프로젝트 제안서

**프로젝트명 : DB System by java**

팀 명 : ChanProgram

팀 원 : 박찬양

[1. 프로젝트 개요 3](#_Toc311796440)

[1.1 프로젝트 명](#_Toc311796441)

[1.2 개발 동기 3](#_Toc311796442)

[1.3 개발 목적 4](#_Toc311796443)

[1.4 향후 발전 방향 4](#_Toc311796444)

[1.5 개발환경 및 기기 5](#_Toc311796445)

[2. 배경 지식 5](#_Toc311796446)

[2.1 데이터베이스 5](#_Toc311796447)

[2.2 데이터베이스의 특징 5](#_Toc311796448)

[2.3 DB 라이브러리 구조 6](#_Toc311796449)

[3. 유사 서비스 8](#_Toc311796450)

[3.1 오라클 데이터베이스 8](#_Toc311796451)

[**3.1.1 분석 9**](#_Toc311796452)

[3.2 MySQL 9](#_Toc311796453)

[**3.2.1 분석 1**](#_Toc311796454)

[4. 개발 내용 10](#_Toc311796455)

4.0 클래스구조

[4.1 메인화면](#_Toc311796469)

[4.2 Define](#_Toc311796465)

[4.3 Enter](#_Toc311796466)

[4.4 Browse](#_Toc311796467)

[4.5 Search](#_Toc311796468)

[4.6 Modify](#_Toc311796469)

[4.7 Delete](#_Toc311796469)

[4.8 Print](#_Toc311796469)

[4.9 Save](#_Toc311796469)

[4.10 Load](#_Toc311796469)

[5.개발 일정](#_Toc311796463)

[6. 체크포인트 리스트(Check Point List)](#_Toc311796464)

[6.1 Define](#_Toc311796465)

[6.2 Enter](#_Toc311796466)

[6.3 Browse](#_Toc311796467)

[6.4 Search](#_Toc311796468)

[6.5 Modify](#_Toc311796469)

[6.6 Delete](#_Toc311796469)

[6.7 Print](#_Toc311796469)

[6.8 Save](#_Toc311796469)

[6.9 Load](#_Toc311796469)

[7. 멘토링 보고서](#_Toc311796470)

[8. 자료 및 참고 문헌 출처](#_Toc311796471)

[9. 설계서 수정 내용](#_Toc311796469)

프로젝트 개요

## **1.1 프로젝트 명**

**DB System by java**

## **1.2 개발 동기**

1994년도에 개발된 Turbo C로 구현된 데이터베이스 프로그램을 알게 되고 실행을 해 보았는데 옛날에 작성된 프로그램이라 도스 환경에서만 실행 가능하여 불편함이 있었다.

이에 불편함을 해소하고자, 또 개선할 부분은 개선하고자 프로그램의 운영체제 확장을 통한 호환성 확보와 더불어 현재 가장 많이 사용되는 프로그래밍 언어 중 하나인 java의 객체지향 특징을 채택하여 이 프로그램을 수정하고 재구현하기 위한 개발을 시작하게 되었다.

## **1.3 개발 목적**

첫번째로, 가장 많이 쓰이는 언어 중 하나인 자바의 기본적인 데이터베이스를 구현함으로써 운영체제 호환성을 늘리는 것. 두번째로, 필수적인 기능을 담고 있는 기본 데이터베이스를 구현할 것이다; 초기 데이터베이스의 기능은 스키마 정의, 값 입력, 값 변경, 값 삭제, 출력, 저장, 불러오기 등이 있이 있다. 세번째로, 자바의 객체지향 특징을 잘 살린 구조로 오픈소스로 공개될 이 코드에 대해서 다른 사용자들이 편하게 사용 및 응용할 수 있도록 하는 것이 목표이다.

## **1.4 향후 발전 방향**

기존의 TurboC에서 구현된 DB 시스템에서 더나아가 JAVA에서 구현할 DB 시스템은 자바의 장점인 객체지향을 중요시 하여 객체 지향적인 구성으로 유지보수 효율과 코더가 코드를 이해하는데 어려움이 없도록 구성하여 이에 응용에 있어서 큰 어려움이 없게 하는 것이 목표이다.

현재 데이터베이스 시장을 점유하고 있는 많은 DBMS들이 존재하는데,

이러한 DBMS들이 지원하는 기능들은 매우 방대해 기능에 있어서 이 프로그램이 경쟁에서

생존하는것은 자원과 시간의 부족으로 힘들어 보인다.

따라서 기능 자체는 많지 않더라도 필수적인 기능을 갖춘 가볍고 빠른 실행 속도를 이 DB 시스템의 장점으로 삼을 것이다.

