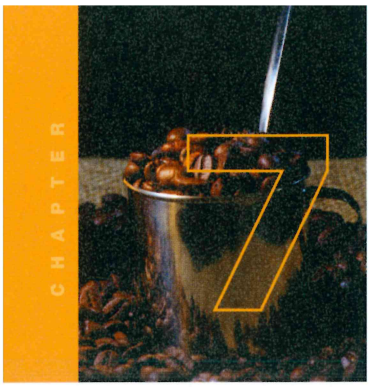


7

제네릭과 컬렉션

Objectives

- 컬렉션의 개념을 이해한다.
- 컬렉션을 구현한 클래스의 활용 방법을 안다.
- Vector<E>, ArrayList<E>, HashMap<K, V> 컬렉션을 활용할 수 있다.
- Iterator를 활용하여 컬렉션의 모든 요소를 검색할 수 있다.
- Collections 클래스를 활용할 수 있다.
- 제네릭의 개념을 이해한다.
- 제네릭 클래스와 인터페이스를 작성할 수 있다.
- 제네릭 메소드를 작성할 수 있다.



CHAPTER 7

JAVA PROGRAMMING

제네릭과 컬렉션

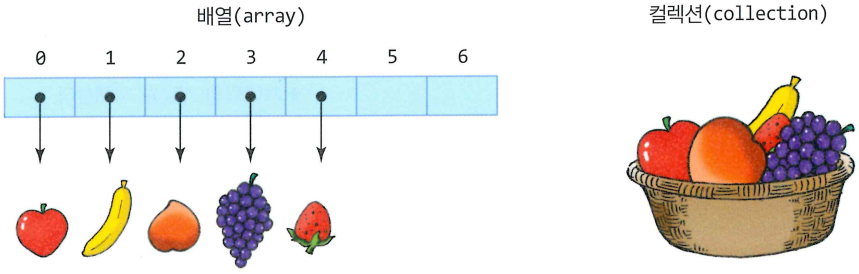
7.1 컬렉션과 제네릭

컬렉션의 개념

컬렉션
제네릭

컬렉션(collection)은 안드로이드를 비롯한 자바 프로그램을 작성하는데 빼놓을 수 없는 중요한 도구이다. 자바의 JDK는 소위 자료 구조 과목에서 배운 많은 자료 구조들을 컬렉션으로 만들어 제공하기 때문에 개발자는 이 장을 잘 이해할 필요가 있다. 또한 컬렉션은 제네릭(generics)이라는 기법으로 구현되어 있기 때문에 컬렉션을 공부하기 위해서는 제네릭에 대한 공부도 필요하다.

배열은 여러 개의 데이터를 다루는 데 편리한 자료 구조이지만, 삽입 삭제가 빈번하고, 데이터의 크기를 예측할 수 없는 응용프로그램에서 사용하기에는 불편하다. 예를 들어 많은 사람의 이름과 전화번호를 저장되고 삽입 삭제가 빈번한 전화번호부 관리 프로그램, 선, 원, 타원, 사각형 등 빈번한 도형 생성과 삭제를 관리하는 그래픽 편집기, 블록들이 새로 생성되고 움직이며 위치를 바꾸고 어떤 조건을 만나면 사라지기도 하는 테트리스 게임 등에 고정 크기의 배열을 사용한다면 프로그램 작성에 상당한 어려움을 겪을 것이다.



- 고정 크기 이상의 객체를 관리할 수 없다.
- 배열의 중간에 객체가 삭제되면 응용프로그램에서 자리를 옮겨야 한다.

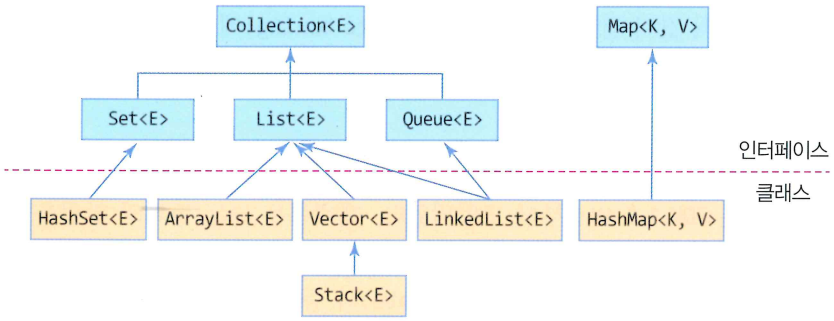
- 가변 크기로서 객체의 개수를 염려할 필요 없다.
- 컬렉션 내의 한 객체가 삭제되면 컬렉션이 자동으로 자리를 옮겨준다.

[그림 7-1] 배열과 컬렉션의 개념 차이

컬렉션은 배열이 가진 고정 크기의 단점을 극복하기 위해 객체들을 쉽게 삽입, 삭제, 검색할 수 있는 가변 크기의 컨테이너(container)이다. [그림 7-1]은 배열과 컬렉션의 개념적 차이를 비교하여 보여준다.

컬렉션을 위한 자바 인터페이스와 클래스

java.util 패키지는 다양한 컬렉션 인터페이스와 컬렉션 클래스를 제공한다. 핵심적인 컬렉션 인터페이스와 클래스를 정리하면 [그림 7-2]와 같다.



[그림 7-2] 컬렉션 인터페이스와 클래스

컬렉션 클래스는 개발자가 바로 사용할 수 있는 것들로서, Vector<E>와 ArrayList<E>는 가변 크기의 배열을 구현하며, LinkedList<E>는 노드들이 링크로 연결되는 리스트를 구현한다. Stack<E>는 스택을 구현하며, HashSet<E>은 집합을 구현한다. 이들은 모두 Collection<E>를 상속받고, 단일 클래스의 객체만을 요소로 다루는 공통점이 있다.

이와 달리 HashMap<K, V>는 '키(K)'와 '값(V)'의 쌍으로 이루어지는 데이터를 저장하고, '키'로 쉽게 검색하도록 만든 컬렉션이다.

컬렉션의 특징

컬렉션을 이해하기 위해 컬렉션의 중요한 특징을 알아보자.

첫째, 컬렉션은 제네릭(generics)이라는 기법으로 만들어져 있다. 컬렉션 클래스의 이름에는 [그림 7-2]와 같이 <E>, <K>, <V> 등이 항상 포함된다. 이들은 '타입 매개변수'라고 하며, Vector<E>에서 E 대신 Integer와 같이 구체적인 타입을 지정하면, Vector<Integer>는 정수 값만 저장하는 벡터로, Vector<String>은 문자열만 저장하는 벡터로 사용할 수 있다. 특정 타입만 다루지 않고 여러 종류의 타입으로 변신할 수 있도록, 컬렉션을 일반화시키기 위해 <E>를 사용하는 것이다. 그러므로 E를 일반화시

제네릭
타입 매개변수

킨 타입 혹은 제네릭 타입(generic type)이라고 부른다. 컬렉션은 여러 타입의 값을 다룰 수 있도록 변신이 가능한 자료 구조이지만, 컬렉션을 사용할 때는 지정된 특정 타입의 값만 저장 가능하다.

객체들만 가능

둘째, 컬렉션의 요소는 객체들만 가능하다. int, char, double 등의 기본 타입의 데이터는 원칙적으로 컬렉션의 요소로 불가능하다. 다음 코드를 참고하라.

오류

```
Vector<int> v = new Vector<int>();          // 컴파일 오류. int는 사용 불가
Vector<Integer> v = new Vector<Integer>(); // 정상 코드
```

하지만, 기본 타입의 값이 삽입되면 자동 박싱(auto boxing)에 의해 Wrapper 클래스 타입으로 변환되어 객체로 저장한다. 자동 박싱은 6.6절을 참고하기 바란다.

제네릭의 기본 개념

템플릿
일반화(generic)

제네릭(generics)은 JDK 1.5 버전부터 도입되었다. 제네릭은 모든 종류의 타입을 다룰 수 있도록, 클래스나 메소드를 타입 매개변수(generic type)를 이용하여 선언하는 기법이다. 자바의 제네릭은 C++의 템플릿(template)과 동일하다. 템플릿이란 국어사전에서 '형판'이란 뜻이다. 다른 말로 '본 떠 찍어내기 위해 만들어진 틀'이다. 자바의 제네릭은 클래스 코드를 찍어내듯이 생산할 수 있도록 일반화(generic)시키는 도구이다.

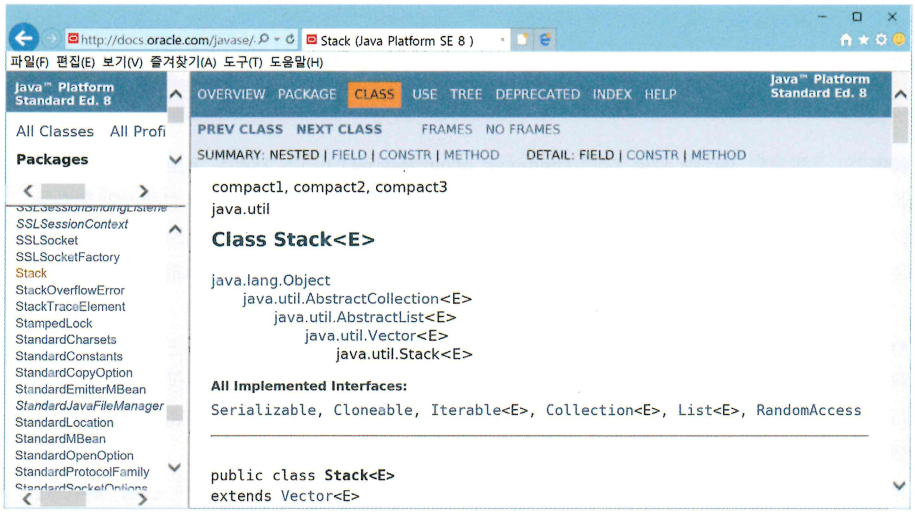
Stack<E> 클래스의 예를 들어보자. Stack<E> 클래스는 제네릭 타입 <E>를 가진 제네릭 클래스이며 그 모양은 다음과 같고, [그림 7-3]은 JDK의 API 레퍼런스 페이지를 캡처한 내용이다.

```
class Stack<E> {
    ...
    void push(E element) { ... }
    E pop() { ... }
    ...
}
```

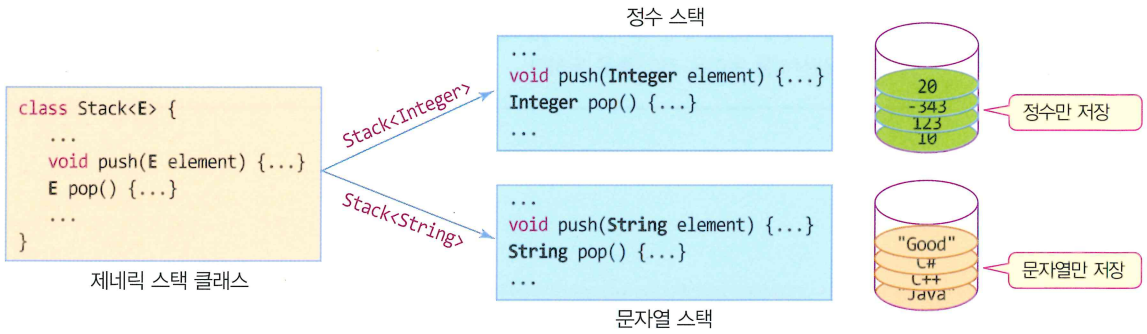
구체화

Stack<E>에서 E에 구체적인 타입을 지정하면, 지정된 타입만 다룰 수 있는 구체화된(specialized) 스택이 된다. 예를 들어, Stack<Integer>는 Integer 타입만 다루는 스택이 되고, Stack<Point>는 Point 타입의 객체만 푸시(push)하고 팝(pop)할 수 있는 스택이 된다. [그림 7-4]는 제네릭 Stack<E>를 구체화한 사례를 보여준다.

7.3절에서는 개발자가 스스로 제네릭 클래스나 제네릭 메소드를 만드는 방법을 소개한다. 개발자가 필요한 웬만한 컬렉션 클래스들은 이미 JDK에 제네릭으로 만들어져 있기 때문에, JDK에 있는 컬렉션을 사용하는 방법을 아는 것이 보다 바람직하다.



[그림 7-3] 제네릭 Stack<E> 클래스



[그림 7-4] 제네릭 Stack<E>를 특정 타입으로 구체화(specialization)한 경우

제네릭 타입 매개변수

컬렉션 클래스에서 타입 매개변수로 사용하는 문자는 다른 변수와 혼동을 피하기 위해 일반적으로 하나의 대문자를 사용한다. 다음은 관례적으로 사용하는 문자이다.

하나의 대문자

- E: Element를 의미하며 컬렉션에서 요소임을 나타냄
- T: Type을 의미
- V: Value를 의미
- K: Key를 의미



잠깐!

2000년도 초반부터 제네릭 프로그래밍(generic programming)이라는 새로운 프로그래밍 패러다임이 도입되었다. 동일한 프로그램 코드에 다양한 데이터 타입을 적용할 수 있도록 클래스와 함수들을 일반화시켜 제네릭 함수와 제네릭 클래스를 만들고, 제네릭 함수나 제네릭 클래스에 개발자가 원하는 데이터 타입으로 구체화시켜 함수나 클래스의 프로그램 코드를 틀에서 찍어내는 듯이 생산하는 기법이다.

C++의 경우 제네릭은 템플릿(template)으로도 불리는데, 2003년 C++ 표준위에서는 입출력 라이브러리를 모두 제네릭 클래스와 제네릭 함수로 표준화하였으며, 자바의 컬렉션에 해당하는 컨테이너, 알고리즘 등을 일반화시킨 라이브러리를 STL(Standard Template Library)로 제공하는 등 대부분의 C++ 라이브러리는 제네릭으로 구성되어 있다.

자바는 뒤이어 2004년 JDK 1.5 beta 버전에서부터 제네릭을 도입하였다. 제네릭은 현재, 자바뿐만 아니라 C++, C#, Visual Basic, Delphi 등 다양한 언어에 도입되어 있으며, 점점 그 사용이 확산되고 있다. 개발자들은 더 이상 지체하지 말고 제네릭 개념과 제네릭을 이용한 제네릭 프로그래밍을 학습하기 바란다.

스택, 큐, 리스트, 링크드 리스트, 벡터, 해시맵, 셋(set) 등 웬만한 자료 구조나 알고리즘은 대부분의 언어에서 제네릭으로 구현되어 있기 때문에, 개발자의 입장에서 이들을 잘 알고 잘 사용하여 프로그래밍 생산성을 높이는 것이 좋을 듯하다.

CHECK TIME



1 컬렉션이 배열보다 좋은 점은 무엇인가?

2 다음 중 컬렉션이 아닌 것은?

① Vector ② ArrayList ③ StringBuffer ④ HashMap

3 아래 빈칸에 적절한 말을 넣어라.

Stack<E>는 스택을 일반화시킨 _____으로서 E를 _____라고 부르며, E에 Integer, String, Point 등 특정 타입을 지정하여 특정 타입만 다루는 스택을 만들 수 있다. 예를 들어, Stack<E>가 실수만 다루는 스택이 되도록 구체화하면 _____로 표현한다.

7.2 제네릭 컬렉션 활용

Vector<E>

Vector<E>

Vector<E>(이하 Vector 또는 벡터)는 배열을 가변 크기로 다룰 수 있게 하고, 객체의 삽입, 삭제, 이동이 쉽도록 구성한 컬렉션 클래스이다. 벡터는 삽입되는 요소의 개수에 따라 자동으로 크기를 조절하고, 요소의 삽입과 삭제에 따라 자동으로 요소들의 자리를 이동한다. 〈표 7-1〉은 Vector<E> 컬렉션의 주요 메소드를 보여준다.

● 벡터 생성

벡터를 생성할 때, Vector<E>의 E에 요소로 사용할 타입을 지정해야 한다. 예를 들어, 정수 값만 삽입 가능한 벡터를 만들고자 하면 다음과 같이 E에 Integer를 지정하여 벡터를 생성한다.

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

벡터 v는 [그림 7-5]와 같이 구성되며, int, char, double 등의 기본 타입은 E에 사용할 수는 없다.

오류

```
Vector<int> v = new Vector<int>();          // 오류. int는 사용 불가
```

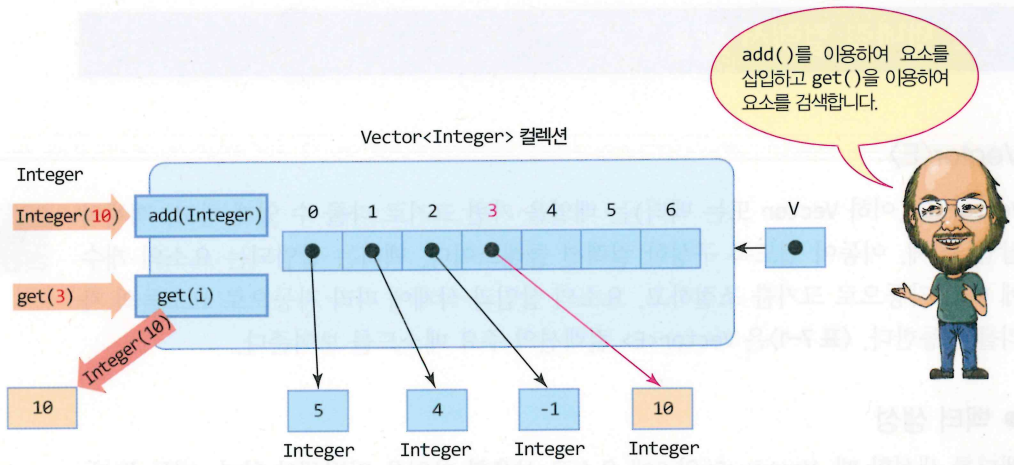
레퍼런스 변수 선언과 벡터 생성을 분리하여 코드를 만들 수 있으며, 문자열만 다루는 벡터는 다음과 같이 생성할 수 있다.

```
Vector<String> stringVector;                // 제네릭 컬렉션에 대한 레퍼런스 변수 선언
StringVector = new Vector<String>();        // 문자열 벡터 생성
```

만일 Vector<E>에서 E에 구체적인 타입을 지정하지 않고 Vector로만 사용하면, 컴파일러가 [그림 7-6]과 같은 경고 메시지를 출력한다.

개발자는 생성된 벡터의 용량을 굳이 알 필요 없다. 컬렉션은 자신의 용량을 스스로 조절하기 때문이다. 만일, 용량을 초기에 설정하고자 한다면 다음과 같이 생성자에 용량을 지정하면 된다.

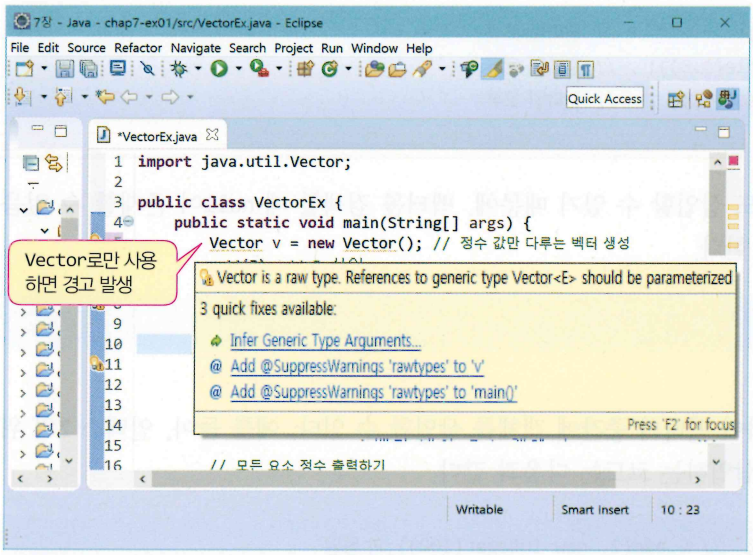
```
Vector<Integer> v = new Vector<Integer>(5);     // 초기 용량이 5인 벡터 생성
```



[그림 7-5] Vector<Integer> v = new Vector<Integer>() 벡터 생성

〈표 7-1〉 Vector<E> 클래스의 주요 메소드

메소드	설명
boolean add(E element)	벡터의 맨 뒤에 element 추가
void add(int index, E element)	인덱스 index에 element를 삽입
int capacity()	벡터의 현재 용량 리턴
boolean addAll(Collection<? extends E> c)	컬렉션 c의 모든 요소를 벡터의 맨 뒤에 추가
void clear()	벡터의 모든 요소 삭제
boolean contains(Object o)	벡터가 지정된 객체 o를 포함하고 있으면 true 리턴
E elementAt(int index)	인덱스 index의 요소 리턴
E get(int index)	인덱스 index의 요소 리턴
int indexOf(Object o)	o와 같은 첫 번째 요소의 인덱스 리턴, 없으면 -1 리턴
boolean isEmpty()	벡터가 비어 있으면 true 리턴
E remove(int index)	인덱스 index의 요소 삭제
boolean remove(Object o)	객체 o와 같은 첫 번째 요소를 벡터에서 삭제
void removeAllElements()	벡터의 모든 요소를 삭제하고 크기를 0으로 만듦
int size()	벡터가 포함하는 요소의 개수 리턴
Object[] toArray()	벡터의 모든 요소를 포함하는 배열 리턴



[그림 7-6] Vector<E>에 E 대신 구체적인 타입을 지정하지 않은 경우 경고 발생

● 벡터에 요소 삽입

add() 메소드를 이용하면 벡터의 끝이나 중간에 요소를 삽입할 수 있다. 다음은 정수 5, 4, -1을 순서대로 벡터의 맨 뒤에 삽입하는 예로서, 정수를 Integer 객체를 만들어 삽입한다.

add()

```
v.add(new Integer(5));
v.add(new Integer(4));
v.add(new Integer(-1));
```

자동 박싱 기능을 활용하면 앞의 코드는 다음과 같이 해도 된다.

```
v.add(5);  // 5 -> new Integer(5)로 자동 박싱됨
v.add(4);
v.add(-1)
```

자동 박싱에 의해 int 타입의 정수는 자동으로 Integer 객체로 변환되어 삽입된다. 그러나 벡터 v에는 Integer 외의 다른 타입의 객체를 삽입할 수 없다. 다음은 오류 코드이다.

자동 박싱

오류

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

null

벡터에는 null도 삽입할 수 있기 때문에, 벡터를 검색할 때 null이 존재할 수 있음을 염두에 두어야 한다.

```
v.add(null);
```

벡터의 중간 삽입

add()를 이용하여 벡터의 중간에 객체를 삽입할 수 있다. 예를 들어, 인덱스 2의 위치에 정수 100을 삽입하는 코드는 다음과 같다.

```
v.add(2, 100);  // v.add(2, new Integer(100));과 동일
```

이 코드는 인덱스 2의 위치에 정수 100을 삽입하고 기존의 인덱스 2와 그 뒤에 있는 요소들을 모두 한 자리씩 뒤로 이동시킨다. 하지만, 벡터에 1개의 요소(인덱스 0의 위치)만 들어있는 상태라면 이 코드가 실행될 때 예외가 발생한다. 인덱스 1이 빈 공간이 되기 때문이다.

● 벡터 내의 요소 알아내기

get()
elementAt()

벡터 내에 존재하는 요소를 알아내기 위해서는 get()이나 elementAt() 메소드를 이용한다. 다음과 같이 벡터에 3개의 정수가 들어 있다고 하자.

```
Vector<Integer> v = new Vector<Integer>();
v.add(5);
v.add(4);
v.add(-1);
```

get()이나 elementAt() 메소드는 인자로 주어진 인덱스에 있는 Integer 객체를 리턴한다. 다음은 벡터 v의 인덱스 1 위치에 있는 정수 값을 읽는 코드이다.

```
Integer obj = v.get(1);  // 벡터의 1번째 Integer 객체를 얻어낸다.
int i = obj.intValue();   // obj에 있는 정수를 알아냄. 이 값은 4
```

앞의 두 문장은 다음 한 문장으로 써도 된다.

```
int i = v.get(1);  // 자동 언박싱
```

자동 언박싱에 의해 v.get(1)이 리턴하는 Integer 객체의 정수 값(4)이 변수 i에 저장된다.

● 벡터의 크기와 용량 알아내기

size()

벡터의 크기란 벡터에 들어 있는 요소의 개수를 말하며, 벡터의 용량이란 수용할 수 있는 크기를 말한다. 벡터의 크기는 다음과 같이 size() 메소드를 호출한다.

```
int len = v.size();  // 벡터의 크기. 벡터에 존재하는 요소 객체의 수
```

capacity()

벡터의 용량은 다음 capacity() 메소드를 호출한다.

```
int cap = v.capacity();  // 벡터의 용량
```

● 벡터에서 요소 삭제

remove()

벡터 내에 임의의 인덱스에 있는 요소를 삭제할 수 있다. 다음과 같이 remove() 메소드를 이용한다.

```
v.remove(1);  // 인덱스 1의 위치에 있는 요소 삭제
```

이 코드는 인덱스 1의 위치에 있는 요소를 삭제한다. 코드의 실행 결과 뒤에 있는 요소들이 한 자리씩 앞으로 이동한다. 다음과 같이 객체 레퍼런스를 이용하여 remove()를 호출할 수도 있다.

