

# chapter04

## 컬렉션프레임워크

1. 제네릭(**Generic**)
2. 컬렉션프레임워크 소개
3. List
4. Set
5. Map

# 01 제네릭(Generic)

## ■ 제네릭(Generic)

- 클래스 내부에서 사용할 데이터 타입을 외부에서 지정하는 기법  
(클래스 내부에서 사용할 데이터 타입을 나중에 인스턴스를 생성할 때 확정 하여 사용)
- 객체의 타입을 컴파일 시에 체크하기 때문에  
객체의 타입 안정성을 높이고 형변환의 번거로움이 줄어듦

사용법: `MyList<Point> myList = new MyList<Point>();`

### PointList.java, CircleList.java


```
public class PointList {  
    private Point pArray;  
    private int crtPos;  
  
    public PointList() {  
        this.pArray = new Point[3];  
        this.crtPos = 0;  
    }  
  
    public void add(Point o) {  
        pArray[crtPos] = o;  
        crtPos++;  
    }  
}
```

### MyList.java

```
public class MyList {  
    private Object[] pArray;  
    private int crtPos;  
  
    public MyList() {  
        this.pArray = new Object[3];  
        this.crtPos = 0;  
    }  
  
    public void add(Object o) {  
        pArray[crtPos] = o;  
        crtPos++;  
    }  
}
```

### MyList.java

```
public class MyList<T> {  
    private T[] pArray;  
    private int crtPos;  
  
    public PointList() {  
        this.pArray = (T[])new Object[3];  
        this.crtPos = 0;  
    }  
  
    public void add(T o) {  
        pArray[crtPos] = o;  
        crtPos++;  
    }  
}
```



# chapter04

## 컬렉션프레임워크

1. 제네릭(Generic)
2. **컬렉션프레임워크 소개**
3. List
4. Set
5. Map

## 02 컬렉션프레임워크 소개

### ■ 프레임워크(framework)

- 표준화, 정형화된 체계적인 프로그래밍 방식

### ■ 컬렉션(collection)

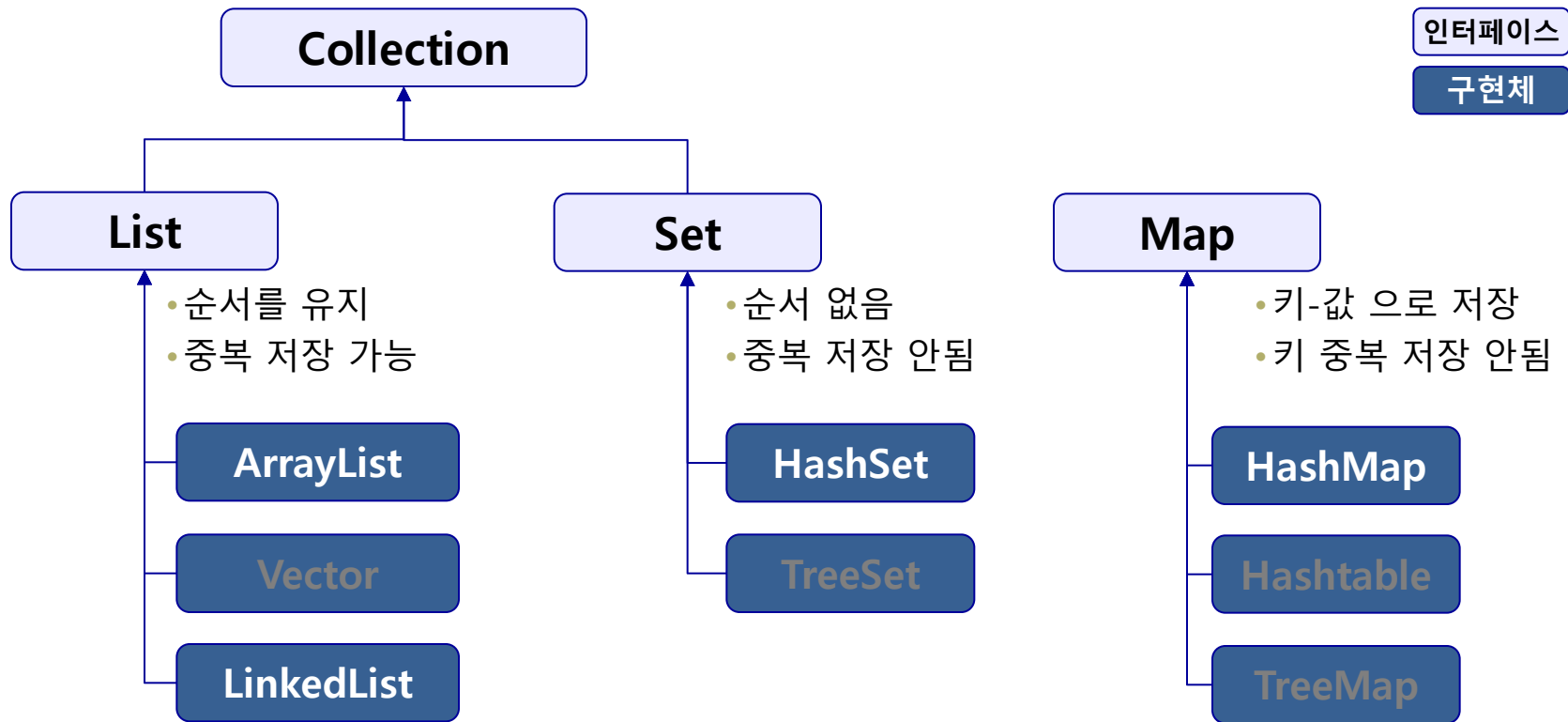
- 다수의 데이터, 즉 데이터 그룹을 말한다.

