Music Management 명세서

프로그램의 구조는 Base class에 Member List가 선언되어져 있다. Application class에서 메뉴를 출력하고 수행할 기능을 선택받아서 적절한 함수를 호출한다. 모든 자료들은 각자에 알맞은 자료형으로 관리된다. ItemType으로는 Song, Genre, Artist, Album, Simple이 있다. 인터페이스를 위해 가장 하위 타입에서도 리스트내 삭제 및 수정을 가능하기 위해서 Base를 필요한 클래스에 상속시켰다.

프로그램 실행 시 메인 메뉴가 출력되고 메인 메뉴의 내용은 다음과 같다.

1. 로그인
2. 회원가입
3. 프로그램 종료

회원 가입을 할 경우 User란 폴더 내에 ID를 제목으로 한 파일이 생기고, 파일 내에는 비밀번호와 그 유저의 음악목록을 저장한다.

로그인을 하는 중 admin이라는 특수한 command를 입력하면 유저들을 관리할 수 있는 커맨드화면으로 넘어가서 파일의 삭제가 가능해진다.

로그인을 선택 하여서 회원ID로 로그인할 경우 회원 메뉴를 출력한다

회원 메뉴는 다음과 같다.

1. 음악 추가
2. 음악 검색
3. 음악 보기
4. 사용 종료

여기서 음악 추가를 할 경우, 각 List 내에 해당 음악이 추가된다.

음악 검색을 하는 경우

1. 제목으로 검색
2. 가수별 검색
3. 앨범별 검색
4. 작곡자별 검색
5. 작사자별 검색
6. 소속사별 검색

커맨드가 출력되는데, enum을 활용한 searchType으로 MusicList내에서 해당 data에 대한 정보들을 검색한다.

음악 보기를 하는경우

1. 모든 음악보기
2. 장르별 음악 보기
3. 가수별 음악 보기
4. 앨범 목록 보기
5. 최근 추가 목록 보기

화면이 나오는데, 모든 음악 보기를 할 시 MusicList 내에 모든 정보를 출력하고, 장르, 가수, 앨범 목록 보기를 선택 할 시 각 GenreList, ArtistList, AlbumList 내에서 갖고있는 타입들의 정보를 보여준다. 여기서 그 Type을 선택할시 해당 Type에 수록된 곡들의 Simple정보가 나오는데, 여기서 방향키로 또 곡을 선택하고 Enter를 누를 시

1. 곡 보기 및 재생
2. 삭제
3. 수정

커맨드가 좌측에 생긴다. 여기서 곡 보기 및 재생을 하면 곡의 자세한 정보와 노래가 나오고, 삭제를 할 시 모든 List내에서 그 곡을 삭제한다. 수정도 마찬가지로 모든 List내에서 함께 수정이 이루어진다.

각 클래스의 ADT는 다음과 같다.

