(INTERMEDIATE) JAVA PROGRAMMING

14. Generic Programming

Chapter 7

JAVA:
COLLECTION

컬렉션

- 컬렉션(collection)은 자바에서 자료 구조를 구현한 클래스
- 자료 구조로는 리스트(list), 스택(stack), 큐(queue), 집합(set), 해쉬 테이블(hash table) 등이 있다.

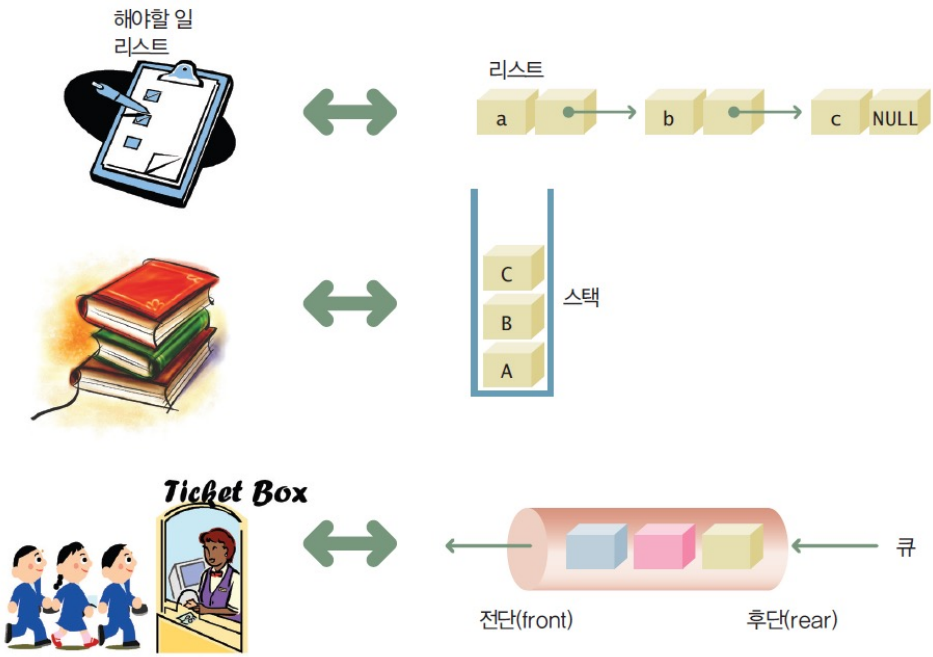
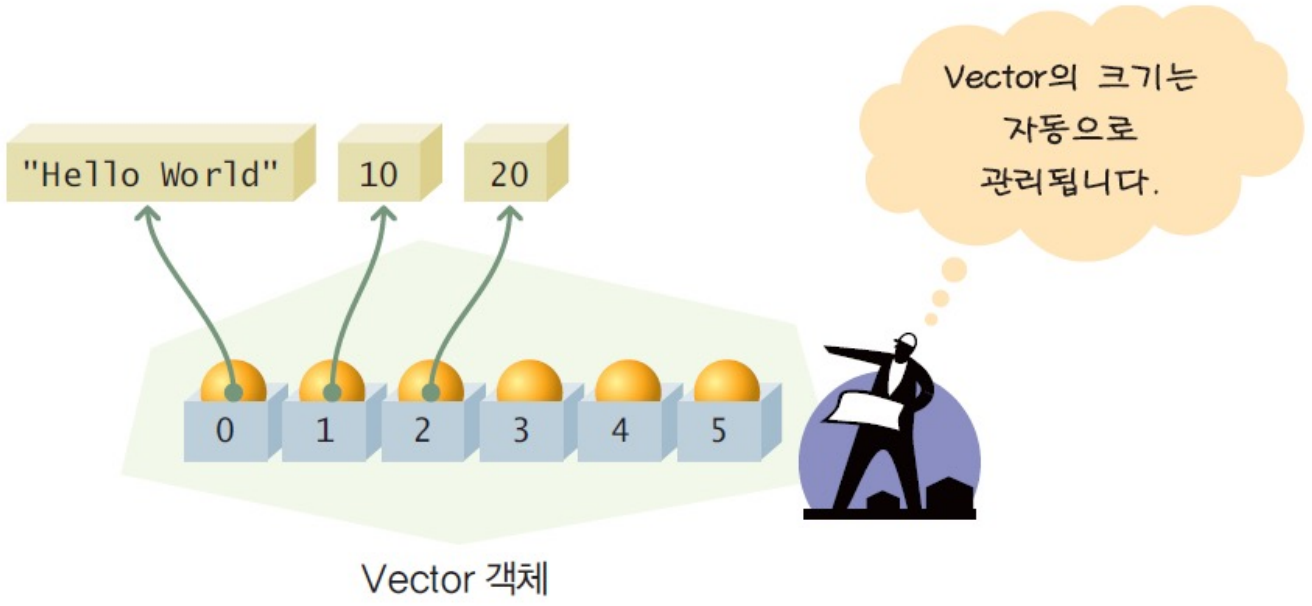


그림22-4. 자료 구조의 예

컬렉션의 예: Vector 클래스

- Vector 클래스는 java.util 패키지에 있는 컬렉션의 일종으로 가변 크기의 배열(dynamic array)을 구현



예제

VectorTest.java

```
01  import java.util.Vector;
02
03  public class VectorTest {
04
05     public static void main(String[] args) {
06
07        Vector vc = new Vector();
08
09        vc.add("Hello World!");
10        vc.add(new Integer(10));
11        vc.add(20);
12
13        System.out.println("vector size :" + vc.size());
14
15        for (int i = 0; i < vc.size(); i++) {
16           System.out.println("vector element " + i + " :" + vc.get(i));
17        }
18        String s = (String)vc.get(0);
19
20     }
21  }
```

벡터 객체를 생성할 때, 크기를 안 주어도 된다. 물론 크기를 줄 수도 있다.

