Java JDBC & Java FX 프로젝트 보고서

**수입 및 지출 관리 프로그램**

|  |  |
| --- | --- |
| 팀명 | Oteam |
| 팀원 | 김대영 |
| 김종원 |
| 김준영 |
| 박주영 |

**목 차(쪽수추가필요)**

**Ⅰ.** **프로젝트 개요**

1. 배경 및 목적 1
2. 프로그램 소개 2
3. 기대효과 2

**Ⅱ.** **프로젝트 방법**

1. 개발 환경 1
2. 프로젝트 계획 2

**Ⅲ. S/W 설계**

1. DB 설계 1
2. 패키지 구성 및 클래스 설명 2
3. 기능 요구사항 2
4. 실행 결과 2

**Ⅰ. 프로젝트 개요**

1. **배경 및 목적**

1) 배경

한정된 자원을 가지고 다양한 욕구 충족을 해야 하는 상황에서 자원의 선택적 배분 및 욕구의 우선순위 설정 등에서 의사결정 능력의 중요성이 강조되고 있다. 생활목표의 달성을 위한 다양한 방법 중에 효과적인 방법의 선택 문제 등에 직면해 있으며 또한 보다 나은 생활의 실현, 생활을 충실하게 하기 위해 의식적인 결정과 실행이 요구되고 있다.

또한 경제성장이 둔화되고 있고 고령화가 가속화되며 가계재무관리의 중요성은 더욱 커지고 있다. 재무관리의 중요한 내용 중의 하나는 예산을 세우고 기록하고 결산을 하는 것이다. 이 때 이용되는 도구가 가계부이다. 가계부는 단기적으로는 소비지출을 기록하여 예산에 맞추어 제대로 소비하였는지를 평가하는데 이용될 수 있으며 장기적으로는 재무 목표를 달성하는 것을 목적으로 한다.



자료1. 가계부 소득 관리 도움 평가 (엠브레인트레인모니터, 2014)

최근 물가 상승으로 인해 많은 사람들이 경제적으로 부담을 느끼고 있다. 상승한 물가에 비해 나의 수입을 늘리는 것이 쉽지 않은 상황이다. .



자료2. 2022년 3분기 지역별 소비자물가 (통계청, 2022-11-21)

(자료. 김성기, 2022년 3분기 소비자물가, CBS 노컷뉴스, 2022-11-21, https://han.gl/AqhEF)

통계청이 발표한 '2022년 3분기 지역경제동향'에 따르면 3분기 전국 소비자물가는 1년 전보다 5.9% 상승했다. 이는 3분기 기준으로 1997년(7.0%) 이후 24년 만에 최고 상승 폭이다.

2) 가계부 작성의 효과

시장조사전문기관 엠브레인트렌드모니터(trendmonitor.co.kr)에 따르면 가계부 작성 비율은 그리 높지 않았으나 가계부의 효용성에 대해서 높게 평가되고 있음을 알 수 있다. 전체 64.7%가 가계부 작성이 소득을 효율적으로 관리하는 데 도움이 된다고 응답하였다. 가계부 작성이 도움이 되는 가장 큰 이유는 지출 내역 및 규모의 파악이 가능하다는 점(64%) 이다. 또한 소비 습관의 파악이 가능하고(51%), 불필요한 지출을 방지할 수 있으며(41.9%), 지출계획을 초과했는지의 여부를 알 수 있다는 점(34.2%)도 가계부의 작성 장점으로 꼽혔다.

이처럼 가계부는 소비절제를 위한 좋은 방법 중 하나로 여겨진다. 이는 어린이나 청소년기의 금융교육이 출발점으로 용돈기입장을 작성하도록 하는 이유이기도 하며, 연말이 되면 서점에서 가계부가 중요 품목으로 판매되고 소비자들은 새해에 대한 다짐으로 가계부를 구매하는 이유이기도 하다.

3) **현재 출시된 가계부 서비스의 단점**

최근에는 신용카드 및 체크카드의 지출 내역을 입력하여 소비관리를 도와주는 스마트폰 가계부 어플리케이션을 사용하는 사람들이 늘어나고 있다. 이에 따라 뱅크샐러드, 토스 등 많은 핀테크에서 \*가계부 서비스 를 제공하고 있고, 최근에는 자산관리 서비스를 출시하는 은행들에서도 가계부 기능을 연달아 내놓고 있다.

