⚠ 본 사이트를 이용함으로써 **쿠키 정책**과 **개인정보처리방침**을 읽고 이해하였다는 의사표시를 하게 됩니다. 동의함

COMP2500 실습 6

목차

오~ 제가 해낸 것 같아요! 다음과 같은 최고의 피자 레시피를 만들었는데 엄청 히트 칠 것 같네요.

1. 하우스 피자(House Pizza)
    - 기본 토핑: 검정 올리브, 빨간 양파, 피망(green peppers), 모차렐라 치즈
    - 다음 육류 중 2개 선택
        - 베이컨
        - 페페로니
        - 소시지
2. 고기사랑 피자(Meat Lover's Pizza)
    - 기본 토핑: 베이컨, 페페로니, 햄, 소시지, 체다 치즈
    - 다음 야채 중 1개 선택
        - 검정 올리브
        - 빨간 양파
        - 피망
3. 야채 피자(Veggie Pizza)
    - 기본 토핑: 검정 올리브, 빨간 양파, 피망
    - 다음 치즈 중 2개 선택
        - 모차렐라
        - 체다
        - 페타
4. 자유 영혼 피자(Free Soul Pizza)
    - 기본 토핑: 없음
    - 다음 육류 중 2개 선택
        - 닭고기
        - 베이컨
        - 페페로니
        - 소시지
        - 햄
    - 다음 야채 중 2개 선택
        - 검정 올리브
        - 빨간 양파
        - 피망
    - 다음 치즈 중 1개 선택
        - 모차렐라
        - 체다
        - 페타