또한 이 프로그램은 오픈소스라는 것에 있어서 무료인 것이 큰 장점으로 예를 들면 DB를 사용해야 하지만, 자금에 부담이 되어 사용하지 못하거나 기업에 최적화하기 위한 자사만의 가벼운 데이터베이스 시스템이 필요한 기업들이 가져다 응용해 사용하는데 부담이 없다는 것을 장점으로 개발할 계획이다.

## **1.5 개발환경 및 기기**

지원 OS: Windows

Language : Java SDK

개발환경 : Windows 10

# **2. 배경 지식**

## **2.1 Database**

데이터베이스는 여러 사람이 관리, 사용 등을 할 수 있는 데이터 집합으로써

1950년 미국에서 처음 사용된 용어이다.

기존의 데이터를 저장하는 데 사용하던 파일 시스템은 데이터가 중복되거나, 관리하기가 힘든 여러 단점이 있었는데, 이에 고안된 데이터베이스는 자료의 중복을 없애고 자료를 구조적으로 형성해 여러 동작에 있어서 효율성을 높인다.

데이터베이스는 여러 가지 종류가 있는데, 시간 순서대로 계층형 데이터베이스, 관계형 데이터베이스, 객체 지향형 데이터베이스, 객체 관계형 데이터베이스, NoSQL, NewSQL 등이 있다.

데이터베이스는 실시간으로 접근 가능하고 지속적인 변화를 하고 있으며 동시 공유, 내용에 대해 참조 가능하며 데이터가 중복되지 않아야 하는 등의 특징을 가지고 있다.

데이터베이스의 장점은 데이터 중복을 피할 수 있고 입력된 데이터를 공유하고 데이터 접근이 쉬우며 데이터가 저장되는 공간을 줄일 수 있는 등의 장점이 있다.

단점으로는 사용에 전문가가 필요하며 비용 문제와 데이터 백업과 복구의 어려움, 복잡성 등이 있다.

더불어 현재 시대에 이르러서 더욱더 많은 데이터가 생성되는 이른바 빅 데이터 시대에 도달하게 되면서, 더 많은 데이터를 저장해야 할 필요성과 함께, 데이터베이스의 필요성은 매우 커지고 있다.

현재 데이터베이스 시장을 크게 점유하고 있는 대부분의 DBMS는 데이터베이스의 기본적인 기능과 더불어서 데이터베이스 백업 및 복구 등의 서비스를 지원하며 개발자의 편의를 위한 GUI 등을 많이 갖추고 있다.

## **2.2 DB의 특징**

이전의 DB의 역할을 하던 파일 시스템과 비교하였을 때, DB의 특징은 다음과 같다.

첫번째로 실시간 접근이 가능하다는 것이다; 실시간 처리와 그에 대한 응답이 가능하다.

두번째로 지속적인 변화를 한다는 것이다; 새로운 데이터가 들어가거나 삭제되거나 조정되는 등의 지속적인 변화가 일어나는 것이 DB의 특징이다.

세번째로, 여러 사람이 한번에 같은 데이터를 보거나 사용할 수 있어야 한다.

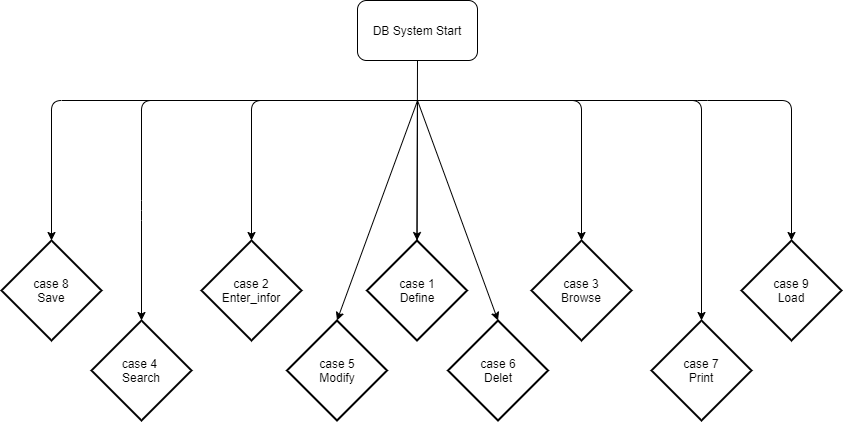
네번째로 사용자가 필요로 하는 데이터를 기준으로 하여서 다른 데이터를 참조한다.

마지막으로 데이터 독립성이 있어야한다; 데이터 독립성이란 하위 데이터가 바뀌어도 상위 데이터에게는 영향이 없어야한다는 것으로 파일 시스템에서는 데이터가 변경되면 프로그램이 수정되어야 하고, 데이터가 일관성이 없어지는 등의 문제가 생겨 이를 막고자 DBSM이 나오게 된 것이다. 이 데이터 독립성을 지키기 위한 방법은 여러가지인데, 논리적 데이터 독립성과 물리적 데이터 독립성이 있다. 예를 들면 3레벨 스키마구조 방법을 이용하여 데이터 성립성을 지킬 수 있다. 3레벨 스키마 구조는 외부, 개념, 내부로 스키마를 나누어서 모든 데이터베이스 부분을 사용자에게 보여주는 것이 아닌, 일부만 보여주는 방법이다.