어떤 타입의 객체도 추가가 가능하다.

get()은 Object 타입으로 반환하므로 형변환하여서 사용한다.

실행결과

실행결과

```
vector size :3
vector element 0 :Hello World!
vector element 1 :10
vector element 2 :20
```

컬렉션의 종류

인터페이스	설명
Collection	모든 자료 구조의 부모 인터페이스로서 객체의 모임을 나타낸다.
Set	집합(중복된 원소를 가지지 않는)을 나타내는 자료 구조
List	순서가 있는 자료 구조로 중복된 원소를 가질 수 있다.
Map	키와 값들이 연관되어 있는 사전과 같은 자료 구조
Queue	극장에서의 대기줄과 같이 들어온 순서대로 나가는 자료구조

중간점검

1. 컬렉션에는 어떤 것들이 있는가?
2. 컬렉션 클래스들은 어디에 이용하면 좋은가?

Collection 인터페이스

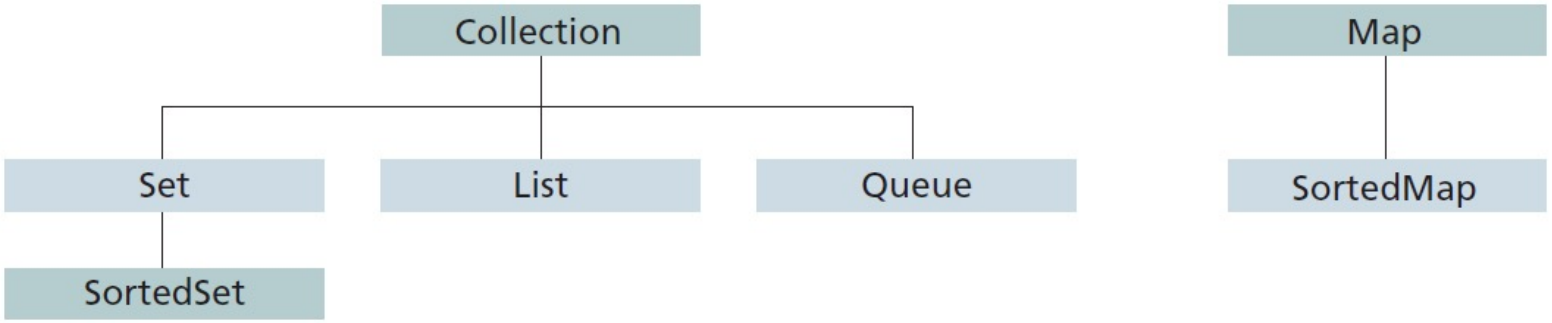
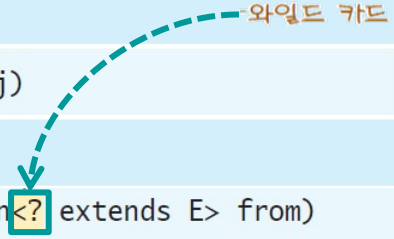


