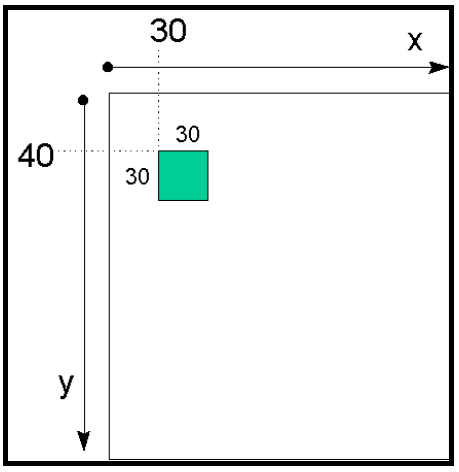
<그림 13> 게임진행 화면

* 한 이미지안에서 부분 이미지를 추출하는 법

본 예제에서는 그림화일이 한 개가 사용되고 있는데, 그것은 이 이미지안에 게임에 등장하는 모슨 사물의 이미지들을 합쳐 놓았기 때문이다. 그러므로 게임내에서 사물의 이미지를 사용하려면, 프로그램내에서 사물별 이미지를 쪼개는 코딩을 해야 한다.

이미지를 부분으로 쪼개는데, 필요한 패키지는 java.awt.Image이다.

전체 이미지에서 <그림 14>처럼 전체이미지의 (30, 40)에서 가로, 세로 30픽셀의 부분 이미지를 쪼개서 이미지로 할당하는 코드순서는 대략 다음과 같다.



<그림 14> 부분 이미지 쪼개기

```
Image collection; // 게임에 사용되는 사물들의 이미지를 모아놓은 한 이미지
ImageFilter filter;
ImageProducer collectionProducer;
Image part; // collection에 할당된 이미지중에서 쪼개서 할당될 사물 이미지

public void init() {
  .....
  collection = getImage(getCodeBase(),"iceblox.gif"); // 전체이미지를 가져온다.
  collectionProducer=collection.getSource(); // 그 이미지로부터 ImageProducer객체를 생성한다.
  filter=new CropImageFilter(30, 40, 30, 30); // (30,40)을 탑코너로 하고, 가로세로 30을 쪼갠다.
  part = createImage(new FilteredImageSource(collectionProducer,filter)); // 부분이미지를 할당
  ....
}
```

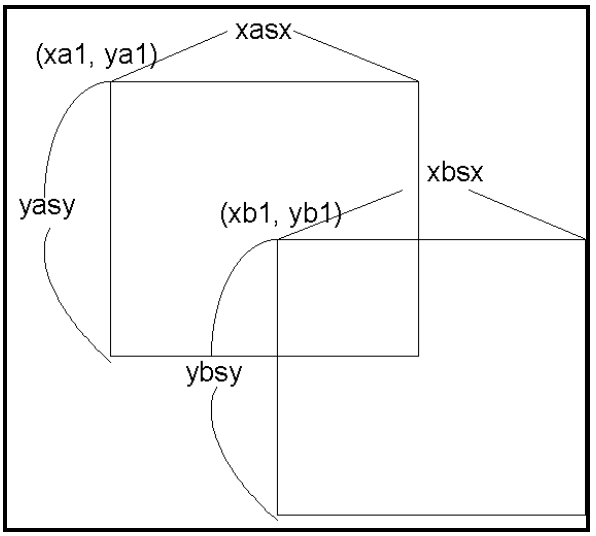
이런식으로 원하는 사물들을 별개의 이미지화일로 할당할 수 있다.

* 충돌검사하는 법

오락 프로그래밍을 할 때, 자주 사용되는 테크닉 중에 하나가 두 사물이 충돌하는 지를 어떻게 파악하느냐는 것이다. 즉 총알을 적 우주선에게 발사했을 때, 두 사물이 충돌했는지를 어떻게 아는가?

일반적으로 게임내에서 사물들은 사각형으로 둘러싸인 이미지로 표현된다. 만약 이 두 사물이 서로 겹친다면, 그 때 충돌이 일어났다고 판단한다.

<그림 15>과 같이 두 사각형이 있다고 하자.



<그림 15> 충돌하는 두 사각형

이 두 사각형이 충돌할 때 사용되는 조건식은 다음과 같다.

(xa1<(xb1+xbsx)) and (xb1<(xa1+xasx)) and (ya1<(yb1+ybsy)) and (yb1<(ya1+yasy))

위 식이 참이면, 두 사각형은 충돌하는 것이다. 본 예제에서도 펭귄과 얼음, 펭귄과 불,

얼음과 불 등의 충돌을 파악할 때, 위의 식을 사용하였다.

제 4부 네트워킹 프로그래밍

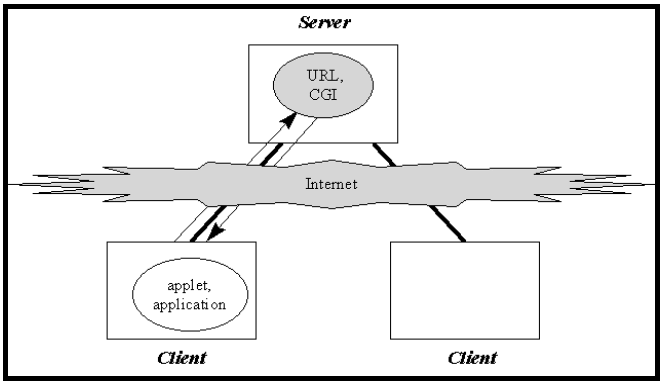
자바는 네트웍을 지원하는 클래스 라이브러리를 기본적으로 내장하고 있기 때문에, 쉽게 네트웍 관련 프로그래밍을 할 수 있다. 네트웍 라이브러리는 java.net이라는 패키지안에 저장되어 있다.

네트웍을 이용해 인터넷으로 연결된 다른 컴퓨터와 데이터를 주고 받기 위한 수단으로, URL클래스나 Socket, ServerSocket, DatagramSocket클래스 등을 이용한다.

참고로 Java 1.0.X버전에서 애플릿은 자신이 원래 있던 서버이외의 다른 컴퓨터와는 보안 문제로 인해 네트웍연결을 할 수 없다. 물론 어플리케이션은 어떤 컴퓨터와도 네트웍 연결을 할 수 있다.

1. URL클래스

URL클래스는 <그림 1> 처럼 특정서버의 URL(Uniform Resource Locator)의 자원(resource)과 서로 데이터를 주고 받을 때 사용한다. 자바의 네트웍관련 라이브러리중에서 최상위 클래스이다.



<그림 1> URL클래스를 사용한 네트웍 구조도

URL클래스의 객체를 생성하기 위해 사용되는 예는 다음과 같다.

URL클래스의 생성자가 많이 있지만, 그 중 자주 사용되는 형태는 다음과 같다.

1) URL(String url)

```
URL selab = new URL("http://object.cse.cau.ac.kr");
```

- selab객체를 통해서, http://object.cse.cau.ac.kr라는 URL의 정보를 얻을 수 있다.

2) URL(URL baseURL , String relativeURL)

먼저 위의 절대 URL객체를 생성한 후, URL seintro =new URL(selab, intro.html) ;

-"http://object.cse.cau.ac.kr/intro.html" 라는 URL정보를 seintro 객체를 통해서 얻을 수 있다.

URL클래스는 자바프로그램내에서 주로 특정 CGI프로그램에게 데이터를 보내고, 그 데이터를 처리한 결과값을 얻고자 할 때 사용된다.

먼저 "http://object.cse.cau.ac.kr" 이라는 특정 URL의 데이터를 받아서 화면에 출력하는 어플리케이션을 살펴보자.

```
import java.net.*;
import java.io.*;

class URLInput {
  public static void main( String args[]) {
    try {
      URL selab = new URL("http://object.cse.cau.ac.kr/");

            DataInputStream in_stream; // URL로부터 읽어들일 입력 스트림 생성
            String inputLine; // URL로부터 오는 한 줄의 문자열을 할당받는 String객체

            in_stream = new DataInputStream(selab.openStream());

            while ((inputLine = in_stream.readLine()) != null) {
                  System.out.println(inputLine) ; //화면에 출력
            }

            in_stream.close(); // 모든 일이 끝나면 스트림을 닫는다.

            }catch (MalformedURLException me) {
                  System.out.println("MalformedURLException:  " + me);
            } catch (IOException ioe) {
                  System.out.println("IOException:  "+ioe);
            }
      }
     }
```

<예제> URLInput.java

명령라인을 통해 "java URLInput"을 입력함으로 어플리케이션을 실행시킨다.

위에서 가장 중요한 코드는

```
in_stream = new DataInputStream(selab.openStream());
```

이다. URL클래스의 openStream()메쏘드를 통해서 URL객체와의 통신 선로를 마련한 후, 그것을 DataInputStream객체로 변환하여, DataInputStream의 읽기 관련 메쏘드를 통하여 URL로부터 데이터를 읽을 수 있게되는 것이다. 그리하여

```
while ((inputLine = in_stream.readLine()) != null) {
```

는 “http://object.cse.cau.ac.kr"의 내용을 한줄씩 마지막 줄까지 읽어 나갈 수 있게 된다.

좀 더 발전된 예제를 보자. 다른 어떤 서버(exam.co.kr)에 사용자가 수험번호를 보내면, 수능시험합격여부를 알려주는 CGI프로그램(enter)이 이미 만들어져 있다고 하자. 이 CGI프로그램은 수험번호가 입력으로 들어오면 합격했으면 "Success", 불합격했으면 ”Failure"라는 문자열을 반환한다. 이미 만들어져있는 CGI프로그램을 이용할려면, 즉 특정서버의 CGI프로그램을 실행시켜서 그 결과를 보고자 할려면 어떻게 해야 하는가?

다음 예제를 보자.

```
import java.io.* ;
import java.net.* ;

public class Success {
        public static void main(String args[]) {
         try{
                if (args.length != 1) { // 수험번호를 입력하지 않았을 때
                        System.err.println("Usage: java Success seat_number");
                        System.exit (1);
           }

           String seat_num = URLEncoder.encode(args[0]); // 수험번호를 인코딩(encoding)
           URL url = new URL("http://exam.co.kr/cgi-bin/enter"); // CGI 프로그램의 URL
           URLConnection connection = url.openConnection();
           PrintStream outStream = new PrintStream(connection.getOutputStream());

           DataInputStream inStream;
           String inputLine;

           outStream.println(seat_num);
           outStream.close();

           inStream = new DataInputStream(connection.getInputStream());
           while ((inputLine = inStream.readLine()) != null) {
                System.out.println(inputLine) ; // CGI 프로그램의 결과값을 화면에 출력
           }
           inStream.close();
           }catch (MalformedURLException me) {
                System.err.println("MalformedURLExceprion:  " + me);
           }catch (IOException ioe) {
                System.err.println("IOException:  " + ioe);
                }
        }
        }
```

<예제> Success.java

본 어플리케이션은 "java Success 수험번호"로 실행시킨다. 물론 exam.co.kr이란 서버의 cgi-bin/enter라는 CGI프로그램은 없기 때문에, 실제로 실행시키면 에러메시지가 나올 것이다.

위에서 주목할 코드는

```
String seat_num = URLEncoder.encode(args[0]);
```

이다. URLEncoder클래스의 정적(static)메쏘드인 encode(String str)은 str을 "x-www-form-urlencoded"라는 MIME타입의 문자열을 반환한다. CGI프로그램으로 보내는 문자열 데이터들은 "x-www-form-urlencoded" 형태이기 때문에, 웹상으로 데이터를 보내기 전에 전송할 일반 문자열을 이 타입으로 변환시켜줘야 한다. 그래서 수험번호를 서버에게 보내기 전에 이 타입으로 바꾸는 것이다.

CGI프로그램의 URL객체를 생성시킨후에,

```
URLConnection connection = url.openConnection();
PrintStream outStream = new PrintStream(connection.getOutputStream());
```

문장은 URL객체로 데이터를 보내기 위해, PrintStream이라는 출력 스트림객체를 생성한다. 이렇게 PrintStream객체 outStream을 생성한 후에, outStream의 출력 메쏘드를 사용한 문장 outStream.println(seat_num); 은 문자열(seat_num) 이 지정한 URL, 즉 서버의 CGI프로그램으로 전송되게 된다.

그 다음, CGI프로그램이 처리한 결과값을 받기 위해서

```
inStream = n ew DataInputStream(connection.getInputStream());
    while ((inputLine = inStream.readLine()) != null) {
            System.out.println(inputLine) ;
    }
```

문장들이 있다. 이것은 입력 스트림인 inStream을 통해서, inStream.readLine()문장을 실행하여 CGI프로그램이 처리한 문자열을 읽어 처리하는 기능을 한다.

참고로 텍스트 데이터를 위한 입력스트림은 DataInputStream, 출력 스트림은 PrintStream을 많이 사용한다.

위의 코드를 정리하여, 특정서버의 CGI프로그램에게 데이터를 보내고 받는 프로그래밍을 일반화하면 다음과 같은 단계를 밟는다.

1. 특정서버의 CGI프로그램에 해당하는 URL객체와 관련 객체를 생성한다.

```
URL url = new URL("http://exam.co.kr/cgi-bin/enter");
URLConnection connection = url.openConnection();
```

2. URL객체로부터 입력 스트림(Stream)과 출력 스트림을 얻는다.

```
PrintStream outStream = new PrintStream(connection.getOutputStream());
DataInputStream inStream = new DataInputStream(connection.getInputStream());
```

3. 출력 스트림을 통해, CGI프로그램에게 데이터를 보낸다.

```
outStream.println(seat_num);
```

4. 서버의 CGI프로그램이 처리한 결과값을, 입력 스트림을 통해 데이터를 받고 처리한다.

```
inStream = new DataInputStream(connection.getInputStream());
while ((inputLine = inStream.readLine()) != null) {
        System.out.println(inputLine) ;
}
```

5. 입출력 스트림을 닫는다.

```
outStream.close();
inStream.close();
```

2. 클라이언트/서버 프로그래밍

위의 URL클래스는 최상위 클래스로서, 프로그래밍하기는 쉽지만 세부적이고 자세한 프로그래밍을 하기가 어렵다. 그래서 자바는 소켓(Socket)프로그래밍을 위한 라이브러리를 제공한다.

