

MiniProject_Document

*MiniProject_1

프로젝트 #1

❖ 우리는 스마트폰에 이름, 전화번호, 이메일등과 같은 연락처 정보를 저장하고 사용합니다.

아래의 데이터들을 저장하는 클래스(Addr.java)와 출력하는 메소드를 가지는 클래스 (AddrMain.java)를 만들어봅시다.

(Addr.java)

- ❖ 저장 정보
 - 이름/전화번호/이메일
 - 주소/생일
 - 생일
 - 그룹(친구,가족)
- ❖ 기능
 - 위 저장 정보를 출력하는 기능 (printlnf())

❖ 추가 요구사항

- 변수들은 직접 참조를 막아 캡슐화 처리를 합니다.
변수의 직접참조는 안되지만 변수의 값을 얻을 수 있는 메소드 getter와 변수에 값을 저장할 수 있는 메소드 setter를 정의합니다.
- 인스턴스 생성과 함께 데이터를 초기화 할 수 있도록 생성자룰 정의합니다.

❖ 실행과정

- main()메소드를 정의합니다.
- 연락처 데이터를 저장하는 인스턴스를 생성합니다
- 생성된 인스턴스의 정보 출력 메소드를 호출합니다.
- 인스턴스의 각 변수에 값을 바꾸는 메소드를 이용해서 데이터를 수정합니다.
(그룹 정보 변경 : 친구 -> 가족)
- 인스턴스의 출력메소드를 다시 실행합니다.

Console 출력 예시



```
<terminated> AddrMain [Java Application] C:\WProgram Files\Java\jdk-
이름 : 최윤희
전화번호 : 010-0000-0000
이메일 : choi@gmail.com
주소 : 서울
그룹 : 친구

-----
그룹 정보 변경
-----
이름 : 최윤희
전화번호 : 010-0000-0000
이메일 : choi@gmail.com
주소 : 서울
그룹 : 가족
```

*MiniProject_2

- 위 주소록(addr) 클래스를 바탕으로 CRUD기능을 추가합니다.

프로젝트 #2

❖ 프로젝트 #1에서 정의한 Addr 클래스를 기반으로 아래의 요구사항을 추가해서 프로그램을 만듭니다.

1. SmartPhone 클래스를 정의합니다(SmartPhone.java). 이 클래스는 연락처 정보를 관리하는 클래스입니다.
 - Addr 클래스의 인스턴스 10개를 저장할 수 있는 배열을 정의합니다.
 - 배열에 인스턴스를 저장하고, 수정하고, 삭제, 저장된 데이터의 리스트를 출력하는 메소드를 정의합니다
2. main()메소드를 아래의 요구조건을 정의해봅시다.(SmartPhoneMain.java)
 - Smartphone클래스의 인스턴스를 생성합니다.
 - 사용자로부터 입력을 받아 Addr 인스턴스를 생성해서 Smartphone클래스의 인스턴스가 가지고 있는 배열에 추가합니다. (처음에 연락처 등록 데이터 2개를 입력합니다)
 - 배열의 모든 요소를 출력합니다 (위의 연락처 2개 정보가 출력되는 확인한다.
 - 배열에 연락처 정보를 5개정도 추가해봅니다.
 - 배열의 모든 요소를 출력합니다.
 - 배열의 모든 요소를 검색합니다. (using contentEquals())
 - 첫날은 위의 메소드까지만 개발합니다.
 - 배열의 요소를 삭제해 봅니다.
 - 배열의 요소를 수정해 봅니다.

