COMP2500 과제 1

목차

본 과제는 컴퓨터에서 해야 하는 과제입니다. 코드 작성이 끝났다면 실습 1에서 만들었던 깃 저장소에 커밋 및 푸시를 하고 슬랙을 통해 자동으로 채점을 받으세요.

여러분은 블로깃(Blogit) 회사에 최근에 입사한 프로그래머입니다. 블로깃은 차세대 블로그 시스템을 개발하려는 중인데 여기에 올바른 OOP 방법론을 사용하는 것을 매우 중시하고 있습니다.

이 프로젝트의 제품 책임자(product owner)가 사용자 스토리(user story)의 형태로 명세서를 작성하여 개발 팀에 전달하였고, 개발 팀장님은 여러분에게 메서드 매개변수에 허용할 이름들을 뽑아오라고 지시했습니다. 그리 어렵지 않은 일 같아 곧바로 일에 착수해 블로그 시스템에 사용할 법한 매개변수 이름 목록(아래)을 개발 팀장님에게 전달했습니다.

```
article
articleId
at
author
authorId
authorName
blog
blogId
body
comment
commentId
content
count
created
createdAt
createdDateTime
displayName
email
emailAddress
emoji
emojiId
emojiType
emoticon
emoticonId
emoticonType
firstName
fullName
id
lastName
modified
modifiedAt
modifiedDateTime
name
nickname
post
postId
reaction
reactionId
reactionType
sorting
sortingType
subcomment
subcommentId
tag
text
title
type
user
userId
```

이 목록을 쓰윽 훑어보면서 사내 코딩 표준에 어긋나지 않음을 확인한 개발 팀장님은 public 메서드에 들어가는 매개변수에는 이 이름들만 사용하라고 모든 프로그래머에게 지시했습니다. 블로그 시스템의 기능들을 설명하는데 서로 다른 용어를 사용하지 말라는 취지였죠. 이제 준비작업은 끝. 여러분에게 진짜 일이 떨어졌습니다. 명세서를 읽고 블로그 시스템을 설계/구현하랍니다. 갑자기 뭔가 일이 커진 거 같은데... 그래도 할 수 있죠?

1. 프로젝트를 준비한다

1. 실습 1을 반드시 끝내도록 하세요. 그래야만 본인의 컴퓨터에 `Assignment1` 폴더가 올바르게 설정되어 있을 겁니다.
2. `Assignment1` 폴더로 이동합니다.
3. 자, 이제 자리에서 일어나서 국민체조 한 번 하시고... 프로그래밍 시작하겠습니다. :)

2. 블로그 시스템을 설계 및 구현한다

2.1 전반적인 규칙

- 코드 속에서 `assert` 키워드를 사용한다면 프로그램을 실행할 때 `-ea` VM 옵션을 사용해야 합니다. 그래야 `assert` 키워드가 작동하거든요. 아래 섹션 4에서 `Registry` 클래스에 제대로 메서드를 등록했는지를 확인할 때도 `assert`를 사용합니다. 따라서 프로그램 실행 시 이 옵션을 사용하는 걸 추천합니다.
- 아래에 나와있는 사용자 스토리에 따라 마음껏 클래스를 설계하세요. 그러나 빌드봇이 여러분의 설계가 적절하지 못하다 생각하면 점수를 깎을 것입니다. :P
- 여러분이 작성한 코드는 사용자 스토리를 모두 충족해야 합니다. 역시 빌드봇이 기능도 꼼꼼히 검사해줍니다.
- 인자를 받는 public 메서드가 있다면 위 목록에 있는 매개변수 이름만 사용해야 합니다. 이 매개변수에 적합한 자료형은 여러분이 알아서 잘 선택하세요. 빌드봇이 어떤 매개변수 형이 적절하지 못하다고 생각하면 역시 점수를 깎을 것입니다.
  - 예: `int emailAddress`. 이메일 주소를 정수로 표현할 수 없는 건 당연하죠?
- 위 목록에 있는 매개변수의 복수형도 허용됩니다. 하지만 올바른 영단어를 사용해 주세요.
  - 예: `user -> users`는 허용. 그러나 `modified -> modifieds`는 올바른 영단어가 아니므로 허용 안 됨.

