COMP2500 실습 9

목차

포마존(Pomazon)은 신생 온라인 서점입니다. 이 회사로 이직하자마자 장바구니(Cart) 팀에 배정된 여러분은 막대한 사명감을 가지고 잠시 코드를 쓰윽 훑어보았습니다. 근데 `Cart` 클래스를 만든 프로그래머가 왠지 OOP를 잘 모르는 것 같네요. `Book` 클래스와의 결합도(coupling)가 꽤 높아서 `Book` 클래스가 바뀔 때 `Cart` 클래스도 손 봐야 하는 경우가 꽤 있을 듯합니다. 오늘 오전 회의에서 이 문제를 언급하니 팀장님도 동의하신다는군요. 오호~ 팀장님이 동의하셨다는 건 무슨 의미? 바로 여러분이 이걸 고쳐야 한다는 거지요!

1. 프로젝트를 준비한다

1. `Lab9` 폴더로 이동합니다.

2. 다음의 클래스들을 `src/academy/pocu/comp2500/lab9` 폴더에 추가합니다.

```java
package academy.pocu.comp2500.lab9;

import java.util.UUID;

public final class Book {
    private final UUID sku;
    private final String title;
    private final int price;
    private final int publishedYear;

    public Book(final UUID sku, final String title, final int price, final int publishedYear) {
        this.sku = sku;
        this.title = title;
        this.price = price;
        this.publishedYear = publishedYear;
    }

    public int getPrice() {
        return this.price;
    }

    public int getPublishedYear() {
        return this.publishedYear;
    }

    public String getTitle() {
        return this.title;
    }

    public UUID getSku() {
        return this.sku;
    }
}
```

```java
package academy.pocu.comp2500.lab9;

import java.util.ArrayList;
import java.util.UUID;

public final class Cart {
    private ArrayList<Book> books = new ArrayList<>();

    public Book getBookOrNull(final int index) {
        if (this.books.size() <= index) {
            return null;
        }

        return this.books.get(index);
    }

    public int getBookCount() {
        return this.books.size();
    }

    public void addBooks(final UUID[] skus, final String[] titles, final int[] prices, final int[] publis
        if (skus.length != titles.length || skus.length != prices.length || skus.length != publishedYear
            return;
        }

        for (int i = 0; i < skus.length; ++i) {
            Book book = new Book(skus[i], titles[i], prices[i], publishedYears[i]);
            this.books.add(book);
        }
    }

    public void addBook(final UUID sku, final String title, final int price, final int publishedYear) {
        Book book = new Book(sku, title, price, publishedYear);
        this.books.add(book);
    }

    public boolean remove(final int index) {
        if (this.books.size() <= index) {
            return false;
        }

        this.books.remove(index);

        return true;
    }

    public int getTotalPrice() {
        int sum = 0;

        for (Book book : this.books) {
            sum += book.getPrice();
        }

        return sum;
    }
}
```

3. 자, 스트레칭 한 번 하시고... 설계와 프로그래밍 시작하겠습니다. :)

2. 포마존 장바구니 구현하기

전반적인 규칙

- 기존 메서드의 반환형을 변경하는 것을 금합니다.
- 기존 메서드의 기능을 변경하는 것을 금합니다.
- 기존 메서드의 매개변수를 변경하는 것은 허용합니다. 하지만 무턱대고 아무거나 변경하면 점수를 잃을 수 있으니 잘 생각해보고 결정하세요! ;)
- 새로 추가하는 클래스들은 `academy.pocu.comp2500.lab9` 패키지에 속해야 합니다.
- `private static final` 멤버 변수 외에 어떤 정적 멤버 변수도 사용할 수 없습니다.
- `instanceof`, `Object.getClass()` 그리고 `Class<T>.cast()` 메서드를 사용할 수 없습니다. 사용하면 곧바로 0점이 뜨니 시도조차 하지 마세요. :)
- 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 `Cart` 와 `Book` 의 커플링을 제거한다

현재 `Cart` 와 `Book` 클래스는 커플링 되어 있습니다. 이걸 제거하세요.

