여러분. 편의점에 있는 상품을 집어 올리는 행위 만으로(집어올리는 제스쳐) 해당 상품을 구매했다고 말할 수 있을까요? 아니죠. 상품을 pos기 앞으로 가져가서 바코드를 찍고 금액을 지불한 이후에야 우리는 비로소 상품을 구매했다고 말합니다. 즉, 편의점의 상품을 구매하기 위해서는 반드시 pos기를 통한 일련의 과정을 거쳐야 한다는 건데요. 지금부터 저희가 만든 pos기가 어떻게 이러한 과정을 구현하는지 살펴보겠습니다.

목차는 다음과 같습니다. //

"편의점 pos"라는 기계의 근간을 이루고 있는 컨셉은 바로 "손님과 편의점의 상호작용"을 가능하게 하는 것입니다. 저희는 손님과 편의점, 너무나도 다른 두 객체가 이루는 상호작용에 주목하여 //

총 4가지 카테고리로 pos기의 기능을 재구성했습니다. 저희의 편의점 pos기는 "상품판매 / 상품환불 / 재고관리 / 회계관리" 크게 총 4가지 기능을 수행합니다. 이 4가지 기능들은 "상품"을 통해 서로 소통하며, 각 기능의 연쇄적 변동은 편의점 내부 상태에 변화를 줍니다. 손님에 의해 상품이 구매되는 상황을 생각해 봅시다. //

손님의 구매에 의해 편의점 내부 상품이 판매되면 편의점 내부 있던 상품이 손님에게 이동하고, 그에 따라 재고와 회계의 변동이 일어납니다. 환불의 경우도 마찬가지겠죠 //

손님에게 판매되었던 상품이 다시 편의점으로 이동하고, 그에 따라 연쇄적으로 재고와 회계의 변동이 일어납니다. //

재고의 변동 //

회계의 변동 역시 마찬가지로 각 기능의 연쇄적인 변화를 발생시킵니다. //

저희는 이렇게 복잡한 4가지 기능의 상호작용을 어떻게 하면 효과적으로 관리할 수 있을 지에 주목했습니다. 그리고 oop적 컵셉을 최대한 설계에 녹여냈습니다. //

이제 본격적으로 우리 pos기의 내부 설계를 살펴봅시다. //

각각의 기능은 상품이라는 메시지를 주고받으며 상호작용 한다고 설명 드렸습니다. //

따라서 저희는 상품 내부의 설계를 최우선적으로 고려했고 //

이를 위해 "추상화" 라는 개념을 도입했습니다. 편의점의 상품이 필수적으로 가지고 있어야할 요소들에 대해 생각해 봅시다. 대부분 이름, 가격, 수량 등을 먼저 떠올릴 겁니다. 이렇게 모든 상품이 가져야 하는 공통된 속성과 그에 관련된 기능을 하나로 묶어 //

다음과 같이 추상 class를 만들었습니다. 이제 좀 더 구체화 하여 "바나나"와 "종이컵" 이라는 두 가지의 구체적인 상품을 떠올려봅시다 보겠습니다. 두 상품의 가장 두드러지는 차이점이 무엇이라고 생각하시나요? 저희는 이 두 상품을//

유통기한의 유무로 구별했습니다. 편의점 내의 모든 상품을 유통기한이 있는 상품과 없는 상품으로 type을 구분하여 관리하였습니다. //

왜, 하필이면 상품의 구조를 유통기한을 통해 판단 하였을까요? //

편의점에서는 매일매일 폐기 관리가 진행되고 이는 유통기한이라는 정보를 통해 이루어집니다. 따라서 유통기한을 가장 중요한 식별정보라고 판단 했습니다. 또한 휴지나 소주와 같이 유통기한이 없는 상품은 품질 보증 기간이라는 식별자를 설정해 상품의 관리를 진행했습니다. //