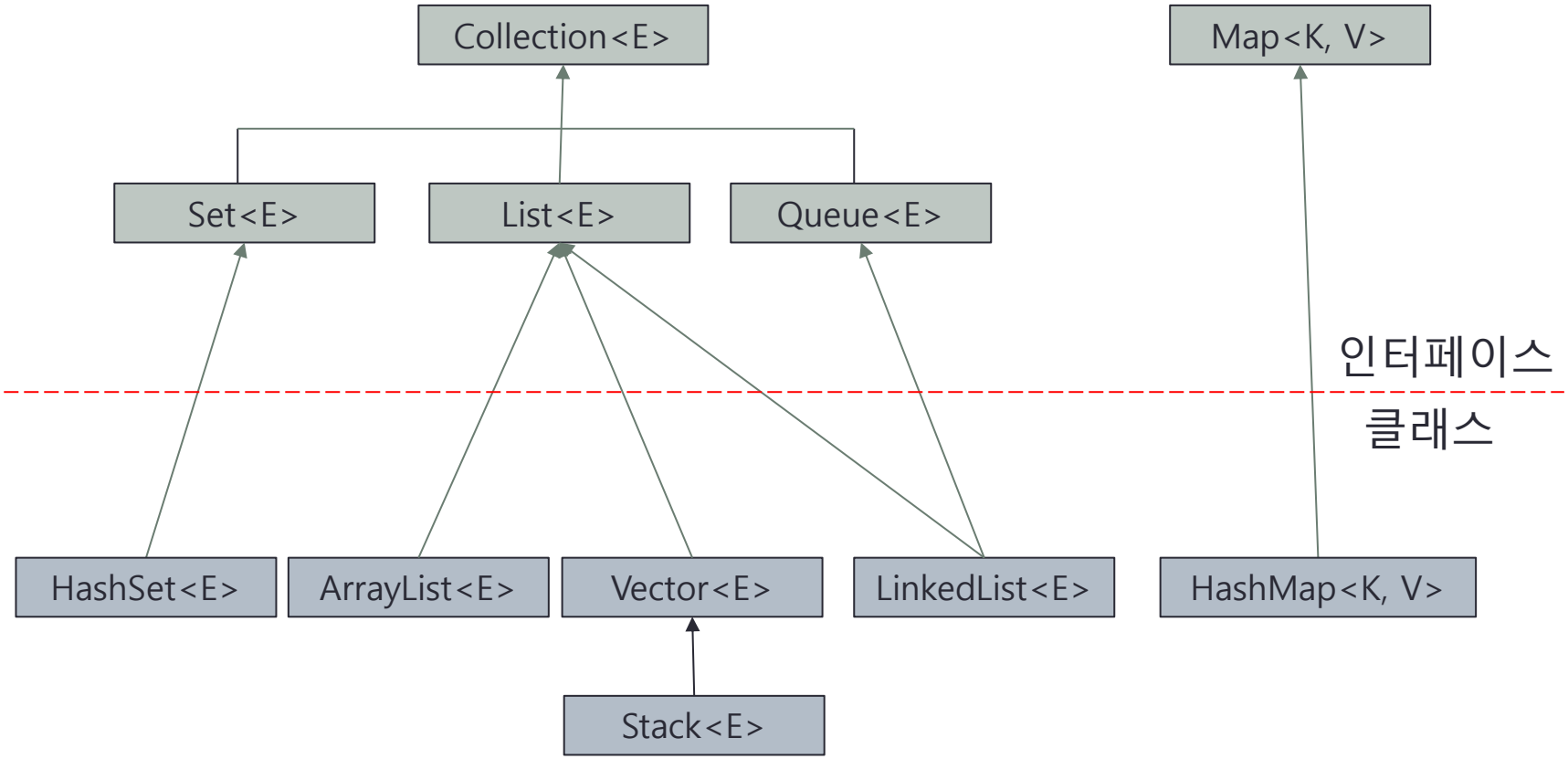
그림22-4. 인터페이스들의 계층구조

Collection 인터페이스

분류	메소드	설명
기본 연산	`int size()`	원소의 개수 반환
	`boolean isEmpty()`	공백 상태이면 true 반환
	`boolean contains(Object obj)`	obj를 포함하고 있으면 true 반환
	`boolean add(E element);`	원소 추가
	`boolean remove(Object obj)`	원소 삭제
	`Iterator<E> iterator();`	원소 방문
벌크 연산	`boolean addAll(Collection<? extends E> from)`	c에 있는 모든 원소 추가
	`boolean containsAll(Collection<?> c)`	c에 있는 모든 원소가 포함되어 있으면 true
	`boolean removeAll(Collection<?> c)`	c에 있는 모든 원소 삭제
	`void clear()`	모든 원소 삭제
배열 연산	`Object[] toArray()`	컬렉션을 배열로 변환
	`<T> T[] toArray(T[] a);`	컬렉션을 배열로 변환



컬렉션을 위한 자바 인터페이스와 클래스



중간점검

중간점검

1. Collection 인터페이스의 각 메소드들의 기능을 자바 API 웹페이지를 이용하여서 조사하여 보자.

JAVA:
ARRAYLIST

ArrayList

그림22-5. 리스트

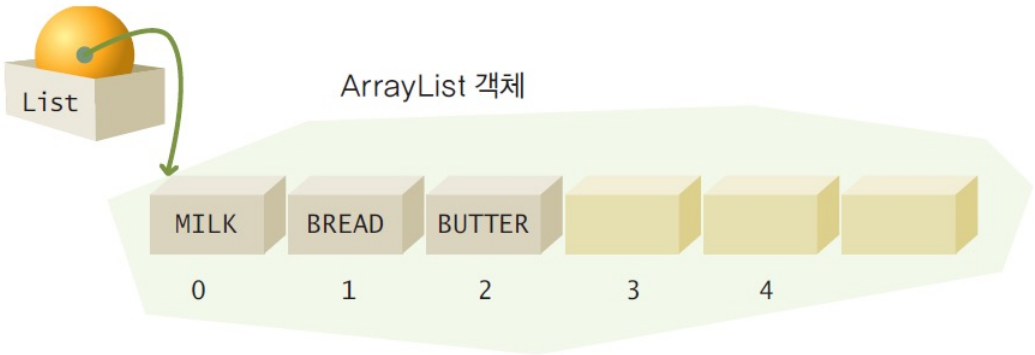
ArrayList의 기본 연산

- ArrayList 는 타입 매개변수를 가지는 제네릭 클래스로 제공된다.

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

- 생성된 ArrayList 객체에 데이터를 저장하려면 add() 메소드를 사용한다. add() 메소드는 Collection 인터페이스에 정의된 메소드로서 ArrayList 클래스가 구현한 메소드이다.

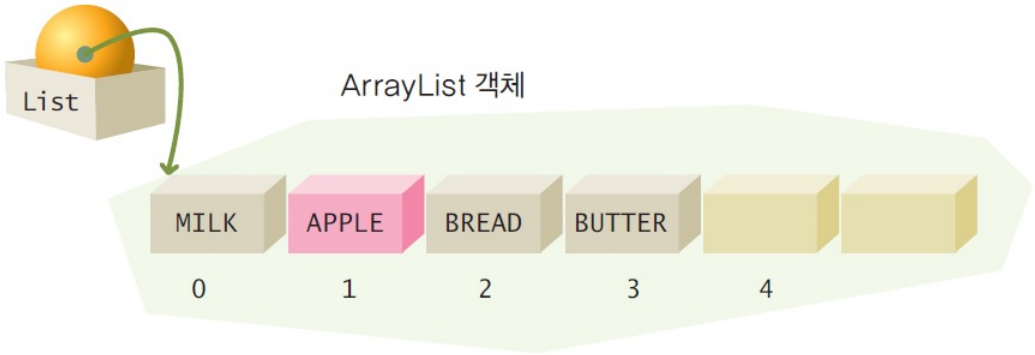
```
list.add( "MILK" );
list.add( "BREAD" );
list.add( "BUTTER" );
```



ArrayList의 기본 연산

- 만약에 기존의 데이터가 들어 있는 위치를 지정하여서 add()를 호출하면 새로운 데이터는 중간에 삽 입된다.

