컬렉션

컴퓨터공학전공
박요한

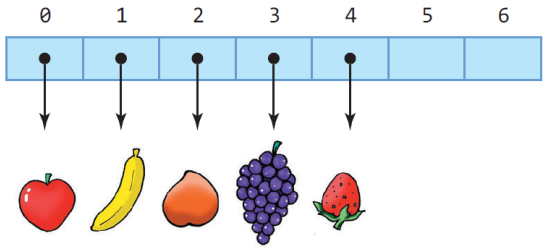
수업내용

- 컬렉션 프레임워크의 이해
- Collection<E> 인터페이스
  - ✓List
  - ✓Set
  - ✓Queue
- Map(K,V)
  - ✓Hash

컬렉션(collection)의 개념

- 컬렉션
  - ✓ 요소(element) 객체들의 저장소
    - 객체들의 컨테이너라고도 불림
    - 요소의 개수에 따라 크기 자동 조절
    - 요소의 삽입, 삭제에 따른 요소의 위치 자동 이동
  - ✓ 고정 크기의 배열을 다루는 어려움 해소
    - 저장할 수 있는 객체 수가 배열을 생성할 때 결정 -> 불특정 다수의 객체 저장 문제
    - 객체를 삭제했을 때 해당 인덱스가 비게 됨 -> 객체 저장시 빈 공간 확인 문제

배열(array)



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

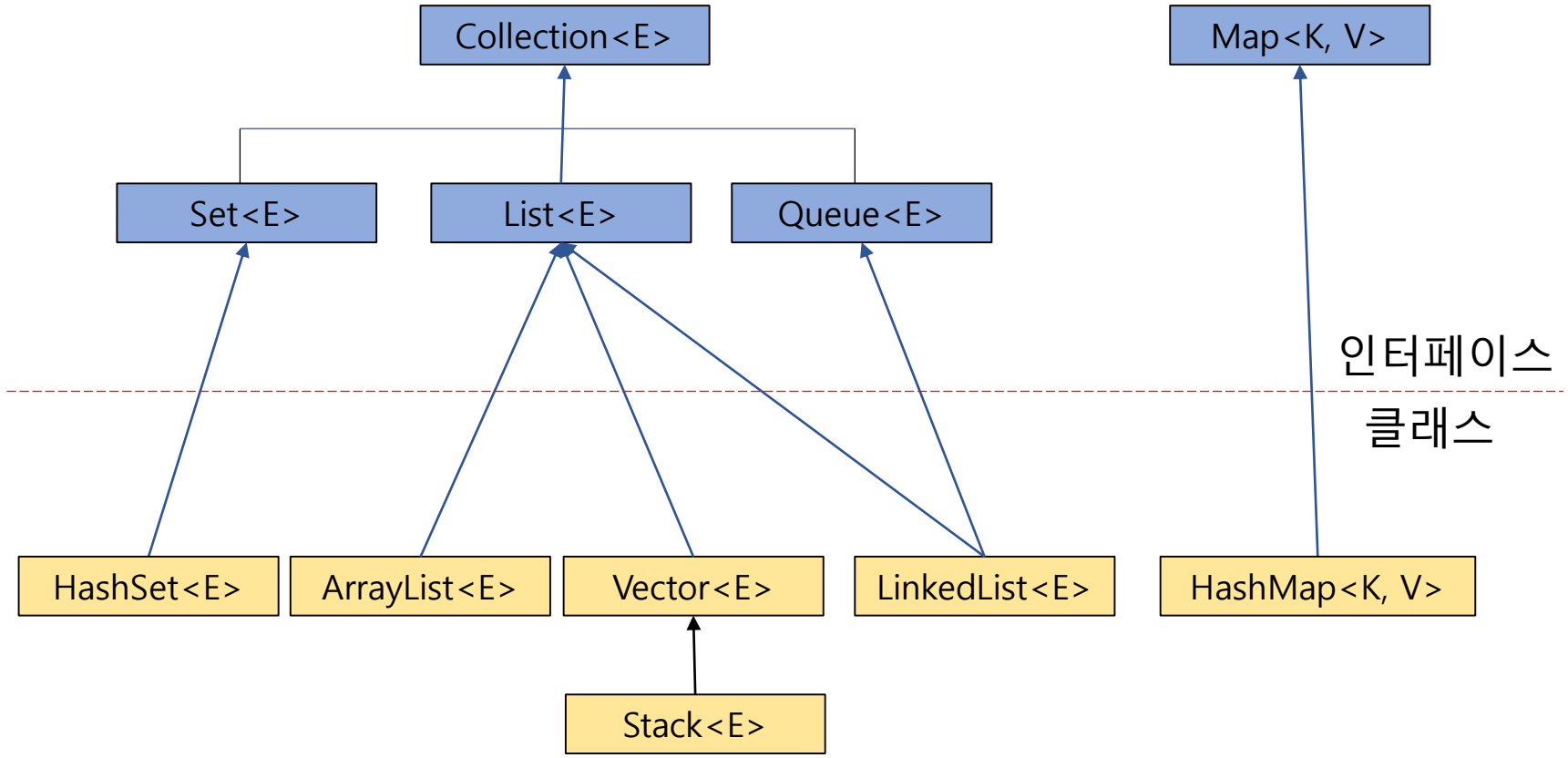
컬렉션(collection)