소켓은 네트웍으로 연결된 컴퓨터사이에 돌아가는 양쪽 프로그램내에서, 서로간 통신을 할 때 양쪽 끝에서 그 연결을 처리하는 프로그램의 일부분이라고 생각하면 된다. 예를 들어 공을 서로 주고 받는 투수와 포수가 있을 때, 투수와 포수는 클라이언트와 서버프로그램이고, 공은 네트웍을 이동하는 데이터, 투수와 포스가 손에 낀 글러브는 소켓이라고 생각하면 된다. 글러브는 공을 받는 최종 수단이 된다. 글러브가 없으면 공을 못 주고 받듯이, 네트웍 프로그래밍에서는 소켓이 꼭 필요하다.

소켓 프로그래밍은 포수와 투수가 어떻게 글러브를 활용하여 공을 상대방과 잘 주고 받게 하느냐가 관건이 된다.

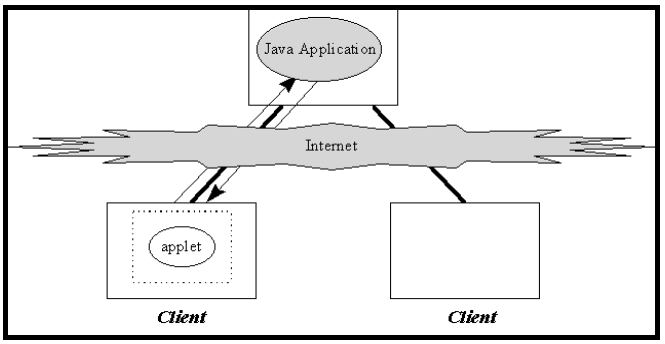
소켓 프로그래밍을 위해서, java.net패키지안의 Socket, ServerSocket, DatagramSocket클래스를 제공한다. 이 클래스들을 이용해 TCP와 UDP방식, 2가지로 데이터를 송수신할 수가 있는데, 본 책에서는 UDP보다 데이터 송수신이 안정적인 TCP방식 위주로 설명할 것이다.

소켓 프로그래밍을 통하여, 주로 클라이언트/서버 어플리케이션을 구현한다. 클라이언트/서버 프로그래밍은, 클라이언트가 서버에게 처리하고자 하는 데이터를 보내면, 서버는 해당데이터를 처리하여 클라이언트에게 보내고, 클라이언트는 받은 데이터를 사용자에게 보여주는 어플리키에션을 작성할 때 사용하는 프로그래밍을 의미한다.

본 프로그래밍에서는 사용자는 클라이언트와 서버, 2종류의 프로그램을 구현해야 한다. 일반적으로 웹상에서 프로그래밍할 때는 클라이언트 프로그램은 애플릿으로 구현하고, 서버 프로그램은 어플리케이션으로 구현하는 경우가 많다. 이럴 때, 클라이언트 컴퓨터는 애플릿을 가져온 웹브라우저가 있는 컴퓨터, 서버 컴퓨터는 애플릿이 있던 웹 서버가 된다. 서버 프로그램은 항상 실행시켜야 하기 때문에, 주로 서버에서 데몬상태로 실행시킨다.

클라이언트/서버 프로그래밍을 통하여 주로 작성되는 프로그램의 종류는 대략 3가지 정도로 나올 수 있는데, 다음과 같다.

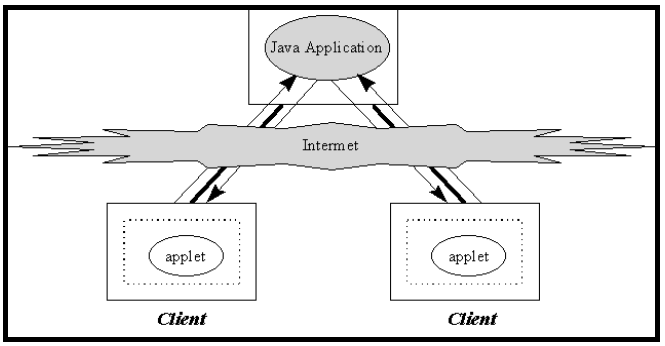
1. 단순 클라이언트/서버 구조



<그림 1> 단순/클라이언트 서버 구조 그림

본 그림을 통해 알 수 있듯이, 다양한 클라이언트(애플릿)들이 한 서버에게 처리할 데이터를 보내면, 서버는 그 데이터를 처리하여 각 클라이언트에게 데이터를 보낸다.

2. 클라이언트끼리의 통신을 위한 클라이언/서버 구조

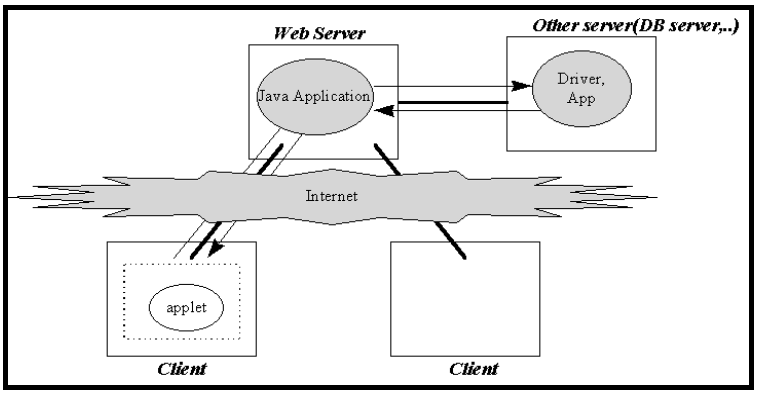


<그림 2> 통신을 위한 클라이언트/서버 구조

다양한 클라이언트(애플릿)가 서로간의 통신을 하기 위해, 사용되는 구조이다. 채팅이 대표적인 경우다. 어떤 클라이언트가 데이터를 서버에게 보내면, 서버는 등록된 모든 클라이언트에게 데이터를 전송한다. 이런식으로 등록된 다른 애플릿에게 데이터를 보낼 수 있으면, 다른 애플릿에서 보내온 데이터를 받을 수 있다. MUD게임같은 경우도 위와 같은 구조를 취한다.

3. 서버가 다른 서버와 통신하는 클라이언트/서버 구조

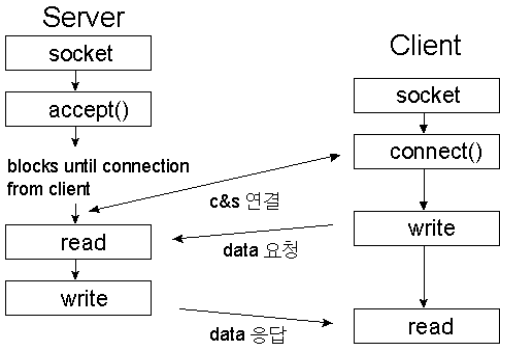
서버가 직접 데이터를 처리하기도 하지만, 다른 서버에게 데이터를 보내어 도움을 요청하는 경우도 있다. 이미 데이터베이스 서버가 구축이 되어 있다고 가정하자. 그때, 애플릿이 웹서버의 서버프로그램에게 DB서버의 값을 원할 때, 서버프로그램이 DB서버에 값을 문의하여 받아서 다시 애플릿에게 보내주는 형태이다.



<그림 3> 다른 서버와 통신하는 클라이언트/서버 구조

실제로는 1, 2구조가 많이 사용되고 있다.

위와 같은 프로그래밍을 하기 위해서는 <그림 4>과 같은 클라이언트/서버 프로그램의 원리를 반드시 이해해야한다.



<그림 4> 클라이언트/서버 프로그램 일반 구조

클라이언트 프로그램

클라이언트 프로그램은 먼저 소켓을 만들고, 서버컴퓨터와 접속을 시도하여 송수신 선로를 만든다. 그 다음 데이터를 송신하고 그 결과값을 받아 처리한다.

구체적인 코드형식은 다음과 같다.

1. 소켓을 생성하고, 서버와 연결시도

```
Socket echoSocket = new Socket(연결할 서버의 도메인이름, 포트번호);
```

2. 데이터를 서버에 송신하기 위한, 출력 스트림을 생성한다.

```
PrintStream os = new PrintStream(echoSocket.getOutputStream());
```

일단, OutputStream을 생성하면, 표준 입출력의 형태로 데이터를 보낼 수 있다.

3. 서버에 데이터를 송신한다.

os.println(보낼 텍스트 데이터);

4. 서버에서 처리된 결과값을 수신하기 위한, 입력 스트림을 생성한다.

DataInputStream is = new DataInputStream(echoSocket.getInputStream());

5. 데이터를 읽어 들인다.

String in_string = is.readLine();

6. 모든 처리가 끝나면, 입출력 스트림과 소켓을 닫는다.

os.close();

is.close();

echoSocket.close();

서버 프로그램

서버 프로그램은 먼저 클라이언트가 자신에게 접속하기를 원하는지를 계속 파악하면서, 접속을 원하는 클라이언트가 있을 때, 소켓을 생성하여 그것을 통해 클라이언트로부터 데이터를 수신하고, 처리한 후 다시 클라이언트에게 전송한다.

구체적인 코드형식은 다음과 같다.

l. 접속을 시도하는 클라이언트와 송수신할 수 있는 소켓을 생성한다.

ServerSocket serverSocket = new ServerSocket(클라이언트가 접속을 시도하는 포트번호);

- 클라이언트 프로그램내에서 사용하는 포트번호와 일치해야 한다.

Socket clientSocket = serverSocket.accept();

- 클라이언트가 통신을 시도하여, 서버가 감시하고 있는 포트번호에 접속할때까지 기다리는 ServerSocket의 accept()메쏘드가 그 연결을 인식하여 실질적인 통로인 소켓을 생성한다.

2. 클라이언트로 오는 데이터를 수신하기 위한, 입력 스크림을 생성한다.

DataInputStream is = new DataInputStream(clientSocket.getInputStream()) ;

3. 클라이언트로부터 데이터를 읽는다.

String in_str = is.readLine();

4. 클라이언트로 데이터를 송신할, 출력 스트림을 생성한다.

PrintStream os = new PrintStream(clientSocket.getOutputStream());

5. 클라이언트로 데이터를 송신한다.

os.println(전송할 데이터);

6. 다음 순서로, 입출력 스트림과 소켓을 닫는다.

```
is.close() ;
os.close();
clientSocket.close();
serverSocket.close();
```

위의 예에선, 전송하는 데이터가 텍스트 데이터일 경우로 한정하여, 입출력 스트림을 Data InputStream, PrintStream으로 제한하였다. 그렇지만 송수신하고자 하는 데이터가 다른 종류일 경우에는, 다른 스트림을 생성하면 된다. 예를 들어 BufferedInputStream을 원한다면, 다음과 같이 코딩하면된다.

```
BufferedInputStream = new BufferedInputStream(clientSocket.getInputStream()) ;
```

3. 관련 예제

단순 클라이언트/서버 프로그램

자! 이제 관련 예제를 살펴보자. 먼저 <그림 1> 처럼 단순 클라이언트/서버 구조를 갖는 프로그램을 살펴보자. 이 프로그램은 사용자가 애플릿을 통해서 어떤 문자열을 치면, 그 문자열이 서버프로그램에 전달되고, 서버프로그램은 전달받는 문자열을 순서를 뒤집은 문자열로 만들어서 애플릿에게 보내면 애플릿은 그 뒤집어진 문자열을 사용자에게 보여준다.