### ■ 컬렉션 프레임워크(collection framework)

- 다수의 데이터를 저장하는 클래스들을 표준화한 설계
- 다수의 데이터를 쉽게 처리할 수 있는 방법을 제공하는 클래스들로 구성
- JDK 1.2부터 제공

## 02 컬렉션프레임워크 소개

### ■ 구성



# chapter04

## 컬렉션프레임워크

1. 제네릭(Generic)
2. 컬렉션프레임워크 소개
3. **List**
4. Set
5. Map

## ■ 리스트(List)

- 순서를 유지, 중복 저장 가능

## ■ 배열(Array) vs 리스트(List)

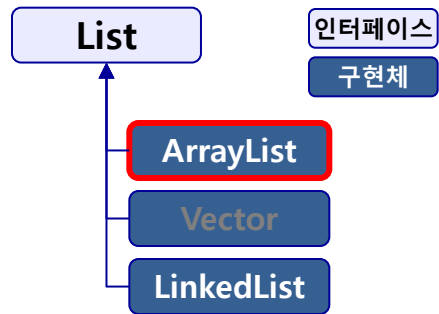
	배열(Array)	리스트(List)
장점	<ul style="list-style-type: none"><li>▪ 빠른 접근 (faster access)</li><li>▪ 기본 자료유형 사용 가능</li></ul>	<ul style="list-style-type: none"><li>▪ 가변적인 크기</li><li>▪ 필요시 메모리를 할당하므로 효율적</li></ul>
단점	<ul style="list-style-type: none"><li>▪ 고정된 크기</li><li>▪ 비효율적 메모리 점유</li><li>▪ 최대 크기를 넘어서는 사용을 위해서는 배열을 새로 정의해야 함</li></ul>	<ul style="list-style-type: none"><li>▪ 느린 접근 (slower access)</li><li>▪ 참조자료 유형만 사용 가능</li></ul>
사용	▪ Java에서 자체적으로 지원	▪ java.util.List 인터페이스 정의

## 02 List

### ■ 리스트(List) > ArrayList

- 인덱스로 관리된다.
- 배열과 유사한 방법으로 관리할 수 있다.