## **2.3 DB 라이브러리(메소드) 구조**

이번 프로젝트에서 말하는 DB 라이브러리 구조는 곧 메소드 구조를 의미한다.

e



[그림 2.1] 데이터베이스 시스템 주 메소드

그림을 보면 이번 프로젝트의 데이터베이스 시스템에서 지원하는 기능들이 적혀있다.

아래에 열거된 기능은 DbSystem 클래스 내부에 구현된다.

세부적으로 아래 기능의 구현에 필요한 부가적인 기능은 DbOperator 클래스에서 구현한다.

세부적인 구현에 관한 내용은 4.0 개발내용에서 다루도록 한다.

첫번째로 define()은 데이터베이스를 정의하는 부분인데 정의할 필드의 개수와 각 필드에 들어갈 데이터 크기등을 지정하는 기능을 가진다.

두번째로 enterInfo()는 필드에 레코드 value를 입력하는 메소드이다.

세번째로 browse는 각 필드의 레코드들이 무엇이 있는지 확인할 수 있으며 레코드 위치를 반환하는 메소드이다; browse를 이용하여 search, modify, delete 등 함수에도 응용할 수 있다.

네번째 search는 찾으려는 레코드를 입력하여 레코드가 존재한다면 계속 검색을 하거나 찾은 레코드의 위치를 반환하는 메소드이다.

다섯번째, modify는 레코드를 선택하고 레코드 값을 바꿀 수 있는 메소드이다.

여섯번째, delete는 필드를 선택하고 삭제하려는 레코드를 입력하고 삭제해주는 메소드이다.

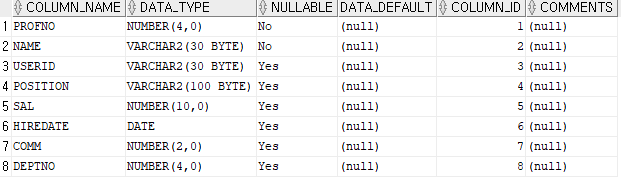
일곱번째, print는 모든 레코드를 출력하는 메소드이다.

여덟번째, save는 지금까지 입력되고 정의된 데이터베이스를 저장하는 메소드이다.

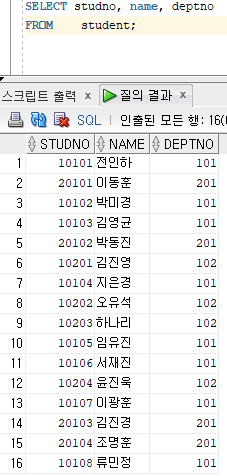
아홉번째, load는 저장된 데이터베이스를 불러오는 메소드이다.

# **3. 유사 서비스**

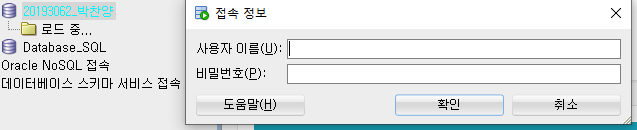
## **3.1 Oracle**



[그림 3.1] 데이터 출력 화면



[그림 3.2] 사용자가 원하는 방식대로 데이터를 표현해주는 모습 (질의어 이용)



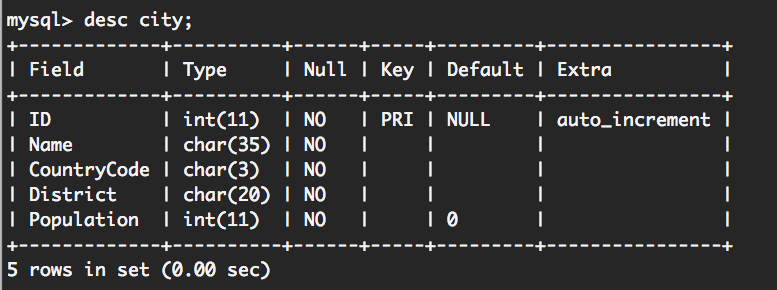
[그림 3.3] 사용자별 데이터베이스를 다르게 설정할 수 있고 로그인 기능도 지원하는 모습