스마트폰 인터넷 환경이 확산되고 있지만 디지털 전략이 강화될수록 고령층이 받는 차별이 심화되고 있다. 시장조사 전문기관 제니스에 따르면 우리나라 성인 스마트폰 보급률은 84.2%인 것으로 조사되었으며 65세 이상 고령층은 스마트폰을 가지고는 있지만 실질적으로 스마트폰의 기능을 충분히 활용하지 못하고 있음을 알 수 있다. 이는 과다한 기능 추가로 복잡화된 인터페이스로 앱을 사용하기 어려워 디지털 사각지대 속에서 차별받는 사용자들이 늘어감을 알 수 있다.

\* 가계부 서비스 : 수입, 지출 등 거래 내역을 모아주고, 소비(예산) 관리 등을 사용자에게 제공하는 서비스

**2. 프로그램 소개**



1) 프로그램명 : Ojam

수입과 지출을 기록하여 사용자에게 지출 관리를 제공하는 ‘가계부 서비스’

2) 프로그램 기능

- 기록된 수입과 지출 내역으로 소비 패턴 파악

- 항목별 지출 비율과 월별 지출 추이를 한 눈에 볼 수 있는 통계 데이터

- 기간 동안의 소비와 지출의 데이터를 보여주고, 그에 따른 현재 자산 확인

3) 기대효과

- 복잡하지 않은 인터페이스

모든 연령층이 쉽게 이용 가능직관적인 인터페이스로 소비/지출 추이, 항목 별 소비 비중, 총 내 자산 현황을 보기 쉬움

* 기본에 충실한 기능 중심으로 구현

수기로 작성하는 전통적 가계부를 작성하는 사용자의 접근 용이할것으로 기대

**Ⅱ.** **프로젝트 방법**

**1. 개발 환경**

1) 개발 언어 : Java

2) 개발 환경 : Windows 10 64bit

3) 실행 환경 : Web Browser

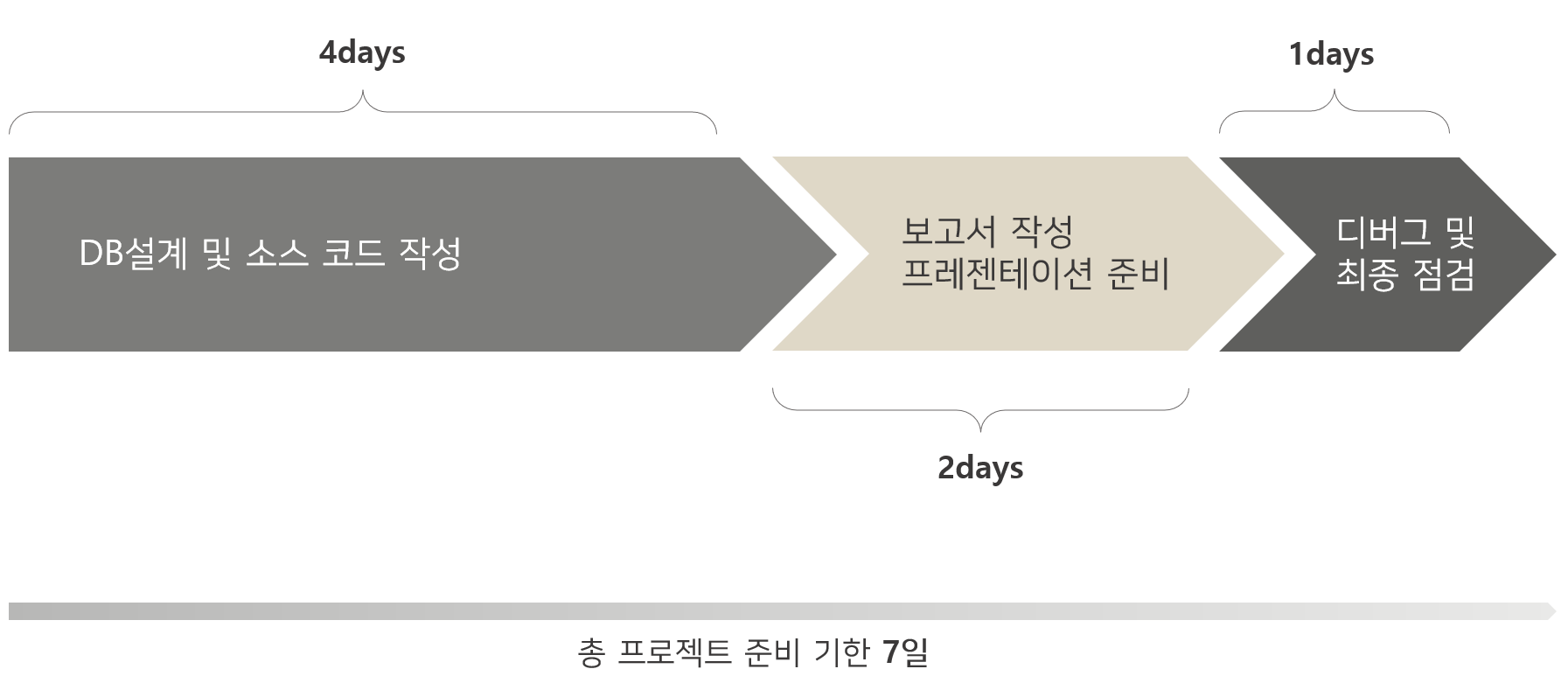
4) 개발 툴 : eclipse, sqldeveloper(?)

