

# 객체 지향 프로그래밍

팀 프로젝트 최종 보고서

## Product Management System

제품 관리 시스템

## 6팀 BugMaker

(전형진, 이현준, 노승준, 응우옌 만 히에우)

객체 지향 프로그래밍의 팀 프로젝트, 'Product Management System' 을 소개하고자 한다.

Product Management System, 일명 'PMS' 또는 '품목 관리 시스템'은, 다양한 속성을 가진 여러 유형의 품목을, 관리자의 입장에서, 보다 수월하게 정보를 관리할 수 있도록 하는 것을 목표로 하는 프로그램이다.

본래 중간 발표 때에는 'Market Management System', 즉 '매장 관리 시스템' 을 구현하려고 했었으나, 주어진 기간에 비해 지나치게 방대하고 복잡한 수준의 주제로 잘못 선정하였다는 것을 깨달았다.

그래서 기존 프로젝트에서 구현하려고 했던 '고객', '프로모션', '통계' 등의 시스템을 제거하되, 본래 프로젝트의 핵심 기능이었던 '품목 관리' 기능에 집중하겠다는 선택을 하게 되었다.

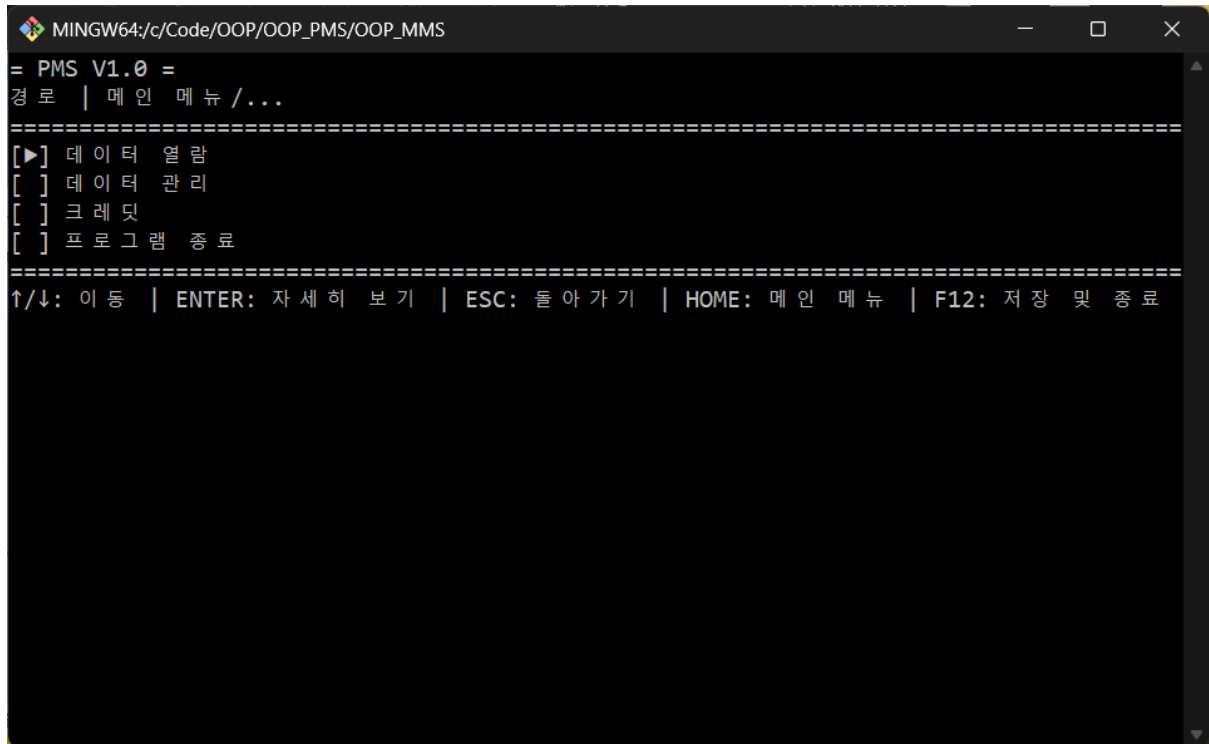
따라서, 최종 프로젝트의 이름은 'Product Management System' 로 변경되었다.

그러나, 그마저도 시간 부족으로 프로그램을 완성할 수 없게 되었다.

부끄러움과 죄책감을 느낀다.

다음 장에서는, 지금까지 구현한 프로그램에 대해 설명하고자 한다.