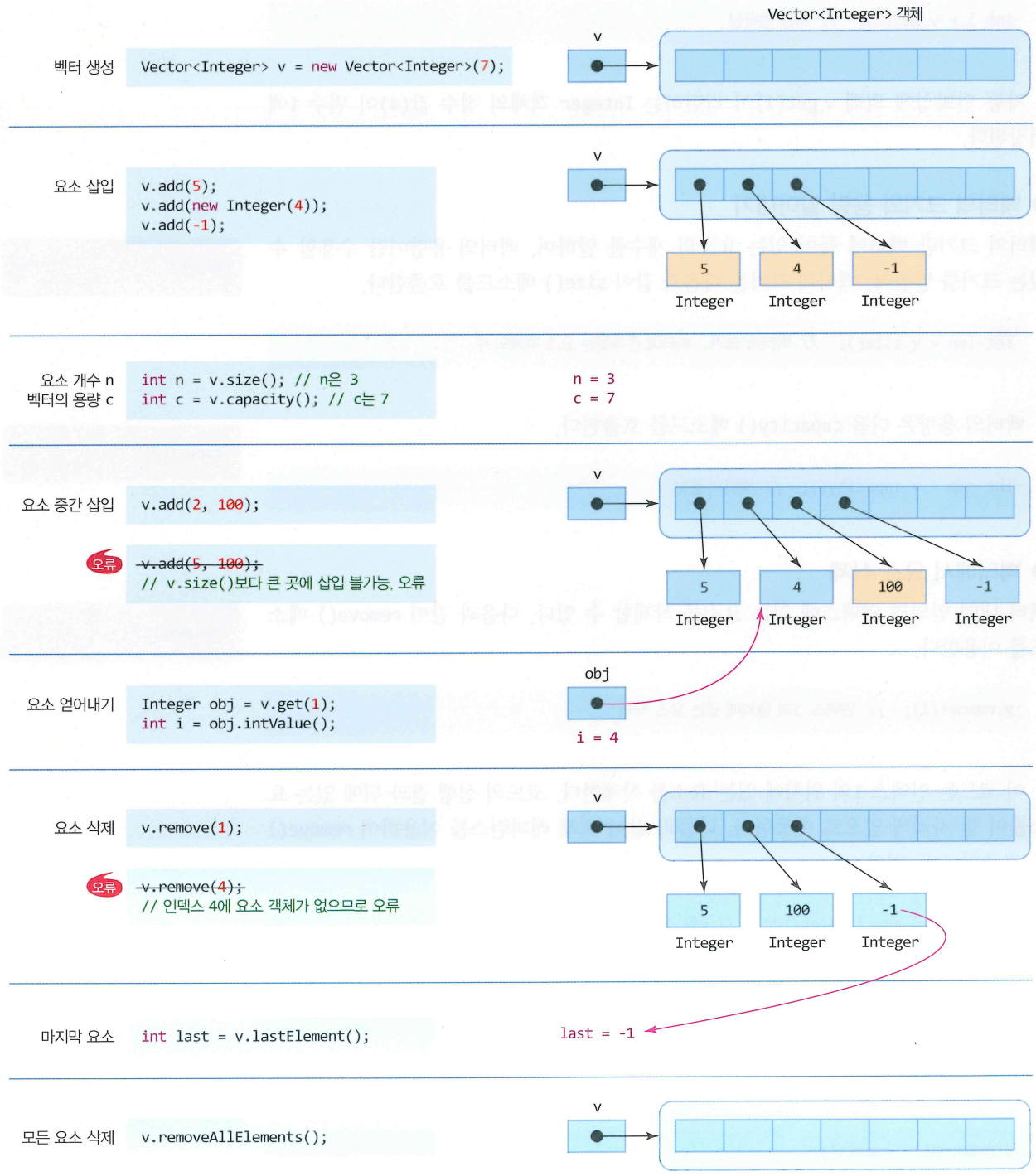
```
Integer m = new Integer(100); // m은 객체 레퍼런스
v.add(m);
...
v.remove(m);  // 레퍼런스 m의 요소 삭제
```

removeAllElements()

벡터의 모든 요소를 삭제하려면, 다음과 같이 removeAllElements()를 호출한다.

```
v.removeAllElements();
```

벡터를 생성하고 활용하는 코드와 과정을 [그림 7-7]에 자세히 표현하였다.



[그림 7-7] Vector<Integer> 생성 및 삽입 삭제 사례

컬렉션과 자동 박싱/언박싱

컬렉션은 객체들만 요소(element)로 다룬다고 설명하였다. 그러므로 기본 타입의 값은 Wrapper 클래스로 객체화하여 삽입한다. 다음은 정수를 Integer 객체로 만들어 저장하는 사례이다.

```
Vector<Integer> v = new Vector<Integer>();
v.add(new Integer(4));
v.add(new Integer(-1));
```

그러나 자동 박싱(auto boxing)에 의해 int 타입을 값을 사용하면 자동으로 Integer 객체로 변환되어 삽입된다. 자동 박싱을 이용하여 위 코드를 수정하면 다음과 같다.

자동 박싱

```
v.add(4);  // 정수 4가 Integer(4)로 자동 박싱됨
v.add(-1); // 정수 -10| Integer(-1)로 자동 박싱됨
```

컬렉션으로부터 값을 얻어내는 과정에서는 자동 언박싱(auto unboxing)이 일어난다. 다음 코드는 자동 언박싱을 활용하여 벡터로부터 정수 값을 바로 얻어내는 코드이다.

자동 언박싱

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

자동 박싱/언박싱은 모든 컬렉션 클래스에서 작동한다.





예제 7-1 정수만 다루는 Vector<Integer> 컬렉션 활용

정수만 다루는 벡터를 생성하고, 활용하는 기본 사례를 보인다.

```
1  import java.util.Vector;
2
3  public class VectorEx {
4     public static void main(String[] args) {
5        Vector<Integer> v = new Vector<Integer>(); // 정수 값만 다루는 벡터 생성
6        v.add(5); // 5 삽입
7        v.add(4); // 4 삽입
8        v.add(-1); // -1 삽입
9
10       // 벡터 중간에 삽입하기
11       v.add(2, 100); // 4와 -1 사이에 정수 100 삽입
12
13       System.out.println("벡터 내의 요소 객체 수 : " + v.size()); // 크기 3
14       System.out.println("벡터의 현재 용량 : " + v.capacity()); // 벡터 용량 10
15
16       // 모든 요소 정수 출력하기
17       for(int i=0; i<v.size(); i++) {
18          int n = v.get(i); // 벡터의 i 번째 정수
19          System.out.println(n);
20       }
21
22       // 벡터 속의 모든 정수 더하기
23       int sum = 0;
24       for(int i=0; i<v.size(); i++) {
25          int n = v.elementAt(i); // 벡터의 i 번째 정수
26          sum += n;
27       }
28       System.out.println("벡터에 있는 정수 합 : " + sum);
29    }
30 }
```

실행 결과

```
벡터 내의 요소 객체 수 : 4
벡터의 현재 용량 : 10
5
4
100
-1
벡터에 있는 정수 합 : 108
```

예제 7-2 Point 클래스만 다루는 Vector<Point> 컬렉션 활용



점 (x, y)를 표현하는 Point 클래스를 만들고, Point의 객체만 다루는 벡터를 작성하라.

```
1  import java.util.Vector;
2
3  class Point {
4     private int x, y;
5     public Point(int x, int y) {
6        this.x = x; this.y = y;
7     }
8     public String toString() {
9        return "(" + x + "," + y + ")";
10    }
11 }
12
13 public class PointVectorEx {
14    public static void main(String[] args) {
15       Vector<Point> v = new Vector<Point>(); // Point 객체를 요소로 다루는 벡터 생성
16
17       // 3 개의 Point 객체 삽입
18       v.add(new Point(2, 3));
19       v.add(new Point(-5, 20));
20       v.add(new Point(30, -8));
21
22       v.remove(1); // 인덱스 1의 Point(-5, 20) 객체 삭제
23
24       // 벡터에 있는 Point 객체 모두 검색하여 출력
25       for(int i=0; i<v.size(); i++) {
26          Point p = v.get(i); // 벡터의 i 번째 Point 객체 얻어내기
27          System.out.println(p); // p.toString()을 이용하여 객체 p 출력
28       }
29    }
30 }
```

실행 결과

```
(2,3)
(30,-8)
```

컬렉션을 매개변수로 받는 메소드 만들기

컬렉션을 매개변수로 전달받는 메소드를 선언하는 법을 알아보자. Integer 벡터를 매개변수로 받아 원소를 모두 출력하는 printVector() 메소드는 다음과 같이 작성하며,

```
public void printVector(Vector<Integer> v) {
   for(int i=0; i<v.size(); i++) {
       int n = v.get(i); // 벡터의 i 번째 정수
       System.out.println(n);
   }
}
```

이 메소드를 호출하는 코드는 다음과 같이 작성한다.

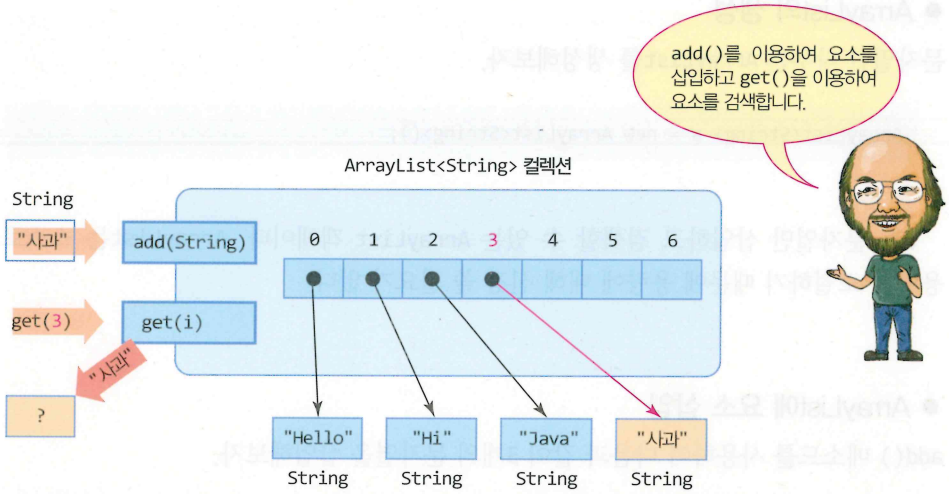
```
Vector<Integer> v = new Vector<Integer>(); // Integer 벡터 생성
printVector(v); // 메소드 호출
```

ArrayList〈E〉

ArrayList<E>

ArrayList<E>(이하 ArrayList)는 가변 크기의 배열을 구현한 컬렉션 클래스로서 정로명은 java.util.ArrayList이며, Vector 클래스와 거의 동일하다. 크게 다른 점은 ArrayList는 스레드 간에 동기화를 지원하지 않기 때문에, 다수의 스레드가 동시에 ArrayList에 요소를 삽입하거나 삭제할 때 ArrayList의 데이터가 훼손될 우려가 있다. 하지만 멀티스레드 동기화를 위한 시간 소모가 없기 때문에, ArrayList는 Vector보다 속도가 빨라, 단일 스레드 응용에는 더 효과적이다.

ArrayList는 Vector와 마찬가지로 요소의 삽입이나 삭제 시 중간에 빈 공간이 생기지 않도록 요소들의 위치를 앞뒤로 이동시킨다. [그림 7-8]은 ArrayList 객체의 내부 구성을 보여준다. 내부에 배열을 가지고 있으며 이 배열을 가변 크기로 관리한다. ArrayList는 인덱스로 요소를 접근할 수 있으며, 인덱스는 0부터 시작한다. 〈표 7-2〉는 ArrayList 클래스의 주요 메소드를 보여준다.



[그림 7-8] ArrayList al = new ArrayList<String>() 객체의 내부 구성

〈표 7-2〉 ArrayList<E> 클래스의 메소드

메소드	설명
boolean add(E element)	ArrayList의 맨 뒤에 element 추가
void add(int index, E element)	인덱스 index 위치에 element 삽입
boolean addAll(Collection<? extends E> c)	컬렉션 c의 모든 요소를 ArrayList의 맨 뒤에 추가
void clear()	ArrayList의 모든 요소 삭제
boolean contains(Object o)	ArrayList가 지정된 객체를 포함하고 있으면 true 리턴
E elementAt(int index)	index 인덱스의 요소 리턴
E get(int index)	index 인덱스의 요소 리턴
int indexOf(Object o)	o와 같은 첫 번째 요소의 인덱스 리턴, 없으면 -1 리턴
boolean isEmpty()	ArrayList가 비어있으면 true 리턴
E remove(int index)	index 인덱스의 요소 삭제
boolean remove(Object o)	o와 같은 첫 번째 요소를 ArrayList에서 삭제
int size()	ArrayList가 포함하는 요소의 개수 리턴
Object[] toArray()	ArrayList의 모든 요소를 포함하는 배열 리턴

● ArrayList의 생성

문자열만 다루는 ArrayList를 생성해보자.

```
ArrayList<String> a = new ArrayList<String>();
```

a는 문자열만 삽입하고 검색할 수 있는 ArrayList 객체이다. ArrayList는 스스로 용량을 조절하기 때문에 용량에 대해 신경 쓸 필요가 없다.

● ArrayList에 요소 삽입

add()

add() 메소드를 사용하여 다음과 같이 3개의 문자열을 삽입해보자.

```
a.add("Hello");
a.add("Hi");
a.add("Java");
```

a에 삽입할 수 있는 요소는 String 타입의 문자열만 가능하다. a에 문자열이 아닌 객체나 값을 삽입하면 다음과 같이 오류가 발생한다.

오류

```
a.add(5); // 컴파일 오류. 정수 삽입 불가
a.add(new Point(3,5)); // 컴파일 오류. Point 객체 삽입 불가
```

null

ArrayList에도 Vector와 마찬가지로 null을 삽입할 수 있다.

```
a.add(null);
```

add() 메소드를 이용하면 ArrayList의 중간에 요소를 삽입할 수 있다. 다음은 인덱스 2의 위치에 "Sahni"를 삽입하는 코드이다.

```
a.add(2, "Sahni");
```

"Sahni"를 인덱스 2에 삽입하고 기존의 인덱스 2와 그 뒤에 있는 요소들을 한 자리씩 뒤로 이동시킨다. 하지만, 이 코드의 실행 전, ArrayList에 들어 있는 요소의 개수가 2보다 작으면 예외가 발생한다. add()는 끝이나 중간에만 요소를 삽입한다.

● ArrayList 내의 요소 알아내기

get()이나 elementAt() 메소드를 이용하면 ArrayList 내의 요소를 알아낼 수 있다. 다음 코드는 인덱스 1의 위치에 있는 요소를 리턴한다.

get()
elementAt()

```
String str = a.get(1); // "Hi" 리턴
```

● ArrayList의 크기 알아내기

size() 메소드를 호출하면 현재 ArrayList에 들어 있는 요소의 개수를 알아낼 수 있다.

size()

```
int len = a.size(); // ArrayList에 들어 있는 요소의 개수
```

ArrayList는 벡터와는 달리 현재 용량을 리턴하는 메소드가 없다.

● ArrayList에서 요소 삭제

remove() 메소드를 이용하면 ArrayList 내 임의의 인덱스에 있는 요소를 삭제할 수 있다. 다음 코드는 인덱스 1의 위치에 있는 요소를 삭제한다. 이 결과 뒤에 있는 요소들이 한 자리씩 앞으로 이동한다.

remove()

```
a.remove(1); // 인덱스 1의 위치에 있는 요소 삭제
```

다음과 같이 객체 레퍼런스를 이용하여 remove()를 호출할 수도 있다.

```
String s = new String("bye");
a.add(s);
...
a.remove(s); // a에서 문자열 s 삭제
```

ArrayList에 있는 모든 요소를 삭제하려면 다음과 같이 clear()를 호출한다.

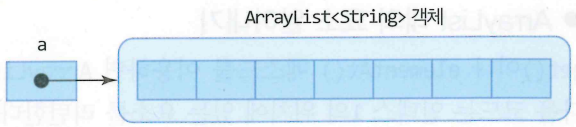
clear()

```
a.clear();
```

ArrayList 컬렉션의 생성, 요소 삽입, 삭제 등 ArrayList를 다루는 과정을 [그림 7-9]에 표현하였다.

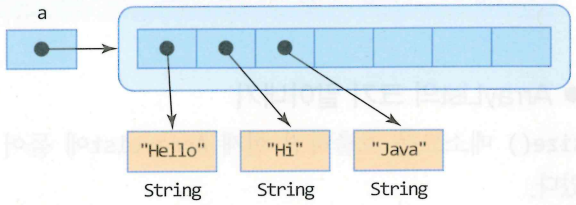
ArrayList 생성

```
ArrayList<String> a = new ArrayList<String>(7);
```



요소 삽입

```
a.add("Hello");
a.add("Hi");
a.add("Java");
```



요소 개수 n

```
int n = a.size(); // n은 3
```

용량

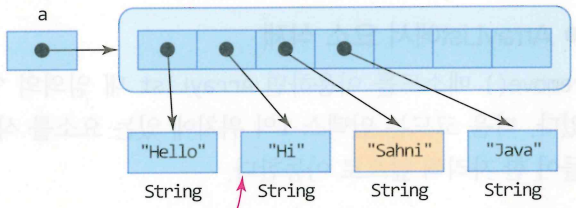
~~`int c = a.capacity();`~~ // capacity() 메소드 없음 (오류)

n = 3

요소 중간 삽입

```
a.add(2, "Sahni");
```

~~`a.add(5, "Sahni");`~~ (오류)
// a.size()보다 큰 위치에 삽입 불가능, 오류



요소 알아내기

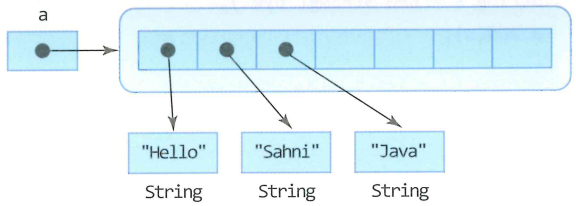
```
String str = a.get(1);
```



요소 삭제

```
a.remove(1);
```

~~`a.remove(4);`~~ // 오류 (오류)



모든 요소 삭제

```
a.clear();
```



[그림 7-9] ArrayList<String>의 생성 요소 삽입 삭제 사례

문자열 입력받아 ArrayList에 저장 예제 7-3



이름을 4개 입력받아 ArrayList에 저장하고 모두 출력한 후 제일 긴 이름을 출력하라.

```
1  import java.util.*;
2
3  public class ArrayListEx {
4     public static void main(String[] args) {
5        // 문자열만 삽입 가능한 ArrayList 생성
6        ArrayList<String> a = new ArrayList<String>();
7
8        // 키보드로부터 4개의 이름 입력받아 ArrayList에 삽입
9        Scanner scanner = new Scanner(System.in); // Scanner 객체 생성
10       for(int i=0; i<4; i++) {
11          System.out.print("이름을 입력하세요>>");
12          String s = scanner.next(); // 키보드로부터 이름을 입력
13          a.add(s); // ArrayList 컬렉션에 삽입
14       }
15
16       // ArrayList에 들어 있는 모든 이름 출력
17       for(int i=0; i<a.size(); i++) {
18          String name = a.get(i); // ArrayList의 i 번째 문자열 얻어오기
19          System.out.print(name + " ");
20       }
21
22       // 가장 긴 이름 출력
23       int longestIndex = 0; // 현재 가장 긴 이름이 있는 ArrayList 내의 인덱스
24       for(int i=1; i<a.size(); i++) {
25          if(a.get(longestIndex).length() < a.get(i).length()) // 이름 길이 비교
26             longestIndex = i; // i 번째 이름이 더 긴 이름임
27       }
28       System.out.println("\n가장 긴 이름은 : " + a.get(longestIndex));
29       scanner.close();
30    }
31 }
```

실행 결과

```
이름을 입력하세요>>Mike
이름을 입력하세요>>Jane
이름을 입력하세요>>Ashley
이름을 입력하세요>>Helen
Mike Jane Ashley Helen
가장 긴 이름은 : Ashley
```

컬렉션의 순차 검색을 위한 Iterator

순차 검색
Iterator<E> 인터페이스

Vector, ArrayList, LinkedList, Set과 같이 요소가 순서대로 저장된 컬렉션에서 요소를 순차적으로 검색할 때는 java.util 패키지의 Iterator<E> 인터페이스를 사용하면 편리하다. 여기서 <E>에는 컬렉션의 매개변수와 동일한 타입을 지정해야 한다. Iterator<E>의 메소드는 〈표 7-3〉과 같이 간단하지만 매우 강력하다.

예를 들어, 다음 벡터가 있다고 하자.

```
Vector<Integer> v = new Vector<Integer>(); // 요소가 Integer 타입인 벡터
```

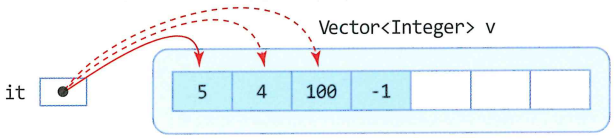
iterator()

다음과 같이 벡터 v의 iterator()를 호출하여, 벡터 v의 각 요소를 순차적으로 검색할 수 있는 Iterator 객체를 얻어낸다. Iterator 객체를 반복자라고 부른다.

```
Iterator<Integer> it = v.iterator(); // 벡터 v의 요소를 순차 검색할 Iterator 객체 리턴
```

벡터 v의 요소 타입(Integer)에 맞추어 Iterator<E>의 <E>에 Integer를 지정하였다. 이제, it 객체를 이용하면 벡터의 각 요소를 순차 검색할 수 있다. [그림 7-10]은 벡터 v와 it의 관계를 보여준다. 다음은 it로 v의 각 요소들을 순차 방문하는 코드이며 처음 it.next()는 v의 첫 번째 요소를 리턴하고, it는 다음 요소를 가리킨다.

```
while(it.hasNext()) { // it로 벡터의 끝까지 반복
    int n = it.next(); // it가 가리키는 요소 리턴. it의 요소 타입은 Integer이므로 정수 리턴
    ....
}
```



[그림 7-10] Vector<Integer>객체와 Iterator 객체의 관계

〈표 7-3〉
Iterator<E> 인터페이스의 메소드

메소드	설명
boolean hasNext()	방문할 요소가 남아 있으면 true 리턴
E next()	다음 요소 리턴
void remove()	마지막으로 리턴된 요소 제거

Iterator를 이용하여 Vector 속의 모든 요소를 출력하고 합 구하기 예제 7-4

예제 7-1의 코드를 Iterator<Integer>를 이용하여 수정하라.

```
1  import java.util.*;
2
3  public class IteratorEx {
4     public static void main(String[] args) {
5        // 정수 값만 다루는 제네릭 벡터 생성
6        Vector<Integer> v = new Vector<Integer>();
7        v.add(5); // 5 삽입
8        v.add(4); // 4 삽입
9        v.add(-1); // -1 삽입
10       v.add(2, 100); // 4와 -1 사이에 정수 100 삽입
11
12       // Iterator를 이용한 모든 정수 출력하기
13       Iterator<Integer> it = v.iterator();  // Iterator 객체 얻기
14       while(it.hasNext()) {
15          int n = it.next();
16          System.out.println(n);
17       }
18
19       // Iterator를 이용하여 모든 정수 더하기
20       int sum = 0;
21       it = v.iterator();  // it 다시 설정
22       while(it.hasNext()) {
23          int n = it.next();
24          sum += n;
25       }
26       System.out.println("벡터에 있는 정수 합 : " + sum);
27    }
28 }
```

while 문은 벡터 v의 모든 정수 출력

while 문은 벡터 v의 모든 정수합 계산

실행 결과

```
5
4
100
-1
벡터에 있는 정수 합 : 108
```

HashMap<K, V>

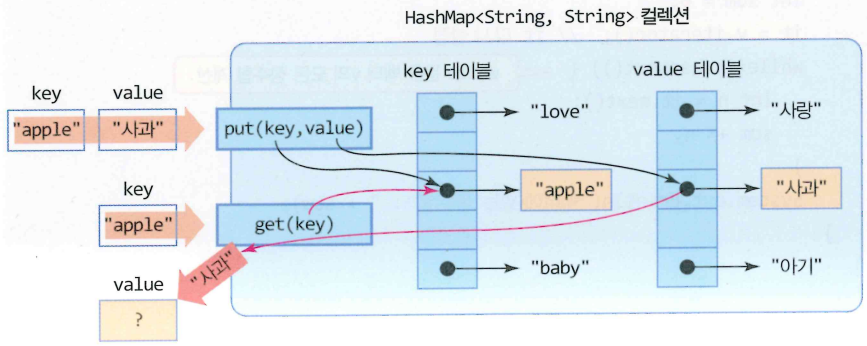
HashMap<K, V>
키(key)
값(value)

HashMap<K, V> 컬렉션은 경로명이 java.util.HashMap이며, '키(key)'와 '값(value)'의 쌍으로 구성되는 요소를 다룬다. K는 '키'로 사용할 데이터 타입을, V는 '값'으로 사용할 데이터 타입의 타입매개변수이다. HashMap(이하 해시맵) 객체의 내부 구성은 [그림 7-11]과 같다. 해시맵은 내부에 '키'와 '값'을 저장하는 자료 구조를 가지고, 다음과 같이 put(), get() 메소드를 이용하여 요소를 삽입하거나 검색한다.

```
HashMap<String, String> h = new HashMap<String, String>(); // 해시맵 생성
h.put("apple", "사과");              // "apple" 키와 "사과" 값의 쌍을 h에 삽입
String kor = h.get("apple");         // "apple" 키로 값 검색. kor는 검색된 값, "사과"
```

put(key, value)
get(key)

[그림 7-11]에서 put(key, value) 메소드는 '키'와 '값'을 받아, '키'를 이용하여 해시 함수(hash function)를 실행하고 해시 함수가 리턴하는 위치에 '키'와 '값'을 저장한다. 반대로 get(key)은 다시 '키'를 이용하여 동일한 해시 함수를 실행하여 '값'이 저장된 위치를 알아내어 '값'을 리턴한다. 해시맵은 해시 함수를 통해 '키'와 '값'이 저장되는 위치를 결정하므로, 사용자는 그 위치를 알 수 없고, 삽입되는 순서와 들어 있는 위치 또한 관계가 없다. HashMap 컬렉션의 주요 메소드는 〈표 7-4〉와 같다.



[그림 7-11] HashMap<String, String> 컬렉션의 내부 구성과 put(), get() 메소드

● 해시맵의 장단점

해시맵은 List<E> 인터페이스를 상속받은 Vector<E>나 ArrayList<E>와는 다른 점이 있다.

삽입, 삭제 매우 빠르다

첫째, 요소의 삽입, 삭제 시간이 매우 빠르다. 요소의 위치를 결정하는 해시 함수가 간단한 코드로 이루어지며, Vector<E>나 ArrayList<E>와 달리 요소의 삽입 삭제 시 다른 요소들의 위치 이동이 필요 없기 때문이다.