클라이언트 프로그램을 애플릿으로 만들어 보면, 다음과 같다.

```
import java.applet.*;
import java.awt.*;
import java.io.*;
import java.net.*;
import java.lang.*;

public class nut_client extends Applet {
  int PORT = 9878; // 서버 어플리케이션으로 데이터를 보낼 포트번호
  Socket c_socket;
  DataInputStream in;
  PrintStream out;
  TextField inputfield; // 사용자로부터 문자열을 입력받는 텍스트 필드 객체
  TextArea outputarea; // 사용자한테 뒤집혀진 문자열을 보여줄 텍스트 영역 객체
  StreamListener listener; // 데이터의 송수신을 전담하는 쓰레드가 될 객체

  public void init() {
    try {
      // 애플릿이 있는 서버와 통신할 소켓 생성
      c_socket = new Socket(this.getCodeBase().getHost(), PORT);
```

```
      System.out.println("Client! make socket successfully");

      in = new DataInputStream(c_socket.getInputStream());
      out = new PrintStream(c_socket.getOutputStream());
      System.out.println("Client! make stream successfully!");

      inputfield = new TextField();
      outputarea = new TextArea();
      outputarea.setEditable(false); // 편집을 못하게 함.
      this.setLayout(new BorderLayout());
      this.add("North", inputfield);
      this.add("Center", outputarea);

      listener = new StreamListener(in, outputarea); // 데이터 송수신을 위한 객체 생성

      this.showStatus("Connected to "+c_socket.getInetAddress().getHostName()+" :
"+c_socket.getPort());
    }catch (IOException e){ this.showStatus(e.toString());}
  }

 /** GUI관련 이벤트를 받아서 처리하는 부분 */
  public boolean action(Event e, Object what) {
    if (e.target == inputfield) { // 만약 이벤트가 입력필드에서 있으면, 즉 Enter키를 누르면
      out.println((String)e.arg); // 사용자가 입력필드에 데이터를 입력하면, 서버로 송신한다.
      inputfield.setText(""); // 입력 필드를 공란으로 만든다.
      return true;
    }
    return false;
  }
}

/** 데이터를 송수신할 쓰레드이다. */
class StreamListener extends Thread {
  DataInputStream in;
  TextArea output;

 /* 서버와의 입출력 스트림을 패러미터로 받아서 초기화한다 */
  public StreamListener(DataInputStream in, TextArea output) {
    System.out.println("Client! enter in StreamListener");
    this.in = in;
    this.output = output;
    this.start(); // 쓰레드를 실행시킨다.
  }

  public void run() { // 실제 쓰레드 실행부분
    String line;
```

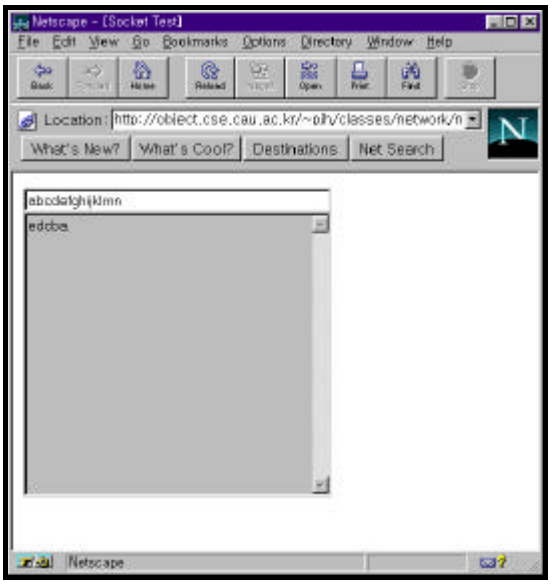
```
   System.out.println("enter in run()");
   try {
    for ( ; ; ) {
            line = in.readLine(); // 서버로부터 오는 데이터를 읽는다.
     System.out.println("Client! line : "+line);
     if (line == null)
              break;
     output.setText(line); // 그 데이터를 텍스트 영역에 출력한다.
    }
   } catch (IOException e){ output.setText(e.toString());}

    output.setText("Connection closed by server.");
  }
}
```

<예제> nut_client.java

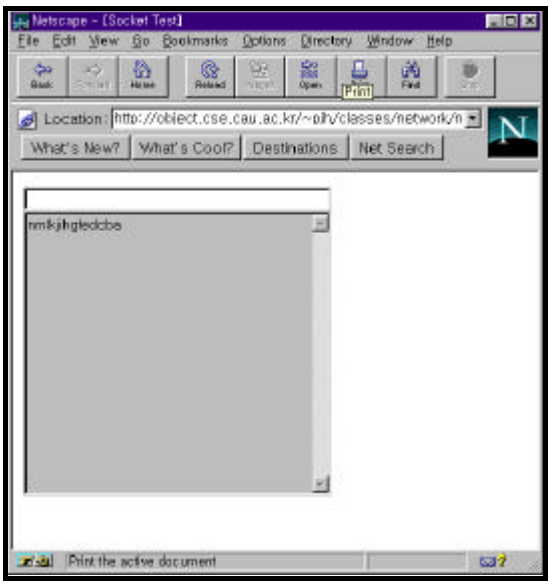
이 애플릿을 다음과 같은 HTML**문서에 끼워 넣는다**.

```
<HTML>
<HEAD>
<TITLE> Socket Test </TITLE>
</HEAD>

<APPLET CODE="nut_client.class" WIDTH=300 HEIGHT=300>
</APPLET>
</BODY>

</HTML>
```

이 애플릿을 웹 브라우저를 통해 서버로부터 가져오면, 다음 <그림 5>**와 같다**.



<그림 5> 문자열을 입력하고 엔터키를 치기 전 애플릿



<그림 6> 엔터키를 친 후에 뒤집혀진 문자열

소스를 살펴보면, 애플릿은 데이터를 송수신하는 부분을 쓰레드로 정의하고 있음을 알 수 있다. 이것은 사용자의 이벤트를 받아들이면서, 네트웍으로 송수신하는 것을 동시에 하길 원하기 때문이다.

c_socket = new Socket(this.getCodeBase().getHost(), PORT);

에서 this.getCodeBase().getHost()는 애플릿이 있던 서버 컴퓨터의 이름을 반환하게 된다. g

etCodeBase()가 애플릿이 있는 컴퓨터의 URL객체를 반환하고, URL클래스안에 URL이 가리키는 컴퓨터이름을 반환하는 메쏘드가 있기 때문에 그와 같은 기능을 하게 되는 것이다. 된다.

나머지 코드들은 위에서 설명한 클라이언트/서버 프로그램 구조에서 제시한 것과 비슷할 것이다. 다른 부분은 사용자와의 인터페이스 부분인데, 인터페이스를 GUI 컴포넌트로 구성하고, action()메쏘드를 통하여 사용자의 이벤트를 데이터의 송수신과 연결시켜주면 된다. 위의 action()에서는 사용자가 Enter키를 누르면, 이전까지 입력했던 문자열을 서버로 송신하도록 되어 있다.

물론 이 애플릿을 웹브라우저로 가져오기 전에 이미 서버 프로그램은 데몬으로 실행되어 있어야 한다. 그렇지 않으면 당연히 에러가 날 것이다. 만약 웹서버가 유닉스라면 다음과 같은 명령이 이미 입력되었어야 한다.

"java nut_server &"(&는 백그라운드 프로세스로 만드는 명령어이다.)

그러면 서버프로그램을 살펴보자.

```
import java.io.*;
import java.net.*;
import java.lang.*;

public class nut_server extends Thread {
 int PORT = 9878; // 접속을 시도하는 클라이언트 프로그램과 같은 포트번호
 ServerSocket s_socket;

 public static void main(String[] args) {
  int port=0;
  new nut_server(port);  // 쓰레드객체를 생성한다.
 }

 public nut_server(int port) {
  if (port == 0)
   port = PORT;

  try {
   s_socket = new ServerSocket(port); // 클라이언트를 감시한 ServerSocket 객체 생성
   System.out.println("open server socket successfully!");
  } catch (IOException e) { error_handling(e, "fail to open server socket") ;}
  System.out.println("Server : listening on port "+port);

  this.start(); // 쓰레드를 시작한다.
 }

 public void run() { // 실질적인 쓰레드 실행 부분
```

```
    try {
      while (true) { // 무한히 실행한다.
        // 접속을 시도하는 클라이언트와의 소켓을 생성한다.
              Socket c_socket = s_socket.accept();
        System.out.println("accept client socket successfully!");
        Connection c = new Connection(c_socket); // 접속하는 송수신을 전담하는 쓰레드 생성
      }
    } catch (IOException e) {error_handling(e, "fail in accepting a client") ;}
  }


  // error handling routine
  public static void error_handling(IOException e, String msg) {
    System.err.println(msg+" : "+e);
    System.exit(1);
  }

}

  /** 특정 클라이언트와 송수신을 전담하는 쓰레드로서, 접속하는 클라이언트마다 한 쓰레드 *
 씩 생성된다.
   */
class Connection extends Thread {
  Socket client=null;
  DataInputStream in=null;
  PrintStream out=null;

  public Connection(Socket c_socket) {
    client = c_socket;
    try {
      // 클라이언트와 송수신할 입출력 스트림 생성
      in = new DataInputStream(client.getInputStream());
      out = new PrintStream(client.getOutputStream());
      System.out.println("make data stream successfully.");
    } catch (IOException e) {
      try {
              client.close();
        System.out.println("close the client socket successfully");
      } catch (IOException e1) {error_handling(e1, "fail in closing client socket");}

      error_handling(e, "fail in getting socket streams");
    }

    this.start(); // 쓰레드 생성한다.
  }
```

```
public void run() {
  String line;
  StringBuffer revline;
  int len;

  try {
   for ( ; ; ) {
          line = in.readLine(); // 클라이언트로부터 데이터를 읽는다.
    System.out.println("line : "+line);

          if (line == null)
           break;
    len = line.length();
    System.out.println("the length of this string : "+len);
          revline = new StringBuffer(len);
          for (int i=len-1; i>=0; i--)  // 읽은 문자열을 뒤집는다.
           revline.insert(len-1-i, line.charAt(i));
    out.println(revline); // 클라이언트에게 뒤집은 문자열을 보낸다.
    System.out.println("revline : "+revline);
   }
  } catch (IOException e) {error_handling(e, "line input or output error");}

  try{
   client.close();}
  catch (IOException e) { error_handling(e, "finally fail in closing client socket");}
 }

 // error handling routine
 public static void error_handling(IOException e, String msg) {
  System.err.println(msg+" : "+e);
  System.exit(1);
 }
}
```

<예제> nut_server.java

서버에서 "java nut_server &" 처럼 항상 서버프로그램은 항상 실행한 상태이어야 한다.

서버프로그램은 여러 클라이언트가 동시에 접속을 시도할 경우가 있기 때문에, 접속된 각 클라이언트와의 데이터 송수신 및 데이터 처리를 멀티 쓰레드로 하는 것이 가장 큰 특징이다. 그 이유는 이렇다. 만약 한 서버에 동시에 100개의 클라이언트가 접속했을때, 각 클라이언트에게 접속한 순서대로 처리한 결과값을 송신하면 맨 마지막에 접속한 클라이언트가 가장 느린 응답을 받게 될 것이다. 그것보다 각 클라이언트와의 송수신을 멀티 쓰레드로 동시에 처리한다면, 모든 클라이언트는 비슷한 응답시간을 가지게 될 것이다.

위의 서버에서 멀티쓰레드 역할을 하는 객체가 Connection클래스이다.

```
        while (true) {
        Socket c_socket = s_socket.accept();
 System.out.println("accept client socket successfully!");
 Connection c = new Connection(c_socket); // 접속하는 송수신을 전담하는 쓰레드 생성
}
```

라는 문장을 통해, 접속을 시도하는 클라이언트가 있을 때마다, 송수신을 담당하는 쓰레드인 Connection 객체를 생성하고 실행시킨다. while루프이기 때문에, 접속하는 클라이언트가 있을 때 마다 해당 쓰레드를 생성시킬 수 있다.

Connection 쓰레드가 하는 일은 간단하다. 클라이언트가 보내온 문자열을 뒤집어서 다시 클라이언트에게 보내는 것이다. 이것은 run()안에 코딩되어 있다.

채팅 프로그램

이번엔 좀 더 복잡한 프로그램으로서, 클라이언트(애플릿)끼리의 통신을 목적으로 하는 채팅 프로그램을 살펴보자. 이 프로그램은 애플릿을 통하여 사용자를 등록하고 그들끼리 대화를 주고 받을수 있다. 자신이 입력한 문자열이 등록된 다른 모든 클라이언트에게 전송이 되고, 다른 클라이언트가 입력한 문자열은 자신이 볼 수 있는 기능을 한다.

애플릿 사용법은 이렇다. 사용자의 이름을 입력하고 "Connect"버튼을 누르면 서버에 사용자를 등록시킨다. 그 후 사용자가 텍스트 필드에 입력한 문자열은 다른 등록된 사용자에게 전달된다. 만약 "Log Out" 버튼을 누르면 접속을 끊게 된다.

