비교 요소	메소드 오버로딩	메소드 오버라이딩
정의	같은 클래스나 상속 관계에서 동일한 이름의 메소드 중복 작성	서브 클래스에서 슈퍼 클래스에 있는 메소드와 동일한 이름의 메소드 재작성
관계	동일한 클래스 내 혹은 상속 관계	상속 관계
목적	이름이 같은 여러 개의 메소드를 중복 정의하여 사용의 편리성 향상	슈퍼 클래스에 구현된 메소드를 무시하고 서브 클래스에서 새로운 기능의 메소드를 재정의하고자 함
조건	메소드 이름은 반드시 동일함. 메소드의 인자의 개수나 인자의 타입이 달라야 성립	메소드의 이름, 인자의 타입, 인자의 개수, 인자의 리턴 타입 등이 모두 동일하여야 성립
바인딩	**정적 바인딩.** 컴파일 시에 중복된 메소드 중 호출되는 메소드 결정	**동적 바인딩.** 실행 시간에 오버라이딩된 메소드 찾아 호출

5.1.20 추상 메소드와 추상 클래스

-추상 메소드(abstract method)

■ 선언되어 있으나 구현되어 있지 않은 메소드

■ 추상 메소드 정의

• 접근 지정자 abstract 반환형 메소드이름();

-ex) public abstract int getValue();

-추상 메소드는 서브 클래스에서 오버라이딩하여 구현

-추상 클래스(abstract class)

■ 추상 클래스를 하나라도 가지면 추상 클래스임

• 클래스 앞에 반드시 abstract라고 선언해야 함

■ 추상 메소드가 하나도 없지만 클래스 앞에 abstract로 선언한 경우

```
abstract class DObject {
  public DObject next;

  public DObject() { next = null;}
  abstract public void draw() ;
}
```

5.1.21 추상 클래스 특성

-추상 클래스의 객체는 생성할 수 없다.

-추상 클래스 필요성

■ 계층적 상속 관계를 갖는 클래스 구조를 만들 때

■ 설계와 구현 분리

• 슈퍼 클래스에서는 개념적 특징 정의, 서브 클래스에서 구체적 행위 구현

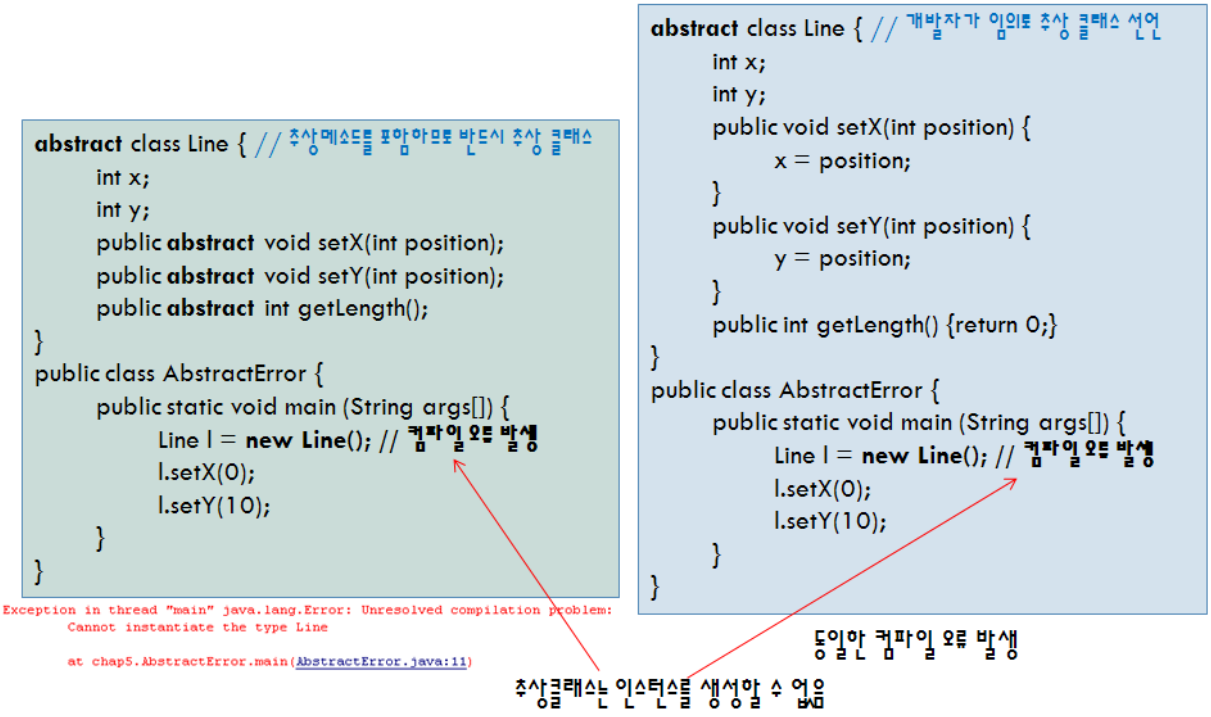
-추상 클래스의 상속

■ 추상 클래스를 상속받아, 추상 메소드를 구현하지 않으면 서브 클래스도 추상 클래스 됨.

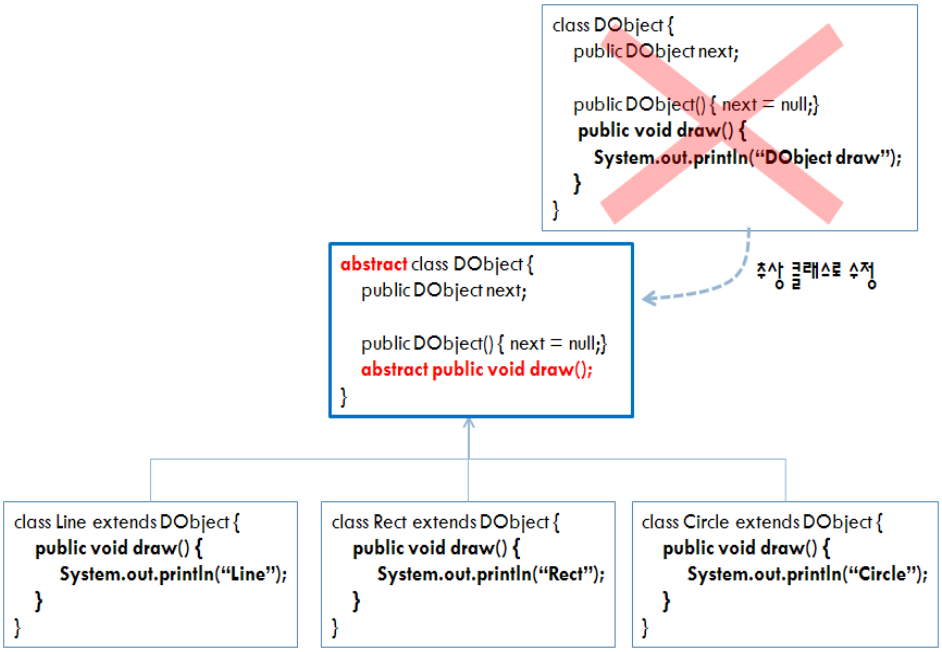
• abstract로 정의하여야 한다.

■서브 클래스에서 추상 메소드를 구현하면 서브 클래스는 추상 클래스가 되지 않는다.

5.1.22 2 가지 종류의 추상 클래스



-추상 클래스의 활용 예



-예제 5-6 : 추상 클래스의 구현

■다음의 추상 클래스 Calculator를 상속받는 GoodCalc 클래스를 독자 임의로 작성하라.

```
abstract class Calculator {
	public abstract int add(int a, int b);
	public abstract int subtract(int a, int b);
	public abstract double average(int[] a);
}
```

-예제 5-6 정답

```
class GoodCalc extends Calculator {
	public int add(int a, int b) {
		return a+b;
	}
	public int subtract(int a, int b) {
		return a - b;
	}
	public double average(int[] a) {
		double sum = 0;
		for (int i = 0; i < a.length; i++)
			sum += a[i];
		return sum/a.length;
	}
	public static void main(String [] args) {
		Calculator c = new GoodCalc();
		System.out.println(c.add(2,3));
		System.out.println(c.add(2,3));
		System.out.println(c.add(new int [] {2,3,4}));
	}
}
```

```
5
-1
3.0
```

21. 실세계의 인터페이스와 인터페이스의 필요성

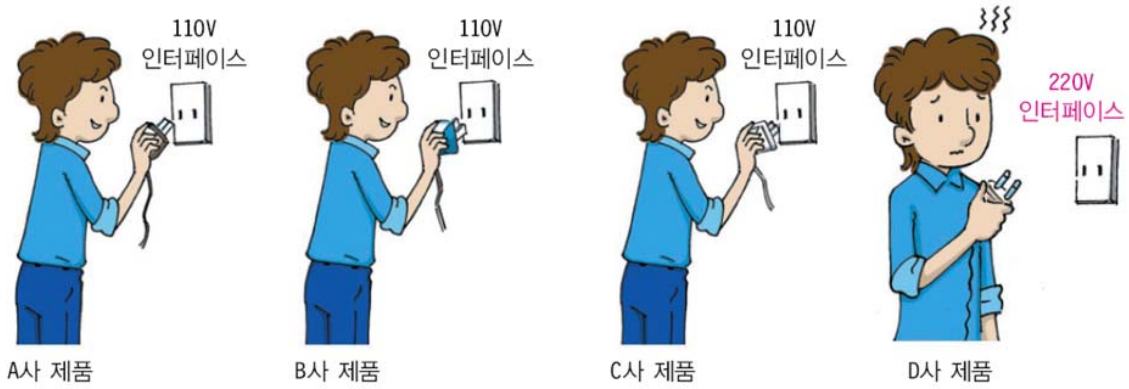


그림 5.2

5.1.23 자바의 인터페이스

-인터페이스(interface)

■ 모든 메소드가 추상 메소드인 클래스

■ 인터페이스는 상수와 메소드만 갖는다. 필드는 없음

-인터페이스 정의

■ interface 키워드로 정의된 클래스

■ ex) public interface SerialDriver {…}

-인터페이스의 특징

■ 메소드 선언 시 abstract 키워드를 사용하지 않아도 된다.

■ 모든 메소드는 public으로 가정, public 접근 지정자 생략 가능

■ 객체를 생성할 수 없음

■ 레퍼런스 변수 타입으로 사용 가능

■ 인터페이스의 메소드 속성

  • public, static, final으로 가정되므로 키워드 생략 가능

-자바 인터페이스 사례

```
public interface Clock {
   public static final int ONEDAY = 24; // 상수 필드 선언
   abstract public int getMinute();
   abstract public int getHour();
   abstract void setMinute(int i);
   abstract void setHour(int i);
}

public interface Car {
   int MAXIMUM_SPEED = 260; // 상수 필드 선언
   int moveHandle(int degree); // abstract 생략 가능
   int changeGear(int gear); // public 생략 가능
}
```

-인터페이스의 필요성

■ 인터페이스를 이용하여 다중 상속 구현

  • 클래스는 다중 상속 불가

■ 인터페이스는 명세서와 같음

  • 구현은 블랙 박스와 같아 인터페이스의 사용자는 구현에 대해 알 필요가 없음

■ 인터페이스만 정의하고 구현을 분리하여, 작업자가 다양한 구현을 할 수 있음

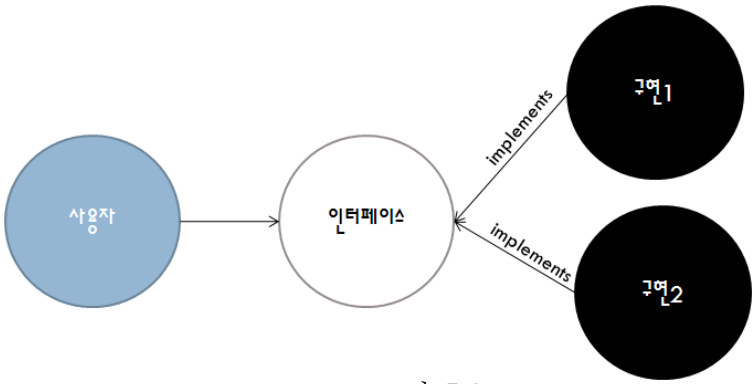


그림 5.3

-인터페이스 상속

■ 인터페이스 간에도 상속 가능

  • 인터페이스 상속하여 확장된 인터페이스 작성 가능

■ 다중 상속 허용

```
interface MobilePhone {
        public boolean sendCall();
        public boolean receiveCall();
        public boolean sendSMS();
        public boolean receiveSMS();
}

interface MP3 {
        public void play();
        public void stop();
}

interface MusicPhone  extends MobilePhone, MP3 {
        public void playMP3RingTone();
}
```

-인터페이스 구현

■ 인터페이스 구현

■ implements 키워드 사용

■ 여러 개의 인터페이스 동시 구현 가능

■ 상속과 구현이 동시에 가능

```
interface USBMouseInterface {
   void mouseMove();
   void mouseClick();
}

public class MouseDriver implements USBMouseInterface { // 인터페이스 구현. 클래스 작성
   void mouseMove() { .... }
   void mouseClick() { ... }

   // 추가적으로 다른 메소드를 작성할 수 있다.
   int getStatus() { ... }
   int getButton() { ... }
}
```

-인터페이스의 다중 구현

```
interface USBMouseInterface {
   void mouseMove();
   void mouseClick();
}

public class MouseDriver implements USBMouseInterface { // 인터페이스 구현. 클래스 작성
   void mouseMove() { .... }
   void mouseClick() { ... }

   // 추가적으로 다른 메소드를 작성할 수 있다.
   int getStatus() { ... }
   int getButton() { ... }
}
```

-추상 클래스와 인터페이스 비교

표 5.2

비교	내용
추상 클래스	• 일반 메소드 포함 가능 • 상수, 변수 필드 포함 가능 • 모든 서브 클래스에 공통된 메소드가 있는 경우에는 추상 클래스가 적합
인터페이스	• 모든 메소드가 추상 메소드 • 상수 필드만 포함 가능 • 다중 상속 지원

제6장 패키지 개념과 자바 기본 패키지

6.1 패키지 개념과 필요성

6.1.1 패키지 개념과 필요성

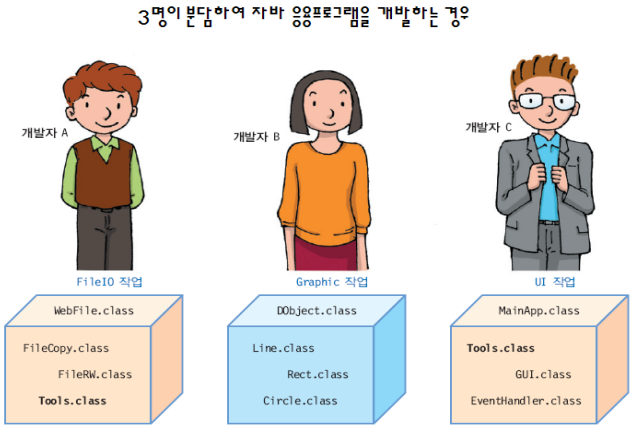
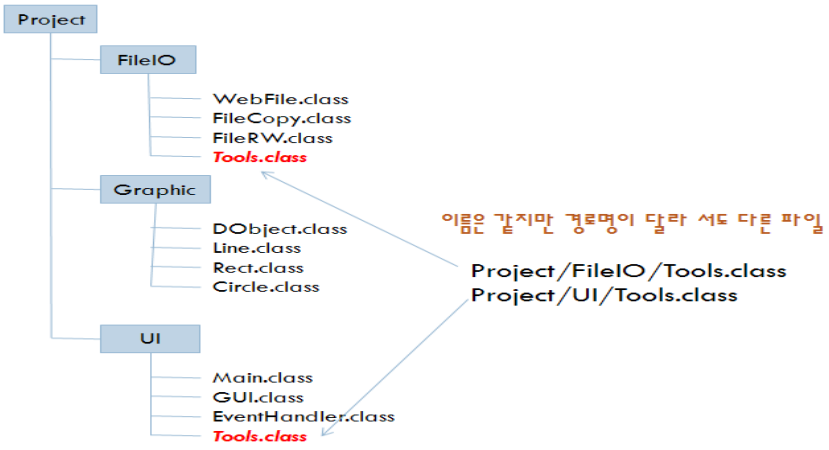


그림 6.1

6.1.2 샘플의 디렉터리 관리(패키지)



6.1.3 자바의 패키지 (package)

-패키지란

■서로 관련된 클래스와 인터페이스의 컴파일된 파일들을 하나의 디렉터리에 묶어 놓은 것

-하나의 응용프로그램은 여러 개의 패키지로 구성

■하나의 패키지에 모든 클래스 파일을 넣어 둘 수 있음

-패키지는 jar 파일로 압축할 수 있음

■예) JDK에서 제공하는 표준 패키지는 rt.jar에 압축

6.1.4 JDK에서 제공되는 클래스의 패키지

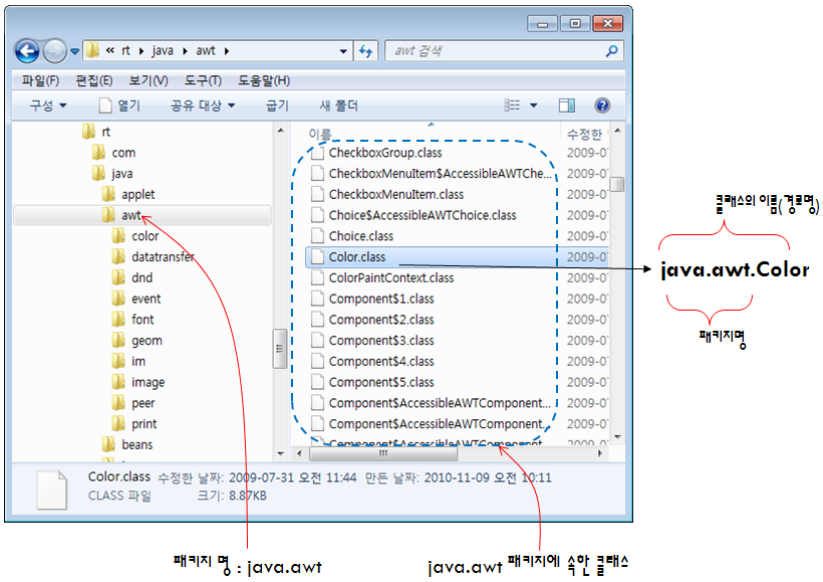


그림 6.2

6.1.5 패키지 사용하기, import문

-다른 패키지 갖다 쓰기

■import를 이용하지 않는 경우

- 소스 내에서 매번 전체 패키지 이름과 클래스명을 써주어야 함

```
public class ImportExample {
    public static void main(String[] args) {
        java.util.Scanner scanner = new java.util.Scanner(System.in);
    }
}
```

■import 키워드 이용하는 경우

- 소스의 시작 부분에 사용하려는 패키지 명시
- 소스 내에서 클래스 명만 명시하면 된다.

```
import java.util.Scanner;
public class ImportExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in)
    }
}
```

-import 사용

■ 특정 클래스의 경로명만 포함하는 경우

- import java.util.Scanner;

■ 패키지 내의 모든 클래스를 포함시키는 경우

- import java.util.*;
- *는 현재 패키지 내의 클래스만을 의미하며 하위 패키지의 클래스까지 포함하지 않는다.

```
import java.util.*;
public class ImportExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in)
    }
}
```

6.1.6 클래스 경로

-클래스의 위치(경로) 지정

■ 클래스 탐색 경로를 지정하는 방법 2 가지

- 클래스 경로의 환경 변수 : 시스템 환경 변수 CLASSPATH

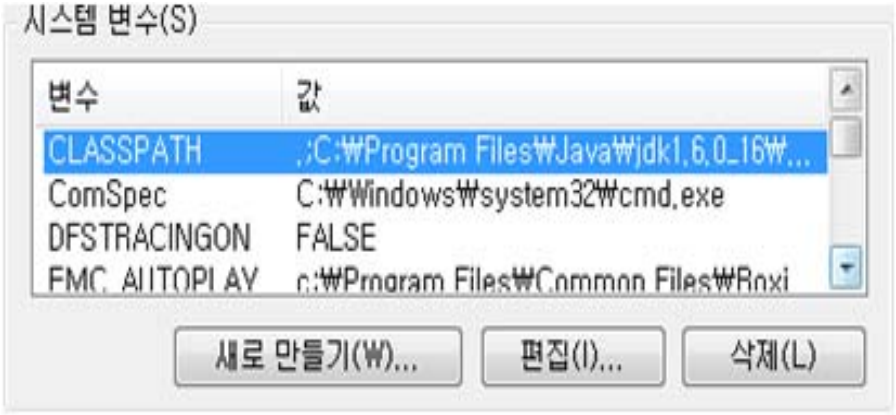


그림 6.3

- java의 옵션 -classpath

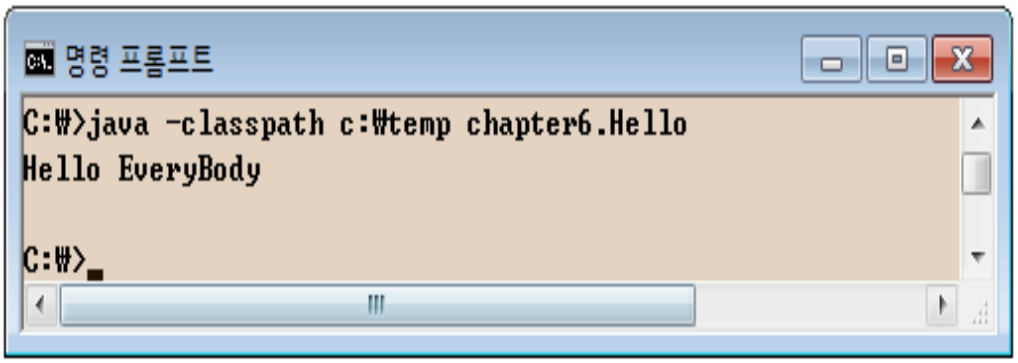


그림 6.4

-실행 시 클래스 파일이 존재하는 패키지 디렉터리 정보를 -classpath 옵션에 지정

-CLASSPATH 지정 방법

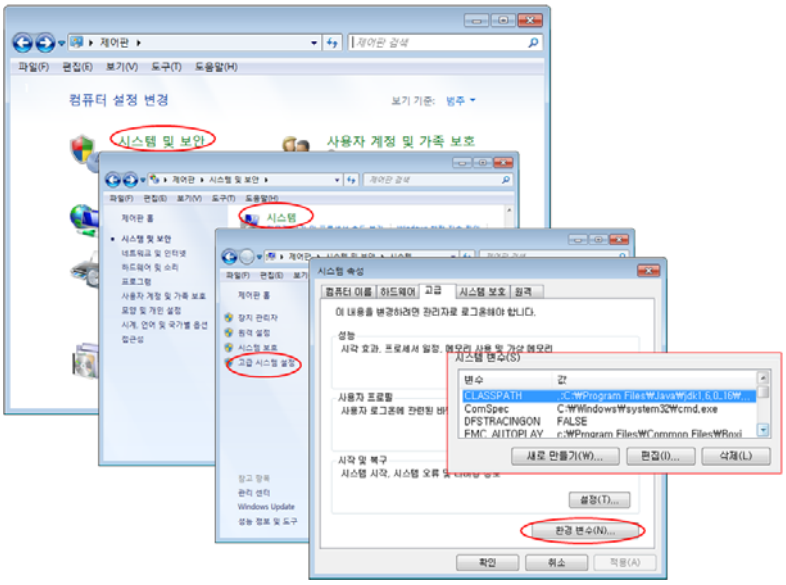


그림 6.5

6.1.7 수동으로 패키지를 만드는 과정

-수동으로 패키지를 만드는 과정

1. 패키지로 사용할 디렉터리 생성
2. 자바 소스 내의 패키지 선언

■ package 패키지 이름;

- 이 소스 파일이 컴파일되면 생성된 클래스 파일이 속할 패키지 이름 선언

```
package 패키지명;
class TestClass {
   ..............
}
```

3. 자바 소스를 컴파일한 클래스 파일을 패키지 디렉터리에 저장

-패키지에 속한 자바프로그램 실행

■ main() 메소드를 가진 클래스의 정확한 경로명 지정

- 패키지가 포함된 디렉터리로 이동하여 경로명을 지정하여 실행하거나
- 임의의 디렉터리에서 java -classpath (패키지경로명) 클래스 이름으로 실행

-패키지를 포함하는 응용프로그램 개발 사례 1

-패키지 디렉터리 chapter6 생성

■ 현재 디렉터리 C:₩Temp

■ C:₩Temp₩chapter6 디렉터리 생성

-소스 파일 내에 패키지 선언

-Hello.java 컴파일

■ C:₩Temp>javac Hello.java

- 현재 디렉터리에서 컴파일
- Hello.class 생성

```
package 패키지명;
class TestClass {
.............
}
```

-Hello.class 파일 복사 혹은 이동

■ Hello.class 파일을 C:₩Temp에서 C:₩Temp₩chapter6로 복사 혹은 이동

-실행

■ C:₩temp>java chapter6.Hello

- 현재 디렉터리에서 실행
- 정상적으로 실행됨

-패키지를 포함하는 응용프로그램 개발 사례 2

-chapter6 디렉터리 생성하지 않음

■ 현재 디렉터리 C:₩Temp

-컴파일

■ C:₩Temp>javac -d . Hello.java

- 현재 디렉터리에서 컴파일
- -d 옵션 : 패키지 디렉터리 자동 생성
- . : 현재 디렉터리를 기준으로 클래스 파일을 찾도록 지시
- 결과 : Hello.class 파일을 C:₩Temp₩chapter6에 생성

-실행

■ C:₩ temp>java chapter6.Hello

- 현재 디렉터리에서 실행
- 정상적으로 실행됨

-사례 2의 컴파일 과정 및 실행

■ java -d . Hello.java 로 컴파일 하면

- 현재 디렉터리 (.) 밑에 chapter6 디렉터리 자동 생성되고 동시에 컴파일된 클래스 파일이 생성됨

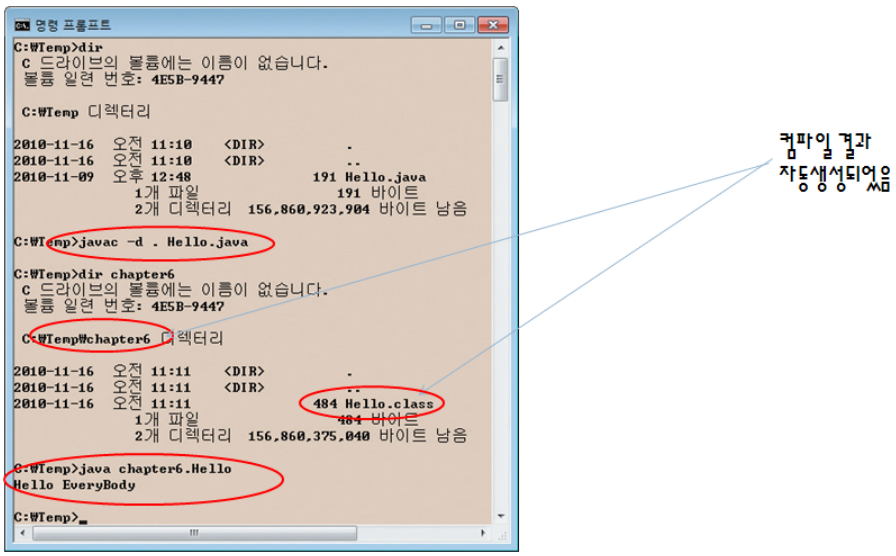


그림 6.6

6.1.8 이클립스에서 쉽게 패키지 만들기

■ 예제로 사용할 샘플 소스(5장의 예제 5-6)

```
abstract class Calculator {
    public abstract int add(int a, int b);// 두 정수의 합을 구하여 리턴
    public abstract int subtract(int a, int b);// 두 정수의 차를 구하여 리턴
    public abstract double average(int[] a);// 배열에 저장된 정수의 평균을 구해 실수로 리턴
}

class GoodCalc extends Calculator {
    public int add(int a, int b) {
        return a+b;
    }
    public int subtract(int a, int b) {
        return a - b;
    }
    public double average(int[] a) {
        double sum = 0;
        for (int i = 0; i < a.length; i++)
            sum += a[i];
        return sum/a.length;
    }
    public static void main(String [] args) {
        Calculator c = new GoodCalc();
        System.out.println(c.add(2,3));
        System.out.println(c.subtract(2,3));
        System.out.println(c.average(new int [] {2,3,4 }));
    }
}
```

6.1.9 프로젝트 작성(프로젝트 이름 : PackageEx)