저는 온라인으로만 운영하는 피자 레스토랑을 만들 거예요. 가게 임대료가 너무 비싸거든요. 웹사이트에서 주문을 받은 후에 집에서 피자를 굽고, 배달은 '민족의 배달', '저기요' 등의 배달 서비스를 사용하면 될 것 같네요. 흠... 그러려면 우선 웹사이트를 만들어야겠죠? 그래서 외주를 줬답니다. 근데 외주자가 피자 모델링을 다음과 같이 해왔네요.

```java
package com.pocurestaurant.menu;

public enum Topping {
    CHICKEN,
    PEPERONI,
    SAUSAGES,
    HAM,
    BACON,
    BLACK_OLIVES,
    RED_ONIONS,
    GREEN_PEPPERS,
    MOZZARELLA_CHEESE,
    CHEDDAR_CHEESE,
    FETA_CHEESE
}
```

```java
package com.pocurestaurant.menu;

import java.util.ArrayList;

public class HousePizza {
    private static final int PRICE = 20;
    private static final int MAX_MEAT_COUNT = 2;

    private int price = PRICE;
    private ArrayList<Topping> toppings = new ArrayList<>();

    private int meatCount;

    public HousePizza() {
        this.toppings.add(Topping.BLACK_OLIVES);
        this.toppings.add(Topping.RED_ONIONS);
        this.toppings.add(Topping.GREEN_PEPPERS);
        this.toppings.add(Topping.MOZZARELLA_CHEESE);
    }

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

    public boolean isValid() {
        return this.meatCount == MAX_MEAT_COUNT;
    }

    public ArrayList<Topping> getToppings() {
        return this.toppings;
    }

    public boolean addBacon() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.BACON);
        ++this.meatCount;
        return true;
    }

    public boolean removeBacon() {
        boolean isRemoved = this.toppings.remove(Topping.BACON);

        if (isRemoved) {
            --this.meatCount;
        }

        return isRemoved;
    }

    public boolean addPeperoni() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.PEPERONI);
        ++this.meatCount;
        return true;
    }

    public boolean removePeperoni() {
        boolean isRemoved = this.toppings.remove(Topping.PEPERONI);

        if (isRemoved) {
            --this.meatCount;
        }

        return isRemoved;
    }

    public boolean addSausages() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.SAUSAGES);
        ++this.meatCount;
        return true;
    }
```

```
        public boolean removeSausages() {
            boolean isRemoved = this.toppings.remove(Topping.SAUSAGES);

            if (isRemoved) {
                --this.meatCount;
            }

            return isRemoved;
        }
    }
```

```java
package com.pocurestaurant.menu;

import java.util.ArrayList;

public class MeatLoverPizza {
    private static final int PRICE = 21;

    private int price = PRICE;
    private ArrayList<Topping> toppings = new ArrayList<>();
    private boolean isVeggieAdded;

    public MeatLoverPizza() {
        this.toppings.add(Topping.BACON);
        this.toppings.add(Topping.PEPERONI);
        this.toppings.add(Topping.HAM);
        this.toppings.add(Topping.SAUSAGES);
        this.toppings.add(Topping.CHEDDAR_CHEESE);
    }

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

    public boolean isValid() {
        return this.isVeggieAdded;
    }

    public ArrayList<Topping> getToppings() {
        return this.toppings;
    }

    public boolean addBlackOlives() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.BLACK_OLIVES);
        this.isVeggieAdded = true;
        return true;
    }

    public boolean removeBlackOlives() {
        boolean isRemoved = this.toppings.remove(Topping.BLACK_OLIVES);

        if (isRemoved) {
            this.isVeggieAdded = false;
        }

        return isRemoved;
    }

    public boolean addRedOnions() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.RED_ONIONS);
        this.isVeggieAdded = true;
        return true;
    }

    public boolean removeRedOnions() {
        boolean isRemoved = this.toppings.remove(Topping.RED_ONIONS);

        if (isRemoved) {
            this.isVeggieAdded = false;
        }

        return isRemoved;
    }

    public boolean addGreenPeppers() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.GREEN_PEPPERS);
        this.isVeggieAdded = true;
        return true;
    }
```

```
        public boolean removeGreenPeppers() {
            boolean isRemoved = this.toppings.remove(Topping.GREEN_PEPPERS);

            if (isRemoved) {
                this.isVeggieAdded = false;
            }

            return isRemoved;
        }
    }
```

```java
package com.pocurestaurant.menu;

import java.util.ArrayList;

public class VeggiePizza {
    private static final int PRICE = 17;
    private static final int MAX_CHEESE_COUNT = 2;

    private int price = PRICE;
    private ArrayList<Topping> toppings = new ArrayList<>();

    private int cheeseCount;

    public VeggiePizza() {
        this.toppings.add(Topping.BLACK_OLIVES);
        this.toppings.add(Topping.RED_ONIONS);
        this.toppings.add(Topping.GREEN_PEPPERS);
    }

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

    public boolean isValid() {
        return this.cheeseCount == MAX_CHEESE_COUNT;
    }

    public ArrayList<Topping> getToppings() {
        return this.toppings;
    }

    public boolean addMozzarellaCheese() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.MOZZARELLA_CHEESE);
        ++this.cheeseCount;
        return true;
    }

    public boolean removeMozzarellaCheese() {
        boolean isRemoved = this.toppings.remove(Topping.MOZZARELLA_CHEESE);

        if (isRemoved) {
            --this.cheeseCount;
        }

        return isRemoved;
    }

    public boolean addCheddarCheese() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.CHEDDAR_CHEESE);
        ++this.cheeseCount;
        return true;
    }

    public boolean removeCheddarCheese() {
        boolean isRemoved = this.toppings.remove(Topping.CHEDDAR_CHEESE);

        if (isRemoved) {
            --this.cheeseCount;
        }

        return isRemoved;
    }

    public boolean addFetaCheese() {
        if (isValid()) {
            return false;
        }

        this.toppings.add(Topping.FETA_CHEESE);
        ++this.cheeseCount;
        return true;
    }
```

```
        public boolean removeFetaCheese() {
            boolean isRemoved = this.toppings.remove(Topping.FETA_CHEESE);

            if (isRemoved) {
                --this.cheeseCount;
            }

            return isRemoved;
        }
    }
```

```java
package com.pocurestaurant.menu;

import java.util.ArrayList;

public class FreeSoulPizza {
    private static final int PRICE = 25;
    private static final int MAX_MEAT_COUNT = 2;
    private static final int MAX_VEGGIE_COUNT = 2;

    private int price = PRICE;
    private ArrayList<Topping> toppings = new ArrayList<>();

    private int veggieCount;
    private int meatCount;
    private boolean isCheeseAdded;

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

    public boolean isValid() {
        return this.meatCount == MAX_MEAT_COUNT
                && this.veggieCount == MAX_VEGGIE_COUNT
                && this.isCheeseAdded;
    }

    public ArrayList<Topping> getToppings() {
        return this.toppings;
    }

    public boolean addTopping(Topping topping) {
        if ((isMeat(topping) && this.meatCount >= MAX_MEAT_COUNT)
                || (isVeggie(topping) && this.veggieCount >= MAX_VEGGIE_COUNT)
                || (isCheese(topping) && this.isCheeseAdded)) {
            return false;
        }

        this.toppings.add(topping);

        if (isMeat(topping)) {
            ++this.meatCount;
        }

        if (isVeggie(topping)) {
            ++this.veggieCount;
        }

        if (isCheese(topping)) {
            this.isCheeseAdded = true;
        }

        return true;
    }

    public boolean removeTopping(Topping topping) {
        boolean isRemoved = this.toppings.remove(topping);

        if (isRemoved) {
            if (isMeat(topping)) {
                --this.meatCount;
            }

            if (isVeggie(topping)) {
                --this.veggieCount;
            }

            if (isCheese(topping)) {
                this.isCheeseAdded = false;
            }
        }

        return isRemoved;
    }

    private static boolean isMeat(Topping topping) {
        return topping == Topping.BACON
                || topping == Topping.CHICKEN
                || topping == Topping.PEPERONI
                || topping == Topping.SAUSAGES
                || topping == Topping.HAM;
    }
```

```
    private static boolean isVeggie(Topping topping) {
        return topping == Topping.BLACK_OLIVES
                || topping == Topping.RED_ONIONS
                || topping == Topping.GREEN_PEPPERS;
    }

    private static boolean isCheese(Topping topping) {
        return topping == Topping.MOZZARELLA_CHEESE
                || topping == Topping.CHEDDAR_CHEESE
                || topping == Topping.FETA_CHEESE;
    }
}
```