먼저 애플릿의 소스코드를 보자.

```
import java.awt.*;
import java.applet.Applet;
import java.net.*;
import java.io.*;
public class Client extends Applet implements Runnable {
        boolean is_connected = false;
        TextField textField;
        TextArea textArea; // 대화가 출력되는 텍스트 영역
        TextField  name; // 이름을 입력하는 텍스트 필드
        Socket to_server;
        DataInputStream is;
        PrintStream os;
        Thread thread;
        Label name_box;
        Label input_box;
        Label users;
        TextArea user_list; // 현재 등록된 사용자 이름을 보여주는 텍스트 영역
        Button quit; // 중단 버튼
```

```
        Button connect; // 연결 버튼

public void init() {
        setBackground(Color.lightGray);
        setFont(new Font("Helvetica", Font.PLAIN, 12));
        textField = new TextField(55);
        textArea = new Text Area(10, 55);
        user_list = new TextArea(10,30);
        textArea.setEditable(false);
        user_list.setEditable(false);
        name = new TextField(25);

        quit = new Button("Log Out");
        connect = new Button ("Connect");
        name_box = new Label("Your name is: ",Label.LEFT);
        input_box = new Label("Type your text on the line below.",Label.LEFT);
        users = new Label("These are the people online: ");

  // 애플릿에 컴포넌트를 붙인다.

        add(name_box);
        add(name);
        add(connect);
        add(quit);
        add(textArea);
        add(input_box);
        add(textField);
        add(users);
        add(user_list);
        validate();
}

public void start(){
        thread = new Thread(this,"main");
        thread.start();
}

public void run(){
        try{
                textArea.appendText("Hit the connect button to start.\n");
                while (true){
                        try{thread.sleep(300);}catch(InterruptedException e){}
                        if (!is_connected || is == null || os == null) // 연결이 안된 상태면,
                                continue; // 처음부터 다시...
                         if (is.available() > 0 ){ // 서버로부터 오는 데이터가 있으면,
                                String text = is.readLine() + '\n'; // text로 읽는다.
                                if (text.startsWith("##names##")) // 등록 사용자 리스트라면
```

```
                        write_list(text.substring(9)); // 사용자이름을 갱신한다.
                            else
                    textArea.appendText(text);//일반 채팅 문자열이면, 그냥 보여줌
                        }
                }
        }catch(IOException e){}
}
/** 등록된 사용자 리스트를 화면에 보여준다 */
public void write_list(String list){
        int i;
        int next;
        user_list.setText (""); // 현재 리스트를 지운다.
        for (i=0;i<list.length();i++){ // 리스트에 등록된 사용자수만큼
                next = list.indexOf('|',i); // 사용자이름을 구분할 수 있는 정수
                if (next >= list.length() || next == -1)
                        break;
                user_list.appendText(list.substring(i,next)+'\n'); // 사용자를 쓴다.
                i = next; // 새로운 구분자값
        }
}

/** 사용자로 부터 이벤트가 발생했을 때, 처리하는 루틴 */
public boolean handleEvent(Event evt) {

        if (!is_connected && evt.id == Event.ACTION_EVENT){// 이벤트가 GUI컴포넌트이면,
                textArea.appendText("Hit the connect button to connect.\n");
                if(evt.target==name ){ // 이벤트가 이름을 입력하는 텍스트필드인 name이라면,
                        String text=name.getText();
            textArea.appendText("Trying to connect\n");
            try{
                try{
                    to_server = new Socket(getCodeBase().getHost(),4444);
                      }catch(UnknownHostException e){
                 textArea.appendText("Not able to connect, sorry.\n");
                       return super.handleEvent(evt);
                    }

                 is = new DataInputStream(new BufferedInputStream(to_server.getInputStream()));
                 os = new PrintStream (new BufferedOutputStream(to_server.getOutputStream(),1024),
false);
                 } catch(IOException e){}
                 is_connected = true;  //make a connection
                 os.println("##name##" + text);
                 os.flush();
                 textArea.appendText("\nYour name is "'+text+'" right now. \n");
        }
```

```
        }

        //서버와 연결된 상태에서 GUI이벤트 발생
        if (evt.id==Event.ACTION_EVENT&&is_connected){
                if (os == null || is == null){
                        textArea.appendText("An error has occured.\n");
                        return super.handleEvent(evt);
                }
                if (evt.target == textField){ // 채팅 텍스트를 입력한다면
                        String text = textField.getText();
                        textField.setText("");
                        os.println(text); // 서버로 보낸다.
                        os.flush();
                }
        }

        if (evt.target == quit && is_connected){ // 중단하기를 원하면
                textArea.appendText("Preparing to log off.\n");
                if (os == null || is == null){
                        textArea.appendText("An error has occured. Not logged on\n");
                        return super.handleEvent(evt);
                }
                os.println("##quit##");
                os.flush();
                try{
                        if (os != null)
                                os.close();
                        if (is != null)
                                is.close();
                        if (to_server != null)
                                to_server.close();
                }catch(IOException e){textArea.appendText("Error loggin off.\n");}
                textArea.appendText("Sucessfully logged off\n");
                user_list.setText("");  //clear the user list
                is_connected = false;  //no longer connected
        }
    if (evt.target == connect && !is_connected){ // 연결되지 않은 상태에서, 연결버튼을 누르면
                String     text=name.getText();
                int len = text.length();
                if(len != 0 ) {
                        textArea.appendText("Trying to connect\n");
                        try{
                                try{
                                to_server = new Socket(getCodeBase().getHost(),4444);
                                }catch(UnknownHostException e){
                                        textArea.appendText("Not     able     to     connect,
sorry.\n");
```

```
                                        return super.handleEvent(evt);
                                    }

                                    is = new DataInputStream(new
BufferedInputStream(to_server.getInputStream()));
                                    os = new PrintStream (new
BufferedOutputStream(to_server.getOutputStream(),1024), false);
                                } catch(IOException e){}
                                is_connected = true;  // 연결이 되었기에 참값
                                os.println("##name##" + text);
                                os.flush();
                                textArea.appendText("\nYour name is "'+text+'" right now. \n");
                        }
                }

                return super.handleEvent(evt);
        }

        /* 프로그램이 중단될 때, 각종 소켓과 스트림을 닫는다. */
        public void stop(){
                textArea.appendText("Preparing to log off.\n");
                if (os != null){
                        os.println("##quit##");
                        os.flush();
                }
                try{
                        if (os != null)
                                os.close();
                        if (is != null)
                                is.close();
                        if (to_server != null)
                                to_server.close();
                }catch(IOException e){textArea.appendText("Error loggin off.\n");}
                textArea.appendText("Sucessfully logged off\n");
                user_list.setText("");  // 등록 사용자 리스트를 지운다.
                is_connected = false;  // 연결상태를 거짓으로 한다.
        }

}
```

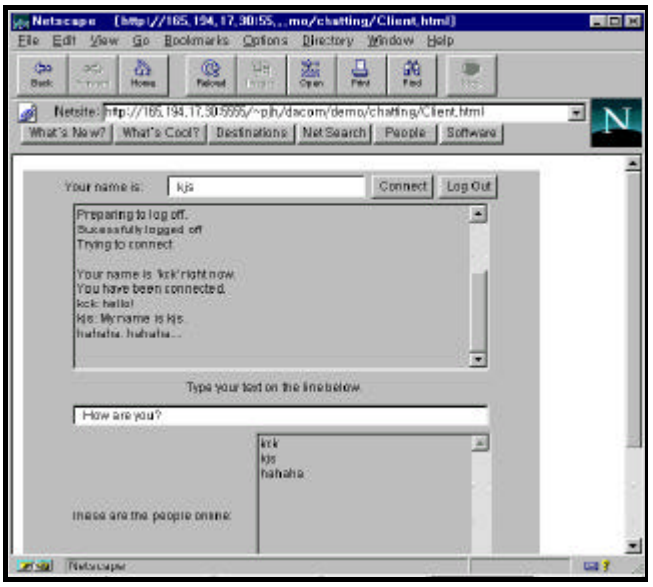
<예제> Client.java

위의 애플릿을 다음 HTML문서를 만들어 끼워넣는다.

```
<html>
<applet code = "Client.class", width = 500 height = 500>
</applet>
</html>
```

<예제> Client.html

웹브라우저가 이 HTML문서를 읽어 들이면, <그림 6>과 같은 모습을 갖게 된다.



<그림 6> 채팅하는 모습

위의 그림은 애플릿을 실행한 사용자의 이름은 "kck"이고, "kjs"와 "hahaha"란 다른 사용자와 대화하는 모습이다.

소스코드를 살펴보자.

사용자의 이벤트를 받아들이는 부분을 handleEvent안에 정의하였다. 그렇지만 GUI컴포넌트들에게 발생하는 이벤트는 action()을 사용할 수도 있고, 또한 그것을 선호하는 경향이 있다. 서버와 송수신을 하는 부분을 쓰레드화시킨 것을 유념하라. 이것은 GUI컴포넌트에서 발생하는 이벤트를 받아들이기 위한 방법이다.

이 애플릿이 제대로 동작하기 위해서는, 애플릿이 있는 웹서버상에서 서버프로그램이 실행되어 있어야 한다.

서버 프로그램은 채팅을 희망하는 클라이언트들의 정보를 등록하여, 한 클라이언트로부터 오는 문자열을 등록된 모든 다른 클라이언트들에게 전송한다.

서버 프로그램의 소스는 다음과 같다.

```
import java.net.*;
import java.io.*;
import java.lang.*;

class Server{
```

```
static int users = 0;   // 등록된 사용자의 수
static socket_stuff clients[] = new socket_stuff[29]; // 동시접속자 29명
 public static void main(String args[]) {
        ServerSocket serverSocket = null;
        int i;
   try {
     serverSocket = new ServerSocket(4444);
   } catch (IOException e) {
          System.out.println("Could not listen on port: " + 4444 + ", " + e);
     System.exit(1);
   }
        Socket clientSocket = null;
        connection user = new connection(serverSocket);

        if (users == 0){  // 등록된 사용자가 없으면, 기다린다.
        try {
         clientSocket = serverSocket.accept();
        } catch (IOException e) {
         //System.out.println("Accept failed: " + 4444 + ", " + e);
         System.exit(1);
        }
        add_client(new socket_stuff(clientSocket)); // 클라이언트를 등록한다.
        }
        String inputLine,outputLine; // 입출력을 위한 String 객체선언

        user.start();  // 새로운 사용자를 받아들이는 connection 쓰레드를 실행한다.

        try {
          inputLine = "Greetings user.";
          i = users; // 마지막 등록된 사용자의 차례
          int test;
          boolean flag = true;

          while (users >= 0)  { // 사용자가 1사람이상일 경우
           if (users>0 && i<=users ){
                test = clients[i].can_read();
                if (test == -1){ // 만약 클라이언트에게 송신할 상태가 안되면,
                 remove_client(i);  // 클라이언트를 제거
                }
                else if (test == 1){ // 만약 클라이언트로부터 데어터가 수신가능하면,
                 inputLine = clients[i].read_string();  // 문자열을 읽는다.

                 flag = process_line(inputLine,i); // 문자열의 종류에 따라 처리

                 if (flag){ // 만약 평범한 채팅문자열이면
                  if (clients[i] != null)
```

```
                    write_all(clients[i].name+": "+inputLine);  // 모든사용자에게 전송
                  }
                }

                }

                // 2 밀리세컨드 만큼, 쓰레드를 실행한다. 클라이언트와의 연결 파악
                try{
                user.join(2);
                } catch(InterruptedException e) {}

                if (user.connected() == true){ // 새로운 클라이언트가 연결을 시도했으면,
                        add_client(new socket_stuff(user.get_socket())); // 그 클라이언트를 등록
                        user.resume(); // 쓰레드를 다시 시작시킨다.
                }

                i++;
                if (i>users && users>=1)  //  등록된 사용자가 수신할 기회를 줌
                        i=1;
                if (i>users && users<1)
                        i=0;
               }

                serverSocket.close();
              }catch(IOException e){}
  }

 /** 등록된 모든 사용자에게 현재 사용자 리스트를 전송한다 */
 static void send_names(){ //sends the name list to every body
  String names = new String("##names##"); //사용자리스트임을 알리는 스트링
  int i;
  if (users < 1)  // 만약 사용자가 없으면,
    return;
  for (i=1;i<=users;i++)  // 현재 모든 사용자에게
   if (clients[i] != null) //if everthing is in order
          names = names + clients[i].name + "|";  // 이름과 구분자를 덧붙인다.
  write_all(names);  // 모든 사용자에게 사용자 리스트를 보낸다.
 }

 /** 클라이언트가 보낸 문자열의 종류에 따라 처리한다. 일반 채팅문자열, 등록할 사용자 이름,
* "Log out" 버튼을 눌렀을 때의 문자열에 따라 처리한다.
  */
 static boolean process_line (String line,int i){
  //false if don't echo line
  if (line.startsWith("##name##")){  // 만약 등록 사용자의 이름이 보내졌으면,
```

```
      clients[i].name = line.substring(8);  // 이름을 할당
//     clients[i].write_string("Name changed to "+line.substring(8));
      send_names(); // 갱신된 사용자 리스트를 전송한다.
      return false;
    }
    else if (line.equals("##quit##")){ // 만약 사용자가 "Log out"버튼을 눌렀으면,
      write_all(clients[i].name+" is logging out."); // 모든 사용자에게 "Logout"을 알린다.
      remove_client(i);
      return false;
    }

    return true; // 일반 채팅 문자열면,
  }

  /** 모든 클라이언트에게 문자열을 송신한다. */
  static void write_all (String text){
    int i;

    for (i=1;i<=users;i++)
      clients[i].write_string(text);
  }

  /** 접속을 시도하는 클라이언트를 등록한다 */
  static  void add_client(socket_stuff client){
    if (users >= 25){ // 만약 사용자가 25명 이상이면,
      client.write_string("Sorry, there are too many users right now.");
      client.kill();
      return;
    }
    clients[++users] = client;  // 현재 등록된 사용자의 다음 배열원소로 사용자를 등록

    clients[users].write_string("You have been connected."); //등록된 사용자에게 연결을 알림
    send_names();  // 등록된 모든 사용자에게 갱신된 사용자 리스트를 전송한다.
  }

  /** 등록된 사용자의 등록을 해제한다. */
  static  void remove_client(int client){
    clients[client].kill();  // 스트림과 소켓을 닫는다.
    clients[client]=clients[users]; // 마지막에 등록된 사용자를 해제된 사용자의 배열원소로 할당
    clients[users]=null; // 마지막 등록된 사용자의 배열원소에 널을 할당
    users--; // 사용자의 수가 1만큼 감소한다.
    send_names();  // 다시 갱신된 사용자를 모든 사용자에게 전송한다.
  }

}
```

```
/** 클라이언트들이 접속을 시도할 때, 각 클라이언트에게 송수신을 하는 소켓을 생성하는 쓰레
 * 드. 클라이언트들이 접속을 시도하는 지를 계속 감시하다가, 클라이언트와의 소켓을 생성.
 */
class connection extends Thread{
 Socket socket = null;  // 클라이언트와의 소켓
 public boolean is_connected = false;  // 연결상태
 ServerSocket server = null;

 connection (ServerSocket server_sock){
  server = server_sock;
 }

 boolean connected(){
  return is_connected;
 }
 /** 접속이 이미 된 소켓을 반환하다. */
 Socket get_socket(){
  System.out.println("Returning the socket");
  is_connected = false;
  return socket;

 }
public void run(){
 while (true){ // 무한히 반복
  is_connected = false;
  //System.out.println("In run Method.");

  // 한 클라이언트와 연결되면, 그것이 add_client()로 등록될 때까지 다른 클라이언트와 연결  //
을 미룬다.
    if (!is_connected)
    {
            //System.out.println("Trying for new connection");
            try{
             socket = server.accept(); // 접속하려는 클라이언트를 감시
            } catch(IOException e) {
             //System.out.println("Problem with accept");
            }
            //System.out.println("Received new connection");
            is_connected = true;
            this.suspend(); // 클라이언트와 연결된 후, 정식 등록될 때까지 쓰레들 실행을 중단한다.
    }
 }

}
}
```

```
/** 각 클라이언트와의 송수신을 전담하는 클래스 */
class socket_stuff{
  Socket socket;
  DataInputStream is;
  PrintStream os;
  String name = new String("Guest");  // 사용자의 디폴트 이름

  socket_stuff(Socket client){
    //System.out.println("getting the socket info");
    //get the streams
    try{
      is = new DataInputStream(new BufferedInputStream(client.getInputStream()));
      os = new PrintStream (new BufferedOutputStream(client.getOutputStream(), 1024), false);
      socket = client;
   } catch(IOException e){}
  }

/** 클라이언트에게 문자열을 송신한다 */
public void write_string (String s){
  //System.out.println ("Writing to socket");
  os.println(s);  // 소켓을 통해 전송
  os.flush();
}

/** 클라이언트로부터 문자열을 수신한다. */
public String read_string(){
  //System.out.println("Reading from socket");
  try{
    String inputLine = is.readLine(); // 클라이언트로 한 줄을 읽는다
    return inputLine;
  }catch(IOException e){}
  return "null";
}

/** 소켓으로부터 데이터를 읽을 수 있는지의 상태를 반환 */
public int can_read(){
  boolean test;

  if (os.checkError()){  // 클라이언트에게 송신할 수 없을 경우,
    System.out.println("os.checkError");
    return -1;
  }
  try{
  test = (is.available()>1);  // 만약 읽을 수 있는 상태면 참
  }catch(IOException e){ return -1; }
```

```
  if (test)   // 읽을 수 있으면 1을 반환
    return 1;
  else
    return 0; // 읽을수 없으면 0을 반환

}

/** 스트림과 소켓을 닫는다. */
public void kill(){
  try{
    if (os != null)
      os.close();
    if (is != null)
      is.close();
    if (socket != null)
      socket.close();
  }catch (IOException e){}
}
}
```