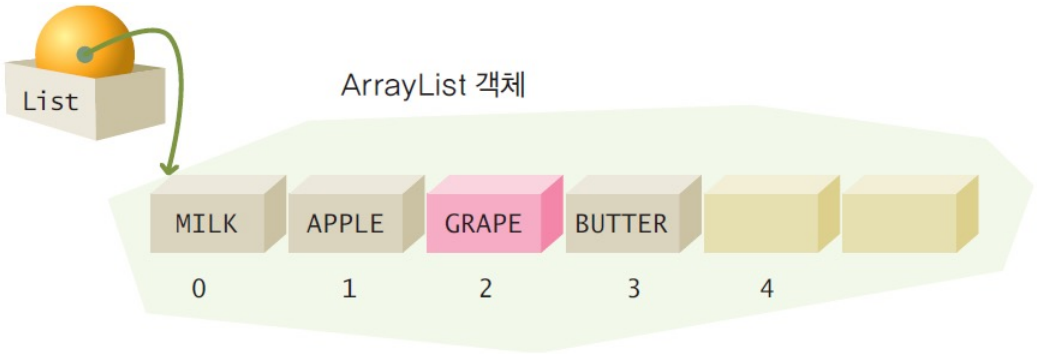
```
list.add( 1, "APPLE" );      // 인덱스 1에 "APPLE"을 삽입
```



ArrayList의 기본 연산

- 만약 특정한 위치에 있는 원소를 바꾸려면 set() 메소드를 사용한다.

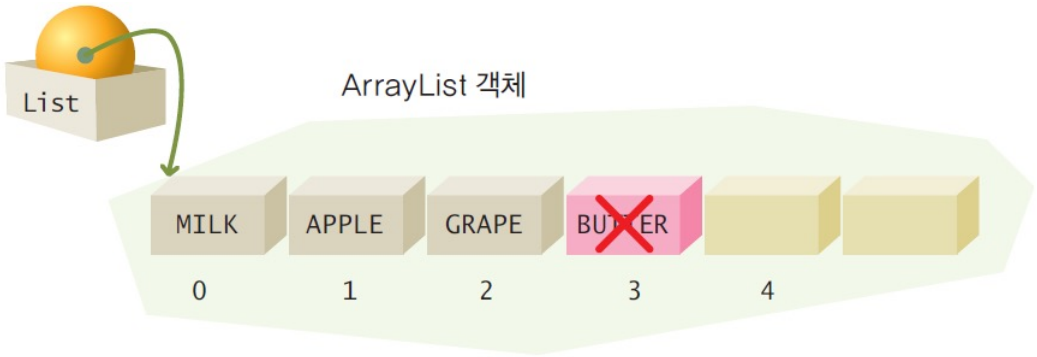
```
list.set( 2, "GRAPE" );      // 인덱스 2의 원소를 "GRAPE"로 대체
```



ArrayList의 기본 연산

- 데이터를 삭제하려면 remove() 메소드를 사용한다.

```
list.remove( 3 );            // 인덱스 3의 원소를 삭제한다.
```



- ArrayList 객체에 저장된 객체를 가져오는 메소드는 get()이다. get()은 인덱스를 받아서 그 위치에 저장된 원소를 반환한다. 예를 들어서 list.get(1)이라고 하면 인덱스 1에 저장된 데이터가 반환된다.

```
String s = list.get(1);
```

예제

ArrayListTest.java

```
01  import java.util.*;
02
03  public class ArrayListTest {
04     public static void main(String args[]) {
05        ArrayList<String> list = new ArrayList<String>();
06
07        list.add("MILK");
08        list.add("BREAD");
09        list.add("BUTTER");
10        list.add(1, "APPLE");        // 인덱스 1에 "APPLE"을 삽입
11        list.set(2, "GRAPE");        // 인덱스 2의 원소를 "GRAPE"로 대체
12        list.remove(3);              // 인덱스 3의 원소를 삭제한다.
13
14        for (int i = 0; i < list.size(); i++)
15           System.out.println(list.get(i));
16     }
17  }
```

String 타입의 객체를 저장할 수 있는 ArrayList 객체 생성

실행결과

실행결과

```
MILK
APPLE
GRAPE
```

프로그램설명

위의 코드에서는 get() 메소드의 사용을 보이기 위하여 표준적인 for 루프를 사용했지만 사실 ArrayList에 들어 있는 데이터를 모두 출력하려면 다음과 같은 for-each 루프를 사용하는 것이 좋다.

```
for (String s : list)
    System.out.println(s);
```

ArrayList의 추가 연산 indexOf

- ArrayList는 동일한 데이터도 여러 번 저장될 수 있으므로 맨 처음에 있는 데이터의 위치가 반환된다.

```
int index = list.indexOf("APPLE");    // 1이 반환된다.
```

- 검색을 반대 방향으로 하려면 lastIndexOf()를 사용한다.

```
int index = list.lastIndexOf("MILK");    // 0이 반환된다.
```

참고사항

불행하게도 자바에서는 배열, ArrayList, 문자열 객체의 크기를 알아내는 방법이 약간 다르다.

- 배열: `array.length`
- ArrayList: `arrayList.size()`
- 문자열: `string.length()`

반복자 사용하기

- ArrayList에 있는 원소에 접근하는 또 하나의 방법은 반복자**(iterator)**를 사용하는 것이다.

메소드	설명
hasNext()	아직 방문하지 않은 원소가 있으면 true를 반환
next()	다음 원소를 반환
remove()	최근에 반환된 원소를 삭제한다.

반복자 사용하기

- 반복자 객체의 hasNext()와 next() 메소드를 이용하여서 컬렉션의 각 원소들을 접근 하게 된다.

```
ArrayList<String> list = new ArrayList<String>();
list.add("하나");
list.add("둘");
list.add("셋");
list.add("넷");

String s;
Iterator e = list.iterator();
while(e.hasNext())                                    다음 원소가 있으면
{
    s = (String)e.next();          // 반복자는 Object 타입을 반환!
    System.out.println(s);                            다음 원소를 얻는다.
}
```

중간점검

참고사항

반복자 사용을 보다 간편하게 한 것이 버전 1.5부터 도입된 for-each 루프이다. 반복자보다는 for-each 루프가 간편하지만 아직도 반복자는 널리 사용되고 있다. 따라서 그 작동 원리를 알아야 한다.