메소드 와 Console 출력 예시

❖ SmartPhone class

- inputAddrData() //키보드로 부터 입력 받아 객체를 생성함
- addAddr(Addr Addr) // 배열에 연락처 객체 저장
- printAddr(Addr Addr) // 객체 정보 출력
- printAllAddr() //모든 연락처 출력
- searchAddr(String name) //연락처 검색
- deleteAddr(String name) // 연락처 삭제
- editAddr(String name, Addr newAddr) // 연락처 수정

❖ SmartPhoneMain class

- printMenu() //메뉴 정보 출력(1. 연락처 등록 등)

```
Problems Javadoc Declaration Console x
SmartPhoneMain [Java Application] C:\Program Files\WJ
# 데이터 2개를 입력합니다.
이름 : 김성호
전화번호 : 010-1234-5678
이메일 : shkim@naver.com
주소 : 고양시 삼송
그룹(친구/가족) : 가족
>>> 데이터가 저장되었습니다. (1)
이름 : 홍길동
전화번호 : 010-1111-2222
이메일 : hong@naver.com
주소 : 서울
그룹(친구/가족) : 친구
>>> 데이터가 저장되었습니다. (2)
주소관리 메뉴-----
>> 1. 연락처 등록
>> 2. 모든 연락처 출력
>> 3. 연락처 검색
>> 4. 연락처 삭제
>> 5. 연락처 수정
>> 6. 프로그램 종료
-----
```

*MiniProject_3

Mini_Project #1-3 Single Linked List

❖ 다음은 연결 자료구조(Linked List)에 대한 내용이다.

❖ 노드

- 연결 자료구조에서 하나의 원소를 표현하기 위한 단위 구조
- <원소, 주소>의 구조



- 데이터 필드(data field)
 - 원소의 값을 저장
 - 저장할 원소의 형태에 따라서 하나 이상의 필드로 구성
- 링크 필드(link field)
 - 다음 노드의 주소를 저장
 - 포인터 변수를 사용하여 주소값을 저장

출력 예시

```
Problems Javadoc Declaration Console X
<terminated> SingleLinkedList [Java Application] C:\#Program
(1) 공백 리스트에 노드 3개 삽입하기
L = (1, 3, 7)
(2) 3 노드 뒤에 5 노드 삽입하기
L = (1, 3, 5, 7)
(3) 리스트의 노드를 역순으로 바꾸기
L = (7, 5, 3, 1)
(4) 리스트의 마지막 노드 삭제하기
L = (7, 5, 3)
```

*MiniProject_4

프로젝트 #1-4

❖ 프로젝트 #1-2에서 정의한 Addr 클래스를 상속의 구조로 만들어 봅시다.

1. Addr 클래스는 기본정보를 저장하고, 기본 정보를 출력하는 메소드를 정의합니다.
 - 생성자를 통해 기본 정보들을 초기화합니다.
2. 그룹에 해당하는 정보들을 추가적으로 정의하는 새로운 클래스들을 정의합니다. 회사, 거래처의 정보를 저장하는 하위 클래스를 정의합니다.
 - 회사의 정보를 저장하는 하위클래스를 정의합니다. (CompanyAddr class)
 - 회사이름, 부서이름, 직급 변수 추가
 - 정보를 출력하는 메소드를 오버라이딩해서 추가된 정보를 추가해서 출력
 - 거래처의 정보를 저장하는 하위 클래스를 정의합니다.(CustomerAddr class)
 - 거래처 이름, 거래품목, 직급 변수 추가
 - 정보를 출력하는 메소드를 오버라이딩해서 추가된 정보를 추가해서 출력
3. SmartPhone 클래스의 배열을 다형성의 특징을 이용해서 상위 타입의 배열을 생성해서 하위 클래스의 인스턴스를 저장하는 형태로 프로그램을 작성해 봅시다. (SmartPhone class)
4. SmartPhoneMain class에서는 지난번 코드와 동일하며 PrintMenu는 다음장의 콘솔 예시처럼 작동될 수 있도록 프로그램 해보십시오