<예제> Server.java

위의 소스는 Server, connection, socket_stuff.등 총 3개의 클래스로 구성이 되어 있다.

Server클래스는 메인 클래스로서, connection와 socket_stuff클래스의 객체를 생성한다. 주된 일은 접속하는 클라이언트가 있는지 파악하여, 등록시키고 한 클라이언트가 보낸 문자열을 검사하여 채팅문자열이면 등록된 다른 클라이언트에게 전송하고, 그외 다른 문자열("quit"등)이면 대응되는 처리를 하는 것이다.

connection 클래스는 쓰레드로서, 클라이언트가 접속을 시도하는지를 계속 감시하다가, 연결을 시도하면 해당하는 소켓을 생성하는 임무를 갖고 있다. 이 쓰레드는 소켓을 생성한 후, 온라인 사용자로 등록될 때까지, 접속을 시도하는 다른 클라이언트의 소켓을 생성하지 않는다.

socket_stuff클래스는 연결된 클라이언트와의 송수신을 하는 메쏘드를 가지고 있어서, 메인 클래스에서 각 클라이언트에게 송수신할 필요가 있을 때 해당서비스를 제공한다.

바둑 프로그램

웹상에서 바둑을 둘 수 있는 시스템을 만들어보자. 이것은 위에서 설명한 채팅 프로그램을 바탕으로 하고 있다. 참고로 본 시스템은 프로토타입(prototype)이기 때문에 완전하지는 못하고, 약간의 버그가 있을 수 있다.

바둑 시스템은 기본적으로 서버프로그램과 자바애플릿으로 구성된다. 각자의 역할은 다음과 같다.

1) 서버프로그램

- 대국을 원하는 사용자들을 네트웍연결을 통하여 등록한다.
- 대국하는 두 사용자들간의 통신을 중재한다. 즉 한 사용자가 바둑판의 특정 위치에 바둑알을 놓으면, 그 정보를 상대방에게 전달하여 상대방의 바둑판에도 바둑알이 나타나도록 한다. 그 외에도 서로간의 채팅도 중재한다.
- 웹서버상에서 항상 실행되어 클라이언트(자바애플릿)들이 통신을 시도하면 연결해야 한다.

2) 자바애플릿

- 브라우저상에서 바둑판모양을 사용자에게 보여주고, 서로간의 채팅을 할 수 있는 윈도우를 생성한다.
- 사용자가 마우스로 바둑판위의 특정 위치에 바둑알을 놓으면, 서버로 그 정보(위치 등)를 보낸다. 만약 서버로부터 상대방의 바둑알 위치정보가 오면, 해당 위치에 바둑알을 표현한다. 그리고 각종 마우스입력을 처리한다.
- 서로간의 채팅을 지원하는 윈도우를 통해서, 대국하는 도중에 의사소통을 할 수 있다.

본 바둑시스템을 설치하기 위해서 다음 단계를 밟는다.

1) 웹서버상에서 데몬으로 서버프로그램을 실행시킨다. 유닉스일 경우, “java ServerBaduk &" 을 실행시킨다.

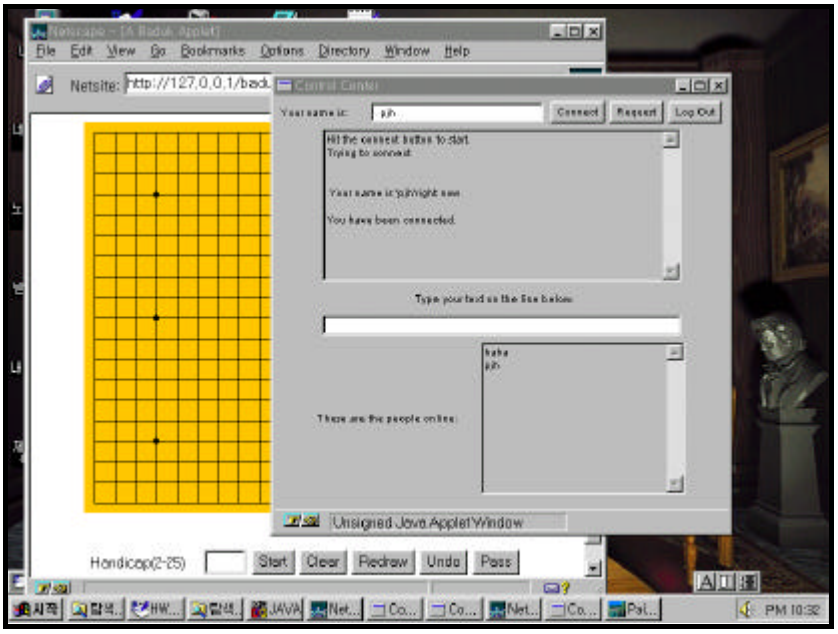
2) 자바애플릿인 "Baduk.class"를 다음 HTML문서에 삽입한다.

```
<html>
<head>
<title> A Baduk Applet</title>
</head>
<body bgcolor="FFFFFF">
<center>
<h2> 본 애플릿으로 바둑을 두실 수 있습니다. 그럼 행운을!</h2>
<applet code="Baduk.class" width=480 height=470>
</applet>
</center>
</body>
</html>
```

<예제> Baduk.html

자! 그럼 바둑을 두기 위한 실행순서를 자세히 살펴보자.

어떤 사용자가 "Baduk.html"을 브라우저로 접속하면, 웹 브라우저상의 바둑판 모양과 채팅을 할 수 있는 윈도우(이하 채팅윈도우)가 생성이 된다. 사용자는 채팅윈도우를 통해 사용자 ID를 등록해야 한다. "pjh"라는 ID로 등록했다고 가정하면, 나중에 이 사용자 ID로 대국 상대방을 선택하게 된다.(<그림 1> 참조)



<그림 1> "pjh"라는 사용자 ID를 등록한 후의 모습

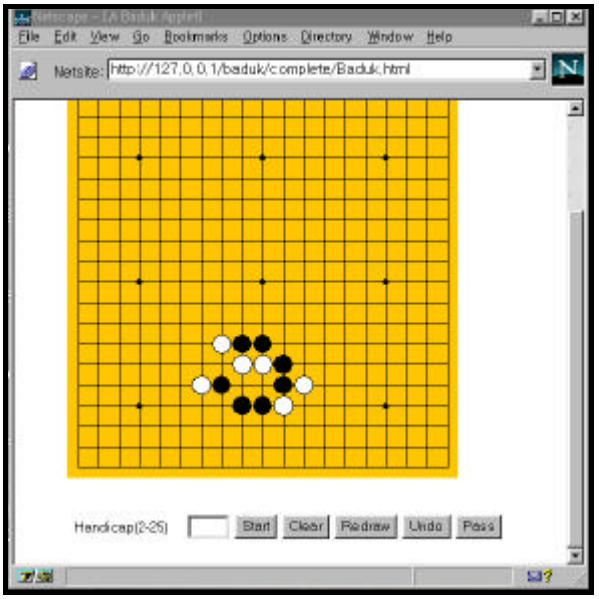
등록된 사용자 중 "haha"와 대국을 하길 원하면, 채팅윈도우의 "Request"버튼을 누른다. 그러면 대국을 원하는 상대방 ID와 자신의 급수를 입력하는 <그림 2>와 같은 윈도우가 생성이 되는데, 그 안에 "haha"와 자신의 급수를 집어넣은 후, "Ok"버튼을 누른다.

<그림 2> "Request"버튼을 누른 후, 생성되는 윈도우

그러면, 웹상의 다른 "haha"라는 사용자의 화면에 <그림 3>과 같이 "pjh"라는 사용자가 대국을 요청했음을 알리는 윈도우가 생성이 된다.

<그림 3> "haha"라는 사용자의 화면에 생성되는 윈도우

이 때, "haha"가 "OK" 버튼을 누르면, 대국을 허락하는 의미가 되어 바둑을 둘 수 있게 된다. 먼저 바둑대국을 신청한 "pjh"는 애플릿의 "Start"버튼을 누르고, 원하는 위치에 마우스를 누름으로서 <그림 4>처럼 본격적인 바둑대결을 하게 된다. 바둑돌의 색깔은 바둑대결을 신청한 사람이 검은돌이다.



<그림 4> "Start"버튼을 누른 후, 대국하는 모습

대국 중간에 서로 대화를 나누고 싶으면, 채팅윈도우에 문자열을 쓰면 된다.(채팅 프로그램 참고). 만약 중간에 바둑을 그만두고 싶으면, 채팅윈도우의 "Log out"버튼을 눌러야 한다.