2.2 가격결정 모델의 도입

세일즈 팀이 한시적으로 일부 책들에 대해 특별 할인 이벤트를 진행하고 싶어 합니다. 특별 할인 이벤트가 진행 중이면 장바구니 안에 있는 책의 총 가격을 다르게 계산해야 하죠. 이걸 지원하려면 가격결정 모델(pricing model)을 표현하는 새로운 클래스를 만들어야 한다고 팀장님이 말씀하셨습니다. 일단 세일즈 팀이 조만간 1+1 기획을 진행한다고 하니 `BuyOneGetOneFree` 클래스부터 만드세요.

2.2.1 `BuyOneGetOneFree` 가격결정 모델을 구현한다

이 가격결정 모델의 자세한 명세는 문서 제일 아래에 있는 **부록 A.1**을 참고하세요.

- 생성자는 다음의 인자를 받습니다.
  - 기획의 대상이 되는 책들의 SKU 번호들을 나타내는 `HashSet`
- 이 클래스의 유일한 메서드는 `getTotalPrice()` 입니다.
  - 이 메서드는 컬렉션(collection)(배열이 아님!)을 유일한 인자로 받습니다. 그 컬렉션 안에는 장바구니에 담겨있는 모든 책들이 들어있습니다.
  - 1+1 가격결정 모델에 따라 계산한 최종 가격을 반환합니다.
- SKU가 동일한 책은 제목, 가격, 출판연도도 동일하다고 가정하셔도 좋습니다.

2.2.2 `Cart` 클래스에 있는 `getTotalPrice()` 메서드를 오버로딩한다.

`BuyOneGetOneFree` 개체를 인자로 받는 `getTotalPrice()` 메서드를 구현하세요.

2.3 다른 가격결정 모델 구현하기

세일즈 팀이 또 다른 기획을 해왔습니다. 이번에는 두 개네요. 첫 번째 기획은 '동시대 서적'으로 같은 시기에 출시된 책들을 여럿 구매하면 할인가를 적용해주는 기획입니다. 동시대의 기준을 10년 단위로 잡는다고 하니 영어로는 Decade Madness라고 부르기로 했다는군요. 두 번째 기획은 '한계 따윈 없다'로 특정 조건에 따라 가장 비싼 책 두 권을 할인해주는 기획입니다. 영어로는 Sky is the Limit이라고 부르기로 했다는군요. 이 둘을 구현하세요.

2.3.1 `DecadeMadness` 가격결정 모델을 구현한다

이 가격결정 모델의 자세한 명세는 **부록 A.2**를 참고하세요.

- 생성자는 아무런 인자도 받지 않습니다.
- 이 클래스의 유일한 메서드는 `getTotalPrice()` 입니다.
  - 이 메서드가 받는 유일한 인자는 북 컬렉션입니다.
  - '동시대 서적' 가격결정 모델에 따라 계산한 최종 가격을 반환합니다.
  - 계산을 할 때는 `double` 을 사용하지만 가격을 반환할 때는 소수점 이하 값들을 버리고 `int` 를 반환하세요.
- 이 가격결정 모델을 사용할 수 있도록 `Cart` 클래스를 변경하는 것도 잊지 마세요.

2.3.2 `SkyIsTheLimit` 가격결정 모델을 구현한다

이 가격결정 모델의 자세한 명세는 **부록 A.3**을 참고하세요.

- 생성자는 다음의 인자를 받습니다.
  - `int price`: 이 가격결정 모델에 정의된 할인이 적용되기 시작하는 최소 가격
- 이 클래스의 유일한 메서드는 `getTotalPrice()` 입니다.
  - 이 메서드가 받는 유일한 인자는 북 컬렉션입니다.
  - '한계 따윈 없다' 가격결정 모델에 따라 계산한 최종 가격을 반환합니다.
  - 계산을 할 때는 `double` 을 사용하지만 가격을 반환할 때는 소수점 이하 값들을 버리고 `int` 를 반환하세요.
- 이 가격결정 모델을 사용할 수 있도록 `Cart` 클래스를 변경하는 것도 잊지 마세요.

2.3.3 `SimplePricing` 가격결정 모델을 구현한다

어떤 이벤트도 진행 중이 아니라면 `SimplePricing` 모델을 사용해야 합니다. 최종 가격은 단순히 장바구니 안에 있는 각 책의 가격을 더한 결과입니다.