0	1	2	3	4	5	6	7	8	9

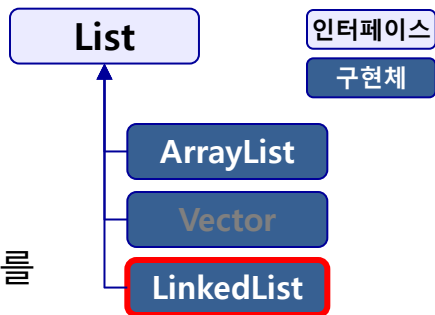




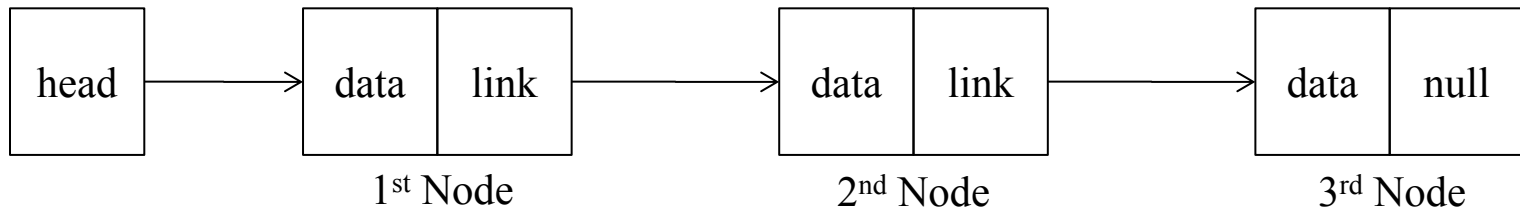
## 02 List

### ■ 리스트(List) > LinkedList

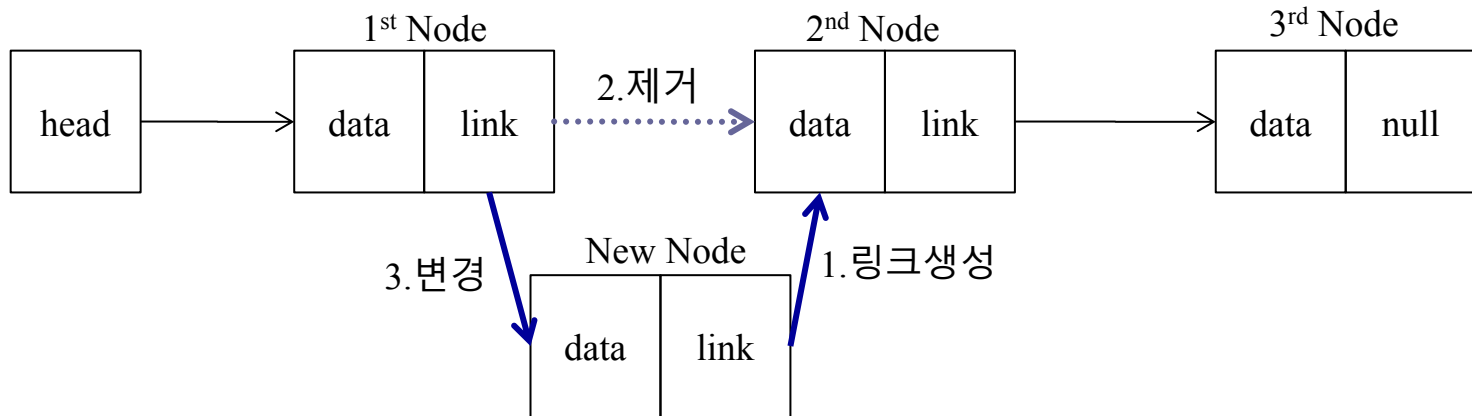
- 링크(Link)로 연결된 노드(Node)의 집합
- java.util.LinkedList 클래스에 정의됨
- Index를 통한 참조 접근은 불가능 head로부터 링크를 따라 가면서 접근
- 각 노드(Node)는 자신이 나타내는 데이터와 다음 노드(Node)로의 링크(Link)를 가지고 있음