자바애플릿의 소스는 다음과 같다.

```
import java.applet.Applet;
import java.awt.*;
import java.io.*;
import java.net.*;


public class Baduk extends Applet implements BadukConstant{
   Panel p,s;
   public Inframe f=new Inframe(this);
   public BadukLogic bl = new BadukLogic(this,f);
   public void init(){
      setBackground(Color.white);
      p = new Panel();
            /* 바둑판의 상태를 조절하는 버튼들이 붙어있는 Panel생성 */
```

```
    p.setLayout(new FlowLayout());
    p.add(new Label("Handicap(2-25)"));
    bl.t = new TextField(3);
      /* 접바둑시 깔고 둘 돌의 수를 입력 받기 위한 TextField를 잡는다. */
    p.add(bl.t);
    p.add(new Button("Start"));
    p.add(new Button("Clear"));
    p.add(new Button("Redraw"));
    p.add(new Button("Undo"));
    p.add(new Button("Pass"));
       /* 버튼들을 만든다. */
    setLayout(new BorderLayout());
    add("North", bl);
    add("South", p);
    resize(480, 460);
        /* 바둑판과 Panel을 화면상에 위치시킨다.*/
    bl.f = new Font("TimeRoman", Font.PLAIN, 17);
    bl.offImage=createImage(this.size().width, this.size().height);
    bl.offGraphics=bl.offImage.getGraphics();
       /* paint시에 더블버퍼링을 사용하기 위해서 image을 잡는다. */
    bl.InitPlace();
    bl.InitDirection();
}
/* 바둑 돌의 위치를 계산한다. */
public void convert(String posi){
                    int X,Y;
                    int index;
                    int index_1;
                    String t_xy;
                    String temp;
                    temp=posi.substring(8);
                    index=temp.indexOf(",");
                    index_1=temp.indexOf("/");
                    t_xy=temp.substring(0,index);
                    X=Integer.valueOf(t_xy).intValue();
                    t_xy=temp.substring(index+1,index_1);
                    Y=Integer.valueOf(t_xy).intValue();
                    bl.input_index(X,Y);
                    if(f.Turn ==1) f.Turn=0;
                    else f.Turn = 1;
                    repaint();

}
public boolean mouseUp(Event e, int x, int y){
 /* 바둑판위에서 마우스 버튼이 눌려졌을때 그 위치를 체크해서 그에 맞는 실행을 한다. */
    bl.mouseUp(e,x,y);
    return true;
```

```
}
public void paint(Graphics g){
   bl.paint(g);
}
public void update(Graphics g){
/* paint시에 화면이 깜박거림을 없애기 위해서 update함수를 override시켰다. */
   paint(g);
}
public boolean action(Event evt, Object arg){
    /* 어느 버튼이 눌려졌는지를 체크한다. */
   if(evt.target instanceof Button)
      MenuAct((String)arg);
   return true;
}
        /* 서버로부터 온 명령을 수행한다. */
public void execute_command(String MenuName){
   if(MenuName.equals("Clear/")){
       /* Clear버튼이 눌려졌을때 바둑판에 관한 모든 정보를 초기화 시킨다. */
      bl.InitPlace();
      bl.InitCheck();
      bl.InitDirection();
      bl.Turn=Black;
      bl.Order=0;
      bl.CaptureByWhite=0;   bl.CaptureByBlack=0;
      bl.HandiFlag=false;
      for(int i=0;i<360;i++)
         for(int j=0;j<4;j++)
            bl.DieDirection[i][j]=false;

                        f.flag=0;
                                f.Turn =0;
      repaint();
   }
   else if(MenuName.equals("Redraw/"))
       /* Redraw버튼이 눌려지면 단순히 화면을 다시 그려준다. */
                    repaint();
   else if(MenuName.equals("Undo/")){
      /* Undo버튼이 눌려지면 전에 놓여졌던 바둑알을 없애고 순서를 바꾸어준다. */
      if(bl.Order>0) bl.MenuUndo();
      repaint();
                                if(f.Turn ==1) f.Turn = 0;
                                else f.Turn =1;
   }
   else if(MenuName.startsWith("Start/")){
      /* 바둑을 시작하기 위해서 Start버튼은 항상 제일 먼저 눌려져야 한다. */
                    String temp=new String();
                    temp=MenuName.substring(6);
```

```
            if(bl.HandiFlag==false){
                    bl.HandiFlag=true;
                if(bl.Handicap(temp)==false){
                                        if(f.flag == 1) f.Turn =1;
                                }
          /* 접바둑의 경우에는 입력에 따라 검은돌이 놓여지고 흰돌먼저 놓게 된다. */
                }
          repaint();
        }
        else if(MenuName.equals("Pass/")){
                /* Pass 버튼이 눌려지면 순서가 넘어간다. */
          for(int i=1;i<20;i++)
             for(int j=1;j<20;j++)
               if(bl.Place[i][j]>Pae) bl.Place[i][j]--;
         /* Pass버튼이 눌려지면 패였던 위치가 다시 풀리게 된다. */
          if(bl.Turn==Black) bl.Turn=White;
          else bl.Turn=Black;
          repaint();
                                if(f.Turn==1) f.Turn =0;
                                else f.Turn = 1;
        }
      }

            /* 사용자가 바둑판에 있는 버튼을 눌렀을 때, 해당하는 기능을 수행한다. */
            public void MenuAct(String MenuName){
                    String temp=new String();

                    if(f.play == true) {
                            if(MenuName.equals("Clear"))
                                    f.send_string("command/Clear/");
                            else if(MenuName.equals("Redraw"))
                                    f.send_string("command/Redraw/");
                            else if(MenuName.equals("Undo"))
                                    f.send_string("command/Undo/");
                            else if(MenuName.equals("Start")){
                                    f.flag =1;
                                    temp=bl.t.getText();
                                    f.send_string("command/Start/"+temp);
                            }
                            else if(MenuName.equals("Pass"))
                                    f.send_string("command/Pass/");
                    }
             }

}

/* 바둑판을 상징하며, 이곳에서 일어나는 이벤트들을 직접 처리 */
```

```
class BadukLogic extends Panel implements BadukConstant{
        Inframe fa;
  public static int Turn=Black;
  public static boolean Die=false;
  public static int[][] Place = new int[21][21];
    /* 바둑판에 관한 정보를 가지고 있는 배열 */
  public static boolean[][] Check = new boolean[21][21];
    /* 사석을 처리하기 위해서 필요한 배열 */
  public static int[][] StoneOrder = new int[360][2];
    /* 수순을 기록하기 위한 배열 */
  public static boolean[][] DieDirection = new boolean[360][4];
    /* Undo시에 먹혔던 돌을 되살리기 위해서 필요한 배열 */
  public static int Order=0;
  public static int CaptureByWhite=0, CaptureByBlack=0;
    /* 사석의 수를 가지고 있는 변수 */
  public Image offImage;
  public Graphics offGraphics;   /* 더블 버퍼링을 위해서 사용된다. */
  public boolean HandiFlag=false;
  public Font f;
  public TextField t;
  Baduk app;

        public BadukLogic(Baduk app,Inframe fa){
                this.app=app;
                this.fa=fa;
        }

        public boolean mouseUp(Event e, int x, int y){
    /* 마우스 버튼이 눌려 졌을때 바둑돌을 놓는 다거나 하는 일을 처리하는 함수 */
        int X, Y;
        /* 바둑판위에서 마우스가 눌려졌을때 바둑판배열을 인덱스 값으로 바꾸어 준다. */
                if(fa.play==true && fa.Turn ==1)
                        if(x>25 && x<405 && y>25 && y<405){
                                String position=new String("position");
                                X=(x-(5+20))/Wide+1;
                                Y=(y-(5+20))/Wide+1;
                                position =position+ X+","+Y+"/";
                                fa.send_string(position);
                        }
                return true;
  }

  public boolean input_index(int X ,int Y )
  {
   /* 마우스가 눌려 졌을때 그 위치에 바둑알이 놓여 질수 있는 위치인지를 먼저 확인한다.*/
            if(Place[X][Y]<Order && Place[X][Y] > Pae)
                Place[X][Y]=Empty;
```

```
            if(Place[X][Y]==Empty){
                    if(Turn==Black){
                Place[X][Y]=Black;
                    Turn=White;
                }
                    else if(Turn==White){
                Place[X][Y]=White;
                    Turn=Black;
                }
            CheckPoint(X,Y);
  /* 바둑알이 놓여 질수 있으면 그 돌을 놓음으로써 죽는 돌이 있는지를 체크한다. */
            }
            return true;
           }

  public void InitPlace() {
/*처음 바둑이 시작될때 모든 바둑판의 위의 점들은 Empty이고 가장자리는 Bound로 초기화 된다.
*/
    for(int i=0;i<21;i++)
      for(int j=0;j<21;j++)
        Place[i][j]=Empty;

   for(int k=0;k<21;k++){
      Place[0][k]=Bound;
      Place[20][k]=Bound;
      Place[k][0]=Bound;
      Place[k][20]=Bound;
    }
  }

  public void InitCheck(){
    Die=false;
    for(int i=0;i<21;i++)
      for(int j=0;j<21;j++)
        Check[i][j]=false;
  }
  public void InitDirection(){
    for(int i=0;i<360;i++){
      DieDirection[i][0]=false;
      DieDirection[i][1]=false;
    }
  }
  public void paint(Graphics g){
    /* 바둑판과 돌들을 그려주는 메소드 */
    offGraphics.setColor(Color.orange);
    offGraphics.fillRect(5+20,5+20,380,380);
```

```
        offGraphics.setColor(Color.black);
        offGraphics.setFont(f);
        for(int i=0;i<19;i++){
        offGraphics.drawLine(Boundary+i*Wide,Boundary,Boundary+i*Wide,Boundary+18*Wide);
        offGraphics.drawLine(Boundary,Boundary+i*Wide,Boundary+18*Wide,Boundary+i*Wide);
        }
        for(int p=0;p<3;p++)
          for(int q=0;q<3;q++)
            offGraphics.fillOval(72+p*Wide*6+20,72+q*Wide*6+20,6,6);
        for(int a=1;a<20;a++)
          for(int b=1;b<20;b++){
            if(Place[a][b]==White){
              offGraphics.setColor(Color.white);
         offGraphics.fillOval(Boundary+(a-1)*Wide-9,Boundary+(b-1)*Wide-9,Wide-2,Wide-2);
              offGraphics.setColor(Color.black);
        offGraphics.drawOval(Boundary+(a-1)*Wide-9,Boundary+(b-1)*Wide-9,Wide-2,Wide-2);
            }
            else if(Place[a][b]==Black){
              offGraphics.setColor(Color.black);
         offGraphics.fillOval(Boundary+(a-1)*Wide-9,Boundary+(b-1)*Wide-9,Wide-2,Wide-2);
            }
            else if(Place[a][b]==BHouse){
              offGraphics.setColor(Color.black);
        offGraphics.drawRect(Boundary+(a-1)*Wide-7,Boundary+(b-1)*Wide-7,Wide-6,Wide-6);
            }
            else if(Place[a][b]==WHouse){
              offGraphics.setColor(Color.white);
        offGraphics.drawRect(Boundary+(a-1)*Wide-7,Boundary+(b-1)*Wide-7,Wide-6,Wide-6);
            }
          }
        g.drawImage(offImage, 0, 0, this);  /* 더블버퍼링을 사용해서 화면상이 image을 뿌려준다. */
      }
      public void CheckPoint(int x,int y){
          /* 바둑판위에 바둑알을 놓음으로해서 네 방향에 죽은 돌이 있는지 그리고 자신이 죽지는 않
는지를 검사하는 메소드 */
        boolean Kill=false;
        boolean Wook=false;
        int Count=0;
        int Turn2;
        if(Turn==Black){
          Turn=White;
          Kill=CheckStone(x,y);
          Turn=Black;
        }
        if(Turn==White){
          Turn=Black;
          Kill=CheckStone(x,y);
```

```
            Turn=White;
        }
        if(Place[x-1][y]==Turn){
            Die=CheckStone(x-1,y);
            if(Die){
                Count=Died();
                DieDirection[Order][0]=true;
            }
            if(!Wook) Wook = Die;
        }
        if(Place[x+1][y]==Turn){
            Die=CheckStone(x+1,y);
            if(Die){
                Count=Died();
                DieDirection[Order][1]=true;
            }
            if(!Wook) Wook = Die;
        }
        if(Place[x][y-1]==Turn){
            Die=CheckStone(x,y-1);
            if(Die){
                Count=Died();
                DieDirection[Order][2]=true;
            }
            if(!Wook) Wook = Die;
        }
        if(Place[x][y+1]==Turn){
            Die=CheckStone(x,y+1);
            if(Die){
                Count=Died();
                DieDirection[Order][3]=true;
            }
            if(!Wook) Wook = Die;
        }
        if(Count==1) PaeCheck(x,y);
        if(Kill && !Wook){
            Place[x][y]=Empty;
            if(Turn==Black) Turn=White;
            else Turn=Black;
        }
        else{    /* 수순을 기록하는 배열에 기록한다. */
            StoneOrder[Order][0]=x;
            StoneOrder[Order++][1]=y;
        }
    }
    public int Died(){  /*죽은 돌이 생길경우에는 그 돌의 수를 세어서 사석의 수에 더한다. */
        int Count=0;
```

```
    for(int i=1;i<20;i++)
      for(int j=1;j<20;j++)
        if(Check[i][j]){
          if(Turn==Black) CaptureByWhite++;
          else CaptureByBlack++;
          Count++;
          Place[i][j]=Empty;
        }
    return Count;
  }
  public boolean CheckStone(int x,int y){
    /* 직접 사석인지를 검사하는 메소드 */
    int Count=0;
    int[][] Stack = new int[50][2];
    boolean Test=false;
    InitCheck();
    Check[x][y]=true;
    Stack[Count][0]=-1;    Stack[Count][1]=-1;
    Stack[++Count][0]=x;    Stack[Count][1]=y;
    while((Count != 0) || (!Test)){
      /* 네방향중 한곳이라도 빈 곳이 있으면 이 돌들은 죽은 돌이 아니다. */
      /* Pae라고 되어 있는 곳도 역시 빈곳이므로 이 돌들은 죽은 돌이 아니다. */
      if((Place[x-1][y] == Empty) || (Place[x+1][y] == Empty) ||
        (Place[x][y-1] == Empty) || (Place[x][y+1] == Empty) ||
        (Place[x-1][y] > Pae) || (Place[x+1][y] > Pae) ||
        (Place[x][y-1] > Pae) || (Place[x][y+1] > Pae))
          return false;
      if((Place[x][y-1]==Turn) && (Check[x][y-1]==false)){
          y--;
          Stack[++Count][0]=x;    Stack[Count][1]=y;
          Check[x][y]=true;
          continue;
      }
      if((Place[x][y+1]==Turn) && (Check[x][y+1]==false)){
          y++;
          Stack[++Count][0]=x;    Stack[Count][1]=y;
          Check[x][y]=true;
          continue;
      }
      if((Place[x-1][y]==Turn) && (Check[x-1][y]==false)){
          x--;
          Stack[++Count][0]=x;    Stack[Count][1]=y;
          Check[x][y]=true;
          continue;
      }
      if((Place[x+1][y]==Turn) && (Check[x+1][y]==false)){
          x++;
```

```
            Stack[++Count][0]=x;    Stack[Count][1]=y;
            Check[x][y]=true;
            continue;
          }
          Count--;
          x=Stack[Count][0];   y=Stack[Count][1];
          if((x==-1) && (y==-1)) Test=true;
        }
        return true;
      }
      public void MenuUndo(){
/* Undo버튼이 눌려졌을때 수순을 기록하는 함수에서 하나를 물려주고 바둑판을 다시 그려준다.
*/
        int x, y;
        x = StoneOrder[--Order][0];
        y = StoneOrder[Order][1];
        if(DieDirection[Order][0]==true) AliveStone(x-1,y);
        if(DieDirection[Order][1]==true) AliveStone(x+1,y);
        if(DieDirection[Order][2]==true) AliveStone(x,y-1);
        if(DieDirection[Order][3]==true) AliveStone(x,y+1);
       /* Undo시 죽은 돌을 되살린다. */
        for(int i=0;i<4;i++)
          DieDirection[Order][i]=false;
        Place[x][y]=Empty;
        if (Turn==Black) Turn=White;
        else Turn=Black;
        repaint();
      }
      public void AliveStone(int x, int y){
       /* Undo시에 죽었던 돌들을 되살리는 메소드 */
        int Count=0;
        int[][] Stack = new int[50][2];
        boolean Test=false;
        Stack[Count][0]=-1;    Stack[Count][1]=-1;
        Stack[++Count][0]=x;    Stack[Count][1]=y;
        Place[x][y]=Turn;
        UndoStone();
        while((Count != 0) || (!Test)){
          if(Place[x][y-1]==Empty){
            y--;
            Stack[++Count][0]=x;   Stack[Count][1]=y;
            UndoStone();
            Place[x][y]=Turn;
            continue;
          }
          if(Place[x][y+1]==Empty){
            y++;
```

```
            Stack[++Count][0]=x;    Stack[Count][1]=y;
            UndoStone();
            Place[x][y]=Turn;
            continue;
        }
        if(Place[x-1][y]==Empty){
            x--;
            Stack[++Count][0]=x;     Stack[Count][1]=y;
            UndoStone();
            Place[x][y]=Turn;
            continue;
        }
        if(Place[x+1][y]==Empty){
            x++;
            Stack[++Count][0]=x;     Stack[Count][1]=y;
            UndoStone();
            Place[x][y]=Turn;
            continue;
        }
        Count--;
        x=Stack[Count][0];    y=Stack[Count][1];
        if((x==-1) && (y==-1)) Test=true;
    }
}
public void UndoStone(){
    /* Undo 시에 되살아 나는 돌들이 있으면 그 수만큼 사석의 수에서 빼준다. */
    if(Turn==Black) CaptureByWhite--;
    else CaptureByBlack--;
}
public boolean Handicap(String temp){
    /* 접바둑시에 사용되는 메소드 */
    String[] Number = {"2","3","4","5","6","7","8","9","10","11","12","13",
                       "14","15","16","17","18","19","20","21","22","23",
                       "24","25"};
    String In = temp;
    int[][] Handi = {{4,4},{16,16},{4,16},{16,4},{4,10},
                     {16,10},{10,4},{10,16},{10,10},{4,7},
                     {16,13},{7,16},{13,4},{4,13},{16,7},
                     {7,4},{13,16},{7,10},{13,10},{10,7},
                     {10,13},{7,7},{13,13},{7,13},{13,7}};
        /* 접바둑이 돌이 놓여질 위치를 미리 가지고 있는 배열이다. */
    int i=0;
    while(!In.equals(Number[i])){
    /* 입력된수가 위의 배열에서 일치하는 수가 있는지를 찾는다. */
        i++;
        if(i==24)
                                    return false;
```

```
                    }


                    if(fa.flag==0) fa.Turn =1;
                        i+=2;
        for(int a=0;a<i;a++)
          Place[Handi[a][0]][Handi[a][1]]=Black;
        Turn=White;
        repaint();
        return true;
      }
      public void PaeCheck(int x,int y){
        /* 한점을 먹었을때 그 위치가 패가 되는 지를 검사한다. */
        int c=0;
        int X=0, Y=0;
        if(Place[x-1][y]==Turn || Place[x-1][y]==Bound) c++;
        else{
          X=x-1;
          Y=y;
        }
        if(Place[x+1][y]==Turn || Place[x+1][y]==Bound) c++;
        else{
          X=x+1;
          Y=y;
        }
        if(Place[x][y-1]==Turn || Place[x][y-1]==Bound) c++;
        else{
          X=x;
          Y=y-1;
        }
        if(Place[x][y+1]==Turn || Place[x][y+1]==Bound) c++;
        else{
          X=x;
          Y=y+1;
        }
        if(c==3){
          Place[X][Y]=Order+1;
        }
      }
}

/* 대국을 요청했을 때, 상대방이 승락을 하면 그것을 알려주는 윈도우 */
class Ok extends Frame
{
      Label ok;
      Button okay;
      Inframe f;
```

```
        String name = new String();

        public Ok(Inframe f,String text)
        {
              this.f = f;
              int index;
              index = text.indexOf("/");
              setFont(new Font("Helvetica",Font.PLAIN, 10));
              setLayout(new FlowLayout());
                        super.setTitle("Are you ready?");
             ok = new Label(text.substring(0,index) + " is answered Yes for your request");
              name = text.substring(0,index);
              okay = new Button("Okay");
              add(ok);
              add(okay);
              validate();
              resize(200,100);
              show();
        }

        public boolean action(Event evt, Object arg)
        {
              if(arg.equals("Okay"))
              {
                f.send_string("play/" + name +"/");
                 hide();
              }
                        else
                          return false;
                        return true;
        }
}

/* 대국을 요청했을 때, 상대방이 거부를 하면 그것을 알려주는 윈도우 */
class No extends Frame
{
        Label no;
        Button okay;

               public No(String text)
        {
              int index;
              index = text.indexOf("/");
              setFont(new Font("Helvetica",Font.PLAIN, 10));
              setLayout(new FlowLayout());
              no = new Label(text.substring(0,index) + "answered No for your request");
                        super.setTitle("Hm...");
```

```
            okay = new Button("Okay");
            add(no);
            add(okay);
            validate();
            resize(200,100);
            show();
        }

        public boolean action(Event evt, Object arg)
             {
            if( arg.equals("Okay"))
              hide();
                        else
                  return false;
            return true;
        }
}
/* 대국을 요청한 상대방의 정보를 보여주는 윈도우 */
class Request_dialog extends Frame
{
        Inframe f;
        Label name ;
        Label level;
        TextField name_field;
        TextField level_field;
        Button okay;
        Button no;
        public Request_dialog(Inframe f)
        {
            this.f=f;
            setFont(new Font("Helvetica",Font.PLAIN, 10));
            setLayout(new FlowLayout());
                        super.setTitle("Please consider...");
            name= new Label  ("Requester      :");
            level = new Label("Requester level :");
            name_field= new TextField(25);
            level_field = new TextField(25);
            okay = new Button("Okay");
            no = new Button ("No");
            add(name);
            add(name_field);
            add(level);
            add(level_field);
            add(okay);
            add(no);
            validate();
            resize(290,140);
```

```
            show();
        }

            public void print_request(String text)
        {
            int i,j;
            String temp1=new String();
            String temp2=new String();
            i=text.indexOf("/");
            temp1=text.substring(0,i);
            j=text.indexOf("/",i+1);
            temp2=text.substring(i+1,j);
            name_field.setText(temp2);
            level_field.setText(temp1);
        }

            public boolean action(Event evt, Object arg)
        {
            if( arg.equals("Okay"))
            {
                String temp;
                temp = name_field.getText();
                f.send_string("okay/" + temp + "/");
                hide();
            }
                        else if( arg.equals("No"))
            {
                String temp;
                temp =name_field.getText();
                f.send_string("no/" + temp + "/");
                hide();
            }
                        else
                                return false;
                        return true;
        }
}
```