중간점검

1. ArrayList가 기존의 배열보다 좋은 점은 무엇인가?
2. ArrayList의 부모 클래스는 무엇인가?
3. 왜 인터페이스 참조 변수를 이용하여서 컬렉션 객체들을 참조할까?
4. ArrayList 안의 객체들을 반복 처리하는 방법들을 모두 설명하라.

JAVA:
LINKED LIST

LinkedList

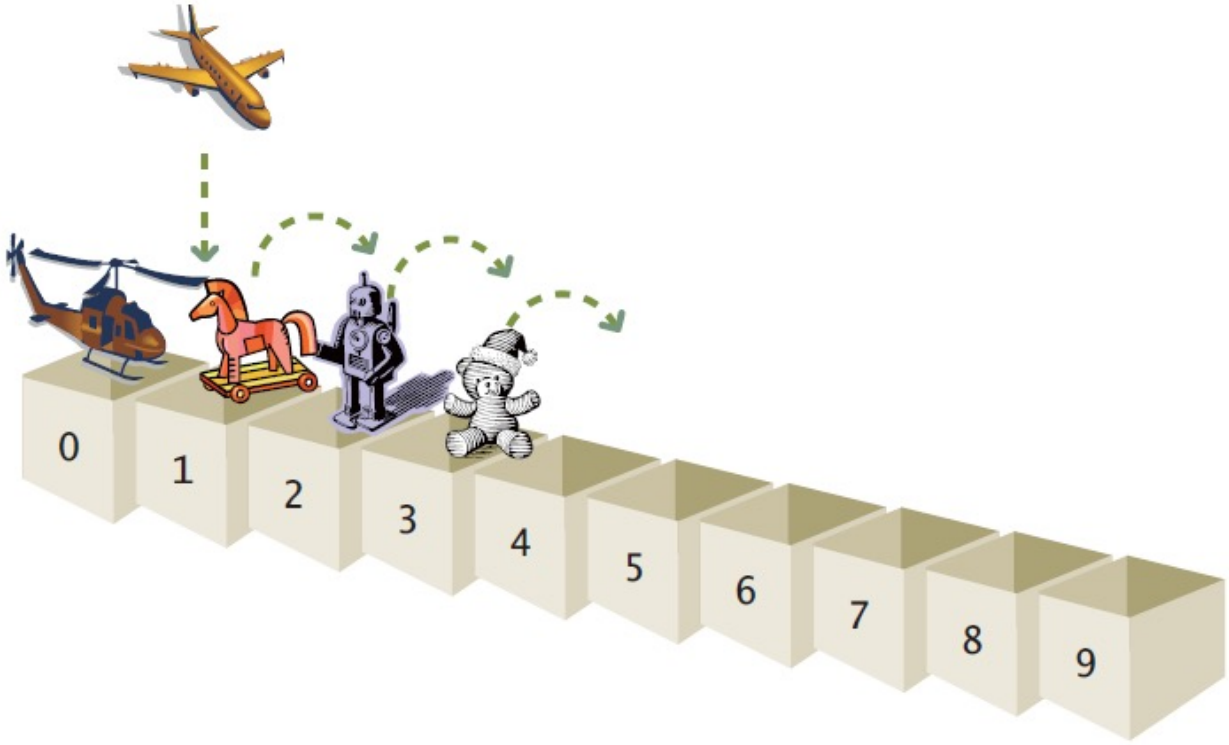


그림22-6. 배열의 중간에 삽입하려면 원소들을 이동하여야 한다.