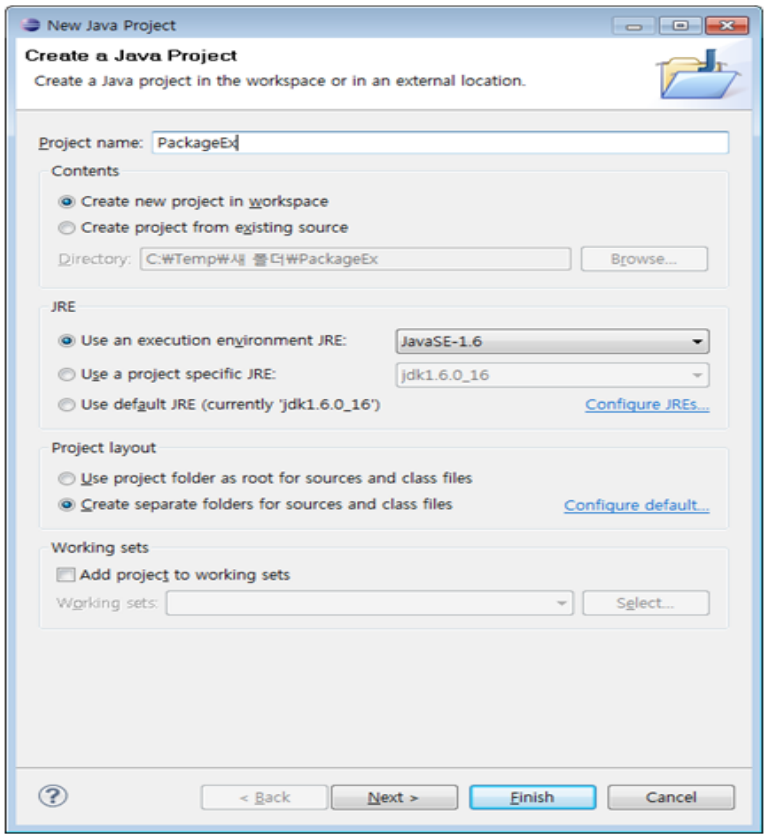


그림 6.7

6.1.10 패키지 lib 작성

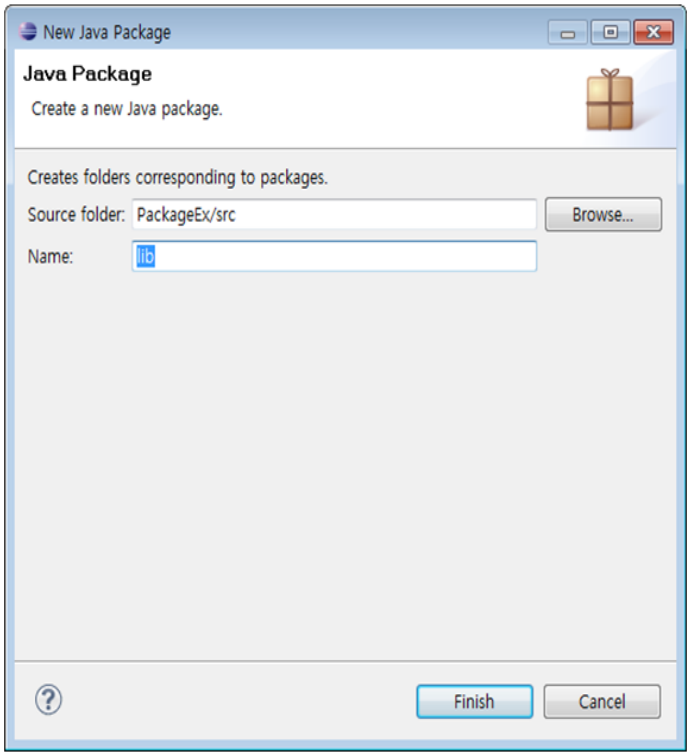


그림 6.8

6.1.11 패키지 app 작성

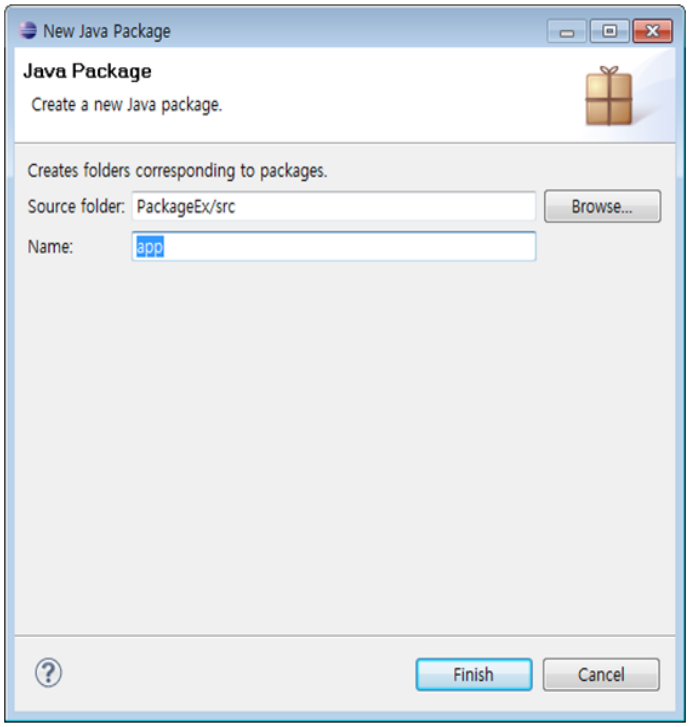


그림 6.9

6.1.12 패키지 작성이 완료된 결과

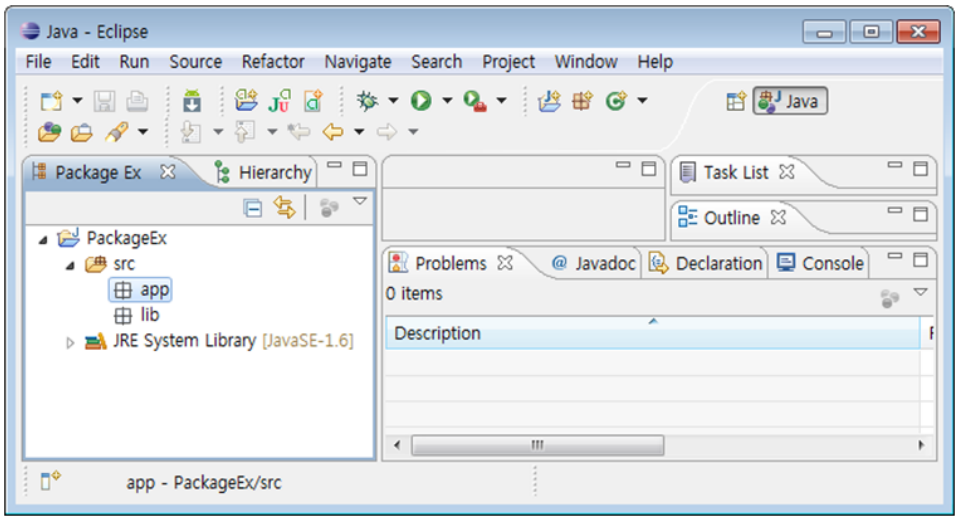


그림 6.10

6.1.13 클래스 Calculator 만들기

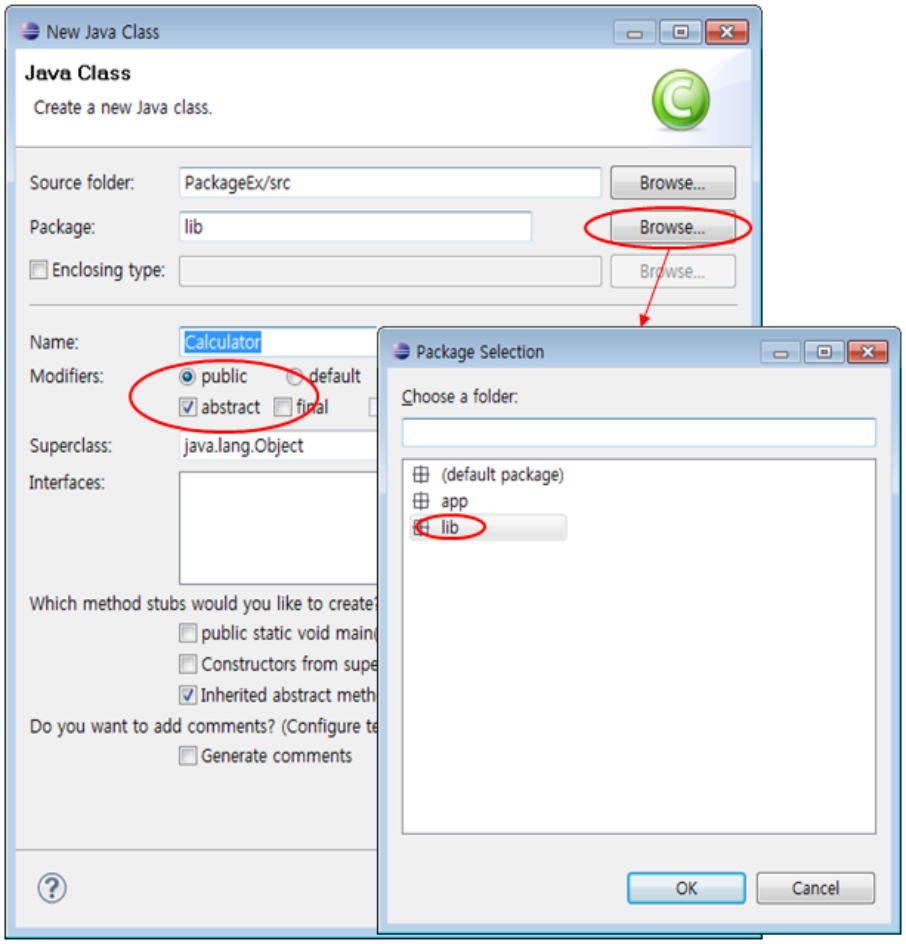


그림 6.11

6.1.14 Calculator 소스 수정

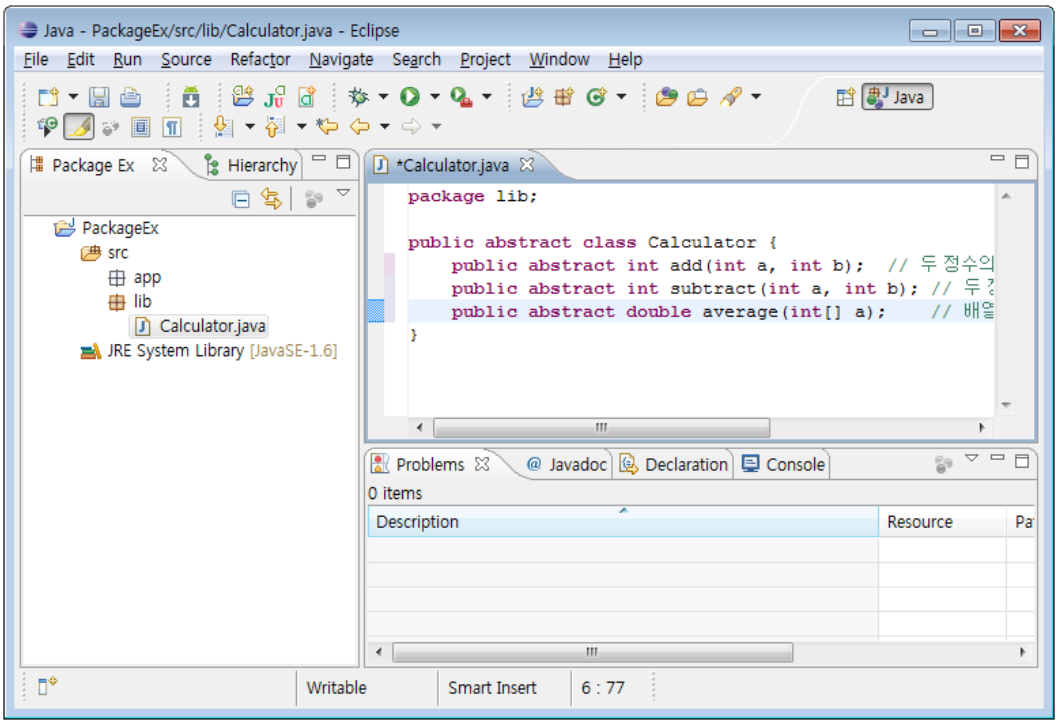


그림 6.12

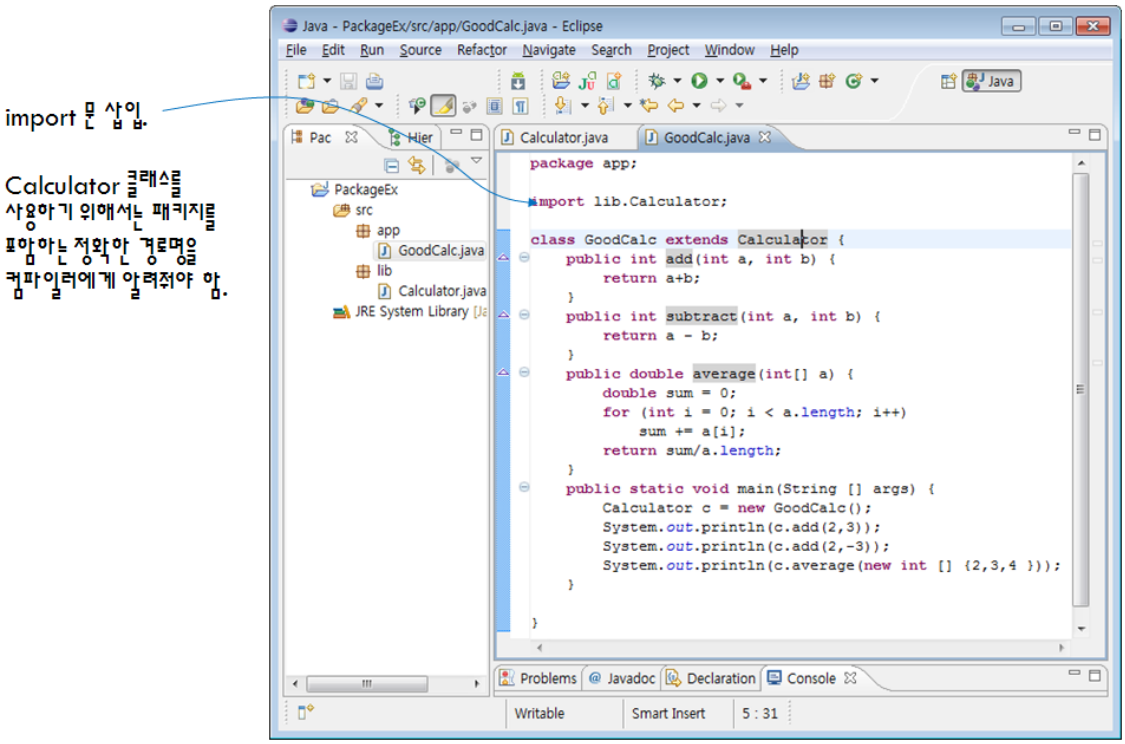


그림 6.13

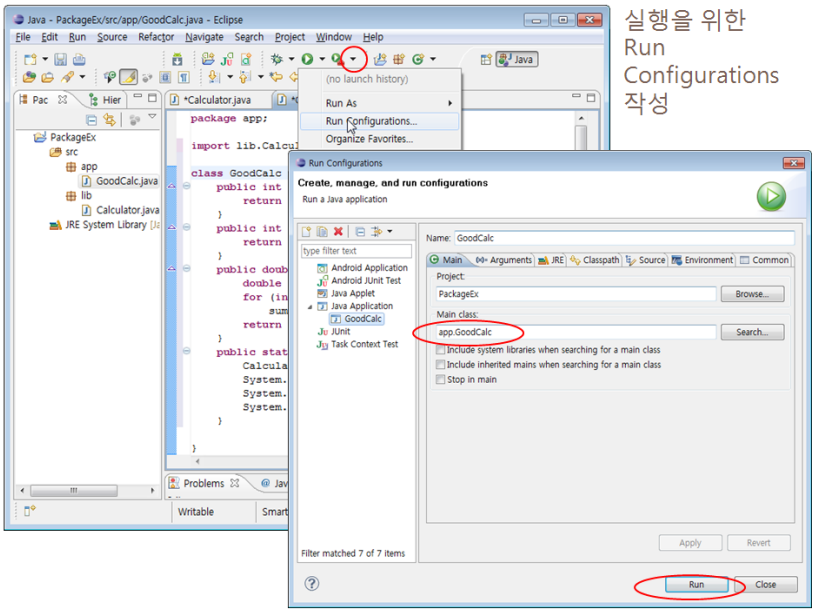


그림 6.14

6.1.15 프로젝트 PackageEx 실행

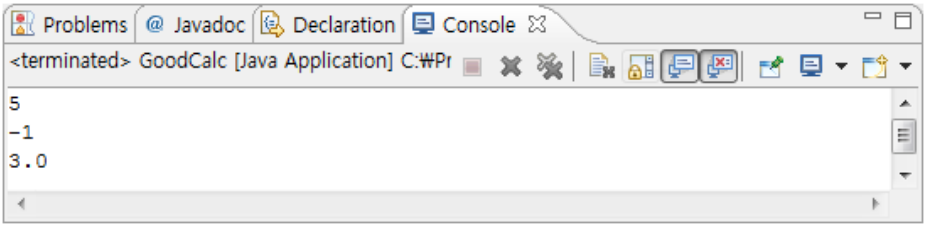


그림 6.15

6.1.16 패키지의 특징

-패키지의 특징

■ 패키지 계층구조
- 클래스나 인터페이스가 너무 많아지면 관리의 어려움
- 관련된 클래스 파일을 하나의 패키지로 계층화하여 관리하면 관리 쉬움

■ 패키지별 접근 제한
- default로 선언된 클래스나 멤버는 동일 패키지 내의 클래스들이 자유롭게 접근하도록 허용

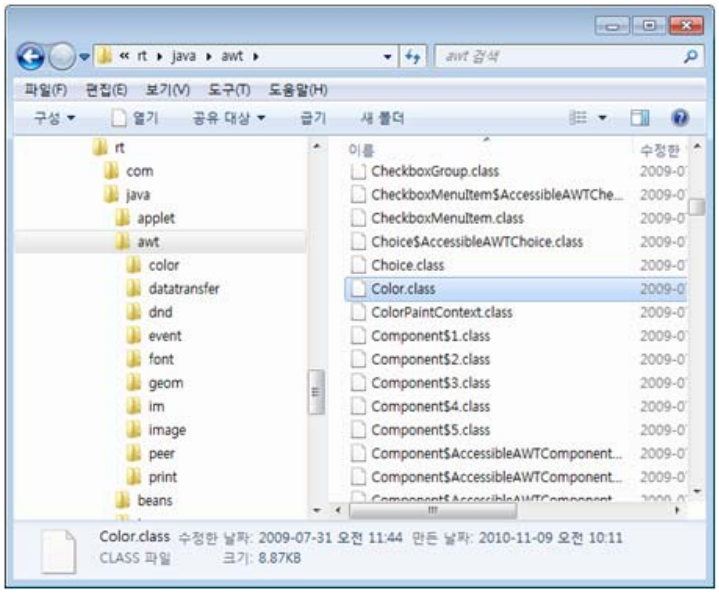
■ 동일한 이름의 클래스와 인터페이스의 사용 가능
- 서로 다른 패키지에 이름이 같은 클래스와 인터페이스 존재 가능

■ 높은 소프트웨어 재사용성
- 오라클에서 제공하는 자바 API는 패키지로 구성되어 있음
- java.lang, java.io 등의 패키지들 덕분에 일일이 코딩하지 않고 입출력 프로그램을 간단히 작성할 수 있음

6.1.17 자바 JDK의 패키지 구조

-JDK 패키지

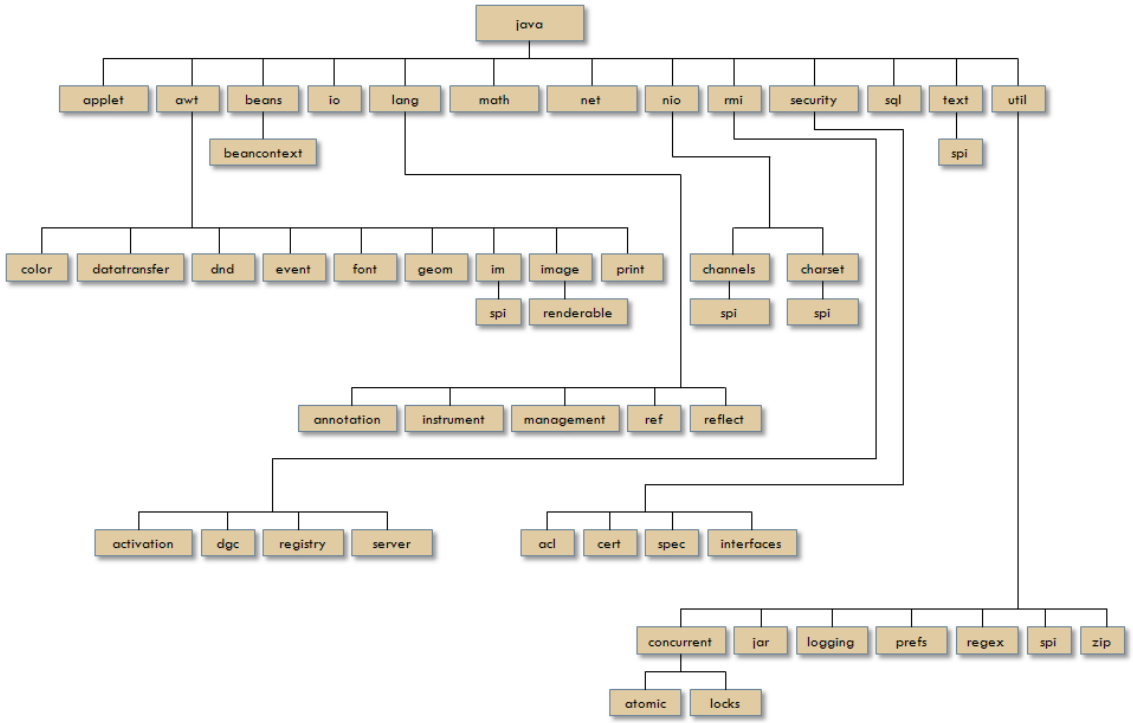
■ 자바에서는 관련된 클래스들을 표준 패키지로 묶어 사용자에게 제공
■ 자바에서 제공하는 패키지는 C언어의 표준 라이브러리와 유사
■ JDK의 표준 패키지는 rt.jar에 담겨 있다.
  • C:₩Program Files₩Java₩jdk1.6.0_16₩jre₩lib₩rt.jar



rt.jar 압축 해제 후 디렉터리 구조

그림 6.16

-자바 패키지 구조



6.1.18 주요 패키지

-java.lang

■자바 language 패키지
• 스트링, 수학 함수, 입출력 등 자바 프로그래밍에 필요한 기본적인 클래스와 인터페이스
■자동으로 import 됨 - import 문이 필요없음
-java.util
■자바 유틸리디 패키지
• 날짜, 시간, 벡터, 해쉬 테이블 등과 같은 다양한 유틸리티 클래스와 인터페이스 제공
-java.io
■키보드, 모니터, 프린터, 디스크 등에 입출력을 할 수 있는 클래스와 인터페이스 제공
-java.awt
■자바 GUI 프로그래밍을 위한 클래스와 인터페이스 제공
-javax.swing
■자바 GUI 프로그래밍을 위한 스윙 패키지

6.1.19 자바 API 참조

-자바 API의 상세 정보
■Oracle Technology Network(http://download-llnw.oracle.com/javase/6/docs/api/)에서 온라인으로 API 규격제공

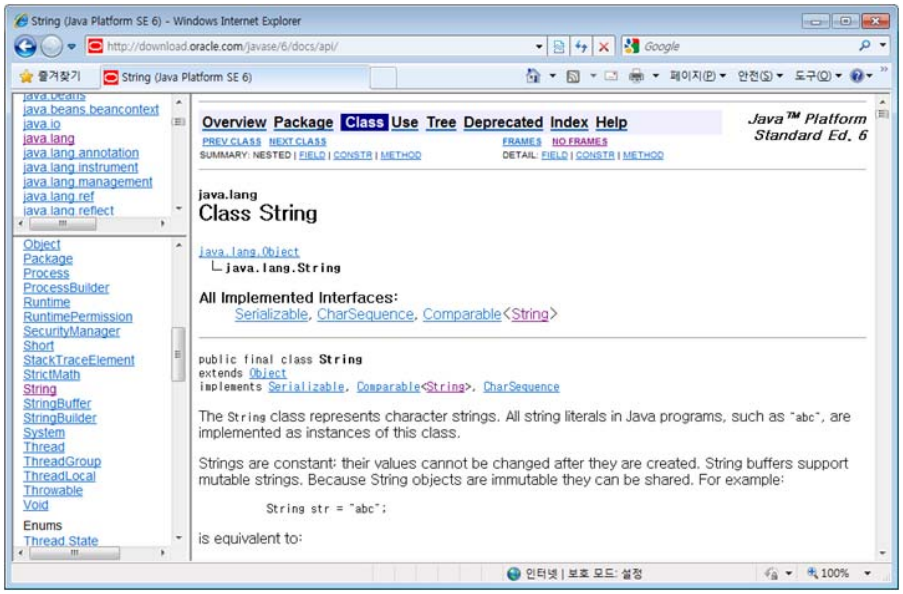


그림 6.17

6.1.20 Object 클래스

-특징
■java.lang 패키지에 포함
■자바 클래스 계 층 구조의 최상위에 위치
■모든 클래스의 수퍼 클래스
-주요 메소드

표 6.1

메소드	설명

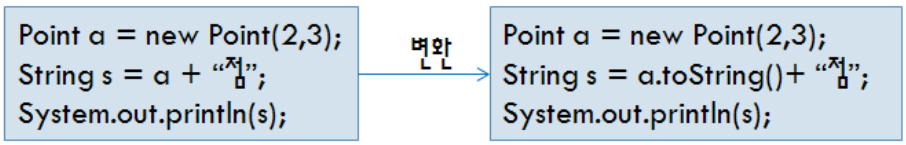
protectedObject clone()	현 객체와 똑같은 객체를 만들어 반환. 오버라이딩 필요
booleanequals(Objectobj)	obj가 가리키는 객체와 현재 객체가 비교하여 같으면 true반환
ClassgetClass()	현 객체의 런타임 클래스를 반환
inthashCode()	현 객체에 대한 해쉬 코드 값 반환
StringtoString()	현 객체에 대한 스트링 표현을 반환
void notify()	현 객체에 대해 대기하고 있는 하나의 쓰레드를 깨운다.
void notifyAll()	현 객체에 대해 대기하고 있는 모든 쓰레드를 깨운다.
void wait()	현 객체의 다른 쓰레드가notify() 또는 notifyAll() 메소드를 호출할 때까지 현 쓰레드를 대기하게 한다.

6.1.21 객체 속성

```
class Point {
   int x, y;
   public Point(int x, int y) {
      this.x = x;
      this.y = y;
   }
}

public class ObjectProperty {
   public static void main(String [] args) {
      Point p = new Point(2,3);
      System.out.println(p.getClass().getName());
      System.out.println(p.hashCode());
      System.out.println(p.toString());
      System.out.println(p);
   }
}
```

```
Point
12677476
Point@c17164
Point@c17164
```

-객체를 문자열로 변환
-String toString()
■ 객체를 텍스트 형태로 표현한 문자열로 반환
■ 반환되는 문자열 : 클래스 이름@객체의 hash code
■ 객체와 문자열이 + 연산이 되는 경우 객체의 toString() 메소드를 호출



Point@c17164점

-새로운 toString() 만들기

```
class Point {
   int x, y;

   public Point(int x, int y) {
      this.x = x; this.y = y;
   }
   public String toString() {
      return "Point(" + x + "," + y+ ")";
   }
}

public class ObjectProperty {
   public static void main(String [] args) {
      Point a = new Point(2,3);
      System.out.println(a.toString());
   }
}
```

Point(2,3)

6.1.22 객체 비교

-객체 레퍼런스의 동일성 비교 : = 연산자 이용

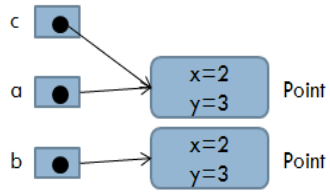
-객체 내용 비교

■ 서로 다른 두 객체가 같은 내용물인지 비교

■ boolean equals(Object obj) 이용

```
class Point {
   int x, y;
   public Point(int x, int y) {
      this.x = x; this.y = y;
   }
}
```

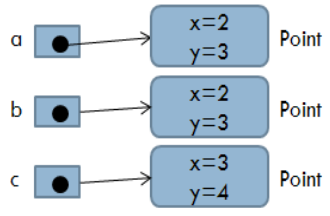
```
Point a = new Point(2,3);
Point b = new Point(2,3);
Point c = a;
if(a == b) // false
   System.out.println("a==b");
if(a == c) // true
   System.out.println("a==c");
```



a==c

```
class Point {
   int x, y;
   public Point(int x, int y) {
      this.x = x; this.y = y;
   }
   public boolean equals(Point p) {
      if(x == p.x && y == p.y) return true;
      else return false;
   }
}
```

```
Point a = new Point(2,3);
Point b = new Point(2,3);
Point c =  new Point(3,4);
if(a == b) // false
   System.out.println("a==b");
if(a.equals(b)) // true
   System.out.println("a is equal to b");
if(a.equals(c)) // false
   System.out.println("a is equal to c");
```



a is equal to b

-예제 6-1 : Rect 클래스 만들고 equals() 만들기

■ int 타입의 width, height의 필드를 가지는 Rect 클래스를 작성하고, 두 Rect 객체의 width, height 필드에 의해 구성되는 면적이 같으면 두 객체가 같은 것으로 판별하도록 equals()를 작성하라. Rect 생성자에서 width, height 필드를 인자로 받아 초기화한다.

```
class Rect {
   int width;
   int height;
   public Rect(int width, int height) {
      this.width = width;
      this.height = height;
   }
   public boolean equals(Rect p) {
      if (width*height == p.width*p.height)
         return true;
      else
         return false;
   }
}
```

```
public class EqualsEx {
   public static void main(String[] args) {
      Rect a = new Rect(2,3);
      Rect b = new Rect(3,2);
      Rect c = new Rect(3,4);
      if(a.equals(b))  System.out.println("a is equal to b");
      if(a.equals(c))  System.out.println("a is equal to c");
      if(b.equals(c))  System.out.println("b is equal to c");
   }
}
```

```
a is equal to b
```

6.1.23 Wrapper 클래스

-자바 기본 데이터 타입을 클래스화한 8개 클래스

표 6.2

기본 데이터 타입	byte	short	int	long	char	float	double	boolean
Wrapper클래스	Byte	Short	Integer	Long	Character	Float	Double	Boolean

-용도

■ 기본 데이터 타입을 사용할 수 없고 객체만 사용하는 컬렉션에 기본 데이터 타입을 Wrapper 클래스로 만들어 사용

6.1.24 Wrapper 객체 생성

-기본 데이터 값을 인자로 Wrapper 클래스 생성자 호출

```
Integer i = new Integer(10);
Character c = new Character('c');
Float f = new Float(3.14);
Boolean b = new Boolean(true);
```

-해당 데이터 값을 나타내는 문자열을 생성자 인자로 사용 가능

■ Boolean, Short, Byte, Integer, Long, Double, Float 경우

```
Boolean b = new Boolean("false");
Integer I = new Integer("10");
Double d = new Double("3.14");
```

-Float는 double 타입의 값을 생성자의 인자로 사용 가능

```
Float f = new Float((double) 3.14);
```

6.1.25 주요 메소드

■ 가장 많이 사용하는 Integer 클래스의 주요 메소드

표 6.3

메소드	설명
staticintbitCount(inti)	이진수 표현에서 1을 개수를 반환
float floatValue()	float타입으로 변환된 값 반환
intintValue()	int타입으로 변환된 값 반환
long longValue()	long 타입으로 변환된 값 반환
short shortValue()	short 타입으로 변환된 값 반환
staticintparseInt(String s)	스트링을10진 정수로 변환된 값 반환
staticintparseInt(String s, intradix)	스트링을 지정된진법의 정수로 변환된 값 반환
static Sting toBinaryString(inti)	이진수 표현으로 변환된 스트링 반환
static Sting toHexString(inti)	16진수 표현으로 변환된 스트링 반환
static Sting toOctalString(inti)	8진수 표현으로 변환된 스트링 반환
static Sting toString(inti)	정수를 스트링으로 변환하여 반환

6.1.26 Wrapper 활용

-Wrapper 객체로부터 기본 데이터 타입 알아내기

```
Integer i = new Integer(10);
int ii = i.intValue(); // ii = 10
Character c = new Character('c');
char cc = c.charValue(); // cc = 'c'
Float f = new Float(3.14);
float ff = f.floatValue(); // ff = 3.14
Boolean b = new Boolean(true);
boolean bb = b.booleanValue(); // bb = true
```

-문자열을 기본 데이터 타입으로 변환

```
int i = Integer.parseInt("123"); // i = 123
boolean b = Boolean.parseBoolean("true"); // b = true
float f = Float.parseFloat("3.141592"); // f = 3.141592
```

-기본 데이터 타입을 문자열로 변환

```
String s1 = Integer.toString(123); // 정수 123을 문자열 "123"으로 변환
String s2 = Integer.toHexString(123); // 정수 123을 16진수의 문자열 "7b"로 변환
String s3 = Float.toString(3.141592f); // 실수 3.141592를 문자열 "3.141592"로 변환
String s4 = Charater.toString( a ); // 문자 'a'를 문자열 "a"로 변환
String s5 = Boolean.toString(true); // 불린 값 true를 문자열 "true"로 변환
```

-예제 6-2 : Wrapper 클래스 활용

■ 다음은 Wrapper 클래스를 활용하는 예이다. 다음 프로그램의 결과는 무엇인가?

```
public class WrapperClassEx {
   public static void main(String[] args) {
      Integer i = new Integer(10);
      char c = '4';
      Double d = new Double(3.1234566);
      System.out.println(Character.toLowerCase('A'));
      if (Character.isDigit(c))
         System.out.println(Character.getNumericValue(c));
      System.out.println(Integer.parseInt("-123"));
      System.out.println(Integer.toBinaryString(28));
      System.out.println(Integer.toHexString(28));
      System.out.println(i.doubleValue());
      System.out.println(d.toString());
      System.out.println(Double.parseDouble("44.13e-6"));
   }
}
```

```
a
4
-123
16
11100
3
1c
10.0
3.1234566
4.413E-5
```

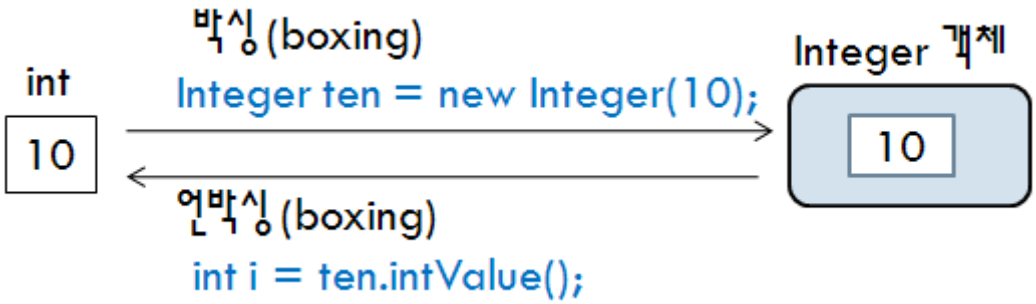
6.1.27 박싱과 언박싱

-박싱(boxing)

■ 기본 데이터 타입을 Wrapper 클래스로 변환하는 것

-언박싱(unboxing)

■ 반대의 경우를 언박싱이라고 한다.



6.1.28 Auto boxing & unboxing

-JDK 5.0 이상부터 지원

-Auto boxing

■기본 데이터 타입을 자동으로 Wrapper 클래스로 변환
-Auto unboxing
■Wrapper 클래스를 자동으로 기본 데이터 타입으로 변환

```
// JDK5.0이상에서
Integer ten = 10; // 자동 박싱
int i = ten; // 자동 언박싱
```

-예제 6-3 : 박싱 언박싱의 예
다음 코드에 대한 결과는 무엇인가?

```
public class AutoBoxingUnBoxing {
   public static void main(String[] args) {
      int i = 10;
      Integer intObject = i;// auto boxing
      System.out.println("intObject = " + intObject);
      i = intObject + 10;// auto unboxing
      System.out.println("i = " + i);
   }
}
```

```
intObject = 10
i = 20
```

6.1.29 String의 생성과 특징

-String - java.lang.String
■String 클래스는 하나의 스트링만 표현

```
//스트링 리터럴로 스트링 객체 생성
String str1 = "abcd";

// String 클래스의 생성자를 이용하여 스트링 생성
char data[] = {'a', 'b', 'c', 'd'};
String str2 = new String(data);
String str3 = new String("abcd"); // str2와 str3은 모두  "abcd" 스트링임
```

-String 생성자

표 6.4

생성자	설명
String()	빈스트링 객체 생성
String(byte[] bytes)	플랫폼의 기본 문자집합을 이용하여 바이트배열을 디코딩하여스트링 객체 생성
String(String original)	인자로 주어진 스트링과 똑같은 스트링 객체를 생성
String(StringBuffer buffer)	스트링 버퍼에 포함된 일련의 문자들을 나타내는 스트링 객체 생성

6.1.30 스트링 리터럴과 new String()

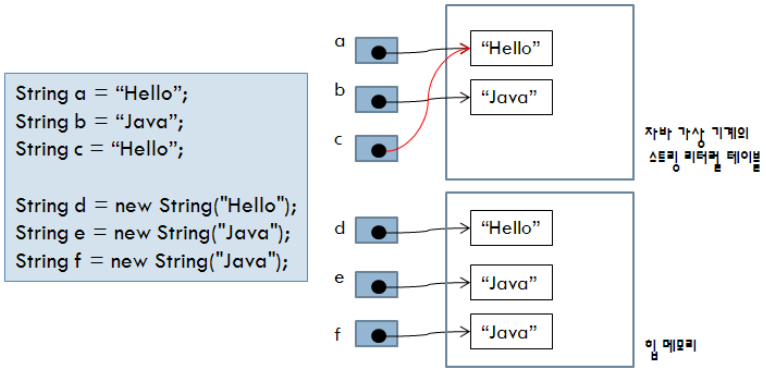
-스트링 생성

■ 단순 리터럴로 생성, String s = "Hello";

• JVM이 리터럴 관리, 응용프로그램 내에서 공유됨

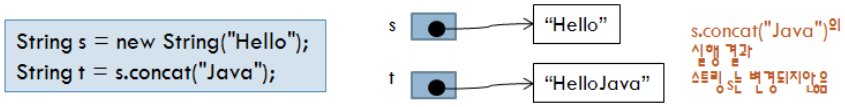
■ String 객체로 생성, String t = new String("Hello");

• 힙에 String 객체 생성



6.1.31 스트링 객체의 주요 특징

-스트링 객체는 수정 불가능



-==과 equals()

■ 두 스트링을 비교할 때 반드시 equals()를 사용하여야 함

• equals()는 내용을 비교하기 때문

6.1.32 주요 메소드

표 6.5

메소드	설명
charcharAt(int index)	지정된 인덱스에 있는 문자값을 반환
intindexOf(intch)	ch 문자가 있는 인덱스 리턴. 없으면 -1리턴
inindexOf(intch, intfromIndex)	fromIndex위치부터 끝까지 문자 ch 탐색. 인덱스 리턴. 없으면 -1리턴
String concat(String str)	지정된 스트링을 현재 스트링 뒤에 덧붙인 스트링 반환
boolean contains(CharSequence s)	지정된일련의 문자들을 포함하고 있으면 true 반환
intlength()	스트링의 길이 반환
String replace(Charsequece target, Charsequence replacement)	target지정하는 일련의 문자들을 replacement가 지정하는 문자들로 변경한 스트링 반환

String[] split(String regex)	정규식에 일치하는 부분을 중심으로 스트링 분리하여 스트링 배열로 반환
String subString(intbeginIndex)	지정된 인덱스부터 시작하는 서브 스트링 반환
String toLowerCase()	스트링을 소문자로 변경한 스트링 반환
String toUpperCase()	스트링을 대문자로 변경한 스트링 반환
String trim()	스트링 앞뒤의 공백문자들을 제거한 스트링 반환

6.1.33 문자열 비교

-int compareTo(String anotherString)

■문자열이 같으면 0을 리턴

■비교 연산자 ==는 레퍼런스를 비교하므로 문자열 비교에는 사용할 수 없다.

```
String a = "java";
String b = "jasa";
int res = a.compareTo(b);
if(res == 0)
   System.out.println("the same");
else if(res < 0)
   System.out.println(a +"<"+b);
else
   System.out.println(a +">"+b);
```

```
java>jasa
```

6.1.34 문자열 연결

-String concat(String str)

■자바에서 스트링 연결 연산자 + 지원

+ 연산에서 문자열이 인자로 포함되어 있으면 산술 연산이 아닌 문자열 연결 연산으로 간주.

-객체가 문자열 연결 연산에 포함되어 있는 경우

■toString() 메소드를 호출하여 객체를 문자열로 변환한 후 문자열 연결

-기본 데이터 타입

■그대로 문자열로 변환된 후에 문자열 연결

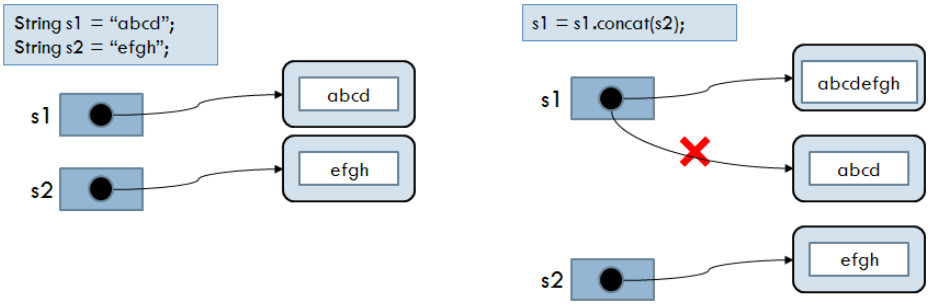
```
System.out.print("abcd" + 1 + true + 3.13e-2 + 'E'+ "fgh" );
// abcd1true0.0313Efgh 출력
```

-String 클래스의 concat(String str) 메소드를 이용하여 스트링 연결

```
"abcd".concat("efgh");
// "abcdefg" 리턴
```

-기존 String 객체에 연결되지 않고 새로운 스트링 객체를 생성하여 리턴

6.1.35 concat()은 새로운 문자열을 생성



6.1.36 문자열 내의 공백 제거, 문자열의 각 문자 접근

-공백 제거

■ String trim()

• 스트링 앞 뒤 공백 문자(tab, enter, space) 제거한 스트링 리턴

```
String a = " abcd def ";
String b = "\txyz\t";
String c = a.trim(); // c = "abcd def"
String d = b.trim(); // d = "xyz"
```

-문자열의 문자

■ char charAt(int index)

• 스트링에 포함된 각 문자 접근

```
String a = "class";
char c = a.charAt(2); // c = 'a'
```

```
// "class"에 몇 개의 's'가 포함되었는지 알아내는 코드
int count = 0;
String a = "class";
for(int i=0; i<a.length(); i++) { // a.length()는 5
   if(a.charAt(i) == 's')
      count++;
}
System.out.println(count); // 2 출력
```

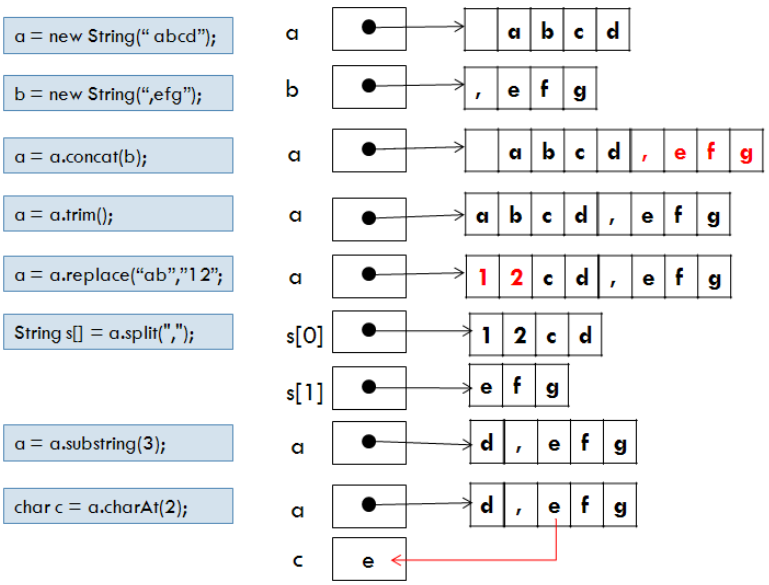
-예제 6-4 : String 클래스 메소드 활용

■ String 클래스의 다양한 메소드를 활용하는 예를 보여라.

```
public class StringEx {
   public static void main(String[] args) {
      String a = new String(" abcd");
      String b = new String(",efg");
      //문자열 연결
      a = a.concat(b);
      System.out.println(a);

      //공백 제거
      a = a.trim();
      System.out.println(a);

      //문자열 대치
      a = a.replace("ab","12");
      System.out.println(a);

      //문자열 분리
      String s[] = a.split(",");
      for (int i=0; i<s.length; i++)
         System.out.println("분리된 " + i + "번 문자열: " + s[i]);

      //서브 스트링
      a = a.substring(3);
      System.out.println(a);

      //문자열의 문자
      char c = a.charAt(2);
      System.out.println(c);
   }
}
```

```
 abcd,efg
abcd,efg
12cd,efg
분리된 0번 문자열: 12cd
분리된 1번 문자열: efg
d,efg
e
```

-예제 실행 과정



6.1.37 StringBuffer 클래스

-java.lang.StringBuffer

■ 스트링과 달리 객체 생성 후 스트링 값 변경 가능

■ append와 insert 메소드를 통해 스트링 조작

■ StringBuffer 객체의 크기는 스트링 길이에 따라 가변적

-생성자

```
StringBuffer sb = new StringBuffer("java");
```

표 6.6

생성자	설명
StringBuffer()	문자를 포함하고 있지 않고 초기 크기가 16인 스트링 버퍼 생성
StringBuffer(charSequenceseq)	seq가지정하는 일련의 문자들을 포함하는 스트링 버퍼 생성
StringBuffer(int capacity)	문자를 포함하고 있지 않고 지정된초기 크기를 갖는 스트링 버퍼 생성
StringBuffer(String str)	지정된 스트링으로 초기화된 스트링 버퍼 생성

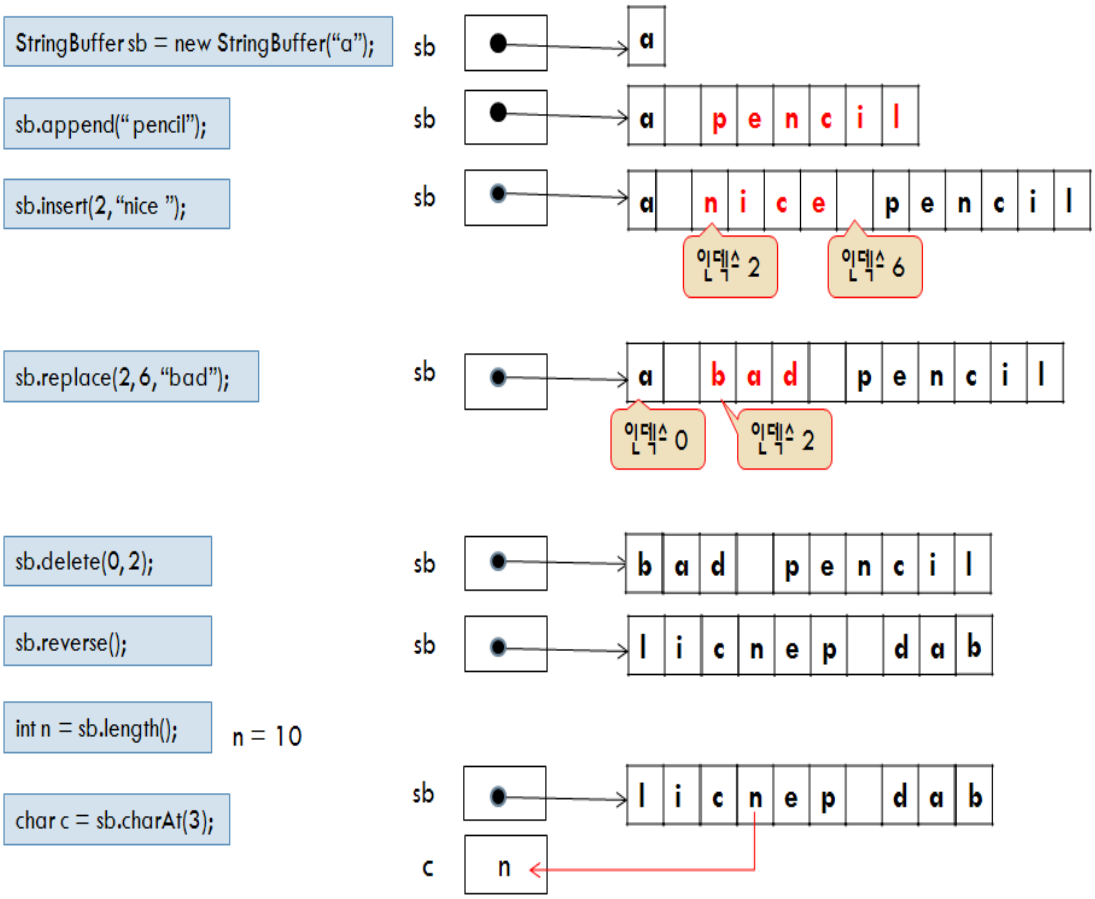
6.1.38 주요 메소드

표 6.7

메소드	설명
StringBuffer append(String str)	지정된 스트링을스트링 버퍼에 덧붙인다.