Console 출력 예시



```
SmartPhoneMain (Java Application) C:\Program Files\Java\W...
# 데이터 2개를 입력합니다.
이름 : 김성호
전화번호 : 010-5466-5720
이메일 : shkim@naver.com
주소 : 상송
생일 : 0805
그룹 : 회사
회사이름 : 네이버
부서이름 : 개발팀
직급 : 팀장
>>> 데이터가 저장되었습니다. (1)
이름 : 김정호
전화번호 : 010-5466-5720
이메일 : shkimzebra@naver.com
주소 : 서울
생일 : 0806
그룹 : 거래처
거래처이름 : 네이버부동산
품목이름 : 부동산
직급 : 대리
>>> 데이터가 저장되었습니다. (2)
Contact -----
>> 1. 연락처 등록(회사)
>> 2. 연락처 등록(거래처)
>> 3. 모든 연락처 출력
>> 4. 연락처 검색
>> 5. 연락처 삭제
>> 6. 연락처 수정
>> 7. 프로그램 종료
```


*MiniProject_5

Mini_Project # 1-5

❖ 시나리오

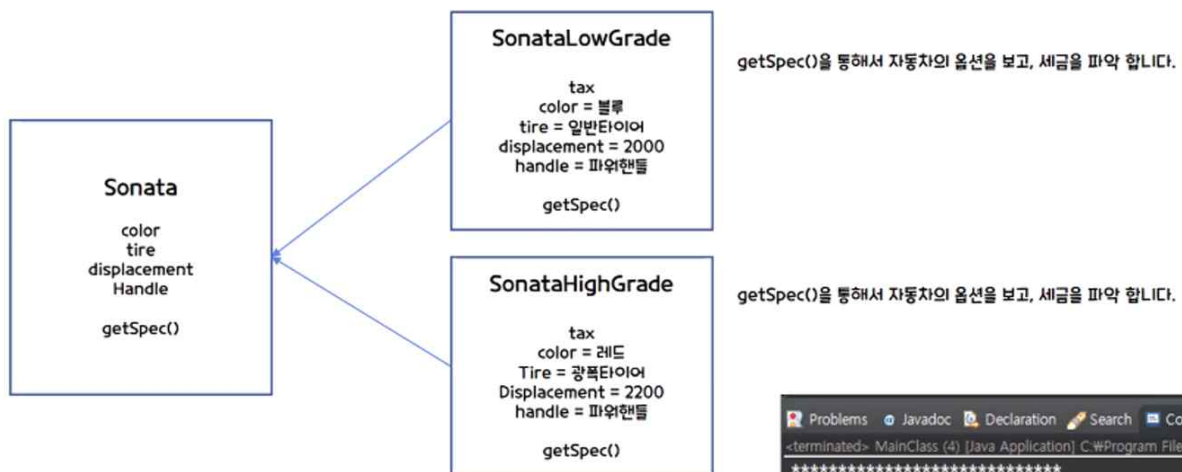
- 자동차의 경우 동일한 자동차 모델에서 옵션 사항에 따라 세금에 차이가 있습니다.
- 배기량이 높은 자동차의 경우에는 세금이 높습니다.
- 클래스 설계를 해 보도록 합니다.

❖ 자동차 스펙

- color (red,blue), tire(normal, wide), displacement(2000,2500), handle(normal, power)

❖ Class 설계

- CarSpecs class : 자동차 스펙에 관한 정보를 갖고있는 클래스
- Sonata class : Sonata 정보에 대한 abstract class
- SonataHighGrade class : high grade에 대한 정보를 갖고 있는 class
- SonataLowGrade class : low grade에 대한 정보를 갖고 있는 class
- MainClass : highGrade와 lowGrade에 대한 객체 생성 및 스펙정보를 호출하는 class



```
Problems Javadoc Declaration Search Console
<terminated> MainClass (4) [Java Application] C:\Program Files\Java\jre1...
*****
색상:블루
타이어:일반타이어
배기량:2000
핸들:파워핸들
세금 :1000
*****
색상:레드
타이어:광폭타이어
배기량:2500
핸들:일반핸들
세금 :1500
*****
```

*MiniProject_6

Mini_Project # 1-6

❖ 시나리오

S전자에서 다양한 스마트폰을 만듭니다. 모델별 스펙은 아래와 같습니다.