LinkedList

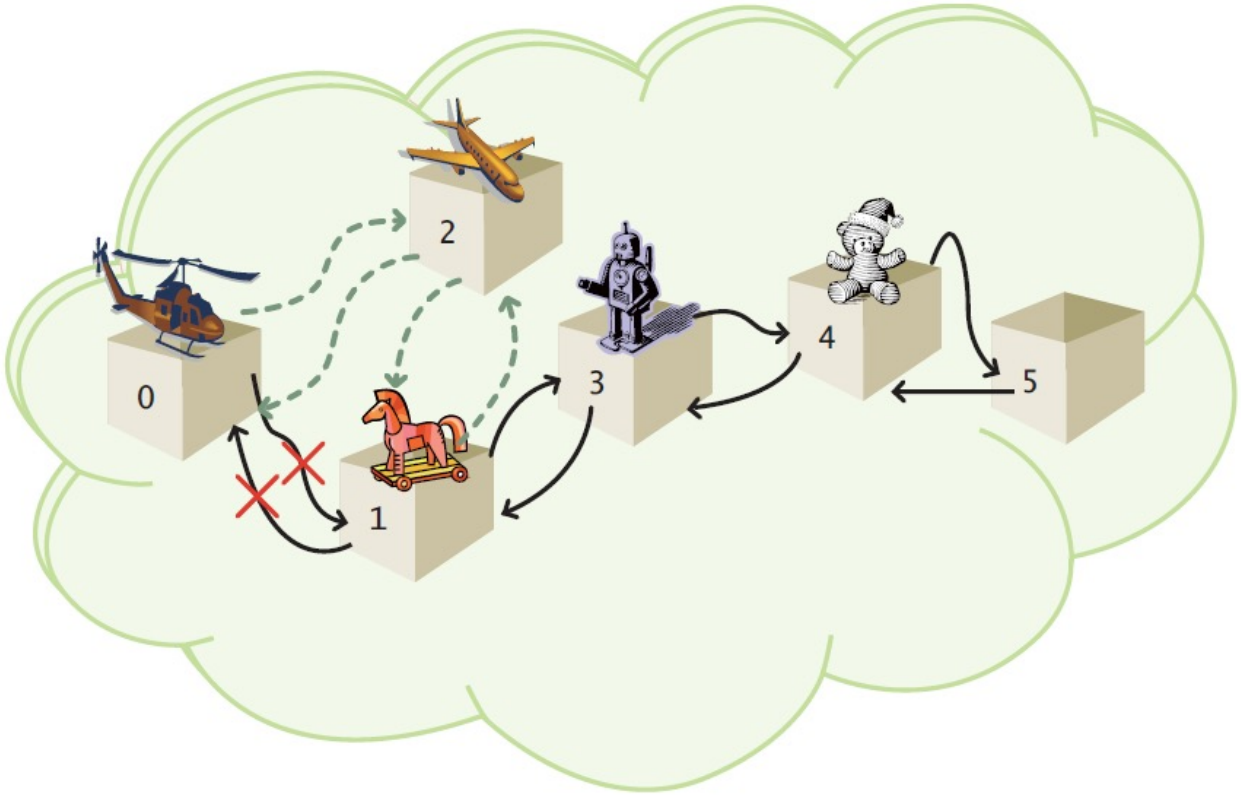


그림22-7. 연결 리스트 중간에 삽입하려면 링크만 수정하면 된다.

예제

LinkedListTest.java

```
01  import java.util.*;
02
03  public class LinkedListTest {
04     public static void main(String args[]) {
05        LinkedList<String> list = new LinkedList<String>();
06
07        list.add("MILK");
08        list.add("BREAD");
09        list.add("BUTTER");
10        list.add(1, "APPLE");        // 인덱스 1에 "APPLE"을 삽입
11        list.set(2, "GRAPE");        // 인덱스 2의 원소를 "GRAPE"로 대체
12        list.remove(3);              // 인덱스 3의 원소를 삭제한다.
13
14        for (int i = 0; i < list.size(); i++)
15           System.out.println(list.get(i));
16     }
17  }
```

실행결과

실행결과

```
MILK
APPLE
GRAPE
```

반복자 사용하기

- LinkedList도 반복자를 지원한다. 다음과 같은 형식으로 사용하면 된다

```
Iterator e = list.iterator();
String first = e.next();      // 첫 번째 원소
String second = e.next();     // 두 번째 원소
e.remove();                   // 최근 방문한 원소 삭제
```

- ArrayList나 LinkedList와 같은 리스트에서 사용하기가 편리한 반복자는 다음과 같이 정의되는 ListIterator이다.

```
interface ListIterator<E> extends Iterator<E>
{
   void add(E element);
   E previous();
   boolean hasPrevious();
   ...
}
```

- 참고자료: how to remove an item well
http://stackoverflow.com/questions/223918/iterating-through-a-list-avoiding-concurrentmodificationexception-when-removing

배열을 리스트로 변경하기

- Arrays.asList() 메소드는 배열을 받아서 리스트 형태로 반환한다.

```
List<String> list = Arrays.asList(new String[size]);
```

중간점검

1. ArrayList와 LinkedList의 차이점은 무엇인가?
2. 어떤 경우에 LinkedList를 사용하여야 하는가?

JAVA:
HASH MAP

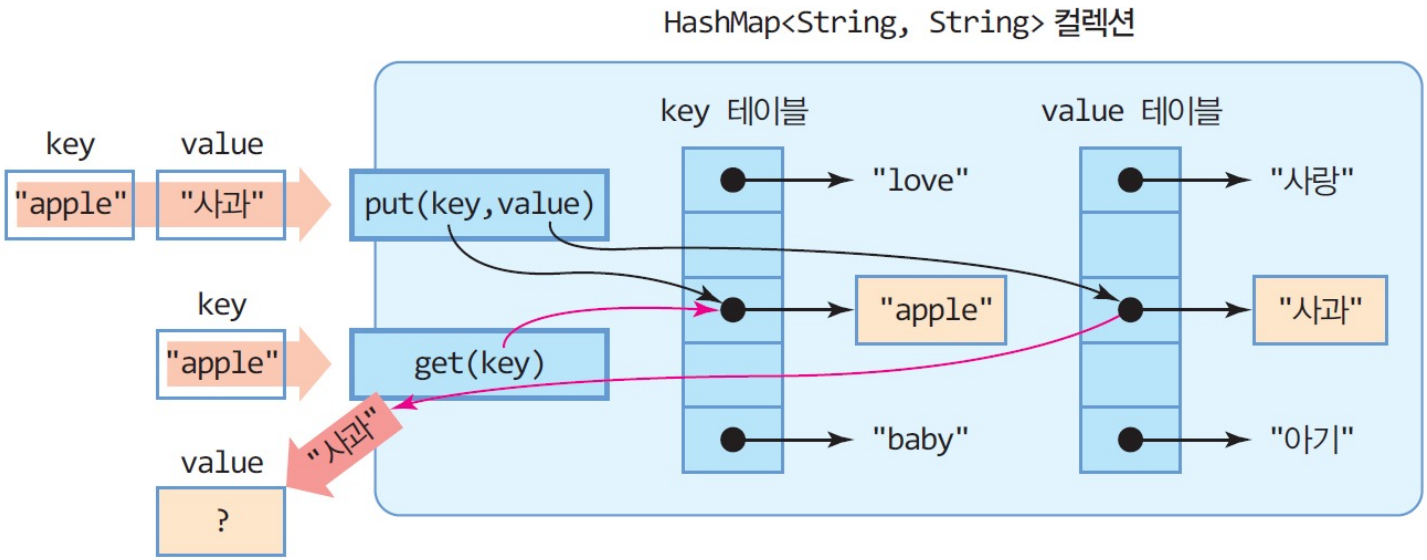
HashMap<K,V>

- HashMap<K,V>
  - 키(key)와 값(value)의 쌍으로 구성되는 요소를 다루는 컬렉션
    - java.util.HashMap
    - K는 키로 사용할 요소의 타입, V는 값으로 사용할 요소의 타입 지정
    - 키와 값이 한 쌍으로 삽입
    - 키는 해시맵에 삽입되는 위치 결정에 사용
    - 값을 검색하기 위해서는 반드시 키 이용
  - 삽입, 삭제, 검색이 빠른 특징
    - 요소 삽입 : put() 메소드
    - 요소 검색 : get() 메소드
  - 예) HashMap<String, String> 생성, 요소 삽입, 요소 검색

