리스트 는 데이터를 순서대로 저장하는 선형 자료구조이다. 크기가 동적으로 변할 수 있고 데이터를 추가하거나 삭제할 때 마다 크기를 조절할 수 있어 유연하게 사용할 수 있다.

리스트의 특징

1. 순차적 저장 : 데이터가 순서대로 저장된다.
2. 인덱스 기반 : 요소는 인덱스를 가지고 인덱스를 통해 요소에 접근할 수 있다.
3. 동일 자료형 : 저장되는 모든 요소는 동일한 자료형이어야 한다.
4. 가변적인 크기 : 데이터를 추가하거나 삭제해도 크기가 자동으로 조절된다.
5. 중복 허용 : 동일한 값을 여러 개 저장할 수 있다.

자바에서 리스트 선언시

자바에서는 List 인터페이스를 통해 리스트를 사용하며 ArrayList와 LinkedList가 대표적인 구현체이다.

```java
import java.util.ArrayList;
import java.util.List;

public class ListExample {
    public static void main(String[] args) {
        List<String>  listData = new ArrayList<>();
        //데이터 추가
        listData.add(1);
        listData.add(2);
        listData.add(3);
        //데이터 접근
        System.out.println(listData.get(1)); //banana 출력
        //데이터 삭제
        listData.remove(0); //인덱스 0의 데이터 삭제
        //리스트 크기 확인
        System.out.println(listData.size()); // 2출력
    }
}
```

배열 리스트 (ArrayList)

배열을 기반으로 연속된 메모리 공간에 데이터를 저장한다.

배열 리스트 특징

1. 랜덤 접근 빠름 : 연속된 메모리 공간에 데이터를 저장하므로 인덱스를 이용하여 메모리 주소를 직접 계산할 수 있다. 데이터의 크기에 상관없이 일정한 시간에 특정 요소에 접근할 수 있어랜덤 접근시 연결 리스트 보다 빠르다. (O(1) 시간 복잡도)
2. 데이터 삽입,삭제 느림 : 연속된 메모리 공간을 사용하기 때문에 중간에 데이터를 삽입하거나

삭제하려면 메모리 블록 전체를 이동해야하고 데이터를 이동할 때 기존 데이터를 새로운 위치로 복사해야 하므로 추가적인 메모리 연산이 필요해 시간이 오래 걸린다. (O(n) 시간 복잡도)
5. 메모리 효율성 : 배열을 기반으로 연속된 메모리 공간에 데이터를 저장하기 때문에 메모리 효율성이 좋다.

데이터를 순서대로 저장하고 자주 랜덤 접근을 해야하는 경우 많이 사용한다.

연결 리스트 (LinkedList)

노드를 기반으로 객체를 연결하여 데이터를 저장한다. 각 노드는 데이터 와 다음 노드를 가리키는 포인터를 가지고 있다.

연결 리스트 특징
1. 랜덤 접근이 느림 : 특정 인덱스의 데이터를 찾으려면 처음부터 노드를 따라가야 하므로 시간이 오래 걸린다. (O(n) 시간 복잡도)
2. 데이터 삽입,삭제가 빠름 : 특정 위치에 데이터를 삽입하거나 삭제할 때 앞뒤 노드의 포인터만 변경하면 되므로 빠르다.
3. 메모리 비효율적 : 각 노드에 포인터를 저장해야 하므로 배열 리스트에 비해 메모리를 더 많이 사용한다.

데이터를 자주 삽입하거나 삭제해야 하는 경우 스택이나 큐를 구현할 때 많이 사용한다.

메소드설명

메소드	ArrayList 설명	LinkedList 설명
add(e)	리스트의 끝에 요소 추가	리스트의 끝에 요소 추가
add(i, e)	특정 인덱스에 요소 삽입	특정 인덱스에 요소 삽입
get(int i)	특정 인덱스의 요소 반환	특정 인덱스의 요소 반환
remove(int i)	지정된 인덱스의 요소를 삭제하고, 삭제된 요소를 반환	지정된 인덱스의 요소를 삭제하고, 삭제된 요소를 반환
remove(Object o)	리스트에서 처음으로 발견되는 특정 객체를 삭제하고 삭제 여부를 boolean 값으로 반환	리스트에서 처음으로 발견되는 특정 객체를 삭제하고 삭제 여부를 boolean 값으로 반환
set(int i, e)	지정된 인덱스의 요소를 새로운 요소로 변경하고, 이전 요소를 반환	지정된 인덱스의 요소를 새로운 요소로 변경하고, 이전 요소를 반환
size()	리스트의 크기를 반환	리스트의 크기를 반환
isEmpty()	리스트가 비어 있는지 확인 (true: 비어 있음, false: 비어 있지 않음)	리스트가 비어 있는지 확인 (true: 비어 있음, false: 비어 있지 않음)
contains(Object o)	리스트에 특정 객체가 포함되어 있는지 확인	리스트에 특정 객체가 포함되어 있는지 확인
clear()	리스트의 모든 요소를 삭제	리스트의 모든 요소를 삭제

메소드	ArrayList 설명	LinkedList 설명
addFirst(e)	없음	리스트의 처음에 요소 추가
addLast(e)	add(e)와 동일	리스트의 끝에 요소 추가
removeFirst()	없음	리스트의 첫 번째 요소 삭제
removeLast()	없음	리스트의 마지막 요소 삭제