기능	전화 송/수신	접속속도	TV리모컨 기능
a제품	가능	3G	기본 미탑재
b제품	가능	5G	기본 탑재
c제품	가능	4G	기본 미탑재

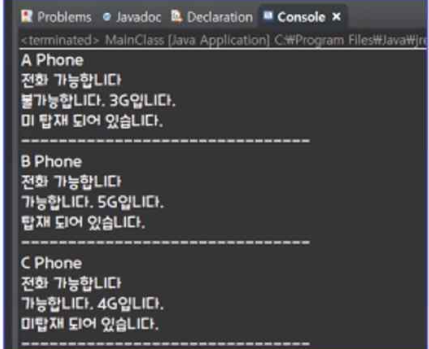
❖ Interface 설계 (IFunction.java)

각 기능으로 인터페이스를 설계한다.

❖ 구현 클래스(APhone/BPhone/CPhone.java)

❖ 결과값(MainClass.java)

각 제품 해당 기능을 콘솔에 Print out 합니다.



```
<terminated> MainClass [Java Application] C:\Program Files\Java\jre\bin\java.exe
A Phone
전화 가능합니다.
불가능합니다. 3G입니다.
미 탑재 되어 있습니다.
-----
B Phone
전화 가능합니다.
가능합니다. 5G입니다.
탑재 되어 있습니다.
-----
C Phone
전화 가능합니다.
가능합니다. 4G입니다.
미탑재 되어 있습니다.
-----
```

*MiniProject_7

Ch10-1 Exception handling basic programming

콘솔로부터 태어난 년도를 입력 받을 때 int 타입으로 입력을 받아 정상적으로 숫자를 입력 받았는지 확인하는 프로그램을 작성해봅시다.

사용자 예외 클래스를 정의해서 예외를 발생 시켜 봅시다. (InputException.java)

NumberTest.java에서 main()은 다음의 내용을 실행 합니다.

- 콘솔로 부터 숫자를 입력 받는다.
- 입력 받은 데이터가 숫자가 아닐 경우 "입력하신 데이터는 숫자가 아닙니다"라는 exception 메시지와 원인에 대해 출력이 되도록 코딩해봅시다.

Ch10-2 Exception handling basic programming

콘솔에서 사용자 아이디를 입력 받아 정상적인 영문자의 숫자로만 이루어진 값을 입력 했는지 확인하는 프로그램을 작성해봅시다.

사용자 예외 클래스를 정의해서 예외를 발생 시켜 봅시다. (InputException.java)

InputTest.java에서 main()은 다음의 내용을 실행합니다.

- 콘솔로 부터 아이디를 입력 받는다.
 - 입력 받은 아이디가 null, empty인지 확인한다
 - 입력 받은 이름이 정상이면 이름을 출력한다.
 - 입력 받은 이름이 비정상이면 예외 클래스를 호출하여
"잘못된 이름을 입력했습니다" 라는 메시지가 출력되도록 합니다.
-

*MiniProject_8

-MiniProject_4를 각 Collection Framework를 사용하여 변환합니다

프로젝트 #1-8

❖ 지금까지 진행한 프로젝트는 배열을 이용해서 연락처를 저장했습니다.

이번 프로젝트에서는 컬렉션 프레임워크를 이용해서 연락처 인스턴스를 저장하도록 프로그램을 만듭시다.

1. List<E>를 이용해서 저장 및 관리하는 프로그램을 만들어 봅시다. (using ArrayList)
2. HashSet<E>을 이용해서 저장 및 관리하는 프로그램을 만들어 봅시다.(using HashSet)
3. HashMap<K,V>를 이용해서 저장 및 관리하는 프로그램을 만들어 봅시다. (using HashMap)
(Key: phoneNumber, Value: Addr)

*MiniProject_9

프로젝트 #1-9

❖ 연락처 저장 및 불러오기 기능을 만들어 봅시다.