/* 대국하기로 선택한 상대방에게 자신의 정보(이름, 급수)를 보내기 위한 윈도우 */

```
class dilog extends Frame
{
    Inframe f;
    Label want;
    Label level;
    TextField want_name;
    TextField my_level;
    Button okay;
```

```
Button no;

public dilog(Inframe f)
{
      this.f = f;
      setFont(new Font("Helvetica",Font.PLAIN, 10));
      setLayout(new FlowLayout());
                  super.setTitle("Please response...");
      want = new Label("Want name     : ");
      level = new Label("Your level(1-18)");
      want_name = new TextField(25);
      my_level = new TextField(25);
      okay = new Button("Okay");
      no = new Button("No");
      add(want);
      add(want_name);
      add(level);
      add(my_level);
      add(okay);
      add(no);
      validate();
      resize(290,140);
      show();
}

public boolean action(Event evt, Object arg)
       {
      if( arg.equals("Okay"))
            send_request();
      else if(arg.equals("No"))
            send_no();
                  else
                              return false;
                  return true;
           }

public void send_request()
{
      String text = new String("request");
      String name=want_name.getText();
      String level=my_level.getText();
      if(name.length() == 0 || level.length() == 0)
            return;
      text=text + "/" +name + "/"+ level + "/";
      f.send_string(text);
      hide();
}
```

```
        public void send_no()
        {
            String text = new String("no/");
            hide();
        }

}
/* 바둑을 두기 위해, 서로간의 통신을 전담하는 채팅윈도우. */
class Inframe extends Frame implements Runnable
{
        dilog request_frame;
                int Turn =0;
                int flag =0;
        TextField input_field;
                boolean play =false;
        boolean is_connect=false;
        TextField name;
        TextArea talk;
        TextArea talk_name;
        Button connect;
        Button quit;
        Button match;
        Label namelabel;
        Label talklabel;
        Label userlabel;
        Socket server;
        DataInputStream is;
                        PrintStream os;
        Thread thread;
        Baduk app;
        Request_dialog rd;

        public Inframe(Baduk app){
                        this.app = app;
                        setFont( new Font("Helvetica",Font.PLAIN,10));
                        setLayout(new FlowLayout());
                        super.setTitle("Control Center");
                input_field = new TextField(55);
                name = new TextField(25);
                talk = new TextArea(10,55);
                talk_name= new TextArea(10,30);
                talk.setEditable(false);
                talk_name.setEditable(false);
                quit = new Button("Log Out");
                match = new Button("Request");
                connect = new Button("Connect");
```

```
        namelabel = new Label("Your name is: ", Label.LEFT);
        talklabel = new Label("Type your text on the line below.", Label.LEFT);
        userlabel = new Label("These are the people online: ");
        add(namelabel);
        add(name);
        add(connect);
        add(match);
        add(quit);
        add(talk);
        add(talklabel);
        add(input_field);
        add(userlabel);
        add(talk_name);
        validate();
        resize(450,450);
        show();
        start();
    }

    public void start()
    {
        thread = new Thread(this,"main");
        thread.start();
    }

        /* 서버에게 문자열을 전송 */
    public void send_string(String text)
    {
        os.println(text);
        os.flush();
    }

    public void run()
    {
        talk.appendText("Hit the connect button to start.\n");
        while(true)
        {
          try{thread.sleep(300);}
          catch(InterruptedException e){}

                    if( !is_connect || is ==null || os == null)
            continue;
          try
          {
                            /* 만약 서버로부터 들어오는 데이터가 있을 경우 */
                            if(is.available() > 0)
                            {
```

```
                String text = is.readLine();

                if(text.startsWith("name/")) write_user_list(text.substring(5));
                else if(text.startsWith("request/"))
                {
                    rd = new Request_dialog(this);
                    rd.print_request(text.substring(8));
                }
                else if(text.startsWith("okay/"))
                {
                    Ok ok_dialog =new Ok (this,text.substring(5));
                }
                else if(text.startsWith("no/"))
                {
                    No no_dialog = new No(text.substring(3));
                }
                        else if (text.startsWith("position"))
                            {
                                app.convert(text);
                            }
                else if(text.startsWith("play/"))
                {
                                play=true;
                    talk_name.setText("");
                    talk.setText("");
                    print_talk("Now You play!!\n");
                }
                            else if(text.startsWith("command/"))
                            {
                                app.execute_command(text.substring(8));
                            }
                            else if(text.startsWith("playquit"))
                            {
                                talk.setText("");
                                print_talk("Now You play end!!\n");
                                app.execute_command("Clear/");
                                play=false;
                            }
                else
                  print_talk(text);
                }
            }
            catch(IOException e){}
        }
    }

    public void write_user_list(String text)
```

```
{
    int next;
    int i;
    talk_name.setText("");
    for(i=0;i<text.length();i++)
    {
        next=text.indexOf('|',i);
        if(next >= text.length() || next ==-1)
            break;
        print_talk_name(text.substring(i,next) +"\n");
        i=next;
    }
}

public void print_talk_name(String text)
{
    talk_name.appendText(text);
}

public void print_talk(String text)
{
    talk.appendText(text+"\n");
}

    public boolean action(Event evt, Object arg)
{

    if( is_connect )
    {
        if(os == null || is == null)
        {
            print_talk("An error has occured.\n");
            return true;
        }

        if(evt.target==input_field)
        {
            String text = input_field.getText();
            input_field.setText("");
            os.println(text);
            os.flush();
        }
    }

            if (arg.equals("Request") && is_connect && play==false)
    {
        request_frame = new dilog(this);
```

```
}
if(arg.equals("Log Out") && is_connect)
{
                        if(play==true)
                        {
                                os.println("quit");
                                os.flush();
                                play=false;
                        }
                        else {
          talk.appendText('Preparing to log off.\n");
       if(os == null || is ==null)
              {
              print_talk("An error has occured. Not logged on\n");
                     return true;
             }
       os.println("quit");
              os.flush();
                  try{
            if(os!=null) os.close();
            if(is!=null) is.close();
            if(server !=null) server.close();
             }
       catch(IOException e){}
              print_talk("Sucessfully logged off\n");
                  talk_name.setText("");
             is_connect = false;
                        }
              }

if( arg.equals("Connect") && !is_connect)
{
      String text = name.getText();
      int len = text.length();
      if(len == 0){
            print_talk("Inpur your name!!\n");
            return true;
      }
      else{
            try{
                  print_talk("Trying to connect\n");
                  try{
             server = new Socket(app.getCodeBase().getHost(),4443);
                  }
                  catch(UnknownHostException e){
                                                print_talk("Not able to connect\n");
                        return true;
                  }
```

```
                        if(server !=null){
                            is = new DataInputStream(new
BufferedInputStream(server.getInputStream()));
                            os = new PrintStream ( new
BufferedOutputStream(server.getOutputStream(),1024),false);
                        }
                        else
                                                                return true;
                    }
                    catch(IOException e){}
                    is_connect = true;
                    os.println("name/" + text);
                    os.flush();
                    print_talk("\n Your name is '" + text+"'right now\n");
                }
                    }
                    return true;
    }

     public void stop()
    {
        print_talk("Preparing to log off.\n");
        if(os !=null){
            os.println("quit");
            os.flush();
        }
        try {
            if(os !=null) os.close();
            if(is!=null) is.close();
            if(server !=null) server.close();
        }
        catch(IOException e)
        {
            print_talk("Error loggin off\n");
        }
        print_talk("Sucessfully logged off\n");
        talk_name.setText("");
        is_connect=false;
    }

}

interface BadukConstant{
    /* 바둑에서 사용되는 모든 상수를 가지고 있는 interface */
   static final int Bound=-2;
   static final int Boundary=35;
   static final int Wide=20;
```

```
    static final int Black=1;
    static final int White=0;
    static final int BHouse=2;
    static final int WHouse=3;
    static final int Pae=4;
    static final int Empty=-1;
}
```