StringBuffer append(StringBuffersb)	지정된스트링 버퍼를 스트링 버퍼에 덧붙인다.
int capacity()	현재 스트링 버퍼의 크기 반환
StringBuffer delete(int start, int end)	지정된서브 스트링을스트링 버퍼에서 제거
StringBuffer insert(int offset, String str)	지정된스트링을스트링 버퍼의 특정 위치에 삽입
StringBuffer replace(int start, int end, String str)	스트링버퍼 내의 서브 스트링을 지정된 스트링으로 대치
StringBuffer reverse()	스트링 버퍼 내의 문자들을 반대 순서로 변경
voidsetLength()	스트링버퍼 내 저장된 문자열 길이를 설정. 현재 길이보다 큰 경우 널 문자로 채우며 작은 경우는 문자열이 잘린다.

6.1.39 StringBuffer의 메소드 활용 예



-예제 6-5 : StringBuffer 클래스 메소드 활용

■ StringBuffer 클래스의 메소드를 이용하여 문자열을 조작하는 예를 보여라.

```
public class StringBufferEx {
   public static void main(String[] args) {
      StringBuffer sb = new StringBuffer("This");
      System.out.println(sb.hashCode());
      sb.append(" is pencil"); // 문자열 덧붙이기
      System.out.println(sb);

      sb.insert(7, " my"); // 문자열 삽입
      System.out.println(sb);

      sb.replace(8, 10, "your"); // 문자열 대치
      System.out.println(sb);

      sb.setLength(5); // 스트링 버퍼 내 문자열 길이 설정

      System.out.println(sb);
      System.out.println(sb.hashCode());
   }
}
```

```
14576877
This is pencil
This is my pencil
This is your pencil
This
14576877
```

6.1.40 StringTokenizer 클래스

-java.util.StringTokenizer

■ 하나의 스트링을 구분자로 분리하여 토큰 형태로 파싱

- 스트링을 구분할 때 사용되는 문자들을 구분 문자(delimeter)라고 함

```
String query = "name=kitae&addr=seoul&age=21";
StringTokenizer st = new StringTokenizer(query, "&");
```

-위의 예에서 '&'가 구분 문자

-토큰(token)

- 구분 문자로 분리된 스트링

-String 클래스의 split 메소드를 이용하여 동일한 구현 가능

6.1.41 생성자와 주요 메소드

-생성자

표 6.8

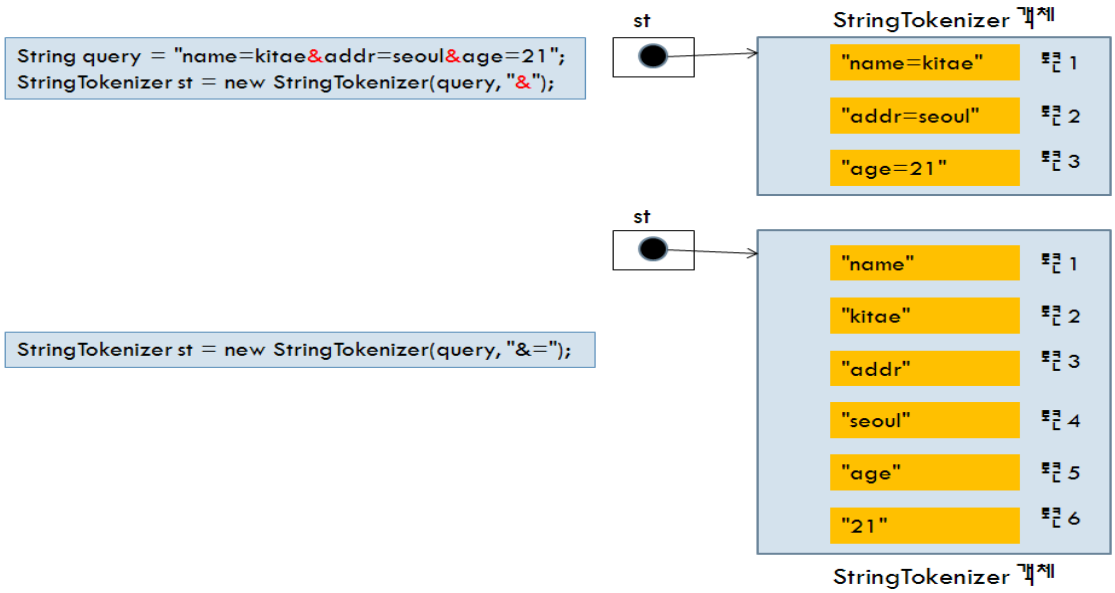
생성자	설명
StringTokenizer(String str)	지정된 스트링으로 초기화된 스트링토크나이저 생성
StringTokenizer(String str, String delim)	지정된 스트링과 구분 문자로 초기화된 스트링토크나이저 생성
StringTokenizer(String str, String delim, booleanreturnDelims)	지정된 스트링과 구분 문자로 초기화된 스트링토크나이저 생성. returnDelims가 true이면 구분 문자로 지정된 문자도 분리된 토큰에 포함된다.

-주요 메소드

표 6.9

메소드	설명
intcountTokens()	스트링에남아 토큰 수 반환
booleanhasMoreTokens()	스트링에 토큰이 남아 있으면 true 반환
String nextToken()	다음 토큰 반환
StringnextToken(String delim)	지정된분리자에 대한 다음 토큰 반환

6.1.42 StringTokenizer 객체 생성과 문자열 파싱



-예제 6-6 : StringTokenizer 클래스 메소드 활용

■ "홍길동/장화/홍련/콩쥐/팥쥐"문자열을'/'를 구분 문자로 하여 토큰을 분리하여 각 토큰을 출력하라.

```
import java.util.StringTokenizer;

public class StringTokenizerEx {
   public static void main(String[] args) {
      StringTokenizer st = new StringTokenizer("홍길동/장화/홍련/콩쥐/팥쥐", "/");
      while (st.hasMoreTokens())
         System.out.println(st.nextToken());
   }
}
```

홍길동
장화
홍련
콩쥐
팥쥐

6.1.43 Math 클래스

-java.lang.Math

■ 기본적인 산술 연산을 수행하는 메소드 제공

■ 모든 멤버 메소드는 static으로 정의됨

■ 객체를 만들어서 사용할 필요 없음

6.1.44 주요 메소드

■ double 타입에 대한 주요 메소드

표 6.10

메소드	설명
static doubleabs(double a)	절대값 반환
static double cos(double a)	cosine 값 반환
static double sin(double a)	sine 값 반환
static double tan(double a)	tangent값 반환
static double exp(double a)	값 반환
static double ceil(double a)	지정된실수보다 크거나 같은 수 중에서 가장 작은 정수를 실수 타입으로 반환
static double floor(double a)	지정된실수보다 작거나 같은 수 중에서 가장 큰 정수를 실수 타입으로 반환
static double max(double a, double b)	두 수 중에서 큰 수 반환
static double min(double a, double b)	두 수 중에서 작은 수 반환
static double random()	0.0보다 크거나 같고 1.0보다 작은 임의의 수 반환
static double rint(double a)	지정된 실수와 가장근접한 정수를 실수 타입으로 반환
static double round(double a)	지정된 실수를 소수 첫째 자리에서 반올림한 정수를 실수 타입으로 반환
static double sqrt(double a)	제곱근을 반환

6.1.45 Math 클래스를 활용한 난수 발생

-난수 발생

■ static double random()

• 0.0 이상 1.0 미만의 임의의 double 값을 반환.

```
for(int x=0; x<10; x++) {
   double d = Math.random()*100; // [0.0 ~ 99.9999] 실수 발생
   int n = (int)(Math.round(d)); // Math.round(d)는 d에 가장 가까운 정수를 리턴
   System.out.println(n);
}
```

• 위의 코드에서 round() 메소드는 Math. round(55.3)은 55.0을 리턴하며, Math. round(55.9)는 56.0을 리턴

-java.util의 Random 클래스를 이용하면 좀 더 다양한 형태로 난수 발생 가능

-예제 6-7 : Math 클래스 메소드 활용

■ Math 클래스의 다양한 메소드 활용 예를 보여라.

```
public class MathEx {
   public static void main(String[] args) {
      double a = -2.78987434;
      // 절대값 구하기
      System.out.println(Math.abs(a));
      System.out.println(Math.ceil(a)); // ceil
      System.out.println(Math.floor(a)); // floor
      System.out.println(Math.sqrt(9.0)); // 제곱근
      System.out.println(Math.exp(1.5)); // exp
      System.out.println(Math.rint(3.141592)); // rint

      // [1,45] 사이의 난수 발생
      System.out.print("이번주 행운의 번호는");
      for (int i=0; i<5; i++)
         System.out.print(Math.round(1 + Math.random() * 44) + " ");
      System.out.println("입니다.");
   }
}
```

```
2.78987434
-2.0
-3.0
3.0
4.4816890703380645
3.0
이번주 행운의 번호는 35 42 18 31 33
```

제7장 입출력 스트림

7.1 스트림

7.1.1. 스트림이란?

■ 데이터의 흐름

■ 입출력 스트림은 입출력 장치와 프로그램 사이의 일련의 데이터 흐름을 의미

■ 스트림을 통해 흘러가는 데이터의 기본 단위는 바이트

그림 7.1

-스트림의 특징

■ 스트림은 단방향이다.

• 방향에 따라 입력 스트림과 출력 스트림으로 나뉜다.

■ 스트림은 선입선출, 즉 FIFO 구조이다.
■ 스트림은 연결될 수 있다.
■ 스트림은 지연될 수 있다.
  • 입력 스트림이 흐르는 통로인 파이프가 비어 있다면, 컴퓨터는 읽어갈 데이터가 없으므로 기다리게 되고, 반대로 출력 스트림이 흐르는 통로인 파이프에 데이터가 꽉 차 있다면 컴퓨터는 빈 공간이 생길 때까지 기다린다.

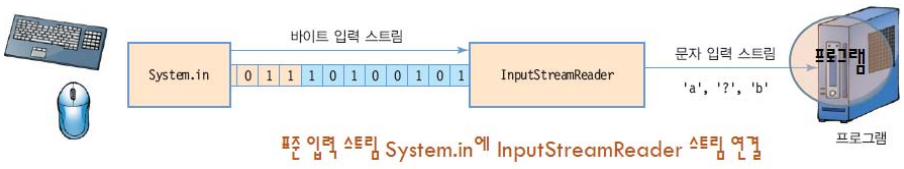
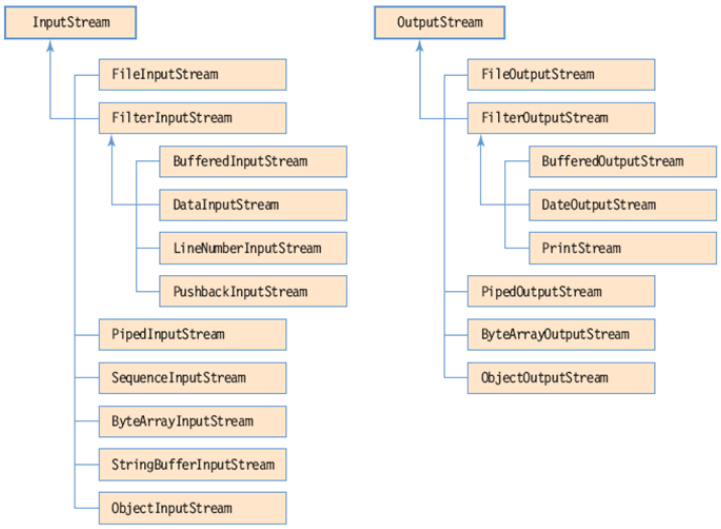
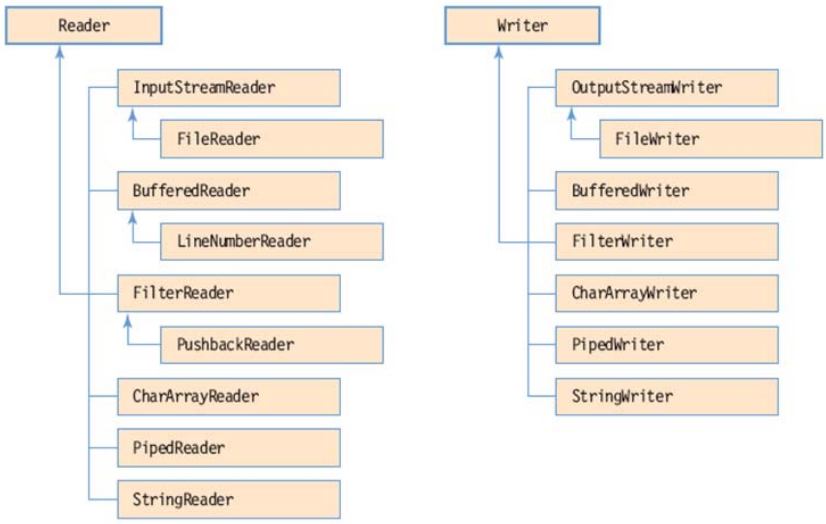


그림 7.2

7.1.2 바이트 스트림 클래스 계층 구조



7.1.3 문자 스트림 클래스 계층 구조



7.1.4 바이트 스트림

-바이트 스트림

■ 바이트 단위의 바이너리 값을 읽고 쓰는 스트림

■ 모든 바이트 스트림은 InputStream과 OutputStream의 서브 클래스

-바이트 스트림 클래스

■ InputStream/OutputStream

- java.io 패키지에 포함
- 추상 클래스로서 바이트 입출력 스트림을 나타내는 모든 클래스의 수퍼 클래스

■ FileInputStream/FileOutputStream

- 파일로부터 바이트 데이터를 읽거나 파일로 저장하는 클래스
- 바이너리 데이터인 raw 데이터의 입출력.

■ DataInputStream/DataOutputStream

- 자바의 기본 데이터 타입의 값을 바이너리 값으로 입출력
- String을 바이너리 값으로 입출력

7.1.5 FileInputStream 사용 예

-c:₩test.txt 파일로부터 바이트 단위로 읽을 수 있는 바이트 스트림 생성

```
FileInputStream fin = new FileInputStream("c:\\test.txt");
```

-파일 전체를 읽어 화면에 출력

```
FileInputStream fin = new FileInputStream("c:\\test.txt");
int c;
while((c = fin.read()) != -1) { // 파일 끝까지 반복하며 한 바이트씩 c에 읽어들인다.
   System.out.print((char)c); // 바이트 c를 문자로 변환하여 화면에 출력한다.
}
fin.close(); // 스트림을 닫는다. 더 이상 스트림으로부터 읽을 수 없다.
```

-예제 7-1 : System.ini 파일을 읽어 화면에 출력하기

■ FileInputStream을 이용하여 사용자 컴퓨터의 windows 디렉터리에 있는 System.ini 파일을 읽고 화면에 출력하라. System.ini 파일은 텍스트 파일이다.

```
import java.io.*;

public class FileInputStreamEx {
   public static void main(String[] args) {
      FileInputStream in = null;
      try {
         // 파일과 연결될 바이트 스트림 생성
         in = new FileInputStream("c:\\windows\\system.ini");
                  키보드에서 ctrl-z를 누르면 입력을 종료한다.
         int c;
         while ((c = in.read()) != -1) { // -1을 만나면 더 이상 입력이 없음
            System.out.print((char)c); // 바이트 c를 문자로 변환하여 화면에 출력
         }
         in.close(); //스트림을 닫는다. 더 이상 스트림으로부터 읽을 수 없다.
      } catch (IOException e) {
        System.out.println("입출력 오류");
      }
   }
}
```

```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

[drivers]
wave=mmdrv.dll
timer=timer.drv

[mci]
```

7.1.6 FileOutputStream을 이용한 파일 출력 스트림

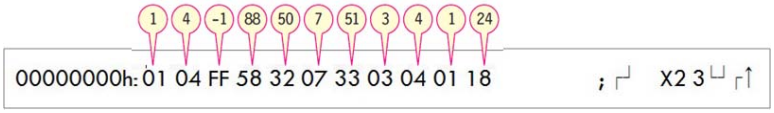
-c:₩test.out 파일에 바이트 단위로 쓸 수 있는 바이트 스트림을 생성 코드

```
FileOutputStream fout = new FileOutputStream("c:\\test.out");
```

-파일에 데이터 기록

```
FileOutputStream fout = new FileOutputStream("c:\\test.out");
int num[]={1,4,-1,88,50};
byte b[]={7,51,3,4,1,24};
for(int i=0; i<num.length; i++)
   fout.write(num[i]); // 파일에 정수 값을 바이트 정보로 기록한다.
fout.write(b); // 파일에 바이트 배열의 내용을 모두 그대로 기록한다.
fout.close();
```

-c:₩test.out 파일에 기록된 내용



-예제 7-2 : FileOutputStream과 DataOutputStream을 이용한 파일 출력

■ DataOutputStream을 이용하여 자바 기본 데이터 타입 값을 tmp.out 파일에 저장하고 파일의 내부를 살펴보라.

```
import java.io.*;

public class DataOutputStreamEx {
    public static void main(String[] args){
        FileOutputStream f = null;
        DataOutputStream out = null;
        try {
            f = new FileOutputStream("c:\\temp\\tmp.out");
            out = new DataOutputStream(f);
            out.writeBoolean(false); // 0x00 출력
            out.writeByte(0x33); // 0x33 출력
            out.writeShort(0x1234); // 0x1234 출력
            out.writeInt(0x5678); // 0x00005678 출력
            out.writeLong(0x12abcdef); // 0x0000000012abcdef 출력
            out.writeChar('c'); // 0x0063 출력
            out.writeDouble(0.12e-3); // long으로 변환 후 0x3f1f75104d551d69 출력
            out.writeFloat((float) 3.14); // int로 변환 후 0x4048f5c3 출력
            out.writeUTF("가나다라"); // UTF-8로 인코딩 후 0x000ceab080eb8298eb8ba4eb9dbc 출력
            out.close();
            f.close();
        } catch (IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```

-예제 실행 결과 - 출력 파일의 내용

■ 출력된 tmp.out 파일은 텍스트 파일이 아니므로 바이너리로 파일을 열어 16진수 형식으로 보면 다음과 같다.

```
00000000h: 00 33 12 34 00 00 56 78 00 00 00 00 12 AB CD EF ; .3.4..Vx.....ㅊ?
00000010h: 00 63 3F 1F 75 10 4D 55 1D 69 40 48 F5 C3 00 0C ; .c?.u.MU.i@H憮..
00000020h: EA B0 80 EB 82 98 EB 8B A4 EB 9D BC             ; 媛?섎뜝??
```

-예제 7-3 : FileInputStream과 DataInputStream을 이용하여 파일로부터 기본 데이터 타

입 읽기

■ DataInputStream을 이용하여 앞의 예제에서 만든 tmp.out 파일을 읽어 값을 화면에 출력하고 앞 예제에서 기록된 대로 읽혀졌는지 확인하라.

```
import java.io.*;

public class ReadData {
    public static void main(String[] args){
        FileInputStream f = null;
        DataInputStream in = null;
        try {
            f = new FileInputStream("c:\\temp\\tmp.out");
            in = new DataInputStream(f); // 파일에 연결된 데이터 입력 스트림 생성
            System.out.println(in.readBoolean()); // 1바이트 읽어 boolean값 반환
            System.out.printf("%x\n",(byte)in.readByte()); // 1바이트 읽어 byte값 반환
            System.out.printf("%x\n",in.readShort()); // 2바이트 읽어 short값 반환
            System.out.printf("%x\n",in.readInt()); // 4바이트 읽어 int값 반환
            System.out.printf("%x\n",in.readLong()); // 8바이트 읽어 long값 반환
            System.out.println((char)in.readChar()); // 2바이트 읽어 char값 반환
            System.out.println(in.readDouble()); // 8바이트 읽어 double값 반환
            System.out.println(in.readFloat()); // 4바이트 읽어 float값 반환
            System.out.println(in.readUTF()); // UTF-8로 인코딩된 문자열 읽어 반환
            in.close();
            f.close();
        } catch (IOException e) {
            System.out.println("입출력 오류");
        }
    }
}
```

```
false
33
1234
5678
12abcdef
c
1.2E-4
3.14
가나다라
```

7.1.7 문자 스트림

-문자 스트림

■ 유니 코드로 된 문자 단위의 데이터가 흐르는 스트림

■ 로컬 문자 집합으로 데이터를 자동 변환

■ 모든 문자 스트림은 Reader과 Writer의 서브 클래스이다.

-바이트 스트림과 차이점

■ 바이트 스트림은 이미지 파일, 동영상 파일 등과 같은 바이너리 데이터 스트림 처리에 적합

■ 문자 스트림은 문자 데이터 스트림만 처리할 수 있다.

-문자 입력 스트림의 종류

■ Reader, InputStreamReader, FileReader 등

-문자 출력 스트림의 종류

■ Writer, OutputStreamWriter, FileWriter 등

-문자 스트림의 종류

■ Reader/Writer

- java.io 패키지에 포함
- 추상 클래스로
- 문자 입출력 스트림을 나타내는 모든 클래스의 수퍼 클래스

■ InputStreamReader/OutputStreamWriter

- 바이트 스트림과 문자 스트림을 연결시켜주는 브릿지 역할
- 지정된 문자집합을 이용하여 입력 스트림에서 바이트를 읽어 문자로 인코드 또는 문자를 바이트로 디코드하여 출력 스트림으로 출력

■ FileReader/FileWriter

- 텍스트 파일로 문자 데이터를 입출력 한다.

• FileInputStream과 FileOutputStream을 이용하여 바이트 스트림을 문자 스트림으로 변환

-예제 7-4 : System.ini 파일 읽기

■ FileReader를 이용하여 사용자 컴퓨터의 windows 디렉터리에 있는 System.ini 파일을 읽고 화면에 출력하라. System.ini 파일은 텍스트 파일이다.

```
import java.io.*;

public class FileReaderEx {
   public static void main(String[] args) {
      FileReader in = null;
      try {
         // 파일로부터 문자 입력 스트림 생성
         in = new FileReader("c:\\windows\\system.ini");
          int c;
          while ((c = in.read()) != -1) { // 문자 단위로 읽는다.
              System.out.print((char)c);
          }
         in.close();
      } catch (IOException e) {
         System.out.println("입출력 오류");
      }
   }
}
```

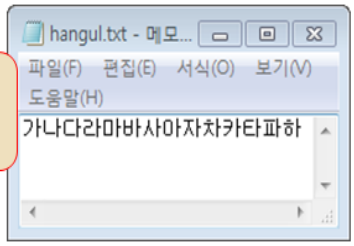
```
; for 16-bit app support
[386Enh]
woafont=dosapp.fon
EGA80WOA.FON=EGA80WOA.FON
EGA40WOA.FON=EGA40WOA.FON
CGA80WOA.FON=CGA80WOA.FON
CGA40WOA.FON=CGA40WOA.FON

[drivers]
wave=mmdrv.dll
timer=timer.drv

[mci]
```

-예제 7-5 : 문자 집합 지정이 잘못된 한글 파일 읽기

■ InputStreamReader의 문자 집합을 US-ASCII로 지정하여 한글 파일을 읽고 출력하라.

```
import java.io.*;

public class FileReadHangulEx {
   public static void main(String[] args) {
      InputStreamReader in = null;
      FileInputStream fin = null;
      try {
         fin = new FileInputStream("c:\\temp\\hangul.txt");
         in = new InputStreamReader(fin, "US-ASCII");
         int c;

         System.out.println("인코딩 문자 집합은 " + in.getEncoding()); // 문자 집합 출력
         while ((c = in.read()) != -1) { // 문자 단위로 읽는다.
             System.out.print((char)c);
         }
         in.close();
         fin.close();
      } catch (IOException e) {
         System.out.println("입출력 오류");
      }
   }
}
```

문자 집합 지정이 잘못된 경우의 예를 보이기 위해 일부러 틀린 문자 집합 지정



hangul.txt 내용

```
인코딩 문자 집합은 ASCII
??????????????????????????????
```

그림 7.3

-예제 7-6 : 한글 파일 읽기

■ 예제 7-5의 인코딩을 수정하여 정상적으로 한글 파일을 읽고 출력하라

```
import java.io.*;

public class FileReadHangul {
   public static void main(String[] args) {
      InputStreamReader in = null;
      FileInputStream fin = null;
      try {
         fin = new FileInputStream("c:\\temp\\hangul.txt");
         in = new InputStreamReader(fin, "MS949");
         int c;

         System.out.println("인코딩문자 집합은 " + in.getEncoding());
         while ((c = in.read()) != -1) {
            System.out.print((char)c);
         }
         in.close();
         fin.close();
      } catch (IOException e) {
         System.out.println("입출력오류");
      }
   }
}
```

MS에서 만든 한글 확장 완성형문자 집합

```
인코딩문자 집합은 MS949
가나다라마바사아자차카타파하
```

7.1.8 FileWriter 사용 예

-c:₩test.txt 파일에 문자 출력 스트림을 생성하는 코드

```
FileWriter fout = new FileWriter("c:\\tmp\\test.txt");
```

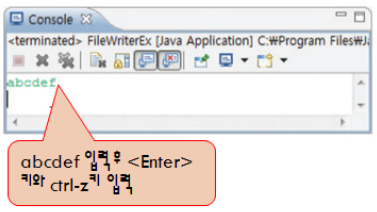
-파일에 문자 출력

```
FileWriter fout = new FileWriter("c:\\tmp\\test.txt");
fout.write('A'); // 한 문자 출력
fout.close();
```

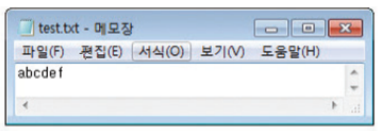
-예제 7-7 : 키보드 입력을 파일로 저장하기

■ 키보드에서 입력받은 데이터를 c:₩temp₩test.txt 파일에 저장하는 코드를 작성하라.

```
import java.io.*;

public class FileWriterEx {
   public static void main(String[] args) {
      InputStreamReader in = new
InputStreamReader(System.in);
      FileWriter fout = null;
      int c;
      try {
         fout = new FileWriter("c:\\temp\\test.txt");
         while ((c = in.read()) != -1) {
            fout.write(c);
         }
          in.close();
         fout.close();
      } catch (IOException e) {
         System.out.println("입출력 오류");
      }
   }
}
```



abcdef 입력후 <Enter> 키와 ctrl-z키 입력



실행 결과 test.txt 파일 생성

7.1.9 버퍼 입출력 스트림과 버퍼 입출력의 특징

-버퍼 입출력(Buffered I/O)이란?

■ 버퍼 입출력은 입출력 장치와 프로그램 사이에 버퍼를 두어 효율적으로 입출력 처리

-버퍼 입출력의 필요성

■ 프로그램의 실행 속도와 입출력 장치의 데이타 처리 속도의 불일치에 대해 대처하기 쉬움

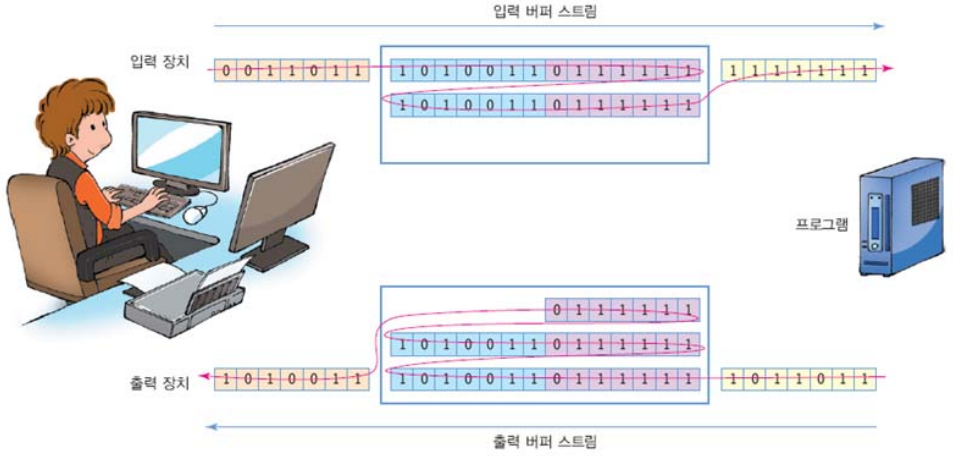


그림 7.4

7.1.10 버퍼 스트림의 종류

-바이트 스트림

■ 바이트 단위의 바이너리 데이타를 처리하는 버퍼 스트림

■ BufferedInputStream와 BufferedOutputStream

-문자 스트림

■ 유니코드의 문자 데이타만을 처리하는 버퍼 스트림

■ BufferedReader와 BufferedWriter

-BufferedOutputStream 사용 예

■ 20바이트 크기의 버퍼를 사용하고, System.out 표준 스트림에 출력하는 버퍼 스트림 생성

```
BufferedOutputStream out = new BufferedOutputStream(System.out, 20);
```

■ 파일 전체를 읽어 화면에 출력

```
BufferedOutputStream bout = new BufferedOutputStream(System.out, 20);
FileReader fin = new FileReader("c:\\windows\\system.ini");
int c;
while ((c = fin.read()) != -1) {
   bout.write((char)c);
}
fin.close();
bout.close();
```

■ 버퍼에 남아 있는 데이터 출력

```
bout.flush();
```

-예제 7-8 : 버퍼 출력 이용 예제

■ 버퍼 크기가 5인 출력 버퍼 스트림을 표준 출력 스트림과 연결하고 키보드에서 입력받은 문자를 출력 스트림에 출력하고, 입력의 끝을 알리면 버퍼에 남아 있는 모든 문자를 출력하는 프로그램을 작성하라.

```
import java.io.*;

public class BufferedIOEx {
    public static void main(String[] args) {
        InputStreamReader in = new InputStreamReader(System.in);
        BufferedOutputStream out = new BufferedOutputStream(System.out, 5);
        try {
            int c;
            while ((c = in.read()) != -1) {
                out.write(c);
            }
            out.flush(); // 버퍼에 남아 있던 문자 출력
            if (in != null) {
                in.close();
                out.close();
            }
        } catch (IOException e) {
            System.out.println("입출력오류");
        }
    }
}
```

ctrl-z가 입력될 때까지 반복

실제 버퍼가 다 찰 때만 문자가 화면으로 출력

버퍼가 5이므로 "12345678" 을 입력하고 리턴을 치면 화면에 "12345" 가먼저 출력 되고 ctrl-z를 눌러 입력의 끝을 알리면 버퍼에 남아 있던 "678" 이출력된다

```
12345678
12345678
```

7.1.11 File 클래스

-File 클래스의 특징

■ 파일의 경로명을 다루는 클래스
  • File 클래스는 파일과 디렉터리 경로명의 추상적 표현

■ 파일 클래스의 객체는 파일 또는 디렉터리를 다룸

■ File 클래스는 파일의 삭제, 디렉터리 생성 등과 같은 파일 관리
  • 파일의 내용 읽고 쓰기 - 파일 스트림 이용

■ File 객체와 스트림을 연결하여 파일 접근

7.1.12 File 클래스 생성자와 주요 메소드

표 7.1

메소드	설명
File(File parent, String child)	추상 경로명 parent가 나타내는 디렉터리에 문자열 child가 나타내는 새로운 File 객체 생성
File(String pathname)	문자열 pathname이 나타내는 File 객체 생성
File(String parent, String child)	문자열 parent가 나타내는 디렉터리에 문자열 child가 나타내는 새로운 File 객체 생성
File(URI uri)	file:URI를 추상 경로명으로 변환하여 File 객체 생성

메소드	설명
boolean mkdir()	추상 경로명이 나타내는 새로운 디렉터리 생성
String[] list()	추상 경로명이 나타내는 디렉터리 내에 있는 파일과 디렉터리 이름을 문자열 배열로 반환
String[] listFiles()	추상 경로명이 나타내는 디렉터리 내에 있는 파일의 이름을 문자열 배열로 반환
boolean renameTo(File dest)	dest가 지정하는 추상 경로명으로 파일 이름 변경
boolean delete()	추상 경로명이 나타내는 파일 또는 디렉터리 삭제
long length()	추상 경로명이 나타내는 파일의 크기 반환
String getPath()	추상 경로명 전체를 문자열로 변환하여 반환
String getName()	추상 경로명이 나타내는 파일 또는 디렉터리 이름을 문자열로 변환하여 반환
boolean isFile()	추상 경로명이 일반 파일이면 true 반환
boolean isDirectory()	추상 경로명이 디렉터리이면 true 반환
long lastModified()	추상 경로명이 나타내는 파일이 마지막으로 변경된 시간 반환
boolean exists()	추상 경로명이 나타내는 파일 또는 디렉터리가 존재하면 true 반환

7.1.13 File 클래스 사용 예

-c:₩test.txt 파일을 나타내는 File 객체를 생성

```
File f = new File("c:\\test.txt");
```

-파일 관리

■ 파일 타입 구분하기, isFile()과 isDirectory()

```
File f = new File("c:\\windows\\system.ini");
String res;
if(f.isFile()) // 파일 타입이면
   res = "파일";
else // 디렉터리 타입이면
   res = "디렉터리";
System.out.println(f.getPath() + "은 " + res + "입니다.");
```

```
c:\windows\system.ini은 파일입니다.
```

■ 디렉터리에 있는 파일 리스트 얻기, list()

```
File f = new File("c:\\tmp\\java_sample");
String[] filenames = f.list(); // 파일명 리스트 얻기
for (int i=0; i<filenames.length; i++) {
   File sf = new File(f, filenames[i]);
   System.out.print(filenames[i]);
   System.out.print("\t파일크기: " + sf.length());
}
```

-예제 7-9 : File 클래스 활용한 파일 관리

■ File 클래스를 이용하여 파일의 타입을 알아내고, 디렉터리에 있는 파일들을 나열하며, 디렉터리 이름을 변경하는 프로그램을 작성해보자.

