COMP2500 과제 2

목차

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

지난 3주간 블로깃(BlogIt)에서 야근에 시달린 여러분은 좀 더 즐겁게 일 할 수 있는 회사로 이직했습니다. 어느 정도 물이 오른 엔지니어인 여러분은 프린팃(PrintIt)이라는 회사에 취직을 했죠. 이 회사는 기업고객을 상대로 사무용품을 판매하는 전자상거래 플랫폼을 개발하고 있습니다. 이 회사에서 판매하는 상품 중에는 스탬프, 달력, 배너(현수막), 명함 등 별도의 출력 과정을 거쳐야 하는 것들이 있습니다. 이 상품들은 고객들이 색상, 크기 등을 선택할 수 있을 뿐만 아니라 고객 마음대로 문구(text aperture)나 사진(image aperture)을 넣을 수도 있습니다. 그러면 PrintIt 내부 인쇄소에서 이 디자인을 출력해 최종 완성품을 만들어 고객에게 전달합니다.

다음은 고객이 PrintIt 웹사이트에서 상품을 구매하는 흔한 모습을 설명한 것입니다.

1. https://design.printit.com 에서 상품 목록을 본다.
2. 배너를 고른다.
3. 크기, 재질, 색상 등의 옵션을 선택한다. (이 옵션들은 상품마다 다를 수 있음)
4. 본인이 원하는 문구나 사진을 추가한다. (자유로이 문구나 사진을 추가할 수 없는 상품들도 있음)
5. 완성된 디자인을 장바구니(cart)에 추가한다.
6. 장바구니 페이지에서 각 구매품마다 배송 방법을 선택한다.
7. 결제

여러분은 위 과정 중에 단계 5부터 7까지를 책임지고 코딩해야 합니다. 직속상관인 김지만 팀장님이 여러분에게 5~7 단계를 관리할 클래스들을 설계하라는 지시를 내리면서 다음과 같은 매개변수명의 사용을 허락하였습니다.

```
product
card
banner
poster
stamp
calendar
wallCalendar
magnetCalendar
deskCalendar
signature
print
text
color
paperColor
bannerColor
backgroundColor
stampColor
cardColor
calendarColor
size
stampSize
bannerSize
cardSize
calendarSize
bannerMaterial
material
type
paperType
calendarType
stampType
bannerType
cardType
orientation
cardOrientation
bannerOrientation
calendarOrientation
stampOrientation
emblem
index
deliveryMethod
delivery
element
textElement
imageElement
image
x
y
price
totalPrice
sides
cardSides
bannerSides
calendarSides
j
k
landscapeCard
portraitCard
ivoryCard
whiteCard
grayCard
laidCard
linenCard
smoothCard
glossBanner
scrimBanner
meshBanner
singleSidedCard
doubleSidedCard
width
height
length
cart
id
productId
apertureId
displayName
r
g
b
businessCard
businessCardColor
```

```
paperColor
dimensions
businessCardSize
paperStock
paperMaterial
stock
businessCardType
businessCardOrientation
i
shippingMethod
shipment
aperture
textAperture
imageAperture
imagePath
businessCardSides
landscapeBusinessCard
portraitBusinessCard
singleSidedBusinessCard
doubleSidedBusinessCard
shoppingCart
name
red
green
blue
elementId
```

팀장님 말에 따르면 이 클래스로부터 생성한 개체들을 출력소에도 보낸다고 합니다. 출력소 직원들이 정확히 제품을 찍으려면 크기, 재질, 색상, 문구, 이미지 경로 등 필요한 정보들을 읽을 수 있어야 한다는군요. 그렇다고 클래스 설계의 추상화/일반화를 망가뜨리지 마세요. 출력소의 시스템은 필요에 따라 다운 캐스팅(downcast, 자식 클래스로 명시적으로 캐스팅함)을 한다고 합니다. 따라서 부모 클래스가 자식 클래스에 대해 너무 많이 알게 만들지 마세요.