각 클래스 안에서 중복되는 코드가 꽤 많네요. 이럴 땐 당연히 부모 클래스를 만들어서 공통된 로직들을 추상화시켜 한 곳에 넣어야 한다던데... 아무래도 외주자가 OOP를 잘 모르시는 것 같아 그분께 맡길 수는 없고... 도와주세요! 여러분이 저의 유일한 희망이에요!

1. 프로젝트를 준비한다

1. `Lab6` 폴더로 이동합니다.
2. 위에 있는 모든 코드를 `academy.pocu.comp2500.lab6` 패키지 안에 추가합니다.
3. 자, 스트레칭 한 번 하시고... 설계와 프로그래밍 시작하겠습니다. :)

2. 전반적인 규칙

- 다형성을 사용할 수 없습니다.
- 추상 클래스를 사용할 수 없습니다.
- `instanceof` 키워드를 사용할 수 없습니다.
- `getClass()` 메서드를 사용할 수 없습니다.
- 여러분이 작성한 코드는 반드시 `academy.pocu.comp2500.lab6` 패키지 안에 있어야 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.

3. 피자 클래스들의 구조를 바꾼다

- 다음 클래스들에 있는 공통 로직을 부모 클래스(들)를 만들어 추상화하세요.
    - `HousePizza`
    - `MeatLoverPizza`
    - `VeggiePizza`
    - `FreeSoulPizza`
- 각 클래스 안에 현재 있는 기능을 변경하면 안 됩니다.

- 이미 존재하는 메서드 시그내처를 변경할 수 없습니다. 즉, 다음과 같은 코드들이 피자 클래스들의 구조를 바꾸기 전과 후에 모두 작동해야 합니다.

```
MeatLoverPizza meatLoverPizza = new MeatLoverPizza();
boolean isAdded = meatLoverPizza.addGreenPeppers(); // true
boolean isRemoved = meatLoverPizza.removeGreenPeppers(); // true
isAdded = meatLoverPizza.addRedOnions(); // true
boolean isValid = meatLoverPizza.isValid(); // true
int price = meatLoverPizza.getPrice(); // 21
ArrayList<Topping> toppings = meatLoverPizza.getToppings();
```

- 멤버 변수나 메서드는 자유롭게 추가해도 됩니다. 하지만 반드시 `private` 이나 `protected` 여야 합니다.