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

컬렉션(collection)의 개념

- 컬렉션 프레임워크
  - ✓다양한 객체들의 삽입, 삭제, 검색 등의 관리 용이
  - ✓객체들을 효과적으로 추가, 삭제, 검색할 수 있도록 제공되는 컬렉션 라이브러리
  - ✓Java.util 패키지에 포함
  - ✓인터페이스를 통해서 정형화된 방법으로 다양한 컬렉션 클래스 이용 가능

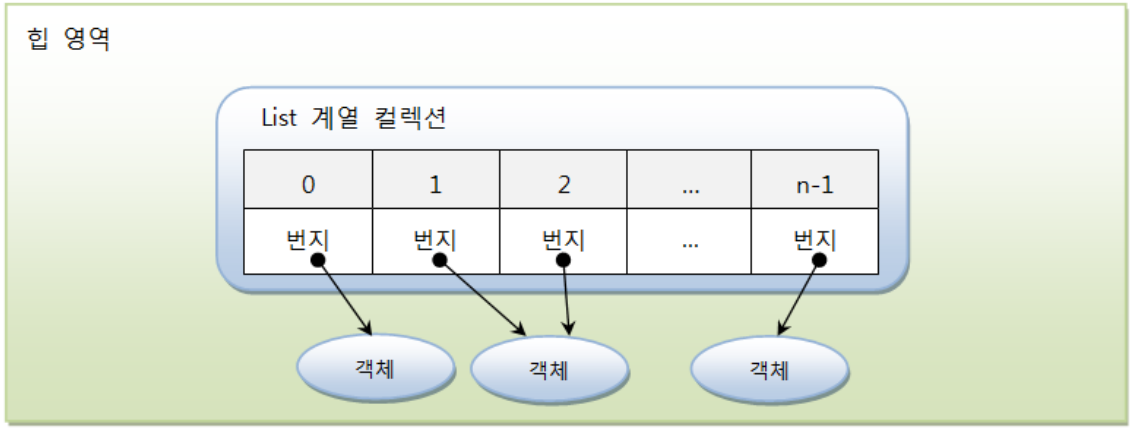
컬렉션을 위한 자바 인터페이스와 클래스



List 컬렉션

List 컬렉션

- List 컬렉션의 특징 및 주요 메소드
  - ✓특징
    - 인덱스로 관리
    - 중복해서 객체 저장 가능
  - ✓구현 클래스
    - ArrayList
    - Vector
    - LinkedList



List 컬렉션

- List 컬렉션의 특징 및 주요 메소드
  - ✓주요 메소드

기능	메소드	설명
객체 추가	boolean add(E e)	주어진 객체를 맨끝에 추가
	void add(int index, E element)	주어진 인덱스에 객체를 추가
	set(int index, E element)	주어진 인덱스에 저장된 객체를 주어진 객체로 바꿈
객체 검색	boolean contains(Object o)	주어진 객체가 저장되어 있는지 여부
	E get(int index)	주어진 인덱스에 저장된 객체를 리턴
	isEmpty()	컬렉션이 비어 있는지 조사
	int size()	저장되어있는 전체 객체수를 리턴
객체 삭제	void clear()	저장된 모든 객체를 삭제
	E remove(int index)	주어진 인덱스에 저장된 객체를 삭제
	boolean remove(Object o)	주어진 객체를 삭제