- 매개변수가 `null` 값을 받을 수 있다면 `OrNull` 접미사를 붙이는 것도 허용됩니다.
  - 예: `user` -> `userOrNull`
- Assignment1 프로젝트 안에 `App.java`, `Interface.java`, `Registry.java` 등, 미리 정의된 클래스들이 있습니다. 일단은 신경 쓰지 마세요! 섹션 4에서 사용합니다.
- 여러분이 작성하는 클래스들은 반드시 `academy.pocu.comp2500.assignment1` 패키지 안에 속해야 합니다.
- 날짜와 시간을 표현하는 자료형으로는 `OffsetDateTime`을 사용해야 합니다.
- 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.
- `private static final` 멤버 변수 외에 어떤 정적 멤버 변수도 사용할 수 없습니다.
- `public static`와 패키지 `static` 메서드를 사용할 수 없습니다.

2.2 사용자 스토리(User Stories)

1. 블로그 주인은
   새 블로그를 만들어
   그 블로그에 새로운 글들을 작성한다.

2. 블로그 방문자는
   블로그에 있는 모든 글들을 가져와
   그 목록에서 읽고 싶은 글들을 선택한다.

3. 블로그 방문자는
   태그 필터를 사용하여
   지정된 태그들이 달린 블로그 글들만 가져온다.

4. 블로그 방문자는
   작성자 필터를 사용하여
   지정된 작성자가 쓴 블로그 글들만 가져온다.

5. 블로그 방문자는
   기본적으로 작성 일시 기준으로 내림차순 정렬된 글들을 받아와
   최신 글을 제일 먼저 보게 된다.

6. 블로그 방문자는
   블로그 글 정렬 방법을 선택하여
   그 방법에 따라 정렬된 블로그 글들을 받아온다.

   본 시스템에서 사용할 수 있는 정렬 방법은 다음과 같다.

   1. 작성 일시 내림차순
   2. 작성 일시 오름차순
   3. 수정 일시 내림차순
   4. 수정 일시 오름차순
   5. 제목(사전 순서) 오름차순

7. 블로그 작성자는
   블로그에 새 글을 추가하여
   자신의 생각과 경험을 방문자들과 공유한다.

8. 블로그 작성자는
   이전에 작성했던 블로그 글의 제목을 바꿔
   오타 등을 고친다.

9. 블로그 작성자는
   이전에 작성했던 블로그 글의 본문을 바꿔
   잘못된 정보나 오타 등을 고친다.

10. 블로그 작성자는
    어떤 블로그 글에 태그를 달아
    방문자들이 태그를 사용하여 글 목록을 필터링할 수 있게 해준다.

11. 블로그 방문자는
    어떤 블로그 글에 달린 모든 댓글을 가져와
    댓글들을 하나씩 읽는다.

12. 블로그 방문자는
'추천(upvote) 수 - 비추천(downvote) 수' 결과대로 내림차순 정렬된 댓글들을 가져와
가장 인기 있는 댓글을 먼저 읽는다.

13. 블로그 방문자는
발행된 블로그 글에 댓글을 달아
자신의 의견을 표출하거나 블로그 글 작성자에게 추가 질문을 한다.

14. 블로그 방문자는
발행된 댓글에 댓글을 달아(하위 댓글)
자신의 의견을 표출하거나 그 댓글을 단 다른 방문자에게 추가 질문을 한다.

15. 블로그 방문자는
자신이 달았던 댓글을 바꿔
잘못된 정보나 오타를 고친다.

16. 블로그 방문자는
어떤 블로그 글에 리액션을 추가하거나 그 글로부터 리액션을 제거하여
그 글에 대한 자신의 느낌을 표출한다.

    본 시스템에서 사용할 수 있는 리액션은 다음과 같다.

    1. 최고! (Great)
    2. 슬퍼요! (Sad)
    3. 화나요! (Angry)
    4. 잼나요! (Fun)
    5. 사랑해요! (Love)

17. 블로그 방문자는
댓글을 추천(upvote)하여
'추천 수 - 비추천 수' 결과가 최고인 댓글이 제일 먼저 보이게 한다.

18. 블로그 방문자는
댓글을 비추천(downvote)하여
'추천 수 - 비추천 수' 결과가 최저인 댓글이 제일 마지막에 보이게 한다.