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

HashMap<String, String>의 내부 구성

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

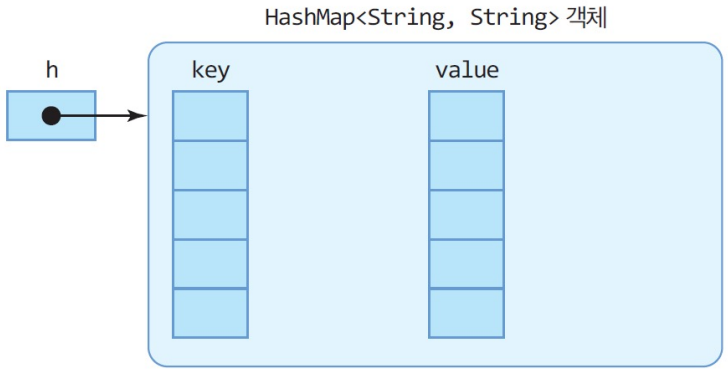


HashMap<K,V>의 주요 메소드

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

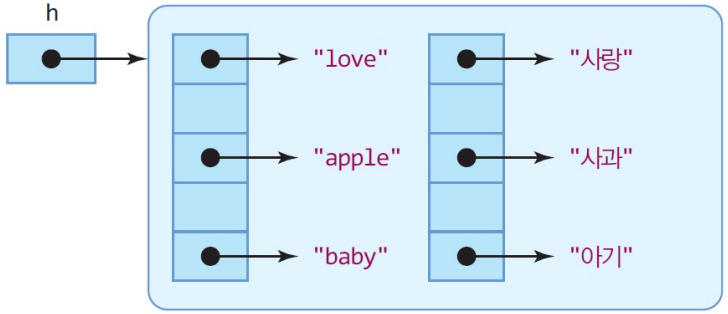
해시맵 생성

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



(키, 값) 삽입

```
h.put("baby", "아기");
h.put("love", "사랑");
h.put("apple", "사과");
```



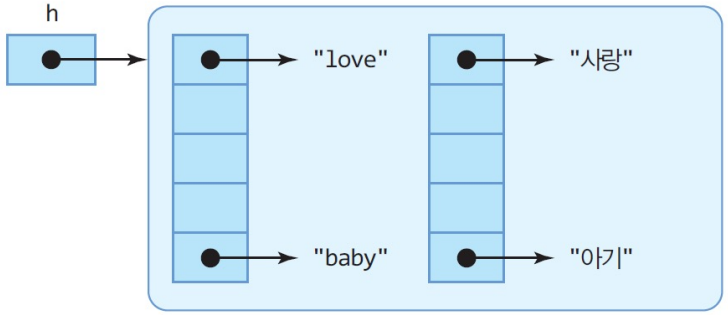
키로 값 읽기

```
String kor = h.get("love");
```

kor = "사랑"

키로 요소 삭제

```
h.remove("apple");
```



요소 개수

```
int n = h.size();
```

n = 2

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.

```
import java.util.*;

public class HashMapDicEx {
    public static void main(String[] args) {
        // 영어 단어와 한글 단어의 쌍을 저장하는 HashMap 컬렉션 생성
        HashMap<String, String> dic =
                new HashMap<String, String>();

        // 3 개의 (key, value) 쌍을 dic에 저장
        dic.put("baby", "아기"); // "baby"는 key, "아기"은 value
        dic.put("love", "사랑");
        dic.put("apple", "사과");

        // 영어 단어를 입력받고 한글 단어 검색. "exit" 입력받으면 종료
        Scanner scanner = new Scanner(System.in);
        while(true) {
            System.out.print("찾고 싶은 단어는?");
            String eng = scanner.next();
            if(eng.equals("exit")) {
                System.out.println("종료합니다...");
                break;
            }
            // 해시맵에서 '키' eng의 '값' kor 검색
            String kor = dic.get(eng);
            if(kor == null)
                System.out.println(eng +
                        "는 없는 단어 입니다.");
            else
                System.out.println(kor);
        }
        scanner.close();
    }
}
```

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

"babo"를 해시맵에서 찾을 수 없기 때문에 null 리턴

예제 7-6 HashMap을 이용하여 자바 과목의 이름과 점수 관리

해시맵을 이용하여 학생의 이름과 자바 점수를 기록 관리하는 프로그램을 작성하라

```
import java.util.*;

public class HashMapScoreEx {
    public static void main(String[] args) {
        // 사용자 이름과 점수를 기록하는 HashMap 컬렉션 생성
        HashMap<String, Integer> javaScore =
                new HashMap<String, Integer>();

        // 5 개의 점수 저장
        scoreMap.put("김성동", 97);
        scoreMap.put("황기태", 88);
        scoreMap.put("김남윤", 98);
        scoreMap.put("이재문", 70);
        scoreMap.put("한원선", 99);

        System.out.println("HashMap의 요소 개수 :"
                + javaScore.size());

        // 모든 사람의 점수 출력.
        // javaScore에 들어 있는 모든 (key, value) 쌍 출력
         // key 문자열을 가진 집합 Set 컬렉션 리턴
        Set<String> keys = javaScore.keySet();

        // key 문자열을 순서대로 접근할 수 있는 Iterator 리턴
        Iterator<String> it = keys.iterator();
        while(it.hasNext()) {
            String name = it.next();
            int score = javaScore.get(name);
            System.out.println(name + " : " + score);
        }
    }
}
```