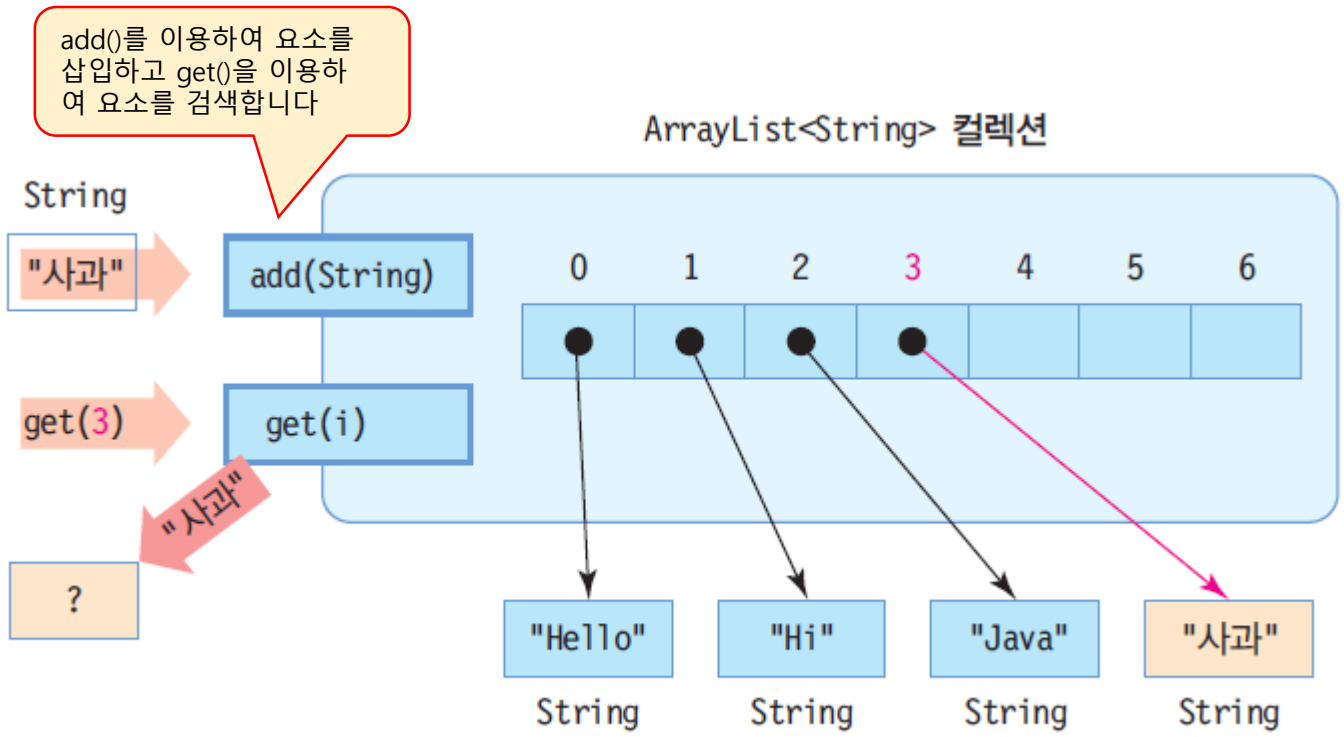
ArrayList

ArrayList<E>

- ArrayList<E>의 특성
  - ✓java.util.ArrayList, 가변 크기 배열을 구현한 클래스
    - <E>에서 E 대신 요소로 사용할 특정 타입으로 구체화
  - ✓ArrayList에 삽입 가능한 것
    - 객체, null
    - 기본 타입은 박싱/언박싱으로 Wrapper 객체로 만들어 저장
  - ✓ArrayList에 객체 삽입/삭제
    - 리스트의 맨 뒤에 객체 추가
    - 리스트의 중간에 객체 삽입
    - 임의의 위치에 있는 객체 삭제 가능
  - ✓벡터와 달리 스레드 동기화 기능 없음
    - 다수 스레드가 동시에 ArrayList에 접근할 때 동기화되지 않음
    - 개발자가 스레드 동기화 코드 작성