3.1.1 분석

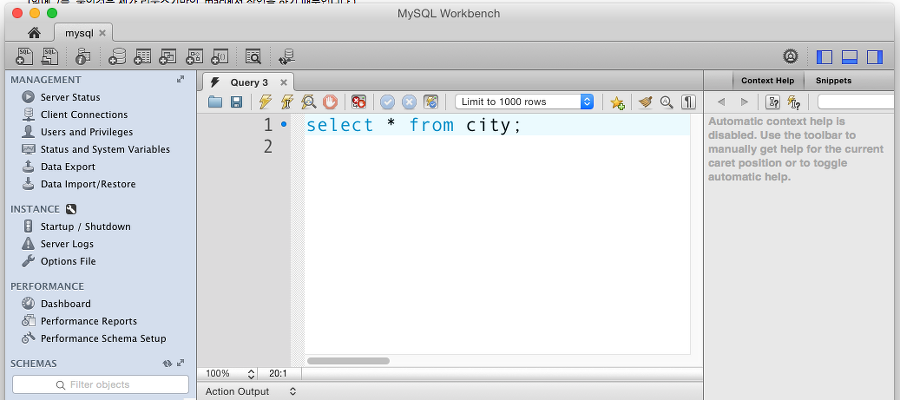
현재 데이터베이스 시장을 점유하고 있는 기업인만큼,오라클은 지원하는 기능이 매우 방대하고 사용자의 편의도 충족한 것으로 보인다.

UI 디자인은 약간 낙후됐다고 느껴지며 기능만큼 프로그램이 무거운 것이 단점이다.

3.2 MySql



[그림 3.4] MySql에서 테이블의 구조를 확인하는 모습



[그림 3.5] Workbench를 이용하여 MySql에 접속한 모습

3.2.1 분석

오라클과는 다르게, UI가 한글을 지원하지 않아 한국인에게 약간 불편할 수 있다.

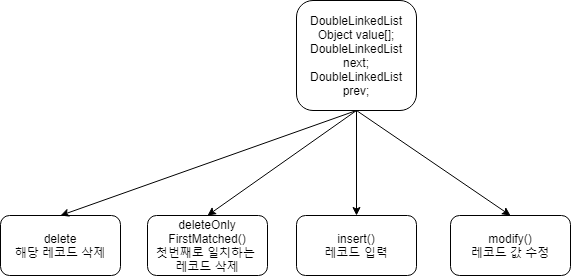
오라클보다 기능은 적겠지만 그만큼 가벼운 프로그램이다.

또한 가격면에서도 오라클에 비하면 가격대비 성능이 나쁘지 않은 것으로 보인다.

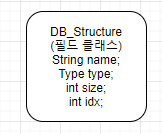
두 DBMS는 명령어도 조금씩 다르기 때문에, DBMS 마다 약간의 착오가 생길 수 있다.

**4. 개발 내용**

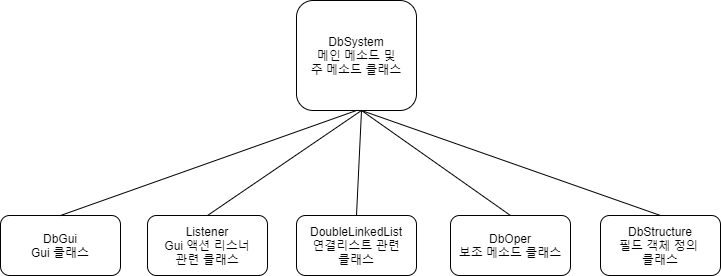
**4.0 클래스 구조**



[그림 4.0.0] 연결리스트 클래스



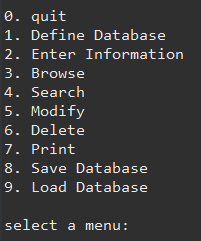
[그림 4.0.1] 필드 클래스



[그림 4.0.2] 대략적인 전체 클래스 구조

## 

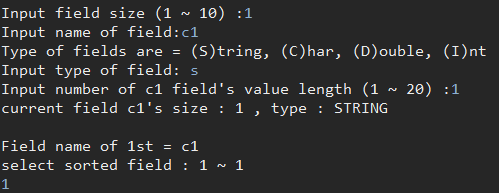
## **4.1 메인 화면**



[그림 4.1] 메인 화면

케이스를 선택하여 여러 기능을 이용할 수 있다.

## **4.2 Define()**



[그림 4.2] Define

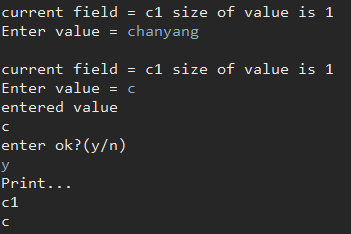
필드는 그림 4.0.1에서와 같이 DbStructure에서 String name, int size,int idx,Type type으로 정의한다.

Type은 열거형 클래스로서 STRING,CHAR,DOUBLE,INT를 담는다.

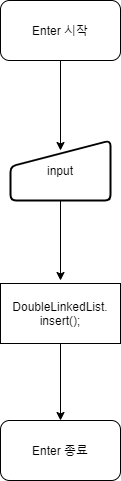
필드는 각 객체로서 DbSturcture을 갖는 배열인 dbControl를 필드에 선언하여 dbControl을 통해 각 필드에 접근할 수 있게 한다.

dbControl의 크기는 코드 내에 설정한 maxFieldNum과 같다.