<**예제**> Baduk.java

서버에서 실행되는 자바 어플리케이션의 소스는 다음과 같다.

```
import java.net.*;
import java.io.*;

class ServerBaduk
{
        /* 현재 바둑을 두고 있는 팀을 기억하는 변수 */
        static Play_user play_user[]= new Play_user[5];
        /* 현재 대화를 나누고 있는 사람의 변수 */
        static On_user on_user[]= new On_user[14];
        /* 현재 바둑을 두고 있는 팀의 수 */
        static int playuser=0;
        /* 현재 대화를 나누고 있는 사람의 수 */
        static int onuser=0;

        public static void main(String args[])
        {
                int i;
                int j;
                /* ServerSocket */
                ServerSocket serversocket = null;
                Socket client = null;
                /* 초기화 */
                for(i=0;i<5;i++)
                        play_user[i]=null;
                for(i=0;i<14;i++)
                        on_user[i]=null;
                try
                {
                        /* serversocket 생성 */
                        serversocket = new ServerSocket(4443);
                }
                catch(IOException e){
                        System.out.println("Connect error" + e);
                        System.exit(1);
                }
```

```
/* 새로운 사용자의 연결을 담당하는 method*/
Polling user = new Polling(serversocket);
/* 현재 연결이 된 사람이 없을 때 우선 한명의 연결을 기다림 */
if(onuser==0)
{
        try
        {
                client = serversocket.accept();
        }
        catch(IOException e){}
        /* 배열에 저장 */
        add_client(new On_user(client));
}

/* 또 다른 접속자의 연결을 기다리는 쓰레드 실행 */
user.start();
/* 현재 접속이 된 사람과 바둑을 두고 있는 사람들의 message 전달 */
try
{
        int test;
        i=0;
        j=0;
        int k=0;
        String text= new String();
        while(onuser>=0)
        {
                /* 바둑을 두고 있는 팀이 있을 경우 서로의 입력을 조사 */
                if( playuser > 0 && k< playuser)
                {
                        /* 접속자의 입력이 있는 가를 check*/
                  if( play_user[k]!=null)
                        if(play_user[k].on_user[1]!=null){
                      test = play_user[k].on_user[1].can_read();
                      if(test == 1)
                          {
                                text = play_user[k].on_user[1].read_text();
                                play_analysis(text,k,1);
                          }
                      }

                  if(play_user[k]!=null)
                        if(play_user[k].on_user[0]!=null){
                          test = play_user[k].on_user[0].can_read();
                          if(test==1)
                          {
                                text = play_user[k].on_user[0].read_text();
```

```
                                        play_analysis(text,k,0);
                                    }
                                }
                            }

                            /* 접속이 되어 있는 사람들의 messgae 교환 */
                            if(onuser>0 && i<onuser)
                            {
                                test=on_user[i].can_read();
                                /* 연결 중지를 요구한 경우 */
                                if(test == -1)
                                    remove_client(i);
                                /* 그렇지 않은 경우 message 분석 */
                                else if(test==1)
                                {
                                    text = on_user[i].read_text();
                                    analysis(text,i);
                                }
                            }
                            try{
                                user.join(2);
                            }
                            catch(InterruptedException e){}
                            /* 새로운 접속자가 있는 가를 검사 */
                            if(user.connected() == true)
                            {
                                add_client(new On_user(user.return_socket()));
                                user.resume();
                            }
                            i++;
                            k++;
                            if(i>=onuser) i=0;
                            if(k>=playuser) k=0;

                            if(onuser==0)
                            {
                                for(j=0;j<14;j++)
                                    on_user[j]=null;
                            }
                            if(playuser==0)
                            {
                                for(j=0;j<5;j++)
                                    play_user[j]=null;
                            }
                        }
                        serversocket.close();
                    }
```

```
            catch(IOException e){}
}

/* 바둑을 두고 있는 사용자의 message 분석 */
static void play_analysis(String text, int k, int who)
{
            /* 돌의 위치인 경우 */
            if(text.startsWith("position"))
            {
                        play_user[k].on_user[0].send_text(text);
                play_user[k].on_user[1].send_text(text);
            }
            /* undo와 같은 명령일 경우 */
            else if(text.startsWith("command"))
            {
                        play_user[k].on_user[0].send_text(text);
                    play_user[k].on_user[1].send_text(text);
}
            /* 바둑을 그만 둔다는 명령 */
            else if(text.startsWith("quit"))
            {
                        /* 바둑을 두고 있는 사용자를 대화를 나누는 곳을로 이동 */
                        play_user[k].on_user[0].send_text("playquit");
                        play_user[k].on_user[1].send_text("playquit");
                        on_user[onuser]=play_user[k].on_user[0];
                        onuser++;
                        on_user[onuser]=play_user[k].on_user[1];
                        onuser++;

                        if(playuser==1)
                        {
                                    play_user[0]=null;
                                    playuser--;
                        }
                        else
                        {
                                    play_user[k]=play_user[playuser-1];
                                    play_user[playuser-1]=null;
                                    playuser--;
                        }
                        send_all_name();
            }
            /* 서로간의 대화인 경우 */
            else
            {
             play_user[k].on_user[0].send_text(play_user[k].on_user[who].name +":" +text);
play_user[k].on_user[1].send_text(play_user[k].on_user[who].name +":" + text);
```

```
		}
	}

	/* 대화를 나누는 user의 message 분석 */
	static void analysis(String text, int i)
	{
		if(text!=null)
		{
			if(text.startsWith("name/"))
			{
				on_user[i].name=text.substring(5);
				send_all_name();
			}
			/* 대화 중지를 요구 */
			else if(text.equals("quit"))
			{
				send_all_user(on_user[i].name + "is logging out.\n");
				remove_client(i);
			}
			/* 다른 사용자와 대국의 요청*/
			else if(text.startsWith("request/"))
				send_request(text.substring(8)+on_user[i].name + "/");
			/* 대국의 요청을 거절 */
			else if(text.startsWith("no/"))
				send_no(text.substring(3));
			/* 대국을 승낙 */
			else if(text.startsWith("okay/"))
				send_ok(text.substring(5),on_user[i].name);
			/* 두명의 사용자의 대국을 시작 */
			else if(text.startsWith("play/")){
				input_play(text.substring(5)+on_user[i].name +"/");
			}
			/* 사용자간의 대화 */
			else
				send_all_user(on_user[i].name + ":"+ text);
		}
	}
	/* 대국의 승낙을 보내는 method*/
	static void send_ok(String text,String temp)
	{
		String name;
		int index;
		int i;
		index=text.indexOf("/");
		name = text.substring(0,index);
		for(i=0;i<onuser;i++)
		{
```

```
                    if(name.equals(on_user[i].name))
                {
                 on_user[i].send_text("okay/"+temp+"/");
            break;
                }
        }
        }

        /* 대화를 나누고 있는 사용자를 대국으로 전환시키는 method*/
        static void input_play(String text)
{
                int i;
        int user_index1=0;
        int user_index2=0;
        String name1=new String();
        String name2=new String();
        int index1= text.indexOf("/");
        int index2= text.indexOf("/",index1+1);
        name1=text.substring(0,index1);
        name2=text.substring(index1+1,index2);
                /* 대국을 요청한 user를 검색 */
        for(i=0;i<onuser;i++)
            if(name1.equals(on_user[i].name))
            {
                user_index1=i;
                break;
            }
                for(i=0;i<onuser;i++)
            if(name2.equals(on_user[i].name))
            {
                user_index2=i;
                break;
            }

                play_user[playuser]=new Play_user(on_user[user_index1],on_user[user_index2]);
        playuser++;
        on_user[user_index1].send_text("play/");
                remove(user_index1);
                user_index2=0;
                for(i=0;i<onuser;i++){
                        if(name2.equals(on_user[i].name))
                        {
                                user_index2=i;
                                break;
                        }
                }
```

```
                    on_user[user_index2].send_text("play/");
                    remove(user_index2);
/* 대국을 요청한 사용자를 대화에서 제거하고 다른 접속자에게 현재의 대화자들의 이름을 전달 */
    }

     /* 대국의 거부를 요청자에게 전달 */
     static void send_no(String text)
     {
          String name;
    int index;
    int i;
    index=text.indexOf("/");
    name = text.substring(0,index);
    for(i=0;i<onuser;i++)
    {
        if(name.equals(on_user[i].name))
        {
            on_user[i].send_text("no/"+on_user[i].name+"/");
            break;
        }
    }
}

     /* 현재 대화를 나누고 있는 사람들의 이름을 사용자에게 절달 */
     static  void send_all_name()
     {
                int i;
                String text=new String("name/");
                for(i=0;i<onuser;i++)
                        text= text + on_user[i].name + "|";
                send_all_user(text);
     }

     /* 대국의 요청을 하는 method*/
     static void send_request(String text)
     {
                String temp;
                String name;
                int ind1,i;
                int flag;
                ind1=text.indexOf("/");
                name=text.substring(0,ind1);
                temp=text.substring(ind1);
                /* 요청을 받은 사용자에게 message 전달 */
                for(i=0;i<onuser;i++)
                {
                        if(name.equals(on_user[i].name))
```

```
                    {
                        on_user[i].send_text("request"+temp);
                        break;
                    }
                }
            }
            /* 대화 중지를 요구한 사용자를 제거 */
            static  void remove_client(int i)
            {
                if(onuser==1)
                {
                    on_user[i].stop_user();
                    on_user[i]=null;
                    onuser--;
                }
                else
                {
                    on_user[i].stop_user();
                    on_user[i]=on_user[onuser-1];
                    on_user[onuser-1]=null;
                    onuser--;
                    send_all_name();
                }
            }
            /* 대국을 중지를 요구한 팀을 제거하고 대화에 연결 */
            static void remove(int i)
            {
                if(onuser ==1)
                {
                    on_user[i]=null;
                    onuser--;
                }
                else
                {
                    on_user[i]=on_user[onuser-1];
                    on_user[onuser-1]=null;
                    onuser--;
                    send_all_name();
                }
            }

            /* 새로운 접속자를 추가 */
            static void add_client( On_user client)
            {
                String text;
                /* 사람이 많은 경우 */
                if(onuser >=10)
```

```
                {
                        client.send_text("Sorry! There are too many users right now.\n");
                        client.stop_user();
                        return ;
                }
                on_user[onuser]=client;
                client.send_text("You have been connected.\n");
                onuser++;
        }
        /* 접속된 모든 사용자에게 message를 전달하는 method*/
        static  void send_all_user(String text)
        {
                int i;
                for(i=0;i<onuser;i++)
                        on_user[i].send_text(text);
        }
}
/* 서버에 접속하려는 사용자들을 관리하는 쓰레드 */
class Polling extends Thread
{
        ServerSocket server=null;
        Socket socket = null;
        boolean is_connect = false;
        public Polling (ServerSocket serversocket)
        {
                server = serversocket;
        }
        public boolean connected()
        {
                return is_connect;
        }
        /* 새롭게 접속된 사용자의 소켓을 반환 */
        public Socket return_socket()
        {
                is_connect = false;
                return socket;
        }

        public void run()
        {
                while(true)
                {
                        is_connect = false;
                        if(!is_connect)
                        {
                                try
                                {
```

```
                                        /* 새로운 사용자의 연결을 기다림 */
                                        socket=server.accept();
                                }
                                catch(IOException e){}
                                is_connect=true;
                        /* 새롭게 연결된 사용자를 처리하는 동안 쓰레드를 일시 중지 */
                                this.suspend();
                        }
                }
        }
}



/* 바둑을 두고 있는 사용자*/
class Play_user
{
        On_user on_user[]= new On_user[2];
        public Play_user(On_user socket1, On_user socket2)
        {
                on_user[0]=socket1;
                on_user[1]=socket2;
        }
}

/* 일반 대화를 나누고 있는 사용자 */
class On_user
{
        Socket socket ;
        DataInputStream is;
        PrintStream os;
        String name;

        public On_user(Socket client)
        {
                socket=client;
                try
                {
                        is =  new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
                        os = new PrintStream(new
BufferedOutputStream(socket.getOutputStream(),1024),false);
                }
                catch(IOException e){}
        }
```

```
/* 사용자에게 message 전달 */
public void send_text(String text)
{
        os.println(text);
        os.flush();
}

/* 사용자로 부터 message 입력 */
public String read_text()
{
        String in;
        try
        {
                in =is.readLine();
                return in;
        }
        catch(IOException e) {}
        return "null";
}

/* 연결된 사용자에게서 입력이 있는 가를 검사 */
public int can_read()
{
        boolean can;
        if(os!=null)
                if(os.checkError())
                {
                        System.out.println("os.checkError");
                        return -1;
                }

        if(is!=null)
        {
                try
                {
                        can=(is.available()>1);
                }
                catch(IOException e)
                {
                        return -1;
                }
                if(can) return 1;
                else return 0;
        }
        return 0;
}
/* 대화중지를 요구한 사용자를 제거 */
```

```
        public void stop_user()
        {
                try
                {
                        if(os!=null) os.close();
                        if (is!=null) is.close();
                        if(socket !=null)socket.close();
                }
                catch(IOException e){}
        }
}
```

<예제> ServerBaduk.java