실습 1:

-실습코드

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

-결과출력

텍스트, 폰트, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

-코드관련 설명

# BankTransferPayment 클래스를 추가하여, pay 메서드를 포함하여 BankTransferPayment 클래스를 작성했습니다.

# PaymentProcessor 클래스 수정합니다. BankTransferPayment 필드를 추가하고, 생성자에서 이를 초기화했습니다. 또한 processPayment 메서드에 "BankTransfer" 옵션을 추가하여 BankTransfer 결제 방식을 처리할 수 있도록 했습니다.

# main 메서드에서 processPayment 메서드를 사용하여 "BankTransfer" 방식으로 결제를 테스트했습니다.

실습 2:

-실습 코드

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

-결과 출력

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

-코드관련 설명

# PaymentMethod 추상 클래스는 모든 결제 방식 클래스가 구현해야 하는 pay 메서드를 정의한 추상 클래스입니다.

# 개별 결제 방식 클래스인 CreditCardPayment, PayPalPayment, BankTransferPayment는 모두 PaymentMethod 클래스를 상속받아 pay 메서드를 오버라이딩하여 각 결제 방식에 맞는 로직을 구현합니다.

# PaymentProcessor 클래스는 PaymentMethod 타입을 생성자로 받아오며, processPayment 메서드를 통해 결제를 수행합니다. PaymentProcessor는 특정 결제 방식에 의존하지 않으므로 결제 방식이 추가되어도 코드 수정이 필요하지 않습니다.

#SOLID 원칙:

1. Open-Closed Principle): 코드가 확장에는 열려 있고 수정에는 닫혀 있습니다. 새로운 결제 방식이 추가될 때 PaymentProcessor 코드를 수정할 필요 없이 새로운 결제 클래스를 작성해 추가할 수 있습니다.
2. Dependency Inversion Principle : PaymentProcessor는 구체적인 결제 방식이 아닌 추상화된 PaymentMethod에 의존하여 유연성이 높아졌습니다.

이렇게 수정하면, 새로운 결제 방식이 추가될 때도 PaymentProcessor를 수정하지 않아도 되기 때문에 코드의 유연성과 유지보수성이 높아집니다.