5) 기술 스택: java 8, Oracle 18c xe, JavaFX

**2. 프로젝트 계획**

1) 프로젝트 총 기한 : 2022년 11월 18일 ~ 2022년 11월 24일

2) 프로젝트 발표 시간 : 2022년 11월 24일 15시 00분

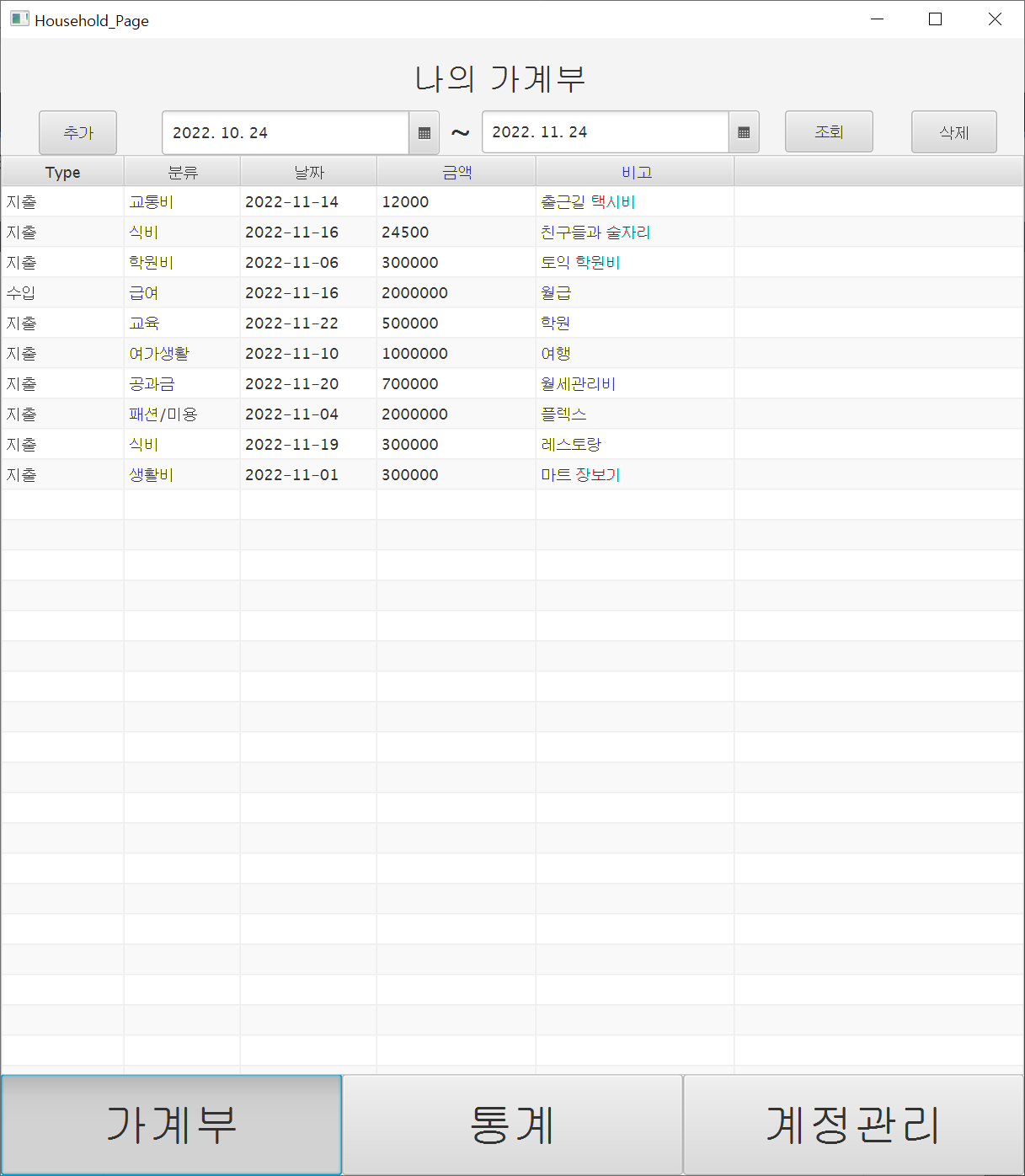


**2. 기능 설계**

1) 가계부 기능

가계부 기능은 기본적인 가계부 기능을 구현하였다. 사용자는 전통적인 가계부를 작성하는 것과 같이 사용자가 직접 지출 분류를 선택하고 금액과 지출 내용을 입력하여 날짜 선택 후 저장한다.