또한 팀장님은 제품 사양들과 장바구니의 UI 디자인도 주셨습니다. 여러분의 전문 OOP 지식을 이용하여 퍼펙트한 클래스들을 설계하세요! 그래야 이 클래스들을 사용할 새 동료들이 행복해하겠죠?

1. 프로젝트를 준비한다

1. `Assignment2` 프로젝트를 엽니다.
2. 자, 이제 자리에서 일어나서 국민체조 한 번 하시고... 설계와 프로그래밍 시작하겠습니다. :)

2. 장바구니(cart) 시스템을 설계 및 구현한다

2.1 전반적인 규칙

- 아래에 나와있는 제품 사양과 UI 디자인에 따라 마음껏 클래스를 설계하세요. 그러나 빌드봇이 여러분의 설계가 적절하지 못하다 생각하면 점수를 깎을 것입니다. :P
- 여러분이 작성한 코드는 아래의 UI 디자인을 제대로 지원할 수 있어야 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`
- 여러분이 작성하는 클래스들은 반드시 `academy.pocu.comp2500.assignment2` 패키지 안에 속해야 합니다.
- 날짜와 시간을 표현하는 자료형으로는 `OffsetDateTime` 을 사용해야 합니다.
- 색상을 반환할 때는 RGB 값을 사용해야 합니다.
- 크기를 표현할 때는 밀리미터를 사용해야 합니다.
- 본 과제에서 다형성은 사용하실 수 없습니다.
- `instanceof` 와 `getClass()` 메서드도 허용하지 않습니다. 사용하면 0점이 뜨니 시도조차 하지 마세요. :)
- public 정적 (`static`) 함수의 사용을 금합니다
- 클래스 내부에 데이터를 저장할 때 사용하는 자료형은 직접 만들어 사용하세요. (예외: String과 Map, ArrayList, HashSet와 같은 컨테이너, 그리고 기본 자료형)
- 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.

2.2 제품 사양

다음은 PrintIt에서 판매하는 모든 출력 가능한 제품의 사양입니다.

1. 스탬프(Stamp)

스탬프	크기	가격
스탬프(잉크: 빨강)	4 x 3 cm	2300
	5 x 2 cm	2300
	7 x 4 cm	2600
스탬프(잉크: 파랑)	4 x 3 cm	2300
	5 x 2 cm	2300
	7 x 4 cm	2600
스탬프(잉크: 녹색)	4 x 3 cm	2300
	5 x 2 cm	2300
	7 x 4 cm	2600

추가 정보

- 고객이 직접 선택한 문구가 스탬프 사용 시 찍힙니다.
- 잉크 색상의 RGB 값은 다음과 같습니다.
  - 빨강(RGB(FF,0,0))
  - 파랑(RGB(0,0,FF))
  - 녹색(RGB(0,80,0))

2. 달력(Calendar)

달력	크기	색상	가격
벽걸이(Wall) 달력	40 x 40 cm	흰색	1000
탁상(Desk) 달력	20 x 15 cm	흰색	1000
자석(Magnet) 달력	10 x 20 cm	흰색	1500

3. 배너/현수막(Banner)

배너	크기	가격
반사(Gloss) 배너	1 x 0.5 m	5000
	1 x 1 m	5200
	2 x 0.5 m	5300
	3 x 1 m	6000
스크림(Scrim) 배너	1 x 0.5 m	5100
	1 x 1 m	5300
	2 x 0.5 m	5400
	3 x 1 m	6100
메쉬(Mesh) 배너	1 x 0.5 m	5100
	1 x 1 m	5300
	2 x 0.5 m	5400
	3 x 1 m	6100