둘째, 요소 검색은 더욱 빠르다. 해시맵의 get(key) 메소드가 호출되면 해시 함수가 key가 저장된 위치를 단번에 찾아내므로, Vector<E>나 ArrayList<E>에서처럼 모든 요소들을 하나씩 비교하는 시간 낭비가 전혀 없기 때문이다.

검색 빠르다

셋째, 하지만 해시맵은 인덱스를 이용하여 요소에 접근할 수 없고 오직 '키'로만 검색해야 한다. 그러므로 해시맵은 빠른 삽입과 검색이 필요한 응용에 적합하다.

'키'로만 검색

〈표 7-4〉 HashMap<K, V> 클래스의 주요 메소드

메소드	설명
void clear()	해시맵의 모든 요소 삭제
boolean containsKey(Object key)	지정된 키(key)를 포함하고 있으면 true 리턴
boolean containsValue(Object value)	지정된 값(value)에 일치하는 키가 있으면 true 리턴
V get(Object key)	지정된 키(key)의 값 리턴. 키가 없으면 null 리턴
boolean isEmpty()	해시맵이 비어 있으면 true 리턴
Set<K> keySet()	해시맵의 모든 키를 담은 Set<K> 컬렉션 리턴
V put(K key, V value)	key와 value 쌍을 해시맵에 저장
V remove(Object key)	지정된 키(key)를 찾아 키와 값 모두 삭제
int size()	HashMap에 포함된 요소의 개수 리턴

● 해시맵 생성

해시맵은 HashMap<K, V>에서 K에는 '키'로, V에는 '값'으로 사용할 구체적인 타입을 지정하여 생성한다. 다음은 (영어, 한글) 단어 사전을 만들기 위해 K와 V를 모두 String 타입으로 지정한 사례이다.

```
HashMap<String, String> h = new HashMap<String, String>();
```

● 해시맵에 요소 삽입

요소를 삽입할 때는 put() 메소드에 '키'와 '값'을 전달한다.

put()

```
h.put("baby", "아기"); // "baby"는 키, "아기"는 값
h.put("love", "사랑");
h.put("apple", "사과");
```

● '키'로 '값' 읽기

다음과 같이 get() 메소드에 '키'를 전달하면, '값'을 얻을 수 있다.

```
String kor1 = h.get("love"); // kor1 = "사랑"
```

get()

만일, 해시맵에 없는 '키'로 get()을 호출하면 null을 리턴한다. 다음과 같이 "babo"라는 문자열은 현재 h에 없는 '키'이므로 get("babo")는 null을 리턴한다.

```
String kor3 = h.get("babo"); // kor3 = null
```

● '키'로 요소 삭제

remove()

'키'를 이용하여 요소를 삭제할 때 다음과 같이 remove() 메소드를 이용한다.

```
h.remove("apple"); // put("apple", "사과")로 삽입한 요소 삭제
```

● 요소 개수 알아내기

size()

요소의 개수는 다음과 같이 size() 메소드를 호출하면 된다.

```
int n = h.size(); // 현재 h 내에 있는 요소의 개수 리턴
```

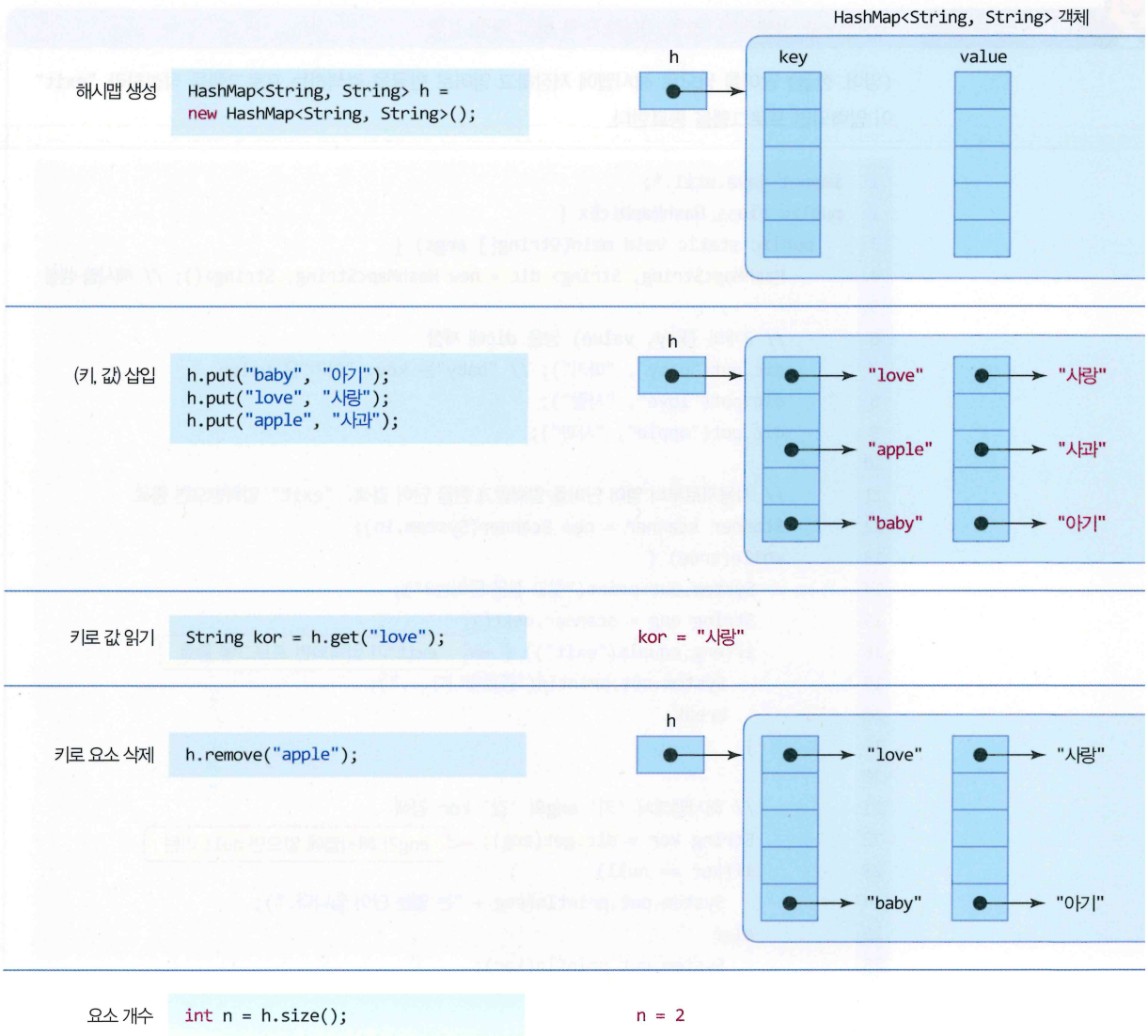
[그림 7-12]는 HashMap<String, String> 컬렉션의 생성, 입력, 검색 등의 과정을 보여준다. 그림에서 볼 수 있듯이 요소들은 해시맵에 삽입된 순서로 저장되지 않는다.

● 해시맵의 전체 검색

keySet()

해시맵에 들어 있는 요소들을 모두 알아내어 보자. 해시맵의 모든 '키'를 알아낸 후, 각 '키'에 대해 하나씩 '값'을 알아내는 방식으로 작성하면 된다. HashMap의 keySet() 메소드는 모든 '키'를 Set 컬렉션으로 만들어 리턴한다. 코드는 다음과 같다.

```
Set<String> keys = h.keySet(); // 해시맵 h에 있는 모든 키를 Set 컬렉션으로 리턴
Iterator<String> it = keys.iterator(); // Set의 각 문자열을 순차 검색하는 Iterator 리턴
while(it.hasNext()) {
    String key = it.next(); // 키
    String value = h.get(key); // 값
    System.out.println("(" + key + "," + value + ")"); // 키와 값의 쌍 출력
}
```



[그림 7-12] HashMap<String, String> 컬렉션의 생성 및 삽입 삭제 사례





예제 7-5 HashMap을 이용하여 (영어, 한글) 단어 쌍의 저장 검색

(영어, 한글) 단어를 쌍으로 해시맵에 저장하고 영어로 한글을 검색하는 프로그램을 작성하라. "exit"이 입력되면 프로그램을 종료한다.

```
1  import java.util.*;
2  public class HashMapDicEx {
3     public static void main(String[] args) {
4        HashMap<String, String> dic = new HashMap<String, String>(); // 해시맵 생성
5
6        // 3개의 (key, value) 쌍을 dic에 저장
7        dic.put("baby", "아기"); // "baby"는 key, "아기"은 value
8        dic.put("love", "사랑");
9        dic.put("apple", "사과");
10
11       // 사용자로부터 영어 단어를 입력받고 한글 단어 검색. "exit" 입력받으면 종료
12       Scanner scanner = new Scanner(System.in);
13       while(true) {
14          System.out.print("찾고 싶은 단어는?");
15          String eng = scanner.next();
16          if(eng.equals("exit")) {          "exit"가 입력되면 프로그램 종료
17             System.out.println("종료합니다...");
18             break;
19          }
20
21          // 해시맵에서 '키' eng의 '값' kor 검색
22          String kor = dic.get(eng);        eng가 해시맵에 없으면 null 리턴
23          if(kor == null)
24             System.out.println(eng + "는 없는 단어 입니다.");
25          else
26             System.out.println(kor);
27       }
28       scanner.close();
29    }
30 }
```

실행 결과

```
찾고 싶은 단어는?apple
사과
찾고 싶은 단어는?babo
babo는 없는 단어 입니다.
찾고 싶은 단어는?exit
종료합니다...
```

HashMap을 이용하여 자바 과목의 이름과 점수 관리 예제 7-6



해시맵을 이용하여 학생의 이름과 자바 점수를 기록 관리하는 프로그램을 작성하라.

```
1  import java.util.*;
2  public class HashMapScoreEx {
3     public static void main(String[] args) {
4        // 이름과 점수를 저장할 HashMap 컬렉션 생성
5        HashMap<String, Integer> scoreMap = new HashMap<String, Integer>();
6
7        // 5개의 점수 저장
8        scoreMap.put("김성동", 97);
9        scoreMap.put("황기태", 88);
10       scoreMap.put("김남윤", 98);
11       scoreMap.put("이재문", 70);
12       scoreMap.put("한원선", 99);
13
14       System.out.println("HashMap의 요소 개수 :" + scoreMap.size());
15
16       // 모든 사람의 점수 출력. scoreMap에 들어 있는 모든 (key, value) 쌍 출력
17       Set<String> keys = scoreMap.keySet();  // 모든 key를 가진 Set 컬렉션 리턴
18       Iterator<String> it = keys.iterator();  // Set에 있는 모든 key를 순서대로 검
                                                    색하는 Iterator 리턴
19       while(it.hasNext()) {
20          String name = it.next();          // 다음 키. 학생 이름
21          int score = scoreMap.get(name);  // 점수 알아내기
22          System.out.println(name + " : " + score);
23       }
24    }
25 }
```

이름은 String, 점수는 Integer로 구성된 해시맵

while 문은 모든 점수 출력

실행 결과

```
HashMap의 요소 개수 :5
이재문 : 70
한원선 : 99
김남윤 : 98
김성동 : 97
황기태 : 88
```

출력된 결과는 삽입된 순서와 다르다는 점을 기억하기 바란다.



예제 7-7 HashMap에 객체 저장, 학생 정보 관리

id와 tel(전화번호)로 구성되는 Student 클래스를 만들고, 이름을 '키'로 하고 Student 객체를 '값'으로 하는 해시맵을 작성하라.

이름과 Student 객체를 쌍으로 저장

```
1  import java.util.*;
2  class Student {
3     private int id;
4     private String tel;
5     public Student(int id, String tel) { this.id = id; this.tel = tel; }
6     public int getId() { return id; }
7     public String getTel() { return tel; }
8  }
9
10 public class HashMapStudentEx {
11    public static void main(String[] args) {
12       // (학생 이름, Student 객체)를 저장하는 해시맵 생성
13       HashMap<String, Student> map = new HashMap<String, Student>();
14       map.put("황기태", new Student(1, "010-111-1111")); // 3명의 학생 저장
15       map.put("이재문", new Student(2, "010-222-2222"));
16       map.put("김남윤", new Student(3, "010-333-3333"));
17
18       Scanner scanner = new Scanner(System.in);
19       while(true) {
20          System.out.print("검색할 이름?");
21          String name = scanner.nextLine(); // 사용자로부터 이름 입력
22          if(name.equals("exit"))
23             break; // while 문을 벗어나 프로그램 종료
24          Student student = map.get(name);  // 이름에 해당하는 Student 객체 검색
25          if(student == null)
26             System.out.println(name + "은 없는 사람입니다.");
27          else
28             System.out.println("id:" + student.getId() + ", 전화:" + student.getTel());
29       }
30       scanner.close();
31    }
32 }
```

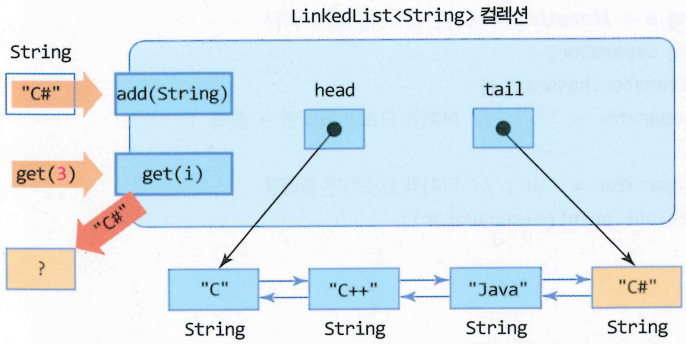
실행 결과

```
검색할 이름?이재문
id:2, 전화:010-222-2222
검색할 이름?김남윤
id:3, 전화:010-333-3333
검색할 이름?
```

LinkedList<E>

LinkedList<E>

LinkedList<E>는 List<E> 인터페이스를 구현한 클래스로서 경로명이 java.util.LinkedList이다. LinkedList는 요소들을 양방향으로 연결하여 관리한다는 점을 제외하고 Vector, ArrayList와 거의 같다. [그림 7-13]은 add() 메소드와 get() 메소드를 가진 LinkedList 객체의 내부 구성을 보여준다. LinkedList는 맨 앞과 맨 뒤를 가리키는 head, tail 레퍼런스를 가지고 있어, 맨 앞이나 맨 뒤, 중간에 요소의 삽입이 가능하며 인덱스를 이용하여 요소에 접근할 수도 있다.



[그림 7-13] LinkedList<String>의 내부 구성과 add(), get() 메소드

Collections 클래스 활용

Collections

java.util 패키지에 포함된 Collections 클래스는 다음과 같이 컬렉션을 다루는 유용한 여러 메소드를 지원한다.

- sort() - 컬렉션에 포함된 요소들의 정렬
- reverse() - 요소를 반대 순으로 정렬
- max(), min() - 요소들의 최댓값과 최솟값 찾아내기
- binarySearch() - 이진 검색

static 타입

Collections 클래스의 메소드는 모두 static 타입이므로 Collections 객체를 생성할 필요는 없다. 이 유틸리티 메소드들은 인자로 컬렉션 객체를 전달받아 처리하며, 예제 7-8은 이들을 활용하는 사례를 보여준다. 예제에서 sort(), reverse(), binarySearch() 메소드의 활용을 주목하여 보기 바란다.



예제 7-8 Collections 클래스의 활용

Collections 클래스를 활용하여 문자열 정렬, 반대로 정렬, 이진 검색을 실행하는 사례를 살펴보자.

```
1  import java.util.*;
2
3  public class CollectionsEx {
4      static void printList(LinkedList<String> l) { // 리스트의 요소를 모두 출력하
                                                       는 메소드
5          Iterator<String> iterator = l.iterator(); // Iterator 객체 리턴
6          while (iterator.hasNext()) { // Iterator 객체에 요소가 있을 때까지 반복
7              String e = iterator.next(); // 다음 요소 리턴
8              String separator;
9              if (iterator.hasNext())
10                 separator = "->"; // 마지막 요소가 아니면 → 출력
11             else
12                 separator = "\n"; // 마지막 요소이면 줄바꿈
13             System.out.print(e+separator);
14         }
15     }
16
17     public static void main(String[] args) {
18         LinkedList<String> myList = new LinkedList<String>(); // 빈 리스트 생성
19         myList.add("트랜스포머");
20         myList.add("스타워즈");
21         myList.add("매트릭스");
22         myList.add(0,"터미네이터");
23         myList.add(2,"아바타");
24
25         Collections.sort(myList);  // 요소 정렬
26         printList(myList); // 정렬된 요소 출력
27
28         Collections.reverse(myList);  // 요소의 순서를 반대로 구성
29         printList(myList); // 요소 출력
30
31         int index = Collections.binarySearch(myList, "아바타") + 1;
32         System.out.println("아바타는 " + index + "번째 요소입니다.");
33     }
34 }
```

sort()는 static 메소드이므로 클래스 이름으로 바로 호출한다.

binarySearch()를 이용하여 "아바타"의 인덱스 검색

실행 결과

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

소팅된 순서대로 출력

거꾸로 출력



잠깐!

본래 Collections 클래스는 java.lang.Comparable을 상속받는 element에 대해서만 작동한다. int, char, double 등의 기본 타입과 String 클래스는 이 조건을 충족하지만, 사용자가 클래스를 작성하는 경우 java.lang.Comparable을 상속받아야 한다.



1 다음 코드를 실행할 때 인덱스 2에 있는 요소는?

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(1);
a.add(2);
a.add(3);
a.add(4);
```

2 다음 코드에서 컴파일 오류가 발생하는 라인은?

```
Vector<Integer> v = new Vector<Integer>();
v.add(5);                  // ①
v.add(new Integer(10));    // ②
v.add("100");              // ③
int n = v.get(0);          // ④
```

3 다음 빈칸에 적절한 코드를 삽입하라.

```
_____________ m = new _____________ ;
m.put(10, "ten");
String val = m.get(10);
```

4 다음은 Iterator를 이용하여 벡터에 삽입된 모든 요소를 출력하는 프로그램이다. 빈칸에 적절한 코드를 삽입하라.

```
Vector<Integer> v = new Vector<Integer>();
Iterator<_____________> it = v._____________; // Iterator 객체를 얻어온다.
while(it._____________) { // 요소를 모두 방문할 때까지 루프
   System.out.println(it._____________); // 다음 요소 출력
}
```

7.3 제네릭 만들기

지금까지 JDK에 제네릭으로 구현된 컬렉션을 사용하는 방법을 알아보았다. 이 절에서는 독자가 제네릭 클래스를 만드는 과정에 대해 설명한다.

제네릭 클래스

제네릭 클래스

제네릭 클래스를 작성하는 방법은 기존의 클래스 작성 방법과 유사한데, 클래스 이름 다음에 일반화된 타입(generic type)의 매개변수를 <와 > 사이에 추가한다는 차이가 있다. 간단한 제네릭 클래스를 작성해보자.

● 제네릭 클래스 작성

타입 매개변수 T를 가진 제네릭 클래스 MyClass는 다음과 같이 작성한다.

```
public class MyClass<T> { // 제네릭 클래스 Myclass, 타입 매개변수 T
    T val; // 변수 val의 타입은 T
    void set(T a) {
        val = a; // T 타입의 값 a를 val에 지정
    }
    T get() {
        return val; // T 타입의 값 val 리턴
    }
}
```

● 제네릭 클래스에 대한 레퍼런스 변수 선언

제네릭 클래스의 레퍼런스 변수를 선언할 때 다음과 같이 타입 매개변수에 구체적인 타입을 적는다.

```
MyClass<String> s;  // <T>를 String으로 구체화
List<Integer> li;   // <E>를 Integer로 구체화
Vector<String> vs;  // <E>를 String으로 구체화
```

● 제네릭 객체 생성 – 구체화(specialization)

구체화

제네릭 클래스에 구체적인 타입을 대입하여 구체적인 객체을 생성하는 과정을 구체화라고 부르며, 이 과정은 자바 컴파일러에 의해 이루어진다. MyClass<T>에서 T에 구체적인 타입을 지정하여 객체를 생성하는 예는 다음과 같다.

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

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

MyClass<String>은 String만 다루는 구체적인 클래스가 되며, MyClass<Integer>는 정수만 다루는 구체적인 클래스가 된다. 구체화된 MyClass<String>의 코드는 [그림 7-14]와 같다.

```
public class MyClass<String> {
    String val; // 변수 val의 타입은 String
    void set(String a) {
        val = a; // String 타입의 문자열 a를 val에 지정
    }
    String get() {
        return val; // String 타입의 문자열 val 리턴
    }
}
```

[그림 7-14] 컴파일러에 의해 String으로 구체화된 MyClass<String>

제네릭의 구체화에는 기본 타입은 사용할 수 없음에 유의해야 한다.

오류
```
Vector<int> vi = new Vector<int>();  // 컴파일 오류. int는 사용 불가
Vector<Integer> vi = new Vector<Integer>();  // 정상 코드
```

● 타입 매개변수

제네릭 클래스 내에서 제네릭 타입을 가진 객체의 생성은 허용되지 않는다. 예를 들면 다음 코드는 허용되지 않는다.

```
public class MyVector<E> {
    E create() {
        E a = new E(); // 컴파일 오류. 제네릭 타입의 객체 생성 불가
        return a;
    }
}
```

오류

컴파일러가 MyVector<E> 클래스의 new E() 라인을 컴파일할 때, E에 대한 구체적인 타입을 알 수 없어, 호출될 생성자를 결정할 수 없고, 또한 객체 생성 시 어떤 크기의 메모리를 할당해야 할 지 전혀 알 수 없기 때문이다.



예제 7-9 제네릭 스택 만들기

스택 자료 구조를 제네릭 클래스로 선언하고, String과 Integer 스택을 각각 사용하는 예를 보여라.

```
1  class GStack<T> { // 제네릭 스택 선언. 제네릭 타입 T
2     int tos;
3     Object [] stck; // 스택에 요소를 저장할 공간 배열
4     public GStack() {
5        tos = 0;
6        stck = new Object [10];
7     }
8     public void push(T item) {
9        if(tos == 10) // 스택이 꽉 차서 더 이상 요소를 삽입할 수 없음
10          return;
11       stck[tos] = item;
12       tos++;
13    }
14    public T pop() {
15       if(tos == 0) // 스택이 비어 있어 꺼낼 요소가 없음
16          return null;
17       tos--;
18       return (T)stck[tos]; // 타입 매개변수 타입으로 캐스팅
19    }
20 }
21
22 public class MyStack {
23    public static void main(String[] args) {
24       GStack<String> stringStack = new GStack<String>(); // String 타입의
                                                              GStack 생성
25       stringStack.push("seoul");
26       stringStack.push("busan");
27       stringStack.push("LA");
28
29       for(int n=0; n<3; n++)
30          System.out.println(stringStack.pop()); // stringStack 스택에 있는 3 개
                                                     의 문자열 팝
31
32       GStack<Integer> intStack = new GStack<Integer>(); // Integer 타입의
                                                             GStack 생성
```

제네릭 매개변수로 객체를 생성하거나 배열을 생성할 수 없으므로 Object 배열을 생성하여 실제 타입의 객체를 요소로 삽입한다.

주목

```
33        intStack.push(1);
34        intStack.push(3);
35        intStack.push(5);
36
37        for(int n=0; n<3; n++)
38           System.out.println(intStack.pop()); // intStack 스택에 있는 3 개의 정수 팝
39     }
40  }
```

실행 결과

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

예제 7-9에서 주의할 부분은 6행의 stck = new Object [10]과 18행의 return (T) stck[tos]이다. 제네릭 클래스 내에서 제네릭 타입의 객체를 생성할 수 없는 것과 같은 이유로 배열도 생성할 수 없으므로 Object의 배열로 생성하였다. 만일, 다음과 같이 쓰면 오류가 발생한다.

오류

```
stck = new T[10]; // 컴파일 오류. 제네릭에서는 T 타입의 배열을 생성할 수 없다.
```

데이터를 저장해둘 배열을 Object[] 배열로 선언하였으므로 아래와 같이 타입 매개변수의 타입으로 강제 캐스팅하여 리턴해야 한다.

```
return (T)stck[tos]; // 타입 매개변수 T 타입으로 캐스팅
```

제네릭과 배열

제네릭에서는 배열에 대한 제한을 두고 있다. 제네릭 클래스 또는 인터페이스 타입의 배열은 선언할 수 없다.

배열에 대한 제한

오류

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

그러나 제네릭 타입의 배열 선언은 허용된다.

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

제네릭 메소드

클래스의 일부 메소드만 제네릭으로 구현할 수도 있다. toStack() 메소드를 제네릭으로 구현한 예는 다음과 같다.

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

타입 매개변수는 메소드의 리턴 타입 앞에 선언된다. 위의 toStack()에서 <T>가 타입 매개변수의 선언이다. 제네릭 메소드를 호출할 때는 컴파일러가 메소드의 인자를 통해 타입을 유추할 수 있어 제네릭 클래스나 인터페이스와는 달리 타입을 명시하지 않아도 된다.

다음 코드는 컴파일러가 toStack()의 호출문으로부터 타입 매개변수 T를 Object로 유추하는 경우이다.

```
Object[] oArray = new Object[100];
GStack<Object> objectStack = new GStack<Object>();
GenericMethodEx.toStack(oArray, objectStack); // 타입 매개변수 T를 Object로 유추함
```

또 다른 경우로, 아래의 코드는 컴파일러가 toStack()의 호출문으로부터 타입 매개변수 T를 String으로 유추하는 경우이다.

```
String[] sArray = new String[100];
GStack<String> stringStack = new GStack<String>();
GenericMethodEx.toStack(sArray, stringStack); // 타입 매개변수 T를 String으로 유추함
```

다음의 경우는 타입 매개변수 T를 Object로 유추한다.

```
GenericMethodEx.toStack(sArray, objectStack);
```

여기서 sArray는 String [] 타입이며, objectStack은 GStack<Object>이다. Object가 String의 슈퍼 클래스이므로 컴파일러는 Object 타입으로 유추한다.

스택의 내용을 반대로 만드는 제네릭 메소드 만들기 예제 7-10



예제 7-9의 GStack을 이용하여 매개변수로 주어진 스택의 내용을 반대로 만드는 제네릭 메소드 reverse()를 작성하라.