- 생성자는 아무런 인자도 받지 않습니다.
- 이 클래스의 유일한 메서드는 `getTotalPrice()` 입니다.
  - 이 메서드가 받는 유일한 인자는 북 컬렉션입니다.
  - 각 책의 가격을 합한 결과를 반환합니다.
- 이 가격결정 모델을 사용할 수 있도록 `Cart` 클래스를 변경하는 것도 잊지 마세요.

2.3.4 `Cart` 클래스에 있는 결합도를 줄이세요

팀장님이 말씀하시길 `Cart` 클래스는 가격결정 모델 클래스들과의 결합도가 높으면 안 된다고 하십니다. 장바구니 팀은 장바구니에 책을 추가/삭제하는 등의 일만 담당하고 가격 계산 로직이 바뀌는 건 다른 팀이 책임지는 게 맞다네요. 따라서 앞으로는 세일즈 팀이 새로운 기획을 준비할 때 그걸 지원하는 코드를 작성하는 것은 다른 팀의 일이며, 장바구니 팀이 관리하는 코드는 바뀌지 않아야 한다고 합니다.

완전히 커플링을 제거하려면 앞에서 했던 일들을 좀 바꿔야할 수도 있습니다 :P

3. 본인 컴퓨터에서 테스트하는 법

다음 코드는 `Cart` 클래스에 있는 메서드들을 테스트하지 않습니다. 이건 의도된 바이며 여러분이 직접 테스트 코드를 작성하세요. :)

- `Program.java` 를 아래처럼 바꾼 뒤 실행하세요.

```
package academy.pocu.comp2500.lab9.app;

import academy.pocu.comp2500.lab9.Book;
import academy.pocu.comp2500.lab9.BuyOneGetOneFree;
import academy.pocu.comp2500.lab9.DecadeMadness;
import academy.pocu.comp2500.lab9.SkyIsTheLimit;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.UUID;

public class Program {

    public static void main(String[] args) {
        UUID sku0 = UUID.randomUUID();

        Book book0 = new Book(sku0, "Hello", 10, 1980);
        Book book1 = new Book(sku0, "Hello", 10, 1980);
        Book book2 = new Book(sku0, "Hello", 10, 1980);
        Book book3 = new Book(sku0, "Hello", 10, 1980);
        Book book4 = new Book(sku0, "Hello", 10, 1980);
        Book book5 = new Book(UUID.randomUUID(), "Millenium", 15, 2001);
        Book book6 = new Book(UUID.randomUUID(), "Halfway Millenium", 21, 2005);
        Book book7 = new Book(UUID.randomUUID(), "Decade almost over", 17, 2009);
        Book book8 = new Book(UUID.randomUUID(), "FIFA", 17, 2002);
        Book book9 = new Book(UUID.randomUUID(), "University", 5, 2008);

        ArrayList<Book> books = new ArrayList<>();

        books.add(book0);
        books.add(book1);
        books.add(book2);
        books.add(book3);
        books.add(book4);
        books.add(book5);
        books.add(book6);
        books.add(book7);
        books.add(book8);
        books.add(book9);

        HashSet<UUID> skus = new HashSet<>();
        skus.add(sku0);

        BuyOneGetOneFree model0 = new BuyOneGetOneFree(skus);
        DecadeMadness model1 = new DecadeMadness();
        SkyIsTheLimit model2 = new SkyIsTheLimit(100);

        assert (model0.getTotalPrice(books) == 105);
        assert (model1.getTotalPrice(books) == 100);
        assert (model2.getTotalPrice(books) == 106);
    }
}
```

4. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)

A. 부록

A.1 1+1 기획(Buy One Get One Free)

이 기획전에서는 세일즈 팀이 대상이 되는 책의 SKU를 지정합니다. 만약 고객이 대상이 되는 책 중에 하나를 두 권 구매하면, 두 번째 책은 공짜입니다.

예를 들어 현재 기획 대상인 책의 SKU가 123과 321이고 장바구니에 다음과 같은 책이 들어있다고 해봅시다.