PMS 프로그램을 실행하면, '메인 메뉴' 화면과 함께 4가지 선택지가 나타나게 된다. '데이터 열람', '데이터 관리', '크레딧', '프로그램 종료' 항목 중에서, 키보드의 상/하 방향 화살표 키를 이용하여 원하는 항목으로 이동한 후, 엔터 키를 눌러 해당 항목을 선택할 수 있다.



```
MINGW64/c/Code/OOP/OOP_PMS/OOP_MMS
= PMS V1.0 =
경로 | 메인 메뉴 /...

=====
[▶] 데이터 열람
[ ] 데이터 관리
[ ] 크레딧
[ ] 프로그램 종료
=====

↑/↓: 이동 | ENTER: 자세히 보기 | ESC: 돌아가기 | HOME: 메인 메뉴 | F12: 저장 및 종료
```

'데이터 열람'에서는 생성된 데이터를 열람할 수 있다.

'데이터 관리'에서는 데이터를 새로 생성하거나, 수정하거나, 삭제할 수 있다.

'크레딧'에서는 프로그램과 제작자 관련 정보를 열람할 수 있다.

'프로그램 종료'를 선택하면 프로그램이 정상적으로 종료된다.

PMS는 6개의 헤더 파일과, main() 함수가 포함된 1개의 cpp 파일로 구성되어 있다. 각 헤더 파일은 조건부 컴파일(ifdef)을 적용하여, 한 번씩만 컴파일되도록 하였다.

각 헤더 파일은 담당하고 있는 역할에 따라 크게 4가지로 구분지을 수 있다.

- 사용자 정의된 날짜 구조체와, 시간 관련 함수를 정의하는 <datetimes.h>
- 재사용할 수 있는 특수 문자 또는 문자열의 집합을 구성하는 <strings\_defined.h>
- 사용자 정의된 품목 또는 카테고리 클래스와, 그에 맞는 데이터 베이스를 구성하는 <productDB.h>, <itemDB.h>, <categoryDB.h>
- 실제 프로그램의 콘솔을 제어하고 감지하는 기능을 구성하는 <console\_controls.h>

이 중 <console\_controls.h> 파일의 역할이 독특한데, 사용자로부터 키 입력을 감지하여 선택, 취소 또는 돌아가기 등의 동작을 수행하거나, 사용자와 상호작용하기 위한 텍스트를 출력한다.

또, 실제 콘솔에 액세스하여 콘솔 화면의 너비를 구한 후, 출력할 수 있는 문자열의 길이를 계산하여 그만큼의 길이를 가진 문자열 구분선을 출력하는 등의, 직관적인 그래픽이나 상호작용 요소를 제어한다.

이것이 독특하다고 평가하는 이유는, 특히 콘솔 화면의 너비를 구하고 텍스트를 출력하는 작업이 빠르게 적용되어 실시간으로 이루어지는 것처럼 보이기 때문이다.

그 덕분에 마우스를 이용하여 콘솔 화면의 너비를 조절하는 순간에도 빠르게 변경 사항이 적용되어, 구분선 텍스트가 다음 줄로 넘어가지 않고 자동으로 딱 콘솔 화면의 너비만큼 맞춰져서 출력된다.

이제는 PMS에서 제공하려 했던 품목의 유형에 대해서 서술하고자 한다.

PMS는 'Product', 'Item', 'Package', 'SPackage' 총 4가지의 품목 유형을 가지고 있다.

```
// Struct - Product
typedef struct Struct_product {
    // 해당 구조체 객체의 데이터는 최초로 생성된 후 수정하는 것을 금지함

    Date date_assigned; // 해당 제품의 데이터가 처음으로 기입(등록)된 날짜

    char* name_product; // 해당 제품의 이름
    char* description; // 해당 제품의 설명
    char* name_vendor; // 해당 제품 공급자의 이름

    // Struct constructor
    Struct_product( const char* product_name, const char* product_description=(str_u8_none.c_str()),
    {
        strcpy(name_product, product_name); // assign the pointer, name of product
        strcpy(description, product_description); // assign the pointer, description of product
        strcpy(name_vendor, product_vendor); // assign the pointer, vendor name of product

        assignCurrentDate(date_assigned); // assign the current date time
    }
} Product;
```

'Product' 유형은 사용자 정의 클래스로 선언되어 있으며, 일명 '제품' 으로 불린다.

여기에는 데이터 생성 날짜와, 제품의 이름, 제품의 설명, 제품의 공급자 정보만 포함되어 있다.