```
1  public class GenericMethodEx {
2     public static <T> GStack<T> reverse(GStack<T> a) { // T가 타입 매개변수인 제네
                                                          릭 메소드
3        GStack<T> s = new GStack<T>(); // 스택 a를 반대로 저장할 목적 GStack 생성
4        while (true) {
5           T tmp;
6           tmp = a.pop(); // 원래 스택에서 요소 하나를 꺼냄
7           if (tmp == null) // 스택이 비었음
8              break; // 거꾸로 만드는 작업 종료
9           else
10             s.push(tmp); // 새 스택에 요소를 삽입
11       }
12       return s; // 새 스택을 리턴
13    }
14
15    public static void main(String[] args) {
16       GStack<Double> gs = new GStack<Double>(); // Double 타입의 GStack 생성
17
18       for (int i=0; i<5; i++) { // 5개의 요소를 스택에 push
19          gs.push(new Double(i));
20       }
21       gs = reverse(gs);
22       for (int i=0; i<5; i++) {
23          System.out.println(gs.pop());
24       }
25    }
26 }
```

컴파일러는 제네릭 메소드 reverse()의 타입 매개변수를 Double로 유추

이 예제를 실행하기 위해서는 예제 7-9의 GStack 클래스가 필요함

실행 결과

```
0.0
1.0
2.0
3.0
4.0
```

0.0 1.0 2.0 3.0 4.0 순으로 푸시된 스택의 내용을 팝하면 4.0 3.0 2.0 1.0 1.0으로 출력되어야 하지만 스택을 반대순으로 뒤집었기 때문에 0.0, 1.0 2.0 3.0 4.0 순으로 팝되었음

제네릭의 장점

제네릭은 다음과 같은 장점이 있다.

- 동적으로 타입이 결정되지 않고 컴파일 시에 타입이 결정되므로 보다 안전한 프로그래밍 가능
- 런타임 타입 충돌 문제 방지
- 개발 시 타입 캐스팅 절차 불필요
- ClassCastException 방지



1 다음 LinkedList를 생성하는 문장에서 잘못된 부분을 지적하고 이유를 설명하라.

```
LinkedList<int> myList = new LinkedList<int>();
```

2 타입 매개변수가 T이며, T 타입의 인자를 전달받고 void를 리턴하는 add 이름의 메소드와, 매개변수 없이 T 타입을 리턴하는 get 이름의 메소드를 가진 제네릭 인터페이스 MyList를 작성하라.

3 다음 코드에서 잘못된 부분을 찾고 이유를 설명하라.

```
public class MyManager<E> {
   E newElement() {
      E a = new E();
      return a;
   }
}
```

4 앞의 예제 7-10을 활용하여 예제 7-9에서 작성한 GStack의 모든 원소를 출력하는 printStack 메소드를 작성하라.

요약 SUMMARY



- 컬렉션은 요소들의 리스트나 집합을 관리하는 자료 구조로서 크기를 자동 조절하여 크기에 구애받지 않고 요소의 추가, 삭제, 검색 등을 쉽게 할 수 있어 배열보다 용이하다.
- 주요한 컬렉션 클래스에는 `Vector<E>`, `ArrayList<E>`, `HashMap<K, V>`, `LinkedList<E>`, `Stack<E>` 등이 있다.
- `Vector`나 `ArrayList` 등은 요소 객체들이 리스트 형식으로 순차적으로 저장되며, 인덱스로 검색할 수 있어 배열을 대신할 때 매우 효과적이다.
- `HashMap` 컬렉션에는 '키'와 '값'의 쌍으로 저장되고, '키'를 이용하여 요소가 검색된다. 삽입, 삭제, 검색 속도가 매우 빠른 장점이 있고, 요소의 순서가 필요 없고 '키'를 이용하여 삽입하고 검색하는 응용에 매우 적합하다.
- 컬렉션은 JDK 1.5 버전 이후 제네릭 기법으로 만들어져 있다.
- 제네릭은 일반화시킨 타입을 사용하여 특정 타입에 종속되지 않도록 클래스나 인터페이스, 메소드를 일반화시키는 기법이다. 그러므로 제네릭에 구체적인 타입을 지정하여 특정 타입으로 변신하여 사용할 수 있다.
- 컬렉션은 다음과 같이 구체적인 타입을 지정하여 사용한다.

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

- 컬렉션의 요소로는 객체들만 사용된다.
- `int`, `char`, `double` 등의 기본 타입을 컬렉션의 요소로 삽입하려면 `Wrapper` 클래스를 이용하여 기본 타입을 객체로 만들어 사용하면 된다.
- JDK 1.5부터는 자동 박싱 기능에 의해 기본 타입의 값이 컬렉션에 삽입될 때 자동으로 `Wrapper` 클래스로 포장되어 삽입되고, 추출될 때 자동 언박싱 기능에 의해 기본 타입의 값으로 자동 변환된다.
- `Collection<E>` 인터페이스를 구현한 컬렉션의 경우 `iterator()` 메소드를 호출하면 `Iterator` 객체를 리턴하며, 이 객체를 이용하여 인덱스 없이 컬렉션의 요소에 대해 순차 검색이 가능하다.
- `Collections` 클래스는 컬렉션에 대해 요소 정렬, 검색, 최대 최솟값 구하기 등 다양한 유틸리티 메소드를 제공한다.
- 제네릭을 이용하면 컴파일 시에 타입이 결정되어 불필요한 오류를 방지하므로 안전한 프로그래밍이 가능하다.
- 제네릭 클래스 또는 인터페이스 선언은 타입 매개변수를 클래스나 인터페이스 이름 다음에 추가하여 선언한다.
- 제네릭 타입으로 제네릭 클래스 내에서 객체와 배열을 생성할 수 없다.



영어 단어 테스트 프로그램 만들기

Open Challenge

목적 Vector 컬렉션 활용

영어 단어 뜻 맞추기 게임을 만들어 보자. 영어 단어와 한글 단어로 구성되는 Word 클래스를 작성하고, 프로그램 내에서 미리 여러 개의 Word 객체를 Vector<Word> 컬렉션에 삽입해둔다. 그리고 다음 결과와 같이 랜덤하게 사용자에게 문제를 던진다. 벡터 내에 정답이 아닌 단어를 랜덤하게 3개 선택하고 정답과 함께 4개의 보기를 출력한다. 난이도 7

```
"명품영어"의 단어 테스트를 시작합니다. -1을 입력하면 종료합니다.
현재 17개의 단어가 들어 있습니다.
painting?
(1)그림 (2)감정 (3)아기 (4)오류 :>1
Excellent !!
bear?
(1)사회 (2)인형 (3)감정 (4)곰 :>4
Excellent !!
eye?
(1)아기 (2)보기 (3)눈 (4)거래 :>3
Excellent !!
picture?
(1)거래 (2)보기 (3)사진 (4)인간 :>2
No. !!
bear?
(1)조각상 (2)곰 (3)인간 (4)사회 :>1
No. !!
society?
(1)조각상 (2)눈 (3)사회 (4)거래 :>3
Excellent !!
human?
(1)인간 (2)눈 (3)인형 (4)거래 :>-1   -1이 입력되면 종료
"명품영어"를 종료합니다...
```

힌트 Hint

- Word 클래스와 WordQuiz 클래스를 작성하고 WordQuiz 클래스에 main() 메소드를 둔다.
- WordQuiz 클래스에서 다음과 같이 벡터를 만들고 미리 여러 개의 Word 객체를 삽입하는 코드를 작성한다. 참고로 저자는 총 17개의 단어를 이런 방식으로 삽입해 두었다.

```
Vector<Word> v = new Vector<Word>();
v.add(new Word("love", "사랑"));
v.add(new Word("animal", "동물"));
```

연습문제 EXERCISE



이론문제

• 홀수 문제는 정답이 공개됩니다.

1. 컬렉션에 관한 설명으로 틀린 것은?

① 가변 크기라서 배열보다 사용하기 편리하다.
② 삽입 삭제 시 원소의 위치를 컬렉션이 자동 조절하므로 편하다.
③ 객체들만 삽입되므로 int와 같은 기본 타입의 값을 저장하는 방법이 없어 아쉽다.
④ 컬렉션은 모두 제네릭(generics)으로 만들어져 있다.

2. 다음 빈칸에 적절한 단어를 기입하라.

클래스, 인터페이스, 메소드를 특정 타입에 종속되지 않게 _____시켜 작성하는 기술을 _____이라 한다. Vector<E>에서 E는 제네릭 타입 혹은 _____라고 부른다. E에 Integer 등과 같이 객체 타입을 지정하여 특정 타입의 데이터만 다루도록 만드는 작업을 _____라고 한다. 제네릭 기능은 C++에서 먼저 만들어졌으며 C++에서는 _____이라고 부른다.

보기

추상화, 템플릿, 일반화, 구체화, 제네릭, 컬렉션, 상속, 제네릭 객체, 타입 매개변수, 다형성, 상수

3. 다음 코드에 대한 설명으로 틀린 것은?

```
Vector<Integer> v = new Vector<Integer>(30);
```

① Integer 타입 객체만 저장할 수 있는 구체화된 벡터를 생성하는 코드이다.
② 벡터 v는 원소를 30개만 저장할 수 있는 벡터이다.
③ v.add(10)를 호출하여 정수 10을 벡터에 삽입할 수 있다.
④ 벡터 v에는 실수 값을 삽입할 수 없다.

4. 다음 물음에 적합한 코드를 간단히 작성하라.
(1) 문자열만 저장할 수 있는 벡터 sv 생성
(2) '키'가 문자열이고 '값'이 실수인 해시맵 h 생성
(3) Person 타입의 객체를 10개 저장할 공간을 갖춘 ArrayList 컬렉션 pa 생성
(4) 나라 이름과 인구를 저장하고 관리하는 해시맵 pop 생성

5. 다음 코드에서 맞는 것은?
① Stack<String> ss;
② ArrayList<E> ar = new ArrayList<E>();
③ HashMap<String> h = new HashMap<String>();
④ Set<Integer> s = new Set<Integer>(100);

6. 다음 코드에서 자동 박싱과 자동 언박싱이 일어나는 곳을 지적하라.

```
Vector<Double> v = new Vector<Double>();
v.add(3.14);
double d = v.get(0);
```

7. 다음 프로그램은 초기에 벡터의 크기를 3으로 설정하였다. for 문을 실행하고 벗어
을 때 벡터의 용량이 어떻게 변했는지 확인하라. 벡터는 용량이 모자랄 때, 한 번에
마만큼씩 용량을 늘리는가?

```
Vector<Integer> v = new Vector<Integer>(3);      // 현재 벡터 용량은 3
for (int i=0; i<10; i++) v.add(i);               // 정수를 10개 벡터에 삽입
System.out.println(v.capacity());                // 벡터에 변한 용량은 얼마?
```

8. 주석문을 참고하여 빈칸에 적절한 코드로 채워라.

```
ArrayList<String> a = _____; // 초기 용량이 10인 ArrayList 생성
_____ // a의 맨 끝에 "Java" 삽입
_____ // a의 맨 앞에 "C++" 삽입
_____ // a에 현재 삽입된 개수 출력
_____ // a의 마지막에 있는 문자열 삭제
```

9. 다음에서 밑줄친 부분과 동일하게 작동하도록 Iterator를 이용하여 다시 작성하라.

```
Vector<Integer> v = new Vector<Integer>();
for(int i=0; i<10; i++) v.add(i);
for(int i=0; i<v.size(); i++)
    System.out.print(v.get(i) + " ");
```

10. 다음은 해시맵 h를 만들고, func(h) 메소드를 호출하여 h에 들어 있는 원소의 개수를 리턴받아 출력하는 코드이다. func() 메소드를 작성하라.

```
HashMap<String, Integer> h = new HashMap<String, Integer>();
h.put("a", 10);
h.put("b", 20);
System.out.println(func(h)); // 해시맵 h에 들어 있는 원소 개수 출력
```

11. 다음 제네릭 클래스에 대해 아래 문항에 답하라.

```
class JGeneric<W> {
    private W x, y;
    public JGeneric(W x, W y) { this.x = x; this.y = y; }

    _____________________ // (3) 여러 줄로 작성 가능

    _____________________ // (4) 여러 줄로 작성 가능

    _____________________ // (5) 여러 줄로 작성 가능
}
```

(1) JGeneric의 타입 매개변수는 몇 개이며, 무엇인가?
(2) String으로 구체화한 JGeneric 객체를 생성하는 코드를 작성하라.
(3) 첫 번째 값 x를 리턴하는 메소드 first()를 작성하라.
(4) 두 번째 값 y를 리턴하는 메소드 second()를 작성하라.
(5) equal() 메소드를 작성하라. equal()는 필드 x와 y를 비교하여 같으면 true, 아니면 false를 리턴한다.
(6) <W>를 String으로 구체화한 JGeneric 객체를 활용하는 예를 들어라.

12. 다음 질문에 답하라.

```
class JClass {
    static String take(String s[], int index) {
        if (index > s.length) {
            System.out.println("인덱스가 범위를 벗어났습니다.");
            return null;
        }
        return s[index];
    }
}
```

(1) JClass의 take() 메소드는 무엇을 처리하는 코드인가?

(2) JClass의 take() 메소드가 임의의 타입에 대해 동작하도록 제네릭 메소드로 수정하여 JGenClass를 작성하라.

(3) JGenClass를 활용하는 코드 사례를 보여라.

실습문제

• 홀수 문제는 정답이 공개됩니다.

목적 Vector<Integer> 연습

1. Scanner 클래스로 -1이 입력될 때까지 양의 정수를 입력받아 벡터에 저장하고 벡터를 검색하여 가장 큰 수를 출력하는 프로그램을 작성하라. 난이도 4

```
정수(-1이 입력될 때까지)>> 10 6 22 6 88 77 -1
가장 큰 수는 88
```

목적 ArrayList<Character>

2. Scanner 클래스를 사용하여 6개 학점('A', 'B', 'C', 'D', 'F')을 문자로 입력받아 ArrayList에 저장하고, ArrayList를 검색하여 학점을 점수(A=4.0, B=3.0, C=2.0, D=1.0, F=0)로 변환하여 평균을 출력하는 프로그램을 작성하라. 난이도 5

```
6개의 학점을 빈 칸으로 분리 입력(A/B/C/D/F)>>A C A B F D
2.3333333333333335
```

3. "그만"이 입력될 때까지 나라 이름과 인구를 입력받아 저장하고, 다시 나라 이름을 입력받아 인구를 출력하는 프로그램을 작성하라. 다음 해시맵을 이용하라. 난이도 5

목적 HashMap<Str Integer> 다루기

```
HashMap<String, Integer> nations = new HashMap<String, Integer>();
```

```
나라 이름과 인구를 10개 입력하세요.(예: Korea 5000)
나라 이름, 인구 >> Korea 5000
나라 이름, 인구 >> USA 1000000
나라 이름, 인구 >> Swiss 2000
나라 이름, 인구 >> France 3000
나라 이름, 인구 >> 그만          입력 종료
인구 검색 >> France
France의 인구는 3000
인구 검색 >> 스위스
스위스 나라는 없습니다.
인구 검색 >> 그만          프로그램 종료
```

4. Vector 컬렉션을 이용하여 강수량의 평균을 유지 관리하는 프로그램을 작성하라. 강수량을 입력하면 벡터에 추가하고 현재 입력된 모든 강수량과 평균을 출력한다.

목적 Vector<Integer>

난이도 5

```
강수량 입력 (0 입력시 종료)>> 5
5
현재 평균 5
강수량 입력 (0 입력시 종료)>> 80
5 80
현재 평균 42
강수량 입력 (0 입력시 종료)>> 53
5 80 53
현재 평균 46
강수량 입력 (0 입력시 종료)>> 22
5 80 53 22
현재 평균 40
강수량 입력 (0 입력시 종료)>> 0
```

5. 하나의 학생 정보를 나타내는 Student 클래스에는 이름, 학과, 학번, 학점 평균을 저장하는 필드가 있다.

목적 ArrayList<E>에 객체 저장 및 검색

(1) 학생마다 Student 객체를 생성하고 4명의 학생 정보를 ArrayList<Student> 컬렉션에 저장한 후에, ArrayList<Student>의 모든 학생(4명) 정보를 출력하고 학생 이름을 입력받아 해당 학생의 학점 평균을 출력하는 프로그램을 작성하라. 난이도 6

```
학생 이름,학과,학번,학점평균 입력하세요.
>> 황기태, 모바일, 1, 4.1
>> 이재문, 안드로이드, 2, 3.9
>> 김남윤, 웹공학, 3, 3.5
>> 최찬미, 빅데이터, 4, 4.25
--------------------------
이름:황기태
학과:모바일
학번:1
학점평균:4.1
--------------------------
이름:이재문
학과:안드로이드
학번:2
학점평균:3.9
--------------------------
이름:김남윤
학과:웹공학
학번:3
학점평균:3.5
--------------------------
이름:최찬미
학과:빅데이터
학번:4
학점평균:4.25
--------------------------
학생 이름 >> 최찬미
최찬미, 빅데이터, 4, 4.25
학생 이름 >> 이재문
이재문, 안드로이드, 2, 3.9
학생 이름 >> 그만
```

4명의 학생 저장됨

ArrayList<Student>의 모든 학생 (4명) 정보 출력

이름을 입력받고 학생 정보 출력

목적 동일한 프로그램을 HashMap으로 작성. 두 방법 비교

(2) ArrayList<Student> 대신, HashMap<String, Student> 해시맵을 이용하여 다시 작성하라. 해시맵에서 키는 학생 이름으로 한다. 난이도 6

6. 도시 이름, 위도, 경도 정보를 가진 Location 클래스를 작성하고, 도시 이름을 '키'로 하는 HashMap<String, Location> 컬렉션을 만들고, 사용자로부터 입력 받아 4개의 도시를 저장하라. 그리고 도시 이름으로 검색하는 프로그램을 작성하라.

목적 HashMap에 객체 연습

난이도 6

```
도시,경도,위도를 입력하세요.
>> 서울, 37, 126
>> LA, 34, -118
>> 파리, 2, 48
>> 시드니, 151, -33
---------------------------
서울   37    126
LA     34    -118
파리   2     48
시드니 151   -33
---------------------------
도시 이름 >> 피리
피리는 없습니다.
도시 이름 >> 파리
파리   2  48
도시 이름 >> 그만
```

4개의 도시를 입력받아 해시맵에 저장

해시맵에 있는 모든 도시 출력

도시 이름으로 검색

7. 이름과 학점(4.5만점)을 5개 입력받아 해시맵에 저장하고, 장학생 선발 기준을 입력받아 장학생 명단을 출력하라. 난이도 6

목적 HashMap의 전체 연습

```
미래장학금관리시스템입니다.
이름과 학점>> 적당히 3.1
이름과 학점>> 나탈락 2.4
이름과 학점>> 최고조 4.3
이름과 학점>> 상당히 3.9
이름과 학점>> 고득점 4.0
장학생 선발 학점 기준 입력>> 3.2
장학생 명단 : 최고조 상당히 고득점
```

3.2 이상 학생

힌트 Hint

HashMap의 전체 요소를 검색하여 학점이 3.2 이상인 학생을 알아내야 한다. 예제 7-6은 해시맵 전체를 검색하는 코드 사례를 보여준다.

목적 HashMap의 요소 값을 수정하는 연습

8. 고객의 이름과 포인트 점수를 관리하는 프로그램을 해시맵을 이용하여 작성하라. 프로그램은 고객의 이름과 포인트를 함께 저장 관리하는데, 포인트는 추가될 때마다 누적하여 저장된다. 난이도 6

```
** 포인트 관리 프로그램입니다 **
이름과 포인트 입력>> 이재문 40
(이재문,40)
이름과 포인트 입력>> 황기태 50
(이재문,40)(황기태,50)
이름과 포인트 입력>> 황기태 60
(이재문,40)(황기태,110)
이름과 포인트 입력>> 김남윤 30
(이재문,40)(김남윤,30)(황기태,110)
이름과 포인트 입력>> 이재문 20
(이재문,60)(김남윤,30)(황기태,110)
이름과 포인트 입력>> 그만
```

황기태의 포인트 점수 증가

목적 제네릭 클래스 작성 연습

9. 다음 IStack 인터페이스가 있다.

```
interface IStack<T> {
   T pop();
   boolean push(T ob);
}
```

IStack<T> 인터페이스를 구현(implements)하는 MyStack<T> 클래스를 작성하라. 스택의 원소는 Vector<E>를 이용하여 저장하라. 다음은 MyStack<Integer>로 구체화한 정수 스택을 생성하고 활용하는 코드와 실행 결과이다. 난이도 6

```
public class StackManager {
    public static void main (String[] args) {
        IStack<Integer> stack = new MyStack<Integer>();
        for (int i=0; i<10; i++) stack.push(i); // 10개의 정수 푸시
        while (true) { // 스택이 빌 때까지 pop
            Integer n = stack.pop();
            if(n == null) break; // 스택이 빈 경우
            System.out.print(n + " ");
        }
    }
}
```

```
9 8 7 6 5 4 3 2 1 0
```

10. Vector<Shape>의 벡터를 이용하여 그래픽 편집기를 만들어보자. 본문 5.6절과 5.7절에서 사례로 든 추상 클래스 Shape과 Line, Rect, Circle 클래스 코드를 잘 완성하고 이를 활용하여 "삽입", "삭제", "모두 보기", "종료"의 4가지 그래픽 편집 기능을 프로그램을 작성하라. 6장 실습문제 6번을 Vector<Shape>을 이용하여 재작성하는 연습이다. Vector를 이용하면 6장 실습문제 6번보다 훨씬 간단히 작성됨을 경험할 수 있다. 난이도 7

목적 Vector의 진정한 경험하기

```
그래픽 에디터 beauty을 실행합니다.
삽입(1), 삭제(2), 모두 보기(3), 종료(4)>>1
Line(1), Rect(2), Circle(3)>>2
삽입(1), 삭제(2), 모두 보기(3), 종료(4)>>1
Line(1), Rect(2), Circle(3)>>3
삽입(1), 삭제(2), 모두 보기(3), 종료(4)>>3
Rect
Circle
삽입(1), 삭제(2), 모두 보기(3), 종료(4)>>2
삭제할 도형의 위치>>3
삭제할 수 없습니다.
삽입(1), 삭제(2), 모두 보기(3), 종료(4)>>4
beauty을 종료합니다.
```

11. 나라와 수도 맞추기 게임의 실행 예시는 다음과 같다. 난이도 7

목적 컬렉션의 실전 응용하기

```
**** 수도 맞추기 게임을 시작합니다. ****
입력:1, 퀴즈:2, 종료:3>> 1          ← 입력 메뉴 선택
현재 9개 나라와 수도가 입력되어 있습니다.
나라와 수도 입력10>> 한국 서울
나라와 수도 입력11>> 그리스 아테네
그리스는 이미 있습니다!
나라와 수도 입력11>> 호주 시드니
나라와 수도 입력12>> 이탈리아 로마
나라와 수도 입력13>> 그만           ← 입력 종료
입력:1, 퀴즈:2, 종료:3>> 2          ← 퀴즈 메뉴 선택
호주의 수도는? 시드니
정답!!
독일의 수도는? 하얼빈
아닙니다!!
멕시코의 수도는? 멕시코시티
정답!!
이탈리아의 수도는? 로마
정답!!
한국의 수도는? 서울
```

```
정답!!
영국의 수도는? 런던
정답!!
중국의 수도는? 그만        퀴즈 종료
입력:1, 퀴즈:2, 종료:3>> 3    게임 종료
게임을 종료합니다.
```

(1) 나라 이름(country)과 수도(capital)로 구성된 Nation 클래스를 작성하고 Vector<Nation> 컬렉션을 이용하여 프로그램을 작성하라.

(2) 이 프로그램을 HashMap<String, String>을 이용하여 작성하라. '키'는 나라 이름이고 '값'은 수도이다.

목적 컬렉션의 실전 응용 경험하기

12. Open Challenge를 수정하여 사용자가 단어를 삽입할 수 있도록 기능을 추가하라.

난이도 8

```
**** 영어 단어 테스트 프로그램 "명품영어" 입니다. ****
단어 테스트:1, 단어 삽입:2. 종료:3>> 1
현재 17개의 단어가 들어 있습니다. -1을 입력하면 테스트를 종료합니다.
eye?
(1)눈 (2)동물 (3)사랑 (4)감정 :>1
Excellent !!
human?
(1)인간 (2)감정 (3)거래 (4)그림 :>2
No. !!
fault?
(1)오류 (2)감정 (3)사회 (4)조각상 :>1
Excellent !!
emotion?
(1)그림 (2)사랑 (3)거래 (4)감정 :>ㅁ러미    잘못 입력한 경우
숫자를 입력하세요 !!
painting?
(1)거래 (2)눈 (3)애인 (4)그림 :>4
Excellent !!
animal?
(1)사회 (2)그림 (3)동물 (4)눈 :>-1    단어 테스트 종료

단어 테스트:1, 단어 삽입:2. 종료:3>> 2    단어 삽입 메뉴
영어 단어에 그만을 입력하면 입력을 종료합니다.
영어 한글 입력 >> flag 깃발
영어 한글 입력 >> friend 친구
영어 한글 입력 >> computer 컴퓨터    단어 삽입
영어 한글 입력 >> imagine 상상
```

```
영어 한글 입력 >> smile 웃음
영어 한글 입력 >> 그만

단어 테스트:1, 단어 삽입:2. 종료:3>> 1
현재 22개의 단어가 들어 있습니다. -1을 입력하면 테스트를 종료합니다.
smile?
(1)상상 (2)사회 (3)웃음 (4)주식 :>3
Excellent !!
friend?
(1)오류 (2)친구 (3)자기 (4)사진 :>2
Excellent !!
dall?
(1)컴퓨터 (2)인형 (3)상상 (4)사랑 :>-1