추가 정보

- 색상: RGB로 표현 가능한 아무 색상
- 출력 방향: 세로 방향(portrait) 또는 가로 방향(landscape)
- 고객이 자유로이 문구나 사진을 추가하고 위치를 변경할 수 있음 (출력되는 문구/사진마다 +5의 추가 비용 발생)
- 제품에 사진을 추가하려는 고객은 우선 프린팃의 서버에 사진을 업로드한 뒤에야 그 이미지를 사용할 수 있음. (즉, 웹 서버에 이미지가 저장되어 있음)

4. 명함(Business Card)

명함	단면/양면	가격
린넨커버(Linen) 명함	단면	110
	양면	140
레이드지(Laid) 명함	단면	120
	양면	150
스노우지(Smooth) 명함	단면	100
	양면	130

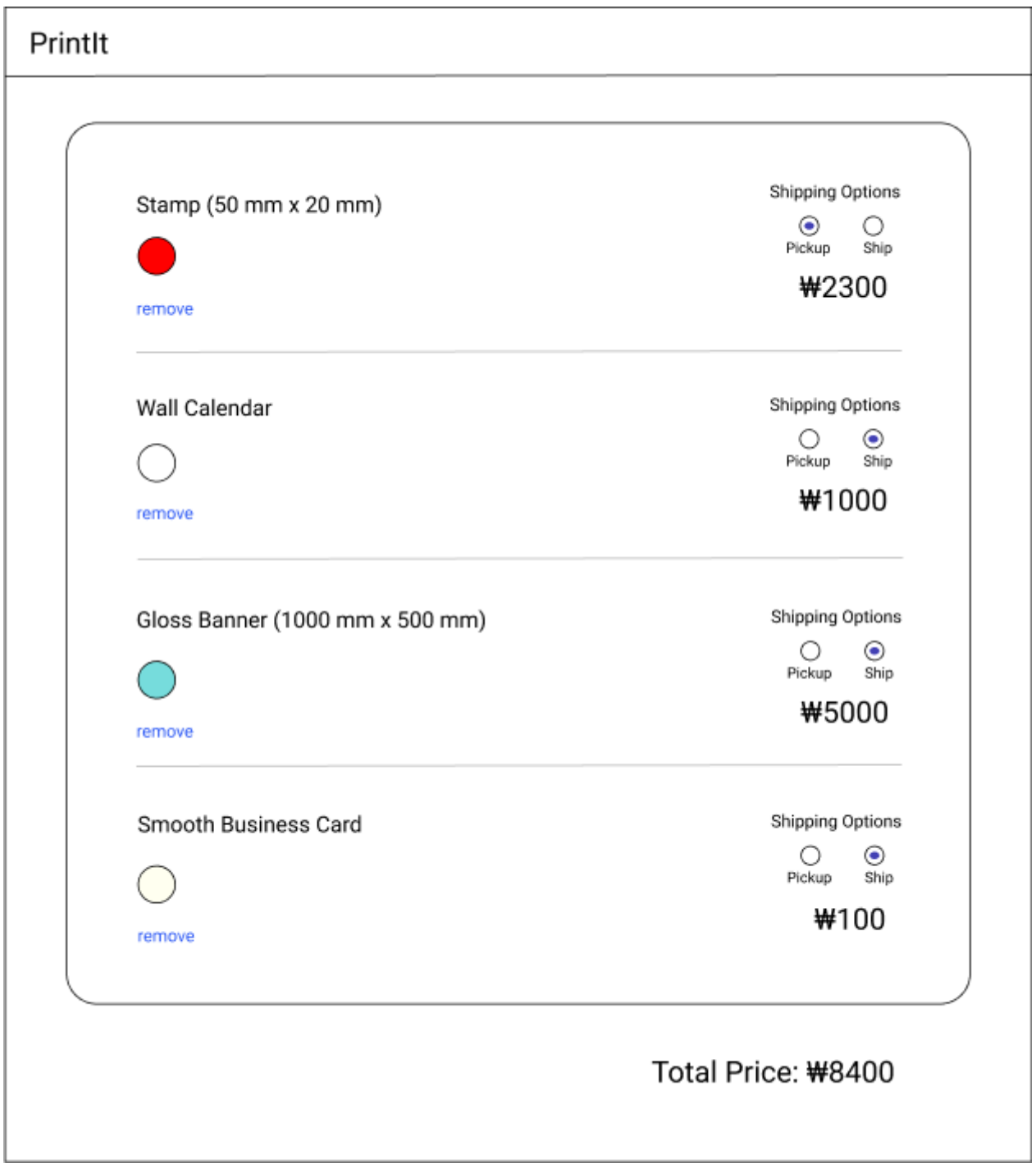
추가 정보

- 모든 명함의 크기는 9 x 5 cm 입니다.
- 출력 방향: 세로 방향(portrait) 또는 가로 방향(landscape)
- 종이 색: 회색(RGB(E6,E6,E6)), 아이보리(RGB(FF,FF,F0)), 흰색(RGB(FF,FF,FF))
- 고객이 자유로이 문구나 사진을 추가하고 위치를 변경할 수 있음 (출력되는 문구/사진마다 +5의 추가 비용 발생)

- 제품에 사진을 추가하려는 고객은 우선 프린팃의 서버에 사진을 업로드한 뒤에야 그 이미지를 사용할 수 있음. (즉, 웹 서버에 이미지가 저장되어 있음)

2.3 장바구니 UI 디자인

다음은 장바구니의 UI 디자인입니다.



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 빨강 잉크 4 x 3 cm 스탬프 만들기

1. 빨강 잉크가 들은 4 x 3 cm 스탬프를 만든다.