* 사용자 가계에 대한 데이터베이스 관리
* 소비에 따른 조회와 입력, 삭제 기능 제공



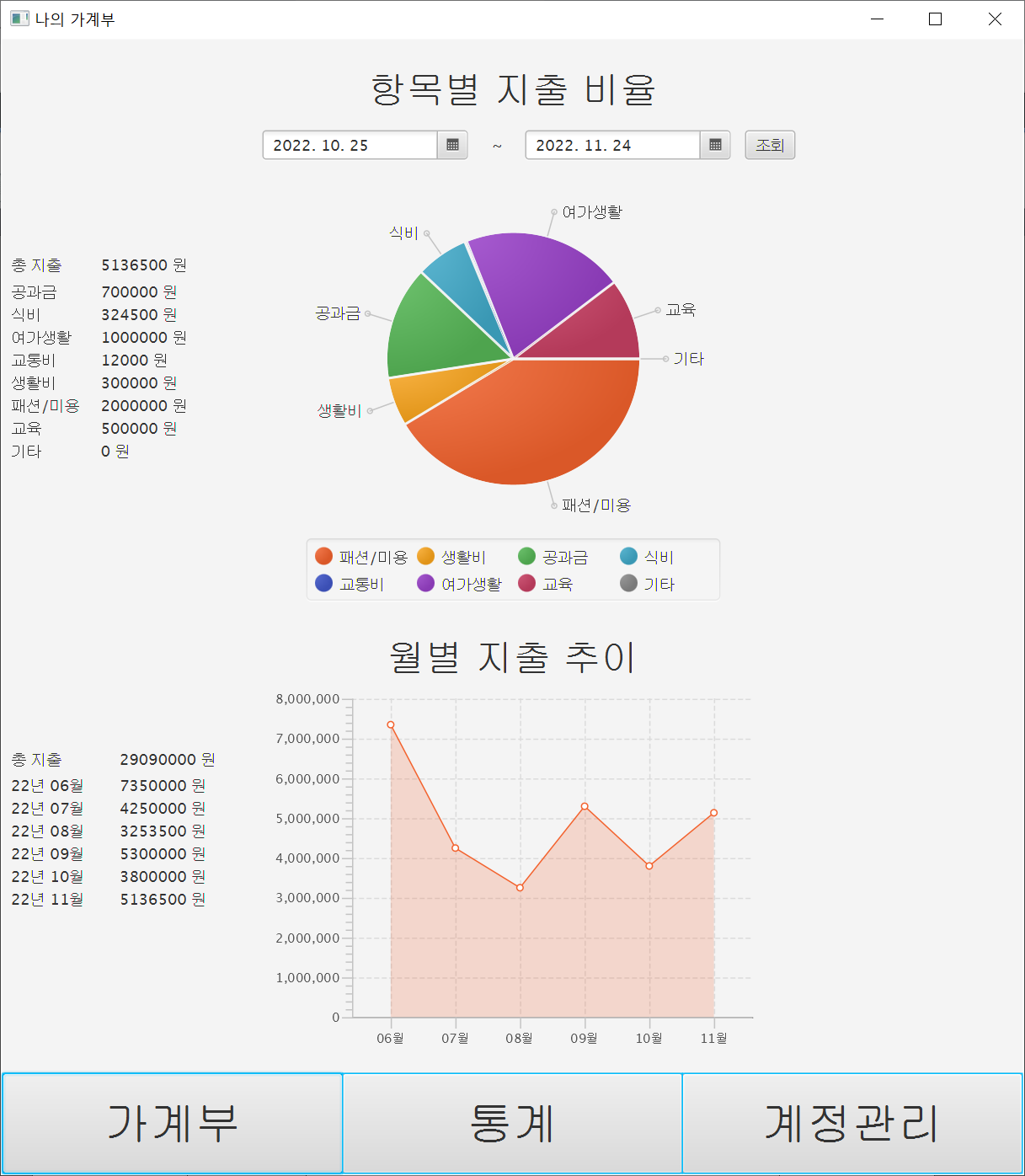
(그림1. 가계부 페이지)

일정 기간 동안의 기간을 설정하면 해당 기간 동안의 수입과 지출 내역을 조회해볼 수 있다. 그림1은 기본적인 가계부 입력 화면의 예이다.

