COMP2500 실습 7

목차

Java에 있는 모든 클래스는 `Object` 클래스를 상속받는다고 수업시간에 배웠습니다. `Object` 클래스에 정의된 메서드 중에 특히 다음의 세 메서드를 오버라이딩할 일이 많죠.

1. `toString()` : 이 메서드를 오버라이딩하여 개체의 문자열 표현을 바꿈
2. `equals()` : 이 메서드를 오버라이딩하여 두 개체의 같음을 판단하는 기준을 바꿈
3. `hashCode()` : 이 메서드를 오버라이딩하여 각 개체가 고유한 해시 코드 값을 가지도록 함. 이 해시 코드는 `HashMap` 이나 `HashSet` 의 키(key)로 사용함. `equals()` 메서드를 오버라이딩하면 보통 이 메서드도 오버라이딩해야 함.

언제 어떻게 이 세 메서드를 구현해야 하는지는 클래스를 사용하는 방법에 따라 달라집니다. 예를 들어 두 개체의 동치 여부를 판단할 때, 두 개체 속에 저장되어 있는 모든 데이터가 같을 때에만 이 두 개체가 같다고 할 수도 있고, ID 멤버 변수의 값만 같으면 이 둘이 같다고 할 수도 있습니다. 어느 경우이든 간에 `equals()` 메서드를 오버라이딩하여 이 동작을 정의해야 합니다.

이 실습에서는 다양한 클래스에서 `toString()`, `equals()`, `hashCode()` 를 여러 가지 방법으로 구현해보겠습니다.

1. 프로젝트를 준비한다

1. `Lab7` 폴더로 이동합니다.
2. 다음의 클래스들을 `academy.pocu.comp2500.lab7` 패키지 아래에 추가하세요.
    - `Author`
    - `Book`
    - `Genre`
    - `Bookshelf`
    - `Bundle`
    - `ReadingList`
3. 자, 스트레칭 한 번 하시고... 설계와 프로그래밍 시작하겠습니다. :)

2. 클래스들 구현하기

전반적인 규칙

- 여러분이 작성한 코드는 반드시 `academy.pocu.comp2500.lab7` 패키지 안에 있어야 합니다.
- 정적 (static) 멤버 변수는 사용할 수 없습니다.
- 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 `Author` 클래스를 구현한다

- `Author` 클래스의 생성자는 다음의 인자를 받습니다.
    - 저자의 이름(first name)을 나타내는 `String`
    - 저자의 성(last name)을 나타내는 `String`
- 이 클래스의 문자열 표현은 다음의 포맷을 따라야 합니다.

<이름> <성>

예> James Bond

- 두 저자 개체는 성과 이름이 모두 같아야 동치입니다. 이에 따라 적절히 `equals()` 및 `hashCode()` 메서드를 구현하세요.

2.2 Book 클래스를 구현한다

- `Book` 클래스의 생성자는 다음의 인자를 받습니다.
  - 책의 제목을 나타내는 `String`
  - 책의 저자를 나타내는 `Author`
  - 책의 출판 연도를 의미하는 `int`
  - 책의 장르를 의미하는 `Genre`
- 장르로 사용할 수 있는 값은 다음과 같습니다.
  - 공상과학(Science Fiction)
  - 로맨스(Romance)
  - 위인전(Biography)
  - 판타지(Fantasy)
  - 미스터리(Mystery)
  - 서스펜스(Suspense)
- 이 클래스의 문자열 표현은 다음의 포맷을 따라야 합니다.

<제목> [<저자>]

예> Hello Coding [Pope Kim]

- 두 책 개체는 그 속에 있는 모든 데이터가 같아야 동치입니다. 이에 따라 적절히 `equals()` 및 `hashCode()` 메서드를 구현하세요.

2.3 Bookshelf 클래스를 구현한다

- `Bookshelf` 클래스의 생성자는 다음의 인자를 받습니다.
  - 책장에 꼽을 수 있는 최대 책의 수를 나타내는 `int`
- `add()` 메서드를 구현하세요.
  - `add()` 메서드는 유일한 인자로 `Book` 개체를 받습니다.
  - 책을 책장에 추가합니다.
  - 책장에 책이 추가되었다면 `true`를 아니라면 `false`를 반환합니다.
- `remove()` 메서드를 구현하세요.
  - `remove()` 메서드는 유일한 인자로 `Book` 개체를 받습니다.
  - 메서드의 인자와 같은 첫 번째 책을 책장에서 찾아 제거합니다.
  - 책장에서 책을 제거했다면 `true`를 반환합니다.
- 레퍼런스가 동일한 두 책장은 동치입니다. 이에 따라 적절히 `equals()` 및 `hashCode()` 메서드를 구현하세요.