2. 생성 중에 초기화한 멤버 변수들의 값이 올바른지 확인한다.
3. 가격이 올바른지 확인한다.
4. 올바른 색상이 반환되는지 확인한다.

3.2 빨강 잉크 4 x 3 cm 스탬프의 배송 방법 바꾸기

1. 빨강 잉크가 들은 4 x 3 cm 스탬프를 장바구니에 추가한다.
2. 배송 방법을 바꾼다.
3. 빨강 잉크가 들은 4 x 3 cm 스탬프의 배송 방법이 제대로 바뀌었는지 확인한다.

3.3 하얀색 40 x 40 cm 벽걸이 달력 만들기

1. 하얀색 40 x 40 cm 벽걸이 달력을 만든다.
2. 생성 중에 초기화한 멤버 변수들의 값이 올바른지 확인한다.
3. 가격이 올바른지 확인한다.
4. 올바른 크기가 반환되는지 확인한다.
5. 올바른 색상이 반환되는지 확인한다.

3.4 하얀색 40 x 40 cm 벽걸이 달력의 배송 방법 바꾸기

1. 하얀색 40 x 40 cm 벽걸이 달력을 장바구니에 추가한다.
2. 배송 방법을 바꾼다.
3. 하얀색 40 x 40 cm 벽걸이 달력의 배송 방법이 제대로 바뀌었는지 확인한다.

3.5 1 x 0.5 m 반사 배너 만들기

1. 1 x 0.5 m 반사 배너를 만든다.
2. 생성 중에 초기화한 멤버 변수들의 값이 올바른지 확인한다.
3. 가격이 올바른지 확인한다.
4. 크기가 올바른지 확인한다.
5. 색상이 올바른지 확인한다.
6. 출력 방향이 올바른지 확인한다.
7. 문구나 사진을 추가했다면 올바로 저장되어 있는지 확인한다.

3.6 1 x 0.5 m 반사 배너 꾸미기

1. 1 x 0.5 m 반사 배너를 만든다.
2. 새 문구나 사진을 추가한다
3. 새로 추가된 문구나 사진이 1 x 0.5 m 반사 배너 안에 있는지 확인한다.
4. 가격이 올바른지 확인한다.