2) 통계 기능

항목별 지출 통계 내역을 차트로 시각화 하여 사용자로 하여금 한 눈에 지출 현황을 확인할 수 있으며, 설정한 기간 동안의 가계부를 조회하여 시각화한 데이터를 확인하여 지출 내역에 따른 의미를 찾기에 용이하도록 하였다.

* 항목별 지출 비율 파악 가능
* 파이차트로 시각화 된 월별 지출 추이를 한 눈에 볼 수 있도록 구성



(그림2. 통계 페이지)

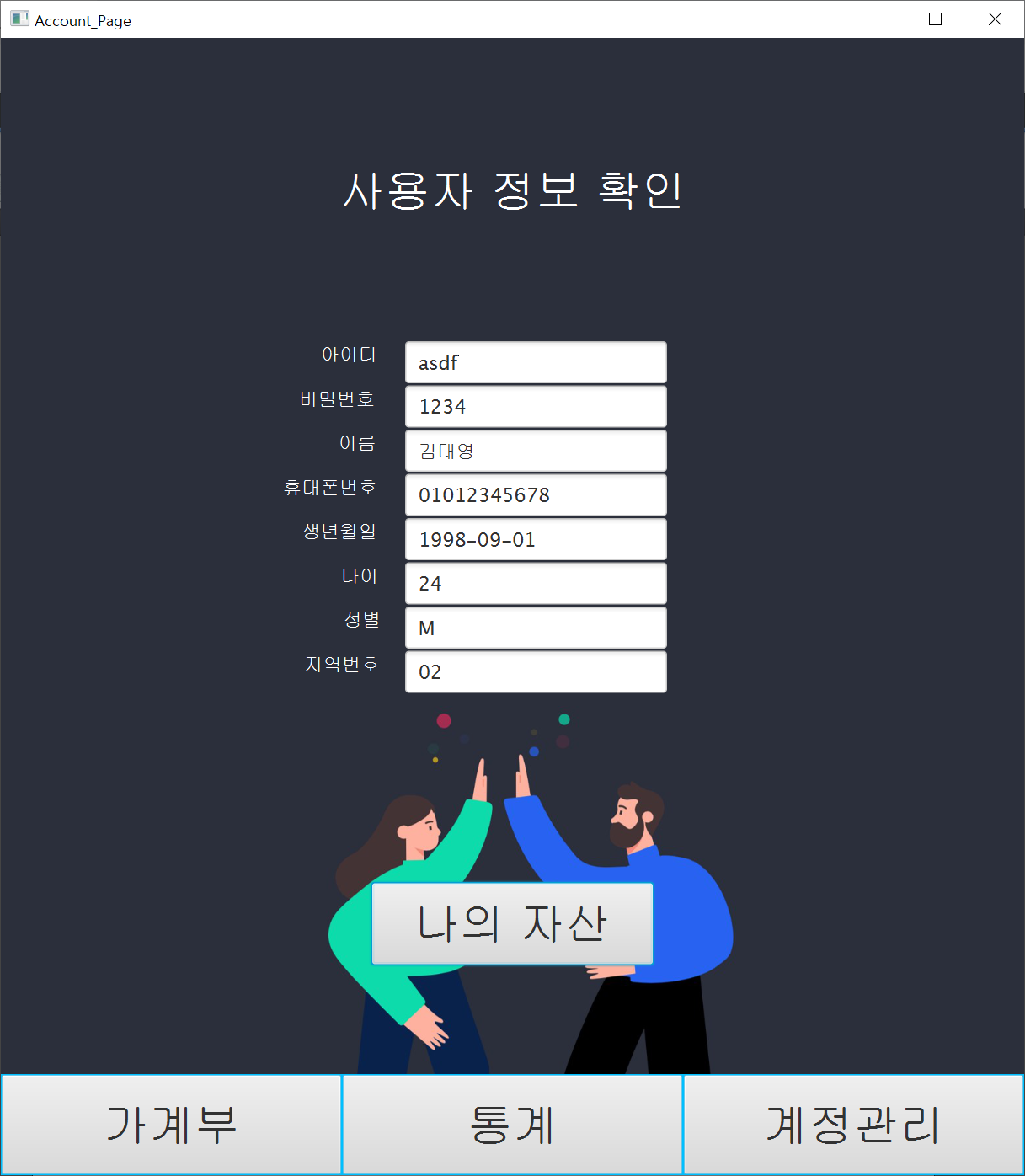
그림2는 임의로 설정한 기간동안 조회한 통계 데이터이다. 파이 차트를 통해서 설정한기간 동안의 지출 항목 비율을 알 수 있으며, 선 그래프를 통해서는 최근 6개월의 지출 추이를 한눈에 확인할 수 있다.