3. 본인 컴퓨터에서 테스트하는 법

이 과제는 설계부터 구현까지 모두 여러분이 하기에 정형화된 테스트 코드를 제공해 드리는 것이 불가능합니다. 따라서 본 과제에서 작성한 코드를 테스트하는 것도 궁극적으로는 여러분의 몫입니다. 단, 몇 가지 테스트 케이스들을 아래에 글로 적어 둘 테니 테스트할 때 참고하세요. 이 외에도 더 많은 테스트를 해야 할 것입니다. 테스트를 많이 할수록 버그가 적어진다는 사실, 잘 아시죠? :)

3.1 블로그 생성하기

1. 생성자를 호출하여 새 블로그를 만든다.
2. 생성 시 초기화한 멤버 변수가 있다면 그 값이 올바른지 확인한다.

3.2 블로그 글 추가하기

1. 새 블로그를 만든다.
2. 블로그에 새 블로그 글을 하나 추가한다.
3. 2번에서 추가한 블로그 글이 블로그에 있는지 확인한다.

3.3 블로그 글 목록 가져오기

1. 새 블로그를 만든다.
2. 블로그에 새 글을 3개 추가한다.
3. 블로그 글들을 가져온다.
4. 2번에서 추가한 모든 블로그 글들이 생성 일시 순으로 정렬되어 있는지 확인한다.

3.4 블로그 글 목록 필터링하기(태그 기준)

1. 새 블로그를 만든다.
2. 블로그에 새 글을 2개 추가하되, 각 글에 다른 태그를 하나씩 붙인다.
3. 2번에서 사용한 태그 중 하나로 필터를 설정한다.
4. 블로그 글들을 가져온다.
5. 3번에서 지정한 태그가 달린 글만 목록에 있는지 확인한다.

3.5 블로그 글 목록 필터링하기(작성자 기준)

1. 새 블로그를 만든다.
2. 블로그에 새 글을 2개 추가하되, 두 글의 작성자를 다른 사람으로 한다.
3. 2번에서 사용한 작성자 중 하나로 필터를 설정한다.
4. 블로그 글들을 가져온다.
5. 3번에서 지정한 작성자가 작성한 글만 목록에 있는지 확인한다.

3.6 블로그 글 목록 정렬하기

1. 새 블로그를 만든다.
2. 새 글을 여럿 추가한다.
3. 정렬 방법 하나를 선택한다.
4. 블로그 글들을 가져온다.
5. 3번에서 지정한 방법에 따라 글들이 정렬되어있는지 확인한다.

3.7 블로그 글에 댓글 달기

1. 새 블로그를 만든다.
2. 새 글을 하나 추가한다.
3. 2번의 글에 새 댓글을 하나 추가한다.
4. 그 글에 달린 모든 댓글들을 가져온다.
5. 3번에서 추가한 댓글이 목록에 있는지 확인한다.

3.8 하위 댓글 달기

1. 새 블로그를 만든다.
2. 새 글을 하나 추가한다.
3. 2번의 글에 새 댓글을 하나 추가한다.
4. 3번의 댓글에 새 하위 댓글을 하나 추가한다.
5. 4번에서 추가한 하위 댓글이 3번에서 추가한 댓글에 달렸는지 확인한다.

3.9 댓글을 추천/비추천하기

1. 새 블로그를 만든다.
2. 새 글을 하나 추가한다.
3. 2번의 글에 댓글을 여럿 추가한다.
4. 몇몇 댓글을 추천 또는 비추천한다.
5. 그 글에 달린 모든 댓글들을 가져온다.
6. '추천 수 - 비추천 수'의 내림차순에 따라 댓글들이 정렬되어 있는지 확인한다.
7. 하위 댓글에 대해 3~6번을 반복한다.

3.10 블로그 글 업데이트하기

1. 새 블로그를 만든다.
2. 새 글을 하나 추가한다.
3. 그 글의 제목이나 본문을 바꾼다.
4. 그 글의 제목이나 본문이 바뀌었는지 확인한다.

3.11 블로그 글에 리액션 달기

1. 새 블로그를 만든다.
2. 새 글을 하나 추가한다.
3. 리액션을 하나 추가한다.
4. 3번의 리액션 숫자가 증가했는지 확인한다.
5. 다른 리액션을 추가한다.
6. 5번의 리액션 숫자가 증가했는지 확인한다.
7. 3번의 리액션 숫자가 줄지 않았는지 확인한다.

4. 클래스와 메서드 등록하기

본인 컴퓨터에서 프로그램 테스트를 마쳤다면 `academy.pocu.comp2500.assignment1.registry` 패키지 안에 있는 `Registry` 클래스에 여러분의 메서드들을 등록하세요. 그래야 빌드봇이 여러분의 코드를 테스트할 때 어떤 public 메서드들을 호출해야 되는지 알 수 있답니다.
`Registry` 클래스에는 총 21개의 `registerXXX()` 메서드가 있습니다.