```
import java.io.File;

public class FileClassExample {
    // 디렉터리에 포함된 파일과 디렉터리의 이름, 크기, 수정 시간을 출력하는 메소드
    public static void dir(File fd) {
        String[] filenames = fd.list(); // 디렉터리에 포함된 파일 리스트 얻기
        for (String s : filenames) {
            File f = new File(fd, s);
            long t = f.lastModified(); // 마지막으로 수정된 시간
            System.out.print(s);
            System.out.print("\t파일크기: " + f.length()); // 파일 크기
            System.out.printf("\t수정한 시간: %tb %td %ta %tT\n",t,t,t,t);
        }
    }
    public static void main(String[] args) {
        File f1 = new File("c:\\windows\\system.ini");
        File f2 = new File("c:\\tmp\\java_sample");
        File f3 = new File("c:\\tmp");
        String res;
        if(f1.isFile()) // 파일 타입이면
            res = "파일";
        else // 디렉터리 타입이면
            res = "디렉터리";
        System.out.println(f1.getPath() + "은 " + res + "입니다.");
        if (!f2.exists()) { //f2가 나타내는 파일이 존재하는지 검사
            if (!f2.mkdir()) // 존재하지 않으면 디렉터리 생성
                System.out.println("디렉터리 생성 실패");
        }
        if(f2.isFile()) // 파일 타입이면
            res = "파일";
        else // 디렉터리 타입이면
            res = "디렉터리";
        System.out.println(f2.getPath() + "은 " + res + "입니다.");
        dir(f3); // c:\tmp에 있는 파일과 디렉터리 화면에 출력
        f2.renameTo(new File("c:\\tmp\\javasample")); // 파일 이름 변경
        dir(f3);
    }
}
```

```
c:\windows\system.ini은 파일입니다.
c:\tmp\java_sample은 디렉터리입니다.
hangul.txt      파일크기: 28    수정한 시간: 11월 29 일 21:04:46
Hello.java      파일크기: 469     수정한 시간: 10월 06 수 13:23:59
Hello2010.java  파일크기: 126   수정한 시간: 10월 06 수 10:01:56
HelloDoc.java   파일크기: 669   수정한 시간: 10월 06 수 14:23:32
java_sample     파일크기: 0     수정한 시간: 11월 14 일 16:46:27
hangul.txt      파일크기: 28    수정한 시간: 11월 29 일 21:04:46
Hello.java      파일크기: 469     수정한 시간: 10월 06 수 13:23:59
Hello2010.java  파일크기: 126   수정한 시간: 10월 06 수 10:01:56
HelloDoc.java   파일크기: 669   수정한 시간: 10월 06 수 14:23:32
javasample      파일크기: 0     수정한 시간: 11월 14 일 16:46:27
```

-예제 7-10 : 텍스트 파일 복사

■문자 스트림을 이용하여 텍스트 파일을 복사하는 프로그램을 작성하라

```
import java.io.*;

public class TextCopy {
    public static void main(String[] args){
        File src = new File("c:\\windows\\system.ini"); // 소스 파일
        File dst = new File("c:\\tmp\\system.txt"); // 목적 파일
        FileReader fr = null;
        FileWriter fw = null;
        BufferedReader in = null;
        BufferedWriter out = null;
        int c;
        try {
            fr = new FileReader(src);
            fw = new FileWriter(dst);
            in = new BufferedReader(fr);
            out = new BufferedWriter(fw);
            while ((c = in.read()) != -1) {
                out.write((char)c);
            }
            in.close();
            out.close();
            fr.close();
            fw.close();
        } catch (IOException e) {
            System.out.println("파일복사 오류");
        }
    }
}
```

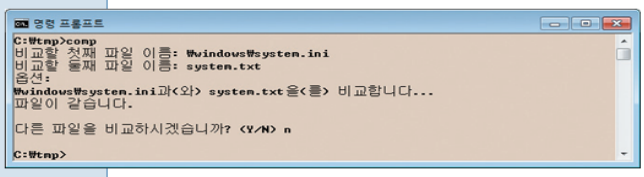


그림 7.5

-예제 7-11 : 바이너리 파일 복사

■바이트 스트림을 이용하여 바이너리 파일을 복사하는 프로그램을 작성하라

```
import java.io.*;

public class BinaryCopy {
    public static void main(String[] args) {
        File src = new File("c:\\windows\\explorer.exe"); // 복사할 파일
        File dst = new File("c:\\tmp\\explorer.bin"); // 복사될 파일
        FileInputStream fi = null;
        FileOutputStream fo = null;
        BufferedInputStream in = null;
        BufferedOutputStream out = null;
        int c;
        try {
            fi = new FileInputStream(src); // 파일 입력 바이트 스트림에 연결
            fo = new FileOutputStream(dst); // 파일출력 바이트 스트림에 연결
            in = new BufferedInputStream(fi); // 버퍼 입력 스트림에 연결
            out = new BufferedOutputStream(fo); // 버퍼 출력 스트림에 연결
            while ((c = in.read()) != -1) {
                out.write((char)c);
            }
            in.close();
            out.close();
            fi.close();
            fo.close();
        } catch (IOException e) {
            System.out.println("파일복사 오류");
        }
    }
}
```

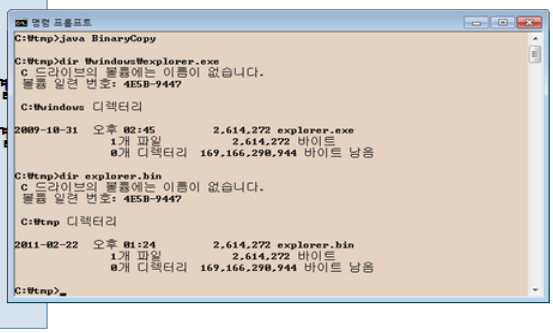


그림 7.6

제8장 제네릭과 컬렉션

8.1 컬렉션(collection)의 개념

8.1.1 컬렉션

-컬렉션은 요소(element)라고 불리는 가변 개수의 객체들의 모음

■ 객체들의 컨테이너라고도 불림

-컬렉션 클래스 사례

■ Vector, ArrayList, Hashtable, LinkedList, HashMap, HashSet, Stack

-java.util 패키지에서 제공

-다양한 객체들을 삽입, 삭제, 검색 등 관리하는데 사용됨

8.1.2 배열과 컬렉션의 개념 차이

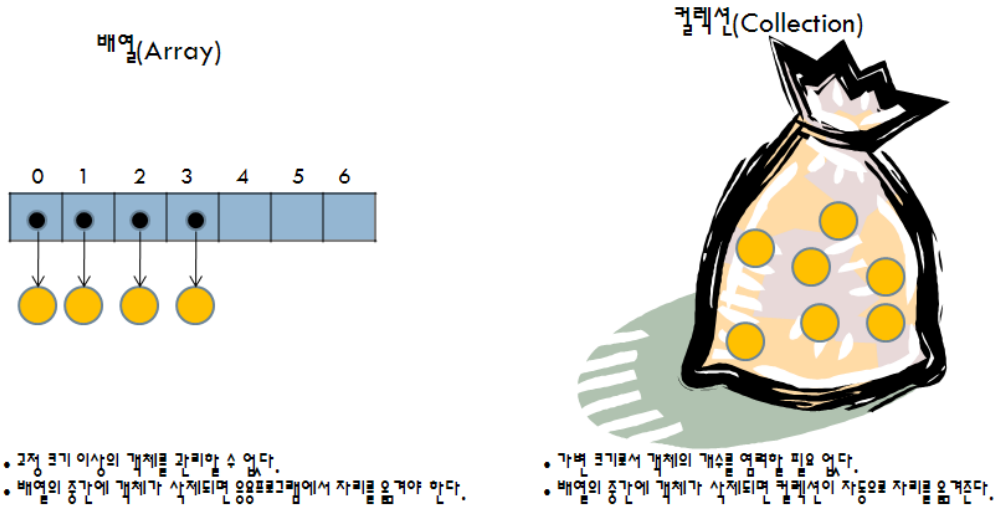


그림 8.1

8.1.3 컬렉션을 위한 인터페이스와 클래스

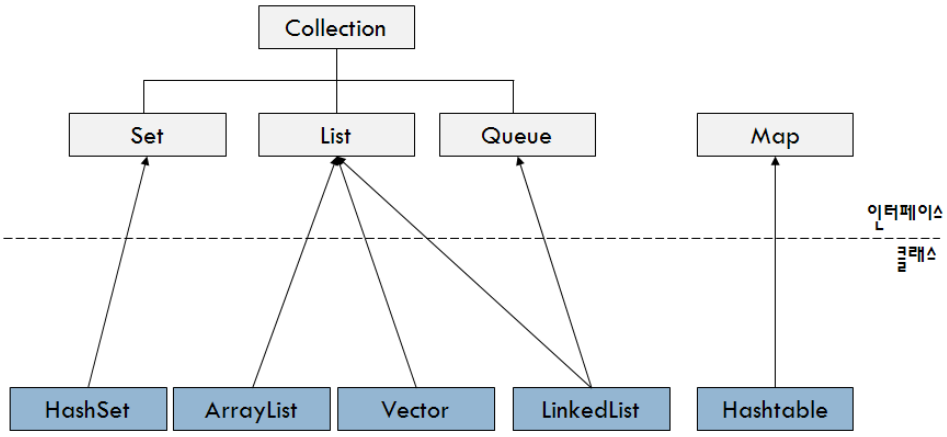


그림 8.2

8.1.4 Vector 클래스

-Vector의 특성

■ java.util.Vector

■ 여러 객체들을 삽입, 삭제, 검색하는 컨테이너 클래스

• 배열의 길이 제한 단점 극복
• 원소의 개수가 넘쳐나면 자동으로 늘어나는 가변 길이

■ Vector에 삽입 가능한 것
• 객체만 가능
• null도 삽입 가능
• 기본 데이터 타입은 불가능
  -Wrapper 객체로 만들어 삽입 가능

■ Vector에 객체 삽입
• 벡터의 맨 뒤에 객체 추가 : 공간이 모자라면 자동 늘림
• 벡터 중간에 객체 삽입 : 삽입된 뒤의 객체는 뒤로 하나씩 이동

■ Vector에서 객체 삭제
• 임의의 위치에 있는 객체 삭제 가능 : 객체 삭제 후 하나씩 앞으로 자동 이동

8.1.5 Vector 객체의 내부 구성

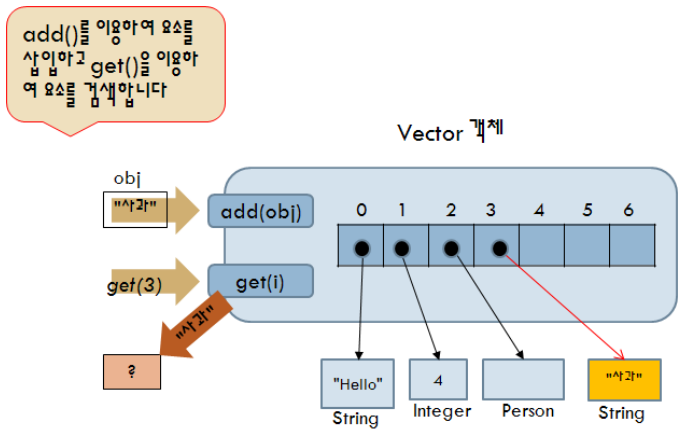


그림 8.3

8.1.6 Vector 클래스의 주요 메소드

표 8.1

메소드	설명
boolean add(E e)	벡터의 맨 뒤에 요소 추가
void add(int index, E element)	지정된 인덱스에 지정된 객체를 삽입
int capacity()	벡터의 현재 용량 반환
booleanaddAll(Collection<? extends E> c)	c가 지정하는 컬렉션의 모든 요소를 벡터의 맨 뒤에 추가
void clear()	벡터의 모든 요소 삭제
boolean contains(Object o)	벡터가 지정된 객체를 포함하고 있으면 true 반환
E elementAt(int index)	지정된 인덱스의 요소 반환
E get(int index)	지정된 인덱스의 요소 반환
intindexOf(Object o)	지정된 객체와 같은 첫 번째 요소의 인덱스 반환. 없으면 -1 반환
booleanisEmpty()	벡터가 비어있으면 true 반환
E remove(int index)	지정된 인덱스의 요소 삭제

boolean remove(Object o)	지정된 객체와 같은 첫 번째 요소를 벡터에서 삭제
void removeAllElements()	벡터의 모든 요소를 삭제하고 크기를 0으로 만듦
int size()	벡터가 포함하는 요소의 개수 반환
Object[] toArray()	벡터의 모든 요소를 포함하는 배열을 반환

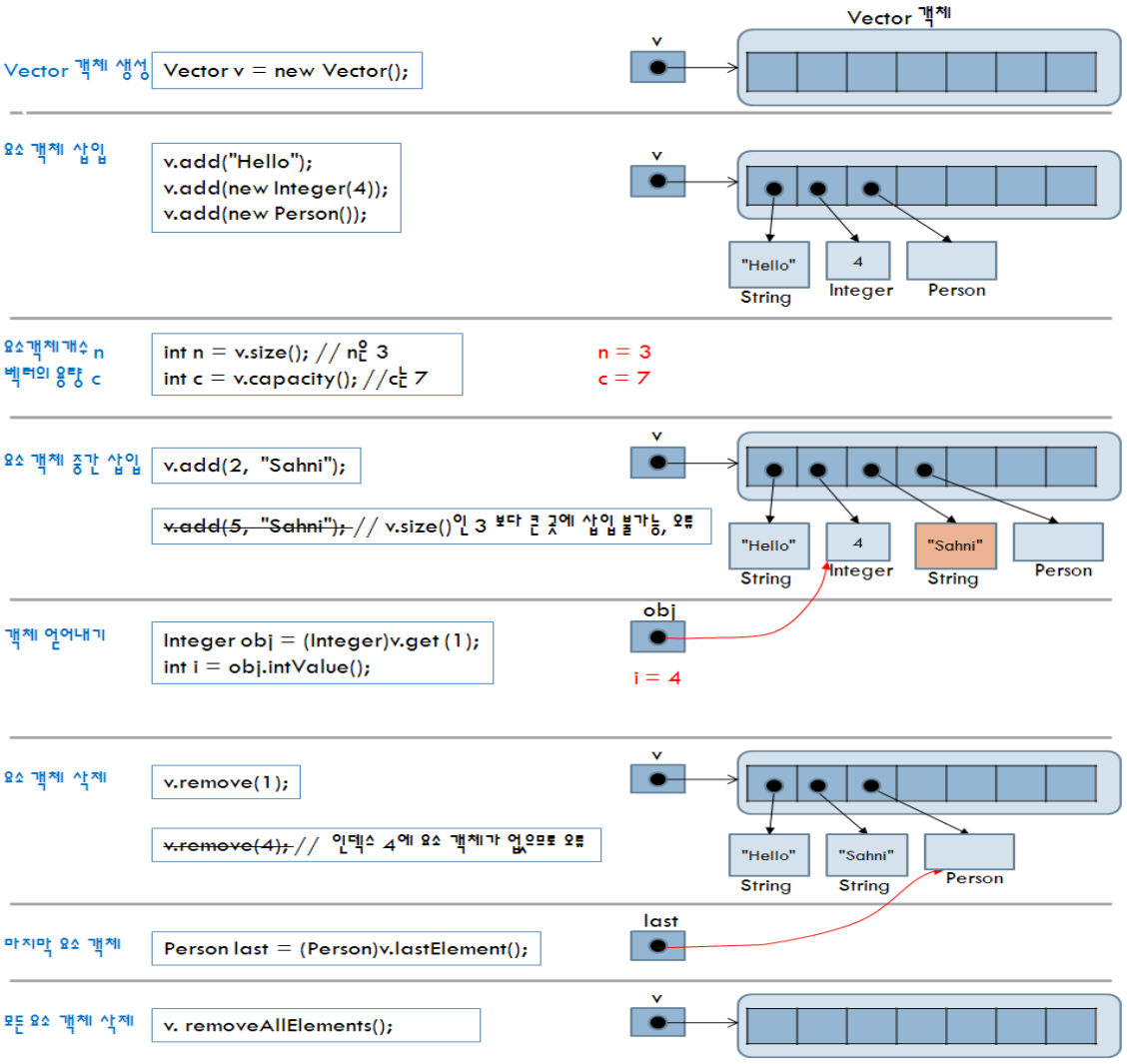


그림 8.4

8.1.7 컬렉션과 자동 박싱/언박싱

-JDK 1.5 이전 버전

■ Wrapper 클래스를 이용하여 기본 타입을 객체로 만들어 사용

```
Vector v = new Vector();
v.add(new Integer(4));
v.add(new Character('r'));
v.add(new Double(3.14));
```

■ 컬렉션으로부터 요소를 얻어올 때도 해당 타입의 값을 얻어오는 메소드를 호출

```
Integer n = (Integer)v.get(0);
int k = n.intValue(); // k = 4
```

-JDK 1.5부터

■ 자동 박싱/언박싱의 기능 추가

```
Vector v = new Vector();
v.add(4); // 4 → new Integer(4)로 자동 박싱
v.add('r'); // 'r' → new Character('r')로 자동 박싱
v.add(3.14); // 3.14 → new Double(3.14)로 자동 박싱
int k = v.get(0); // Integer 타입이 int 타입으로 자동 언박싱, k = 4
```

-예제 8-1 : 벡터 내의 모든 요소 객체 출력하기

■ 다음 코드에 대한 결과는 무엇인가?

```
import java.util.Vector;

public class VectorEx {
   public static void main(String[] args) {
      Vector v = new Vector();
      v.add("Hello");
      v.add(new Integer(4));
      v.add(new Double(3.14));
      System.out.println("벡터내의 요소 객체 수:"+v.size());
      System.out.println("벡터의 현재 용량:"+v.capacity());

      for(int i=0; i<v.size(); i++) {
         Object obj = v.get(i);
         if(obj instanceof String) {
            String str = (String)obj;
            System.out.println(str);
         }
         else if(obj instanceof Integer) {
            Integer x = (Integer)obj;
            int n = x.intValue();
            System.out.println(n);
         }
         else if(obj instanceof Double) {
            Double y = (Double)obj;
            double d = y.doubleValue();
            System.out.println(d);
         }
      }
   }
}
```

```
벡터내의 요소 객체 수:3
벡터의 현재 용량:10
Hello
4
3.14
```

-예제 8-2 : 벡터에 있는 객체 중에서 정수 값만 모두 더하기

■ 벡터에 있는 객체 중에서 정수 값만 모두 더하는 프로그램을 작성해보라

```
import java.util.Vector;

public class VectorAddEx {
   public static void main(String [] args) {
      Vector v = new Vector();
      v.add("Hello");
      v.add(new Integer(4));
      v.add(new Double(3.14));
      v.add(new Integer(5));

      int sum = 0;
      for(int i=0; i<v.size(); i++) {
         Object obj = v.get(i);
         if(obj instanceof Integer) {
            Integer x = (Integer)obj;
            int n = x.intValue();
            sum += n;
         }
      }
      System.out.println("모든 정수의 합은 : " + sum);
   }
}
```

```
모든 정수의 합은 : 9
```

8.1.8 ArrayList 클래스

-ArrayList의 특성

■ java.util.ArrayList

■ 가변 크기 배열을 구현한 클래스

■ ArrayList에 삽입 가능한 것

- 객체만 가능
- 기본 데이타 타입은 불가능
- Wrapper 객체로 만들어 삽입 가능
- null도 삽입 가능

■ ArrayList에 객체 삽입

- 리스트의 맨 뒤에 객체 추가 : 공간이 모자라면 자동 늘림
- 리스트의 중간에 객체 삽입 : 삽입된 뒤의 객체는 뒤로 하나씩 이동

■ ArrayList 에서 객체 삭제

- 임의의 위치에 있는 객체 삭제 가능 : 객체 삭제 후 하나씩 앞으로 이동

■ 벡터와 거의 유사하며 자동으로 스레드 동기화를 지원하지 않는 점이 가장 큰 차이

- 다수의 스레드가 동시에 ArrayList에 접근할 때 ArrayList는 동기화시키지 않음
- ArrayList에 접근하는 곳에서 스레드 동기화를 수행하여야 함

8.1.9 ArrayList 객체의 내부 구성

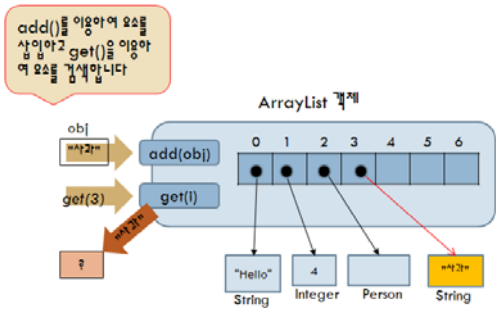


그림 8.5

8.1.10 ArrayList 클래스의 주요 메소드

표 8.2

메소드	설명
boolean add(E e)	ArrayList의 맨 뒤에 요소 추가
void add(int index, E element)	지정된 인덱스에 지정된 객체를 삽입
booleanaddAll(Collection<? extends E> c)	c가 지정하는 컬렉션의 모든 요소를 ArrayList의 맨 뒤에 추가
void clear()	ArrayList의 모든 요소 삭제
boolean contains(Object o)	ArrayList가 지정된 객체를 포함하고 있으면 true 반환

E elementAt(int index)	지정된 인덱스의 요소 반환
E get(int index)	지정된 인덱스의 요소 반환
int indexOf(Object o)	지정된 객체와 같은 첫 번째 요소의 인덱스 반환. 없으면 -1 반환
boolean isEmpty()	ArrayList가 비어있으면 true 반환
E remove(int index)	지정된 인덱스의 요소 삭제
boolean remove(Object o)	지정된 객체와 같은 첫 번째 요소를 ArrayList에서 삭제
int size()	ArrayList가 포함하는 요소의 개수 반환
Object[] toArray()	ArrayList의 모든 요소를 포함하는 배열을 반환

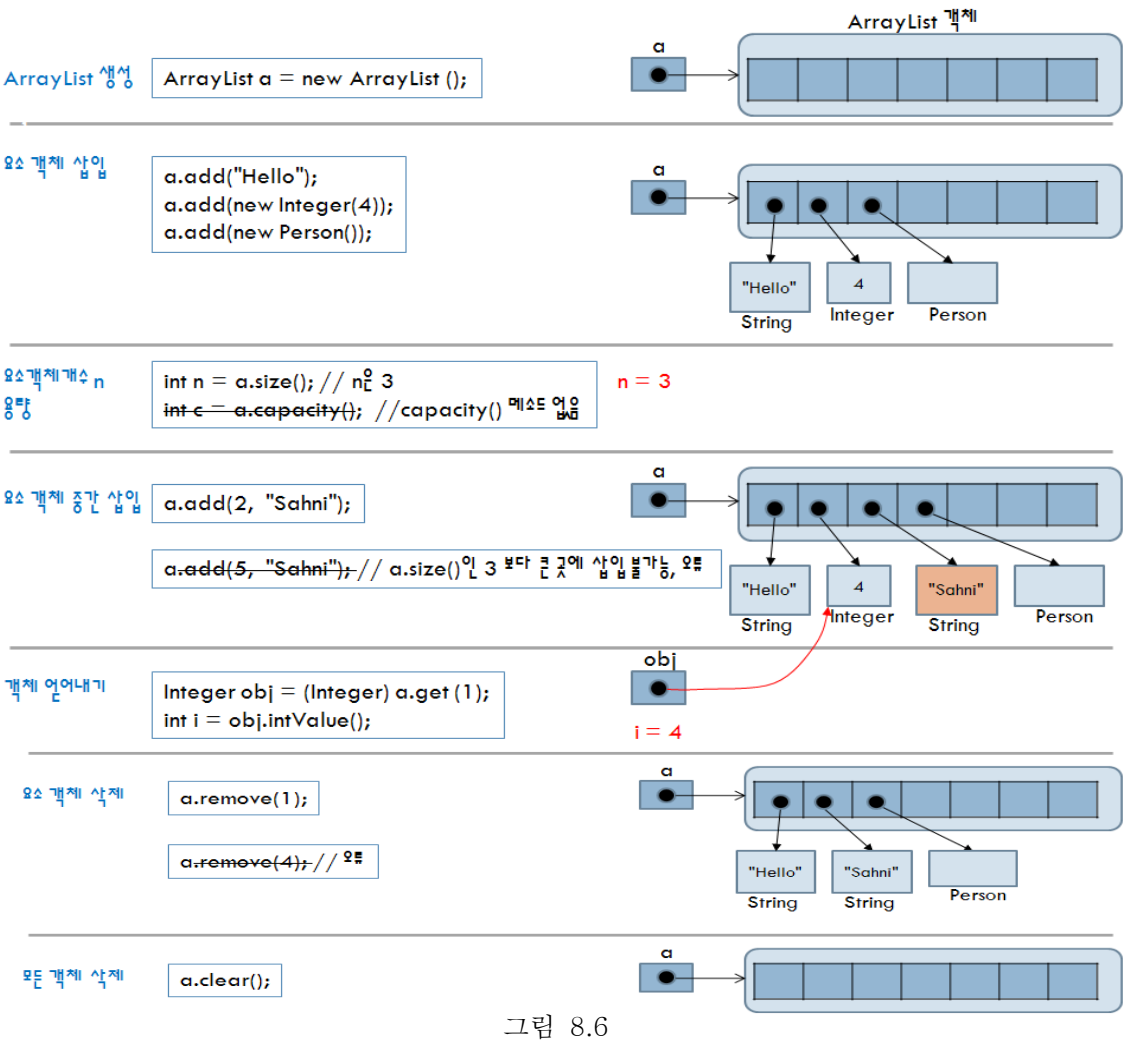


그림 8.6

-예제 8-3 : ArrayList 내의 모든 요소 객체 출력하기

■ 다음 코드에 대한 결과는 무엇인가?

```
import java.util.ArrayList;

public class ArrayListEx {
   public static void main(String[] args) {
      ArrayList a = new ArrayList();
      a.add("Hello");
      a.add(new Integer(3));
      a.add(new Double(3.14));
      a.add(new Double(3.4));
      a.remove(1);
      for(int i=0; i<a.size(); i++) {
         Object obj = a.get(i);
         if(obj instanceof String) {
            String str = (String)obj;
            System.out.println(str);
         }
         else if(obj instanceof Integer) {
            Integer x = (Integer)obj;
            int n = x.intValue();
            System.out.println(n);
         }
         else if(obj instanceof Double) {
            Double y = (Double)obj;
            double d = y.doubleValue();
            System.out.println(d);
         }
      }
   }
}
```

```
Hello
3.14
3.4
```

8.1.11 Hashtable 클래스

-Hashtable의 특성

■ java.util.Hashtable

■ 삽입 및 검색이 빠른 특징

■ 키(key)와 값(value) 사용

- 키와 값이 한쌍으로 삽입됨
- 키는 해쉬 테이블에 삽입되는 위치를 결정하는데 내부적으로 이용
- 값을 검색하기 위해서는 키를 반드시 이용함
- 키, 값 모두 객체만이 이용됨. 기본 데이터 타입은 사용할 수 없음

-Hashtable에 삽입, 검색하는 예

```
Hashtable h = new Hashtable(); // Hashtable 객체 생성

h.put("apple", "사과"); // 키는 "apple"이며 값이 "사과"인 요소 삽입
String s = (String)h.get("apple"); // "apple" 키의 값을 검색. s는 "사과"
```

8.1.12 Hashtable의 내부 구성과 put(), get() 메소드

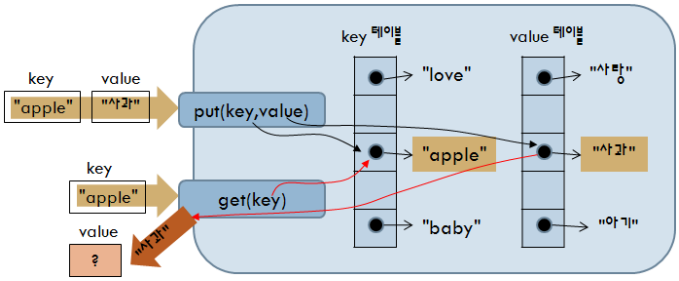


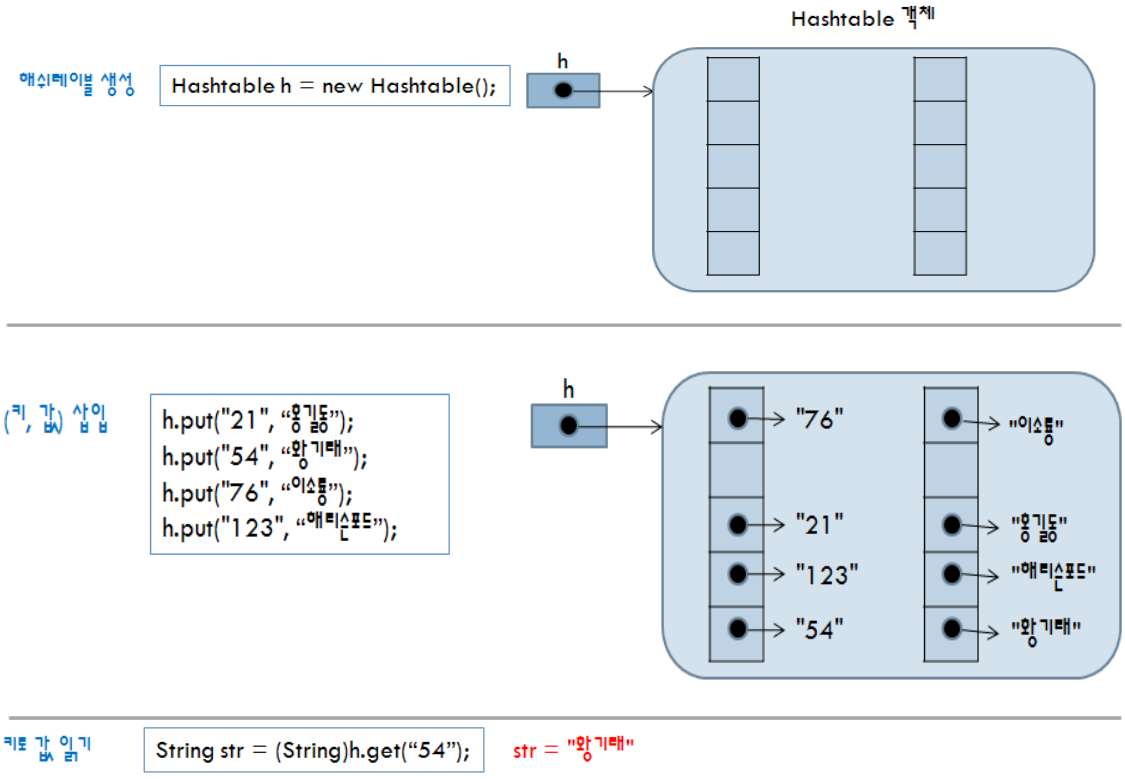
그림 8.7

8.1.13 Hashtable 클래스의 주요 메소드

표 8.3

메소드	설명
void clear()	Hashtable의 모든 키 삭제
boolean contains(Object o)	Hashtable의 어떤 키가 지정된 객체에 매핑되면 true 반환
booleancontainsKey(Object key)	Hashtable이 지정된 키를 포함하고 있으면 true 반환
booleancontainsValue(Object value)	Hashtable이 하나 이상의 키를 지정된 값에 매핑시킬 수 있으면 true 반환
Enumeration<V> elements()	Hashtable의 모든 값의 enumeration 반환
V get(Object key)	지정된 키에 맵핑되는 값을 반환하거나 맵핑되는 값이 없으면 null을 반환
booleanisEmpty()	Hashtable에 키가 없으면 true 반환
Enumeration<K> keys()	Hashtable의 모든 키의 enumeration 반환
V put(K key, V value)	Hashtable에 key를 value에 매핑
V remove(Object key)	지정된 키와 이에 매핑된 모든 값들을 Hashtable에서 삭제
int size()	Hashtable이 포함하는 키의 개수 반환

8.1.14 해쉬테이블의 조작 사례



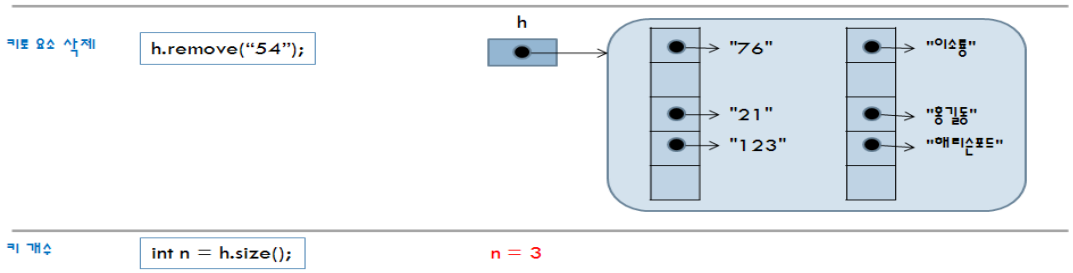


그림 8.8

-예제 8-4 : Hashtable 내의 모든 값 알아내기

■ 다음 코드에 대한 결과는 무엇인가?

```
import java.util.*;
public class HashtableEx {
   public static void main(String [] args) {
      Hashtable h = new Hashtable(); // 디폴트 용량 11의 Hashtable 생성
      h.put("21", "홍길동");
      h.put("54", "황기태");
      h.put("76", "이소룡");
      h.put("123", "애리슨포드");
      System.out.println("Hashtable의 키 개수 : "+h.size()); // 키의 개수

      Enumeration e = h.keys(); // Hashtable의 모든 키들을 얻어옴
      while(e.hasMoreElements()) {
         String key = (String)e.nextElement(); // 키
         String value = (String)h.get(key); // 키에 매핑된 값
         System.out.println(key + ":" + value);
      }
   }
}
```

```
Hashtable의 키 개수 : 4
76:이소룡
123:애리슨포드
21:홍길동
54:황기태
```

-예제 8-5 : Hashtable 검색

■ Scanner 클래스를 이용하여 이름, 전화번호를 입력받아 이름을 키로 하여 해시테이블에 저장하고 이름으로 전화번호를 검색하는 프로그램을 작성하라.

```
import java.util.*;

public class HashTableExample {
   public static void main(String[] args) {
      Hashtable members = new Hashtable();
      Scanner sin = new Scanner(System.in);
      System.out.println("공백으로 분리된 이름과 전화번호 5개를 입력하십시오.");
      for (int i = 0; i < 5; i++) {
         System.out.print("이름, 전화번호 : ");
         String name = sin.next(); // 이름 입력
         String tel = sin.next(); // 전화번호 입력
         members.put(name, tel); // 이름이 키, 전화번호를 값으로 Hashtable에 저장
      }
      System.out.println("전화번호를 검색할 이름을 입력하십시오.");
      String key = sin.next(); // 키인 이름 입력
      String val = (String)members.get(key); // 키로 매핑된 전화번호 검색
      if (val != null)
         System.out.println(key + "의 전화번호는 " + val + "입니다.");
      else
         System.out.println("입력하신 이름을 찾을 수 없습니다.");

   }
}
```

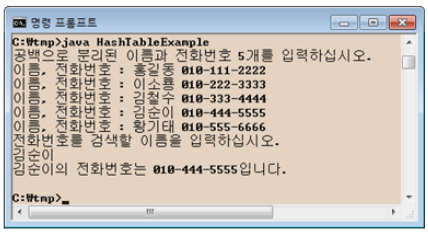


그림 8.9

8.1.15 LinkedList 클래스

-LinkedList의 특성

■ java.util.LinkedList
■ List 인터페이스를 구현한 클래스
■ Vector, ArrayList 클래스와 매우 유사
■ 요소 객체들은 양방향으로 연결되어 관리됨
■ 요소 객체는 맨 앞, 맨 뒤에 추가 가능
■ 요소 객체는 인덱스를 이용하여 중간에 삽입 가능
■ 맨 앞이나 맨 뒤에 요소를 추가하거나 삭제할 수 있어 스택이나 큐로 사용 가능

8.1.16 LinkedList의 내부 구성과 put(), get() 메소드

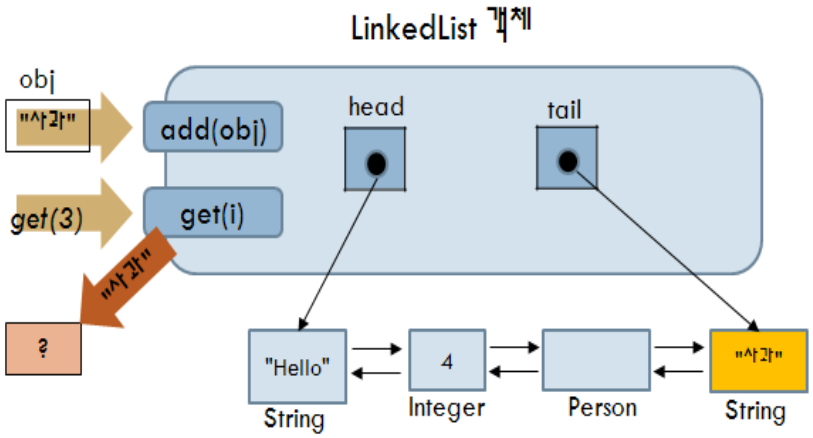


그림 8.10

8.1.17 Iterator 활용

-Iterator 인터페이스

■ Vector, ArrayList, LinkedList와 같은 리스트 자료구조에서 요소를 순차적으로 검색할 때 Iterator 인터페이스를 사용
■ iterator() 메소드를 호출하면 Iterator 객체를 반환
■ Iterator 객체는 검색한 위치를 기억하고 있어 인덱스 없이 순차적 검색이 가능
■ Iterator 인터페이스 메소드

표 8.4

메소드	설명
booleanhasNext()	다음 반복에서 사용될 요소가 있으면 true 반환
E next()	다음 요소 반환
void remove()	마지막으로 반환된 요소를 제거

-예제 8-6 : Iterator를 이용하여 ArrayList 내의 모든 요소 객체 출력하기

■ 다음 코드에 대한 결과는 무엇인가?