* 통계 화면 카테고리 항목 구성

|  |  |
| --- | --- |
| **항목** | **내용** |
| 공과금 | 통신비, 관리비, 가스비, 보험료 등 |
| 식비 | 식료품비, 외식비, 배달음식 등 |
| 교통비 | 대중교통, 주유비, 톨비 등 |
| 생활비 | 생활용품, 반려동물 등 |
| 여가생활 | 운동, 문화생활 등 |
| 패션/미용 | 의복, 미용잡화 등 |
| 교육 | 학원, 학습지 등 |
| 기타 | 모임/회비, 병원비, 경조사비 등 |

(표1. 통계 카테고리 항목)

3) 계정 정보 확인 기능



(그림3. 계정 관리페이지)

사용자가 회원가입 당시 기입한 개인 정보 확인이 가능하다.

* 사용자 개인 정보 확인 기능
* 나의 자산을 확인하는 페이지로 이동

4) 자산 조회 기능

****

(그림4. 자산 페이지)

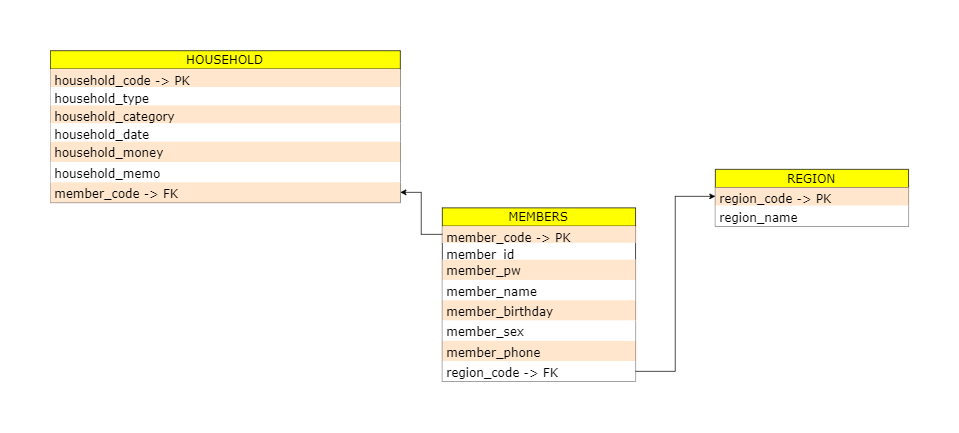
사용자가 입력한 최초의 가계부 날짜와 입력 당시 날짜까지 기간을 설정하여 사용자가 입력한 최초의 가계부 날짜와 입력 당시 날짜까지 기간을 설정하여 총수입과 지출 그리고 잔여 자산을 확인할 수 있도록 하였다.