1. File 클래스를 이용해서 저장 폴더 생성
2. 메모리에 저장된 연락처 정보를 파일로 저장하는 기능 (6. 연락처 파일 저장)
3. 파일로 저장된 기능을 불러오는 기능 (7. 연락처 파일로드)
4. 전체 연락처 리스트를 출력하는 기능 실행 (5. 연락처 전체 리스트 보기)

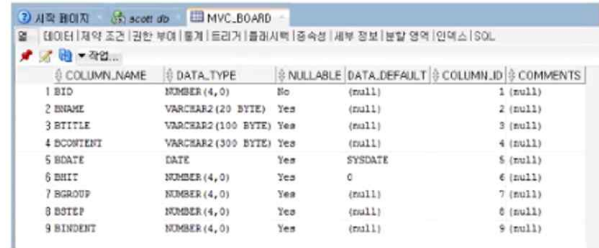
*MiniProject_11

-MVC 패턴으로 게시판 만들기

부록_03. DataBase 생성

```
create table mvc_board(
  bid NUMBER(4) PRIMARY KEY,
  bName VARCHAR2(20),
  bTitle VARCHAR2(100),
  bContent VARCHAR2(300),
  bDate DATE DEFAULT SYSDATE,
  bHit NUMBER(4) DEFAULT 0,
  bGroup NUMBER(4),
  bStep NUMBER(4),
  bIndent NUMBER(4)
);

create sequence mvc_board_seq;
```



COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
BID	NUMBER(4,0)	No	(null)	1	(null)
BNAME	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
BTITLE	VARCHAR2(100 BYTE)	Yes	(null)	3	(null)
BCONTENT	VARCHAR2(300 BYTE)	Yes	(null)	4	(null)
BDATE	DATE	Yes	SYSDATE	5	(null)
BHIT	NUMBER(4,0)	Yes	0	6	(null)
BGROUP	NUMBER(4,0)	Yes	(null)	7	(null)
BSTEP	NUMBER(4,0)	Yes	(null)	8	(null)
BINDENT	NUMBER(4,0)	Yes	(null)	9	(null)

테스트를 위한 Dummy Data 입력

```
insert into mvc_board (bid, bName, bTitle, bContent, bHit, bGroup, bStep, bIndent)
values (mvc_board_seq.nextval, 'abod', 'is title', 'is content', 0, mvc_board_seq.currval, 0, 0);
```

부록_03. FrontController 만들기

❖ 클라이언트의 요청을 받는 역할을 하는 Controller를 만들어 봅시다.(BFrontController)

패키지 : com.javaEdu.ex.frontcontroller

클래스명 : BFrontController

```
private void actionPerformed(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    System.out.println("actionDo");

    request.setCharacterEncoding("EUC-KR");

    String viewPage = null;
    BCommand command = null;

    String uri = request.getRequestURI();
    String conPath = request.getContextPath();
    String com = uri.substring(conPath.length());

    if(com.equals("/write_view.do")) {
        viewPage = "write_view.jsp";
    }

    if(com.equals("/write.do")) {
        command = new BWriteCommand();
        command.execute(request, response);
        viewPage = "list.do";
    }

    RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);
    dispatcher.forward(request, response);
}
```

웹 브라우저에서 글쓰기 화면입니다.

'write.do' 요청이 들어오면 해당 Command를 생성하여 적절한 로직을 실행 후 'list.do' 페이지로 포워딩 합니다.

부록_03. Command 만들기

❖ Command 인터페이스를 이용해서 Command 클래스들을 만듭니다.