```
import java.util.*;

public class IteratorExample {
    public static void main(String[] args) {
        ArrayList a = new ArrayList(); // 빈 리스트 생성
        a.add("Hello");
        a.add(3); // 자동 박싱, JDK 1.5 이후에서만 실행됨
        a.add(3.14); // 자동 박싱
        a.add(2, 3.4); // 자동 박싱, 인덱스 2에 객체 삽입
        Iterator i = a.iterator(); // Iterator 객체 반환
        while (i.hasNext()) { // Iterator 객체에 요소가 있을 때 까지 반복
            Object obj = i.next(); // 다음 요소 반환
            if(obj instanceof String) { // String 객체의 경우
                String str = (String)obj;
                System.out.println(str);
            }
            else if(obj instanceof Integer) { // Integer 객체의 경우
                int n = (Integer)obj; // 자동 언박싱
                System.out.println(n);
            }
            else if(obj instanceof Double) { // Double 객체의 경우
                double d = (Double)obj; // 자동 언박싱
                System.out.println(d);
            }
        }
    }
}
```

```
Hello
3
3.4
3.14
```

8.1.18 Collections 클래스 활용

-Collections 클래스

■ java.util 패키지에 포함

■ 컬렉션에 대해 연산을 수행하고 결과로 컬렉션을 반환

■ 주요 메소드

- 컬렉션에 포함된 요소들을 소팅하는 sort() 메소드
- 요소의 순서를 반대로 하는 reverse() 메소드
- 요소들의 최대, 최소값을 찾아내는 max(), min() 메소드
- 특정 값을 검색하는 binarySearch() 메소드

■ 모두 static 메소드

-예제 8-7 : Collections 클래스의 활용

■ Collections를 사용한 다음 코드에 대한 결과는 무엇인가?

```
import java.util.*;

public class CollectionsExample {

    // 리스트의 요소를 모두 출력하는 메소드
    static void printList(LinkedList l) {
        Iterator iterator = l.iterator(); // Iterator 객체 반환
        while (iterator.hasNext()) {
            // Iterator 객체에 요소가 있을 때 까지 반복
            String e = (String)iterator.next(); // 다음 요소 반환
            String separator;
            if (iterator.hasNext())
                separator = "->"; // 마지막 요소가 아니면 -> 출력
            else
                separator = "\n"; // 마지막 요소이면 줄바꿈
            System.out.print(e+separator);
        }
    }

    public static void main(String[] args) {
        LinkedList myList = new LinkedList(); // 빈 리스트 생성

        myList.add("트랜스포머");
        myList.add("스타워즈");
        myList.add("매트릭스");
        myList.add(0,"터미네이터");
        myList.add(2,"아바타");
        Collections.sort(myList); // 요소 정렬
        printList(myList); // 정렬된 요소 출력
        Collections.reverse(myList); // 요소의 순서를 반대로
        printList(myList); // 요소 출력

        // "아바타" 요소의 인덱스 검색
        int index = Collections.binarySearch(myList, "아바타")+1;
        System.out.println("아바타는 " + index + "번째 요소입니다.");
    }
}
```

소팅된 순서대로 출력

거꾸로 출력

```
매트릭스->스타워즈->아바타->터미네이터->트랜스포머
트랜스포머->터미네이터->아바타->스타워즈->매트릭스
아바타는 3번째 요소입니다.
```

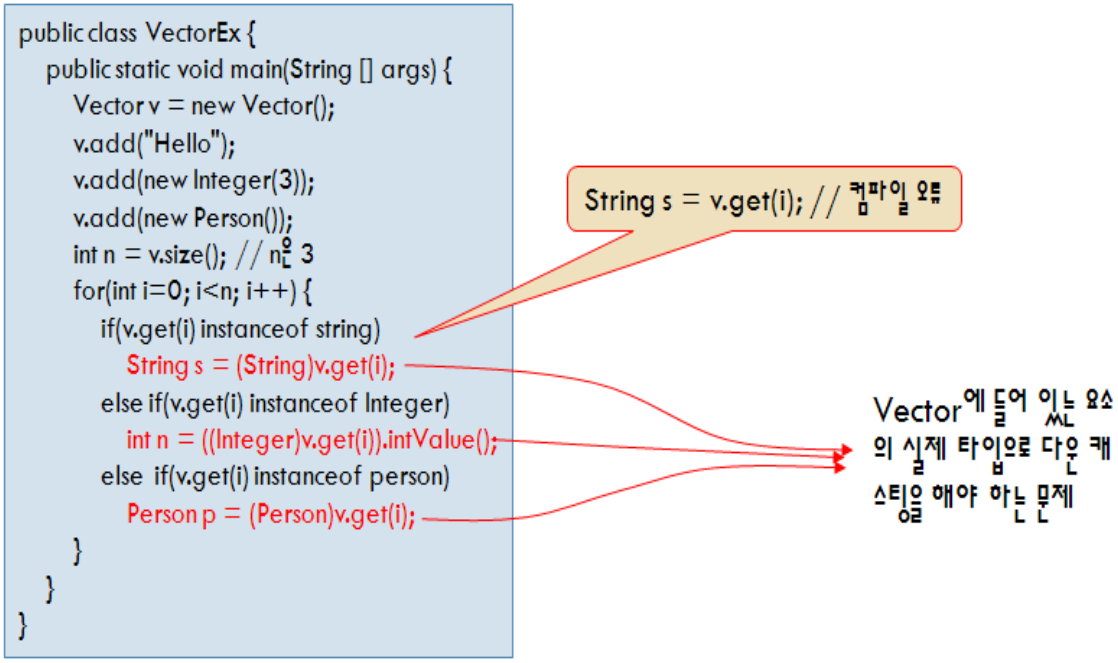
8.1.19 JDK 1.5 이전 자바의 컬렉션 문제점

-삽입되는 원소 객체는 사실 모두 Object 타입이어야 함

■ int, double 등의 기본 타입 데이타는 직접 삽입될 수 없음

-객체를 컬렉션에서 꺼내올 때 다운 캐스팅 필요

■ Object 타입에서 실제 클래스 타입으로의 다운 캐스팅 불편



8.1.20 JDK 1.5 이전 버전의 Vector 클래스의 예

-JDK 1.5 이전 버전의 Vector 클래스 예제

■ http://download.oracle.com/javase/1.4.2/docs/api/java/util/Vector.html참조

```
1: Vector myVector = new Vector();
2: myVector.add(new Integer(0));
3: Integer x = (Integer) myVector.elementAt(0);
```

■ 제네릭 도입 이전에는 Object 클래스를 이용하여 모든 객체들을 참조

■ JDK 1.5이전 Vector는 Object를 요소로 받아 들임

■ 3번 행의 다운 캐스팅이 반드시 필요하며 이 부분이 문제를 발생시킨다.

■ 컴파일러는 elementAt이 Object 타입을 반환할 것이라는 밖에는 알지 못하므로 프로그래머가 명시적으로 타입을 지정하여야 한다.

■ 규모가 큰 프로그램의 작성 시 프로그래머가 타입을 실수로 잘못 지정하는 경우에는 런타임에 ClassCastException이 발생

■ 제네릭을 이용하면 컴파일러가 요소의 타입을 알 수 있어 타입 불일치를 방지

8.1.21 제네릭의 기본 개념

-제네릭

■ 디자인 패턴의 매개 변수화된 타입과 메소드(parameterized type and method)를 의미

■ JDK 1.5부터 도입

■ 컬렉션은 다양한 타입의 객체들을 하나로 모아서 관리하므로 제네릭으로 정의되어 있음

8.1.22 제네릭 Vector를 설명하는 자바 API

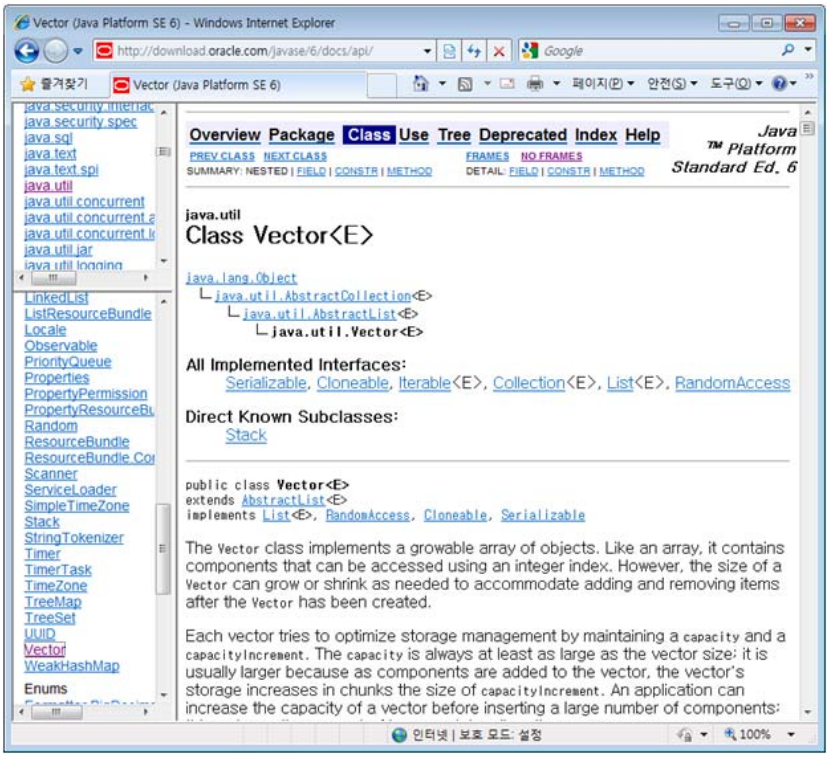


그림 8.11

8.1.23 제네릭 컬렉션 사용하기

-제네릭 Vector 클래스

■ 벡터는 일반 배열과는 달리 크기가 가변적인 객체의 배열을 구현

■ 배열의 원소는 객체만 허용

- 기본 데이터 타입은 원소가 될 수 없음

■ add, remove, indexOf, size, Iterator 등 메소드 이용하여 관리

-제네릭 Vector 사용하기

■ 제네릭 Vector의 생성

```
Vector<Integer> myVector = new Vector<Integer>();
```

■ 제네릭 벡터에 요소 삽입

```
myVector.add(new Integer(5));
myVector.add(new Integer(55));
myVector.add(new Integer(23));
```

■ 제네릭 벡터에서 요소 얻기

```
Integer n = myVector.get(0);
Integer m = myVector.get(1);
Integer k = myVector.get(2);
```

■ 생성할 때 지정한 타입이 아닌 타입은 삽입 불가

```
myVector.add("hello"); // 컴파일 오류
myVector.add(new Double(3.5)); // 컴파일 오류
myVector.add(new Person()); // 컴파일 오류
```

-예제 8-8 : 제네릭 Vector 사용하기

■ 다음 코드에 대한 결과는 무엇인가?

```
import java.util.Collections;
import java.util.Vector;

public class VectorExample {
   public static void main(String[] args) {
      Vector<Integer> v = new Vector<Integer>(3);
      System.out.println("벡터의 초기 크기는 " + v.capacity());
      v.add(new Integer(1));
      v.add(new Integer(22));
      v.add(new Integer(51));
      v.add(new Integer(10));
      System.out.println("벡터의 크기는 " + v.capacity());

      Collections.sort(v);
      for (int i=0;i<v.size();i++) { // 벡터에 있는 모든 요소에 대해 반복
         Integer n = v.elementAt(i); // 요소 객체 알아내기
         System.out.println(n.toString());
      }
   }
}
```

```
벡터의 초기 크기는 3
벡터의 크기는 6
1
10
22
51
```

-예제 8-9 : 제네릭 타입을 두 개 가진 제네릭 Hashtable 사용하기

■ 다음 코드에 대한 결과는 무엇인가?

```
import java.util.Enumeration;
import java.util.Hashtable;

public class HashtableEx {
   public static void main(String [] args) {
      Hashtable<String, String> h = new Hashtable<String, String>();
      h.put("21", "홍길동");
      h.put("54", "황기태");
      h.put("76", "이순룡");
      h.put("123", "애리슨포드");
      System.out.println("해쉬에 담긴 정보 수 : "+h.size());
      Enumeration<String> e = h.keys();
      while(e.hasMoreElements()) {
         String key = e.nextElement();
         String value = h.get(key);
         System.out.println(key + ":" + value);
      }
   }
}
```

```
Hashtable의 키 개수 : 4
76:이순룡
123:애리슨포드
21:홍길동
54:황기태
```

8.1.24 제네릭 클래스와 인터페이스 선언

-제네릭 클래스 작성 예

■ 일반화된 타입(generic type) 매개 변수 T를 가진 MyClass

```
public class MyClass<T> {
   T val;
   void set(T a) {
      val = a;
   }
   T get() {
      return val;
   }
}
```

■ 작성된 MyClass 타입 객체 생성 예

```
MyClass<String> s = new MyClass<String>();
s.set("hello");
System.out.println(s.get()); // "hello" 출력

MyClass<Integer> n = new MyClass<Integer>();
n.set(new Integer(5));
System.out.println(n.get()); // 숫자 5 출력
```

8.1.25 타입 매개 변수

-타입 매개 변수

■ '<'과 '>'사이의 문자로 표현

■ 하나의 대문자를 타입 매개 변수로 사용

■ 많이 사용하는 타입 매개 변수 문자

- E : Element를 의미하며 컬렉션에서 요소를 표시할 때 많이 사용한다.
- T : Type을 의미한다.
- V : Value를 의미한다.
- K : Key를 의미

■ 타입 매개변수가 나타내는 타입의 객체 생성 불가

- ex) T a = new T();

■ 타입 매개 변수는 나중에 실제 타입으로 대체 된다.

■ 어떤 문자도 매개 변수로 사용될 수 있다.

8.1.26 제네릭 클래스, 인터페이스 사용

-변수 선언

■ 제네릭 클래스 또는 인터페이스 타입 변수를 선언할 때는 타입 매개 변수에 실제 타입을 기입

```
List<Integer> li;
Vector<String> vs;
```

-객체 생성

■ 제네릭 클래스 객체를 생성할 때도 타입 매개 변수에 실제 타입을 기입

```
Vector<String> vs = new Vector<String>();
```

-예제 8-10 : 스택

■스택 자료 구조를 제네릭 클래스로 선언하고, String과 Integer형 스택을 사용하는 예를 보여라.

```
class GStack<T> {
   int tos; // 스택에 저장된 요소의 개수
   Object [] stck;
   public GStack() {
      tos = 0;
      stck = new Object [10];
   }
   public void push(T item) {
      if(tos == 10)
         return;
      stck[tos] = item;
      tos++;
   }
   public T pop() {
      if(tos == 0)
         return null;
      tos--;
      return (T)stck[tos];
   }
}
```

```
public class MyStack {
   public static void main(String[] args) {
      // String 타입의 GStack 생성
      GStack<String> g = new GStack<String>();
      g.push("seoul");
      g.push("busan");
      g.push("LA");
      System.out.println(g.pop());
      System.out.println(g.pop());
      System.out.println(g.pop());

      // Integer 타입의 GStack 생성
      GStack<Integer> i = new GStack<Integer>();
      i.push(new Integer(1));
      i.push(new Integer(3));
      i.push(new Integer(5));
      System.out.println(i.pop());
      System.out.println(i.pop());
      System.out.println(i.pop());
   }
}
```

```
LA
busan
seoul
5
3
1
```

8.1.27 제네릭과 배열

-제네릭에서 배열의 제한

-제네릭 클래스 또는 인터페이스의 배열을 허용하지 않음

```
GStack<Integer>[] gs = new GStack<Integer>[10];
```

-타입 매개 변수가 나타내는 타입의 배열을 생성하는 것도 허용되지 않음

```
T[] a = new T[10];
```

■앞 예제에서는 Object 타입으로 배열 생성 후 실제 사용할 때 타입 캐스팅

```
return (T)stck[tos]; // 타입 매개 변수 T타입으로 캐스팅
```

-타입 매개변수가 나타내는 타입의 배열 선언은 허용

```
public void myArray(T[] a) {....}
```

8.1.28 제네릭 메소드

-제네릭 메소드 정의

■메소드에서도 타입 매개 변수를 이용하여 제네릭 메소드 정의 가능

```
class GenericMethodEx {
   static <T> void toStack(T[] a, GStack<T> gs) {
      for (int i = 0; i < a.length; i++) {
         gs.push(a[i]);
      }
   }
}
```

■ 제네릭 메소드를 호출할 때는 컴파일러가 메소드의 인자를 통해 이미 타입을 알고 있으므로 타입을 명시하지 않아도 됨

```
String[] sa = new String[100];
GStack<String> gss = new GStack<String>();
GenericMethodEx.toStack(sa, gss);
```

• sa는 String[], gss는 GStack<String> 타입이므로 T를 String으로 유추

-예제 8-11 : 스택의 내용을 반대로 만드는 제네릭 메소드 만들기

■ 예제 8-10의 GStack을 이용하여 주어진 스택의 내용을 반대로 만드는 제네릭 메소드 reverse()를 작성하라.

```
public class GenericMethodExample {
  // T가 타입 매개 변수인 제네릭 메소드
  public static <T> GStack<T> reverse(GStack<T> a) {
    // T타입의 GStack 생성
    GStack<T> s = new GStack<T>();
    while (true) {
      T tmp;
      tmp = a.pop(); // 원래 스택에서 요소 하나를 꺼냄
      if (tmp==null) // 스택이 비었음
        break;
      else
        s.push(tmp); // 새 스택에 요소를 삽입
    }
    return s; // 새 스택을 반환
  }
```

```
  public static void main(String[] args) {
    // Double 타입의 GStack 생성
    GStack<Double> gs =
        new GStack<Double>();

    // 5개의 요소를 스택에 push
    for (int i=0; i<5; i++) {
      gs.push(new Double(i));
    }
    gs = reverse(gs);
    for (int i=0; i<5; i++) {
      System.out.println(gs.pop());
    }
  }
}
```

```
0.0
1.0
2.0
3.0
4.0
```

8.1.29 제네릭의 장점

-컬렉션과 같은 컨테이너 클래스에 유연성을 해치지 않으며 type-awareness를 첨가
-메소드에 type-awareness 첨가
-컴파일 시에 타입이 결정되어 보다 안전한 프로그래밍 가능
-개발 시 다운캐스팅(타입 캐스팅) 절차 불필요
-런타임 타입 충돌 문제 방지
-ClassCastException 방지

제9장 자바의 GUI, AWT와 Swing

9.1 자바의 GUI(Graphic User Interface)

1. GUI 목적

-GUI 목적
■ 그래픽 이용, 사용자에게 이해하기 쉬운 모양으로 정보 제공
■ 사용자는 마우스나 키보드를 이용하여 쉽게 입력
-자바 GUI 특징
■ 강력한 GUI 컴포넌트 제공
■ 쉬운 GUI 프로그래밍

-자바의 GUI 프로그래밍 방법

■ GUI 컴포넌트 이용

- AWT 패키지와 Swing 패키지
- AWT

  -java.awt 패키지
- Swing

  -javax.swing 패키지

2. 스윙 컴포넌트 예시

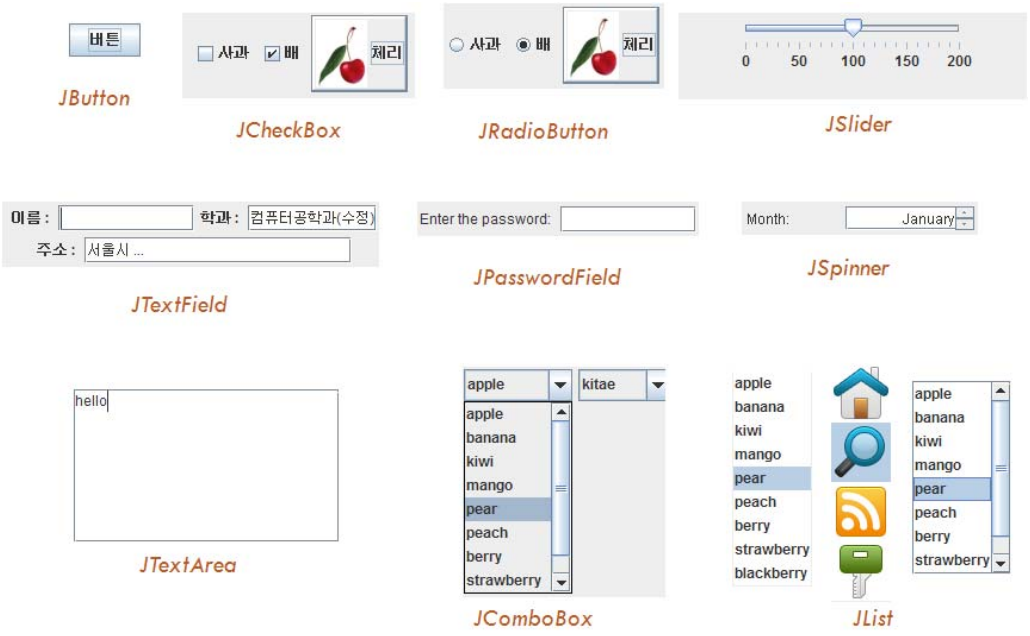


그림 9.1

그림 9.2

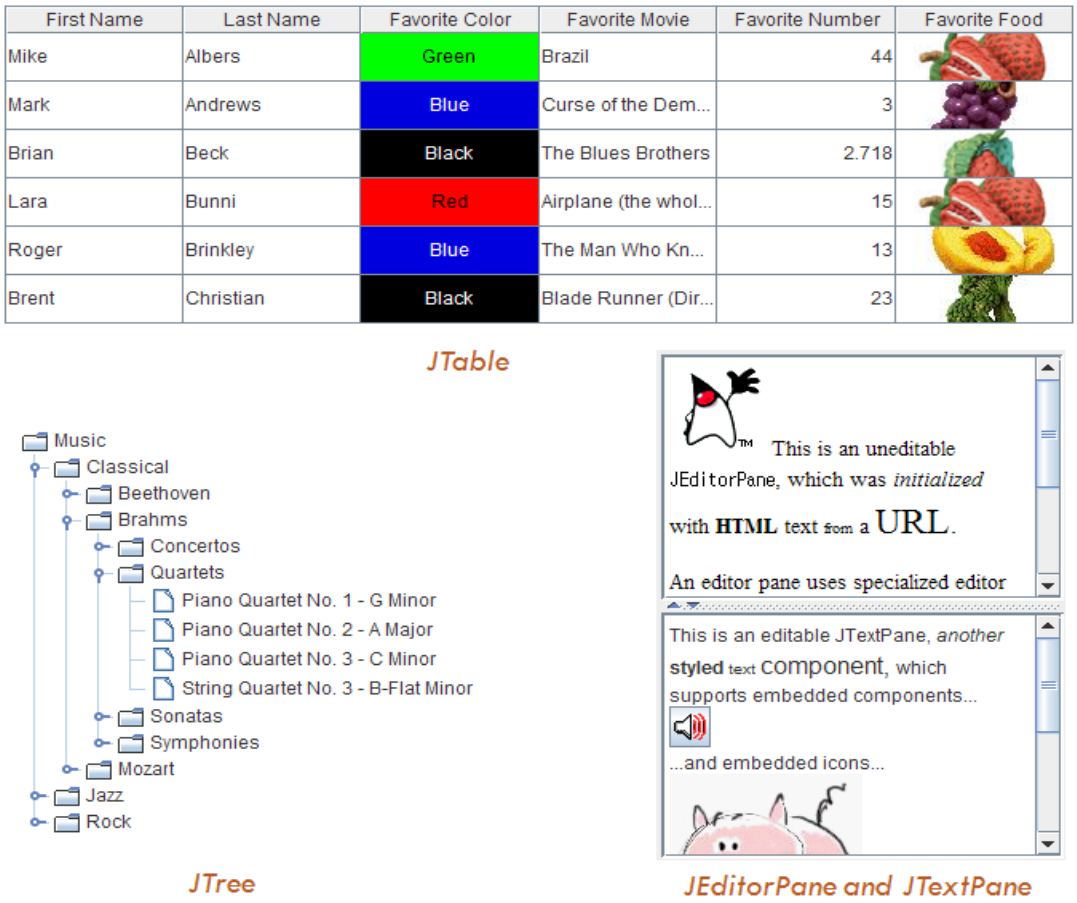

First Name
Last Name
Favorite Color
Favorite Movie
Favorite Number
Favorite Food
Mike
Albers
Green
Brazil
44
Mark
Andrews
Blue
Curse of the Dem...
3
Brian
Beck
Black
The Blues Brothers
2.718
Lara
Bunni
Red
Airplane (the whol...
15
Roger
Brinkley
Blue
The Man Who Kn...
13
Brent
Christian
Black
Blade Runner (Dir...
23
JTable
Music
Classical
Beethoven
Brahms
Concertos
Quartets
Piano Quartet No. 1 - G Minor
Piano Quartet No. 2 - A Major
Piano Quartet No. 3 - C Minor
String Quartet No. 3 - B-Flat Minor
Sonatas
Symphonies
Mozart
Jazz
Rock
JTree
This is an uneditable JEditorPane, which was initialized with HTML text from a URL.
An editor pane uses specialized editor
This is an editable JTextPane, another styled text component, which supports embedded components...
...and embedded icons...
JEditorPane and JTextPane

그림 9.3


New
search text field
Java
JToolBar
tab1
tab2
tab3
JTabbedPane
JSplitPane

그림 9.4

3. Swing 으로 만든 GUI 프로그램 샘플

■ 마우스를 올리면 컴포넌트 타입(예:JButton)을 표시

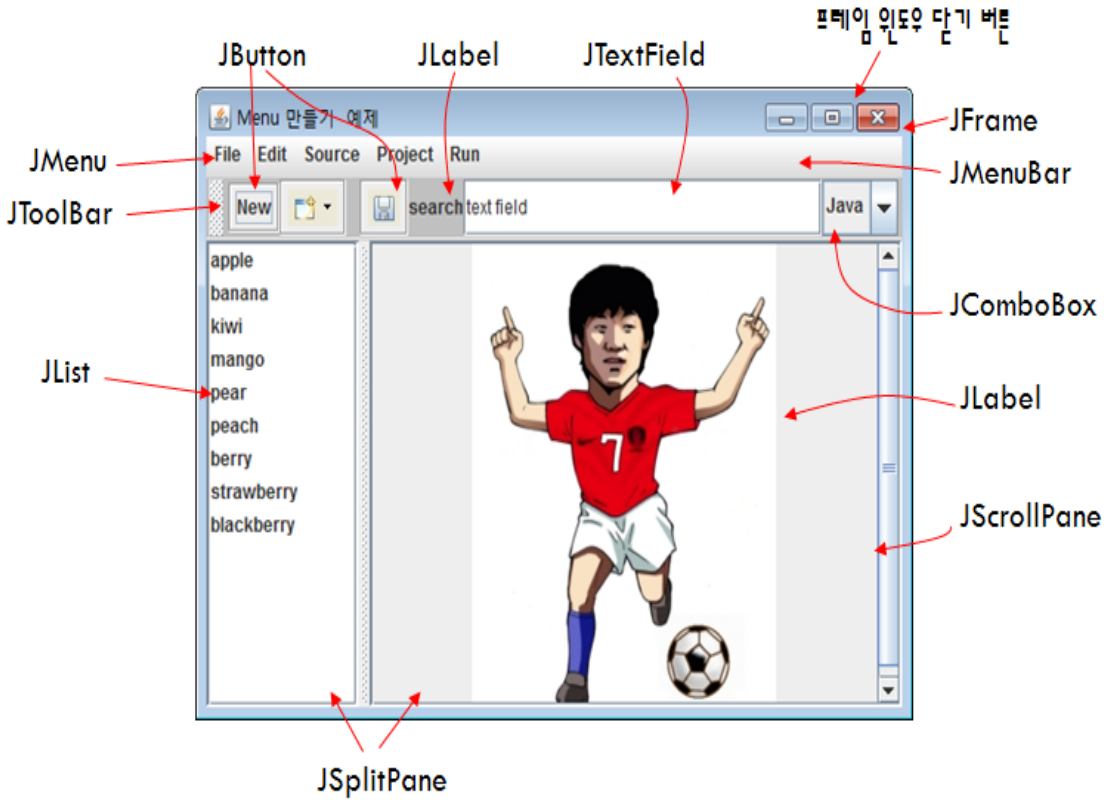


그림 9.5

4. AWT와 Swing 패키지

-AWT(Abstract Windowing Toolkit)

■ 자바가 처음 나왔을 때 함께 배포된 GUI 라이브러리

■ GUI 클래스 패키지 - java.awt 패키지

■ Native 운영체제와 응용프로그램 사이의 연결 라이브러리

- 중량 컴포넌트(Heavy weight components)

  -AWT 컴포넌트는 native(peer)에 의존적임

  -OS의 도움을 받아야 화면에 출력되며 동작하는 컴포넌트. 운영체제에 많은 부담. 오히려 처리 속도는 빠름

-Swing(스윙)

■ AWT 기술을 기반으로 작성된 자바 라이브러리

- 모든 AWT 기능 + 추가된 풍부하고 화려한 고급 컴포넌트
- AWT 컴포넌트에 J자가 덧붙여진 이름의 클래스
- J 자로 시작하는 클래스

■ 순수한 자바 언어로 구현, JDK 1.1 부터 - javax.swing 패키지

■ Swing 컴포넌트는 native(peer)에 의존하지 않음

- 경량 컴포넌트(Light weight components)

5. GUI 라이브러리 계층 구조

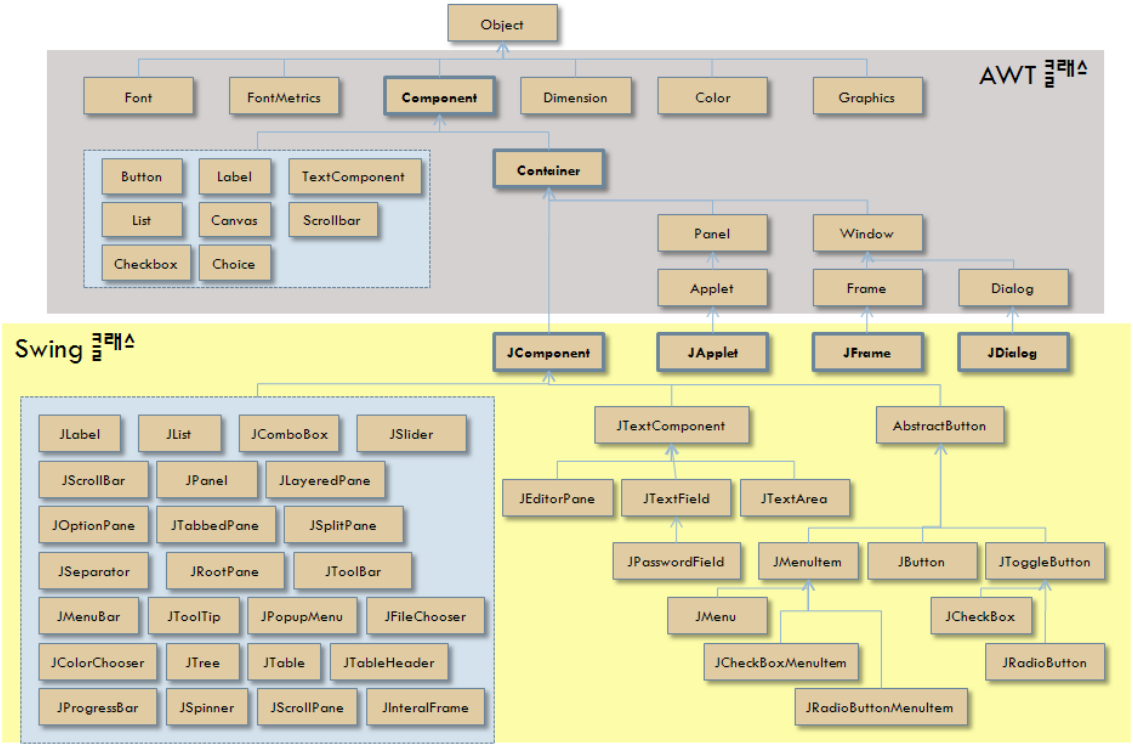


그림 9.6

6. Swing 클래스의 특징

-클래스명 J 자로 시작
-화려하고 다양한 컴포넌트로 쉽게 GUI 프로그래밍
-스윙 컴포넌트는 2 가지 유형
■ JComponent는 상속받는 클래스
■ AWT의 Container를 상속받는 클래스
  • JApplet, JDialog, JFrame 등
-JComponent
■ 매우 중요한 추상 클래스
  • 스윙 컴포넌트의 공통적인 속성 구현
  • new JComponent() 인스턴스를 생성할 수 없음
■ AWT의 Component를 상속받음

7. 컨테이너와 컴포넌트

-컨테이너
다른 컴포넌트를 포함할 수 있는 GUI 컴포넌트
  • java.awt.Container를 상속받아야 함
컨테이너 객체는 다른 컨테이너에 포함될 수 있는 컴포넌트
  • AWT 컨테이너
    -Panel, Frame, Applet, Dialog, Window
  • Swing 컨테이너
    -JPanel JFrame, JApplet, JDialog, JWindow

-최상위 컨테이너
다른 컨테이너에 속하지 않고도 독립적으로 존재하고 화면에 출력가능한 컨테이너
- JFrame, JDialog, JApplet

-컴포넌트
컨테이너에 포함되어야 비로소 화면에 출력될 수 있는 GUI 객체
모든 GUI 컴포넌트의 최상위 클래스
- java.awt.Component

스윙 컴포넌트의 최상위 클래스
- javax.swing.JComponent

8. 컨테이너와 컴포넌트의 포함관계

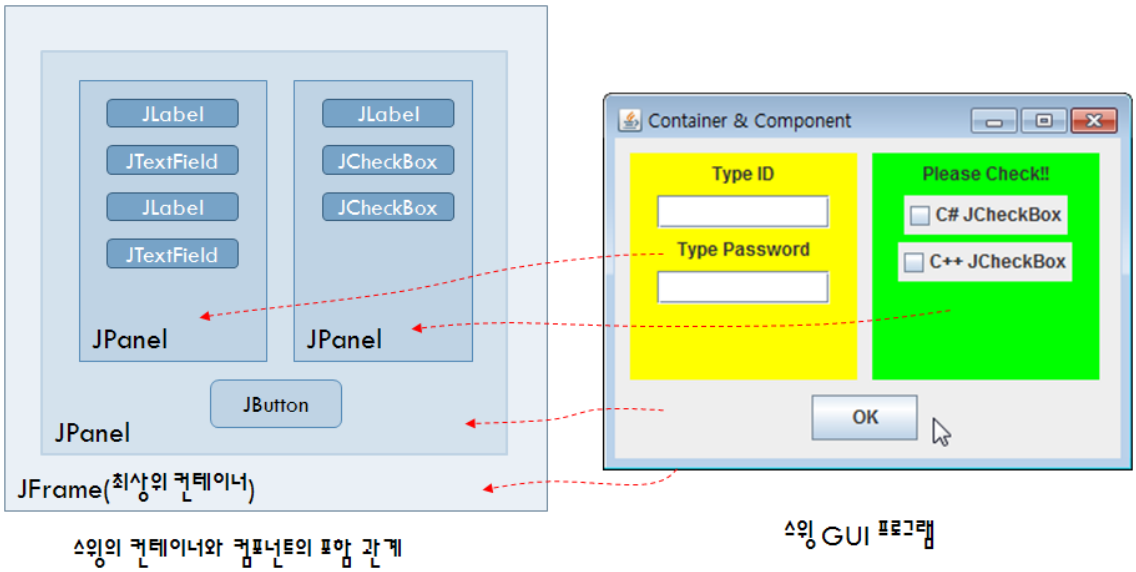


그림 9.7

9. 스윙 GUI 프로그램 만들기

1. 프레임 만들기
2. 프레임에 스윙 컴포넌트 붙이기
3. main() 함수 만들기

-스윙 프로그램을 작성하기 위한 import문
- import java.awt.*; // 그래픽 처리를 위한 클래스들의 경로명
- import java.awt.event.*; // AWT 이벤트 사용을 위한 경로명
- import javax.swing.*; // 스윙 컴포넌트 클래스들의 경로명
- import javax.swing.event.*; // 스윙 이벤트를 위한 경로명

10. 스윙 프레임

-모든 스윙 컴포넌트를 담는 최상위 GUI 컨테이너
- JFrame

-하나의 스윙 응용프로그램에 하나의 프레임 존재
-프레임이 닫히면 프레임 내의 모든 컴포넌트도 사멸

■ 화면에서 사라짐

-스윙 프레임(JFrame) 기본 구성

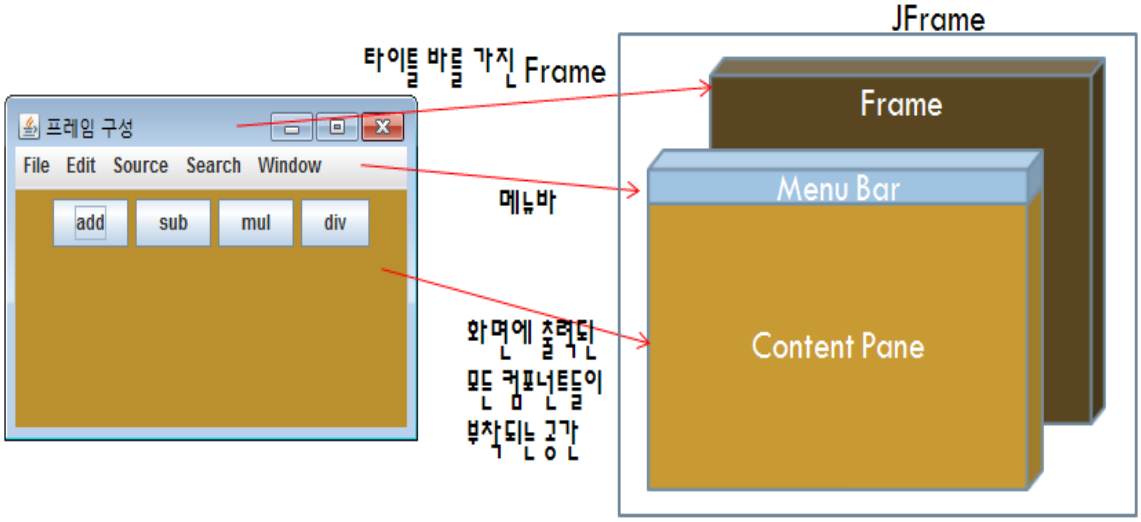
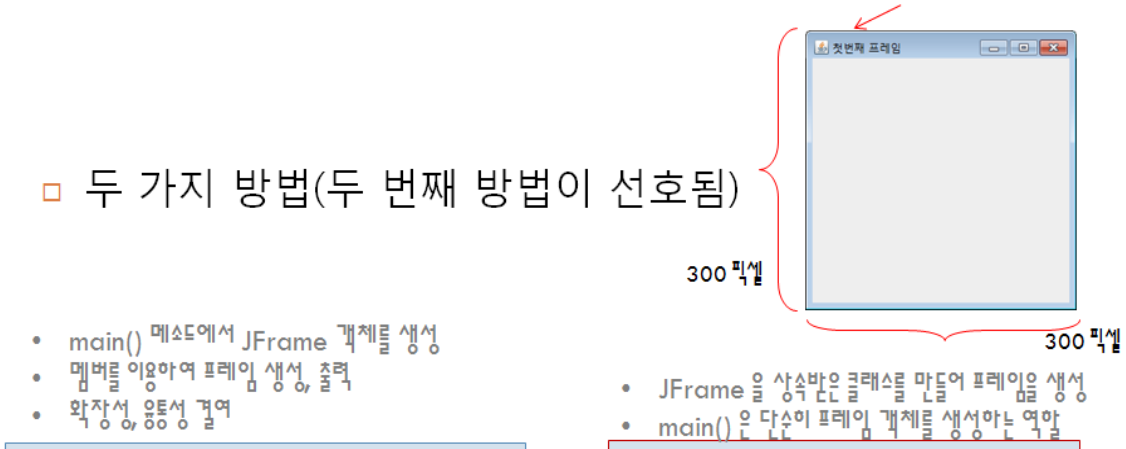


그림 9.8

11. 프레임 만들기

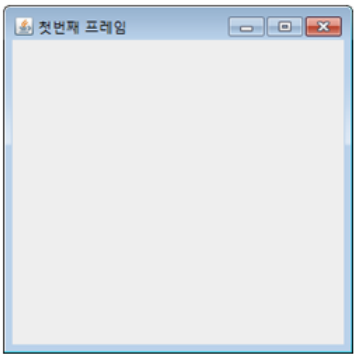



