물품 관리 프로그램

기말과제 – 실행결과

학번: 2018102090

학과: 소프트웨어융합학과

작성자: 고대훈

**, 인터페이스 설계 방향.**

**1. 구현한 시스템의 특징**

내가 구현한 물건관리프로그램의 가장 눈에 띄는 특징은 ItemType에 ‘**검색 횟수**’가 있고 이를 통해 **추천 기능 및 검색 기록 기능**을 구현했다는 것이다.

내가 생각하기에 물건관리프로그램은 물건들을 잘 차곡차곡 저장하기 위함도 있지만 가장 주요한 기능은 물건을 얼마나 잘 찾는가라고 생각했다. 따라서 기존의 내가 검색한 기록들을 토대로 새로운 정보를 만들어 이를 제공하는 함수가 있고 이전 검색으로 돌아가는 등 여러가지 기능을 제공한다.

좀 더 자세히 설명하면, 살다보면 물건을 버러야할 때 무엇을 버려야할 지 못 고를 때가 있다. 따라서 검색 기록이 없고 3년 이상 지난 물건을 버리도록 추천한다. 이런 식으로 검색 기록을 이용하여 다양한 기능을 제공한다.

또한 검색 기록을 STACK으로 구현하여 이전 검색기록으로 하나하나 돌아갈 수 있다.

물건의 수량이 부족하고 물건이 저장된 지 오래되어서 새로 사야된다고 판단되어 사용자에게 사야될 물건을 추천할수도 있다.

**2. 프로그램 주요 기능**

|  |  |
| --- | --- |
| 번호 | 기능 |
|
| **01** | **Add Item** |
| **02** | **Delte Item** |
| **03** | **Modify Item** |
| **04** | **물품 검색하기(SearchByID , SearchByName …)** |
| **05** | **View(Storage-Container-item layer별로 보기)** |
| **06** | **Photo View(포토 이미지 출력 & 사진 text로 인식)** |
| **07** | **버리길 추천하는 물품 리스트 보기 Ex)잘 안 찾고 오래된 물품들…** |
| **08** | **물건 넣은 컨테이너 추천하기** |
| **09** | **살 물건 추천받기** Ex)넣은 지 오래되었고 & 수량적은 물건 |
| **10** | **검색 기록 출력하기** |
| **11** | **이전 검색 기록으로 돌아가기**  **=> 이전에 검색한 아이템 정보 출력** |
| **11** | **파일 저장하기 & 파일 불러오기** |

**3. 자료구조의 타당성**

**1) Sorted List**

모든 아이템 정보를 담는 MasterList와 StroageList는 Sorted List를 사용했다. MasterList는 말할 것도 없이 가장 많은 검색이 일어나는 공간으로 빠른 정보 제공을 위해 빠른 검색 속도를 필요로 한다. 따라서 SortedList를 사용했다. StorageList는 MasterList보다 많은 검색은 일어나지 않지만 모든 Storage가 모여있어 MasterList 다음으로 검색이 많이 일어나는 공간이다. 또한 나는 StorageList를 6개로 정했다. 따라서 추가 삽입/삭제 연산이 빨리 일어날 필요가 없고 검색 속도를 중시해서 SortedList를 사용했다.

모든 일정의 정보가 아이디나 날짜 순으로 정렬되어 있는 것이 더 보기 편할 것이라고 생각했기 때문이다. 말 그대로 MasterList는 모든 아이템 정보가 담겨져있는데 아무런 정렬없이 저장된다면, 추후에 item을 찾는 일이 매우 힘들어질 것이다. 지금은 Id 순으로 sorted List를 만들 생각이지만 추후에 Frequently Search item 이나 버리길 추천하는 물품리스트를 작성할 때 편의성을 위해 변동될 가능성이 있다.

**2) Unsorted List**

ContainerType의 itemList는 Unsorted List를 구현했다. 우리가 평상시에도 컨테이너나 장에 물품을 넣을 때 기존에 있던 물품을 다 꺼내서 정렬 후 넣지 않기때문이다. 그냥 아무런 순서나 정렬없이 컨테이너에 넣기 때문에 itemList도 이에 맞춰 Unsorted List로 구현했다.