단어 테스트:1, 단어 삽입:2. 종료:3>> 3
"명품영어"를 종료합니다.
```

단어 테스트 시작

단어 테스트 종료

13. Bonus 명령을 실행하는 소프트웨어를 작성하라. 명령은 다음과 같이 mov, add, sub, jn0, prt로 구성되며 mov sum 0는 sum 변수에 0을 삽입하고, add sum i는 sum 변수에 i 값을 더하며, sub n 1은 n 변수의 값을 1 감소시키고, jn0 n 3은 변수 n의 값이 0이 아니면 3 번째 명령(실제 4번째 줄)으로 돌아가도록 처리하고, prt sum 0은 sum 변수의 값을 출력하고 프로그램을 종료한다. prt에서 마지막 0은 특별한 의미가 없다. go는 지금까지 입력한 프로그램을 처음부터 실행하도록 하는 지시어이다.
참고로, 실행 예시에서 첫 번째 프로그램은 5에서 15까지 더하는 프로그램이다.

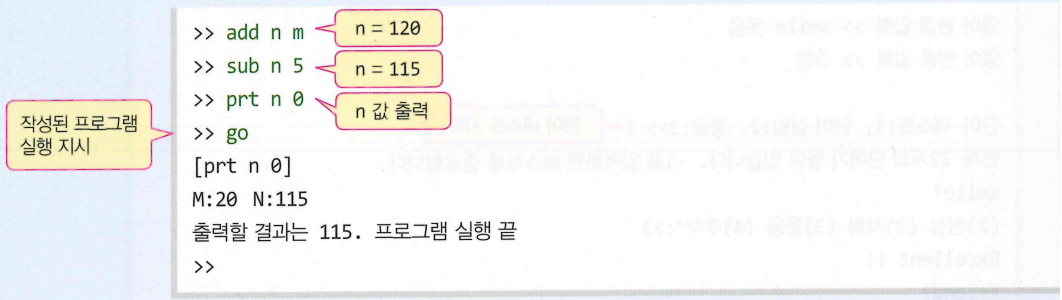
목적 컬렉션의 실전 응용하기

난이도 9

```
수퍼컴이 작동합니다. 프로그램을 입력해주세요. GO를 입력하면 작동합니다.
>> mov sum 0
>> mov i 5
>> mov n 10
>> add sum i
>> add i 1
>> sub n 1
>> jn0 n 3
>> prt sum 0
>> go
[prt sum 0]
I:15 SUM:95  N:0
출력할 결과는 95. 프로그램 실행 끝
>> mov n 100
>> mov m 20
```

5에서 15까지 더하는 사용자 프로그램 입력

입력된 프로그램 실행 지시


작성된 프로그램 실행 지시
>> add n m
n = 120
>> sub n 5
n = 115
>> prt n 0
n 값 출력
>> go
[prt n 0]
M:20 N:115
출력할 결과는 115. 프로그램 실행 끝
>>



8

입출력 스트림과 파일 입출력

Objectives

- 입출력 스트림의 특성에 대해 이해한다.
- 자바의 바이트 스트림과 문자 스트림의 개념을 이해한다.
- 문자 스트림을 이용하여 문자 단위로 입출력할 수 있다.
- 바이트 스트림을 이용하여 바이트 단위로 입출력할 수 있다.
- 버퍼 스트림을 이해하고 활용 방법을 안다.
- File 클래스와 파일 입출력 응용프로그램을 작성할 수 있다.
- 텍스트 파일을 복사하는 프로그램을 작성할 수 있다.
- 이미지 등의 바이너리 파일을 복사하는 프로그램을 작성할 수 있다.
- File 클래스를 이용하여 파일의 크기, 이름, 경로명 등을 알아낼 수 있다.
- File 클래스를 이용하여 파일의 이름을 변경하거나, 디렉터리에 있는 모든 파일의 이름을 알아낼 수 있다.



CHAPTER 8

JAVA PROGRAMMING

입출력 스트림과 파일 입출력

8.1 자바의 입출력 스트림

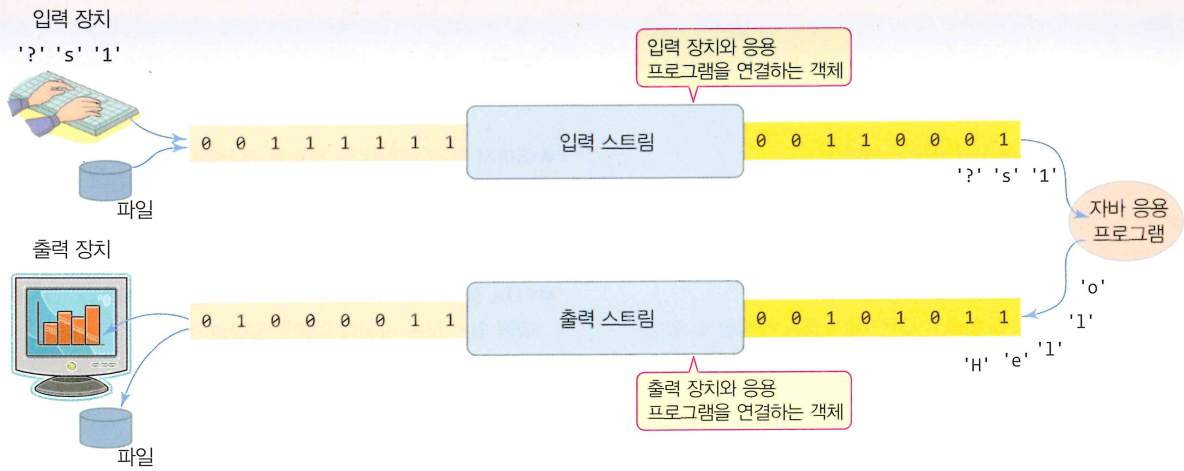
스트림 입출력이란?

스트림
소프트웨어 모듈

영어 단어로 스트림(Stream)이란 흐르는 시냇물을 뜻하며, 컴퓨터 공학에서 스트림은 연속적인 데이터의 흐름 혹은 데이터를 전송하는 소프트웨어 모듈을 일컫는다. 시냇물에 띄어진 종이배가 순서대로 흘러가듯이, 컴퓨터에서 스트림은 도착한 순서대로 데이터를 흘러 보낸다.

입출력 스트림

자바에서 입출력 스트림은 [그림 8-1]과 같이 응용프로그램과 입출력 장치를 연결하는 소프트웨어 모듈이다. 그림에서 응용프로그램은 입력 스트림과 연결하며, 입력 스트림은 키보드 장치를 제어하여 사용자의 키 입력을 받아 응용프로그램에게 전달한다. 또한 응용프로그램은 출력 스트림에 연결하고 출력 스트림에 출력하면, 출력 스트림은 다른 끝단에 연결된 출력 장치를 제어하여 출력을 완성한다. 스트림 입출력 방식



[그림 8-1] 입출력 스트림의 역할

에서, 자바 응용프로그램은 입출력 장치를 직접 제어하는 대신, 입출력 스트림 객체와 연결하여 쉽게 데이터 입출력을 실행한다.

스트림을 사용하지 않고 자바 응용프로그램이 입출력 장치를 직접 제어하는 코드를 작성하여 입출력을 실행하려 한다면, 응용프로그램 작성이 매우 어렵고, 하드웨어 구조나 제어가 다양한 입출력 장치를 모두 수용할 수 없게 될 것이다. 이제, 입출력 스트림의 특징을 간단히 정리해보자.

● 스트림의 양끝에는 입출력 장치와 자바 응용프로그램이 연결된다

자바 응용프로그램은 입력 스트림과 출력 스트림과만 연결하고, 입출력 스트림이 입출력 장치를 제어하고 실질적인 입출력을 담당한다.

● 스트림은 단방향이다

입력 스트림은 입력 장치에서 응용프로그램으로 데이터를 전송하며, 출력 스트림은 응용프로그램으로부터 받은 데이터를 출력 장치로 전송한다. 두 가지 기능을 모두 가진 스트림은 없다.

● 스트림을 통해 흘러가는 기본 단위는 바이트나 문자이다

자바의 스트림 객체는 바이트를 단위로 입출력하는 바이트 스트림과 문자 단위로 입출력하는 문자 스트림으로 나뉜다.

● 스트림은 선입선출, 즉 FIFO 구조이다

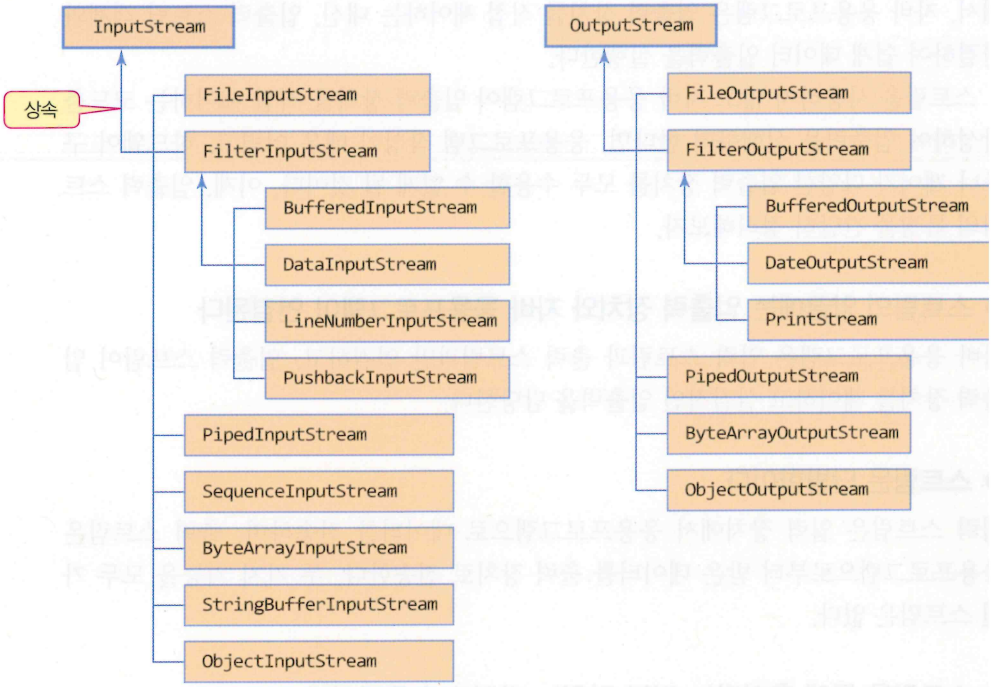
입력 스트림에 먼저 들어온 데이터가 응용프로그램에 먼저 전달되고, 출력 스트림은 응용프로그램이 출력한 순서대로 출력 장치에 보낸다.

바이트 스트림과 문자 스트림

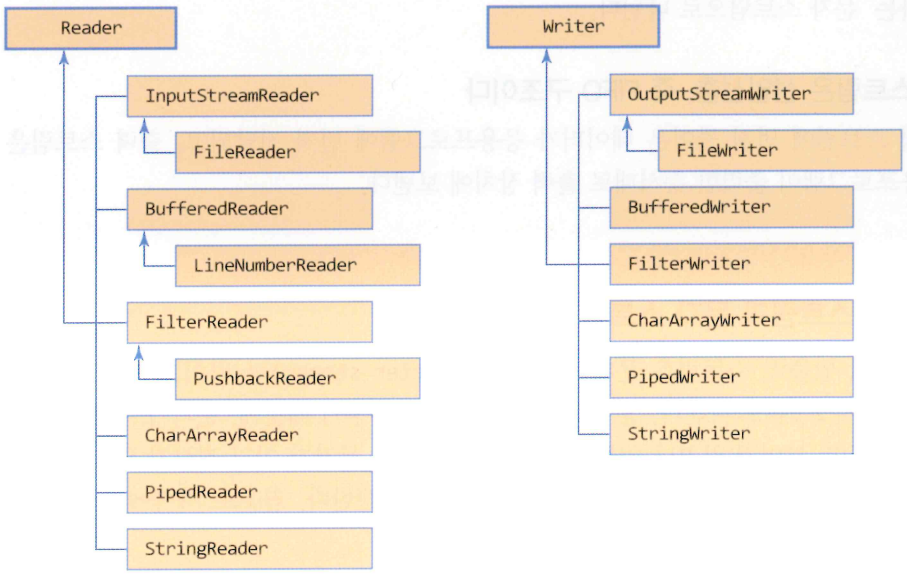
문자 스트림
바이트 스트림

자바에서 입출력 스트림은 문자 스트림(character stream)과 바이트 스트림(byte stream)의 2종류로 나눈다. 문자 스트림은 문자만 다루기 때문에, 문자가 아닌 데이터가 출력되면 보이지 않거나 엉뚱한 기호가 출력되며, 문자가 아닌 정보가 입력되면 응용프로그램에게 엉뚱한 문자가 전달되는 오류가 발생한다. 참고로 자바에서 char 타입, 즉 문자 하나의 크기는 2바이트이다.

한편, 바이트 스트림은 바이트를 단위로 다루는 스트림으로, 스트림에 들어오고 나가는 정보를 단순 바이너리(비트들)로 다루기 때문에, 문자이든 이미지 바이트든 상관없이 흘려보낸다.



(a) 바이트 스트림 클래스 계층 구조



(b) 문자 스트림 클래스 계층 구조

[그림 8-2] JDK에서 제공하는 자바의 입출력 스트림 클래스 계층 구조

Stream
Reader/Writer

자바 플랫폼은 [그림 8-2]와 같이 응용프로그램에서 바이트 단위나 문자 단위로 스트림 입출력을 할 수 있는 다양한 클래스들을 제공하며, 이들은 모두 java.io 패키지에 포함되어 있다. 바이트 스트림을 다루는 클래스는 공통적으로 이름 뒤에 Stream을 붙이고, 문자 스트림을 다루는 클래스는 Reader/Writer를 붙여 구분한다.

메모장으로 작성된 텍스트 파일이나 자바 소스 파일 같이 문자들로만 이루어진 파일을 읽고 쓰는 경우, 문자 스트림 클래스(FileReader, FileWriter)나 바이트 스트림 클래스(FileInputStream, FileOutputStream) 모두 사용 가능하지만, 이미지나 오디오/비디오 파일의 데이터는 문자가 아닌 바이너리 정보들이므로, 이들을 읽거나 쓰는 경우 반드시 바이트 스트림 클래스(FileInputStream, FileOutputStream)를 사용해야 한다.

스트림 연결

스트림은 서로 연결될 수 있다. 이해를 돕기 위해 [그림 8-3]과 같이 쿠키를 굽는 시스템을 생각해보자. 반죽된 밀가루는 적당한 크기로 잘려 컨베이어 벨트에 실려 보내진다. 하나의 스트림인 셈이다. 그리고 다시 반죽을 적당한 크기로 잘라 보내는 커팅 스트림이 있고, 다시 반죽 조각은 별 모양으로 바뀌어 쿠키 스트림을 따라 이동한다.

System.in
InputStreamReader 연결

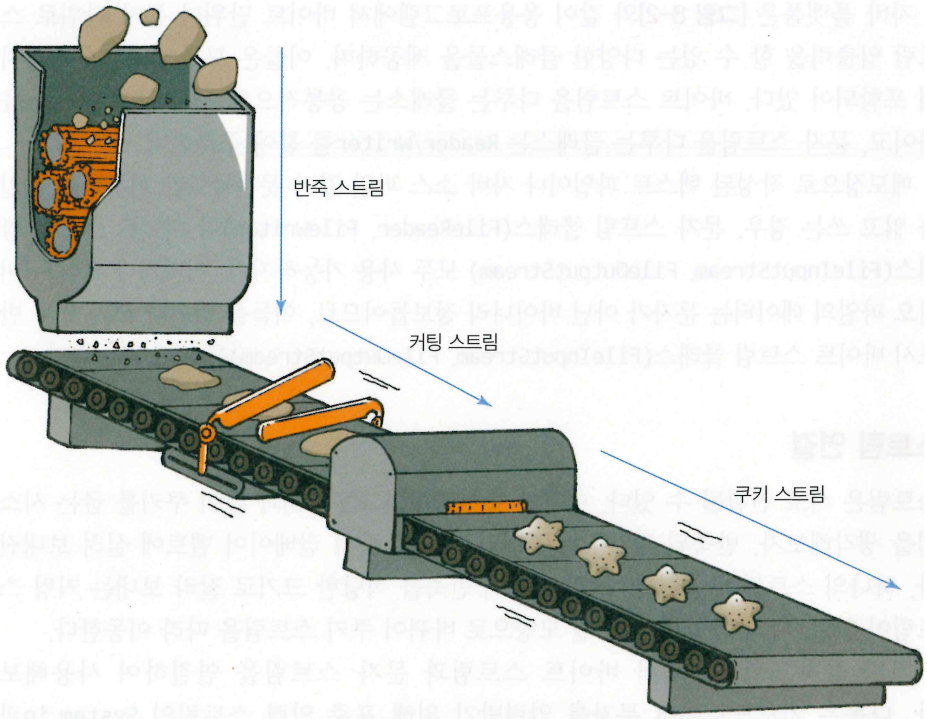
자바 응용프로그램에서 바이트 스트림과 문자 스트림을 연결하여 사용해보자. 다음은 키보드로부터 문자를 입력받기 위해 표준 입력 스트림인 System.in과 InputStreamReader 스트림 객체를 연결하는 코드이다.

```
InputStreamReader rd = new InputStreamReader(System.in);
```

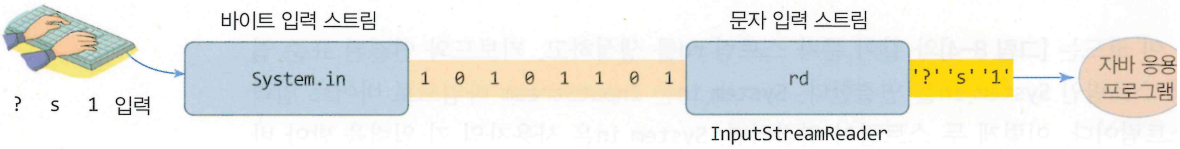
이 코드는 [그림 8-4]와 같이 문자 스트림 rd를 생성하고, 키보드와 연결된 표준 입력 스트림인 System.in을 연결한다. System.in은 InputStream 타입으로 바이트 입력 스트림이다. 이렇게 두 스트림이 연결되면, System.in은 사용자의 키 입력을 받아 바이트 스트림으로 내보내며, rd는 들어오는 바이트 스트림을 문자로 구성하여 응용프로그램에게 전달한다. 자바 응용프로그램은 다음과 같이 rd.read()를 통해 사용자가 입력한 문자를 읽을 수 있다.

```
int c = rd.read(); // 입력 스트림으로부터 키 입력. c는 입력된 키의 문자 값
```

예를 들어, [그림 8-4]에서 사용자가 a, ? 키를 순서대로 입력하면 System.in은 a와 ? 키를 읽고 바이트 정보를 내보낸다. rd는 연속적으로 들어오는 바이트들을 문자로 변환하고 'a', '?' 문자를 응용프로그램에게 전달해 준다. 스트림은 몇 개라도 연결할 수 있다.



[그림 8-3] 별 모양 쿠키를 굽는 스트림. 3개의 스트림이 연결되어 있음



[그림 8-4] 표준 입력 스트림 System.in과 문자 스트림 InputStreamReader의 연결



CHECK TIME

1 다음 파일을 읽을 때 바이트 스트림 클래스와 문자 스트림 클래스 중 더 적합한 것은?

(1) 동영상 파일(.avi)
(2) 메모장으로 작성한 파일(.txt)
(3) 자바 클래스 파일(.class)
(4) HTML 파일(.html)

2 JDK에서 제공하는 스트림 클래스 중 문자 스트림 클래스의 이름에 공통적으로 들어가는 단어는 무엇인가?

8.2 문자 스트림과 파일 입출력

문자 스트림 클래스

문자 스트림은 2바이트의 유니코드 문자를 단위로 입출력하는 스트림이다. 문자화되지 않는 바이너리 바이트 값들은 문자 스트림 클래스에서 처리할 수 없다. 따라서 이미지와 같은 바이너리 정보는 처리할 수 없다. 문자 입력 스트림은 바이트들을 전달받고, 이 바이트들을 '로컬 문자 집합'에 있는 문자인지 비교하여 문자로 변환한다. 만일 '로컬 문자 집합'에서 찾을 수 없는 경우, 문자가 아니거나 다른 언어의 문자들 혹은 문자 집합이 잘못 설정되어 있는 경우이다. 이제, 문자 스트림 클래스를 활용해보자.

2바이트의 유니코드 문자

FileReader를 이용한 텍스트 파일 읽기

〈표 8-1〉와 〈표 8-2〉는 텍스트 파일을 읽는데 사용되는 FileReader 클래스의 생성자와 메소드를 보여준다.

● 파일 입력 스트림 생성

먼저 파일 입력 스트림을 생성하고 스트림을 파일과 연결한다. 다음은 FileReader로 파일 입력 스트림을 생성하고, c:\test.txt 파일을 연결하는 코드이다.

FileReader

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

FileReader의 생성자는 c:\test.txt 파일을 찾아 열고, 파일과 스트림을 연결한다. c:\test.txt 파일은 문자들로만 구성된 텍스트 파일이다.

파일과 스트림 연결

● 파일 읽기

이제 파일 입력 스트림(fin)을 이용하여 파일을 읽어보자. fin.read()는 연결된 파일로부터 문자 하나를 읽어 리턴하며, 파일의 끝(EOF)을 만나면 -1을 리턴한다. fin.read()를 이용하여 파일 전체를 읽어 화면에 출력하는 코드는 다음과 같다.

fin.read()

```
int c;
while((c = fin.read()) != -1) { // 문자 하나를 c에 읽어 들인다. 파일 끝까지 반복한다.
    System.out.print((char)c); // 문자 c를 화면에 출력한다.
}
```

c=fin.read()에서 문자를 읽는데 왜 char 타입이 아닌, int 타입의 변수 c를 사용하는지 궁금하다. 그것은 451쪽 잠깐 부분을 참고하기 바란다.

블록 read()

파일이 큰 경우 한 번에 한 문자씩 읽으면 읽는 속도가 너무 느리기 때문에, 다음과 같이 한 번에 한 블록(버퍼 크기)만큼 읽는 read()를 이용하면 된다.

```
char [] buf = new char [1024]; // 1024는 1KB이다.
int n = fin.read(buf); // 한 번에 1024개 문자를 읽어 buf[]에 저장하고 실제 읽은 문자수 리턴
```

버퍼란 읽고 쓸 데이터를 저장하는 배열을 부르는 용어이다. 이 코드에서 파일에 남아 있는 문자의 개수가 1024개보다 작을 때, read()는 실제로 읽은 문자의 개수를 리턴한다. 그러므로 n과 버퍼의 크기인 1024를 비교하여 n이 작다면, 파일의 끝까지 읽은 것으로 판단하면 된다(예제 8-11 참고).

● 스트림 닫기

close()

파일 읽기가 더 이상 필요 없으면 다음과 같이 close() 메소드를 호출하여 파일 입력 스트림을 닫는다. 닫힌 스트림으로부터는 더 이상 읽을 수 없다.

```
fin.close();
```

〈표 8-1〉 FileReader의 생성자

생성자	설명
FileReader(File file)	file로부터 읽는 FileReader 생성
FileReader(String name)	name 이름의 파일로부터 읽는 FileReader 생성

〈표 8-2〉 FileReader, Reader, InputStreamReader의 공통 주요 메소드

메소드	설명
int read()	한 개의 문자를 읽어 정수형으로 리턴
int read(char[] cbuf)	문자들을 읽어 cbuf 배열에 저장하고 읽은 개수 리턴
int read(char[] cbuf, int off, int len)	최대 len 개수의 문자들을 읽어 cbuf 배열의 off 위치부터 저장하고 실제 읽은 개수 리턴
String getEncoding()	스트림이 사용하는 문자 집합의 이름 리턴
void close()	입력 스트림을 닫고 관련된 시스템 자원 해제



잠깐!

일반적으로 경로명을 말할 때 c:\test.txt와 같이 표현하지만, 윈도우에서 경로명을 문자열로 표현할 때는 "c:\\test.txt"라고 명기해야 한다. 자바는 '\'를 특수 문자에 사용하므로, 문자열에 '\' 문자를 사용하고자 하면 '\\'와 같이 백슬래시를 두 번 사용하도록 하고 있다. 만일, "c:\test.txt"라고 기술하면 문자열 중간에 있는 '\t'를 탭(<Tab>) 문자로 잘못 처리한다.

파일 입출력과 예외 처리

파일 입출력 실행 중 예외가 발생할 수 있다. 첫째, 파일의 경로명이 틀린 경우, FileReader 생성자는 FileNotFoundException 예외를 발생시킨다.

FileNotFoundExceptio

```
FileReader fin = new FileReader("c:\\test.txt"); // FileNotFoundException 발생 가능
```

둘째, 파일 읽기, 쓰기, 닫기를 하는 동안 입출력 오류가 발생하면, read(), write(), close() 메소드는 IOException 예외를 발생시킨다.

IOException

```
int c = fin.read(); // IOException 발생 가능
```

그러므로 파일 입출력 코드에 다음과 같은 try-catch 블록이 필요하다.

```
try {
    FileReader fin = new FileReader("c:\\test.txt");
    ..
    int c = fin.read();
    ...
    fin.close();
} catch(FileNotFoundException e) {
    System.out.println("파일을 열 수 없음");
} catch(IOException e) {
    System.out.println("입출력 오류");
}
```

생략 가능. FileNotFoundException은 IOException을 상속받기 때문에 아래의 catch 블록 하나만 있으면 됨

그러나 FileNotFoundException은 IOException을 상속받기 때문에, 프로그램을 단순화하고자 하면, catch(IOException e) { .. } 블록에서 모두 예외를 처리해도 무관하다.



예제 8-1 FileReader로 텍스트 파일 읽기

FileReader를 이용하여 c:\windows\system.ini 파일을 읽어 화면에 출력하는 프로그램을 작성하라. system.ini는 텍스트 파일이다.

```
1  import java.io.*;
2
3  public class FileReaderEx {
4     public static void main(String[] args) {
5        FileReader fin = null;
6        try {
7           fin = new FileReader("c:\\windows\\system.ini"); // 문자 입력 스트림 생성
8           int c;
9           while ((c = fin.read()) != -1) { // 한 문자씩 파일 끝까지 읽는다.
10             System.out.print((char)c);
11          }
12          fin.close();
13       }
14       catch (IOException e) {
15          System.out.println("입출력 오류");
16       }
17    }
18 }
```

실행 결과

```
; 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]
```



잠깐!

개발자들은 아래 코드와 같이 파일이나 입력 스트림을 읽는 read() 메소드를 보면서 두 가지 의문을 품을 수 있다.