3.7 1 x 0.5 m 반사 배너의 배송 방법 바꾸기

1. 1 x 0.5 m 반사 배너를 장바구니에 추가한다.
2. 배송 방법을 바꾼다.
3. 1 x 0.5 m 반사 배너의 배송 방법이 제대로 바뀌었는지 확인한다.

3.8 단면 명함 만들기

1. 단면 명함을 만든다.
2. 생성 중에 초기화한 멤버 변수들의 값이 올바른지 확인한다.
3. 가격이 올바른지 확인한다.
4. 크기가 올바른지 확인한다.
5. 종이 색상이 올바른지 확인한다.
6. 출력면(단면/양면)이 올바른지 확인한다.
7. 인쇄용지가 올바른지 확인한다.
8. 출력 방향이 올바른지 확인한다.
9. 문구나 사진을 추가했다면 올바로 저장되어 있는지 확인한다.

3.9 단면 명함 꾸미기

1. 단면 명함을 만든다.
2. 새 문구나 사진을 추가한다.
3. 새로 추가된 문구나 사진이 단면 명함 안에 있는지 확인한다.
4. 가격이 올바른지 확인한다.

3.10 단면 명함의 배송 방법 바꾸기

1. 단면 명함을 장바구니에 추가한다.
2. 배송 방법을 바꾼다.
3. 단면 명함의 배송 방법이 제대로 바뀌었는지 확인한다.

3.11 레이드지 명함 만들기

1. 레이드지 명함을 만든다.
2. 생성 중에 초기화한 멤버 변수들의 값이 올바른지 확인한다.
3. 가격이 올바른지 확인한다.
4. 크기가 올바른지 확인한다.
5. 종이 색상이 올바른지 확인한다.
6. 출력면(단면/양면)이 올바른지 확인한다.
7. 인쇄용지가 올바른지 확인한다.
8. 출력 방향이 올바른지 확인한다.
9. 문구나 사진을 추가했다면 올바로 저장되어 있는지 확인한다.

3.12 레이드지 명함 꾸미기

1. 레이드지 명함을 만든다.
2. 새 문구나 사진을 추가한다
3. 새로 추가된 문구나 사진이 레이드지 명함 안에 있는지 확인한다.
4. 가격이 올바른지 확인한다.

3.13 레이드지 명함의 배송 방법 바꾸기

1. 레이드지 명함을 장바구니에 추가한다.
2. 배송 방법을 바꾼다.
3. 레이드지 명함의 배송 방법이 제대로 바뀌었는지 확인한다.

3.14 장바구니를 만든다

1. 장바구니를 만든다.
2. 생성 중에 초기화한 멤버 변수들의 값이 올바른지 확인한다.

3.15 장바구니에 상품 추가하기

1. 장바구니를 만든다.
2. 장바구니에 상품을 하나 추가한다.
3. 장바구니에 상품이 제대로 들어있는지 확인한다.

3.16 장바구니에서 상품 제거하기

1. 장바구니를 만든다.
2. 장바구니에 상품을 하나 추가한다.
3. 장바구니에 상품이 제대로 들어있는지 확인한다.
4. 장바구니로부터 그 상품을 제거한다.
5. 장바구니에 그 상품이 들어있지 않은지 확인한다.

3.17 총액 확인하기

1. 장바구니를 만든다.
2. 장바구니에 여러 상품을 추가한다.
3. 총액을 가져와서 그 금액이 올바른지 확인한다.

4. 클래스와 메서드 등록하기

본인 컴퓨터에서 프로그램 테스트를 마쳤다면 `academy.pocu.comp2500.assignment2.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 55개의 `registerXXX()` 메서드가 있습니다.

