COMP2500 실습 3

목차

블로깃(Blogit) 회사에서 여러분이 속해 있는 팀은 블로그 글 안에 목록을 추가할 수 있는 기능을 개발 중입니다. 목록을 만들 때 사용할 수 있는 클래스를 설계 해오라는 팀장님의 지시를 받은 여러분은 다음과 같은 UML 다이어그램을 그렸습니다. 이 다이어그램을 그릴 때 여러분이 가정한 것은 블로그 글 작성자가 사용하는 목록이 3단계를 넘지 않을 것이라는 것과, 각 단계마다 글머리 기호(bullet) 모양이 다를 것이라는 것이었습니다.

그리고 여러분은 위 클래스를 다음과 같이 사용할 것이라고 생각했습니다.

```
package com.blogit.list;

import java.util.ArrayList;

public class Program {
    public static void main(String[] args) {
        ArrayList<ListItem> list = new ArrayList<>();

        ListItem listItem1 = new ListItem("My first item");

        SublistItem sublistItem1 = new SublistItem("This is sublist item1");
        SublistItem sublistItem2 = new SublistItem("This is sublist item2");

        listItem1.addSublistItem(sublistItem1);
        listItem1.addSublistItem(sublistItem2);

        ListItem listItem2 = new ListItem("My second item");

        ListItem listItem3 = new ListItem("My third item");

        SublistItem sublistItem3 = new SublistItem("This is sublist item3");
        SubSublistItem subSublistItem1 = new SubSublistItem("This is sub-sublist item1");

        sublistItem3.addSubSublistItem(subSublistItem1);
        listItem3.addSublistItem(sublistItem3);

        list.add(listItem1);
        list.add(listItem2);
        list.add(listItem3);

        String listString = toString(list);

        System.out.print(listString);
    }

    private static String toString(ArrayList<ListItem> list) {
        StringBuilder sb = new StringBuilder();
        for (ListItem item : list) {
            sb.append(item);
        }

        return sb.toString();
    }
}
```

그리고 `main()` 메서드 안의 마지막 줄에서 `listString`을 출력하면 다음과 같은 결과가 나오면 좋겠다 생각했고요.

```
*.My.first.item↵
....>.This.is.sublist.item1↵
....>.This.is.sublist.item2↵
*.My.second.item↵
*.My.third.item↵
....>.This.is.sublist.item3↵
........-.This.is.sub-sublist.item1↵
```

기호 설명

- `.` : 스페이스(빈칸 문자)
- ↵ : 줄 바꿈 문자

이제 설레는 마음으로 팀장님이 설계를 보여드렸는데 다음과 같이 말씀하시네요.

"불필요한 클래스가 너무 많네요. 이거 클래스 하나 가지고 구현할 수 있는데 말이죠. 그리고 블로그 글 작성자가 추가할 수 있는 목록 단계에 제한이 없어야 합니다. 아셨죠? ListItem 클래스만 남기고 나머지 클래스는 지우는 방향으로 다시 설계 및 구현하세요."

그리고 팀장님이 다음 사항들을 설계에 추가하라고 지시하십니다.

1. 기본 글머리 기호는 언제나 `*`으로 둘 것
2. 글머리 기호를 매개변수로 받는 생성자도 만들어 `ListItem`에 원하는 글머리 기호를 사용할 수 있게 허용할 것
3. 글머리 기호의 setter를 만들어 언제라도 글머리 기호를 변경할 수 있게 허용할 것
4. 어떤 `ListItem` 개체에 있는 하위 목록을 제거할 수 있는 방법을 추가할 것

이제 팀장님의 피드백을 잘 반영해서 멋지게 일을 끝내야겠죠? 파이팅! :D

1. 프로젝트를 준비한다

1. `Lab3` 폴더로 이동합니다.
2. `src/academy/pocu/comp2500/lab3` 폴더에 `ListItem.java` 파일을 추가합니다.
3. 다음 내용을 위 파일에 추가합니다.

```
package academy.pocu.comp2500.lab3;

public class ListItem {

}
```

2. `ListItem` 클래스 구현하기

전반적인 규칙

- 이 실습에서 제공하는 테스트 코드를 사용하려면 `-ea` VM 옵션으로 `assert` 키워드를 활성화시켜야 합니다. 본인 코드에서 `assert`를 사용한다면 이 옵션을 켜고 프로그램을 실행하세요.
- 이 실습을 완료하려면 오직 `ListItem` 클래스만 필요합니다. 프로젝트에 선언된 다른 클래스가 있다면 불필요한 클래스를 추가한 것으로 간주하여 빌드봇이 점수를 깎습니다. (`Program.java`는 예외)
- 클래스 캡슐화를 잘하세요. 즉, 불필요하게 메서드나 멤버 변수에 `public` 또는 `package`에 접근 제어자를 붙이면 안 됩니다.
- 어떤 클래스 대신 사용할 수 있는 기본 자료형이 존재한다면(예: Boolean/boolean, Integer/int) 그걸 대신 사용하세요. 기본 자료형 대신 클래스 버전이 허용되는 경우는 제네릭(generic) 클래스의 타입(type) 매개변수로 사용할 때 뿐입니다.

2.1 `ListItem` UML 다이어그램에 있는 모든 메서드를 구현한다

우선 `ListItem` 클래스에 연결된 `SublistItem`과 `SubSublistItem` 클래스를 어떻게 제거할지 생각해보세요. 그 생각에 따라 UML 다이어그램을 다시 그린 후, 다음의 메서드들을 구현하세요.

각 메서드의 매개변수 형과 반환형은 UML 다이어그램에 잘 나와있으니 여기서는 각 메서드가 해야 하는 일만 설명합니다.