1. Book1
    - SKU: 123
    - 제목: "Galactic War"

    - 가격: 10
    - 출판연도 : 1991
2. Book2
    - SKU: 123
    - 제목: "Galactic War"
    - 가격: 10
    - 출판연도 : 1991
3. Book3
    - SKU: 123
    - 제목: "Galactic War"
    - 가격: 10
    - 출판연도 : 1991
4. Book4
    - SKU: 123
    - 제목: "Galactic War"
    - 가격: 10
    - 출판연도 : 1991
5. Book5
    - SKU: 123
    - 제목: "Galactic War"
    - 가격: 10
    - 출판연도: 1991
6. Book6
    - SKU: 321
    - 제목: "The Rebels"
    - 가격: 20
    - 출판연도 : 1920
7. Book7
    - SKU: 321
    - 제목: "The Rebels"
    - 가격: 20
    - 출판연도 : 1920

** 간단한 예를 위해 SKU에 정수를 사용했지만 실제로는 UUID여야 합니다.

이때 `BuyOneGetOneFree` 가격결정 모델을 사용하면 최종 가격은 10 + 10 + 10 + 20 = 50입니다. 이것은 Book1, Book2, Book3, Book4, Book5가 모두 동일한 SKU이기 때문에 이 중에 두 권이 공짜가 되기 때문입니다. 또한 book6과 Book7도 같으므로 Book7이 공짜입니다.

A.2 동시대 서적 기획(Decade Madness)

이 가격결정 모델에서는 동시대(10년 기준)에 출판된 책을 두 권 이상 구매하면 그 책들을 20% 할인해줍니다. 예를 들어 장바구니에 다음 책들이 들어있다고 해봅시다.

1. Book1
    - SKU: 123
    - 제목: "Galactic War"
    - 가격: 10
    - 출판연도 : 1991
2. Book2
    - SKU: 142
    - 제목: "You and Me"
    - 가격: 15
    - 출판연도 : 1995
3. Book3
    - SKU: 155
    - 제목: "Me and You"
    - 가격: 10
    - 출판연도 : 1996
4. Book4
    - SKU: 130
    - 제목: "R2D2"
    - 가격: 20
    - 출판연도 : 2011
5. Book5

    - SKU: 321
    - 제목: "The Rebels"
    - 가격: 20
    - 출판연도 : 2003
6. Book6
    - SKU: 167
    - 제목: "Teehee"
    - 가격: 10
    - 출판연도 : 2001

Book1, Book2, Book3은 모두 90년대(1990 - 1999)에 출판된 책입니다. 따라서 이 모두는 20% 할인가를 적용받습니다. Book4는 2010년대(2010 - 2019)에 출판된 유일한 책이므로 정가대로 계산하며, Book5와 Book6는 모두 2000년대(2000 - 2009)에 출판된 책이므로 역시 20% 할인 대상입니다. 이에 따라 최종 가격을 계산하면 다음과 같습니다.

```
0.8 * (10 + 15 + 10) + 20 + 0.8 * (20 + 10) = 72
```

A.3 한계 따윈 없다 기획(Sky is the Limit)

이 가격결정 모델에서는 고객이 장바구니에 5개 이상의 책을 추가하고 그 책의 가격을 모두 더한 결과가 x 이상인 경우, 장바구니에 있는 책 중에 가장 비싼 두 권이 반값이 됩니다. 예를 들어 x = 80이고 장바구니에 들어있는 책이 다음과 같다고 합시다.

1. Book1
    - SKU: 123
    - 제목: "Galactic War"
    - 가격: 10
    - 출판연도 : 1991
2. Book2
    - SKU: 142
    - 제목: "You and Me"
    - 가격: 15
    - 출판연도 : 1995
3. Book3
    - SKU: 155
    - 제목: "Me and You"
    - 가격: 10
    - 출판연도 : 1996
4. Book4
    - SKU: 130
    - 제목: "R2D2"
    - 가격: 20
    - 출판연도 : 2011
5. Book5
    - SKU: 321
    - 제목: "The Rebels"
    - 가격: 20
    - 출판연도 : 2003
6. Book6
    - SKU: 167
    - 제목: "Teehee"
    - 가격: 10
    - 출판연도 : 2001

장바구니에 있는 책들의 가격을 모두 더하면 85입니다. 이는 80 이상이므로 가장 비싼 책인 Book4와 Book5에 50% 할인율이 적용됩니다. 따라서 최종 가격은 다음과 같이 됩니다.

```
10 + 15 + 10 + 20 * 0.5 + 20 * 0.5 + 10 = 65
```