1. `registerBlogCreator()` : 블로그를 생성하는 생성자를 등록한다.
2. `registerTagFilterSetter()` : 태그 필터를 설정하는 메서드를 등록한다.

3. `registerAuthorFilterSetter()` : 작성자 필터를 설정하는 메서드를 등록한다.
4. `registerPostOrderSetter()` : 블로그 글의 정렬 방법을 설정하는 메서드를 등록한다.
5. `registerPostListGetter()` : 블로그 글 목록을 가져오는 메서드를 등록한다.
6. `registerPostAdder()` : 블로그에 글을 추가하는 메서드를 등록한다.
7. `registerPostTitleUpdater()` : 발행된 블로그 글의 제목을 바꾸는 메서드를 등록한다.
8. `registerPostBodyUpdater()` : 발행된 블로그 글의 본문을 바꾸는 메서드를 등록한다.
9. `registerPostTagAdder()` : 블로그 글에 태그를 추가하는 메서드를 등록한다.
10. `registerCommentAdder()` : 블로그 글에 댓글을 추가하는 메서드를 등록한다.
11. `registerSubcommentAdder()` : 댓글에 하위 댓글을 추가하는 메서드를 등록한다.
12. `registerCommentUpdater()` : 댓글의 내용을 바꾸는 메서드를 등록한다.
13. `registerSubcommentUpdater()` : 하위 댓글의 내용을 바꾸는 메서드를 등록한다.
14. `registerReactionAdder()` : 블로그 글에 리액션을 추가하는 메서드를 등록한다.
15. `registerReactionRemover()` : 블로그 글로부터 리액션을 제거하는 메서드를 등록한다.
16. `registerCommentUpvoter()` : 댓글을 추천하는 메서드를 등록한다.
17. `registerCommentDownvoter()` : 댓글을 비추천하는 메서드를 등록한다.
18. `registerCommentListGetter()` : 블로그 글에 달린 댓글들을 가져오는 메서드를 등록한다.
19. `registerSubcommentListGetter()` : 댓글에 달린 하위 댓글들을 가져오는 메서드를 등록한다.
20. `registerSubcommentUpvoter()` : 하위 댓글을 추천하는 메서드를 등록한다.
21. `registerSubcommentDownvoter()` : 하위 댓글을 비추천하는 메서드를 등록한다.

`academy.pocu.comp2500.assignment1.registry` 안에 들어있는 클래스들을 **변경하지 마세요**. 전혀 그럴 필요 없고 그냥 사용하시기만 하면 되요!

`Registry` 클래스를 사용하는 방법을 예를 들어 설명하겠습니다. 여러분이 `Foo` 라는 클래스를 만든 뒤, 그 안에 블로그에 새 글을 추가하는 `bar()` 라는 메서드를 구현했다고 합시다. 그러면 `App` 클래스 생성자에서 다음과 같이 `registerPostAdder()` 를 호출해 주세요.

```
package academy.pocu.comp2500.assignment1;

import academy.pocu.comp2500.assignment1.registry.Registry;

public class App {
    public App(Registry registry) {
        ...

        registry.registerPostAdder("Foo", "bar");

        ...
    }
}
```

이는 빌드봇이 새 글을 추가하는 기능을 테스트하려면 `Foo` 클래스 안에 있는 `bar()` 메서드를 호출해야 한다고 알려줍니다.

모든 메서드들을 등록한 뒤에는 `Program.java` 속에 아래의 코드를 추가한 뒤, 과제를 제출하기 전에 실행해보세요.

```
package academy.pocu.comp2500.assignment1.app;

import academy.pocu.comp2500.assignment1.App;
import academy.pocu.comp2500.assignment1.registry.Registry;

public class Program {

    public static void main(String[] args) {
        Registry registry = new Registry();
        App app = new App(registry);
        registry.validate();
    }
}
```

`validate()` 메서드는 `Registry` 클래스에 메서드를 등록할 때 오타 등을 내지 않았는지 확인해줍니다. 빌드봇이 메서드들을 찾지 못해 어쩔 수 없이 0점을 주기 전에 미리 확인해보는 게 좋겠죠? :) 참고로 이 메서드는 `assert` 키워드를 사용하니 위에서 말했던 VM 옵션을 지정해주는 것도 잊지 마세요.

5. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)