```
FileInputStream in = new FileInputStream("c:\\test.txt");
int c = in.read(); // in.read()는 한 바이트를 읽어 int 타입으로 리턴한다.
while((c = in.read()) != -1) { // 파일의 끝을 만나면 -1을 리턴한다.
    System.out.print((char)c);
}
```

첫째, read()는 한 바이트나 한 문자를 리턴하므로, 리턴 타입이 byte 혹은 char인 것이 합당한데, int로 선언한 이유는 무엇일까? 둘째, read()가 스트림의 끝 혹은 파일의 끝을 만나면 -1을 리턴하는데, 스트림이나 파일에 -1이 있다면 이 둘은 어떻게 구분할 수 있는가?
이 두 의문의 해답은 모두 스트림 혹은 파일의 끝 처리와 연관되어 있다. 만일 스트림이나 파일에 0xFF 값이 있다고 하면, read()는 0xFF를 리턴한다. 이 때 독자들은 "어! 0xFF는 -1인데, 파일의 끝을 나타내는 -1과 혼동되네!"라고 생각할 수 있다. 그러나 read()는 int 타입으로 리턴하므로, 0xFF를 32비트의 0x000000FF로 리턴한다. 이것은 -1이 아니다. int 타입의 -1은 0xFFFFFFFF이다. 스트림이나 파일에서 read()가 0xFF의 값을 읽어 리턴하는 것과 확연히 구분된다. 이것이 바로 read()의 리턴 타입이 int인 이유이다.

문자 집합과 InputStreamReader를 이용한 텍스트 파일 읽기

문자 집합

InputStreamReader는 스트림에 입력되는 바이트 데이터를 문자 집합을 통해 문자로 변환한다. 이를 위해 InputStreamReader의 생성자에는 문자 집합을 지정해야 한다. 만일 읽어 들인 바이트들이 문자 집합에 속하지 않는 경우 해독할 수 없는 글자가 된다.

예제 8-2는 InputStreamReader에 문자 집합을 지정하고 텍스트 파일을 읽는 예를 보인다. InputStreamReader 클래스의 생성자는 〈표 8-3〉과 같고, 메소드는 FileReader의 동일하게 〈표 8-2〉와 같다.

〈표 8-3〉 InputStreamReader 생성

생성자	설명
InputStreamReader(InputStream in)	in으로부터 읽는 기본 문자 집합의 InputStreamReader 생성
InputStreamReader (InputStream in, Charset cs)	in으로부터 읽는 cs 문자 집합의 InputStreamReader 생성
InputStreamReader (InputStream in, String charsetName)	in으로부터 읽는 charsetName 문자 집합의 InputStreamReader 생성

● InputStreamReader로 문자 입력 스트림 생성

InputStreamReader는 바이트 스트림을 전달받아 문자 정보로 변환하는 스트림 객체이다. 그러므로 우선 텍스트 파일을 읽기 위해서는 다음과 같이 바이트 파일 입력 스트림을 먼저 생성한다.

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

여기서 c:\tmp\hangul.txt는 메모장으로 작성한 한글 텍스트 파일이다. 그러고 나서, 다음과 같이 InputStreamReader 객체를 생성한다.

```
InputStreamReader in = new InputStreamReader(fin, "MS949");
```

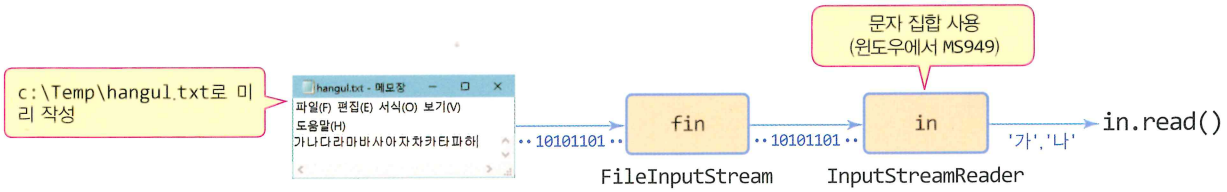
생성자 InputStreamReader()의 두 번째 매개변수에는 fin으로부터 읽어 들인 바이트들을 문자로 인코딩하기 위한 문자 집합을 지정한다.

MS949

윈도우의 메모장에서 한글 텍스트를 입력하면 디폴트로 마이크로소프트(MS)에서 만든 '한글 확장 완성형 문자 집합'에 있는 코드로 저장한다. 윈도우에서 디폴트로 사용하는 문자 집합은 MS949이다. 그러므로 앞의 코드에서 InputStreamReader 생성자에 MS949 문자 집합을 지정하였다.

● 파일 읽기

스트림이 연결된 모양은 [그림 8-5]와 같다. 이제 in 스트림을 이용하여 파일을 읽어 보자. in.read()는 문자 집합의 인코딩 규칙에 따라, fin에게 파일로부터 필요한 바이트들을 읽도록 지시하고, 읽어 들인 바이트들을 MS949 문자 집합에 정의된 문자인지 찾아 한글 문자를 리턴한다. 만일 문자 집합에 없는 바이트들인 경우 정상적이지 않은 값을 리턴한다.



[그림 8-5] FileInputStream 스트림을 연결한 InputStreamReader 스트림

예제 8-2는 InputStreamReader를 이용하여 한글 텍스트 파일을 정상적으로 읽는 예이며, 예제 8-3은 잘못된 문자 집합인 US-ASCII 문자 집합을 지정하여 한글 텍스트 파일을 제대로 읽지 못하는 예이다.

InputStreamReader로 한글 텍스트 파일 읽기 예제 8-2

InputStreamReader을 이용하여 MS949 문자 집합으로 한글 텍스트 파일을 읽고 출력하라.

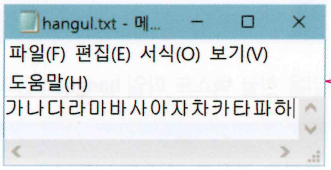
```
1  import java.io.*;
2
3  public class FileReadHangulSuccess {
4     public static void main(String[] args) {
5        InputStreamReader in = null;
6        FileInputStream fin = null;
7        try {
8           fin = new FileInputStream("c:\\Temp\\hangul.txt");
9           in = new InputStreamReader(fin, "MS949"); // 올바른 문자 집합 지정
10          int c;
11
12          System.out.println("인코딩 문자 집합은 " + in.getEncoding());
13          while ((c = in.read()) != -1) {
14             System.out.print((char)c);
15          }
16          in.close();
17          fin.close();
18       } catch (IOException e) {
19          System.out.println("입출력 오류");
20       }
21    }
22 }
```

MS에서 만든 한글 확장 완성형 문자 집합

문자 집합 이름 리턴

실행 결과

```
인코딩 문자 집합은 MS949
가나다라마바사아자차카타파하
```



읽을 한글 텍스트 파일 c:\Temp\hangul.txt는 미리 작성되어 있어야 한다.

잠깐!

한글은 영어와 다른 코딩 방식으로 저장된다. 가장 많이 알려진 KSC5601은 1987년에 국가 표준 코드로 지정되었다. 마이크로소프트는 KSC5601을 확장하여 MS949라고도 불리는 코드 체계를 만들고 윈도우에 기본 코드로 사용하고 있다. 자바는 전 세계의 모든 문자를 표현하기 위해 만들어진 유니코드를 사용하므로, 메모장과 같이 윈도우에서 작성된 한글 텍스트 문서를 자바 프로그램에서 읽고자 하면 '문자 집합'을 MS949로 지정하여야 한다.

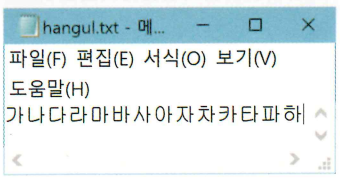
예제 8-3 한글 텍스트 파일 읽기(문자 집합 지정이 잘못된 경우)

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

```
1  import java.io.*;
2
3  public class FileReadHangulFail {
4     public static void main(String[] args) {
5        InputStreamReader in = null;
6        FileInputStream fin = null;
7        try {
8           fin = new FileInputStream("c:\\Temp\\hangul.txt"); // 파일로부터 바이트
                                                               입력 스트림 생성
9           in = new InputStreamReader(fin, "US-ASCII");
10
11          int c;
12
13          System.out.println("인코딩 문자 집합은 " + in.getEncoding()); // 문자집합 출력
14          while ((c = in.read()) != -1) { // 문자 단위로 읽는다.
15             System.out.print((char)c);
16          }
17          in.close();
18          fin.close();
19       } catch (IOException e) {
20          System.out.println("입출력 오류");
21       }
22    }
23 }
```

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

읽을 한글 텍스트 파일 hangul.txt의 내용은 다음과 같다.



US-ASCII 인코딩으로 지정하여 읽었을 때 다음과 같이 출력 결과는 깨져 나온다.

실행 결과

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

FileWriter를 이용한 텍스트 파일 쓰기

FileWriter

이 절에서는 FileWriter를 이용하여 텍스트를 파일에 저장하는 방법을 알아보자. 〈표 8-4〉는 문자 출력 스트림인 FileWriter와 OutputStreamWriter의 생성자를, 〈표 8-5〉는 Writer, OutputStreamWriter, FileWriter의 주요 공통 메소드를 보여준다.

● 파일 출력 스트림 생성

다음 코드는 c:\Temp\test.txt 파일에 텍스트를 쓰는 출력 스트림을 생성한다.

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

FileWriter의 생성자는 c:\Temp\test.txt 파일을 열어 스트림과 연결한다. 파일이 없는 경우 빈 파일을 생성하며, 이미 파일이 있는 경우 파일 내용을 지우고 파일의 처음부터 쓸 준비를 한다.

● 파일 쓰기

write()

fout 스트림의 write() 메소드를 이용하면 문자 단위로 파일에 저장할 수 있다. 다음은 'A'를 파일에 쓰는 코드이다.

```
fout.write('A'); // 문자 'A'를 파일에 저장
```

블록

write()를 다음과 같이 이용하면 한 번에 한 블록씩 쓸 수 있다.

```
char [] buf = new char [1024];
fout.write(buf, 0, buf.length); // buf[] 배열의 처음부터 배열 크기(1024개 문자)만큼 쓰기
```

● 스트림 닫기

close()

텍스트를 모두 파일에 저장하였으면 close()를 호출하여 스트림을 닫는다. 스트림을 닫으면 연결된 파일도 닫힌다.

```
fout.close(); // 스트림을 닫는다. 더 이상 스트림에 기록할 수 없다.
```

〈표 8-4〉 FileWriter와 OutputStreamWriter의 생성자

생성자	설명
OutputStreamWriter(OutputStream out)	out에 출력하는 기본 문자 집합의 OutputStreamWriter 생성
OutputStreamWriter (OutputStream out, Charset cs)	out에 출력하는 cs 문자 집합의 OutputStreamWriter 생성
OutputStreamWriter (OutputStream out, String charsetName)	charsetName 문자 집합의 OutputStreamWriter 생성
FileWriter(File file)	file에 데이터를 저장할 FileWriter 생성
FileWriter(String name)	name 파일에 데이터를 저장할 FileWriter 생성
FileWriter (File file, boolean append)	FileWriter를 생성하며 append가 true이면 파일의 마지막부터 데이터를 저장
FileWriter (String name, boolean append)	FileWriter를 생성하며 append가 true이면 파일의 마지막부터 데이터를 저장

〈표 8-5〉 FileWriter와 OutputStreamWriter의 주요 메소드

메소드	설명
void write(int c)	c를 char로 변환하여 한 개의 문자 출력
void write(String str, int off, int len)	인덱스 off부터 len개의 문자를 str 문자열에서 출력
void write(char[] cbuf, int off, int len)	인덱스 off부터 len개의 문자를 배열 cbuf에서 출력
void flush()	스트림에 남아있는 데이터 모두 출력
String getEncoding()	스트림이 사용하는 문자 집합의 이름 리턴
void close()	출력 스트림을 닫고 관련된 시스템 자원 해제



예제 8-4 키보드 입력을 파일로 저장하기

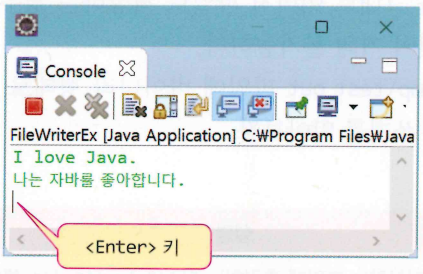
Scanner를 이용하여 키보드에서 입력받은 데이터를 c:\Temp\test.txt 파일에 저장하는 프로그램을 작성하라.

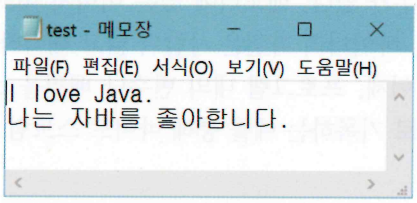
```
1  import java.io.*;
2  import java.util.*;
3
4  public class FileWriterEx {
5     public static void main(String[] args) {
6        Scanner scanner = new Scanner(System.in);
7        FileWriter fout = null;
8        int c;
9        try {
10          fout = new FileWriter("c:\\Temp\\test.txt"); // 파일과 연결된 출력 문자
                                                          스트림 생성
```

```
11          while(true) {
12             String line = scanner.nextLine(); // 빈칸을 포함하여 한 줄 읽기
13             if(line.length() == 0) // 한 줄에 <Enter>키만 입력한 경우
14                break;
15             fout.write(line, 0, line.length()); // 읽은 문자열을 파일에 저장
16             fout.write("\r\n", 0, 2); // 한 줄 띄기 위해 \r\n을 파일에 저장
17          }
18          fout.close();
19       } catch (IOException e) {
20          System.out.println("입출력 오류");
21       }
22       scanner.close();
23    }
24 }
```

line에는 '\n'이 들어가지 않음

실행 결과





실행 결과 test.txt 파일 형성

8.3 바이트 스트림과 파일 입출력

바이트 스트림 클래스

바이트 스트림은 바이트 단위로 바이너리 데이터가 흐르는 스트림이다. 바이트 스트림은 바이너리 데이터를 있는 그대로 입출력하기 때문에 이미지나 동영상 파일 입출력에 필수적이고, 문자들로 구성된 텍스트 파일도 입출력할 수 있다. 다음은 대표적인 바이트 입출력 스트림 클래스이다.

바이너리 데이터

- **InputStream/OutputStream**

추상 클래스이며, 바이트 입출력 처리를 위한 공통 기능을 가진 슈퍼 클래스이다.

- **FileInputStream/FileOutputStream**

파일 입출력을 위한 클래스로서, 파일로부터 바이너리 데이터를 읽거나 파일에 바이너리 데이터를 저장할 수 있다.

- **DataInputStream/DataOutputStream**

이 스트림을 이용하면 boolean, char, byte, short, int, long, float, double 타입의 값을 바이너리 형태로 입출력한다. 문자열도 바이너리 형태로 입출력한다.

FileOutputStream을 이용한 바이너리 파일 쓰기

바이너리 값

프로그램 내의 변수나 배열에 들어 있는 바이너리 값을 그대로 파일에 저장해야 하는 경우가 있다. 메모리에 있는 이미지 비트들을 그대로 이미지 파일로 저장하는 경우이다. 바이너리 파일은 사람이 읽고 해석하는 것이 거의 불가능하다.

이제, 프로그램 내의 변수나 배열을 c:\Temp\test.out 파일에 바이너리 형태로 그대로 기록하는 예를 통해 바이트 스트림 파일 쓰기를 알아보자.

〈표 8-6〉 FileOutputStream 클래스의 생성자

생성자	설명
FileOutputStream(File file)	file이 지정하는 파일에 출력하는 FileOutputStream 생성
FileOutputStream(String name)	name이 지정하는 파일에 출력하는 FileOutputStream 생성
FileOutputStream (File file, boolean append)	FileOutputStream을 생성하며 append가 true이면 file이 지정하는 파일의 마지막부터 데이터 저장
FileOutputStream (String name, boolean append)	FileOutputStream을 생성하며 append가 true이면 name 이름의 파일의 마지막부터 데이터 저장

〈표 8-7〉 OutputStream과 FileOutputStream의 공통 주요 메소드

메소드	설명
void write(int b)	int 형으로 넘겨진 한 바이트를 출력 스트림으로 출력
void write(byte[] b)	배열 b의 바이트를 모두 출력 스트림으로 출력
void write(byte[] b, int off, int len)	len 크기만큼 off부터 배열 b를 출력 스트림으로 출력
void flush()	출력 스트림에서 남아 있는 바이너리 데이터 모두 출력
void close()	출력 스트림을 닫고 관련된 시스템 자원 해제

바이너리 데이터를 파일에 저장할 때 FileOutputStream 클래스를 이용한다. FileOutputStream의 생성자는 〈표 8-6〉과 같으며, 〈표 8-7〉는 FileOutputStream의 주요 메소드를 보여준다.

● 파일 출력 스트림 생성

c:\Temp\test.out 파일에 바이너리 데이터를 저장하는 출력 스트림은 다음과 같이 생성한다.

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

FileOutputStream

FileOutputStream 생성자는 c:\Temp\test.out 파일을 생성하여 연결한다. 파일이 이미 있으면 그 내용을 지우고 스트림에 연결한다. 쓰기가 이루어지면 c:\Temp\test.out 파일은 바이너리 파일이 된다.

● 파일 쓰기

write()

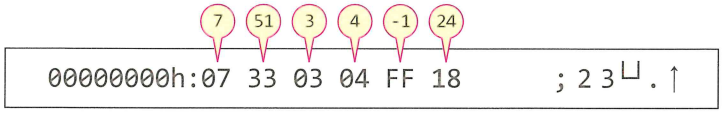
배열을 파일에 저장해보자. write() 메소드를 이용하여 다음과 같이 한 바이트씩 배열 데이터를 기록한다.

```
byte b[] = {7,51,3,4,-1,24};
for(int i=0; i<b.length; i++)
    fout.write(b[i]); // 배열 b[]의 바이트를 바이너리 그대로 파일에 저장
```

for 문 없이 한 번에 배열 b[]를 저장할 수도 있다.

```
fout.write(b); // 배열 b[]의 바이트 모두 파일 저장
```

앞의 코드를 실행한 결과 [그림 8-6]과 같이 c:\Temp\test.out 파일 내부에 바이너리 데이터가 저장된 것은 볼 수 있다.



[그림 8-6] c:\Temp\test.out 파일 내부의 바이너리 데이터



예제 8-5 FileOutputStream으로 바이너리 파일 쓰기

FileOutputStream을 이용하여 byte [] 배열 속에 들어 있는 바이너리 값을 c:\Temp\test.out 파일에 저장하라. 이 파일은 바이너리 파일이 되므로 메모장으로 볼 수 없다. 이 파일은 예제 8-6에서 FileInputStream을 이용하여 읽어 다시 출력한다.

fout.write(b); 한 줄로 코딩할 수 있다.

```
1  import java.io.*;
2
3  public class FileOutputStreamEx {
4     public static void main(String[] args) {
5        byte b[] = {7,51,3,4,-1,24};
6
7        try {
8           FileOutputStream fout = new FileOutputStream("c:\\Temp\\test.out");
9           for(int i=0; i<b.length; i++)
10             fout.write(b[i]); // 배열 b의 바이너리를 그대로 기록
11          fout.close();
12       } catch(IOException e) {
13          System.out.println("c:\\Temp\\test.out에 저장할 수 없었습니다. 경로명을 확
                                인해 주세요");
14          return;
15       }
16       System.out.println("c:\\Temp\\test.out을 저장하였습니다.");
17    }
18 }
```

실행 결과

```
c:\Temp\test.out을 저장하였습니다.
```

FileInputStream을 이용한 바이너리 파일 읽기

바이트 스트림으로 파일을 읽는 스트림 클래스는 FileInputStream이며, 생성자와 주요 메소드는 〈표 8-8〉, 〈표 8-9〉와 같다. 지금부터 예제 8-5에서 배열 byte b[]를 저장한 c:\Temp\test.out 파일을 읽는 코드 예를 통해 파일 읽기를 설명해보자.

〈표 8-8〉 FileInputStream 클래스의 생성자

생성자	설명
FileInputStream(File file)	file이 지정하는 파일로부터 읽는 FileInputStream 생성
FileInputStream(String name)	name이 지정하는 파일로부터 읽는 FileInputStream 생성

〈표 8-9〉 InputStream과 FileInputStream의 공통 메소드

메소드	설명
int read()	입력 스트림에서 한 바이트를 읽어 int형으로 리턴
int read(byte[] b)	최대 배열 b의 크기만큼 바이트를 읽음. 읽는 도중 EOF를 만나면 실제 읽은 바이트 수 리턴
int read(byte[] b, int off, int len)	최대 len개의 바이트를 읽어 b 배열의 off 위치부터 저장. 읽는 도중 EOF를 만나면 실제 읽은 바이트 수 리턴
int available()	입력 스트림에서 현재 읽을 수 있는 바이트 수 리턴
void close()	입력 스트림을 닫고 관련된 시스템 자원 해제

파일 입력 스트림 생성

FileInputStream

FileInputStream 클래스는 파일과 연결한 바이트 스트림을 생성한다. 다음은 c:\Temp\test.out 파일로부터 바이너리 값을 그대로 읽어 들이는 바이트 스트림을 생성하는 코드이다.

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

이 코드를 실행하면 c:\Temp\test.out 파일을 찾아 열고, 이 파일을 연결한 스트림 fin을 생성한다.

● 파일 읽기

read()

fin.read() 메소드를 호출하면 파일 스트림으로부터 한 바이트를 읽어 리턴한다. read() 메소드를 이용하여 파일에 저장된 바이트들을 읽어 배열 byte b[]에 다시 채우는 코드는 다음과 같다.

```
byte b[] = new byte [6]; // 비어있는 배열
int n=0, c;
while((c = fin.read()) != -1) { // 파일 끝(EOF)까지 한 바이트씩 읽기
    b[n] = (byte)c; // 읽은 바이트를 배열에 저장
    n++;
}
```

파일의 끝(EOF)을 만나면 fin.read() 메소드는 -1을 리턴한다. 이 코드의 끝에 있는 다섯 줄은 다음 한 줄로 바꾸어 한 번에 배열로 읽어 들일 수 있다.

```
fin.read(b); // 파일에서 배열 b[]의 크기만큼 바이트 읽기
```

● 스트림 닫기

close()

더 이상 스트림이 필요없을 때 close() 메소드를 호출하여 스트림을 닫는다.

```
fin.close(); // 스트림을 닫는다. 더 이상 스트림으로부터 읽을 수 없다.
```



예제 8-6 FileInputStream으로 바이너리 파일 읽기

FileInputStream을 이용하여 c:\Temp\test.out 파일(예제 8-5에서 저장한 파일)을 읽어 byte [] 배열 속에 저장하고 화면에 출력하라.