* 총 수입/지출 및 잔여 자산 조회
* 계정 관리 페이지로 돌아가는 뒤로가기 기능

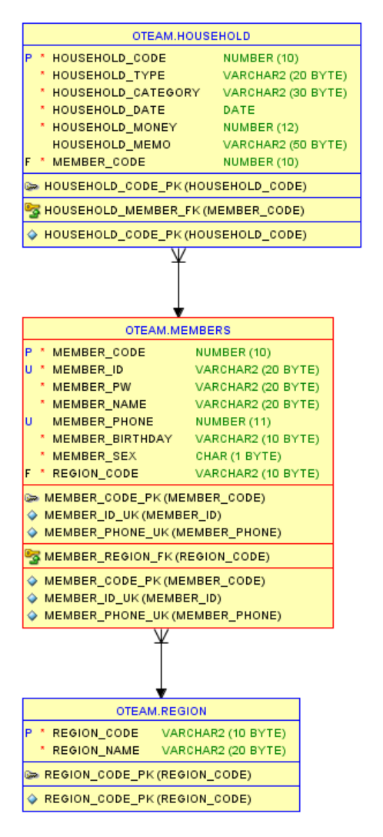
**Ⅲ. S/W 설계**

**1. DB설계**

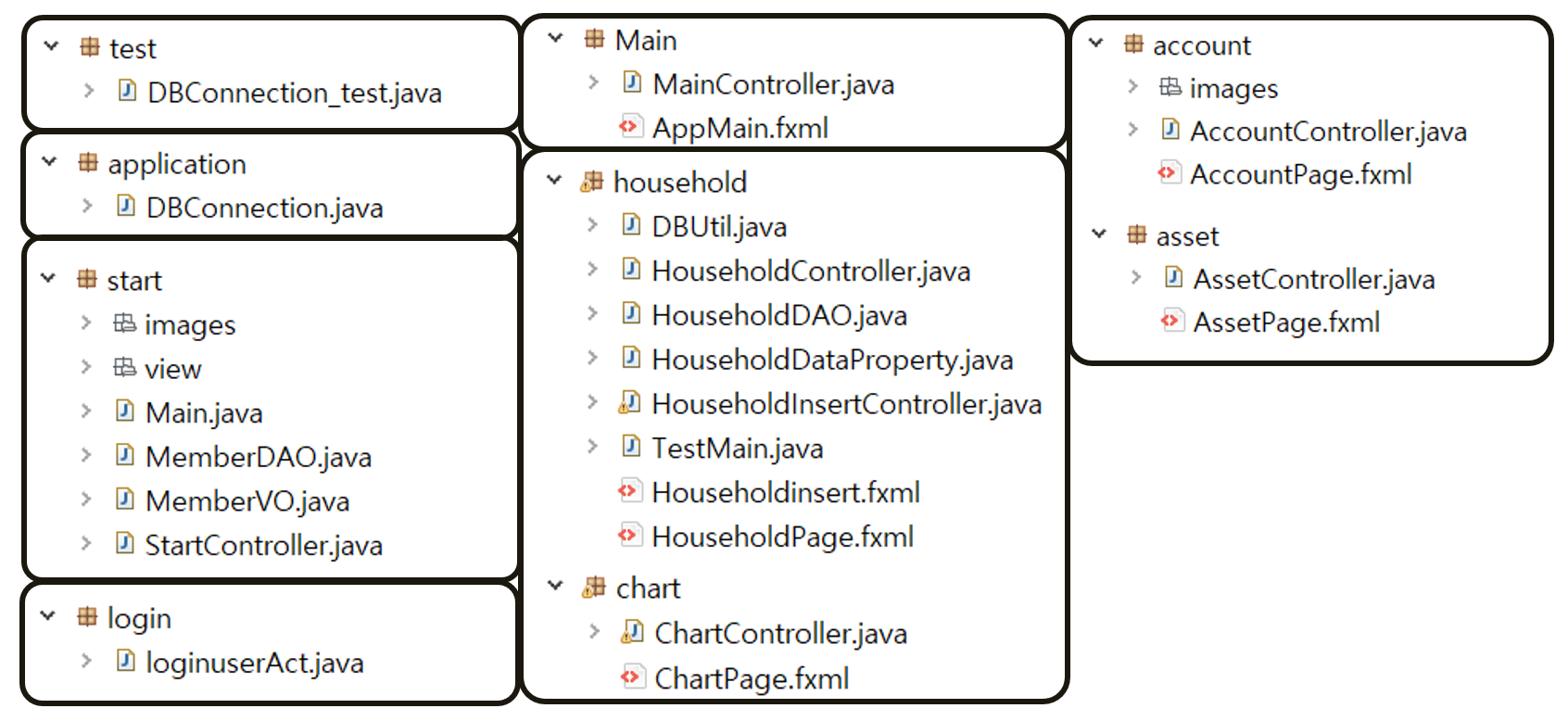
1) 테이블 논리모델



2) 테이블 관계모델



2. 패키지 구성 및 클래스 설명



**- test Package**

****

* DB 연결 상태를 테스트하기 위한 파일을 담은 패키지

**DBConnection\_test.java**

: 프로젝트 실행 전 DB연결상태를 확인할 때 사용한다.

main메서드에서 접속할 오라클 DB의 드라이버와 URL정보를 넣고 프로젝트에서 사용하기위해 새로 만든 사용자(oteam)로 연결 상태를 확인한다.

**- application Package**

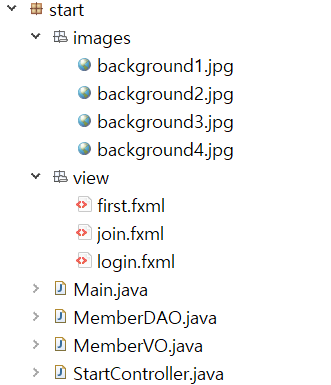


* 프로그램의 DB 연결을 담당하는 파일을 담은 패키지

**DBConnection.java**

: 프로젝트 lib폴더의 oracle.properties 파일을 읽어와 driver, url, user, password에 대한 값을 받고, DB에 연결한다.