- 매개변수를 받지 않는 `public` 생성자는 자유로이 추가해도 됩니다.

4. 세트 메뉴 클래스들을 추가한다

온라인 레스토랑을 연지도 어느덧 한 달이 지났네요! 근데 매출이 별로예요. 제 레시피는 완벽하니 그냥 이 동네에는 피자를 좋아하는 사람들이 많이 없는 거겠죠? 그래서 매출을 올리려고 메뉴를 좀 늘려봤어요. 다양한 애피타이저, 메인, 디저트를 추가했답니다.

애피타이저:

1. 오징어 튀김(Calamari)
2. 만두(Gyoza)
3. 시금치 딥(Spinach Dip)
4. 나초(Nachos)

메인:

1. 잠발라야 파스타(Jambalaya Fettuccine)
2. 패쉬 앤 칩스(Fish & Chips)
3. 아히 참치 포케(Ahi Tuna Poke)
4. 봄베이 버터 치킨(Bombay Butter Chicken)
5. 필레미뇽(Filet Mignon)
6. 치킨 데리야끼 덮밥(Teriyaki Chicken Rice Bowl)

디저트:

1. 망고 푸딩
2. 녹차 아이스크림
3. 초콜릿 무스
4. 사과 파이
5. 이탈리아식 도넛

그리고 다음과 같은 세트 메뉴(combo meal plan)를 기획했습니다.

1. 3 코스(Three Course Meal)
    - 애피타이저, 메인, 디저트를 1개씩 선택
2. 가벼운 식사(No Heavy Meal)
    - 애피타이저 2개, 디저트 1개를 선택
3. 디저트 먹고 죽자(Death by Desserts)
    - 디저트만 4개 선택

웹사이트에 있는 메뉴에 위 3가지 세트 메뉴를 추가하고 싶어서 다른 외주자를 구했어요. 이 사람은 좀 더 OOP 설계를 잘한다고 했는데... 이런 결과물을 가져왔네요.

```
package com.pocurestaurant.menu;

public enum Appetizer {
    CALAMARI,
    GYOZA,
    SPINACH_DIP,
    NACHOS
}
```

```
package com.pocurestaurant.menu;

public enum MainCourse {
    JAMBALAYA_FETTUCCINE,
    FISH_AND_CHIPS,
    AHI_TUNA_POKE,
    BOMBAY_BUTTER_CHICKEN,
    FILET_MIGNON,
    TERIYAKI_CHICKEN_RICE_BOWL
}
```

```
package com.pocurestaurant.menu;

public enum Dessert {
    MANGO_PUDDING,
    GREEN_TEA_ICE_CREAM,
    CHOCOLATE_MOUSSE,
    APPLE_PIE,
    ITALIAN_DONUTS
}
```

```java
package com.pocurestaurant.menu;

import java.util.ArrayList;

public class NoHeavyMeal {
    private static final int PRICE = 15;

    private int price = PRICE;
    private ArrayList<Appetizer> appetizers = new ArrayList<>();
    private Dessert dessert;

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

    public boolean isValid() {
        return this.appetizers.size() == 2 && this.dessert != null;
    }

    public ArrayList<Appetizer> getAppetizers() {
        return this.appetizers;
    }

    public Dessert getDessert() {
        assert (this.dessert != null) : "call isValid() first!";
        return this.dessert;
    }

    public void setAppetizers(Appetizer appetizer1, Appetizer appetizer2) {
        this.appetizers.clear();

        this.appetizers.add(appetizer1);
        this.appetizers.add(appetizer2);
    }

    public void setDessert(Dessert dessert) {
        this.dessert = dessert;
    }
}
```

```java
package com.pocurestaurant.menu;

public class ThreeCourseMeal {
    private static final int PRICE = 25;

    private int price = PRICE;

    private Appetizer appetizer;
    private MainCourse mainCourse;
    private Dessert dessert;

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

    public boolean isValid() {
        return this.appetizer != null && this.mainCourse != null && this.dessert != null;
    }

    public Appetizer getAppetizer() {
        assert (this.appetizer != null) : "call isValid() first!";
        return this.appetizer;
    }

    public MainCourse getMainCourse() {
        assert (this.mainCourse != null) : "call isValid() first!";
        return this.mainCourse;
    }

    public Dessert getDessert() {
        assert (this.dessert != null) : "call isValid() first!";
        return this.dessert;
    }

    public void setMainCourse(MainCourse mainCourse) {
        this.mainCourse = mainCourse;
    }

    public void setAppetizer(Appetizer appetizer) {
        this.appetizer = appetizer;
    }

    public void setDessert(Dessert dessert) {
        this.dessert = dessert;
    }
}
```

```
package com.pocurestaurant.menu;

import java.util.ArrayList;

public class DeathByDesserts {
    private static final int PRICE = 20;

    private int price = PRICE;
    private boolean isValid;
    private ArrayList<Dessert> desserts = new ArrayList<>();

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

    public boolean isValid() {
        return this.isValid;
    }

    public ArrayList<Dessert> getDesserts() {
        return this.desserts;
    }

    public void setDesserts(Dessert dessert1, Dessert dessert2, Dessert dessert3, Dessert dessert4) {
        this.desserts.clear();
        this.desserts.add(dessert1);
        this.desserts.add(dessert2);
        this.desserts.add(dessert3);
        this.desserts.add(dessert4);
        this.isValid = true;
    }
}
```