**4.3 Enter()**



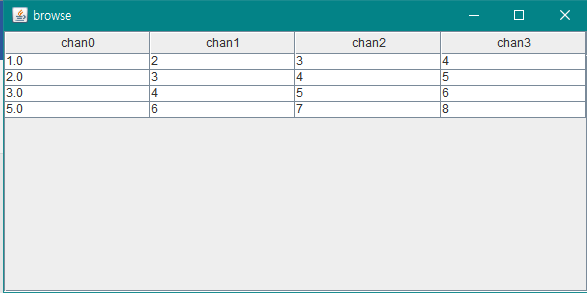
[그림 4.3] Enter



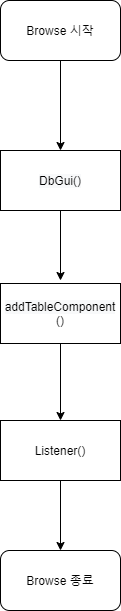
[그림 4.3.1] Enter 순서도

입력된 값은 연결리스트 클래스인 DoubleLinkedList의 value에 들어갈 것이며 각 연결은 dbControl을 통해 각 필드의 name과 idx를 보여주며 입력을 하게 하는 식이다.입력된 값은 DoubleLinkedList 클래스의 insert() 메소드를 통해 연결하도록 한다. 입력이 끝난 후에는 DoubleLinkedList의 print 메소드를 이용하여 현재 입력된 값들을 보여준다.

**4.4 Browse**



[그림 4.4] Browse

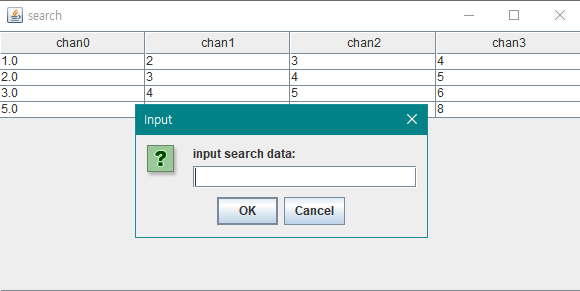


[그림 4.4.1] Browse 순서도

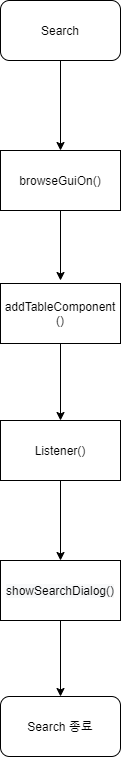
이전 계획서에서 browse는 사용자가 입력해주어야하는 부분이 많아 Gui방식으로 대체하였다. 사용자는 레코드를 선택하여 레코드 위치를 반환할 수 있다. 혹은 그냥 X를 눌러 종료할 수 있으며 이 경우 레코드 위치는 0 0을 반환하며 이것은 각각 몇번째 입력된 row이며 몇번째 필드의 값인지를 의미한다.

맨 위 chan0 chan1..은 필드 name을 의미, 아래 값들은 해당하는 레코드 값이다.

**4.5 Search**



[그림 4.5] Search

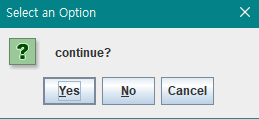


[그림 4.5.1] search 순서도

그림 4.6.0에서 볼 수 있듯이, search는 browse를 응용하여 구현하였다.

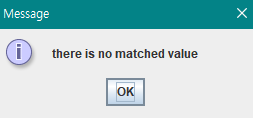
사용자는 찾을려는 데이터를 입력하고 데이터가 존재한다면 테이블 색깔을 바꿔 강조해주며 아래와 같은 팝업이 뜬다.

Yes를 누르면 같은 데이터가 더 있는지 한번 더 검색하며 NO를 누르면 현재 레코드 위치를 반환하고 Cancel을 누르면 기본값인 0,0을 반환한다. (browse에서와 의미 동일)



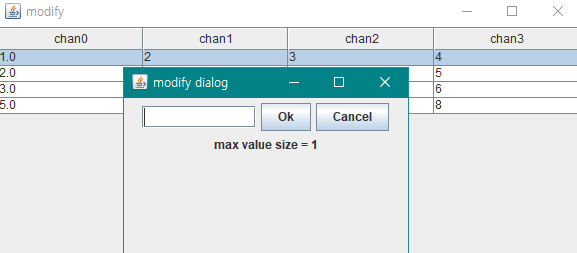
[그림 4.5.2] Search Option

데이터가 존재하지 않을 경우 아래와 같은 메시지를 띄우며 역시 기본값인 0,0을 반환한다.