```
import javax.swing.*;

public class MyApp {
    public static void main(String [] args) {
        JFrame f = new JFrame();
        f.setTitle("첫번째 프레임");
        f.setSize(300,300);
        f.setVisible(true);
    }
}
```

```
import javax.swing.*;

public class MyFrame extends JFrame {
    MyFrame() {
        setTitle("첫번째 프레임");
        setSize(300,300);
        setVisible(true);
    }

    public static void main(String [] args) {
        MyFrame mf = new MyFrame();
    }
}
```

그림 9.9

12. main()의 위치




```
import javax.swing.*;

public class MyApp {
    public static void main(String [] args) {
        JFrame f = new JFrame();
        f. setTitle("첫번째 프레임");
        f.setSize(300,300);
        f.setVisible(true);
    }
}
```

```
import javax.swing.*;

public class MyFrame extends JFrame {
    MyFrame() {
        setTitle("첫번째 프레임");
        setSize(300,300);
        setVisible(true);
    }

    public static void main(String [] args) {
        MyFrame mf=new MyFrame();
    }
}
```

```
import javax.swing.*;

class MyFrame extends JFrame {
    MyFrame() {
        setTitle("첫번째 프레임");
        setSize(300,300);
        setVisible(true);
    }
}
public class MyApp {
    public static void main(String [] args) {
        MyFrame mf = new MyFrame();
    }
}
```

그림 9.10

13. 프레임에 컴포넌트 붙이기

-타이틀 - 프레임 영역의 타이틀 바에 부착

```
JFrame frame = new JFrame("타이틀문자열"); // JFrame의 생성자에 타이틀 달기
frame.setTitle("타이틀문자열"); // JFrame의 setTitle() 메소드를 이용하는 방법
```

-메뉴 - 메뉴바 공간, 14장에서 다룸

-스윙 컴포넌트 - 컨텐트팬(Content Pane)에 부착

■ 컨텐트팬 알아내기

```
JFrame frame = new JFrame();
Container contentPane = frame.getContentPane(); // 프레임의 컨텐트팬을 알아낸다.
```

■ 컨텐트팬에 컴포넌트 달기

```
// JDK 1.5 버전 아래
JFrame frame = new JFrame();
JButton b = new JButton("Click");
Container c = frame.getContentPane();
c.add(b);
```

```
// JDK 1.5 버전부터는 다음과 같이도 가능함
JFrame frame = new JFrame();
JButton b = new JButton("Click");
frame.add(b);
```

■ 컨텐트팬 변경

```
JPanel p = new JPanel();
frame.setContentPane(p);
```

-예제 9-1 : 컴포넌트를 부착한 프레임 예

```
import javax.swing.*;
import java.awt.*;

public class ContentPaneEx extends JFrame {
   ContentPaneEx() {
      setTitle("ContentPane과 JFrame");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      Container contentPane = getContentPane();
      contentPane.setBackground(Color.ORANGE);
       contentPane.setLayout(new FlowLayout());
       contentPane.add(new JButton("OK"));
       contentPane.add(new JButton("Cancel"));
       contentPane.add(new JButton("Ignore"));

      setSize(350, 150);
      setVisible(true);
   }

   public static void main(String[] args) {
      new ContentPaneEx();
   }
}
```

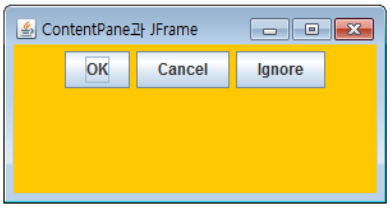


그림 9.11

14. 스윙 응용프로그램의 종료

-프로그램에서 무조건 종료

```
System.exit(0);
```

■ 언제 어디서나 프로그램 종료

-프레임 종료버튼(X) 클릭시 발생하는 현상

- 프레임을 종료하여 프레임 윈도우가 닫히게 됨. 화면에서 보이지 않게 됨
- 프로그램이 종료되지는 않음

-프레임 종료버튼 클릭시 프로그램이 종료 되게 하는 방법

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

■ 프레임이 닫히면 자동으로 자바 프로그램도 함께 종료된다.

15. main() 메소드 종료 뒤에도 프레임이 살아 있는 이유?

-main() 메소드를 실행하는 스레드

■ 메인스레드(main 스레드)

-스윙 프로그램의 생명 사이클

■ 메인 스레드외에 다른 스레드가 생성되어 살아 있으면 메인 스레드가 종료되어도 자바 응용프로그램은 종료되지 않음

-스윙 응용프로그램이 실행되면 자동으로 이벤트 처리 스레드가 보이지 않게 실행됨

■ 메인 스레드가 종료해도 스윙 응용프로그램이 종료되지 않음

16. 컨테이너와 배치 개념

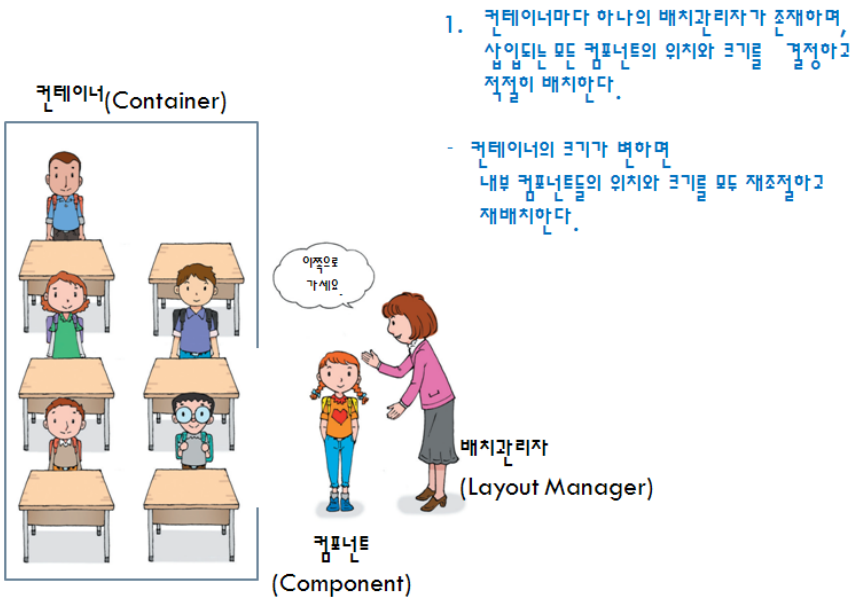


그림 9.12

17. 배치 관리자 대표 유형 4 가지

-java.awt 패키지에 구현되어 있음

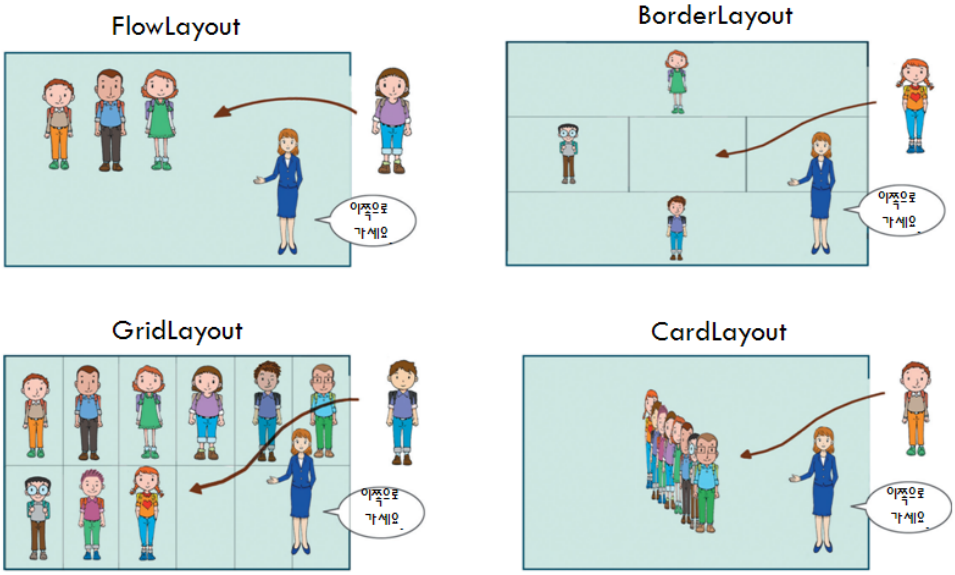


그림 9.13

18. 컨테이너와 배치관리자

-컨테이너의 디폴트 배치관리자

■ 컨테이너는 생성시 디폴트 배치관리자 설정

표 9.1

AWT와 스윙의 컨테이너	디폴트 배치관리자
Window	BorderLayout
Frame, JFrame	BorderLayout
Dialog, JDialog	BorderLayout
Panel, JPanel	FlowLayout
Applet, JApplet	FlowLayout

-컨테이너에 새로운 배치관리자 설정

■ Container.setLayout(LayoutManager lm)

• lm을 Container의 새로운 배치관리자로 설정

```
// JPanel 패널에 BorderLayout 배치관리자를 설정하는 예

JPanel p = new JPanel();
p.setLayout(new BorderLayout());
```

```
JFrame frame = new JFrame();
Container c = frame.getConentPane(); // 프레임의 컨텐트팬
c.setLayout(new FlowLayout()); // 컨텐트팬에 FlowLayout 설정
함
frame.setLayout(new FlowLayout()); // JDK 1.5 이후 버전에서
```

19. FlowLayout

-배치방법

■ 컨테이너 공간 내에 왼쪽에서 오른쪽으로 배치

• 다시 위에서 아래로 순서대로 컴포넌트를 배치한다.

```
container.setLayout(new FlowLayout());
container.add(new JButton("add"));
container.add(new JButton("sub"));
container.add(new JButton("mul"));
container.add(new JButton("div"));
container.add(new JButton("Calculate"));
```

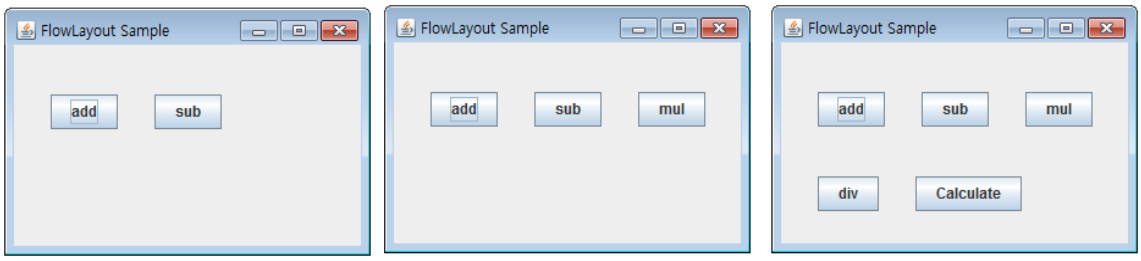


그림 9.14

-컨테이너의 크기가 변하면 재배치

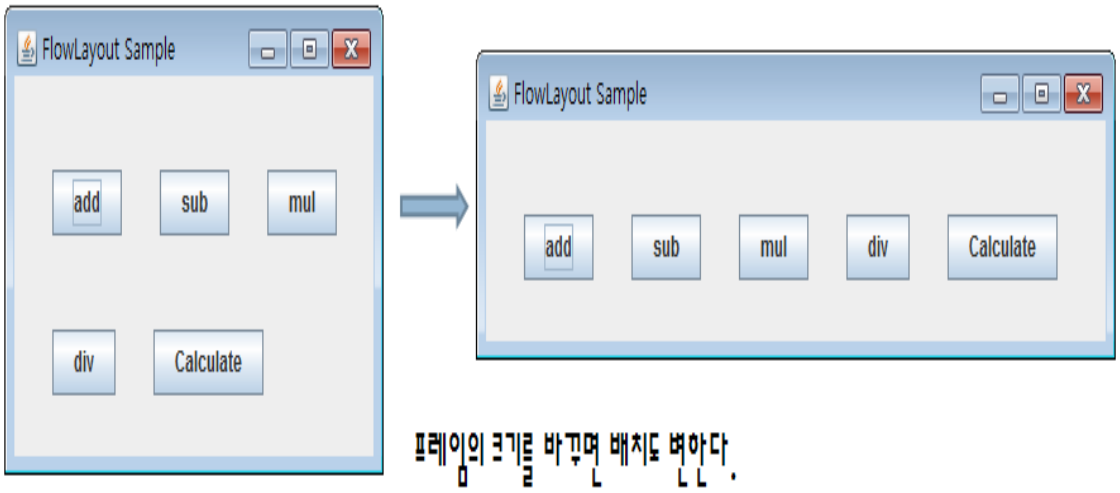


그림 9.15

20. FlowLayout - 생성자와 속성

-생성자

- FlowLayout()
- FlowLayout(int align)
- FlowLayout(int align, int hGap, int vGap)
  - align : 컴포넌트의 정렬(5 가지중 많이 사용되는 3 가지)
    -FlowLayout.LEFT, FlowLayout.RIGHT, FlowLayout.CENTER(디폴트)
  - hGap : 좌우 두 컴포넌트 사이의 수평 간격, 픽셀 단위(디폴트 : 5)
  - vGap : 상하 두 컴포넌트 사이의 수직 간격, 픽셀 단위(디폴트 : 5)

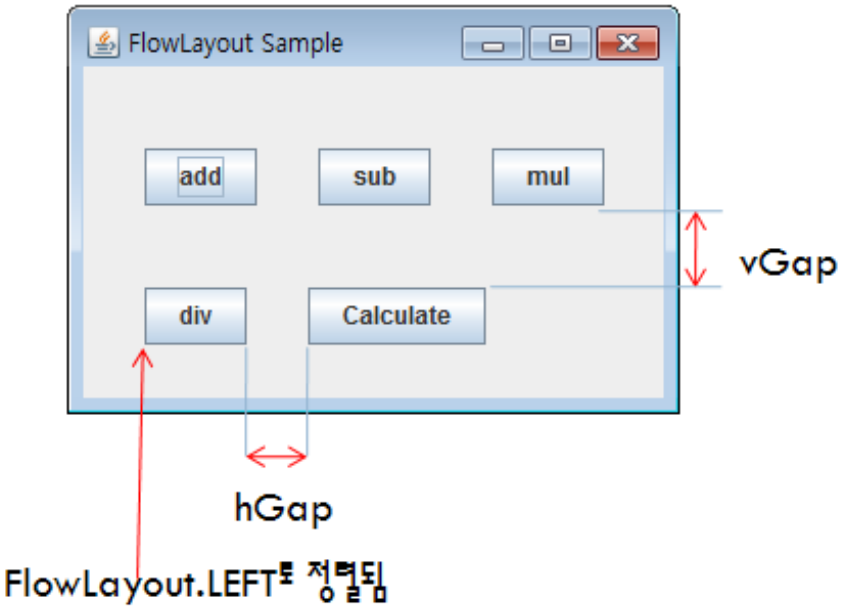


그림 9.16

-예제 9-2 : LEFT로 정렬되는 수평 간격이 30 픽셀, 수직 간격이 40 픽셀인 FlowLayout 사용 예

```
import javax.swing.*;
import java.awt.*;

public class FlowLayoutEx extends JFrame {
	FlowLayoutEx() {
		setTitle("FlowLayout Sample");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		setLayout(new FlowLayout(FlowLayout.LEFT, 30, 40));
		add(new JButton("add"));
		add(new JButton("sub"));
		add(new JButton("mul"));
		add(new JButton("div"));
		add(new JButton("Calculate"));

		setSize(300, 250);
		setVisible(true);
	}
	public static void main(String[] args) {
		new FlowLayoutEx();
	}
}
```

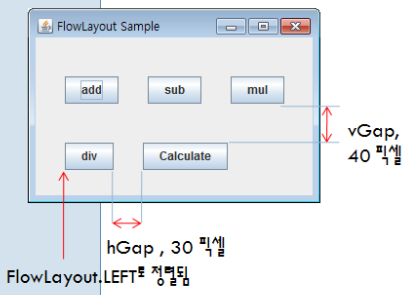


그림 9.17

21. BorderLayout

-배치방법

■ 컨테이너 공간을 5 구역으로 분할, 배치

• East, West, South, North, Center

■ 배치 방법

• add(Component comp, int index)

-comp를 index의 공간에 배치

-index

• BorderLayout.EAST, BorderLayout.WEST, BorderLayout.SOUTH, BorderLayout.NORTH, BorderLayout.CENTER,

■ 컨테이너의 크기가 변하면 재배치

```
container.setLayout(new BorderLayout());
container.add(new JButton("div"), BorderLayout.WEST);
container.add(new JButton("Calculate"), BorderLayout.CENTER);
```

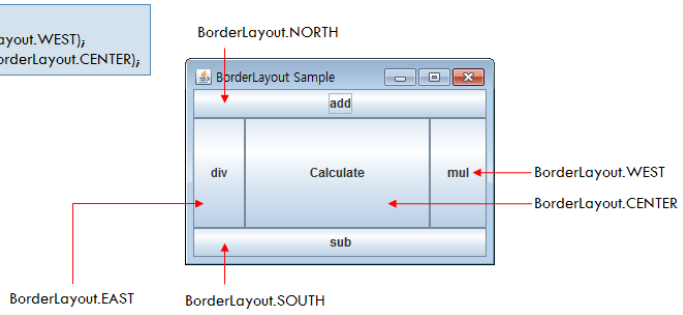


그림 9.18

22. BorderLayout 생성자와 속성

-생성자

■ BorderLayout()

■ BorderLayout(int hGap, int vGap)

• hGap : 좌우 두 컴포넌트 사이의 수평 간격, 픽셀 단위(디폴트 : 0)

• vGap : 상하 두 컴포넌트 사이의 수직 간격, 픽셀 단위(디폴트 : 0)

-BorderLayout의 사용예

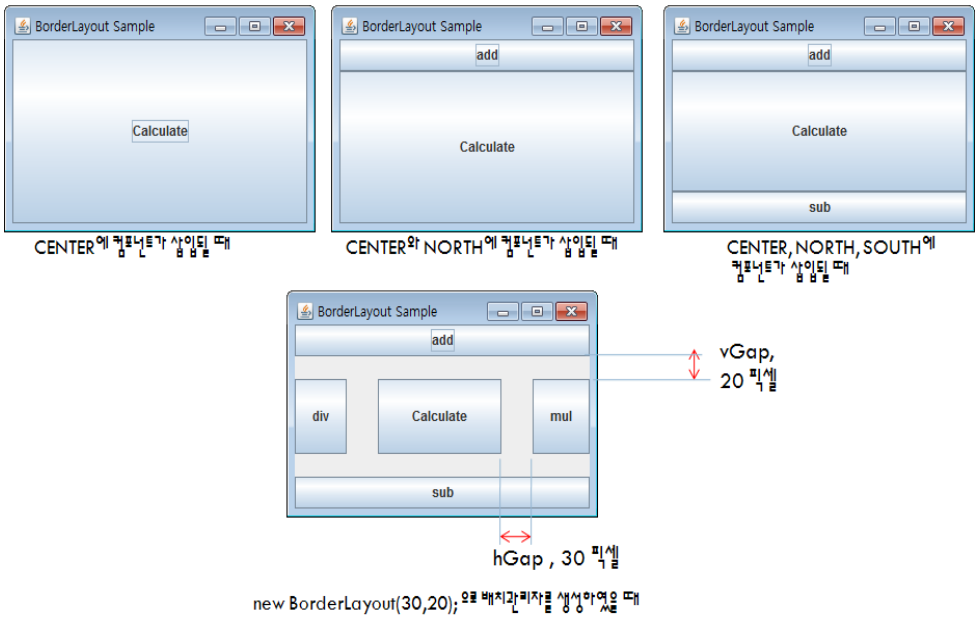


그림 9.19

-예제 9-3 : BorderLayout 사용 예

```
import javax.swing.*;
import java.awt.*;

public class BorderLayoutEx extends JFrame {
	BorderLayoutEx() {
		setTitle("BorderLayout Sample");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		setLayout(new BorderLayout());
		add(new JButton("add"), BorderLayout.NORTH);
		add(new JButton("sub"), BorderLayout.SOUTH);
		add(new JButton("mul"), BorderLayout.EAST);
		add(new JButton("div"), BorderLayout.WEST);
		add(new JButton("Calculate"), BorderLayout.CENTER);

		setSize(300, 200);
		setVisible(true);
	}
	public static void main(String[] args) {
		new BorderLayoutEx();
	}
}
```

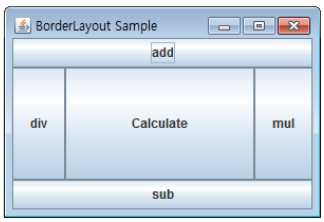
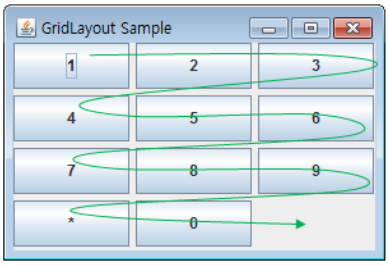


그림 9.20

23. GridLayout

-배치방법

■ 컨테이너 공간을 동일한 사각형 격자(그리드)로 분할하고 각 셀에 하나의 컴포넌트 배치
- 격자 구성은 생성자에서 행수와 열수로 지정
- 셀에 왼쪽에서 오른쪽으로, 다시 위에서 아래로 순서대로 배치




```
container.setLayout(new GridLayout(4,3,5,5)); // 4×3 분할로 컴포넌트 배치
container.add(new JButton("1")); // 상단 왼쪽 첫 번째 셀에 버튼 배치
container.add(new JButton("2")); // 그 옆 셀에 버튼 배치
```

- 4x3 그리드 레이아웃 설정
- 총 11 개의 버튼이 순서대로 add 됨
- 수직 간격 vGap : 5 픽셀
- 수평 간격 hGap : 5 픽셀

그림 9.21

■ 컨테이너의 크기가 변하면 재배치
- 크기 재조정

24. GridLayout 생성자와 속성

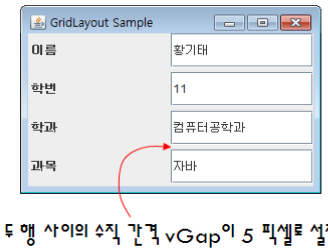
-생성자

■ GridLayout()

■ GridLayout(int rows, int cols)

■ GridLayout(int rows, int cols, int hGap, int vGap)
- rows : 격자의 행수 (디폴트 : 1)
- cols : 격자의 열수 (디폴트 : 1)
- hGap : 좌우 두 컴포넌트 사이의 수평 간격, 픽셀 단위(디폴트 : 0)
- vGap : 상하 두 컴포넌트 사이의 수직 간격, 픽셀 단위(디폴트 : 0)
- rows x cols 만큼의 셀을 가진 격자로 컨테이너 공간을 분할, 배치

-예제 9-4 : GridLayout으로 입력 폼 만들기




```
import javax.swing.*;
import java.awt.*;

public class GridLayoutEx extends JFrame {
	GridLayoutEx() {
		setTitle("GridLayout Sample");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		GridLayout grid = new GridLayout(4, 2);
		grid.setVgap(5);
		setLayout(grid);
		add(new JLabel("이름"));
		add(new JTextField(""));
		add(new JLabel("학번"));
		add(new JTextField(""));
		add(new JLabel("학과"));
		add(new JTextField(""));
		add(new JLabel("과목"));
		add(new JTextField(""));

		setSize(300, 200);
		setVisible(true);
	}
	public static void main(String[] args) {
		new GridLayoutEx();
	}
}
```

그림 9.22

25. 배치관리자 없는 컨테이너

-배치관리자가 없는 컨테이너 개념

■ 응용프로그램에서 컴포넌트의 절대 크기와 절대 위치를 스스로 결정

-용도

■ 컴포넌트의 크기나 위치를 개발자 임의로 결정하고자 하는 경우

■ 게임 프로그램과 같이 시간이나 마우스/키보드의 입력에 따라 컴포넌트들의 위치와 크기가 수시로 변하는 경우

■ 여러 컴포넌트들이 서로 겹치는 효과를 연출하고자 하는 경우

-컨테이너의 배치 관리자 제거 방법

■ Container.setLayout(null);

```
// JPanel 에 배치관리자를 삭제하는 예

JPanel p = new JPanel();
p.setLayout(null);
```

-컴포넌트의 크기와 위치 설정

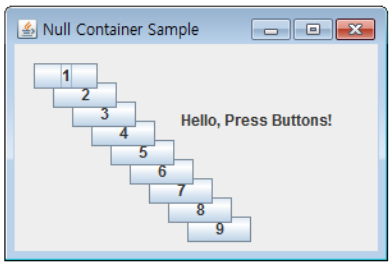
■ 프로그램 내에서 이루어져야 함

■ 컴포넌트들이 서로 겹치는 효과 연출 가능

■ 다음 메소드 이용

- 컴포넌트 크기 설정 : Component.setSize(int width, int height);
- 컴포넌트 위치 설정 : Component.setLocation(int x, int y);
- 컴포넌트 위치와 크기 동시 설정 : Component.setBounds(int x, int y, int width, int height);

-예제 9-5 : 배치관리자 없는 컨테이너에 컴포넌트 위치와 크기를 절대적으로 지정




```
import javax.swing.*;
import java.awt.*;

public class NullContainerEx extends JFrame {
	NullContainerEx() {
		setTitle("Null Container Sample");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLayout(null);

		JLabel la = new JLabel("Hello, Press Buttons!");
		la.setLocation(130, 50);
		la.setSize(200, 20);
		add(la);
		for(int i=1; i<=9; i++) {
			JButton b = new JButton(Integer.toString(i));
			b.setLocation(i*15, i*15);
			b.setSize(50, 20);
			add(b);
		}
		setSize(300, 200);
		setVisible(true);
	}
	public static void main(String[] args) {
		new NullContainerEx();
	}
}
```

그림 9.23

제10장 이벤트 처리

10.1 이벤트 기반 프로그래밍

1.이벤트 기반 프로그래밍

-이벤트 기반 프로그래밍(Event Driven Programming)

■ 이벤트의 발생에 의해 프로그램 흐름이 결정되는 방식
  • 이벤트가 발생하면 이벤트를 처리하는 루틴(이벤트 리스너)이 실행하는 방식
  • 프로그램 내의 어떤 코드가 언제 실행될 지 아무도 모름, 이벤트의 발생에 의해 전적으로 결정

■ 반대되는 개념 : 배치 실행(batch programming)
  • 프로그램의 개발자가 프로그램의 흐름을 결정하는 방식

-이벤트

■ 사용자의 입력 : 마우스 드래그, 마우스 클릭, 키보드 누름

■ 센서로부터의 입력

■ 네트워크로부터 데이타 송수신

■ 다른 응용프로그램이나 다른 스레드로부터의 메시지

-이벤트 기반 프로그램의 구조

■ 프로그램에서 처리하고자 하는 이벤트의 이벤트 처리 리스너 구현

-이벤트 처리 순서

■ 이벤트 발생(예 :마우스나 키보드의 움직임 혹은 입력)

■ 이벤트 객체 생성
  • 현재 발생한 이벤트에 대한 여러 정보를 가진 객체

■ 이벤트 리스너 찾기

■ 이벤트 리스너 호출
  • event 객체가 리스너에 전달됨

■ 이벤트 리스너 실행

-이벤트의 실제 예

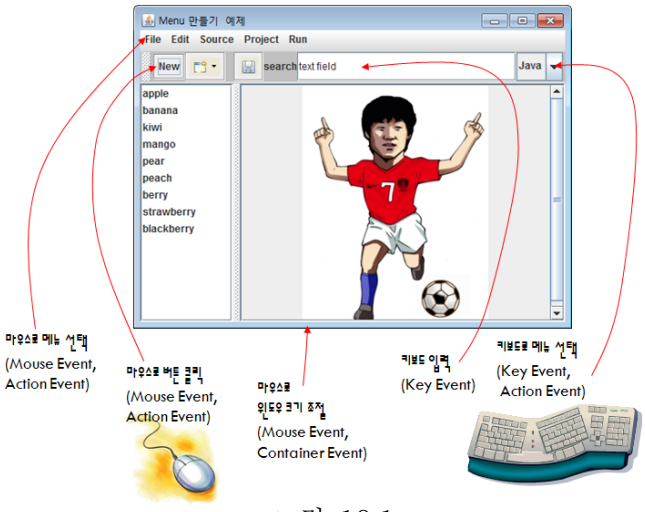


그림 10.1

10.1.2 자바의 이벤트 기반 GUI 응용프로그램 구성

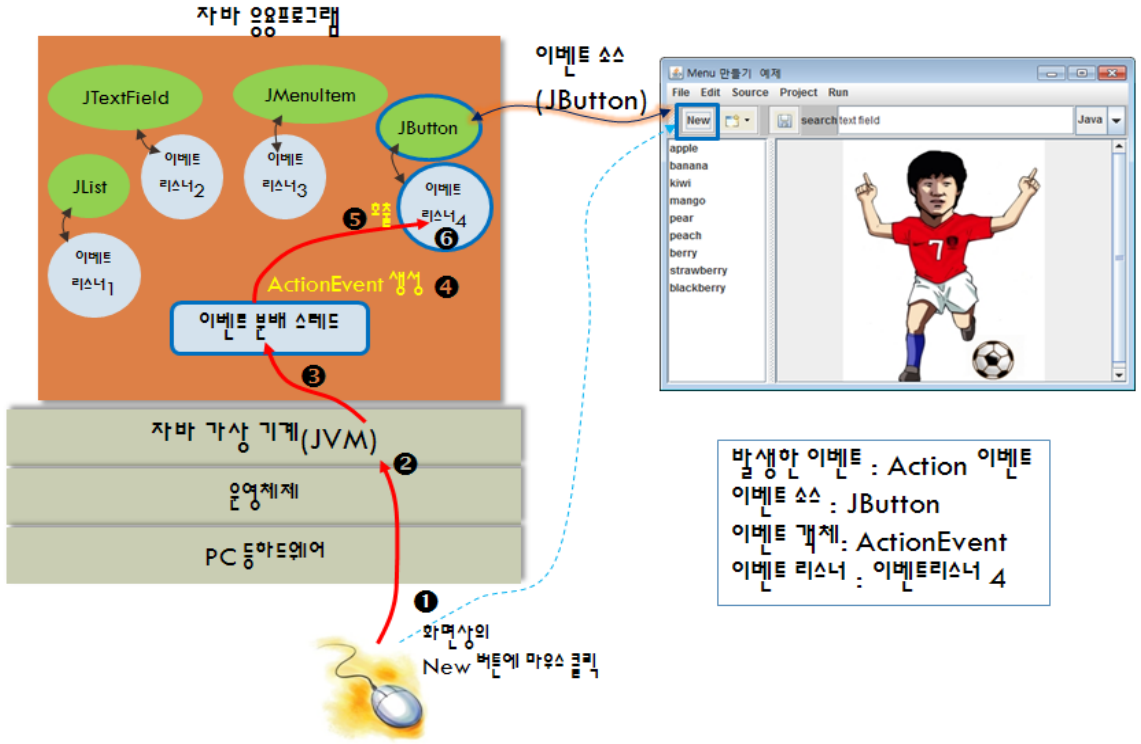


그림 10.2

10.1.3 이벤트 객체

-이벤트 객체란?

■ 이벤트가 발생하면 발생한 이벤트에 관한 정보를 가진 객체

■ 이벤트 리스너에 전달됨

• 이벤트 리스에서 이벤트가 발생한 여러 상황을 파악할 수 있게 함

-이벤트 객체의 종류

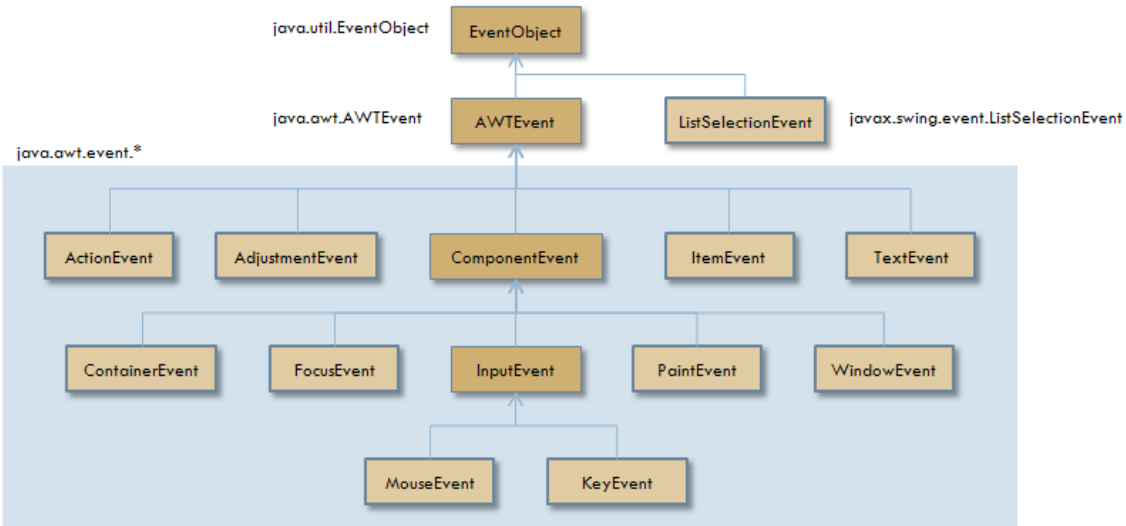


그림 10.3

10.1.4 이벤트 객체에 포함된 정보

-이벤트 객체가 포함하는 정보

■ 이벤트 종류
■ 이벤트 소스
■ 이벤트가 발생한 화면 좌표
■ 이벤트가 발생한 컴포넌트 내 좌표
■ 버튼이나 메뉴 아이템에 이벤트가 발생한 경우 버튼이나 메뉴 아이템의 문자열
■ 클릭된 마우스 버튼 번호
■ 마우스의 클릭 횟수
■ 키가 눌러졌다면 키의 코드 값과 문자 값
■ 체크박스, 라디오버튼 등과 같은 컴포넌트에 이벤트가 발생하였다면 체크 상태

-이벤트에 따라 조금씩 다른 정보 포함

- ActionEvent 객체 : 액션 문자열
- MouseEvent 객체 : 마우스의 위치 정보, 마우스 버튼, 함께 눌러진 키 정보 등
- ItemEvent 객체 : 아이템의 체크 상태

-이벤트 소스 알아 내기

■ object EventObject.getSource()

- 발생한 이벤트의 소스 컴포넌트를 리턴한다.
- Object 타입으로 리턴하므로 캐스팅하여 사용하는 것을 추천한다.
- 모든 이벤트 객체에서 제공됨

10.1.5 이벤트 객체의 메소드

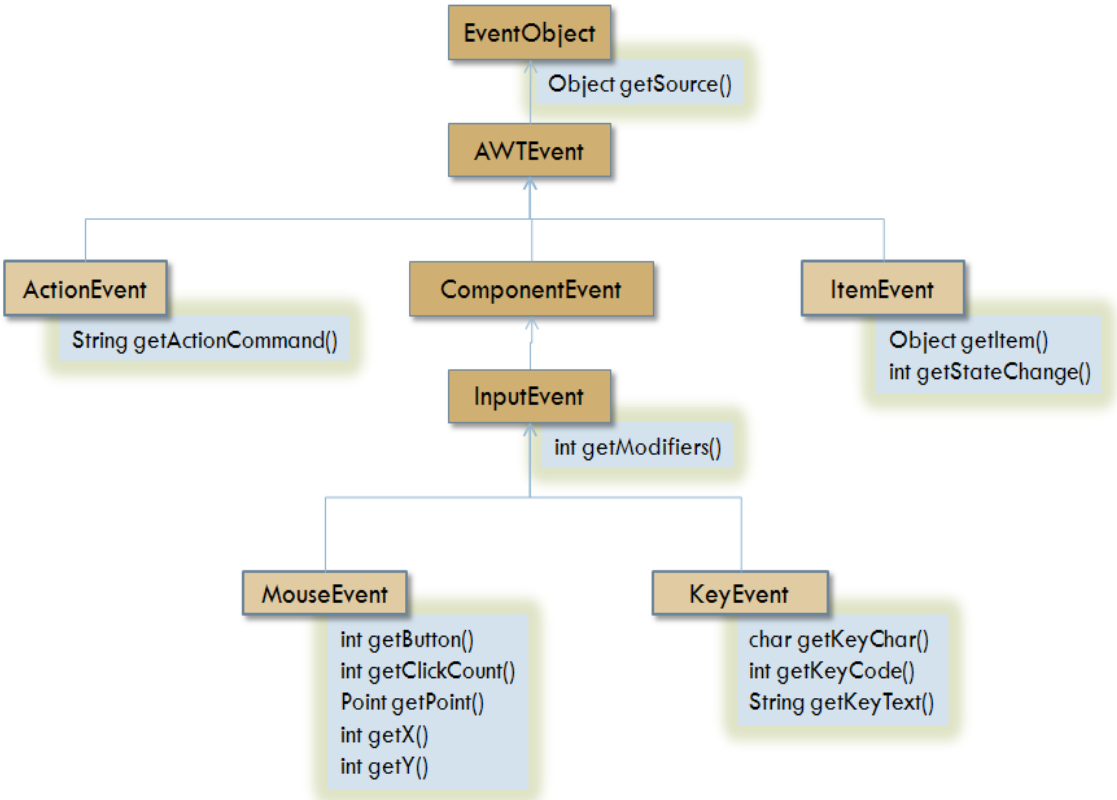


그림 10.4

10.1.6 이벤트 객체와 이벤트 소스

표 10.1

이벤트 객체	이벤트 소스	이벤트가 발생하는 경우
ActionEvent	JButton	마우스로 버튼을 클릭하거나 키로 버튼을 선택한 경우
	JList	리스트 아이템을 더블클릭하여 리스트 아이템을 선택한 경우
	JMenuItem	메뉴 아이템 선택을 선택한 경우
	JTextField	텍스트 입력 중 <Enter> 키를 누른 경우
ItemEvent	JCheckBox	체크박스의 선택 혹은 해제
	JCheckBoxMenuItem	체크박스 메뉴 아이템이 선택 혹은해제 될 때
	JList	리스트 아이템이 선택될 때
KeyEvent	Component	모든 컴포넌트에 대해, 키가 눌러지거나 눌러진 키가 떼어질 때
MouseEvent	Component	모든 컴포넌트에 대해, 마우스 버튼이 눌러지거나 떼어질 때, 클릭될 때, 컴포넌트 위에 마우스가 올라갈 때, 올라간 마우스가 내려올 때, 마우스가 드래그될 때, 마우스가 단순 움직일 때
FocusEvent	Component	모든 컴포넌트에 대해, 컴포넌트가 포커스를 받거나 잃을 때
TextEvent	TextField	텍스트가 변경될 때
	TextArea	텍스트가 변경될 때
WindowEvent	Window	Window를 상속받는 모든 컴포넌트에 대해, 윈도우가 활성화,비활성화, 아이콘화, 아이콘에서 복구 될 때, 윈도우가 열리거나 닫힐 때, 윈도우가 종료될 때 등
AdjustmentEvent	JScrollBar	스크롤바를 사용자가 움직였을 때
ComponentEvent	Component	모든 컴포넌트에 대해, 컴포넌트가 사라지거나, 나타나거나, 이동하거나 크기 변경 될 때
ContainerEvent	Container	Container에 컴포넌트가 추가 혹은 삭제되었을 때

10.1.7 이벤트 리스너(Event Listener)

-이벤트 리스너란? 이벤트를 처리하는 핸들러

-이벤트 리스너는 인터페이스(interface)이다.

■ 개발자가 리스너 인터페이스의 모든 추상 메소드 구현 필요

■ 이벤트가 발생하면 이미 약속된 메소드 호출

■ 예) ActionListener 인터페이스