|  |  |
| --- | --- |
| 클래스명 | 기능 |
| Application | 프로그램의 기능을 총괄한다. 프로그램이 실행되는 동안 사용자가 선택할 수 있는 메뉴를 출력한다. |
| Base | 각 List(MusicList, AlbumList, ArtistList, GenreList)를 static 변수로 갖는다. |
| AlbumList | AlbumType을 RedBlack Tree 형태의 멤버변수로 갖는 List이다. 앨범 목록들을 관리하고 있으며,Application에서 AlbumType으로의 접근을 용이하게 하기 위해서 만든 class이다. |
| AlbumType | 앨범의 커버 사진과 앨범명, 가수를 갖고 수록곡들을 Simple형태로 DoublyLinkedList을 이용해 멤버 변수로 갖고 있다. 앨범의 수록곡은 한정적이여서 검색 속도보단 삽입 삭제에 중점을 두어서 Doubly를 채택했다. Base를 상속하여서 이 class에서 곡의 삭제 및 수정이 이루어져도 상위 다른 리스트에도 모두 적용되게 하였다. |
| ArtistList | ArtistType을 RedBlack Tree 형태의 멤버변수로 갖는 List이다. 앨범 목록들을 관리하고 있으며,Application에서 ArtistType으로의 접근을 용이하게 하기 위해서 만든 class이다. |
| ArtistType | 가수의 이름과 소속사를 갖고 RedBlack Tree에 Simple의 형태의 가수의 곡들을 저장하고 있는 멤버변수를 가진다. Base를 상속하여서 이 class에서 곡의 삭제 및 수정이 이루어져도 상위 다른 리스트에도 모두 적용되게 하였다. |
| RecentMusicList | 최근 추가한 곡 목록을 저장하고 있는 리스트를 가지는 Class이다. 기본적으로 30개의 길이를 가지고 있으며, MAXSIZE를 수정하여 변경할 수 있다. 가득 찼을 시에 가장 먼저 들어온 음악을 삭제하고 새로 추가한 음악을 뒤에서부터 밀어넣는 Circular Queue의 형식을 갖고 있다. |
| MusicType | 음원의 정보가 저장될 클래스이다. 제목, 가수, 작곡자, 작사자, 앨범, 소속사, 출시일, 장르, 가사가 멤버변수로 선언되며, List에 정렬되는 유형에 따라 수정될 Primary Key 변수가 추가로 선언된다. 이 때, Primary Key를 기준으로 하는 비교 연산자 오버로딩을 구현하여 List의 정렬이 수행되도록 한다. |
| MusicList | MusicType을 ReBlackTree형태로 가지고 있는 일종의 MasterList이다. 모든 곡들의 정보를 관리하고 있으며, 그렇기에 Search속도를 향상 시킬 수 있는 RedBlackTree를 활용하였다. 모든 정보를 다 갖고 있다. |
| GenreType | 장르와 장르 고유의 primary key, 해당 장르에 속한 곡들을 Simple 타입으로RedBlack Tree 형태에 저장하는 멤버 변수를 가진다. Base를 상속하여서 이 class에서 곡의 삭제 및 수정이 이루어져도 상위 다른 리스트에도 모두 적용되게 하였다. |
| GenreList | GenreType을 SortedArray 형태로 멤버변수로 갖는다. GenreType은 primary key가 한정적이기 때문에 항목 접근 속도가 빠르고 일정하다는 장점을 가진 Sorted Array를 활용하였다. Application에서 GenreType의 접근을 용이하게 하기 위해 만든 class이다. |
| SimpleMusicType | 곡의 Number와 Name만을 변수로 갖는 Type이다. 연산자 오버로딩을 필수로 넣었고 각 타입들의 멤버변수로 쓰기 위해 만들었다. |
| SortedArray | 배열의 길이를 동적 할당 받는 배열이다. 음악이 추가 될시 delete 후 다시 동적 할당하여 배열의 길이를 조절한다. Template Class로 작성하여 사용자가 원하는 자료형을 List로 관리할 수 있으며, 본 프로젝트에서는 Music을 원소로 갖는 List로 구현한다. Generic Class를 구현하기 위해 List 자료형에 종속적인 메소드를 사용하지 않으며, 자료형 내부의 연산자 오버로딩을 통해 SortedList 내의 정렬 등의 연산이 수행되도록 한다. |
| ,  DoublySorted  LinkedList | Pre 포인터와 next 포인터를 갖고 있는 DoublyNodeType을 이용하여 각 데이터들을 유기적으로 연결해 주는 List이다. Templete로 작성하여 사용자가 원하는 자료형을 List로 관리할 수 있으며, Generic Class를 구현하기 위해 List 자료형에 종속적인 메소드를 사용하지 않으며, 자료형 내부의 연산자 오버로딩을 통해 연산이 수행되도록 한다. 중간 삽입 삭제가 잦거나 검색 그리고 자료의 양이 적당할 때(500개 미만) 사용한다. |
| DoublyIterator | DoublyLinkedList의 data를 읽기 위해 만든 friend class이다. 처음 선언시 해당 list의 포인터를 받아와서 iterator 내장 함수를 통해서 하나 하나 데이터를 읽을 수 있다. |
| RedBlackTree | Templete로 작성하여 사용자가 원하는 자료형을 List로 관리할 수 있으며, Generic Class를 구현하기 위해 List 자료형에 종속적인 메소드를 사용하지 않으며, 자료형 내부의 연산자 오버로딩을 통해 연산이 수행되도록 한다. **BST의 발전 버전인 자가균형트리이며 검색 삽입 삭제등 일정한 속도를 요구하면서 자료형이 어느정도 있어도 괜찮은 자료구조가 필요한 곳에 사용한다.** |
| RedBlackNode | RedBlackTree에 사용하기 위한 Node이다. Left, right, parent의 포인터와 템플릿형 데이터, internal과 external, red와 black의 정보를 가지고 있다. |
| RedBlackIterator | RedBlackTree의 데이터를 하나씩 읽기 위해 만든 iterator. Stack을 활용하여 가장 왼쪽부터 데이터를 읽어와서 오른쪽에 데이터가 더 없을시 pop을 통해 부모노드를 불러내는 방식이다. |
| Stack | 위의 RBIterator에 사용하기 위한 Stack. 선입후출으로 가장 최근에 들어온 정보를 빼주는 역할을 수행한다. |