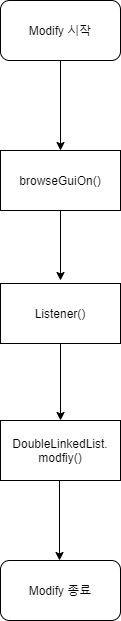


[그림 4.5.3] Search Option

**4.6 Modify**



[그림 4.6] Modify

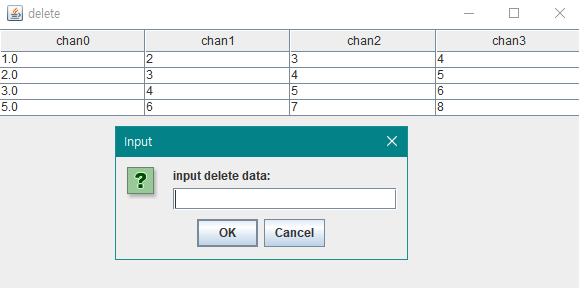


[그림 4.6.1] Modify 순서도

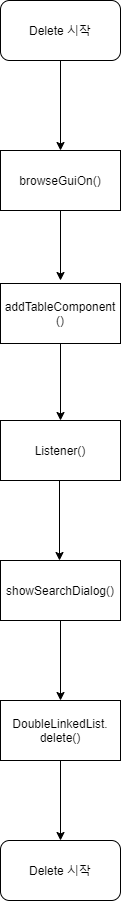
역시 기존코드와 달리 Gui로 구현방향을 바꾸었다. 순서도에서 볼 수 있듯이 browse를 응용하였으며 테이블에서 선택을 하면 그림 4.7과 같은 dialog가 출력되고 바꿀 값을 입력하여 값을 바꿀 수 있다. 이것은 browse가 선택한 레코드의 위치를 반환하는 점에서 고안하였다.

(값 타입이 맞지 않으면 Wrong value type이라는 메시지가 출력되며 다시 입력하도록 한다.)

**4.7 Delete**



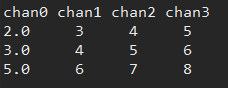
[그림 4.7] Delete



[그림 4.7.1 delete 순서도]

search 메소드를 응용하여 위 화면에서 삭제하려는 값을 입력하여 존재하는 값이라면 [그림 4.5.1]과 같이 선택하도록 하고 No를 선택하면 현재 선택된 값을 삭제하며 종료한다.

**4.8 Print**



[그림 4.8] Print

selective print 기능은 불필요하다 생각하여 삭제함.

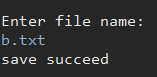
맨 위는 각 필드들의 이름과 그 아래에는 각 값들을 출력함.

레코드 출력은 DoubleLinkedList의 first를 이용하여 출력함.

실제 프린터에 출력되는 것이므로 주의

프린터 출력되는 형식을 콘솔에도 출력함

**4.9 Save**



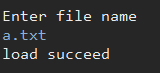
[그림 4.9] Save

원하는 파일 이름을 입력할 수 있고 기존코드에서는 DEF와 DAT파일로 나뉘는데, DEF파일에는 필드에 관한 정보가, DAT파일에는 레코드에 관한 정보가 담겨있다.

이 프로그램에서는 한 파일에 모든 정보를 담도록 하고, 사용자는 파일 이름만 입력하면 알아서 저장을 해준다.

필드의 정보는 DbStructure에 담긴 정보를 저장하도록 하고 공백으로 구분 짓는다. 또한 레코드의 정보는 레코드가 입력된 순서대로 first를 이용해 값을 참조하여 저장하도록 한다.

**4.10 Load**



[그림 4.10] Load

저장된 데이터베이스를 불러 오기 위해서 앞서 정의된 데이터베이스는 지워버리고

새롭게 Load하여 레코드 연결리스트를 연결하여 데이터베이스를 불러오는 메소드이다.

이것은 코드 내에서는 앞서 정의된 데이터베이스를 지우는 것은 자바에서는 갈비지 컬렉터를 지원하므로 메모리 할당해제 부분은 필요없다 다만 DbControl의 값과 클래스 필드값을 모두 초기화 해주어야 한다. 데이터 불러오는 부분은 Save에 저장된 데이터를 순서대로 불러서 DbControl을 이용하여 필드의 변수들을 입력해준다.

레코드 부분은 우선 DoubleLinkedList Save에서 순서대로 저장을 하였으므로 불러오는 것은 순서대로 DoubleLinkedList의 Insert 기능을 이용하여 저장한다.