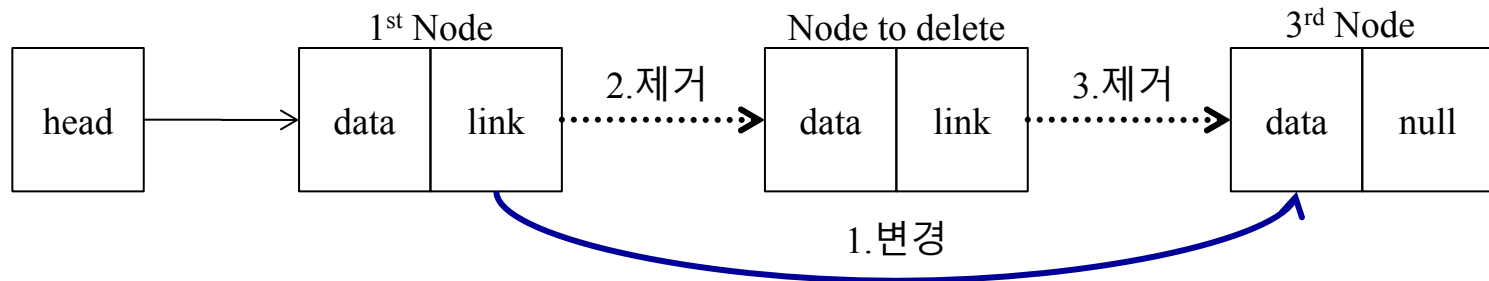
- 노드의 기본 구성



- 새로운 Node추가



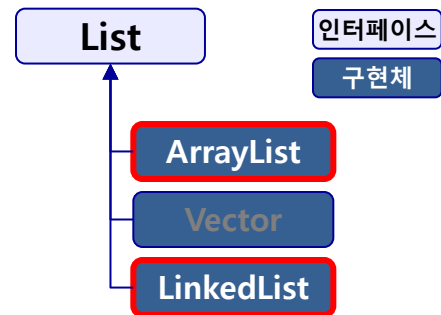
- 기존 Node삭제



## 02 List

### ■ ArrayList vs LinkedList

	순차적 추가/삭제	중간 추가/삭제	검색
ArrayList	빠르다	느리다	빠르다
LinkedList	느리다	빠르다	느리다



# chapter04

## 컬렉션프레임워크

1. 제네릭(Generic)
2. 컬렉션프레임워크 소개
3. List
- 4. Set**
5. Map

## ■ 셋(set)&gt;HashSet

- 순서없이 저장된다.
- 중복값은 저장되지 않는다. 따라서 **중복의 재정의**가 필요하다.
- 수학의 집합에 비유

Set	List
순서없음	순서유지
중복 저장 안됨	중복 저장 가능

중복정의를 중요

→클래스의 hashCode() 와 equals() 메소드 재정의 필요

## ■ HashSet 을 이용하여 미니로또 만들기

## [문제]

1~45 까지의 숫자중 임의의 6개의 숫자를 출력하세요  
(HashSet을 사용하여 중복제거)



```
Problems @ Javadoc Declaration  
<terminated> MiniLotto [Java Application] C  
6 40 20 39 27 25
```



```
Problems @ Javadoc Declaration  
<terminated> MiniLotto [Java Application]  
15 15 23 29 43 26
```

# chapter04

## 컬렉션프레임워크

1. 제네릭(Generic)
2. 컬렉션프레임워크 소개
3. List
4. Set
5. **Map**

## ■ 맵(Map)>HashMap

- 키(key)-값(value) 의 쌍으로 저장
- 키는 중복될 수 없으나 값은 중복될 수 있다.



키값들은 `keySet()`으로 관리된다.  
(중복허용안됨)