**3) Singly Linked List**

StorageType 안에 있는 ContainerList는 Singly Linked List를 사용했다. 이 이유는 검색보다 삽입과 삭제가 더 많이 일어난다고 생각했기 때문이다. 현재는 아이템 정보가 많이 없어서 Container 개수가 많이 없지만 추후에 이 프로그램을 계속 사용하게 된다면 Container 수는 기하급수적으로 늘어날 것이고 삽입/삭제 연산이 빨리 일어나야 한다. 따라서 Singly Linked List를 사용했다. 이때 Doubly Linked List는 이전 포인터, 이후 포인터를 필요해서 메모리 공간을 많이 차지해서 Singly Linked List를 선택했다.

**3) STACK**

SearchList는 STACK 형식으로 구현했다. 이는 SearchList 특성을 고려했기 때문이다. 내가 생각하는 SearchList는 네이버의 뒤로가기 버튼처럼 이전 검색 기록으로 계속 넘어갈 수 있어야 한다. 따라서 SearchList에서 STACK 구조로 하여 계속 POP하여 이전 기록으로 가야한다. 따라서 STACK을 선택했다

**4. 인터페이스 설계방향**

나의 프로그램 인터페이스 설계방향의 편의성이다. 인터페이스의 주목적은 프로그램은 보다 편하게 쓰기 위함이라고 생각했기 때문이다.

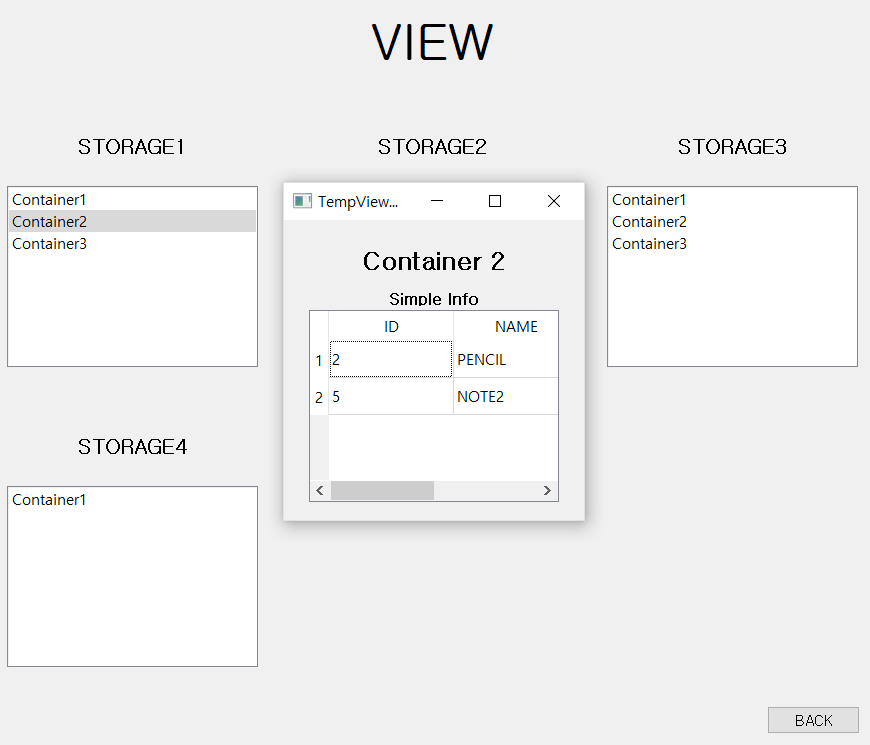
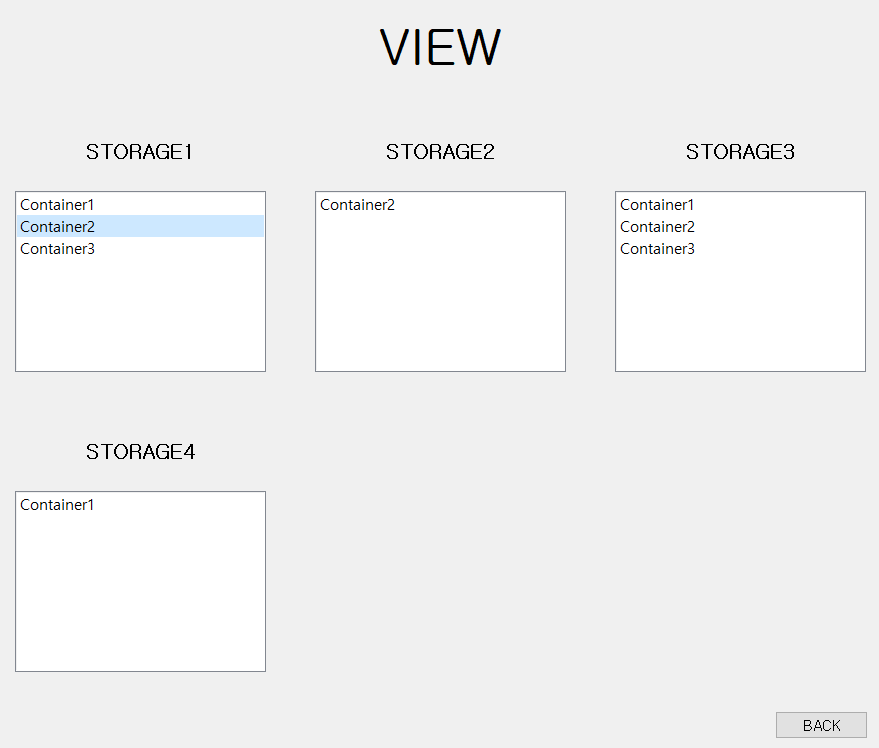
특히 Layer별로 View를 잘 할 수 있도록 했다.