# **5.개발 일정**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **순서** | **기간**  **작업내용** | **9** | | | | **10** |
| **1** | **2** | **3** | **4** | **1** |
| **1** | **참조 프로그램 분석** |  | **A** | **A** | **A** |  |
| **2** | **1차 설계도 작성** |  | **A** | **A** |  |  |
| **3** | **검토 의견서 제출** |  |  | **A** |  |  |
| **4** | **수정 설계도 작성** |  |  | **A** | **A** |  |
| **5** | **전체 프로그램 구조 설계** |  |  | **A** | **A** |  |
| **6** | **코드 레벨 메소드 설계** |  |  | **A** | **A** |  |
| **7** | **메소드 구현** |  |  | **A** | **A** | **A** |
| **7.1** | **Define 구현** |  |  | **A** |  |  |
| **7.2** | **Enter 구현** |  |  | **A** |  |  |
| **7.3** | **Browse 구현** |  |  |  | **A** |  |
| **7.4** | **Search 구현** |  |  |  | **A** |  |
| **7.5** | **Modify 구현** |  |  |  | **A** |  |
| **7.6** | **Print 구현** |  |  |  | **A** |  |
| **7.7** | **Save 구현** |  |  |  | **A** | **A** |
| **7.8** | **Load 구현** |  |  |  | **A** | **A** |
| **8** | **프로그램 테스트 및 디버깅** |  |  |  | **A** | **A** |
| **9** | **최종 테스트** |  |  |  |  | **A** |
| **10** | **문서화 점검** |  |  |  |  | **A** |
| **11** | **최종 제출 및 발표** |  |  |  |  | **A** |

**A : 박찬양**

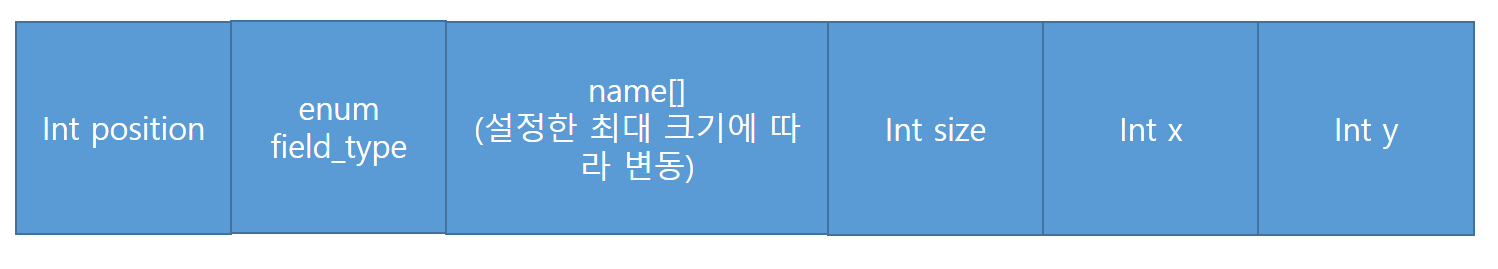
# **6. 체크포인트 리스트(Check Point List)**

**체크포인트 리스트에서는 점검이 필요하거나 향후 필요한 기능에 대해서 정리한다.**

## 

## **6.0 구조체**

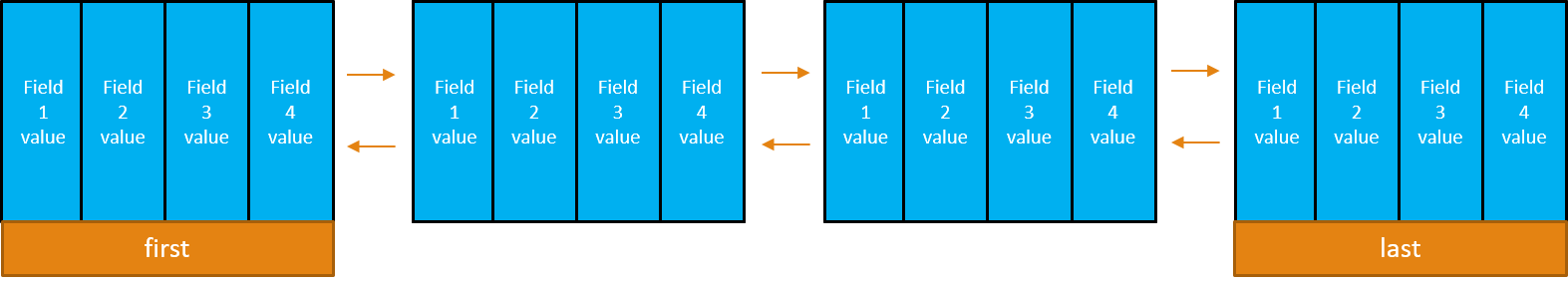
## 기존 코드의 필드의 구조체는 다음과 같다.



## [그림 6.0]구조체

## **각 필드들을 이중 연결리스트를 사용하여 연결한다.**

연결되는 방식은 다음과 같다.



[그림 6.0.1 연결리스트 연결 방식]

필드의 값은 배열로서 관리된다.

## **6.1 Define**

Define은 필드의 개수를 정의하고 각 필드의 이름, 데이터 자료형, 데이터 크기를 정의한다.