1. `registerRedStampCreator()` : 빨강 잉크 스탬프를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
2. `registerBlueStampCreator()` : 파랑 잉크 스탬프를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
3. `registerGreenStampCreator()` : 녹색 잉크 스탬프를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
4. `registerWallCalendarCreator()` : 벽걸이 달력을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
5. `registerMagnetCalendarCreator()` : 자석 달력을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
6. `registerDeskCalendarCreator()` : 탁상 달력을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
7. `registerLandscapeBannerCreator()` : 가로 방향 배너를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
8. `registerPortraitBannerCreator()` : 세로 방향 배너를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
9. `registerGlossBannerCreator()` : 반사 배너를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
10. `registerScrimBannerCreator()` : 스크림 배너를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
11. `registerMeshBannerCreator()` : 메쉬 배너를 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.

12. `registerLandscapeBusinessCardCreator()` : 가로 방향 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
13. `registerPortraitBusinessCardCreator()` : 세로 방향 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
14. `registerIvoryBusinessCardCreator()` : 아이보리 종이에 찍는 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
15. `registerGrayBusinessCardCreator()` : 회색 종이에 찍는 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
16. `registerWhiteBusinessCardCreator()` : 흰색 종이에 찍는 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
17. `registerLaidBusinessCardCreator()` : 레이드지 인쇄용지에 찍는 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
18. `registerLinenBusinessCardCreator()` : 린넨커버 인쇄용지에 찍는 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
19. `registerSmoothBusinessCardCreator()` : 스노우지 인쇄용지에 찍는 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
20. `registerSingleSidedBusinessCardCreator()` : 단면 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
21. `registerDoubleSidedBusinessCardCreator()` : 양면 명함을 만드는 생성자나 메서드를 등록한다. 2개의 오버로딩된 메서드가 있으니 본인의 설계에 적합한 버전을 사용할 것.
22. `registerCartCreator()` : 장바구니를 만드는 생성자를 등록한다.
23. `registerProductAdder()` : 장바구니에 상품을 추가하는 메서드를 등록한다.
24. `registerProductRemover()` : 장바구니에서 상품을 제거하는 메서드를 등록한다.
25. `registerTotalPriceGetter()` : 장바구니에서 총액을 구해오는 메서드를 등록한다.
26. `registerLandscapeBannerTextApertureAdder()` : 가로 방향 배너에 문구를 추가하는 메서드를 등록한다.
27. `registerLandscapeBannerImageApertureAdder()` : 가로 방향 배너에 사진을 추가하는 메서드를 등록한다.
28. `registerPortraitBannerTextApertureAdder()` : 세로 방향 배너에 문구를 추가하는 메서드를 등록한다.
29. `registerPortraitBannerImageApertureAdder()` : 세로 방향 배너에 사진을 추가하는 메서드를 등록한다.
30. `registerGlossBannerTextApertureAdder()` : 반사 배너에 문구를 추가하는 메서드를 등록한다.
31. `registerGlossBannerImageApertureAdder()` : 반사 배너에 사진을 추가하는 메서드를 등록한다.
32. `registerScrimBannerTextApertureAdder()` : 스크림 배너에 문구를 추가하는 메서드를 등록한다.
33. `registerScrimBannerImageApertureAdder()` : 스크림 배너에 사진을 추가하는 메서드를 등록한다.
34. `registerMeshBannerTextApertureAdder()` : 메쉬 배너에 문구를 추가하는 메서드를 등록한다.
35. `registerMeshBannerImageApertureAdder()` : 메쉬 배너에 사진을 추가하는 메서드를 등록한다.
36. `registerLandscapeBusinessCardTextApertureAdder()` : 가로 방향 명함에 문구를 추가하는 메서드를 등록한다.
37. `registerLandscapeBusinessCardImageApertureAdder()` : 가로 방향 명함에 사진을 추가하는 메서드를 등록한다.
38. `registerPortraitBusinessCardTextApertureAdder()` : 세로 방향 명함에 문구를 추가하는 메서드를 등록한다.
39. `registerPortraitBusinessCardImageApertureAdder()` : 세로 방향 명함에 사진을 추가하는 메서드를 등록한다.
40. `registerIvoryBusinessCardTextApertureAdder()` : 아이보리 종이에 찍는 명함에 문구를 추가하는 메서드를 등록한다.
41. `registerIvoryBusinessCardImageApertureAdder()` : 아이보리 종이에 찍는 명함에 사진을 추가하는 메서드를 등록한다.
42. `registerGrayBusinessCardTextApertureAdder()` : 회색 종이에 찍는 명함에 문구를 추가하는 메서드를 등록한다.
43. `registerGrayBusinessCardImageApertureAdder()` : 회색 종이에 찍는 명함에 사진을 추가하는 메서드를 등록한다.
44. `registerWhiteBusinessCardTextApertureAdder()` : 흰색 종이에 찍는 명함에 문구를 추가하는 메서드를 등록한다.
45. `registerWhiteBusinessCardImageApertureAdder()` : 흰색 종이에 찍는 명함에 사진을 추가하는 메서드를 등록한다.
46. `registerLaidBusinessCardTextApertureAdder()` : 레이드지 인쇄용지에 찍는 명함에 문구를 추가하는 메서드를 등록한다.
47. `registerLaidBusinessCardImageApertureAdder()` : 레이드지 인쇄용지에 찍는 명함에 사진을 추가하는 메서드를 등록한다.
48. `registerLinenBusinessCardTextApertureAdder()` : 린넨커버 인쇄용지에 찍는 명함에 문구를 추가하는 메서드를 등록한다.
49. `registerLinenBusinessCardImageApertureAdder()` : 린넨커버 인쇄용지에 찍는 명함에 사진을 추가하는 메서드를 등록한다.
50. `registerSmoothBusinessCardTextApertureAdder()` : 스노우지 인쇄용지에 찍는 명함에 문구를 추가하는 메서드를 등록한다.
51. `registerSmoothBusinessCardImageApertureAdder()` : 스노우지 인쇄용지에 찍는 명함에 사진을 추가하는 메서드를 등록한다.
52. `registerSingleSidedBusinessCardTextApertureAdder()` : 단면 명함에 문구를 추가하는 메서드를 등록한다.
53. `registerSingleSidedBusinessCardImageApertureAdder()` : 단면 명함에 사진을 추가하는 메서드를 등록한다.
54. `registerDoubleSidedBusinessCardTextApertureAdder()` : 양면 명함에 문구를 추가하는 메서드를 등록한다.
55. `registerDoubleSidedBusinessCardImageApertureAdder()` : 양면 명함에 사진을 추가하는 메서드를 등록한다.