```
interface ActionListener { // 아래 메소드를 개발자가 구현해야 함
   public void actionPerformed(ActionEvent e); // Action 이벤트 발생시 호출
}
```

■ 예) MouseListener 인터페이스

```
interface MouseListener { // 아래의 5개 메소드를 개발자가 구현해야 함
   public void mousePressed(MouseEvent e); // 마우스 버튼이 눌러지는 순간 호출
   public void mouseReleased(MouseEvent e); // 눌러진 마우스 버튼이 떼어지는 순간 호출
   public void mouseClicked(MouseEvent e); // 마우스가 클릭되는 순간 호출
   public void mouseEntered(MouseEvent e); // 마우스가 컴포넌트 위에 올라가는 순간 호출
   public void mouseExited(MouseEvent e); // 마우스가 컴포넌트 위에서 내려오는 순간 호출
}
```

10.1.8 이벤트 리스너 등록

-이벤트 리스너 등록

■ 이벤트를 받아 처리하고자 하는 컴포넌트에 이벤트 리스너 등록

-이벤트 리스너 등록시 사용되는 메소드

■ Component.addXXXListener(listener)

- xxx : 이벤트 명
- listener : 이벤트 리스너 객체
- 예) addMouseListener(), addActionListener(), addFocusListener() 등

■ 이벤트 리스너가 등록된 컴포넌트에만 이벤트 처리 가능

10.1.9 리스너 인터페이스와 메소드

표 10.2

리스너 인터페이스	리스너 메소드	메소드가 호출되는 경우
ActionListener	actionPerformed(ActionEvent)	ActionEvent가 발생하는 경우
ItemListener	itemStateChanged(ItemEvent)	ItemEvent 가 발생하는 경우
KeyListener	keyPressed(KeyEvent)	키가 눌러질 때
	keyReleased(KeyEvent)	눌러진 키가 떼어질 때
	keyTyped(KeyEvent)	키가 입력될 때
MouseListener	mousePressed(MouseEvent)	마우스 버튼이 눌러질 때
	mouseReleased(MouseEvent)	눌러진 마우스 버튼이 떼어질 때
	mouseClicked(MouseEvent)	마우스 클릭될 때
	mouseEntered(MouseEvent)	마우스가 임의의 컴포넌트 위에 올라올 때
	mouseExited(MouseEvent)	컴포넌트에 올라온 마우스가 컴포넌트를 벗어날 때
MouseMotionListener	mouseDragged(MouseEvent)	마우스를 임의의 컴포넌트 위에서 드래깅할 때

	mouseMoved(MouseEvent)	마우스를 임의의 컴포넌트 위에서 움직일 때
FocusListener	focusGained(FocusEvent)	컴포넌트가 포커스를 받을 때
	focusLost(FocusEvent)	컴포넌트가 포커스를 잃을 때
TextListener	textValueChanged(TextEvent)	텍스트가 변경될 때
WindowListener	windowOpened(WindowEvent)	윈도우가 생성되어 처음으로 보이게 될 때
	windowClosing(WindowEvent)	사용자가 윈도우의 시스템 메뉴에서 윈도우 닫기를 시도할 때
	windowIconified(WindowEvent)	윈도우가 보통 크기에서 아이콘화될 때
	windowDeiconified(WindowEvent)	아이콘 상태의 윈도우가가 보통 상태로 복귀할 때
	windowClosed(WindowEvent)	윈도우 닫기 절차에 의해 윈도우가 닫혀졌을 때
	windowActivated(WindowEvent)	윈도우가 활성화 윈도우로 설정되어 활성화될 때
	windowDeactivated(WindowEvent)	활성화 상태의 윈도우가 비활성화될 때
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)	스크롤바를 움직였을 때
ComponentListener	componentHidden(ComponentEvent)	컴포넌트가 보이지 않는 상태로 될 때
	componentShown(ComponentEvent)	컴포넌트가 보이는 상태로 될 때
	componentResized(ComponentEvent)	컴포넌트의 크기가 변경될 때
	componentMoved(ComponentEvent)	컴포넌트의 위치가 변경될 때
ContainerListener	componentAdded(ContainerEvent)	컴포넌트가 컨테이너에 추가될 때
	componentRemoved(ContainerEvent)	컴포넌트가 컨테이너에서 삭제될 때

10.1.10 이벤트 리스너작성 예

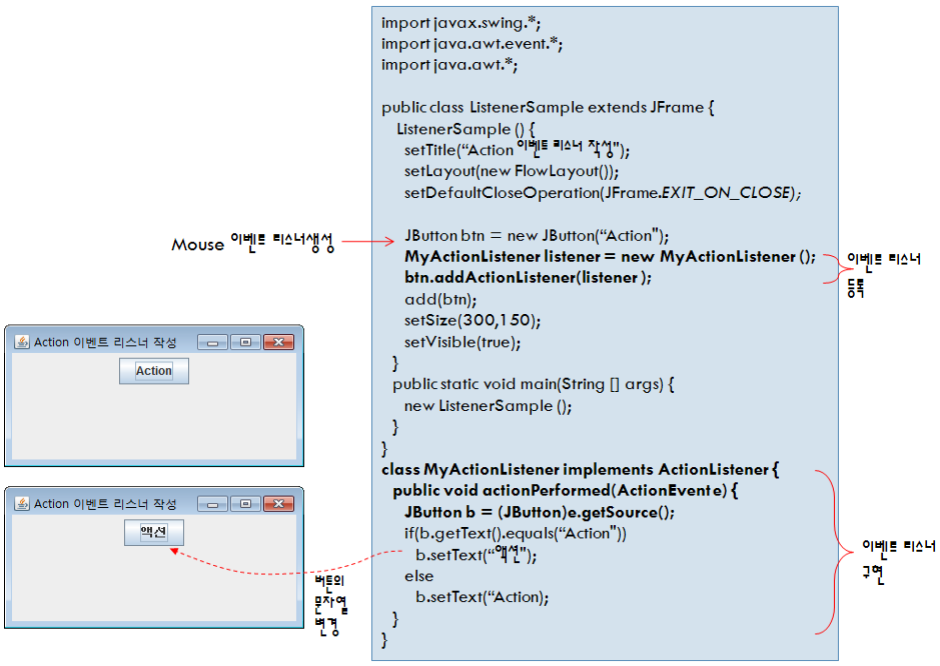
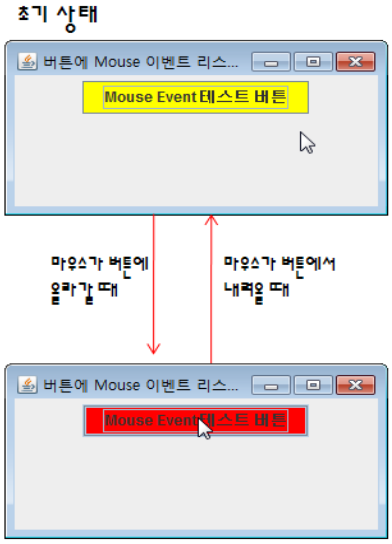


그림 10.5

-예제 10-1 : 버튼에 Mouse이벤트를 처리하는 예제




```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class ListenerMouseEx extends JFrame {
    ListenerMouseEx() {
        setTitle("버튼에 Mouse 이벤트 리스너 작성");
        setLayout(new FlowLayout());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton btn = new JButton("Mouse Event 테스트 버튼");
        btn.setBackground(Color.YELLOW);
        MyMouseListener listener = new MyMouseListener();
        btn.addMouseListener(listener);
        add(btn);
        setSize(300,150);
        setVisible(true);
    }

    public static void main(String [] args) {
        new ListenerMouseEx();
    }
}

class MyMouseListener implements MouseListener {
    public void mouseEntered(MouseEvent e) {
        JButton btn = (JButton)e.getSource();
        btn.setBackground(Color.RED);
    }
    public void mouseExited(MouseEvent e) {
        JButton btn = (JButton)e.getSource();
        btn.setBackground(Color.YELLOW);
    }
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
}
```

그림 10.6

10.1.11 Tip : 리스너 등록 메소드가 addXXXListener인 이유?

-한 컴포넌트는 서로 다른 이벤트에 대한 리스너를 동시에 여러개 가질 수 있다.

■ JButton.addActionListener(); // Action 리스너
■ JButton.addKeyListener(); // Key 리스너
■ JButton.addFocusListener(); // Focus 리스너

-한 컴포넌트는 한 이벤트에 대해 여러 개의 리스너를 동시에 가질 수 있다.

■ JButton.addActionListener(new MyButtonListener1());
■ JButton.addActionListener(new MyButtonListener2());
■ JButton.addActionListener(new MyButtonListener3());
■ 이때, 리스너는 등록된 반대 순으로 모두 실행된다.

10.1.12 이벤트 리스너 작성 방법

-3 가지 방법

■ 독립 클래스로 작성
  • 이벤트 리스너를 완전한 클래스로 작성
■ 내부 클래스(inner class)로 작성
■ 익명 클래스(anonymous class)로 작성

10.1.13 독립 클래스로 리스너 작성

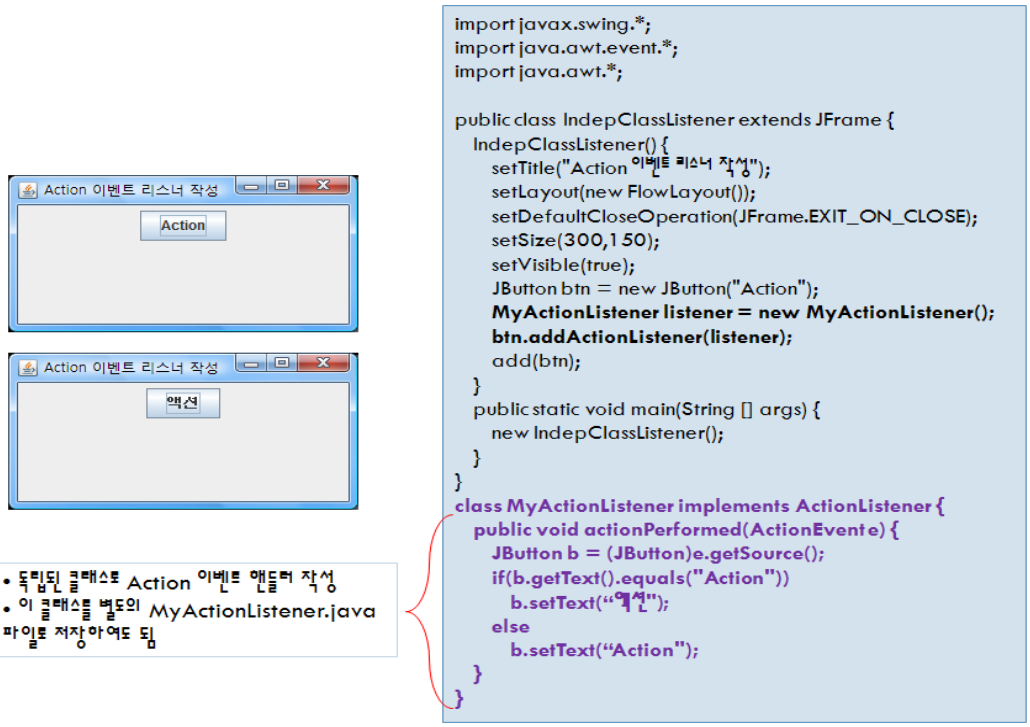


그림 10.7

10.1.14 내부 클래스로 리스너 작성

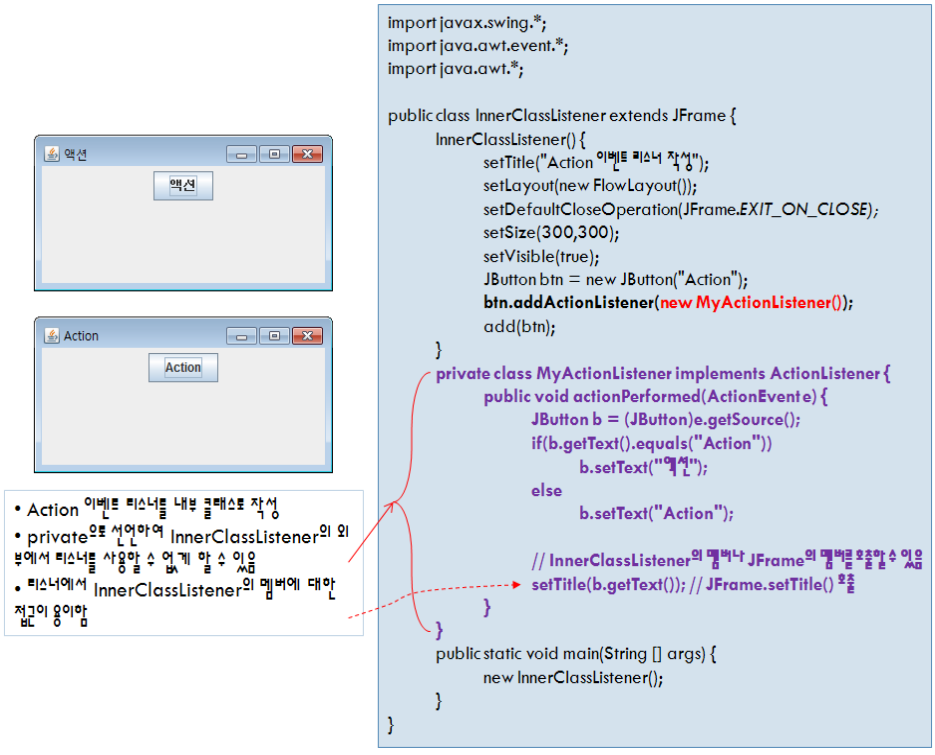


그림 10.8

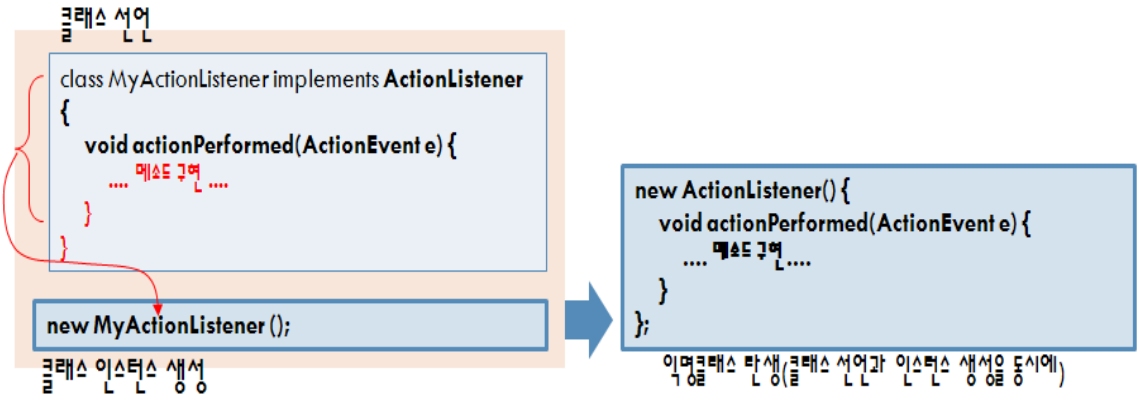
10.1.15 익명 클래스로 이벤트 리스너 작성

-익명 클래스란?

■(클래스 정의 + 인스턴스 생성)을 한번에 작성

```
new 익명클래스의슈퍼클래스이름(생성자의 인자들) {
    ..................
    클래스 정의
    ..................
};
```

■ActionListener를 구현하는 익명의 이벤트 리스너 작성 예



익명클래스로 다시 작성된 결과

```
btn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JButton b = (JButton)e.getSource();
        if(b.getText().equals("Action"))
            b.setText("액션");
        else
            b.setText("Action");
        // AnonymousClassListener의 멤버나
        // JFrame의 멤버를 호출할 수 있음
        setTitle(b.getText());
    }
}
);
}
```

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class AnonymousClassListener extends JFrame {
    AnonymousClassListener() {
        setTitle("Action 이벤트 리스너 작성");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout());
        setSize(300,300);
        setVisible(true);
        JButton btn = new JButton("Action");
        add(btn);
        btn.addActionListener(new MyActionListener() );
    }
    private class MyActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            JButton b = (JButton)e.getSource();
            if(b.getText().equals("Action"))
                b.setText("액션");
            else
                b.setText("Action");

            // InnerClassListener의 멤버나 JFrame의 멤버를 호출할 수 있음
            setTitle(b.getText());
        }
    }
    public static void main(String [] args) {
        new AnonymousClassListener ();
    }
}
```

-예제 10-2 : 마우스로 문자열 이동시키기

■마우스 버튼을 누르면 마우스가 있는 위치로 “hello” 문자열을 이동시킨다.

- 이벤트와 리스너 : MouseEvent와 MouseListener
- 이벤트 소스 : JPanel
- 구현할 리스너의 메소드 : mousePressed() 하나

- “hello” 문자열 표현 : JLabel
- 컨텐트 팬 : JPanel로 교체, 배치관리자 null로 설정

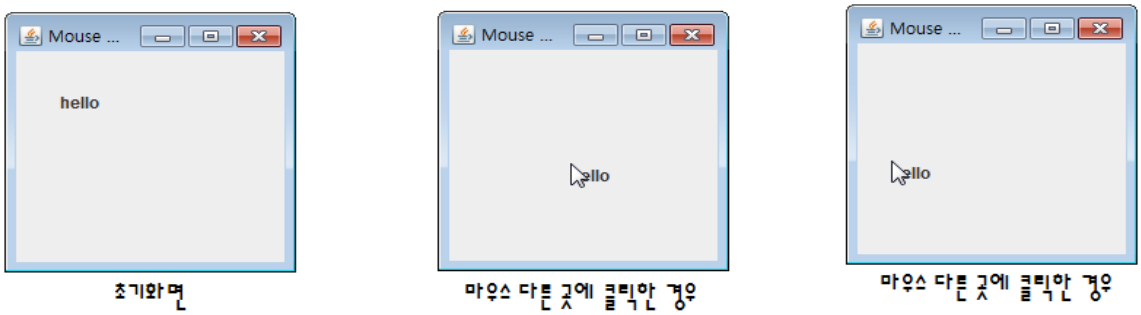
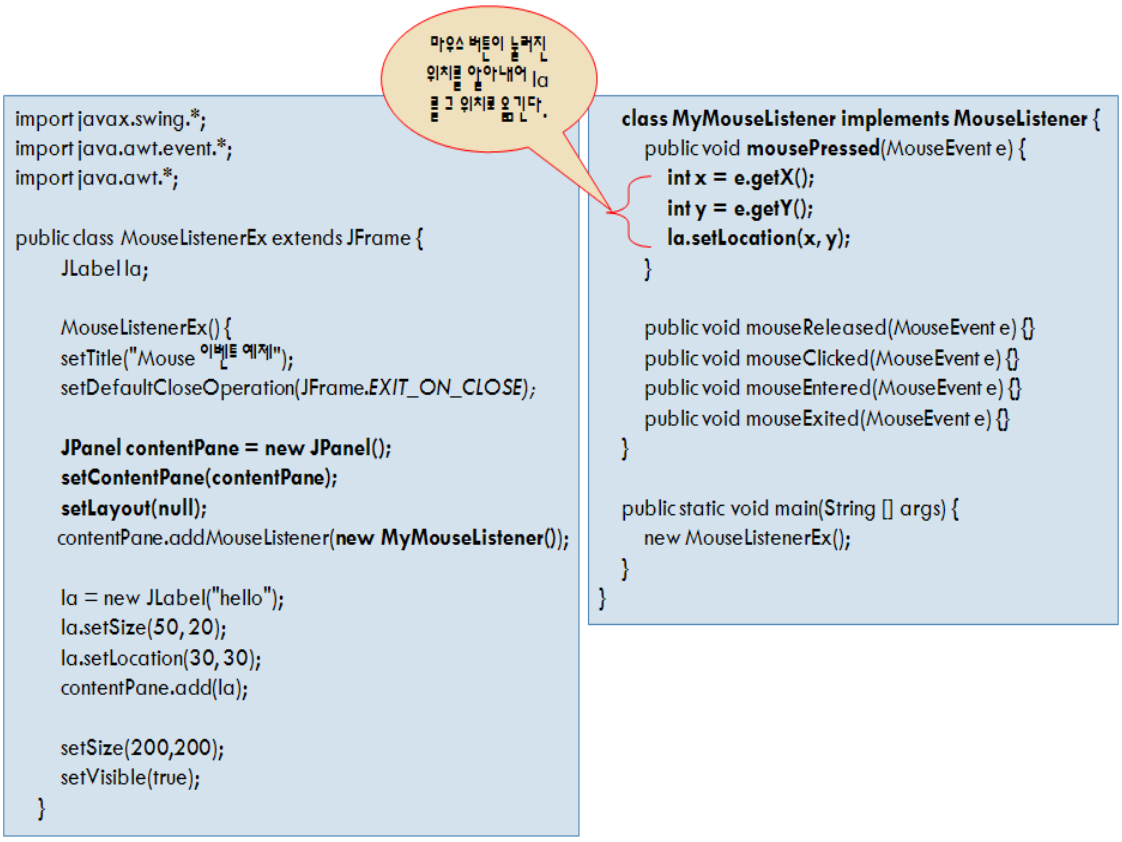


그림 10.9

-예제 10-2의 소스



10.1.16 어댑터(Adapter) 클래스

-이벤트 리스너 구현에 따른 부담

■ 리스너의 추상 메소드들을 모두 구현하여야 하는 부담

- 마우스 리스너에서 마우스가 눌러지는 경우(mousePressed())만 다루고자 하는 경우에도 나머지 4 개의 메소드도 모두 구현하여야 함.

-어댑터 클래스

■ JDK에서 제공

■ 이벤트 리스너를 간단히 구현해 놓은 클래스

- 리스너의 모든 메소드가 단순 리턴하도록 구현됨
- MouseAdapter 예

```
class MouseAdapter implements MouseListener {
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
}
```

■ 메소드를 하나만 가진 리스너는 해당 어댑터가 존재하지 않음
- ActionAdapter, ItemAdapter 등은 존재하지 않음

10.1.17 리스너와 어댑터 클래스

표 10.3

리스너 인터페이스	어댑터 클래스
ActionListener	없음
ItemListener	없음
KeyListener	KeyAdapter
MouseListener	MouseAdapter
MouseMotionListener	MouseMotionAdapter
FocusListener	FocusAdapter
TextListener	없음
WindowListener	WindowAdapter
AdjustmentListener	없음
ComponentListener	ComponentAdapter
ContainerListener	ContainerAdapter

10.1.18 어댑터 사용 예

```
JLabel la;
JPanel contentPane  = new JPanel();
contentPane.addMouseListener(new MyMouseListener());
……….

class MyMouseListener implements MouseListener {
    public void mousePressed(MouseEvent e) {
        int x = e.getX();
        int y = e.getY();
        la.setLocation(x, y);
    }
    public void mouseReleased(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
}
```

MouseListener를 이용한 경우

```
JLabel la;
JPanel contentPane = new JPanel();
contentPane.addMouseListener(new MyMouseAdapter());
……….

class MyMouseAdapter extends MouseAdapter {
    public void mousePressed(MouseEvent e) {
        int x = e.getX();
        int y = e.getY();
        la.setLocation(x, y);
    }
}
```

MouseAdapter를 이용한 경우

-예제 10-3: MouseAdapter 사용하기

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class MouseAdapterEx extends JFrame {
   JPanel contentPane = new JPanel();
   JLabel la;

   MouseAdapterEx() {
      setTitle("Mouse 이벤트 예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setContentPane(contentPane);
      setLayout(null);
      contentPane.addMouseListener(new MyMouseAdapter());
      la = new JLabel("hello");
      la.setSize(50, 20);
      la.setLocation(30, 30);
      contentPane.add(la);
      setSize(200,200);
      setVisible(true);
   }
```

```
class MyMouseAdapter extends MouseApater {
      public void mousePressed(MouseEvent e) {
         int x = e.getX();
         int y = e.getY();
         la.setLocation(x, y);
      }
   }

   public static void main(String [] args) {
      new MouseAdapterEx();
   }
}
```

10.1.19 Key 이벤트와 포커스

-키 이벤트는 키를 입력하는 다음 3 경우에 발생

■ 키를 누르는 순간

■ 누른 키를 떼는 순간

■ 누른 키를 떼는 순간 + Unicode 키가 입력된 경우

-키 이벤트를 받을 수 있는 조건

■ 키가 발생한 컴포넌트가 포커스를 가지고 있어야 함

-포커스

■ 키 이벤트를 독점하는 권한

■ 컴포넌트에 포커스 설정 방법

- component.requestFocus(); // component가 키 이벤트를 받을 수 있게 함

-모든 컴포넌트에 대해 사용자는 키 입력 가능

■ 키 이벤트는 모든 컴포넌트에 기본적으로 발생 가능

10.1.20 KeyListener의 메소드와 키

-KeyListener의 3 개의 메소드

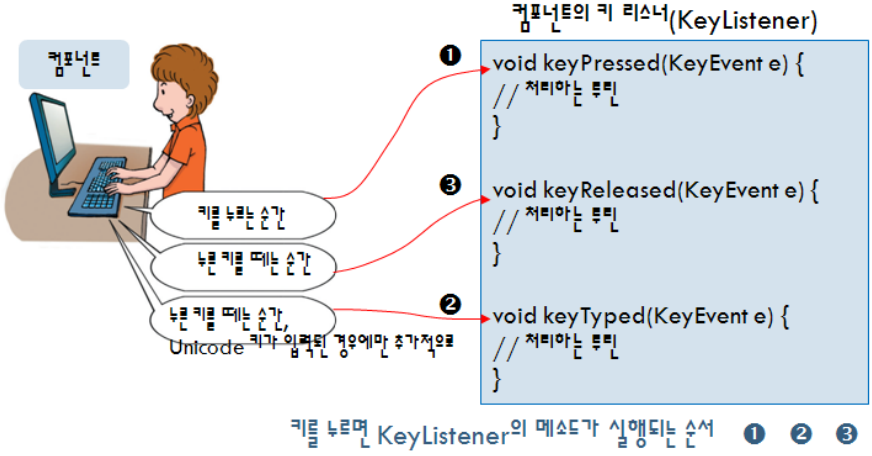


그림 10.10

-컴포넌트에 키 이벤트 리스너 등록

```
component.addKeyListener(myKeyListener);
```

10.1.21 유니코드(Unicode)

-유니코드 키의 특징

■국제 산업 표준

■전 세계의 문자를 컴퓨터에서 일관되게 표현하기 위한 코드 체계

■문자들에 대해서만 코드 값이 정의됨

- A~Z, a~z, 0~9, !, @, & 등

■문자 키가 아닌 경우에는 통일된 키 코드 값이 없음

- <Function> 키, <Home> 키, <Up> 키,<Delete> 키, <Control> 키, <Shift> 키, <Alt> 등

■유니코드 키가 입력되는 경우

- keyPressed(), keyReleased() 만 호출됨

■유니코드 키가 아닌 경우

- keyPressed(), keyTyped(), keyReleased() 가 모두 호출됨

10.1.22 입력된 키 판별

-KeyEvent 객체

■입력된 키 값을 가진 이벤트 객체

-KeyEvent의 메소드로 입력된 키 판별

■Unicode 키의 문자 값을 판별, char KeyEvent.getKeyChar()

- 눌러진 키에 해당하는 문자 값 리턴
- 눌러진 키가 Unicode 문자 키인 경우에만 의미 있음

■Unicode 문자 외 모든 키 판별, int KeyEvent.getKeyCode()

- 눌러진 키에 대한 정수형 키 코드 값 리턴
- Unicode 문자에 관계 없이, function 키, modifier 키, Control 키, Action 키 등 모든 키에 대해 키 코드 값 리턴
- 운영체제나 하드웨어에 따라 키 셋은 서로 다름
- 입력된 키를 판별하기 위해 가상키(Virtual Key) 값과 비교하여야 함
  -가상 키 값은 KeyEvent 클래스의 상수로 정의됨

-키 이름 문자열 리턴 String KeyEvent.getKeyText(int keyCode)

- static
- keyCode에 해당하는 키의 이름을 문자열을 리턴
  -F1 키의 경우 "F1", Shift 키의 경우 "SHIFT" 등의 문자열 리턴

10.1.23 가상 키(Virtual Key)

-가상 키는 KeyEvent 클래스에 상수로 정의되어 있음

-가상 키의 일부분

표 10.4

가상 키	설명	가상 키	설명
VK_0 ~ VK_9	0에서 9까지의 숫자키 '0' ~ '9'까지의 ASCII 값과	VK_LEFT	왼쪽 방향 키

	동일		
VK_A ~ VK_Z	A에서 Z까지의 문자키, 'A' ~ 'Z'까지의 ASCII 값과 동일	VK_RIGHT	오른쪽 방향 키
VK_F1~ VK_F24	<Function> 키 F1 ~ F24까지의 키 코드	VK_UP	<Up>키
VK_HOME	<Home>키	VK_DOWN	<Down> 키
VK_END	<End>키	VK_CONTROL	<Control> 키
VK_PGUP	<Page Up> 키	VK_SHIFT	<Shift> 키
VK_PGDN	<Page Down >키	VK_ALT	<Alt> 키
VK_UNDEFINED	입력된 키의 코드 값이 알 수 없음	VK_TAB	<Tab> 키

10.1.24 KeyListener의 메소드와 키

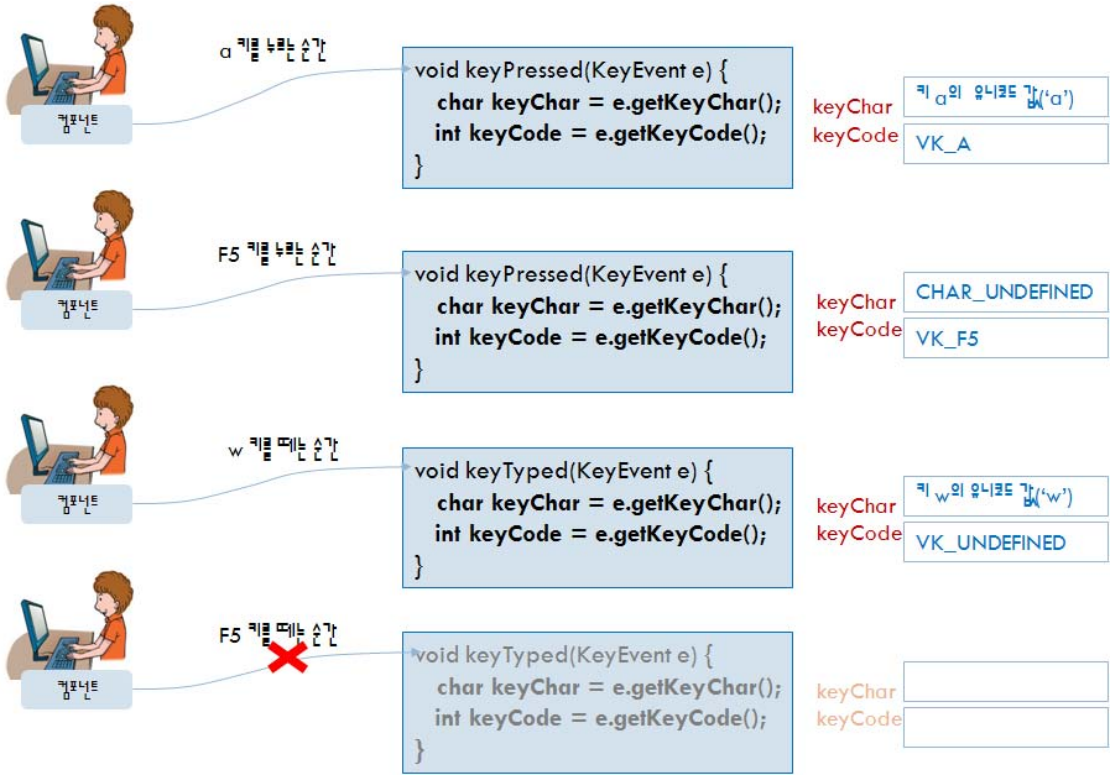


그림 10.11

10.1.25 KeyEvent와 KeyListener의 활용 : getKeyCode(), getKeyChar(), getKeyText() 사용

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class KeyListenerEx extends JFrame {
   JPanel contentPane = new JPanel();
   JLabel [] keyMessage;

   KeyListenerEx() {
      setTitle("KeyListener 예제");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      setContentPane(contentPane);
      contentPane.addKeyListener(new MyKeyListener());

      keyMessage = new JLabel[3];
      keyMessage[0] = new JLabel("  getKeyCode()  ");
      keyMessage[1] = new JLabel("  getKeyChar()  ");
      keyMessage[2] = new JLabel("  getKeyText()  ");

      for(int i=0; i<keyMessage.length; i++) {
         contentPane.add(keyMessage[i]);
         keyMessage[i].setOpaque(true);
         keyMessage[i].setBackground(Color.CYAN);
      }
      setSize(300,150);
      setVisible(true);
      contentPane.requestFocus();
   }

class MyKeyListener extends KeyAdapter {
      public void keyPressed(KeyEvent e) {
         int keyCode = e.getKeyCode();
         char keyChar = e.getKeyChar();
         keyMessage[0].setText(Integer.toString(keyCode));
         keyMessage[1].setText(Character.toString(keyChar));
         keyMessage[2].setText(e.getKeyText(keyCode));
      }
   }

   public static void main(String [] args) {
      new KeyListenerEx();
   }
}
```

• JComponent 컴포넌트에 바탕색을 지정하기 위해서는 사전에 컴포넌트가 불투명을 지정하여야 한다.