```
1  import java.io.*;
2
3  public class FileInputStreamEx {
4     public static void main(String[] args) {
5        byte b[] = new byte [6]; // 비어 있는 byte 배열
6        try {
7           // "c:\\Temp\\test.out" 파일을 읽어 배열 b에 저장
8           FileInputStream fin = new FileInputStream("c:\\Temp\\test.out");
9           int n=0, c;
10          while((c = fin.read())!= -1) { // -1은 파일 끝(EOF)
11             b[n] = (byte)c; // 읽은 바이트를 배열에 저장
12             n++;
13          }
14          // 배열 b[]의 바이트 값을 모두 화면에 출력
15          System.out.println("c:\\Temp\\test.out에서 읽은 배열을 출력합니다.");
16          for(int i=0; i<b.length; i++)
17             System.out.print(b[i] + " ");
18          System.out.println();
19
20          fin.close();
21       } catch(IOException e) {
22          System.out.println("c:\\Temp\\test.out에서 읽지 못했습니다. 경로명을 체크
                                해보세요");
23       }
24    }
25 }
```

fin.read(b);의 한 줄로 코딩할 수 있다.

이 문장의 의미를 다음 페이지의 잠깐에서 설명

실행 결과

```
c:\Temp\test.out에서 읽은 배열을 출력합니다.
7 51 3 4 -1 24
```



잠깐!

c:\Temp\test.out에는 [그림 8-6]과 같이 한 바이트의 -1(FF)이 들어 있다. 예제 8-6의 라인 10에서, fin.read()는 -1(FF)의 한 바이트를 읽어 int 타입의 0x000000FF, 즉 255로 리턴하고 255가 변수 c에 저장된다. 그러고 나서 c(0x000000FF)와 -1(0xFFFFFFFF)을 비교하여 같지 않은 것으로 판단한다. fin.read()가 파일의 끝을 만날 때 비로소 -1(0xFFFFFFFF)을 리턴하여 while 문이 종료된다.



1 FileInputStream 클래스를 이용하여 읽을 수 없는 파일은?

① JPEG 이미지 파일 ② mp3 오디오 파일 ③ 메모장으로 작성한 텍스트 파일 ④ 없다.

2 바이트 값을 그대로 파일에 기록할 때 사용하는 스트림 클래스는 무엇인가?

3 c:\Temp\test.dat 파일에 두 개의 정수 3과 5를 기록하고 파일에서 다시 읽어 화면에 출력하는 프로그램을 작성하라. 가능하면 파일 내부를 들여다보기 바란다.

8.4 버퍼 입출력과 파일 입출력

버퍼 입출력의 필요성

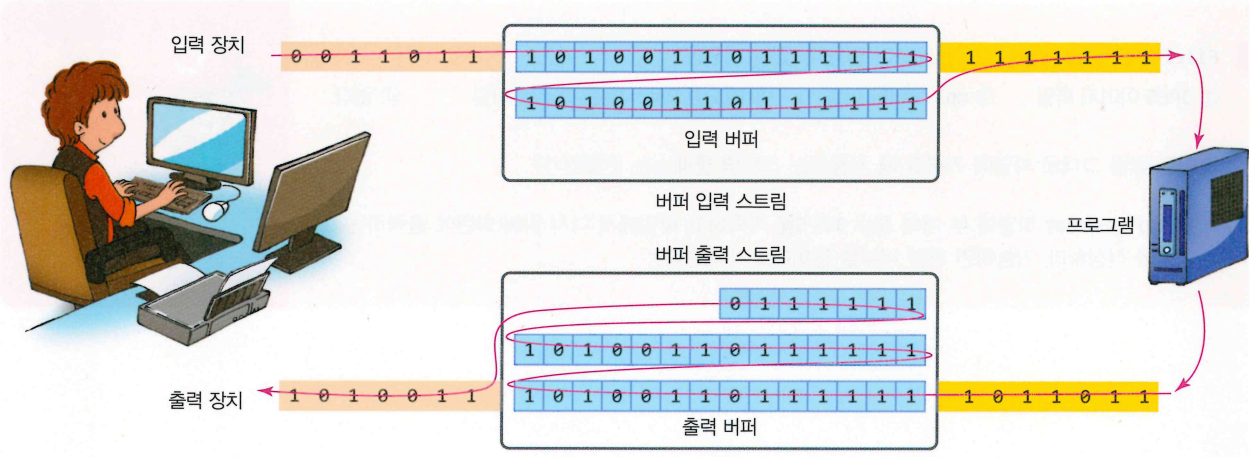
버퍼

입출력 스트림은 운영체제 API를 호출하여 입출력 장치와 프로그램 사이에서 데이터가 전송되도록 한다. 파일 쓰기 메소드는 최종적으로 운영체제 API를 호출하여 파일에 쓰도록 시키고, 운영체제 API는 하드 디스크에게 명령을 내려 파일에 데이터를 기록한다. 자주 운영체제 API가 호출될수록 하드 디스크 장치나 네트워크 장치가 자주 작동하게 되어 시스템의 효율은 나빠지고 프로그램 역시 여러 번 입출력을 진행해야 하므로 입출력의 실행 속도가 떨어진다. 만일 스트림이 버퍼(buffer)를 가지게 되면 보다 효율적으로 작동할 수 있다. 버퍼란 데이터를 일시적으로 저장하기 위한 메모리이다. 파일 출력 스트림이 파일에 쓸 데이터를 버퍼에 모아 두었다가, 한 번에 운영체제 API를 호출하여 파일에 쓰게 하면, 운영체제의 부담을 줄이고 장치를 구동하는 일이 줄어

들게 되어 시스템의 속도나 효율이 올라가게 될 것이다.

버퍼 스트림

8.3절까지 다룬 입출력 스트림은 버퍼(buffer)를 가지지 않는 unbuffered I/O 방
이었다. 버퍼 입출력(Buffered I/O)이란 [그림 8-7]과 같이 입출력 스트림이 버퍼를
지고 보다 효율적으로 입출력을 처리하는 방식이다. 버퍼를 가지는 스트림을 버퍼
트림(Buffered Stream)이라고 부른다. 버퍼 입력 스트림은 [그림 8-7]에서와 같이
력 장치로부터 입력된 데이터를 버퍼에 모아서 프로그램으로 보내며, 버퍼 출력 스
림은 프로그램에서 출력한 데이터를 역시 버퍼에 모아 두었다가 한 번에 출력 장치
출력한다.



[그림 8-7] 버퍼를 가지는 버퍼 스트림

바이트 버퍼 스트림
문자 버퍼 스트림

버퍼 스트림 역시 데이터의 타입에 따라 바이트 버퍼 스트림과 문자 버퍼 스트림으로 구분된다. 바이트 버퍼 스트림을 구현한 입출력 클래스는 BufferedInputStrea과 BufferedOutputStream이 있으며, 문자 버퍼 스트림을 구현한 클래스는 BufferedReader와 BufferedWriter가 있다. 이들을 [그림 8-2]에서 확인해보기 바란다.

버퍼 스트림 생성 및 활용

버퍼 스트림은 내부에 버퍼를 가진다는 사실만 다를 뿐이지 개발자에게 보이는 면은 입출력 스트림과 동일하다.

〈표 8-10〉은 바이트 버퍼 스트림을 구현한 BufferedInputStream과 BufferedOutputStream의 생성자를, 〈표 8-11〉은 문자 버퍼 스트림을 구현한 BufferedReader와 BufferedWriter의 생성자를 각각 보여준다. 두 표에서 볼 수 있듯

이 버퍼 스트림 클래스의 생성자는 모두 바이트 스트림 또는 문자 스트림과 연결하여 사용한다. 그리고 생성자에 버퍼의 크기를 지정한다.

BufferedInputStream과 BufferedOutputStream의 메소드는 〈표 8-7〉, 〈표 8-9〉의 InputStream, OutputStream 클래스와 각각 같으며, BufferedReader와 BufferedWriter의 메소드는 〈표 8-2〉, 〈표 8-5〉의 Reader, Writer 클래스와 각각 같다.

〈표 8-10〉 바이트 버퍼 스트림 클래스의 생성자

생성자	설명
BufferedInputStream(InputStream in)	in을 연결하는 디폴트 크기의 입력 버퍼 스트림 객체 생성
BufferedInputStream (InputStream in, int size)	in을 연결하는 size 크기의 입력 버퍼 스트림 객체 생성
BufferedOutputStream(OutputStream out)	out을 연결하는 디폴트 크기의 출력 버퍼 스트림 생성
BufferedOutputStream (OutputStream out, int size)	out을 연결하는 size 크기의 출력 버퍼 스트림 생성

〈표 8-11〉 문자 버퍼 스트림 클래스의 생성자

생성자	설명
BufferedReader(Reader in)	in을 연결하는 디폴트 크기의 문자 입력 버퍼 스트림 생성
BufferedReader(Reader in, int sz)	in을 연결하는 sz 크기의 문자 입력 버퍼 스트림 생성
BufferedWriter(Writer out)	out을 연결하는 디폴트 크기의 문자 출력 버퍼 스트림 생성
BufferedWriter(Writer out, int sz)	out을 연결하는 sz 크기의 문자 출력 버퍼 스트림 생성

● 버퍼 출력 스트림 생성

BufferedOutputStream 클래스로 화면에 출력하는 버퍼 출력 스트림을 생성해보자.

BufferedOutputStream

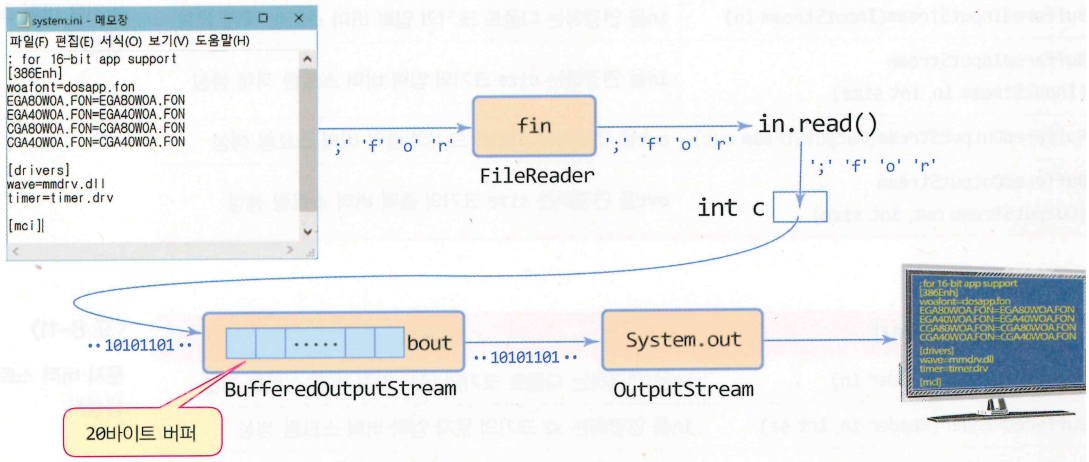
```
BufferedOutputStream bout = new BufferedOutputStream(System.out, 20);
                                                    // 20바이트 버퍼
```

앞 라인은 20바이트 크기의 버퍼를 가지고, 표준 출력 스트림(System.out)에 연결하여 화면에 출력하는 버퍼 스트림을 생성한다.

● 스트림 출력

다음은 c:\windows\system.ini 파일을 읽어, 버퍼 출력 스트림을 통해 화면에 출력하는 프로그램 코드를 보여주며, 스트림이 연결되어 있는 모양은 [그림 8-8]과 같다.

```
FileReader fin = new FileReader("c:\\windows\\system.ini");
int c;
while ((c = fin.read()) != -1) { // 파일 끝을 만날 때까지 문자들을 하나씩 읽는다.
    bout.write((char)c); // 읽은 문자를 버퍼 출력 스트림에 쓴다. 출력 스트림과 연결된 화면에
                         출력된다.
}
```



[그림 8-8] 파일을 읽고 버퍼 출력 스트림을 통해 화면에 출력하는 과정

강제 출력
flush()

● 버퍼에 남아 있는 데이터 출력

버퍼 스트림은 버퍼를 가지고 있기 때문에 버퍼가 꽉 찼을 때만 출력되는 특징이 있다. 그러므로 프로그램에서 데이터를 출력하였지만 버퍼에 들어 있어 출력 장치에 보이지 않을 수 있다. 버퍼가 다 차지 않는 상태에서 버퍼에 있는 데이터를 강제로 출력 장치로 보내려면, 다음과 같이 flush() 메소드를 호출하면 된다.

```
bout.flush(); // bout 스트림의 버퍼에 있는 데이터를 모두 장치에 출력
```

● 스트림 닫기

버퍼 스트림을 모두 사용하였다면 다음과 같이 닫아 주어야 한다.

```
bout.close(); // 버퍼 스트림 닫기
fin.close(); // 파일 입력 스트림 닫기
```

버퍼 스트림을 이용한 출력 예제 8-7

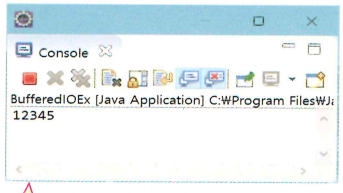
버퍼 크기를 5로 하고, 표준 출력 스트림(System.out)과 연결한 버퍼 출력 스트림을 생성하라. c:\Temp\test2.txt 파일을 저장된 영문 텍스트를 읽어 버퍼 출력 스트림을 통해 출력하라.

프로그램이 실행되면 5개의 문자 밖에 출력되지 않을 것이다. 이것은 버퍼의 크기를 5로 잡았기 때문이다. 여기서 <Enter> 키가 입력되면 flush() 메소드를 호출하여 버퍼에 있는 모든 데이터를 출력시키도록 하라.

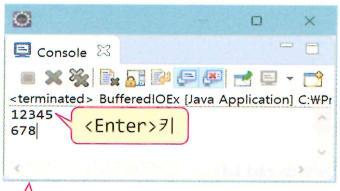
```
1  import java.io.*;
2  import java.util.Scanner;
3
4  public class BufferedIOEx {
5      public static void main(String[] args) {
6          FileReader fin = null;
7          int c;
8          try {
9              fin = new FileReader("c:\\Temp\\test2.txt");
10             BufferedOutputStream out = new BufferedOutputStream(System.out, 5);
11             while ((c = fin.read()) != -1) { // 파일 데이터를 모두 스크린에 출력
12                 out.write(c);
13             }
14             // 파일 데이터가 모두 출력된 상태
15             new Scanner(System.in).nextLine(); // <Enter> 키 입력
16             out.flush(); // 버퍼에 남아 있던 문자 모두 출력
17             fin.close();
18             out.close();
19
20         } catch (IOException e) {
21             e.printStackTrace();
22         }
23     }
24 }
```

BufferedOutputStream는 바이트 단위로 출력하므로, ANS 코드(영어는 한 바이트, 한글은 2바이트)의 영문만 출력할 ... 있다. 한글까지 출력하려면 다음과 같이 수정하면 된다.

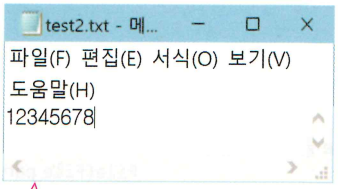
```
BufferedWriter out = new BufferedWriter(new
OutputStreamWriter(System.out), 5);
```



출력 스트림의 버퍼 크기가 5이므로 파일을 읽어 8개의 문자를 출력하였지만 5개의 문자만 출력되고 3개의 문자는 버퍼에 남아 있어 화면에 보이지 않음



프로그램의 라인 15에서 <Enter>키를 입력받으면 out.flush()를 실행하여 버퍼에 남아 있는 3개의 문자를 강제로 출력시킴



이 파일이 작성되어 있어야 함



1. 버퍼를 가진 입출력과 버퍼가 없는 입출력 중 시스템 전체에 이로운 것은 무엇인가?
2. 버퍼에 남아 있는 데이터를 모두 출력시키는 메소드는?
3. 버퍼의 크기를 2048바이트로 하고, FileReader 스트림을 연결하여 윈도우의 c:\windows\system.ini 파일을 읽어 화면에 출력하는 프로그램을 버퍼 스트림을 이용하여 작성하라.

8.5 File 클래스

File 클래스란?

File 클래스
속성 정보
파일 관리

File 클래스는 파일이나 디렉터리에 대해, 경로명, 크기, 타입, 수정 날짜 등의 속성 정보를 제공하고, 파일 삭제, 디렉터리 생성, 파일 이름 변경, 디렉터리 내의 파일 리스트 제공 등 다양한 파일 관리 작업을 지원한다. File 클래스의 경로명은 java.io.File이다. 이름과는 달리, File 클래스에는 파일 입출력 기능은 없다. 파일을 읽고 쓰는 것은 앞서 공부한 FileInputStream, FileOutputStream, FileReader, FileWriter 등 파일 입출력 클래스를 이용해야 한다.

File 객체 생성

File 객체의 생성은 〈표 8-12〉의 생성자를 이용한다. 예를 들어 c:\Temp\test.txt 파일의 File 객체는 다음과 같이 생성한다.

```
File f = new File("c:\\Temp\\test.txt");
File f = new File("c:\\Temp", "test.txt"); // 디렉터리와 파일명을 나누어 전달
```

〈표 8-12〉
File 클래스의 생성자

메소드	설명
File(File parent, String child)	parent 디렉터리에 child 이름의 서브 디렉터리나 파일을 나타내는 File 객체 생성
File(String pathname)	pathname의 완전 경로명이 나타내는 File 객체 생성
File(String parent, String child)	parent 디렉터리에 child 이름의 서브 디렉터리나 파일을 나타내는 File 객체 생성
File(URI uri)	file:URI를 추상 경로명으로 변환하여 File 객체 생성

File 클래스를 이용한 파일 및 디렉터리 관리

파일 크기
타입
파일 삭제
이름 변경
디렉터리 생성
파일 리스트 얻기

File 클래스의 메소드를 이용하면, 파일 크기와 타입을 알아내고, 파일 삭제, 이름 변경, 디렉터리 생성, 디렉터리의 파일 리스트 얻기 등 다양한 파일 관리 작업을 수행할 수 있다. File 클래스의 주요 메소드는 〈표 8-13〉과 같다. 지금부터 File 클래스를 활용하는 사례를 알아보자.

〈표 8-13〉
File 클래스의 주요 메소

메소드	설명
boolean mkdir()	새로운 디렉터리 생성
String[] list()	디렉터리 내의 파일과 서브 디렉터리 리스트를 문자열 배열로 리턴
File [] listFiles()	디렉터리 내의 파일과 서브 디렉터리 리스트를 File [] 배열로 리턴
boolean renameTo(File dest)	dest가 지정하는 경로명으로 파일 이름 변경
boolean delete()	파일 또는 디렉터리 삭제
long length()	파일의 크기 리턴
String getPath()	경로명 전체를 문자열로 변환하여 리턴
String getParent()	파일이나 디렉터리의 부모 디렉터리 이름 리턴
String getName()	파일 또는 디렉터리 이름을 문자열로 리턴
boolean isFile()	일반 파일이면 true 리턴
boolean isDirectory()	디렉터리이면 true 리턴
long lastModified()	파일이 마지막으로 변경된 시간 리턴
boolean exists()	파일 또는 디렉터리가 존재하면 true 리턴

● 파일 크기, length()

length()는 파일이나 디렉터리의 크기를 리턴한다. 예를 들면 다음과 같다.

length()

```
File f = new File("c:\\windows\\system.ini"); // 파일 크기는 219바이트
long size = f.length(); // size = 219
```

파일이 존재하지 않거나, 디렉터리 혹은 운영체제 종속적인 장치 파일의 경우, length()는 운영체제에 따라 0을 리턴하기도 한다.

● 파일의 경로명, getName(), getPath(), getParent()

getName()
getPath()
getParent()

getName()은 파일명만, getPath()는 완전경로명을, getParent()는 부모 디렉터리를 문자열로 리턴한다. 예를 들면 다음과 같다.

```
String filename = f.getName();      // "system.ini"
String path = f.getPath();          // "c:\\windows\\system.ini"
String parent = f.getParent();      // "c:\\windows"
```

● 파일 타입 판별, isFile()과 isDirectory()

isFile()
isDirectory()

isFile()과 isDirectory()는 경로명이 파일인지 디렉터리인지에 따라 true/false를 리턴한다. 예를 들면 다음과 같다.

```
if(f.isFile()) // 파일인 경우
    System.out.println(f.getPath() + "는 파일입니다.");
else if(f.isDirectory())  // 디렉터리인 경우
    System.out.println(f.getPath() + "는 디렉터리입니다.");
```

system.ini는 당연히 파일이므로 아래와 같이 출력된다.

```
c:\windows\system.ini는 파일입니다.
```

● 디렉터리에 있는 파일 리스트 얻기, listFiles()

서브디렉터리의 리스트
list()
listFiles()

File 객체가 디렉터리의 경로명을 가진 경우, 디렉터리의 모든 파일과 서브디렉터리의 리스트를 얻을 수 있다. list()는 파일과 서브디렉터리 경로명을 문자열 배열(String[])로 리턴하는 반면, listFiles()는 파일과 서브디렉터리 경로명을 File[] 배열로 리턴한다. 다음 코드는 listFiles() 메소드를 이용하여 c:\Temp 디렉터리에 있는 모든 파일과 서브디렉터리의 이름을 출력한다.

```
File f = new File("c:\\Temp");
File[] subfiles = f.listFiles(); // c:\Temp 디렉터리의 파일 및 서브디렉터리 리스트 얻기
for(int i=0; i<filenames.length; i++) {
    System.out.print(subfiles[i].getName()); // 파일명 출력
    System.out.println("\t파일 크기: " + subfiles[i].length()); // 파일 크기 출력
}
```

File 클래스 활용한 파일 관리 예제 8-8

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

```
1  import java.io.File;
2  public class FileEx {
3
4     public static void listDirectory(File dir) {
5        System.out.println("-----" + dir.getPath() + "의 서브 리스트 입니다.-----");
6
7        File[] subFiles = dir.listFiles(); // 디렉토리에 포함된 파일과 디렉토리 이름의
                                            리스트 얻기
8        for(int i=0; i<subFiles.length; i++) { // subFiles 배열의 각 File에 대해 루프
9           File f = subFiles[i];
10          long t = f.lastModified(); // 마지막으로 수정된 시간
11          System.out.print(f.getName());
12          System.out.print("\t파일 크기: " + f.length()); // 파일 크기
13          System.out.printf("\t수정한 시간: %tb %td %ta %tT\n",t, t, t, t);
                                                                // 포맷 출력
14       }
15    }
16
17    public static void main(String[] args) {
18       File f1 = new File("c:\\windows\\system.ini");
19       System.out.println(f1.getPath() + ", " + f1.getParent() + ", " + f1.getName());
20
21       String res="";
22       if(f1.isFile()) res = "파일"; // 파일 타입이면
23       else if(f1.isDirectory()) res = "디렉토리"; // 디렉토리 타입이면
24       System.out.println(f1.getPath() + "은 " + res + "입니다.");
25
26       File f2 = new File("c:\\Temp\\java_sample"); // 새로 만들고자 하는 디렉터리
27       if(!f2.exists()) { // f2 디렉터리가 존재하는지 검사
28          f2.mkdir(); // 존재하지 않으면 디렉토리 생성
29       }
30
31       listDirectory(new File("c:\\Temp")); // c:\Temp에 있는 파일 리스트 출력
32       f2.renameTo(new File("c:\\Temp\\javasample")); // java_sample ->
                                                          javasample로 이름 변경
33       listDirectory(new File("c:\\Temp")); // javasample로 변경한 후 리스트 출력
34    }
35 }
```

디렉토리에 포함된 파일과 서브 디렉토리의 이름, 크기, 수정 시간을 출력하는 메소드

실행 결과

```
c:\windows\system.ini, c:\windows, system.ini
c:\windows\system.ini은 파일입니다.
-----c:\Temp의 서브 리스트 입니다.-----
Calc.class     파일          파일 크기: 754       수정한 시간: 3월 17 금 18:13:39
Calc.java      파일          파일 크기: 282       수정한 시간: 3월 17 금 18:13:25
hangul.txt     파일          파일 크기: 14        수정한 시간: 4월 03 월 20:58:51
java_sample    디렉터리      파일 크기: 0         수정한 시간: 4월 04 화 15:32:22
test.out       파일          파일 크기: 6         수정한 시간: 4월 04 화 11:32:10
test.txt       파일          파일 크기: 13        수정한 시간: 4월 03 월 21:17:51
test2.txt      파일          파일 크기: 8         수정한 시간: 4월 04 화 15:05:08
-----c:\Temp의 서브 리스트 입니다.-----
Calc.class     파일          파일 크기: 754       수정한 시간: 3월 17 금 18:13:39
Calc.java      파일          파일 크기: 282       수정한 시간: 3월 17 금 18:13:25
hangul.txt     파일          파일 크기: 14        수정한 시간: 4월 03 월 20:58:51
javasample     디렉터리      파일 크기: 0         수정한 시간: 4월 04 화 15:32:22
test.out       파일          파일 크기: 6         수정한 시간: 4월 04 화 11:32:10
test.txt       파일          파일 크기: 13        수정한 시간: 4월 03 월 21:17:51
test2.txt      파일          파일 크기: 8         수정한 시간: 4월 04 화 15:05:08
```

c:\Temp의 파일과 서브 디렉터리 리스트

java_sample을 javasample로 변경한 이후

8.6 파일 입출력 응용: 파일 복사

이 절에서는 파일 입출력 스트림의 응용으로서 파일을 복사하는 프로그램 작성에 대해 소개한다. 파일은 문자들로 이루어진 텍스트 파일과 이미지나 오디오 혹은 그래픽 등 바이너리 코드로 이루어진 바이너리 파일로 나뉜다. 바이너리 파일은 바이트 스트림으로 읽고 써야 정확하게 복사가 이루어지지만, 텍스트 파일은 문자 스트림이나 바이트 스트림 중 어떤 것을 사용해도 복사에 문제가 없다.

텍스트 파일 복사

FileReader
FileWriter

문자 스트림을 이용하여 텍스트 파일을 복사해보자. 예제 8-9는 텍스트 파일을 복사하는 응용프로그램을 보여준다. FileReader를 이용하여 텍스트 파일을 읽고 FileWriter로 텍스트 파일에 복사한다. 파일 경로명은 File 객체를 이용한다. 이 예제로는 이미지 파일이나 워드 파일, PPT 파일, 한글 파일(.hwp) 등을 복사할 수 없다.



텍스트 파일 복사 예제 8-9

문자 스트림 FileReader와 FileWriter를 이용하여 c:\windows\system.ini를 c:\Temp\system.txt 파일로 복사하는 프로그램을 작성하라.

```
1  import java.io.*;
2
3  public class TextCopyEx {
4     public static void main(String[] args) {
5        File src = new File("c:\\windows\\system.ini"); // 원본 파일 경로명
6        File dest = new File("c:\\Temp\\system.txt"); // 복사 파일 경로명
7
8        int c;
9        try {
10          FileReader fr = new FileReader(src); // 파일 입력 문자 스트림 생성
11          FileWriter fw = new FileWriter(dest); // 파일 출력 문자 스트림 생성
12          while((c = fr.read()) != -1) { // 문자 하나 읽고
13             fw.write((char)c); // 문자 하나 쓰고
14          }
15          fr.close();
16          fw.close();
17          System.out.println(src.getPath()+ "를 " + dest.getPath()+ "로 복사하였습니다.");
18       } catch (IOException e) {
19          System.out.println("파일 복사 오류");
20       }
21    }
22 }
```

fr.read()는 파일 끝을 만나면 -1 리턴

실행 결과

```
c:\windows\system.ini를 c:\Temp\system.txt로 복사하였습니다.
```

예제 8-9의 복사 응용프로그램은 한 바이트씩 복사하기 때문에, 파일이 큰 경우 복사 시간이 오래 걸린다. 파일 복사의 속도를 높이기 위해서는 블록 단위로 읽고 쓰야 한다. 예제 8-11은 블록 단위로 복사하는 고속 복사 응용프로그램을 보여준다.

바이너리 파일 복사

바이트 스트림을 이용하여 바이너리 파일을 복사해보자. 예제 8-10은 바이너리 파일을 복사하는 응용프로그램 사례이다. 이 응용프로그램은 파일을 바이트 단위로 복사하므로 이미지, 동영상, 실행 파일(exe)과 같은 바이너리 파일뿐 아니라 텍스트 파일도 복사할 수 있다.

예제 8-10 바이너리 파일 복사

FileInputStream과 FileOutputStream을 이용하여 이미지 파일을 복사하라.

윈도우 10의 배경 이미지가 있는 폴더가 바뀔 수 있으니 주의

윈도우 10에서 배경으로 사용하는 이미지는 c:\Windows\Web\Wallpaper\Theme1\ 폴더에 있으며, 윈도우 7까지는 사진 이미지들이 c:\\Users\\Public\\Pictures\\Sample Pictures\ 폴더에 있기 때문에, 사용자 컴퓨터에 설치된 운영체제에 따라 예제에서 사용할 원본 이미지를 선택하면 된다. 저자는 윈도우 10에서 이 예제를 실행하였다.