여기서도 3개의 세트 메뉴가 서로 공유할 수 있는 공통 로직이 보이네요. 심지어는 앞서 만든 피자 클래스와도 공유 가능한 로직이 보이는데요? 애피타이저, 메인, 디저트의 게터들도 일반화할 수 있을 것 같아요. 왜 이리 사람들은 OOP를 제대로 하지 않는 걸까요? 믿을 사람은 역시 여러분 밖에 없네요. 한 번만 더 도와주세요. 제대로 추상화된 구조로 바꿔주실 수 있죠?

4.1 피자와 세트 메뉴 클래스들의 구조를 바꾼다

- 다음 클래스들에 있는 공통 로직을 부모 클래스(들)를 만들어 추상화하세요.
    - `DeathByDesserts`
    - `NoHeavyMeal`
    - `ThreeCourseMeal`
- 추상화를 더 해서 섹션 3에서 작성했던 코드와 세트 메뉴 클래스들 간의 공통 로직을 분리시키는 것도 가능합니다.
- 다음 목록에서 각 항목을 일반화하세요.
    - 애피타이저의 getter
    - 메인의 getter
    - 디저트의 getter
- 각 클래스의 현재 기능을 변경할 수 없습니다.
- 기존 메서드 시그니처를 변경할 수 없습니다. (단, 일반화시키려 하는 게터들은 예외) 즉, 다음과 같은 코드들이 클래스들의 구조를 바꾸기 전과 후에 모두 작동해야 합니다.

```
DeathByDesserts deathByDesserts = new DeathByDesserts();
deathByDesserts.setDesserts(Dessert.GREEN_TEA_ICE_CREAM, Dessert.ITALIAN_DONUTS, Dessert.MANGO_PUDDING, I
deathByDesserts.isValid(); // true
```

- 필요에 따라 클래스에 멤버 변수나 메서드를 추가해도 됩니다. 그러나 이들은 반드시 `private` 또는 `protected` 여야 합니다. 이 규칙에 대한 유일한 예외는 애피타이저, 메인, 디저트 용으로 `public` getter 메서드를 추가해야 할 때입니다. 이때는 `public` 메서드를 사용해도 됩니다.
- 매개변수를 받지 않는 `public` 생성자는 자유로이 추가해도 됩니다.