ArrayList<String> 컬렉션의 내부 구성

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

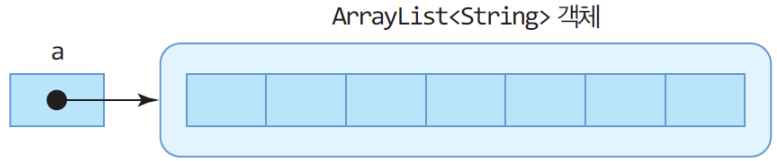


ArrayList<E> 클래스의 주요 메소드

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

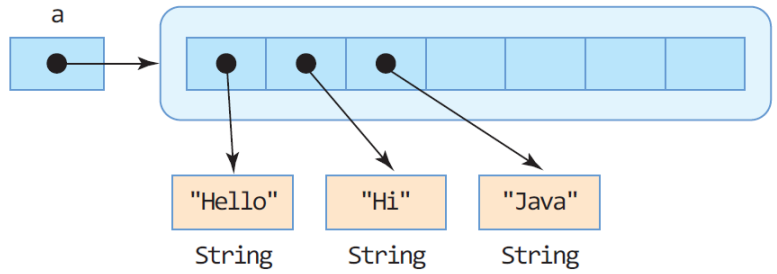
ArrayList 생성

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



요소 삽입

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



요소 개수 n

용량

```
int n = a.size(); // n은 3
int c = a.capacity(); // capacity() 메소드 없음
```

오류

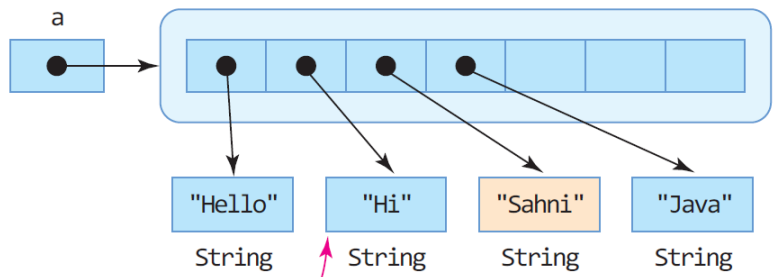
n = 3

요소 중간 삽입

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

오류

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



요소 알아내기

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

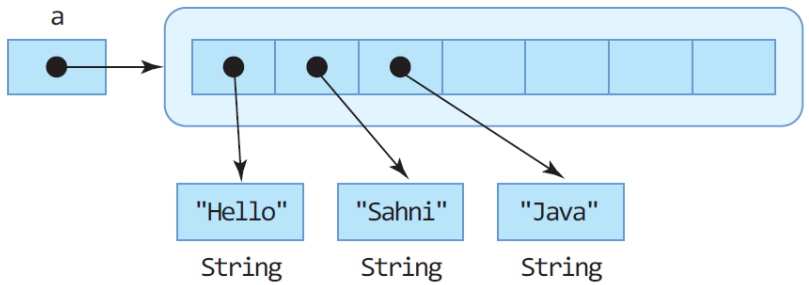
"Hi"

