프로젝트 최종 보고서

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. 프로그램 개요   * 커피숍 매니지먼트 프로그램 * 점원 기능 * (커피 주문, 커피 제공, 주문 취소) * 점장 기능 * (커피 재료 발주, 잔고 확인, 주문 내역 확인)   .  2. 전체 논리 구조도 및 모듈 별 기능   1. App    * 1. 점원 메뉴 출력      2. 점장 메뉴 출력 2. CounterFacade    * 1. 주문 받기: 커피의 아이디, 샷 추가, 휘핑크림 추가, 할인 방법 선택      2. 커피 제공: 주문한 커피를 제공      3. 주문 취소: 주문 목록에 있는 커피를 삭제 3. ManagerFacade    * 1. 재료 발주: 커피, 우유, 모카, 샷, 휘핑크림의 재료를 추가      2. 재고 확인: 현재 남은 재고를 확인      3. 주문 내역 확인: 현재 주문된 음료 목록을 확인      4. 잔여 자본금 확인 4. DiscountAlgorithm    * 1. 할인된 음료의 가격을 계산 5. Coffee    * 1. 고객에게 제공될 음료 객체 6. IngredientDecorator    * 1. 기타 부가재료를 데코레이터 패턴으로 관리 7. Transaction    * 1. 주문 내역 8. ProductPriceCalculator    * 1. 최종 가격 계산 9. CoffeeFactory    * 1. 팩토리 패턴으로 음료를 생성 10. OrderTransactionManager     * 1. 주문 내역 관리 11. StockManager     * 1. 재고 관리 12. 프로세스 처리도 & 시나리오 흐름도     5. 모듈의 함수     |  | | --- | | **App.h** | | #pragma once  #include "CounterFacade.h"  #include "ManagerFacade.h"  class App  {  public:  App();  ~App();  void run();  private:  CounterFacade\* cf; // 점원 Facade  ManagerFacade\* mf; // 점장 Facade  }; |  |  |  | | --- | --- | | **CounterFacade.h** | | | #pragma once  #include "CoffeeFactory.h"  #include "OrderTransactionManager.h"  #include "StockManager.h"  #include "SaleManager.h"  #include "ProductPriceCalculator.h"  class CounterFacade  {  public:  CounterFacade();  CounterFacade(OrderTransactionManager\* \_otm, StockManager\* \_sm, SaleManager\* \_slm);  ~CounterFacade();  void order();  void serveCoffee();  void cancle();  private:  CoffeeFactory cf; //Coffee 팩토리  ProductPriceCalculator ppc; // 가격 계산  OrderTransactionManager\* otm; // 주문 관리  StockManager\* sm; // 재고 관리  SaleManager\* slm; // 가격 관리  }; |  |  | | --- | | **ManagerFacade.h** | | #pragma once  #include "StockManager.h"  #include "OrderTransactionManager.h"  #include "SaleManager.h"  class ManagerFacade  {  public:  ManagerFacade();  ManagerFacade(OrderTransactionManager \* \_otm, StockManager\* \_sm, SaleManager\* \_slm);  ~ManagerFacade();  void orderIngredients();  void printStockStatus();  void printTransactionStatus();  private:  OrderTransactionManager \* otm; // 주문 관리  StockManager\* sm; // 재고 관리  SaleManager\* slm; // 가격 관리  }; |  |  | | --- | | **CoffeeFactory.h** | | #pragma once  //#include "Coffee.h"  #include "BaseCoffee.h"  #include "MilkDecorator.h"  #include "MochaDecorator.h"  #include "ShotDecorator.h"  #include "WhipDecorator.h"  class CoffeeFactory  {  public:  CoffeeFactory();  ~CoffeeFactory();  Coffee\* makeCoffee(int coffeeID); // 커피 생성  }; |  |  | | --- | | **IngredientsDecorator.h** | |  | | #pragma once  #include "Coffee.h"  class IngredientsDecorator :  public Coffee  {  public:  IngredientsDecorator();  IngredientsDecorator(Coffee\* coffee);  ~IngredientsDecorator();  protected:  Coffee\* pCoffee;  }; | |  |  |  | | --- | | **OrderTransactionManager.h** | | #pragma once  #include "Transaction.h"  #include <vector>  #include <iostream>  using namespace std;  class OrderTransactionManager  {  public:  OrderTransactionManager();  ~OrderTransactionManager();  void addTransaction(Transaction t);  Transaction getFrontTransaction();  void deleteTransactionFromFront();  void deleteTransactionByIndex(int idx);  void printTransactions();  private:  vector<Transaction> orderTransactions; // 주문 순서  }; |   6. 설계 원칙  설계 요구 조건에 따라 충실히 설계한다.  - 커피숍 매니지먼트 프로그램을 구현하기 위한 객체는 App, CounterFacade, ManagerFacade, ProductPriceCalculator, DiscountAlgorithm, CoffeeFactory, Coffee, OrderTransactionManager, Transaction, StockManager, SaleManager 이다.  - DiscountAlgorithm은 추상 클래스로써, DiscountDrinkInStore, DiscountTakeOut, DiscountCoupon 클래스의 부모 클래스이다. 순수가상함수인 discount()를 가지고 있다.  - DiscountDrinkInStore, DiscountTakeOut, DiscountCoupon은 discount() 함수를 오버라이딩하여 매장에서 마시는 경우, 테이크아웃 하는 경우, 쿠폰을 사용하는 경우에 대한 할인된 음료의 가격을 계산한다.  - Coffee는 추상 클래스로써, BaseCoffee, IngerdientDecorator의 부모 클래스이다. 순수가상함수인 getPrice()와 getIngredientsID() 함수를 가지고 있다.  - IngredientDecorator는 Coffee 클래스에 데코레이터 패턴들 적용하기 위한 MilkDecorator, MochaDecorator, ShotDecorator, WhipDecorator의 부모클래스이다.  - MilkDecorator, MochaDecorator, ShotDecorator, WhipDecorator 클래스는 getPrice()와 getIngredientsID() 함수를 오버라이딩하여 자신의 가격 및 아이디를 반환한다.  - Transaction 클래스는 고객이 주문한 내역을 저장하기 위한 클래스로써, 음료의 아이디와 추가 재료들의 아이디를 저장하는 int형 vector 변수와, 음료의 가격을 저장하는 정수형 변수를 가지고 있다.  - ProductPriceCalculator는 DiscountAlgorithm 클래스를 활용하여 음료의 할인된 최종 가격을 계산한다..  - CoffeeFactory 클래스는 음료의 아이디를 입력 받아 아이디에 해당되는 음료를 생성하여 반환한다.  - OrderTransactionManager는 고객이 주문한 음료들의 내역을 먼저 주문한 고객이 먼저 받도록 관리한다.  - StockManager는 현재 커피숍이 가지고 있는 재료들의 재고를 관리한다.  - CounterFacade는 점원이 사용하는 기능을 구현한 클래스로써, 주문 받기, 커피 제공, 주문 취소 기능을 가지고 있다.  - ManagerFacade는 점장이 사용하는 기능을 구현한 클래스로써, 재료 발주, 재고 확인, 주문 내역 확인 기능을 가지고 있다.  - 메인에서는 오직 App의 run()만을 호출한다.  7. 개발도구  Microsoft Visual Studio 2013 C++  8. 사용 매뉴얼     * 메뉴 선택       점원 메뉴에서 주문 받기를 선택함으로써 메뉴 선택 및 할인 방법 선택으로 음료 주문이  가능  아메리카노(커피), 카푸치노(커피 + 우유), 모카커피(커피 + 모카)  주문 받기를 선택하면 인덱스로 기본 커피 선택, 추가 메뉴(샷, 휘핑크림)를 선택 가능   * 할인 방법 선택     메뉴 선택을 끝내면 할인 방법을 선택  매장에서 마시고 가면 할인율 0%, 테이크 아웃은 10%, 쿠폰은 100%를 적용  주문이 끝나면 주문 내역에 추가   * 주문 내역     주문 내역에서는 주문한 커피가 큐(Queue)로 저장됨   * 커피 제공       먼저 주문된 순서대로 큐 방식을 이용해 주문된 커피를 제공함   * 주문 취소     아직 제공되지 않은 주문 내역에서 인덱스를 선택하여 주문을 취소함  선택된 인덱스 번째의 주문 내역으로 접근하여그 주문 내역에서 사용된 재료들을 재고로 되돌려놓고  결제된 돈을 감소시킨 후 주문 내역에서 삭제  주문 취소가 성공하면 이를 알리고 새로 갱신된 주문 내역을 출력   * 재료 발주       현재 남아있는 재고를 출력 후 발주할 재료를 사용자로부터 입력 받음  자본금 초과 시 구매 불가능  발주에 성공하면 금액을 출력하고 자본금을 감소시킴   * 잔여금 확인 / 재고 확인 / 주문내역 확인         점장 메뉴에서 확인 가능하도록 구현 |