최대 설정 가능한 필드의 개수는 제한되어 있으며 최대 10개를 MAX로 잡는다.

필드를 다 정의한 뒤에는 정의한 내용들을 보여

주며 맞는지 확인하고 재입력할 수 있는 기능을 넣는다.

또한 레코드 정렬 기능을 위한 필드 선택 기능을 구현한다.

이때, 각 조건에 맞지않는 입력에 대한 예외처리가 필요하다.

## **6.2 Enter**

Enter는 각 필드의 레코드를 입력하는 메소드로서 각 필드는 이중 연결리스트로서 연결되도록 구현하여 수정이나 추가 등에 용의하도록 한다. Enter에서 레코드를 입력하는 것 또한 제대로 입력하였는지 확인하고 재입력할 수 있는 기회를 준다. 레코드는 Define에서 정의한 필드의 크기내에서 입력할 수 있도록 한다.

각 필드의 타입에 맞지 않는 입력 등에 대한 예외처리를 한다.

## **6.3 Browse**

Gui를 이용하여 보다 사용자가 입력해야하는 상호작용이 적도록 구현한다.

이에 Gui를 사용함으로서 생길 수 있는 여러가지 예외처리를 처리해야한다.

## **6.4 Search**

Search 기능 또한 Browse와 함께 현재 레코드를 반환 받는 함수로 다른 기능을 구현할 때 사용될 수 있도록 객체지향적으로 구현한다.

Search에서는 레코드를 선택할 수 있으며 중복된 레코드가 있을 수 있으므로 더 이상 입력된 레코드를 찾을 수 없을 때까지 검색할 수 있도록 한다.

선택된 레코드의 위치를 반환한다.

## **6.5 Modify**

레코드 값을 수정할 수 있도록 하는 Modify 기능은 Browse 기능을 응용하여 레코드를 선택하여 수정을 할 수 있도록 한다. 이때, 이 레코드가 맞는지 확인을 하고 타입에 맞지 않는 데이터타입 등 경우에 대한 예외처리가 필요하다.

## **6.6 Delete**

선택한 레코드를 삭제할 수 있는 Delete 기능은 Search를 응용하여 레코드 값을 입력한다. 존재하는 레코드라면 강조처리를 함으로서 가독성을 추가해주고 추가 확인 기능으로 실수로 삭제하지 않도록 한다.

## **6.7 Print**

모든 레코드들을 출력하는 메소드이다. 필드의 이름을 먼저보여주고 거기에 맞춰 레코드를 입력해주어 출력된 데이터의 가독성을 증가시킨다.

(프린트에 출력을 하는 것이므로 주의)

## **6.8 Save**

지금까지 입력된 필드, 레코드의 정보를 저장하려는 파일이름과 같은 파일로 저장하는 함수이다.

기존 코드에서는 두개의 파일로 구분을 하지만 현 프로그램에서 한 파일로 구분을 하도록 하고 파일을 열어보았을 때 가독성을 증가시킨다.

## **6.9 Load**

저장한 필드, 레코드의 정보를 파일로부터 불러오는 함수다.

불러오기 이전에 기존에 선언된 데이터베이스는 초기화해주어야 한다.

파일이름을 입력받고 그 해당하는 파일을 불러와 바로 사용할 수 있도록 한다.

save에서 저장하는 파일의 양식에 맞게 데이터베이스를 불러온다.

양식에 맞지않는 데이터 예외처리를 구현한다.

**7. 멘토링 보고서**

공란

# **8. 자료 및 참고 문헌 출처**

[그림 3.4] <https://developer88.tistory.com/20>

[그림 3.5] <https://developer88.tistory.com/20>

이외 그림 자체제작

[데이터베이스 개념]

https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%B2%A0%EC%9D%B4%EC%8A%A4\_%EA%B4%80%EB%A6%AC\_%EC%8B%9C%EC%8A%A4%ED%85%9C

**9. 계획서 수정 내용**

clrscr(), bioskey() 함수 설명 미비

* 실질적인 기능이 아니므로 자바에서는 구현하지 않을 것임.

오타 및 폰트 통일성

* 수정함

관련자료 출처 명확하게 표시하지 않음

* 추가함

그림 및 그림의 제목 가운데 정렬되면 좋겠다

* 수정함

일부 그림의 캡션 없음

* 수정함

**기각된 의견**

------------------------------------------------------------------------------------------

쌍반점을 안써도 되는 부분에서 쌍반점을 사용한다

개발환경 및 기기에 대한 설명 부족

* 자바에서,개발환경 및 기기는 의미 없음

개발 내용에서 원래 코드에서 어떤 부분이 자바 코드로 바뀌어서 구현된 건지 설명이 추가되었으면 좋겠다.

* 원래 코드에서 어떻게 자바 코드로 바뀌는 지는 실 구현에 대한 내용이고 이는 제안서에 담기에는 너무 디테일하다 판단함