- `ListItem()` : 유일한 인자로 `text`를 받는 생성자입니다. 기본 글머리 기호는 `*`이 되어야 합니다.

```
ListItem item1 = new ListItem("This is item1");
```

- `getText()` : 목록 아이템의 텍스트를 반환하는 getter

```
ListItem item1 = new ListItem("This is item1");
String s = item1.getText(); // s: This is item1
```

- `setText()` : 목록 아이템의 텍스트를 설정하는 setter

```
ListItem item1 = new ListItem("This is item1");
String s = item1.getText(); // s: This is item1
item1.setText("This is updated item1");
s = item1.getText(); // s: This is updated item1
```

- `getBulletStyle()` : 글머리 기호의 getter

```
ListItem item1 = new ListItem("This is item1");
char c = item.getBulletStyle(); // c: *
```

- `addSublistItem()` : 하위 목록 아이템을 추가합니다.

```
ListItem item1 = new ListItem("This is item1");
item1.addSublistItem(new ListItem("This is sublist item1"));
```

- `getSublistItem()` : 지정된 색인위치에 있는 하위 목록 아이템을 반환합니다. 색인 값은 언제나 유효하다고 가정하세요.

```
ListItem item1 = new ListItem("This is item1");

item1.addSublistItem(new ListItem("This is sublist item1"));

ListItem sublistItem1 = item1.getSublistItem(0);
String s = sublistItem1.getText(); // s: This is sublist item1
```

- `toString()`: `ListItem` 개체를 `String`으로 표현합니다. 저 위에 보여준 `listString`의 포맷과 일치하게 문자열을 만드세요.

```
ListItem item1 = new ListItem("This is item1");
ListItem sublistItem1 = new ListItem("This is sublist item1");
ListItem subSublistItem1 = new ListItem("This is sub-sublist item1");
ListItem subSublistItem2 = new ListItem("This is sub-sublist item2");

sublistItem1.addSublistItem(subSublistItem1);
sublistItem1.addSublistItem(subSublistItem2);
item1.addSublistItem(sublistItem1);

String s = item1.toString();
// s:
/*
* This is item1
    * This is sublist item1
        * This is sub-sublist item1
        * This is sub-sublist item2
*/
```

2.2 새로운 생성자를 추가한다

- 새로운 생성자는 다음의 인자들을 받아야 합니다.
    - 목록 아이템의 텍스트: `String text`
    - 글머리로 사용할 기호: `char bulletStyle`

```
ListItem item1 = new ListItem("This is item1", '1');
ListItem item2 = new ListItem("This is item2", '2');

item1.toString();
/*
1 This is item1

*/

item2.toString();
/*
2 This is item2

*/
```

2.3 글머리 기호의 setter를 추가한다

- 이 setter는 다음의 인자를 받아야 합니다.
    - 목록 아이템에 사용할 글머리 기호: `char bulletStyle`

2.4 `removeSublistItem()` 메서드를 추가한다

- `removeSublistItem()`는 다음의 인자를 받아야 합니다.
    - 제거할 하위 목록 아이템의 색인 위치: `int index`
- 이 메서드는 `index` 위치에 있는 하위 목록 아이템을 제거합니다.
- 이 메서드는 아무것도 반환하지 않습니다.
- 색인 값은 언제나 유효하다고 가정하세요.

```
ListItem item = new ListItem("This is item");

item.addSublistItem(new ListItem("This is sublist item1"));
item.addSublistItem(new ListItem("This is sublist item2"));

item.toString();
/*
* This is item
    * This is sublist item1
    * This is sublist item2

*/

item.removeSublistItem(0);

item.toString();
/*
* This is item
    * This is sublist item2

*/
```

3. 본인 컴퓨터에서 테스트하는 법

- `Program.java`를 아래처럼 바꾼 뒤 실행하세요.

```
package academy.pocu.comp2500.lab3.app;

import academy.pocu.comp2500.lab3.ListItem;

import java.util.ArrayList;

public class Program {

    public static void main(String[] args) {
        ArrayList<ListItem> list = new ArrayList<>();

        ListItem listItem1 = new ListItem("My first item");

        ListItem sublistItem1 = new ListItem("This is sublist item1", '>');
        ListItem sublistItem2 = new ListItem("This is sublist item2", '>');

        listItem1.addSublistItem(sublistItem1);
        listItem1.addSublistItem(sublistItem2);

        ListItem listItem2 = new ListItem("My second item");

        ListItem listItem3 = new ListItem("My third item");

        ListItem sublistItem3 = new ListItem("This is sublist item3", '>');
        ListItem subSublistItem1 = new ListItem("This is sub-sublist item1", '-');
        sublistItem3.addSublistItem(subSublistItem1);
        listItem3.addSublistItem(sublistItem3);

        list.add(listItem1);
        list.add(listItem2);
        list.add(listItem3);

        String actual = toString(list);

        StringBuilder sb = new StringBuilder();
        sb.append(String.format("* My first item%s", System.lineSeparator()));
        sb.append(String.format("    > This is sublist item1%s", System.lineSeparator()));
        sb.append(String.format("    > This is sublist item2%s", System.lineSeparator()));
        sb.append(String.format("* My second item%s", System.lineSeparator()));
        sb.append(String.format("* My third item%s", System.lineSeparator()));
        sb.append(String.format("    > This is sublist item3%s", System.lineSeparator()));
        sb.append(String.format("        - This is sub-sublist item1%s",
                System.lineSeparator()));

        String expected = sb.toString();

        assert expected.equals(actual);

        System.out.print(actual);
    }

    private static String toString(ArrayList<ListItem> list) {
        StringBuilder sb = new StringBuilder();
        for (ListItem item : list) {
            sb.append(item);
        }

        return sb.toString();
    }
}
```

4. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)