품목 유형의 최소 단위로 간주되며, 다른 품목 유형과는 차별화되는 다른 속성을 한 가지 가지고 있는데, 바로 제품 데이터는 생성된 이후 수정할 수 없다는 것이다. 생성된 제품 데이터를 삭제할 수는 있다.

제품은 고객에게 제공되는 데이터가 아니라, 단순히 관리 목적을 위해서만 취급되는 기본 단위 데이터의 속성을 가지도록 설계되었다.

```

class Item {
    // 품목 클래스 : 단일 또는 다수 제품의 집합
private :
    Date date_assigned; // 해당 품목의 데이터가 처음으로 기입(등록)된 날짜
    Date date_renewed; // 해당 품목의 데이터가 가장 마지막으로 수정된 날짜

    std::vector<Product> contents; // 해당 품목의 제품 구성 (벡터 컨테이너)

    char* name; // 해당 품목의 이름
    char* description; // 해당 품목의 설명
    char* manufacturer; // 해당 품목의 제조자(또는 업체)
    char* country_origin; // 해당 품목의 원산지
    char* vendor; // 해당 품목의 공급자(또는 업체)

    Date date_manufactured; // 해당 품목의 제조 일자
    Date date_expiry; // 해당 품목의 만료 기한
    ExpDate date_expiry_type; // 해당 품목 만료 기한의 유형 (유통 또는 소비 기한)

    Icat* category; // 해당 품목의 유형(분류)

    int mass_gram; // 해당 품목의 질량 (단위 : 그램)

    int price_item; // 해당 품목의 가격 (세금 제외)
    int price_tax; // 해당 품목의 세금 (일반적으로 품목가의 10%)
    int price_full; // 해당 품목의 정가(정가=품목가+세금, 할인을 적용 제외)
    double rate_discount; // 해당 품목의 정가 할인을 (예시 : 해당 값이 0.53인 경우, 정가에서 53%가 할

    unsigned int stock_actual; // 해당 품목의 실제 재고 수량

```

'Item' 유형은 사용자 정의 클래스로 선언되어 있으며, 일명 '품목' 으로 불린다.

item은 하나 또는 여러 개의 제품(Product)로 구성된다.

item부터는 실제로 고객에게 제공할 수 있는 수준의 상세한 데이터를 기입할 수 있도록 한다. 또한 생성된 데이터를 추후에 일부 수정할 수도 있다.

item에는 데이터 생성 날짜, 데이터 수정 날짜, 제품 컨테이너, 이름, 설명, 제조자, 원산지, 공급자, 제조 일자, 만료 기한, 만료 기한 유형(유통 기한 또는 소비 기한), 분류(카테고리), 질량, 가격, 세금, 정가, 할인율, 실제 재고 수량, 판매 가능 재고 수량, 무제한 판매 가능 여부, 판매 중 여부 정보가 포함되어 있다.

```

// Class - Package
class Package {
    // 패키지(품목 구성) 클래스 : 다수 품목의 집합
private :
    Date date_assigned; // 해당 패키지의 데이터가 처음으로 기입(등록)된 날짜
    Date date_renewed; // 해당 패키지의 데이터가 가장 마지막으로 수정된 날짜

    std::vector<Item> contents; // 해당 패키지의 품목 구성 (벡터 컨테이너)

    char* name; // 해당 패키지의 이름
    char* description; // 해당 패키지의 설명

    Icat* category; // 해당 패키지의 유형(분류)

    int mass_gram; // 해당 패키지의 질량 (단위 : 그램)

    int price_item; // 해당 패키지의 가격 (세금 제외)
    int price_tax; // 해당 패키지의 세금 (일반적으로 품목가의 10%)
    int price_full; // 해당 패키지의 정가(정가=품목가+세금, 할인을 적용 제외)
    double rate_discount; // 해당 패키지의 정가 할인을 (예시 : 해당 값이 0.53인 경우, 정가에서 53%가

    unsigned int stock_actual; // 해당 패키지의 실제 재고 수량
    unsigned int stock_available; // 해당 패키지의 판매 가능 재고 수량
    bool stock_unlimited = false; // 해당 패키지의 무제한 판매 가능 여부 (true인 경우, 판매 가능 여부
    bool on_sale; // 해당 패키지의 판매 상태 (true=판매 중, false=판매 중이 아님)

```

'Package' 유형은 사용자 정의 클래스로 선언되어 있으며, 일명 '패키지' 로 불린다.

패키지 여러 개의 품목(Item)으로 구성된다. 다양한 종류의 품목을 포함하고 있는 '과자 묶음' 등의 품목에 적합하도록 설계되었다.