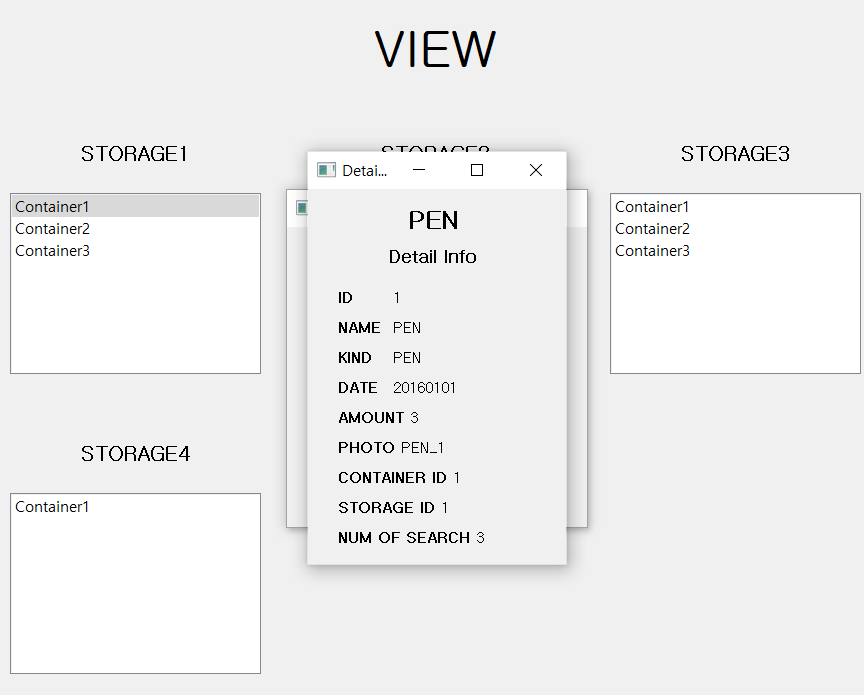
Storage – Container – Item – DetailInfo 순으로 하나하나 클릭하면서 사용자는 하나씩 좀 더 넓은 공간에서 좁은 공간으로 들어가 아이템의 자세한 정보를 검색할 수 있도록 했다.

시각적으로 어디에 무엇이 있는지 볼 수 있게 설계해서 보다 편안한 사용이 가능할 것이라고 생각한다.

아래의 그림을 보면 더 쉽게 이해할 수 있을 것이다.

**실행결과 - 5. View 화면 -**





**– Container 더블 클릭시 Container 안에 정보 뜸**

**- 안에 Item 클릭시 자세한 정보 출력**

이런 식으로 사용자가 아이템을 자세한 정보를 보기위해 하나하나 살펴볼 필요가 없고

Storage - Container - Item 식으로 Layer 별로 볼 수 있게 된다.