5. 본인 컴퓨터에서 테스트하는 법

- `Program.java` 를 아래처럼 바꾼 뒤 실행하세요.

```
package academy.pocu.comp2500.lab6.app;

import academy.pocu.comp2500.lab6.Appetizer;
import academy.pocu.comp2500.lab6.DeathByDesserts;
import academy.pocu.comp2500.lab6.Dessert;
import academy.pocu.comp2500.lab6.FreeSoulPizza;
import academy.pocu.comp2500.lab6.HousePizza;
import academy.pocu.comp2500.lab6.MainCourse;
import academy.pocu.comp2500.lab6.MeatLoverPizza;
import academy.pocu.comp2500.lab6.NoHeavyMeal;
import academy.pocu.comp2500.lab6.ThreeCourseMeal;
import academy.pocu.comp2500.lab6.Topping;
import academy.pocu.comp2500.lab6.VeggiePizza;

public class Program {

    public static void main(String[] args) {
        {
            NoHeavyMeal noHeavyMeal = new NoHeavyMeal();

            assert (!noHeavyMeal.isValid());

            noHeavyMeal.setAppetizers(Appetizer.CALAMARI, Appetizer.GYOZA);

            assert (!noHeavyMeal.isValid());

            noHeavyMeal.setDessert(Dessert.APPLE_PIE);

            assert (noHeavyMeal.isValid());
        }

        {
            DeathByDesserts deathByDesserts = new DeathByDesserts();

            assert (!deathByDesserts.isValid());

            deathByDesserts.setDesserts(Dessert.GREEN_TEA_ICE_CREAM, Dessert.ITALIAN_DONUTS, Dessert.MANGO_PU

            assert (deathByDesserts.isValid());
        }

        {
            ThreeCourseMeal threeCourseMeal = new ThreeCourseMeal();

            assert (!threeCourseMeal.isValid());

            threeCourseMeal.setAppetizer(Appetizer.GYOZA);
            threeCourseMeal.setDessert(Dessert.CHOCOLATE_MOUSSE);

            assert (!threeCourseMeal.isValid());

            threeCourseMeal.setMainCourse(MainCourse.BOMBAY_BUTTER_CHICKEN);

            assert (threeCourseMeal.isValid());
        }

        {
            HousePizza housePizza = new HousePizza();

            assert (!housePizza.isValid());

            assert (housePizza.addBacon());

            assert (!housePizza.isValid());

            assert (housePizza.addPeperoni());

            assert (housePizza.isValid());

            assert (!housePizza.addSausages());

            assert (housePizza.isValid());

            assert (!housePizza.removeSausages());

            assert (housePizza.removeBacon());

            assert (!housePizza.isValid());

            assert (housePizza.addPeperoni());
```

```
        assert (housePizza.isValid());
    }

    {
        MeatLoverPizza meatLoverPizza = new MeatLoverPizza();

        assert (!meatLoverPizza.isValid());

        assert (meatLoverPizza.addGreenPeppers());
        assert (!meatLoverPizza.addGreenPeppers());

        assert (meatLoverPizza.isValid());

        assert (meatLoverPizza.removeGreenPeppers());

        assert (!meatLoverPizza.isValid());
    }

    {
        VeggiePizza veggiePizza = new VeggiePizza();

        assert (!veggiePizza.isValid());

        assert (veggiePizza.addCheddarCheese());

        assert (!veggiePizza.isValid());

        assert (veggiePizza.addFetaCheese());

        assert (veggiePizza.isValid());

        assert (veggiePizza.removeCheddarCheese());

        assert (!veggiePizza.isValid());

        assert (veggiePizza.addMozzarellaCheese());

        assert (veggiePizza.isValid());
    }

    {
        FreeSoulPizza freeSoulPizza = new FreeSoulPizza();

        assert (!freeSoulPizza.isValid());

        assert (freeSoulPizza.addTopping(Topping.FETA_CHEESE));
        assert (!freeSoulPizza.addTopping(Topping.CHEDDAR_CHEESE));

        assert (freeSoulPizza.addTopping(Topping.BLACK_OLIVES));
        assert (freeSoulPizza.addTopping(Topping.RED_ONIONS));

        assert (!freeSoulPizza.isValid());

        assert (!freeSoulPizza.addTopping(Topping.GREEN_PEPPERS));
        assert (!freeSoulPizza.removeTopping(Topping.GREEN_PEPPERS));

        assert (!freeSoulPizza.isValid());

        assert (freeSoulPizza.addTopping(Topping.CHICKEN));
        assert (freeSoulPizza.addTopping(Topping.SAUSAGES));

        assert (freeSoulPizza.isValid());

        assert (!freeSoulPizza.addTopping(Topping.PEPERONI));
    }
}
```

6. 커밋, 푸시 그리고 빌드 요청

이건 어떻게 하는지 이제 다 아시죠? :)

POCU Copyright © 2018 - 2022. POCU Labs Inc.
문의하기 개인정보처리방침 이용 약관 POCU 소개 로드맵 굿즈샵