상품에 대해 추상화를 적용함으로써 서로 다른 type의 상품들이 한 컨테이너에 담겨 저장될 수 있게 되었습니다. 이는 창고에 상품이 마구잡이로 섞여 있어도 외부에서는 이를 구분 할 필요 없이 다양한 기능을 수행할 수 있음을 의미합니다. 또한 추상클래스의 상속을 받는 또 다른 type의 상품군이 추가되더라도 코드 상의 큰 수정 없이 상품 관리가 가능합니다. 이를 통해 확장에는 열려있고 변경에는 닫혀 있게 함으로써 기존의 코드를 변경하지 않으면서 기능을 추가할 수 있게 하였습니다. 즉, 개방-폐쇄 원칙이 매우 잘 지켜질 수 있음을 알 수 있습니다. //

위에서 문답한 두가지 쟁점을 통해 설계한 상품의 구조와 관리 방법을 통해 우리는 상품을 구매할 수 있습니다. //

상품을 손님에게 건네고, 이에 따라 손님이 가진 상품의 개수와 가격의 변동이 일어납니다. (쉬어주기)//

이제 이전에 설명 드렸던 4가지 커다란 기능들이 어떻게 상품을 관리하는지 살펴보겠습니다. 저희는 각 기능을 총괄하는 management라는 총 4개의 관리자 class를 만들었습니다 각 class에게 판매, 환불, 재고관리, 회계관리의 행동을 분업하여 담당하게 하였습니다. 환불 상황의 발생을 가정하여 management class를 통해 얻을 수 있는 이점에 대해 이해 보겠습니다. //

해당 uml은 환불을 담당하는 purchase management class가 없을 시 접근해야 하는 class들 입니다. 상위의 빨간색과 파란색 management class를 제외한 모든 class에 접근해야함을 알 수 있습니다. 그러나 management class를 둠으로써 //

보시는 것처럼 단 4가지 class를 통해 환불이라는 커다란 기능을 수행할 수 있게 됩니다. //

즉, 빈번한 변동이 발생하는 class를 뒤로 숨기고 pos기 사용자는 management class에만 직접적으로 접근하게 하는 것입니다. 저희는 캡슐화를 통해 이를 구현하였고 단일 책임 원칙이 매우 잘 지켜짐을 알 수 있습니다. //

지금까지 설계적 관점에서 핵심 쟁점 4가지를 문답을 통해 살펴보았고 지금부터는 //

pos기의 사용 흐름을 중심으로 2가지 상황을 가정하여 pos기를 이해해 봅시다. //

첫번째로 손님과 pos기의 상호작용이 발생하는 상황입니다. 손님이 상품을 구매하는 경우 손님의 정보를 입력 받습니다. // 바로 넘김 //

이는 구매기록에 추가되어 영수증 출력 및 환불에 이용되고 //

이에 따른 재고의 변경과 //

포스기의 회계상의 변동이 발생합니다. //

손님이 환불을 하는 경우도 마찬가지로 재고와 자산의 변동이 발생합니다. //

이제 두 번째 상황으로 편의점과 pos기 사이의 상호작용을 살펴 봅시다. 재고 추가시 상품 목록에 이미 존재하는 상품과 존재하지 않는 상품을 구별하여 //

각각 입력 받는 정보를 // 다르게 처리합니다. //

pos기에서 마감 정산을 시행할 경우 회계상의 변화 //

매출기록의 변화 //

폐기처리가 진행 됩니다. 조금 쉬고 넘기기//

<결론> 40p

저희는 추상화, 캡슐화, 상속, 다형성, 오버로딩, UML, 디자인패턴, STL 등 무려 15주차에 걸쳐 수업시간에 배운 OOP의 핵심적인 기능을 이번 프로젝트에 녹여내었습니다. 결과적으로 위에서 설명한 모든 기능을 하나도 빠짐없이 구현하여 총 16개의 소스파일과 15개의 헤더파일을 만들어 냈고 성공적으로 작동하는 것을 확인하였습니다. 시간적 제약으로 인해 에러 코드 생성이나 섬세한 기능을 구현하지 못했다는 약간의 아쉬움이 남긴 하지만 가능하다면 추후에 팀원들과 해당 프로젝트를 발전시켜 더 완성도 있는 편의점 POS기를 만들어보고자 합니다. //

결과 화면에 및 프로젝트파일 대한 설명은 시간관계상 생략하도록 하겠습니다. 최종 제출 보고서에 해당 내용이 상세히 서술되어 있으니 참고해주시면 좋을 것 같습니다. //

이상으로 12팀 발표를 마치겠습니다. 감사합니다. ////