**디자인패턴 실습 6**

|  |  |  |  |
| --- | --- | --- | --- |
| **학번** | B889047 | **이름** | 윤준호 |

1. **스트래티지 패턴을 사용해 다음 요구사항을 만족하는 코드를 작성하고, 클래스 다이어그램과 시퀀스 다이어그램을 그리시오.**

* 회원은 이름과 누적 대여 금액을 갖는다.
* 책은 서명, 출판년도, 가격을 갖는다(재고는 무한대라고 가정).
* 회원은 하나의 책을 1권 이상 살 수 있다.
* 회원이 책을 살 때마다 누적 금액이 저장된다.
* 가격 정책에 따라 책 값이 할인되며, 다른 가격 정책이 추가될 수 있다.
* 10년 이상 된 책은 책 자체 할인, 누적 대여 금액이 만 원 이상이면 회원 할인, 그 외의 경우에는 할인이 되지 않는다.

1. 클래스 다이어그램

|  |
| --- |
|  |

1. 시퀀스 다이어그램(객체명, 메서드명, 메서드 호출 흐름만 기술) (변수X, 매개변수X)

|  |
| --- |
|  |

1. **문제1의 다이어그램에 다음의 요구사항을 추가하여 재설계하시오. (OCP를 준수해야 함)**

* 회원은 일반 회원과 VIP 회원으로 나뉜다.
* VIP 회원은 책 값이 추가적으로 5%할인 된다.
* 회원 종류는 추가될 수 있다.

1. 클래스 다이어그램

|  |
| --- |
|  |

1. 소스코드

|  |
| --- |
| /\*  \*  \* Source Code Written by B889047윤준호  \* \*/  import java.util.\*;  //Strategy : ln[55]   // Saleable 인터페이스 interface SaleablePolicy{  public abstract void sale(MemberType member,Book book); }  /\* \* 1번 코드를 기반으로 각 할인 정책에 따른 할인률에 멤버 타입에 따른 할인을 더한 다음에 할인률을 계산하는 방식으로 변경한다. \* \*/   // 10년이 넘은 책에 대한 할인 class BookOverTenYearSale implements SaleablePolicy{  private final float DISCOUNT\_RATE = 10;  private final int year;  BookOverTenYearSale(){  Calendar cal = Calendar.*getInstance*();  this.year = cal.get(Calendar.*YEAR*);  }  @Override  public void sale(MemberType member,Book book) {  if(this.year - book.getPublishedDate() > 10){  float discount\_rate = (float)(1-((DISCOUNT\_RATE + member.getAdditionalDiscount()) / 100)); // Changed  int discounted\_price = (int) (book.getPrice() \* discount\_rate);  int price = member.getAccumulatedPrice() + discounted\_price;  member.setAccumulatedPrice(price);  }else{  int price = member.getAccumulatedPrice() + book.getPrice();  member.setAccumulatedPrice(price);  }  } } // 만원이 넘은 회원에 대해 class AccumulateTenThousandSale implements SaleablePolicy{  private final float DISCOUNT\_RATE = 5;  @Override  public void sale(MemberType member, Book book) {  if(member.getAccumulatedPrice() > 10000){  float discount\_rate = (float)(1-((DISCOUNT\_RATE + member.getAdditionalDiscount())/ 100)); // Changed  int discounted\_price = (int) (book.getPrice() \* discount\_rate);  member.setAccumulatedPrice(member.getAccumulatedPrice() + discounted\_price);  }else{  member.setAccumulatedPrice(member.getAccumulatedPrice() + book.getPrice());  }  } }   // 회원종류에 따른 추상클래스 abstract class MemberType{  private final float additionalDiscount;  private int accumulatedPrice;  private String name;   private Map<String,Integer> buyBookList = new HashMap<>();  MemberType(int additionalDiscount, int accumulatedPrice, String name){  this.additionalDiscount = additionalDiscount;  this.accumulatedPrice = accumulatedPrice;  this.name = name;  }   public void addBuyBook(Book book){  if(buyBookList.containsKey(book.getSignature())){  buyBookList.replace(book.getSignature(),(Integer)(buyBookList.get(book.getSignature())) + 1);  }  else{  buyBookList.put(book.getSignature(),1);  }  /\* For test  System.out.println("---------");  for(Map.Entry<String,Integer> entry : buyBookList.entrySet()){  System.out.println(entry.getKey() + " " + entry.getValue());  }  System.out.println("---------");  \*/  }    public float getAdditionalDiscount(){  return this.additionalDiscount;  }   public String getName() {  return name;  }   public void setName(String name) {  this.name = name;  }   public int getAccumulatedPrice() {  return accumulatedPrice;  }   public void setAccumulatedPrice(int accumulatedPrice) {  this.accumulatedPrice = accumulatedPrice;  } }    // 회원클래스 class NormalMember extends MemberType{   NormalMember(String name){  super(0,0,name);  }   @Override  public void addBuyBook(Book book) {  super.addBuyBook(book);  } }  class VIPMember extends MemberType{  VIPMember(String name){  super(5,0,name);  }   @Override  public void addBuyBook(Book book) {  super.addBuyBook(book);  } }   // 책 구매 클래스 class BookBuyer{  private SaleablePolicy policy;   public void setPolicy(SaleablePolicy policy){  this.policy = policy;  }  public BookBuyer(SaleablePolicy policy){  this.policy = policy;  }   public void buyBook(MemberType member, Book book){  policy.sale(member,book); // Strategy  member.addBuyBook(book);  } }  // 책 클래스 class Book{  private int price;  private String signature;  private int publishedDate;   Book(int price, String signature,int publishedDate){  this.price = price;  this.signature = signature;  this.publishedDate = publishedDate;  }   public String getSignature() {  return signature;  }   public void setSignature(String signature) {  this.signature = signature;  }  public int getPublishedDate() {  return publishedDate;  }   public void setPublishedDate(int publishedDate) {  this.publishedDate = publishedDate;  }  public int getPrice() {  return price;  }   public void setPrice(int price) {  this.price = price;  } }   class main\_t{  public static void main(String[] args){  MemberType m1 = new VIPMember("A");  MemberType m2 = new NormalMember("B");   Book b1 = new Book(5000,"Java",2000);  Book b2 = new Book(10000, "Python",2021);   BookBuyer bb = new BookBuyer(new AccumulateTenThousandSale()); // Basic Poilicy : 5% discount if Accumulate Ten Thousand won   bb.buyBook(m1,b1);  bb.buyBook(m1,b1);  bb.buyBook(m1,b2);  bb.setPolicy(new BookOverTenYearSale()); // Change Policy : 10% discount if book publised more than 10 year  bb.buyBook(m1,b1);  System.*out*.println(m1.getAccumulatedPrice()); //24250    bb.buyBook(m2,b1);  bb.buyBook(m2,b1);  bb.buyBook(m2,b1);  bb.buyBook(m2,b1);  bb.buyBook(m2,b1);  System.*out*.println(m2.getAccumulatedPrice()); //22500  } } |