패키지 : com.javaEdu.ex.command
Command 인터페이스 : BCommand
Write Command 클래스 : BWriteCommand

```
1 package com.javaEdu.ex.command;
2
3 import javax.servlet.http.HttpServletRequest;
4
5 public interface BCommand {
6     void execute(HttpServletRequest request, HttpServletResponse response);
7 }
8
9
10
```

```
@Override
public void execute(HttpServletRequest request, HttpServletResponse response) {
    // TODO Auto-generated method stub

    String bTitle = request.getParameter("bTitle");
    String bName = request.getParameter("bName");
    String bContent = request.getParameter("bContent");

    BDao dao = new BDao();
    dao.write(bTitle, bName, bContent);
}
```

데이터 베이스와 연결하여 사용자가 입력한 내용을 DB에 Insert 합니다.

부록_03. DTO(Data Transfer Object) 만들기

❖ 데이터 베이스의 데이터 DTO객체를 만듭니다.

- 패키지 : com.javaEdu.ex.dto
- DTO 클래스 : BDto

```
public BDto(int bld, String bName, String bTitle, String bContent, Timestamp bDate, int bHit, int bGroup, int bStep, int blndent) {
    // TODO Auto-generated constructor stub
    this.bld = bld;
    this.bName = bName;
    this.bTitle = bTitle;
    this.bContent = bContent;
    this.bDate = bDate;
    this.bHit = bHit;
    this.bGroup = bGroup;
    this.bStep = bStep;
    this.blndent = blndent;
}
```

생성자 및 속성 설정

```
public int getbld() {
    return bld;
}

public void setbld(int bld) {
    this.bld = bld;
}
```

Getter,
Setter

부록_03. DAO(Data Access Object) 만들기

❖ 데이터 베이스에 연결하여 필요한 로직을 수행하는 DAO클래스를 만듭니다.

- ✓ 패키지 com.javaEdu.ex.dao
- ✓ DTO 클래스 : BDao

```
public BDao() {  
    // TODO Auto-generated constructor stub  
  
    try {  
        Context context = new InitialContext();  
        dataSource = (DataSource) context.lookup("java:comp/env/jdbc/Oracle11g");  
    } catch (Exception e) {  
        // TODO: handle exception  
        e.printStackTrace();  
    }  
}  
  
public void write(String bName, String bTitle, String bContent) {  
    // TODO Auto-generated method stub  
  
    Connection connection = null;  
    PreparedStatement preparedStatement = null;  
  
    try {  
        connection = dataSource.getConnection();  
        String query = "insert into mvc_board (bId, bName, bTitle, bContent) values (1, ?, ?, ?)";  
        preparedStatement = connection.prepareStatement(query);  
        preparedStatement.setString(1, bName);  
        preparedStatement.setString(2, bTitle);  
        preparedStatement.setString(3, bContent);  
        int m = preparedStatement.executeUpdate();  
    }  
}
```

← 생성자에서 DBCP를 만듭니다.

← DBCP로부터 Connection을 얻고, 데이터 베이스와 관련한 필요한 작업을 시작합니다.

부록_03. View 페이지 만들기

- ❖ 클라이언트의 요청에 대해서 FrontController에서 작업을 분기하고, 해당 Command클래스가 작동하여 DAO를 이용한 데이터 베이스 작업을 합니다.
- ❖ DAO클래스의 결과물로 DTO객체가 View(.jsp페이지)로 전달되며, View에서는 클라이언트의 요청에 대한 응답으로 화면(UI)를 구성하여 출력 합니다.

View 파일 : list.jsp

```
<table width="500" cellpadding="0" cellspacing="0" border="1">  
    <tr>  
        <td>번호</td>  
        <td>이름</td>  
        <td>제목</td>  
        <td>날짜</td>  
        <td>히트</td>  
    </tr>  
    <:forEach items="${list}" var="dto">  
        <tr>  
            <td>${dto.bId}</td>  
            <td>${dto.bName}</td>  
            <td>${dto.bTitle}</td>  
            <td>${dto.bDate}</td>  
            <td>${dto.bHit}</td>  
        </tr>  
    </forEach>  
</table>
```



번호	이름	제목	날짜	히트
1	abcd	is title	2021-01-10 10:33:29.0	0