자바 List, Generic

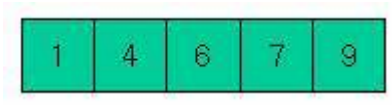
List

java.util.List Interface

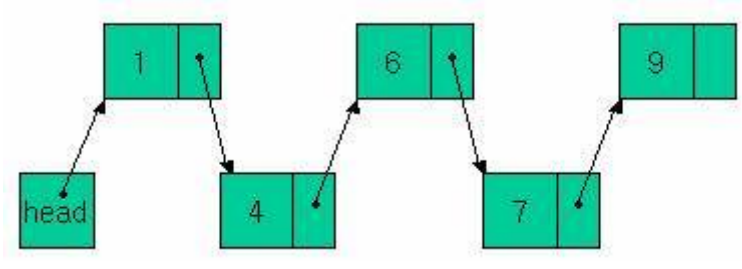
- 자바는 배열과 같은 데이터를 효과적으로 관리하기 위해 배열 대신 List를 제공한다.
- List는 ArrayList, LinkedList 두 가지 종류가 있다. 일반적으로 ArrayList를 가장 많이 사용한다.
- List는 배열에서 지원하지 않는 많은 기능을 제공한다.
- 지금까지 배열을 사용했다면 배열 대신 List를 사용한다.

ArrayList vs LinkedList

- ArrayList와 LinkedList 사용 방법은 같다.
- 다른 점은 데이터를 저장하는 방식이 다르다는 것 뿐이다.
- ArrayList 데이터 저장 방식



- LinkedList 데이터 저장 방식



- **ArrayList, LinkedList 중 어느 클래스를 사용할지 잘 모르겠다면 ArrayList를 사용**한다.

학습 테스트

> 학습 테스트란 자바에서 제공하는 API 사용 방법을 테스트 코드를 구현해 학습하는 것을 의미한다.

ArrayList 학습 테스트

- src/test/java에 study 패키지를 추가한다.
- study 패키지에 ArrayListTest 클래스를 추가한다.
- 다음 테스트 코드를 추가한 후 주석을 하나씩 제거하면서 문제를 해결한다.

```
import java.util.ArrayList;
import java.util.LinkedList;

public class ListTest {
    @Test
    public void arrayList() throws Exception {
        ArrayList<String> values = new ArrayList<>();
        values.add("first");
        values.add("second");

        assertThat([TODO]).isTrue(); // 세 번째 값을 추가하라.
        assertThat([TODO]).isEqualTo(3); // list의 크기를 구하라.
        assertThat([TODO]).isEqualTo("first"); // 첫 번째 값을 찾아라.
        assertThat([TODO]).isTrue(); // "first" 값이 포함되어 있는지를 확인해라.
        assertThat([TODO]).isEqualTo("first"); // 첫 번째 값을 삭제해라.
        assertThat([TODO]).isEqualTo(2); // 값이 삭제 됐는지 확인한다.

        // TODO values에 담긴 모든 값을 출력한다.
    }
}
```


힌트

- 값을 찾을 때는 get() 메소드
- 값을 가지고 있는지 찾으려면 contains() 메소드
- ArrayList가 제공하는 모든 기능은 구글에서 "arraylist javadocs"로 검색해 최신 버전은 자바 문서를 참고한다.

LinkedList 학습 테스트

- LinkedList는 ArrayList와 사용법이 같다.
- 다음 테스트 코드를 추가한 후 모든 assertEquals 메소드를 pass한다.

```
import static org.assertj.core.api.Assertions.assertThat;

import org.junit.Test;

import java.util.ArrayList;
import java.util.LinkedList;

public class ListTest {
    @Test
    public void arrayList() throws Exception {
        LinkedList<String> values = new LinkedList<>();
        values.add("first");
        values.add("second");

        assertThat([TODO]).isTrue(); // 세 번째 값을 추가하라.
        assertThat([TODO]).isEqualTo(3); // list의 크기를 구하라.
        assertThat([TODO]).isEqualTo("first"); // 첫 번째 값을 찾아라.
        assertThat([TODO]).isTrue(); // "first" 값이 포함되어 있는지를 확인해라.
        assertThat([TODO]).isEqualTo("first"); // 첫 번째 값을 삭제해라.
        assertThat([TODO]).isEqualTo(2); // 값이 삭제 됐는지 확인한다.

        // TODO values에 담긴 모든 값을 출력한다.
    }
}
```

Generic

Generic

- List와 같이 다양한 종류의 데이터를 관리하는 경우 데이터의 타입을 특정 타입으로 고정할 수 있다.

Generic 장점

- 특정 타입으로 제한함으로써 타입 안정성을 제공한다.
- 타입 체크와 형변환을 생략할 수 있으므로 코드가 간결해 진다.

- 추가한 객체 타입을 사용하려면 다시 형변환을 해야 한다.(Object => 특정 Type)


```
public class Position {
    private int x;
    private int y;

    public Position(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

```
public class ListTest {
    @Test
    public void noGenerics() throws Exception {
        ArrayList list = new ArrayList();
        list.add("this is string");
        list.add(1);
        list.add(new Position(1,2));

        String first = (String)list.get(0);
        int second = (int)list.get(1);
        Position third = (Position)list.get(2);
    }
}
```

Generic을 사용하는 경우

- `ArrayList<String>` 와 같이 <>를 사용해 ArrayList가 관리하는 타입을 지정한다.
- 이와 같이 타입을 지정하면 해당 타입만 추가할 수 있다. 다른 타입을 저장하면 컴파일 에러가 발생한다.

```
ArrayList<String> values = new ArrayList<String>();
values.add("first");
values.add("second");

String first = values.get(0);
String second = values.get(1);
```

특별히 예외적인 상황이 아니라면 Generic을 사용한다.

JavaDoc 읽기

- 자바에서 제공하는 API를 사용할 때 JavaDoc 문서를 참고해 사용법을 익힌다.

JavaDoc 접근하기

- 구글에서 "java8 javadoc", "java9 javadoc"으로 검색하면 접근할 수 있다.

미션

- 앞에서 학습한 ArrayList, LinkedList의 javadoc 문서를 읽고 ArrayList, LinkedList가 제공하는 기능을 확인해 본다.