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

예제 7-7 HashMap에 객체 저장, 학생 정보 관리

id와 전화번호로 구성되는 Student 클래스를 만들고, 이름을 '키'로 하고 Student 객체를 '값'으로 하는 해시맵을 작성하라.

```
import java.util.*;

class Student { // 학생을 표현하는 클래스
    int id;
    String tel;
    public Student(int id, String tel) {
        this.id = id; this.tel = tel;
    }
}
```

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

```
public class HashMapStudentEx {
    public static void main(String[] args) {
        // 학생 이름과 Student 객체를 쌍으로 저장하는 HashMap 컬렉션 생성
        HashMap<String, Student> map = new HashMap<String, Student>();

        // 3 명의 학생 저장
        map.put("황기태", new Student(1, "010-111-1111"));
        map.put("이재문", new Student(2, "010-222-2222"));
        map.put("김남윤", new Student(3, "010-333-3333"));

        Scanner scanner = new Scanner(System.in);
        while(true) {
            System.out.print("검색할 이름?");
            String name = scanner.nextLine(); // 사용자로부터 이름 입력
            if(name.equals("exit"))
                break; // while 문을 벗어나 프로그램 종료
            Student student = map.get(name); // 이름에 해당하는 Student 객체 검색
            if(student == null)
                System.out.println(name + "은 없는 사람입니다.");
            else
                System.out.println("id:" + student.getId() + ", 전화:" + student.getTel());
        }
        scanner.close();
    }
}
```

JAVA:
COLLECTION UTILITIES

Collections 클래스 활용

- Collections 클래스
  - java.util 패키지에 포함
  - 컬렉션에 대해 연산을 수행하고 결과로 컬렉션 리턴
  - 모든 메소드는 static 타입
  - 주요 메소드
    - 컬렉션에 포함된 요소들을 소팅하는 sort() 메소드
    - 요소의 순서를 반대로 하는 reverse() 메소드
    - 요소들의 최대, 최솟값을 찾아내는 max(), min() 메소드
    - 특정 값을 검색하는 binarySearch() 메소드

예제 7-8 : Collections 클래스의 활용

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

```
import java.util.*;

public class CollectionsEx {
        static void printList(LinkedList<String> l) {
            Iterator<String> iterator = l.iterator();
            while (iterator.hasNext()) {
                    String e = iterator.next();
                    String separator;
                    if (iterator.hasNext())
                            separator = "->";
                    else
                            separator = "\n";
                    System.out.print(e+separator);
            }
        }
```

```
            public static void main(String[] args) {
                LinkedList<String> myList = new LinkedList<String>();
            myList.add("트랜스포머");
                myList.add("스타워즈");
                myList.add("매트릭스");
                myList.add(0,"터미네이터");
                myList.add(2,"아바타");

                Collections.sort(myList); // 요소 정렬
                printList(myList); // 정렬된 요소 출력

                Collections.reverse(myList); // 요소의 순서를 반대로
                printList(myList); // 요소 출력

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

static 메소드이므로 클래스 이름으로 바로 호출

소팅된 순서대로 출력

거꾸로 출력

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

JAVA:
OTHER COLLECTION CLASSES

Queue

- 큐는 먼저 들어온 데이터가 먼저 나가는 자료 구조
- FIFO(First-In First-Out)

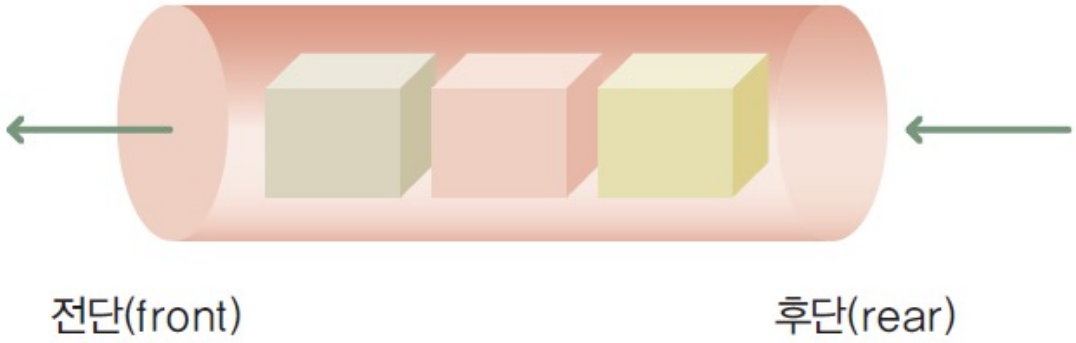


그림22-10.큐

Priority Queue

- 우선순위 큐는 원소들이 무작위로 삽입되었더라도 정렬된 상태로 원소들을 추출한다.
- remove()를 호출할 때마다 가장 작은 원소가 추출된다.
- 우선순위 큐는 힙(heap)라고 하는 자료 구조를 내부적으로 사용한다.
- 힙는 이진 트리의 일종으로서 add()와 remove()를 호출하면 가장 작은 원소가 효율적으로 트리의 루트 로 이동하게 된다.
- 우선순위 큐의 가장 대표적인 예는 작업 스케쥴링(job scheduling)이다. 각 작업은 우선순위를 가지고 있고 가장 높은 우선순위의 작업이 큐에서 먼저 추출되어서 시작된다