```
1  import java.io.*;
2
3  public class BinaryCopyEx {
4     public static void main(String[] args) {
5        File src = new File("c:\\Windows\\Web\\Wallpaper\\Theme1\\img1.jpg");
                                                                  // 원본 파일 경로명
6        // 윈도우 7에서는 File("c:\\Users\\Public\\Pictures\\Sample Pictures\\
            desert.jpg");를 사용하라.
7
8        File dest = new File("c:\\Temp\\copyimg.jpg"); // 복사 파일 경로명
9
10       int c;
11       try {
12          FileInputStream fi = new FileInputStream(src); // 파일 입력 바이트 스트림 생성
13          FileOutputStream fo = new FileOutputStream(dest); // 파일 출력 바이트 스트림
                                                                  생성
14          while((c = fi.read()) != -1) {
15             fo.write((byte)c);
16          }
17          fi.close();
18          fo.close();
19          System.out.println(src.getPath()+ "를 " + dest.getPath()+ "로 복사하였습니다.");
20       } catch (IOException e) {
21          System.out.println("파일 복사 오류");
22       }
23    }
24 }
```

한 바이트씩 복사하므로 복사 시간이 많이 걸림에 유의하라. 고속 복사는 다음 절을 참고하라.

fi.read()는 파일 끝을 만나면 -1 리턴

실행 결과

```
c:\Windows\Web\Wallpaper\Theme1\img1.jpg를 c:\Temp\copyimg.jpg로 복사하였습니다.
```



블록 단위로 파일 고속 복사

예제 8-9나 8-10은 한 바이트씩 읽고 쓰기 때문에, 독자들이 실행해보면 알겠지만 복사 시간이 오래 걸린다. 복사 속도를 높이려면 BufferedInputStream과 BufferedOutputStream을 사용하든지, 아니면 블록 단위로 읽고 쓰도록 수정하면 된다. 예제 8-11은 10KB의 버퍼 메모리를 이용하여 한 번에 10KB씩 복사하는 사례를 보여준다.



블록 단위로 바이너리 파일 고속 복사 — 예제 8-11

예제 8-10을 10KB 단위로 읽고 쓰도록 수정하여 고속으로 파일을 복사하라.

```
1  import java.io.*;
2  public class BlockBinaryCopyEx {
3     public static void main(String[] args) {
4        File src = new File("c:\\Windows\\Web\\Wallpaper\\Theme1\\img1.jpg");
5        // 윈도우 7에서는 File("c:\\Users\\Public\\Pictures\\Sample Pictures\\
            desert.jpg");를 사용하라.
6
7        File dest = new File("c:\\Temp\\desert.jpg"); // 복사 파일
8        try {
9           FileInputStream fi = new FileInputStream(src); // 파일 입력 바이트 스트림
10          FileOutputStream fo = new FileOutputStream(dest); // 파일 출력 바이트 스트림
11          byte [] buf = new byte [1024*10]; // 10KB 버퍼
12          while(true) {
13             int n = fi.read(buf); // 버퍼 크기만큼 읽기. n은 실제 읽은 바이트
14             fo.write(buf, 0, n); // buf[0]부터 n 바이트 쓰기
15             if(n < buf.length)
16                break; // 버퍼 크기보다 작게 읽었기 때문에 파일 끝에 도달. 복사 종료
17          }
18          fi.close();
19          fo.close();
20          System.out.println(src.getPath() + "를 " + dest.getPath() + "로 복사하
                                 였습니다.");
21       } catch (IOException e) { System.out.println("파일 복사 오류"); }
22    }
23 }
```

원본 파일 경로명

한 번에 10KB씩 블록 단위로 고속 복사

실행 결과

```
c:\Windows\Web\Wallpaper\Theme1\img1.jpg를 c:\Temp\copyimg.jpg로 복사하였습니다.
```



1 File 클래스에 대한 설명으로 틀린 것은?

① 디렉터리를 생성할 수 있다. ② 파일을 삭제할 수 있다.

③ 파일의 내용을 읽을 수 있다. ④ 파일의 이름을 변경할 수 있다.

2 PPT 파일(파워포인트)을 복사하기 위해서는 FileReader/FileWriter를 이용하는 것 FileInputStream/ FileOutputStream을 이용하는 것 중 어떤 것이 적합한가?

3 c:\Temp\test.txt 파일이 존재하면 "exist"를 아니면 "no"를 출력하는 프로그램 소스를 보여라

4 c:\Temp\test.txt 파일을 삭제하는 프로그램 소스를 보여라.

5 c:\Temp\test.txt 파일의 크기를 출력하는 프로그램 소스를 보여라.

6 c:\Temp\test.txt 파일을 c:\Temp\res.txt로 파일 이름을 변경하는 프로그램 소스를 보여라.



SUMMARY

요약

- 입출력 스트림은 입출력 장치와 프로그램 사이의 일련의 데이터 흐름을 의미한다.
- 문자 스트림은 유니코드로 된 문자 단위의 데이터가 흐르는 스트림이다. 문자가 아닌 정보는 문자 스트림의 데이터가 될 수 없다.
- 문자 스트림을 다루는 자바의 모든 클래스는 추상 클래스인 `Reader`와 `Writer` 클래스의 서브 클래스이다.
- 문자 스트림은 문자 정보를 가진 텍스트 파일만 처리할 수 있다. 바이너리 파일의 입출력에는 사용할 수 없다.
- 문자 입력 스트림을 다루는 자바 클래스에는 `Reader`, `InputStreamReader`, `FileReader` 등이 있다.
- 문자 출력 스트림을 다루는 자바 클래스에는 `Writer`, `OutputStreamWriter`, `FileWriter` 등이 있다.
- 바이트 스트림은 8비트 바이트 단위의 데이터가 흐르는 스트림이다.
- 바이트 스트림을 다루는 자바의 모든 클래스는 추상 클래스인 `InputStream`과 `OutputStream`의 서브 클래스이다.
- 바이트 스트림은 텍스트 파일의 입출력에도 사용 가능하지만 이미지나 동영상과 같은 바이너리 데이터를 입출력하는 데 보다 적합하다.
- 바이트 입력 스트림을 다루는 자바 클래스에는 `InputStream`, `FileInputStream`, `DataInputStream` 등이 있다.
- 바이트 출력 스트림을 다루는 자바 클래스에는 `OutputStream`, `FileOutputStream`, `DataOutputStream` 등이 있다.
- 버퍼 입출력(`Buffered I/O`)은 입출력 스트림과 프로그램 사이에 버퍼를 두어 입출력 장치의 속도와 프로그램의 처리 속도의 완충 작용을 수행하므로 효율적으로 입출력을 처리한다.
- 바이트 버퍼 스트림을 다루는 자바 클래스는 `BufferedInputStream`과 `BufferedOutputStream`이 있다.
- 문자 버퍼 스트림을 다루는 자바 클래스는 `BufferedReader`와 `BufferedWriter`가 있다.
- `File` 클래스는 파일과 디렉터리 경로명의 추상적 표현이다.
- `File` 클래스는 파일 삭제, 파일 이름 변경, 디렉터리 생성 등과 같은 파일 관리 작업을 수행한다.
- `File` 클래스는 파일 입출력을 위한 기능은 제공하지 않는다.



행맨(HangMan) 게임 만들기

Oper
Challenge

목적

파일 읽기, 벡터 다루기

행맨 게임을 만들면서 파일 입출력을 연습해보자. 행맨은 많이 알려져 있는 전통 있는 게임이다. 2514
의 영단어를 가진 words.txt 파일을 읽고, 사용자 모르게 영어 단어 하나를 선택한다. 그리고 이 단어에
몇 개의 글자를 숨긴 다음 화면에 출력하여 사용자가 단어를 맞추게 하는 게임이다. 숨긴 글자의 수가
으면 그만큼 난이도가 높다. 여기서는 2개의 글자만 숨기도록 한다. 한 단어에 대해 5번 틀리면 프로그
을 종료한다. words.txt 파일은 독자들에게 배포된 자료에 들어 있으며, 한 줄에 하나의 영어 단어가 적
있다. 난이도 7

```
지금부터 행맨 개임을 시작합니다.
o-colog-
>>y
o-cology
>>n
oncology
Next(y/n)?y
b-ckpl-ne
>>a
backplane
Next(y/n)?y
n-dula-
>>k
n-dula-
>>o
nodula-
>>s
nodula-
>>p
nodula-
>>w
nodula-
>>b
5번 실패 하였습니다.
nodular
Next(y/n)?
```

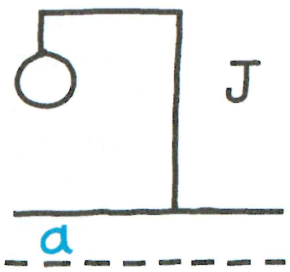
힌트 Hint

- words.txt 파일의 모든 단어를 읽어 다음의 스트링 벡터에 저장한다.

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

- words.txt 파일에는 한 라인에 하나의 영어 단어가 들어 있으며, 다음과 같이 Scanner를 이용하여 한 라인씩 읽을 수 있다.

```
Scanner scanner = new Scanner(new FileReader("words.txt"));
while(scanner.hasNext()) { // 파일의 끝까지 반복하여 읽는다.
    String word = scanner.nextLine(); // 한 라인 단위로 읽는다. word에 '\n'은 없음
    wordVector.add(word); // 단어를 벡터에 저장한다.
}
```



연습문제 EXERCISE



이론문제

• 홀수 문제는 정답이 공개됩니다.

1. 스트림에 대한 설명으로 틀린 것은?
   ① 입출력이 동시에 되는 스트림은 없다.
   ② 스트림은 다른 스트림과 연결될 수 없다.
   ③ 스트림은 먼저 들어온 데이터를 먼저 내보는 방식이다
   ④ 스트림은 바이트를 다루는 스트림과 문자만 다루는 스트림으로 나누어진다.

2. 다음 응용에서 어떤 입출력 스트림 클래스를 사용하여야 하는가?
   (1) 음악 파일 연주 (2) PPT 파일 복사
   (3) 영어와 한글이 기록된 사전 읽기 (4) 선수 이름과 점수를 문자열로 저장

3. 이미지 파일을 읽으려고 한다. 가장 적합한 클래스는?
   ① InputStreamReader ② File
   ③ FileReader ④ FileInputStream

4. 다음은 파일을 읽기 위해 스트림을 생성하는 코드이다. 이 코드가 실행될 때 발생할 수 있는 예외가 어떤 것이 있는지 설명하고, try-catch 블록으로 감싸는 코드를 작성하라.

```
FileInputStream fin = new FileInputStream("song.mp3");
```

5. 파일의 크기를 알려고 할 때 필요한 클래스는?
   ① File ② FileSize
   ③ System ④ FileInputStream

6. File 클래스의 직접적인 기능이 아닌 것은?
   ① 파일 읽고 쓰기 ② 파일 이름 변경
   ③ 새 디렉터리 만들기 ④ 파일 삭제

7. 다음 코드에 대해 물음에 답하라.

```
File file = new File("c:\\windows\\system.ini");
```

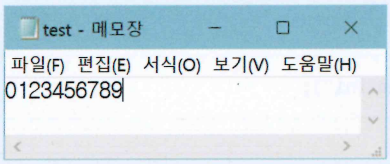
(1) file.isFile()의 리턴 값은?
(2) file.getParent()의 리턴 값은?
(3) file.getPath()의 리턴 값은?
(4) file.getName()의 리턴 값은?
(5) 아래 빈칸을 채워 문제의 코드와 동일한 코드를 구성하라.

```
File file = new File(_____, "system.ini");
```

8. 문자 집합에 대한 설명으로 틀린 것은?
① 문자들의 인코딩 정보로 구성된 문자 인코딩 세트를 뜻한다.
② 텍스트 파일 읽기 시 문자 집합을 알려주지 않으면 제대로 읽지 못하는 경우가 발생한다.
③ 바이트 스트림 입출력 시 문자 집합을 고려하여 입출력해야 한다.
④ 윈도우에서 작성된 텍스트 파일은 기본적으로 "MS949" 문자 집합으로 인코딩된다.

9. c:\temp\test.txt 파일을 읽어 화면에 출력하는 코드이다. 빈칸을 완성하라.

```
FileInputStream fin = null;
try {
    fin = new FileInputStream(_____);
    int c;
    while(true) {
        c = fin.read(); // 파일에서 한 바이트 읽기
        if (_____)  break; // 파일 끝까지 읽었음
        System.out.print((char)c);
    }
    _____; // 파일 입력 스트림 닫기
} catch (_____) {
    System.out.println("파일을 찾을 수 없습니다.");
} catch (IOException e) {
    System.out.println("입출력 오류가 발생했습니다.");
}
```




```
0123456789
```

10. 다음은 50 바이트 크기의 버퍼를 사용하여 C:\temp\sample.txt를 읽어 화면에 출력하는 소스 코드의 일부이다. 빈칸을 채워라.

```
BufferedOutputStream bout = _____;
File f = new File(_____);
FileReader fin = new FileReader(f);
int c;
while (_____) {
        _____;
}
fin.close();
bout.close();
```

실습문제

• 홀수 문제는 정답이 공개됩니다.

목적 FileWriter로 텍스트 파일 저장

1. Scanner로 입력받은 이름과 전화번호를 한 줄에 한 사람씩 c:\temp\phone.txt 파일에 저장하라. "그만"을 입력하면 프로그램을 종료한다. 난이도 4

```
전화번호 입력 프로그램입니다.
이름 전화번호 >> 황기태 010-5555-7777
이름 전화번호 >> 이재문 011-3333-4444
이름 전화번호 >> 김남윤 065-2222-1111
이름 전화번호 >> 그만
c:\temp\phone.txt에 저장하였습니다.
```

2. 앞서 저장한 c:\temp\phone.txt 파일을 읽어 화면에 출력하라. 난이도 4

목적 FileReader로 텍스트 파일 읽기

```
c:\temp\phone.txt를 출력합니다.
황기태 010-5555-7777
이재문 011-3333-4444
김남윤 065-2222-1111
```

힌트 Hint

FileReader를 이용하여 한 문자씩 읽어 출력하고자 하면 예제 8-1을 참고하고, 파일을 한 줄씩 읽고자 하면, Open Challenge의 힌트를 이용하거나 BufferedReader의 readLine() 메소드를 이용하면 된다.

3. c:\windows\system.ini 파일을 읽어 소문자를 모두 대문자로 바꾸어 출력하라.

목적 텍스트 파일 읽기

난이도 4

힌트 Hint

이 문제는 여러 가지로 풀 수 있지만 대문자로 바꾸는 방법은 다음과 같다.

```
c = Character.toUpperCase(c); // c가 문자인 경우
String line;
line = line.toUpperCase(); // 문자열 line의 경우
```

4. c:\windows\system.ini 파일에 라인 번호를 붙여 출력하라. 난이도 5

목적 텍스트 파일 읽기, Scanner와 FileReader의 연결 연습

```
c:\windows\system.ini 파일을 읽어 출력합니다.
   1: ; for 16-bit app support
   2: [386Enh]
   3: woafont=dosapp.fon
   4: EGA80WOA.FON=EGA80WOA.FON
   5: EGA40WOA.FON=EGA40WOA.FON
   6: CGA80WOA.FON=CGA80WOA.FON
   7: CGA40WOA.FON=CGA40WOA.FON
   8:
   9: [drivers]
  10: wave=mmdrv.dll
  11: timer=timer.drv
  12:
  13: [mci]
```

힌트 Hint

파일 입력 스트림과 Scanner를 이용하면 텍스트 파일을 라인 단위로 쉽게 읽을 수 있다.

```
Scanner fScanner = new Scanner(new FileReader("c:\\windows\\system.ini"));
while(scanner.hasNext()) { // 파일에 읽을 내용이 있는 동안
    String line = fScanner.nextLine(); // 파일에서 한 라인 읽기
```

또한 라인 번호를 다음과 같이 칸 맞춰 출력할 수 있다.

```
System.out.printf("%4d", lineNumber); // lineNumber는 라인 번호를 가진 int 변수
```

목적 FileInputStream과 FileOutputStream 활용 연습

5. 2개의 파일을 입력받고 비교하여 같으면 "파일이 같습니다.", 틀리면 "파일이 서로 다릅니다."를 출력하는 프로그램을 작성하라. 텍스트 및 바이너리 파일 모두를 포함한다. 아래 실행 예시에서는 프로젝트 폴더에 elvis1.txt와 복사본 elvis1-복사본.txt를 미리 준비해 두었다. 난이도 5

```
전체 경로명이 아닌 파일 이름만 입력하는 경우, 파일은 프로젝트 폴더에 있어야 합니다.
첫번째 파일 이름을 입력하세요>>elvis1.txt
두번째 파일 이름을 입력하세요>>elvis1-복사본.txt
elvis1.txt와 elvis1-복사본.txt를 비교합니다.
파일이 같습니다.
```

목적 텍스트 파일 읽기와 쓰기

6. 사용자로부터 두 개의 텍스트 파일 이름을 입력받고 첫 번째 파일 뒤에 두 번째 파일을 덧붙인 새로운 파일을 생성하는 프로그램을 작성하라. 아래 실행 예시에서는 프로젝트 폴더에 elvis1.txt와 elvis2.txt를 미리 준비해 두었다. 난이도 5

```
전체 경로명이 아닌 파일 이름만 입력하는 경우, 파일은 프로젝트 폴더에 있어야 합니다.
첫번째 파일 이름을 입력하세요>>elvis1.txt
두번째 파일 이름을 입력하세요>>elvis2.txt
프로젝트 폴더 밑에 appended.txt 파일에 저장하였습니다.
```

목적 바이너리 파일 복사 연습

7. 파일 복사 연습을 해보자. 이미지 복사가 진행하는 동안 10% 진행할 때마다 '*' 하나씩 출력하도록 하라. 아래 실행 예시에서는 프로젝트 폴더 밑에 a.jpg을 미리 준비해 두었다. 난이도 5

```
a.jpg를 b.jpg로 복사합니다.
10%마다 *를 출력합니다.
**********
```

8. File 클래스를 이용하여 c:\에 있는 파일 중에서 제일 큰 파일의 이름과 크기를 출력하라. 난이도 5

목적 File 클래스로 파일 트, 파일 크기 알아내기

```
가장 큰 파일은 c:\hiberfil.sys 3394002944바이트
```

9. c:\temp에 있는 .txt 파일만 삭제하는 프로그램을 작성하라. c:\나 c:\Windows 등의 디렉터리에 적용하면 중요한 파일이 삭제될 수 있으니 조심하라. 난이도 6

목적 File 클래스로 파일 트, 파일 타입, 파일 삭제 연

```
c:\temp디렉터리의 .txt 파일을 모두 삭제합니다....
c:\temp\p.txt 삭제
c:\temp\phone.txt 삭제
c:\temp\test.txt 삭제
총 3개의 .txt 파일을 삭제하였습니다.
```

힌트 Hint

문자열 s가 파일명을 담고 있는 경우 파일명에 .txt가 있는지 확인하려면, int index = s.lastIndexOf(".txt")를 이용하고 index가 -1이면 .txt 파일이 아닌 경우이다. 파일을 삭제하려면 다음과 같이 File 클래스의 delete() 메소드를 호출하면 된다.

```
File f = new File("c:\\temp\\p.txt");
f.delete(); // "c:\\temp\\p.txt" 파일 삭제
```

10. 전화번호를 미리 c:\temp\phone.txt 파일에 여러 개 저장해둔다. 이 파일을 읽어 다음 실행 예시와 같은 작동하는 검색 프로그램을 작성하라. 난이도 6

목적 HashMap 컬렉션, 파 출력 실전 응용

```
총 7개의 전화번호를 읽었습니다.
이름>> 최박사
010-2222-3333
이름>> 백점만
011-1100-1100
이름>> 이상아
찾는 이름이 없습니다.
이름>> 이상형
010-3333-4444
이름>> 그만
```

최박사, 백점만, 이상형 학생을 포함하여 총 7명을 미리 다음과 같이 phone.txt 파일에 저장해 두었다.

```
최박사 010-2222-3333
백점만 011-1100-1100
이상형 010-3333-4444
...
```

힌트 Hint

이름을 '키'를 하고, 전화번호를 '값'으로 하는 HashMap<String, String>을 이용하고, 한 라인씩 읽어 이름과 전화번호를 해시맵에 저장하고, 사용자의 이름을 입력받아 검색하면 된다.

목적 Vector 컬렉션, 파일 입출력 실전 응용

11. words.txt 파일(출판사 홈페이지에 자료로 제공. 현재 이 문제의 정답 프로젝트에도 있음)에는 한 라인에 하나의 영어 단어가 들어 있다. 이 파일을 한 라인씩 읽어 Vector<String>에 라인별로 삽입하여 저장하고, 영어 단어를 입력받아 그 단어로 시작하는 모든 단어를 벡터에서 찾아 출력하는 프로그램을 작성하라. 난이도 7

```
프로젝트 폴더 밑의 words.txt 파일을 읽었습니다...
단어>>lov
love
lovebird
lovelorn
단어>>kitt
kitten
kittenish
kittle
kitty
단어>>ajlfasf
발견할 수 없음
단어>>그만
종료합니다...
```

목적 Vector 컬렉션, 파일 입출력 실전 응용

12. 텍스트 파일에 있는 단어를 검색하는 프로그램을 작성해보자. 실행 예시는 프로젝트 폴더 밑에 자신이 작성한 자바 파일(예: test.java)을 복사하여 두고 읽은 경우이다. 난이도 8

```
전체 경로명이 아닌 파일 이름만 입력하는 경우, 파일은 프로젝트 폴더에 있어야 합니다.
대상 파일명 입력>> test.java
검색할 단어나 문장>> void
4: public static void main(String[] args) {
검색할 단어나 문장>> int
9:    System.out.println("전체 경로명이 아닌 파일 이름만 입력하는 경우, 파일은 프로젝트 폴더에 있어야 합니다.");
11:      System.out.print("첫번째 파일 이름을 입력하세요>>");
14:      System.out.print("두번째 파일 이름을 입력하세요>>");
17:      System.out.println(src + "와 " + dst + "를 비교합니다.");
22:           System.out.println("파일이 같습니다.");
24:           System.out.println("파일이 다릅니다.");
29:        System.out.println("입출력 오류가 발생했습니다.");
39:      int srcCount=0, destCount;
49:         for (int i=0; i<srcCount; i++) {
검색할 단어나 문장>> for (
49:         for (int i=0; i<srcCount; i++) {
검색할 단어나 문장>> 그만
프로그램을 종료합니다.
```

13. 간단한 파일 탐색기를 만들어보자. 처음 시작은 c:\에서부터 시작한다. 명령은 크게 2가지로서 ".."를 입력하면 부모 디렉터리로 이동하고, "디렉터리명"을 입력하면 서브 디렉터리로 이동하여 파일리스트를 보여준다. 난이도 8

목적 File 클래스 종합 활

```
***** 파일 탐색기입니다. *****
   [c:\]
dir     12288바이트        Program Files
dir     12288바이트        Program Files (x86)
dir     8192바이트         ProgramData

..... 생략하였습니다. .....

dir     4096바이트         Users
dir     24576바이트        Windows
dir     8192바이트         Windows10Upgrade
dir     4096바이트         황기태
>>Windows                          c:\Windows로 이동 지시
   [c:\Windows]
dir     4096바이트         AppPatch
dir     0바이트            AppReadiness
dir     4096바이트         assembly
file    0바이트            ativpsrm.bin

..... 생략하였습니다. .....

dir     0바이트            System
file    219바이트          system.ini
dir     1048576바이트      System32
dir     12288바이트        SystemApps
dir     0바이트            Web
file    167바이트          win.ini
>>Web                              c:\Windows\Web으로 이동 지시
   [c:\Windows\Web]
dir     0바이트            4K
dir     0바이트            Screen
dir     0바이트            Wallpaper
>>..                               c:\Windows로 이동 지시
   [c:\Windows]
dir     4096바이트         AppPatch
dir     0바이트            AppReadiness
dir     4096바이트         assembly
file    0바이트            ativpsrm.bin

..... 생략하였습니다. .....

dir     0바이트            System
file    219바이트          system.ini
```

```
dir     1048576바이트     System32
dir     12288바이트       SystemApps
dir     0바이트           Web
file    167바이트         win.ini
>>그만
```

힌트 Hint 예제 8-8을 참고하고 File 클래스의 listFiles() 메소드를 활용하면 어렵지 않게 작성할 수 있다.

목적 File 클래스 활용

14. 문제 13을 확장하여 다음 명령을 추가하라. 난이도 7

```
>>rename phone.txt p.txt // phone.txt를 p.txt로 변경. 파일과 디렉터리 이름 변경
>>mkdir XXX // 현재 디렉터리 밑에 XXX 디렉터리 생성
```

```
***** 파일 탐색기입니다. *****
  [c:\]
dir     12288바이트       Program Files
dir     12288바이트       Program Files (x86)
dir     8192바이트        ProgramData

..... 생략하였습니다 .....

dir     4096바이트        Users
dir     24576바이트       Windows
dir     8192바이트        Windows10Upgrade
dir     4096바이트        황기태
>>mkdir 이재문
이재문 디렉터리를 생성하였습니다.
  [c:\]
dir     12288바이트       Program Files
dir     12288바이트       Program Files (x86)
dir     8192바이트        ProgramData

..... 생략하였습니다 .....

dir     4096바이트        Users
dir     24576바이트       Windows
dir     8192바이트        Windows10Upgrade
dir     0바이트           이재문   생성된 이재문 디렉터리
dir     4096바이트        황기태
```

```
>>temp
   [c:\temp]
dir      0바이트            HncDownload
file     66바이트           phone.txt
file     10바이트           test.txt
>>rename phone.txt
두 개의 파일명이 주어지지 않았습니다.!
>>rename phone.txt p.txt
phone.txt를 p.txt 이름 변경하였습니다.
   [c:\temp]
dir      0바이트            HncDownload
file     66바이트           p.txt
file     10바이트           test.txt
>>
```

phone.txt가 p.txt로 변경

힌트 Hint

rename phone.txt p.txt와 같은 명령을 읽고 문자열을 분리하기 위해 StringTokenizer를 이용하라. 만일 rename phone.txt 이런 식으로 명령을 잘못 입력한 경우 StringTokenizer.nextToken()은 NoSuchElementException 예외를 발생시키므로 try-catch 블록을 만들어 catch 블록에서 사용자에게 오류메시지를 출력하면 된다.