`academy.pocu.comp2500.assignment2.registry` 안에 들어있는 클래스들을 **변경하지 마세요**. 전혀 그럴 필요 없고 그냥 사용하시기만 하면 되요!

`Registry` 클래스를 사용하는 방법을 예를 들어 설명하겠습니다. 여러분이 빨강 잉크 스탬프를 나타내는 `Foo` 라는 클래스를 만들었다고 합시다. 그러면 `App` 클래스 생성자에서 다음과 같이 `registerRedStampCreator()` 를 호출해 주세요.

```
package academy.pocu.comp2500.assignment2;

import academy.pocu.comp2500.assignment2.registry.Registry;

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

        registry.registerRedStampCreator("Foo");

        ...
    }
}
```

그게 아니라 `Foo` 클래스에 정의된 `bar()` 메서드가 빨강 잉크 스탬프를 만든다면 다음과 같이 해주세요.

```
package academy.pocu.comp2500.assignment2;

import academy.pocu.comp2500.assignment2.registry.Registry;

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

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

        ...
    }
}
```

이러면 빌드봇이 빨강 잉크 스탬프를 만드는 법을 알게 됩니다.

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

```
package academy.pocu.comp2500.assignment2.app;

import academy.pocu.comp2500.assignment2.App;
import academy.pocu.comp2500.assignment2.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 적절한 VM 옵션을 지정해주는 것도 잊지 마세요.

5. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)