**소프트웨어 설계 패턴 (실습 9)**

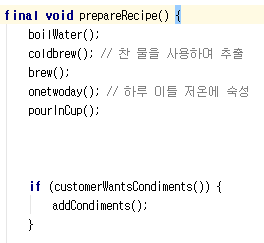
**1494067 이찬희**

이번 실습에서는 예제코드로 주어진barista(탬플릿 메소드 패턴 예)의 CaffeineBeverageWithHook을 바탕으로 진행합니다.

* 새로운 음료인 더치 커피를 추가하려고 합니다. 더치 커피는 일반 커피와는 달리 만드는 방법이 이래와 같이 다소 상이 합니다
  + 찬 물을 사용하여 장시간에 걸쳐 커피액을 추출(coldbrew())
  + 커피 추출 후 하루이틀 저온에서 숙성.(onetwoday())
  + 음료를 컵에 따른다(pourInCup)
  + 첨가물을 추가한다. (선택적)

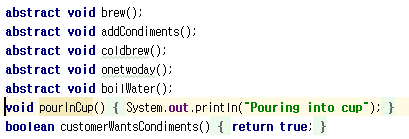
1. Coffee, Tea 뿐만 아니라 위의 더치 커피를 만들기 위해서 CaffeineBeverageWithHook 클래스의 템플릿 메소드를 수정하고, 필요하면 훅 메소드를 추가하시오. 코드에 대한 설명도 포함해야 합니다.

->첫번째로 prepareRecipe() 메소드의 실행 순서를 변경하였다.



두번째로 기존에 선언되있던 boilWater() 메소드를

추상 메소드로 선언하고 각각의 서브 클래스에서 재정의 하게 하였다.



기존의 코드와 같이 void boilWater() { System.out.println(“ ……” )}; 이런 식으로 코딩이 되어 있으면 이 클래스(CaffeineBeverageWithHook)을 상속받는

새로 만든 클래스 DutchCoffeeWithHook은 prepareReceipe() 메소드가 실행될 때 boilWater() 메소드가 실행이 되버린다. 하지만 DutchCoffeeWithHook 클래스는 boilWater()와 같이 물을 끓이는 행동이 필요하지 않다.

공통되지 않은 메소드가 슈퍼 클래스에서 정의되어 있기 때문에 서브 클래스인 DutchCoffeeWithHook 클래스에서 필요없는 메소드가 실행되는 문제가 생기기 때문에, 미리 구현되 있는 코드를 추상 클래스로 선언하도록 수정하고 각 서브 클래스에서 재정의 하도록 하였다.

pourInCup() 메소드는 공통되기 때문에 남겨두었다.

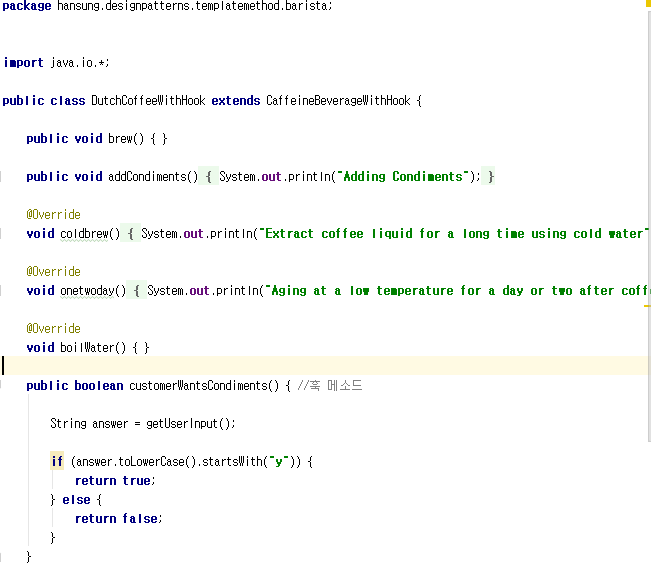
1. 수정된 CaffeineBeverageWithHook 클래스를 바탕으로, Coffee, Tea 클래스를 수정하고, DutchCoffee 클래스를 추가하시오. 코드에 대한 설명도 포함해야 합니다.

->새롭게 추가된 추상 메소드는 coldbrew(), onetwoday()이다.

이를 3개의 서브 클래스인

CoffeeWithHook, DutchCoffeeWithHook, TeaWithHook 클래스에 각각 재정의 하였다.

첫번째로 DutchCoffeeWithHook 클래스이다.



brew() 메소드와 boilWater() 메소드는 이 클래스에서는 필요하지 않으므로 오버라이딩을 하였지만 안에 아무것도 구현하지 않았다.

coldbrew() 메소드가 찬 물을 사용해 장기간에 걸쳐 추출이고

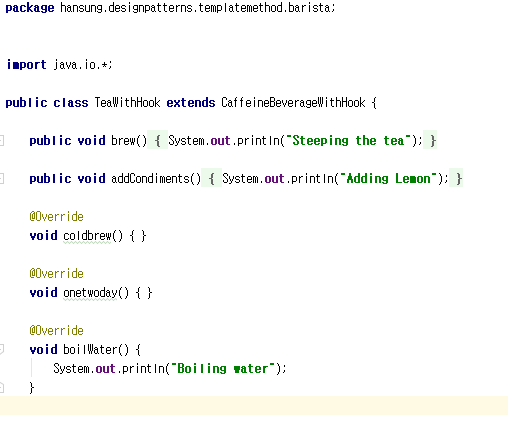
onetwoday() 메소드가 추출 후 하루이틀 숙성하는 메소드이다.

두번째로는 CoffeeWithHook 클래스이다.



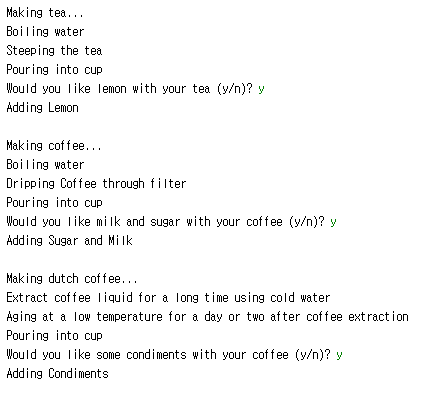
이 클래스에서는 coldbrew() 메소드와 onetwoday() 메소드가 필요하지 않으므로 안에 아무것도 구현하지 않았다.

세번쨰로 TeaWithHook 클래스이다.



마찬가지로 coldbrew() 메소드와 onetwoday() 메소드 안을 비워 놨다.

결과 화면



->더치 커피 만들 때 차가운 물로 추출과, 하루이틀 숙성시키는 것을 출력함

1. 위의 실습 내용을 바탕으로 템플릿 패턴의 장단점을 토의해 봅시다.

->장점으로는 공통된 행동들이 있을 때 그것을 슈퍼 클래서에서 정의함으로써 서브 클래스에서 생성되는 코드의 복잡성을 줄일 수 있다.

하지만 실습과제처럼 공통되지 않은 행동을 가진 클래스들이 지속적으로 추가된다면 결국 이 패턴 자체의 의미가 애매해진다고 생각한다.

공통되는 것을 슈퍼 클래스에서 구현을 해 놨는데 서브 클래스에서 그 공통행동을 필요로 하지 않는다면 결국 공통 메소드가 추상 메소드로 선언되어야 할 것이다.

공통되지 않은 메소드들을 추상메소드로 선언하고 각 클래스에서 필요로 하지 않는 메소드 안은 비워 놓는 방식으로 구현하였는데, 이 방법 말고 다른 방식으로 구현할 수 있다면 효율적인 코딩이 가능할 것이다.