패키지는 item과 마찬가지로 생성 이후에 정보를 수정할 수 있다.

패키지에 포함된 정보는 item과 같다. 다만 제품 컨테이너 대신에, item 컨테이너가 포함되어 있다.

또한, 제조자, 원산지, 공급자, 제조 일자, 만료 일자, 만료 일자 유형 정보가 없다.

```

// Class - Selectable Package
class SPackage {
    // 선택형 패키지(품목 구성) 클래스 : 다수 품목의 집합, 일정 갯수의 품목을 선택 가능
private :
    Date date_assigned; // 해당 패키지의 데이터가 처음으로 기입(등록)된 날짜
    Date date_renewed; // 해당 패키지의 데이터가 가장 마지막으로 수정된 날짜

    std::vector<Item> contents; // 해당 패키지의 품목 구성 (벡터 컨테이너)
    unsigned int content_amount_selectable; // 해당 패키지에서 선택 가능한 품목의 갯수
    bool content_duplicatable = true; // 해당 패키지에서 같은 유형의 품목 선택 가능 여부 (true인 경우

    char* name; // 해당 패키지의 이름
    char* description; // 해당 패키지의 설명

    Icat* category; // 해당 패키지의 유형(분류)

    int mass_gram; // 해당 패키지의 질량 (단위 : 그램)

    int price_item; // 해당 패키지의 가격 (세금 제외)
    int price_tax; // 해당 패키지의 세금 (일반적으로 품목가의 10%)
    int price_full; // 해당 패키지의 정가(정가=품목가+세금, 할인을 적용 제외)
    double rate_discount; // 해당 패키지의 정가 할인을 (예시 : 해당 값이 0.53인 경우, 정가에서 53%가

    unsigned int stock_actual; // 해당 패키지의 실제 재고 수량
    unsigned int stock_available; // 해당 패키지의 판매 가능 재고 수량
    bool stock_unlimited = false; // 해당 패키지의 무제한 판매 가능 여부 (true인 경우, 판매 가능 여부
    bool on_sale; // 해당 패키지의 판매 상태 (true=판매 중, false=판매 중이 아님)

```

'SPackage' 유형은 사용자 정의 클래스로 선언되어 있으며, 일명 '선택형 패키지'로 불린다.

선택형 패키지는 기본적으로 패키지와 같다. 하지만 중요한 차이점이 있다.

선택형 패키지는 여러 개의 품목(Item)으로 구성된다. 그리고 해당 패키지에서 선택 가능한 품목의 갯수와, 같은 유형의 품목 선택 가능 여부를 정의할 수 있다.

선택형 패키지는, 편의점의 '1+1 교차 선택 행사 상품' 등의 품목에 적합하도록 설계되었다.



마지막으로, 품목 분류(카테고리)가 있다.

```
typedef struct ItemCategory {
    Icat* category_parent;
    char* category_this;
    std::vector<Icat*> category_children;

    ItemCategory(const char* name_category_this, Icat* ptr_category_parent=NULL)
    {
        strcpy(category_this, name_category_this);
        category_parent = ptr_category_parent;
    }
} Icat;
```

카테고리는 프로그램 사용자가 문자열 형태로 정의하여 생성할 수 있는, 각 유형의 품목을 더 쉽게 분류하고 검색하기 위한 목적으로 추가한 사용자 정의 구조체이다.

각 카테고리는 본인 카테고리의 이름과, 단일 상위 카테고리를 참조할 수 있도록 부모 카테고리 포인터, 단일 또는 복수의 하위 카테고리를 참조할 수 있도록 자식 카테고리 포인터의 컨테이너를 가지고 있다.

품목 유형은 아직 데이터를 추가/삭제하는 기능을 만들지 못했다.

원래는 각 품목 유형을 관리하는 데이터베이스 또한 있으나, 마찬가지로 시간 부족의 이유로 완성하지 못하여 프로그램의 핵심 기능을 선보일 수 없게 되었다.

본래는 각 품목 유형을 프로그램 사용자가 직접 추가하고, 삭제하고,

이 부분에서 너무나 큰 아쉬움을 느낀다.

처음부터 너무 많은 기능을 구현하려고 욕심을 낸 것이 아닌가 하는 생각도 든다.

발표일 이전까지 구현하지 못한 기능을 구현하여, 조금이라도 청중들을 이해시키는 데 노력하려고 한다.

보고서를 마칩니다.

작성 일자 : 2025. 06. 03.

작성자 : 전형진(5820658)