10.1.26 실행 결과

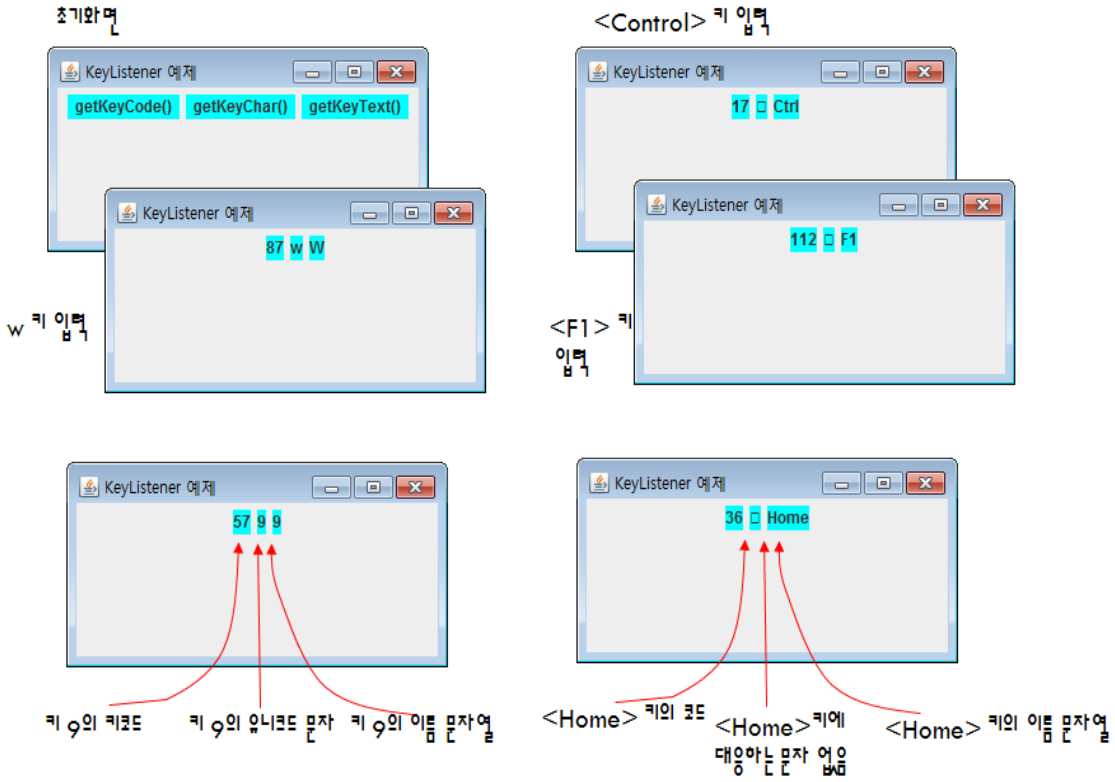


그림 10.12

-예제 10-4 : F1 키를 입력받으면 바탕을 초록색으로, % 키를 입력받으면 바탕을 노란색으로 변경

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class KeyCodeEx extends JFrame {
    JPanel contentPane = new JPanel();
    JLabel la = new JLabel();

    KeyCodeEx() {
        setTitle("Key Code 예제 : F1 키:초록색, % 키 노란색");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setContentPane(contentPane);
        contentPane.addKeyListener(new MyKeyListener());
        contentPane.add(la);
        setSize(300,150);
        setVisible(true);
        contentPane.requestFocus();
    }
```

```
    class MyKeyListener extends KeyAdapter {
        public void keyPressed(KeyEvent e) {
            la.setText(e.getKeyText(e.getKeyCode()));
            if(e.getKeyChar() == '%')
                contentPane.setBackground(Color.YELLOW);
            else if(e.getKeyCode() == KeyEvent.VK_F1)
                contentPane.setBackground(Color.GREEN);
        }
    }
    public static void main(String [] args) {
        new KeyCodeEx();
    }
}
```

JPanel이 키 입력을 받을 수 있도록 포커스를 준다.

• % 키를 판별하기 위해 e.getKeyChar() 이용
• '%' 문자와 비교

• F1 키를 판별하기 위해 e.getKeyChar() 이용
• KeyEvent.VK_F1 값과 비교

-예제 10-4 실행

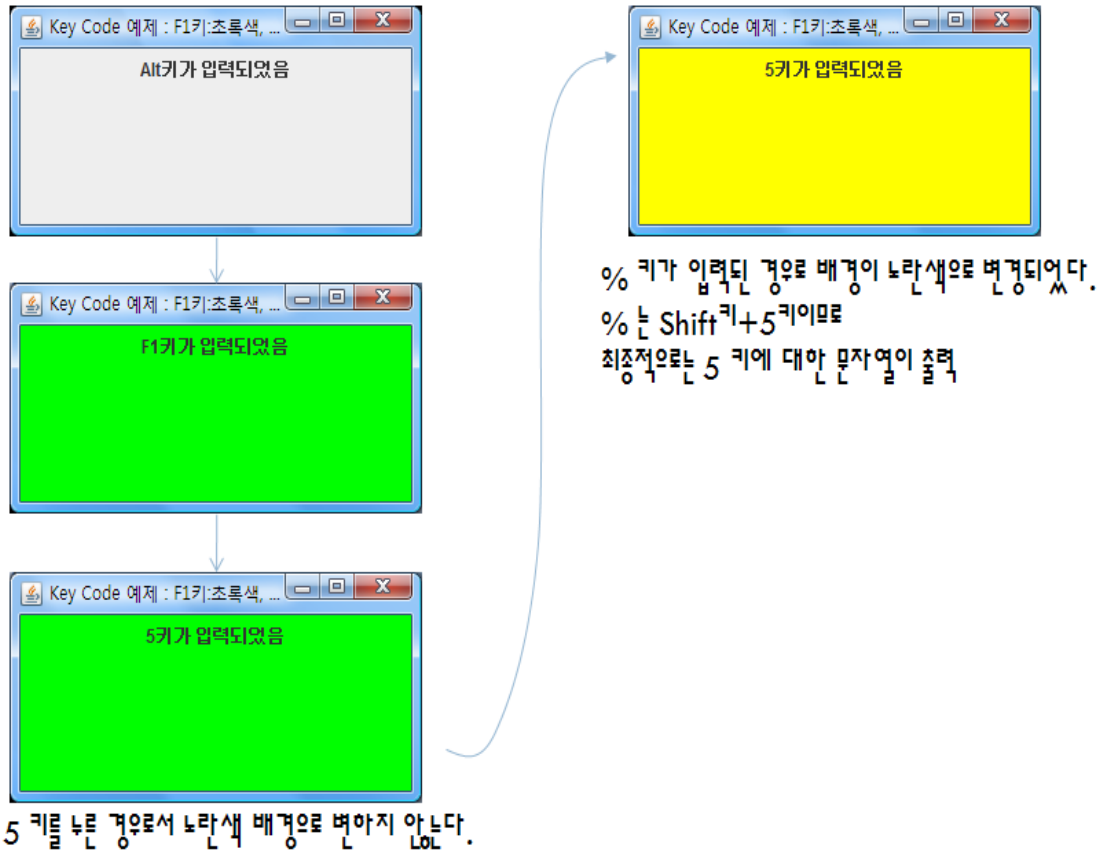


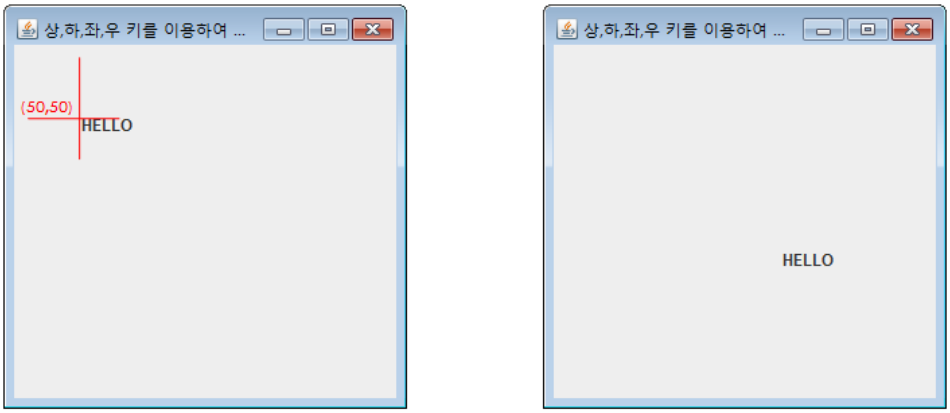
그림 10.13

-예제 10-5 : 상,하,좌,우 키로 “HELLO”문자열 움직이기

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class FlyingTextEx extends JFrame {
   JPanel contentPane = new JPanel();
   JLabel la = new JLabel("HELLO");
   final int FLYING_UNIT = 10;

   FlyingTextEx() {
      setTitle("상,하,좌,우 키를 이용하여 텍스트 움직이기");
      setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);

      setContentPane(p);
      contentPane.setLayout(null);
      contentPane.addKeyListener(new MyKeyListener());
      la.setLocation(50,50);
      la.setSize(100,20);
      contentPane.add(la);
      setSize(300,300);
      setVisible(true);
      contentPane .requestFocus();
   }
```

```
   class MyKeyListener extends KeyAdapter {
      public void keyPressed(KeyEvent e) {
         int keyCode = e.getKeyCode();
         switch(keyCode) {
            case KeyEvent.VK_UP:
               la.setLocation(la.getX(), la.getY()-FLYING_UNIT);
               break;
            case KeyEvent.VK_DOWN:
               la.setLocation(la.getX(), la.getY()+FLYING_UNIT);
               break;
            case KeyEvent.VK_LEFT:
               la.setLocation(la.getX()-FLYING_UNIT, la.getY());
               break;
            case KeyEvent.VK_RIGHT:
               la.setLocation(la.getX()+FLYING_UNIT, la.getY());
               break;
         }
      }
   }

   public static void main(String [] args) {
      new FlyingTextEx();
   }
}
```

-예제 실행: 상,하,좌,우 키로 텍스트 움직이기



상,하,좌,우 키를 움직이면 한 번에 10픽셀씩 “HELLO” 텍스트는 상,하,좌,우로 이동한다.
이 텍스트는 프레임의 영역을 벗어나서 움직일 수 있다.

그림 10.14

10.1.27 MouseEvent와 MouseListener,MouseMotionListener

-Mouse 이벤트

■사용자의 마우스 조작에 따라 발생하는 이벤트

표 10.5

Mouse 이벤트가 발생하는 경우	리스너의메소드	리스너
마우스가 컴포넌트 위에 올라갈 때	void mouseEntered(MouseEvent e)	MouseListener
마우스가 컴포넌트에서 내려올 때	voidmouseExited(MouseEvent e)	MouseListener
마우스 버튼이 눌러졌을 때	void mousePressed(MouseEvent e)	MouseListener
눌러진 버튼이 떼어질 때	void mouseReleased(MouseEvent e)	MouseListener
마우스가 컴포넌트를 클릭하였을 때	void mouseClicked(MouseEvent e)	MouseListener
마우스가 드래그되는 동안	void mouseDragged(MouseEvent e)	MouseMotionListener
마우스가 움직이는 동안	void mouseMoved(MouseEvent e)	MouseMotionListener

■마우스가 클릭되어 한 번 드래그될 때 메소드 호출 순서

```
mousePressed(), mouseDragged(), mouseReleased(), mouseClicked()
```

10.1.28 MouseEvent 로부터 얻을 수 있는 정보

-마우스 포인터의 위치

■int getX(), int getY(),

■Point getPoint()

```
public void mousePressed(MouseEvent e){
    int x = e.getX();
    int y = e.getY();
}
```

-입력된 마우스 버튼

■short getButton()

```
public void mousePressed(MouseEvent e){
    if(e.getButton() == MouseEvent.BUTTON1)
        System.out.println("Left Button Pressed");
}
```

-마우스 클릭 횟수

■int getClickCount()

```
public void mouseClicked(MouseEvent e){
    if(e.getClickCount() == 2){
        // 더블클릭을 처리하는 코드
    }
}
```

-팝업 메뉴 클릭

■boolean isPopupTrigger()

10.1.29 MouseListener와 MouseMotionListener 사용 예

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class MouseListenerAllEx extends JFrame {
    JPanel contentPane = new JPanel();
    JLabel la;

    MouseListenerAllEx() {
        setTitle("MouseListener와 MouseMOtionListener 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(contentPane);
        contentPane.addMouseListener(new MyMouseListener());
        contentPane.addMouseMotionListener(
                new MyMouseListener());
        la = new JLabel("No Mouse Event");
        contentPane.add(la);
        setSize(300,200);
        setVisible(true);
    }
```

```
    class MyMouseListener implements MouseListener,
                                     MouseMotionListener {
        public void mousePressed(MouseEvent e) {
            la.setText("MousePressed ("+e.getX()+","+e.getY()+")");
        }
        public void mouseReleased(MouseEvent e) {
            la.setText("MouseReleased("+e.getX()+","+e.getY()+")");
        }
        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {
            JPanel p = (JPanel)e.getSource();
            p.setBackground(Color.CYAN);
        }
        public void mouseExited(MouseEvent e) {
            JPanel p = (JPanel)e.getSource();
            p.setBackground(Color.YELLOW);
        }
        public void mouseDragged(MouseEvent e) {
            la.setText("MouseDragged ("+e.getX()+","+e.getY()+")");
        }
        public void mouseMoved(MouseEvent e) {
            la.setText("MouseMoved ("+e.getX()+","+e.getY()+")");
        }
    }

    public static void main(String [] args) {
        new MouseListenerAllEx();
    }
}
```

-실행: MouseListener와 MouseMotionListener 사용

초기화면

mouseEntered()에 의해 배경색 변경.
마우스 버튼이 눌러진 순간

눌러진 마우스 버튼이 떼어진 순간

마우스가 패널 위에 이동하는 동안

마우스가 패널 위에 드래깅하는 동안

마우스가 패널 바깥으로 나가면
mouseExited()에 의해 배경색 변경

그림 10.15

-예제 10-6 : 더블클릭시 컨텐트의 배경색 변경


```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class ClickAndDoubleClickEx extends JFrame {
    JPanel contentPane = new JPanel();

    ClickAndDoubleClickEx() {
        setTitle("Click and DoubleClick 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(contentPane);
        contentPane.addMouseListener(new MyMouseListener());
        setSize(300,200);
        setVisible(true);
    }

    class MyMouseListener extends MouseAdapter {
        public void mouseClicked(MouseEvent e) {
            if(e.getClickCount() == 2) {
                int r = (int)(Math.random()*256);
                int g = (int)(Math.random()*256);
                int b = (int)(Math.random()*256);

                JPanel p = (JPanel)e.getSource();
                p.setBackground(new Color(r,b,g));
            }
        }
    }

    public static void main(String [] args) {
        new ClickAndDoubleClickEx();
    }
}
```

그림 10.16

제11장 스윙 컴포넌트와 이벤트 핸들링

11.1 기초적인 스윙 컴포넌트와 상속 관계

1.기초적인 스윙 컴포넌트와 상속 관계

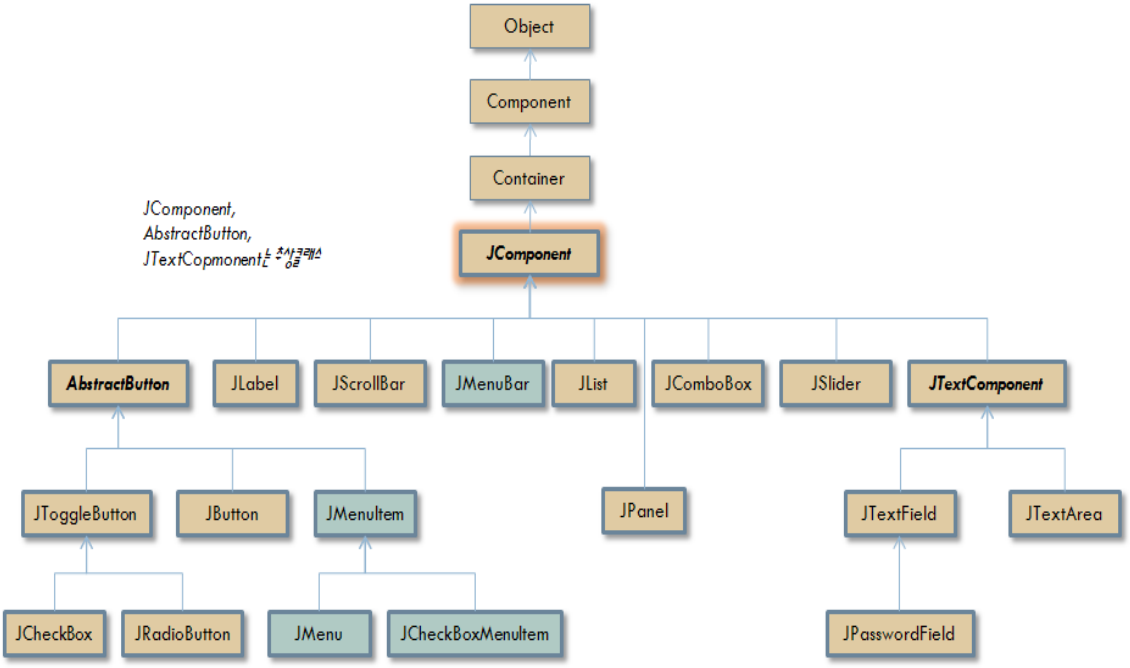


그림 11.1

2. 스윙컴포넌트의 공통 메소드. JComponent의메소드

컴포넌트의 모양과 관련된 메소드

```
void setForeground(Color) 전경색설정
void setBackground(Color) 배경색설정
void setOpaque(boolean) 불투명성설정
void setFont(Font) 폰트 설정
Font getFont() 폰트 리턴
```

컴포넌트의 상태와 관련된 메소드

```
void setEnabled(boolean) 컴포넌트 활성화 /비활성화
void setVisible(boolean) 컴포넌트 보이기 /숨기기
boolean isVisible() 컴포넌트의 보이는 상태 리턴
```

컴포넌트의 위치와 크기에 관련된 메소드

```
int getWidth() 폭 리턴
int getHeight() 높이 리턴
int getX() x 좌표 리턴
int getY() y 좌표 리턴
Point getLocationOnScreen() 스크린 좌표상에서의 컴포넌트 좌표
void setLocation(int, int) 위치 지정
void setSize(int, int) 크기 지정
```

컨테이너를 위한 메소드

```
Component add(Component) 자식 컴포넌트 추가
void remove(Component) 자식 컴포넌트 제거
void removeAll() 모든 자식 컴포넌트 제거
Component[] getComponents() 자식 컴포넌트 리스트 리턴
Container getParent() 부모 컨테이너 리턴
Container getTopLevelAncestor() 최상위 부모 컨테이너 리턴
```

3. 스윙 컴포넌트의 공통 메소드 확인 사례

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class SwingAPIEx extends JFrame {
    Container contentPane;
    JLabel la;
    JButton b1, b2, b3, b4;

    SwingAPIEx() {
        setTitle("Swing 공통 메소드 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout());

        b1 = new JButton("위치와 크기 정보");
        b1.addActionListener(new MyButtonListener());
        contentPane.add(b1);

        b2 = new JButton("모양 정보");
        b2.setOpaque(true);
        b2.setForeground(Color.MAGENTA);
        b2.setBackground(Color.YELLOW);
        b2.setFont(new Font("고딕체", Font.ITALIC, 20));
        b2.addActionListener(new MyButtonListener());
        contentPane.add(b2);

        b3 = new JButton("작동하지 않는 버튼");
        b3.setEnabled(false);
        b3.addActionListener(new MyButtonListener());
        contentPane.add(b3);

        b4 = new JButton("숨기기/보이기");
        b4.addActionListener(new MyButtonListener());
        contentPane.add(b4);

        setSize(250,200);
        setVisible(true);
    }
```

```
    class MyButtonListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            Object source = e.getSource();
            if(source == b1) {
                System.out.println("버튼의 위치와 크기");
                System.out.println("위치= (" + b1.getX() + "," + b1.getY() + ")");
                System.out.println("크기= (" + b1.getWidth() + "x" + b1.getHeight() + ")");

                JPanel c = (JPanel)b2.getParent();
                System.out.println("컨테이너의 위치와 크기");
                System.out.println("위치= (" + c.getX() + "," + c.getY() + ")");
                System.out.println("크기= (" + c.getWidth() + "x" + c.getHeight() + ")");
            }
            else if(source == b2) {
                System.out.println("폰트= " + b2.getFont());
                System.out.println("배경색= " + b2.getBackground());
                System.out.println("글꼴색= " + b2.getForeground());
            }
            else {
                if(b1.isVisible()) {
                    b1.setVisible(false);
                    b2.setVisible(false);
                    b3.setVisible(false);
                }
                else {
                    b1.setVisible(true);
                    b2.setVisible(true);
                    b3.setVisible(true);
                }
            }
        }

        public static void main(String [] args) {
            new SwingAPIEx();
        }
    }
```

4. 실행: 스윙 컴포넌트의 공통 요소

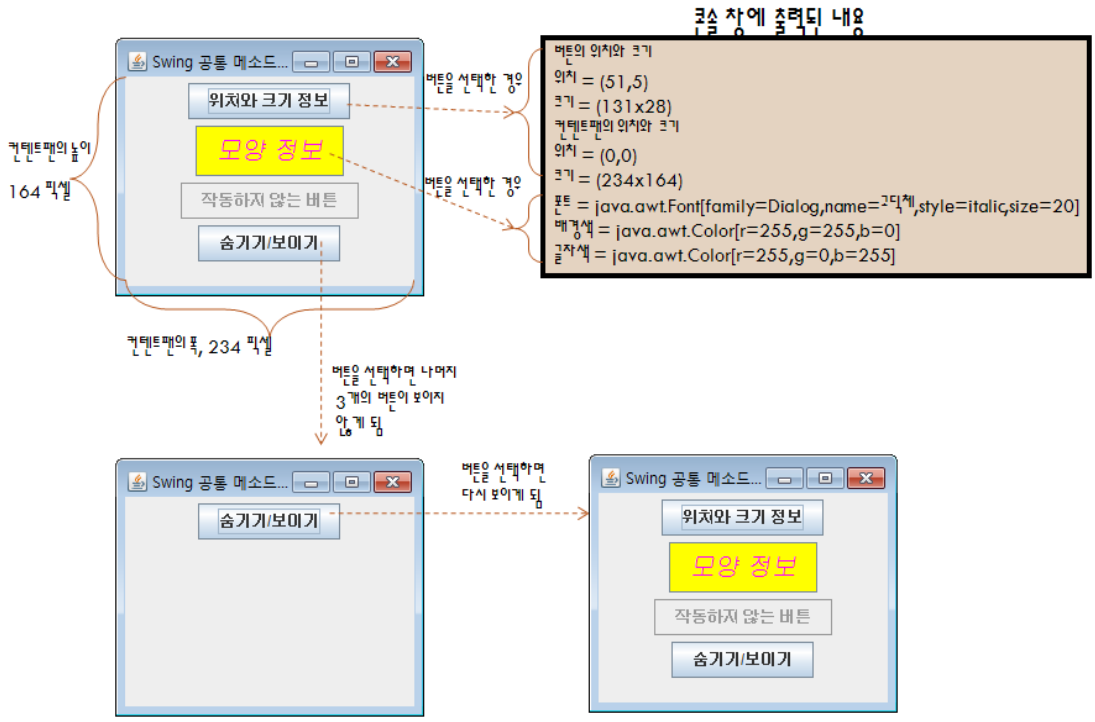


그림 11.2

5. JLabel, 레이블 컴포넌트

-JLabel의 용도

■ 텍스트나 이미지를 컴포넌트화 하기 위한 목적

-레이블 컴포넌트 생성

■ JLabel()

- 텍스트나 이미지 정보가 없는 빈 레이블 컴포넌트 생성

■ JLabel(Icon image)

- 이미지만을 가진 레이블 컴포넌트 생성

■ JLabel(String text)

- 텍스트만을 가진 레이블 컴포넌트 생성

■ JLabel(String text, Icon image, int hAlignment)

- 텍스트와 이미지, 수평 정렬 값을 가진 레이블 컴포넌트 생성
- 수평정렬 값인 hAlignment로 사용가능한 값들.
  -SwingConstants.LEFT, CENTER, RIGHT, LEADING or TRAILING

6. 레이블 컴포넌트 생성 예

-단순 텍스트 만을 가진 레이블 컴포넌트 생성

```
JLabel textLabel = new JLabel("사랑합니다");
```

-이미지를 가진 레이블 컴포넌트 생성

■ 이미지 파일로부터 이미지 생성을 위해 ImageIcon 클래스 사용

■ 출력 가능한 이미지 종류

- png, gif, jpg

■ sunset.jpg의 경로명이 "images/sunset.jpg"인 경우

```
ImageIcon image = new ImageIcon("images/sunset.jpg");
JLabel imageLabel = new JLabel(image);
```

-수평정렬 값을 가진 레이블 컴포넌트 생성

■ 수평정렬 값으로 사용되는 값을 생성자의 3 번째 인자로 지정

텍스트 이미지 모두 출력하고자 하는 경우 수평정렬 값을 지정하여야 함.

```
ImageIcon image = new ImageIcon("images/sunset.jpg");
JLabel label = new JLabel("사랑합니다", image, SwingConstants.CENTER);
```

-예제 11-1 : JLabel 컴포넌트 생성 예

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class LabelEx extends JFrame {
    Container contentPane;
    LabelEx() {
        setTitle("레이블 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout());
        JLabel textLabel = new JLabel("사랑합니다.");
        ImageIcon beauty = new ImageIcon("images/beauty.jpg");
        JLabel imageLabel = new JLabel(beauty);
        ImageIcon normalIcon = new ImageIcon("images/normalIcon.gif");
        JLabel label = new JLabel("보고싶으면 전화하세요", normalIcon,
                                  SwingConstants.CENTER);

        contentPane.add(textLabel);
        contentPane.add(imageLabel);
        contentPane.add(label);

        setSize(400,600);
        setVisible(true);
    }

    public static void main(String [] args) {
        new LabelEx();
    }
}
```

그림 11.3

7. JButton, 버튼 컴포넌트

-버튼 컴포넌트

■ 버튼 모양의 컴포넌트

■ 버튼은 클릭될 때 Action 이벤트를 발생시킴

-버튼 컴포넌트 생성

■ JButton()

- 텍스트나 이미지 아이콘를 가지지 않은 디폴트 버튼 생성

■ JButton(Icon icon)

- 이미지 아이콘만을 가진 버튼 생성

■ JButton(String text)

- 텍스트만을 가진 버튼 생성

■ JButton(String text, Icon icon)

- 텍스트와 이미지 아이콘을 모두 가진 버튼 생성

-버튼 컴포넌트 생성 예

■ “hello” 문자열을 가진 버튼 컴포넌트 생성 예

```
JButton btn = new JButton("hello");
```

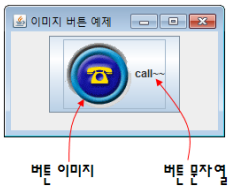


그림 11.4

8. 이미지를 가진 버튼 컴포넌트 만들기

-하나의 버튼에 3 개의 이미지 연결

■ 사용자의 마우스 접근에 따라 3 개의 이미지 중 선택 출력

-3 개의 버튼 이미지

■ 버튼의 보통 상태 때 출력되는 이미지 : normalIcon

- 생성자 호출 시에 주어진 이미지 아이콘

■ 버튼 위에 마우스가 올라갈 때 출력되는 이미지 : rolloverIcon

- 이미지 설정 메소드 : JButton.setRolloverIcon(Icon);

■ 마우스 버튼을 누른 상태 때 출력되는 이미지 : pressedIcon

- 이미지 설정 메소드 : JButton.setPressedIcon(Icon)

-이미지 아이콘 생성

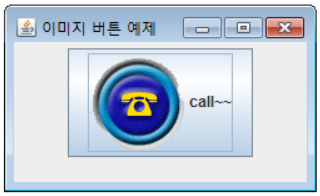
■ new ImageIcon(이미지 경로명);

■ new ImageIcon("images/normalIcon.gif);

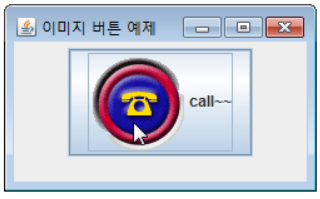
-예제 11-2 : 3 개의 이미지 아이콘을 가진 버튼 만들기

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

public class ButtonImageEx extends JFrame {
    Container contentPane;
    ButtonImageEx() {
        setTitle("버튼에 아이콘 달기 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout());

        ImageIcon normalIcon = new ImageIcon("images/normalIcon.gif");
        ImageIcon rolloverIcon = new ImageIcon("images/rolloverIcon.gif");
        ImageIcon pressedIcon = new ImageIcon("images/pressedIcon.gif");

        JButton btn = new JButton("call~~", normalIcon);
        btn.setRolloverIcon(rolloverIcon);
        btn.setPressedIcon(pressedIcon);
        contentPane.add(btn);

        setSize(250,200);
        setVisible(true);
    }

    public static void main(String [] args) {
        new ButtonImageEx();
    }
}
```

보통 상태에 있는 동안
(normalIcon.gif)



마우스가 버튼 위에
올라간 경우
(rolloverIcon.gif)



마우스가 눌러진 순간
(pressedIcon.gif)

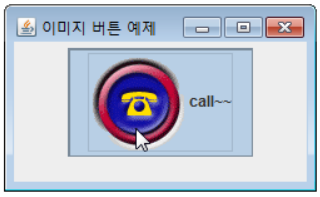


그림 11.5

9. 레이블과 버튼의 정렬(Alignment)

-수평 정렬

■ 컴포넌트 영역 내에 이미지와 텍스트의 수평상의 위치 결정

■ void setHorizontalAlignment(int align)

-수직 정렬

■ 컴포넌트 영역 내에 콘텐츠(이미지와 텍스트)의 수직상의 위치

■ void setVerticalAlignment(int align)

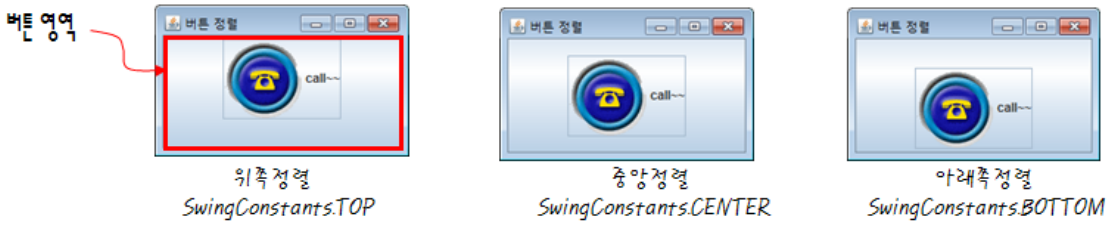


그림 11.6

10. JCheckBox, 체크박스 컴포넌트

-체크박스

■ 선택(selected)과 비선택(deselected)의 두 상태만을 가지는 버튼

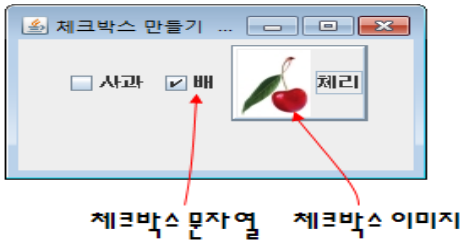


그림 11.7

-생성자

- 디폴트는 선택되지 않은 상태

■ JCheckBox ()

- 텍스트와 이미지가 없는 토글 버튼 생성

■ JCheckBox(Icon icon)

- 이미지만 가진 토글 버튼 생성

■ JCheckBox(Icon icon, boolean selected)

- 이미지와 지정된 선택 상태로 생성

■ JCheckBox(String text)

- 텍스트 만을 가진 토글 버튼 생성

■ JCheckBox(String text, boolean selected)

• 텍스트와 지정된 선택 상태로 생성

■ JCheckBox(String text, Icon icon)

• 텍스트와 이미지 둘 다 가진 토글 버튼 생성

■ JCheckBox(String text, Icon icon, boolean selected)

• 텍스트와 이미지를 가지고 지정된 선택상태로 생성

11. 체크 박스 생성

-텍스트 정보만을 가진 체크 박스 생성

■ "사과" 텍스트를 가진 체크박스 생성

```
JCheckBox c = new JCheckBox("사과");
```

■ "배" 텍스트를 가지고 선택상태로 체크박스 생성

```
JCheckBox c = new JCheckBox("배", true);
```

■ 체크 박스 모양이 명료하게 출력되고 사용자는 이것을 체크

-이미지 아이콘을 가진 체크 박스 생성 예

■ 체크 박스 모양이 출력되지 않음

■ 따로 선택상태를 표현하는 이미지 아이콘을 설정하여야 함

■ cherry.jpg 이미지와 "체리" 텍스트를 가진 체크 박스 생성 예

• 선택 상태의 이미지를 위해 selectedCherry.jpg를 사용하였음

```
ImageIcon cherryIcon = new ImageIcon("images/cherry.jpg");
ImageIcon selectedCherryIcon = new ImageIcon("images/selectedCherry.jpg");
JCheckBox cherry = new JCheckBox("체리", cherryIcon);
cherry.setSelectedIcon(selectedCherryIcon);
```

-예제 11-3 : 체크박스 생성 예

```
import javax.swing.*;
import java.awt.*;

public class CheckBoxEx extends JFrame {
    Container contentPane;
    CheckBoxEx() {
        setTitle("체크박스 만들기 예제");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout());

        ImageIcon cherryIcon = new ImageIcon("images/cherry.jpg");
        ImageIcon selectedCherryIcon =
                                new ImageIcon("images/selectedCherry.jpg");

        JCheckBox apple = new JCheckBox("사과");
        JCheckBox pear = new JCheckBox("배", true);
        JCheckBox cherry = new JCheckBox("체리", cherryIcon);
        cherry.setBorderPainted(true);
        cherry.setSelectedIcon(selectedCherryIcon);

        contentPane.add(apple);
        contentPane.add(pear);
        contentPane.add(cherry);

        setSize(250,150);
        setVisible(true);
    }
    public static void main(String [] args) {
        new CheckBoxEx();
    }
}
```

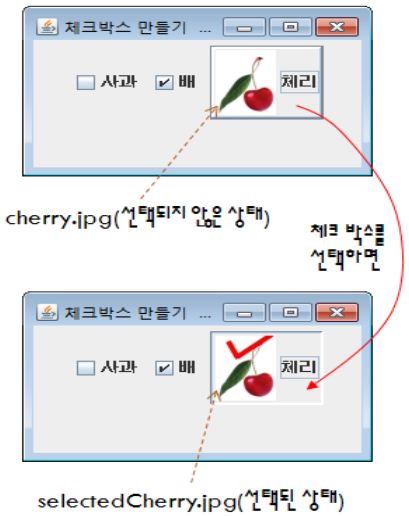


그림 11.8