요소 삭제

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

오류

```
a.remove(4); // 오류
```



모든 요소 삭제

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

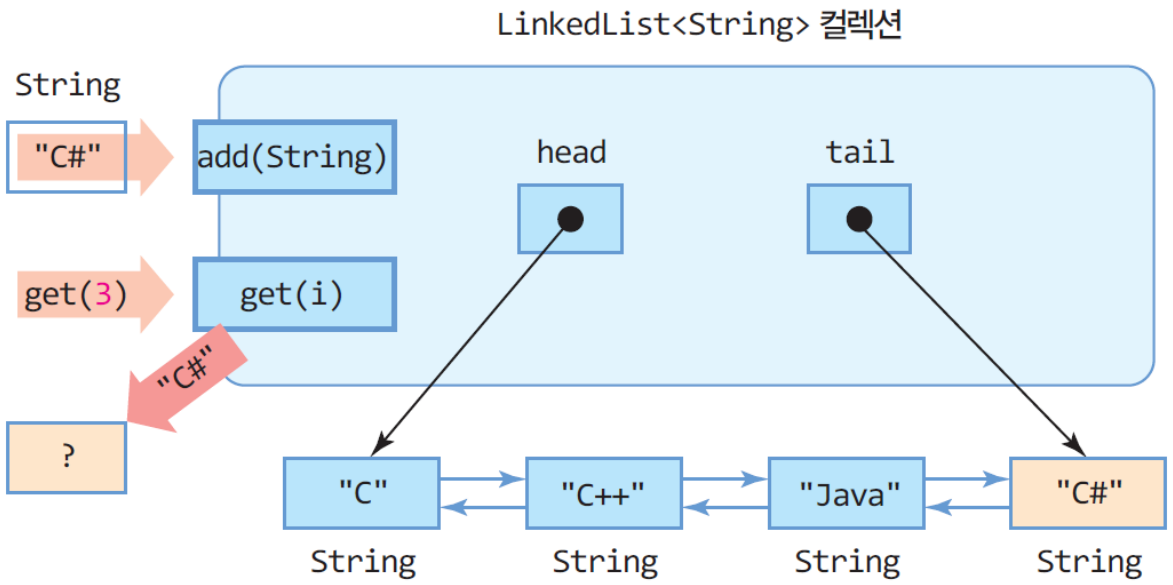
LinkedList

LinkedList<E>

- LinkedList<E>의 특성
  - ✓java.util.LinkedList
    - E에 요소로 사용할 타입 지정하여 구체와
  - ✓List 인터페이스를 구현한 컬렉션 클래스
  - ✓Vector, ArrayList 클래스와 매우 유사하게 작동
  - ✓요소 객체들은 양방향으로 연결되어 관리됨
  - ✓요소 객체는 맨 앞, 맨 뒤에 추가 가능
  - ✓요소 객체는 인덱스를 이용하여 중간에 삽입 가능
  - ✓맨 앞이나 맨 뒤에 요소를 추가하거나 삭제할 수 있어 스택이나 큐로 사용 가능

LinkedList<String>의 내부 구성과 put(), get() 메소드

```
LinkedList<String> l = new LinkedList<String>();
```



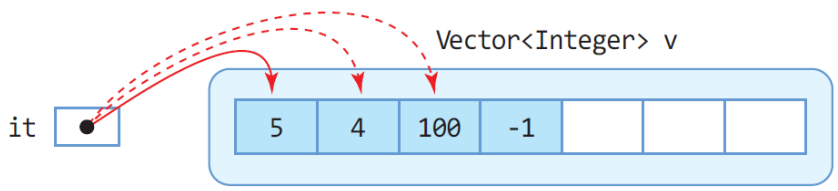
컬렉션의 순차 검색을 위한 Iterator

- Iterator<E> 인터페이스
  - ✓ Vector<E>, ArrayList<E>, LinkedList<E>가 상속받는 인터페이스
    - 리스트 구조의 컬렉션에서 요소의 순차 검색을 위한 메소드 포함
  - ✓ Iterator<E> 인터페이스 메소드

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

  - ✓ iterator() 메소드 : Iterator 객체 반환
    - Iterator 객체를 이용하여 인덱스 없이 순차적 검색 가능

```
Vector<Integer> v = new Vector<Integer>();
Iterator<Integer> it = v.iterator();
while(it.hasNext()) { // 모든 요소 방문
   int n = it.next(); // 다음 요소 리턴
   …
}
```



예제 7-4 : Iterator를 이용하여 Vector의 모든 요소를 출력하고 합 구하기

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

```
import java.util.*;

public class IteratorEx {
    public static void main(String[] args) {
        // 정수 값만 다루는 제네릭 벡터 생성
        Vector<Integer> v = new Vector<Integer>();
        v.add(5); // 5 삽입
        v.add(4); // 4 삽입
        v.add(-1); // -1 삽입
        v.add(2, 100); // 4와 -1 사이에 정수 100 삽입

        // Iterator를 이용한 모든 정수 출력하기
        Iterator<Integer> it = v.iterator();
        while(it.hasNext()) {
            int n = it.next();
            System.out.println(n);
        }
        // Iterator를 이용하여 모든 정수 더하기
        int sum = 0;
        it = v.iterator(); // Iterator 객체 얻기
        while(it.hasNext()) {
            int n = it.next();
            sum += n;
        }
        System.out.println("벡터에 있는 정수 합 : " + sum);
    }
}
}
```

```
5
4
100
-1
벡터에 있는 정수 합 : 108
```