2.4 Bundle 클래스를 구현한다

묶음상품(bundle)은 여러 책들의 집합(set)입니다. 각 묶음상품은 이름을 가지나 그 이름은 중복될 수 있습니다. 묶음상품의 예로 '크리스마스 도서 묶음'이 있다고 합시다. 이 묶음상품에는 4개의 책이 들어 있으며 고객은 이 묶음상품을 할인된 가격에 구매할 수 있습니다.

- `Bundle` 클래스의 생성자는 다음의 인자를 받습니다.
  - 해당 묶음상품의 이름을 나타내는 `String`
- `add()` 메서드를 구현하세요.
  - `add()` 메서드는 유일한 인자로 `Book` 개체를 받습니다.
  - 책을 묶음상품에 추가합니다.
  - 책이 추가되었다면 `true`를 아니라면 `false`를 반환합니다.
- `remove()` 메서드를 구현하세요.
  - `remove()` 메서드는 유일한 인자로 `Book` 개체를 받습니다.
  - 메서드의 인자와 같은 첫 번째 책을 묶음상품에서 찾아 그것을 제거합니다.
  - 묶음상품에서 책을 제거했다면 `true`를 반환합니다.
- 두 묶음상품은 이름이 같고 그 안에 있는 모든 책들이 같을 때 동치입니다. 이에 따라 적절히 `equals()` 및 `hashCode()` 메서드를 구현하세요.

2.5 ReadingList 클래스를 구현한다

'읽기 목록'은 도서의 목록입니다. 읽기 목록은 순서가 있으며 중복 도서를 담고 있을 수 있습니다.

- `ReadingList` 클래스의 생성자는 다음의 인자를 받습니다.

    - 읽기 목록의 이름을 나타내는 `String`
- `add()` 메서드를 구현하세요.
    - `add()` 메서드는 유일한 인자로 `Book` 개체를 받습니다.
    - 책을 읽기 목록에 추가합니다.
    - 이 함수는 아무것도 반환하지 않습니다.
- `remove()` 메서드를 구현하세요.
    - `remove()` 메서드는 유일한 인자로 `Book` 개체를 받습니다.
    - 메서드의 인자와 같은 첫 번째 책을 읽기 목록에서 찾아 그것을 제거합니다.
    - 읽기 목록에서 책을 제거했다면 `true`를 반환합니다.
- 이 클래스의 문자열 표현은 다음의 포맷을 따라야 합니다.

```
1. <첫 번째 책>
2. <두 번째 책>
3. <세 번째 책>
...
```

예>

```
1. Hello Coding [Pope Kim]
2. Intro to Professional Programming [Jane Doe]
3. Mathematics for Software Engineering [John Smith]
4. Hello Coding [Pope Kim]
```

- 두 읽기 목록은 이름이 같고 읽기 목록 안에 있는 책의 순서가 같으면 동치입니다. 이에 따라 적절히 `equals()` 및 `hashCode()` 메서드를 구현하세요.

3. 본인 컴퓨터에서 테스트하는 법

다음 테스트 코드에는 `toString()`, `eqauls()`, `hashCode()` 메서드를 테스트하는 코드가 누락되어 있습니다. 이건 의도된 바이며 여러분이 직접 테스트 코드를 작성하세요. :)

- `Program.java`를 아래처럼 바꾼 뒤 실행하세요.

```java
package academy.pocu.comp2500.lab7.app;

import academy.pocu.comp2500.lab7.Author;
import academy.pocu.comp2500.lab7.Book;
import academy.pocu.comp2500.lab7.Bookshelf;
import academy.pocu.comp2500.lab7.Bundle;
import academy.pocu.comp2500.lab7.Genre;
import academy.pocu.comp2500.lab7.ReadingList;

public class Program {

    public static void main(String[] args) {
        Author author = new Author("James", "Bond");
        Book book0 = new Book("How to be the best", author, 1990, Genre.BIOGRAPHY);
        Bookshelf bookshelf = new Bookshelf(10);

        assert (bookshelf.add(book0));
        assert (bookshelf.remove(book0));
        assert (!bookshelf.remove(book0));

        Book book1 = new Book("C# for dummies", new Author("Jason", "Bourne"), 2005, Genre.ROMANCE);
        Book book2 = new Book("C# for dummies", new Author("Jason", "Bourne"), 2005, Genre.ROMANCE);
        Book book3 = new Book("Java for dummies", new Author("James", "Bond"), 2007, Genre.MYSTERY);

        Bundle bundle = new Bundle("Programming");

        assert (bundle.add(book0));
        assert (bundle.add(book1));
        assert (!bundle.add(book2));
        assert (bundle.add(book3));

        assert (bundle.remove(book3));
        assert (bundle.remove(book0));
        assert (!bundle.remove(book0));

        ReadingList readingList = new ReadingList("Summer Break Homework");

        readingList.add(book0);
        readingList.add(book1);
        readingList.add(book2);
        readingList.add(book3);

        assert (readingList.remove(book3));
        assert (readingList.remove(book0));
        assert (!readingList.remove(book0));
    }
}
```

4. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)