**- start Package**

****

* 프로그램 첫 실행 화면, 로그인, 회원가입 화면 및 기능에 대한 모든 소스코드를 담은 패키지

**Images 폴더**

: 프로그램 첫 실행 화면, 로그인 화면, 회원가입 화면에 사용된 배경 이미지들을 담아둔 폴더

**View 폴더**

: 프로그램 첫 실행 화면, 로그인 화면, 회원가입 화면에 사용된 FXML 파일들을 담아둔 폴더

First.fxml: 프로그램 실행 시 등장하는 첫 화면

Login.fxml: 로그인을 진행하는 화면

Join.fxml: 회원가입을 진행하는 화면

**Main.java**

: 프로그램 첫 실행을 시작하는 클래스

실행을 하면 view 폴더의 first.fxml 파일을 호출한다.

**MemberVO.java**

: Members 테이블의 column을 필드로 정의하고, lombok을 사용하여 별도의 코드 작성 없이 getter/setter 메서드 및 생성자 사용이 가능한 클래스

**MemberDAO.java**

: 로그인, 회원가입, 계정 정보 조회에 필요한 동작을 메서드로 구현한 클래스

**public** **static** **void** JoinInsert(MemberVO mVO)

* 사용자가 회원가입을 진행할 때 호출되는 메서드
* DB의 members 테이블 제약조건을 위배하지 않는다면 정상적으로 사용자 데이터가 추가되며, 회원가입 완료 문구가 출력된다.
* 이 때, 실행되는 쿼리는 아래와 같다.

"INSERT INTO

MEMBERS(member\_code,member\_id,member\_pw,member\_name, member\_phone, member\_birthday, member\_sex, region\_code)

VALUES(member\_seq.nextval,?,?,?,?,?,?,?)";

* 만약 members 테이블의 제약조건을 위배하여 SQLException을 발생시킨다면 경고창과 문구를 출력하고, 회원가입이 진행되지 않는다.

**public** **static** MemberVO MyPage(**int** mem\_code)

* 현재 사용자의 member\_code를 입력 매개변수로 받아 계정 정보 조회를 실행하는 메서드
* 앱의 메인 실행 화면에서 하단의 계정 관리 버튼을 누를 때 호출된다.
* Members 테이블에서 현재 사용자의 모든 정보를 불러오는 쿼리는 아래와 같다.

"SELECT \* FROM MEMBERS WHERE member\_code = ?"

**StartController.java**

: 로그인, 회원가입 화면의 기능 및 동작을 구현한 클래스

**public** **void** moveUser()

* 첫 화면에서 로그인 버튼을 눌렀을 때 로그인 페이지로 이동하는 동작을 구현한 메서드
* View 폴더의 login.fxml 파일을 호출하여 로그인 화면으로 이동한다.

**public** **void** UserLogin(ActionEvent event)

* 로그인 버튼을 눌렀을 때 상황에 맞는 동작을 하도록 구현한 메서드
* 로그인 시도와 관련한 경고창 출력을 제어한다.
* 아이디 또는 비밀번호가 입력되지 않았을 경우, 입력한 아이디 또는 비밀번호가 DB에 존재하지 않는 경우 각각의 상황에 맞는 경고창을 출력한다.
* 올바르게 로그인이 진행 된 경우, loginuserAct 패키지의 userupdate 메서드에 현재 접속된 사용자의 member\_id를 넘겨준다.
* Main 패키지의 AppMain.fxml 파일을 호출하여 앱의 메인 실행 화면으로 이동한다.

**public** **void** goUserSignUp()

* 회원가입 버튼을 눌렀을 때 회원가입 페이지로 이동하는 동작을 구현한 메서드
* View 폴더의 join.fxml 파일을 호출하여 회원가입 화면으로 이동한다.

**public** **void** submitUserSignup(ActionEvent event)

* 회원가입 페이지의 가입 완료 버튼을 눌렀을 때 동작을 구현한 메서드
* Members테이블에서 Unique 제약조건을 설정한 member\_id, member\_phone의 중복값 에러를 제외한 나머지 제약조건을 모두 java의 조건문으로 제어했다.
* 위 조건문으로 모든 제약조건을 만족했다면 MemberDAO클래스의 JoinInsert메서드에 입력받은 사용자 정보를 넘겨준다.
* 회원가입이 완료되면 view 폴더의 login.fxml 파일을 호출하여 자동으로 로그인 화면으로 돌아가도록 설정했다.

**public** **void** cancelAction(ActionEvent event)

* 회원가입 버튼을 눌렀을 때 회원가입 페이지로 이동하는 동작을 구현한 메서드
* View 폴더의 join.fxml 파일을 호출하여 회원가입 화면으로 이동한다.

**public** **void** AlertError(String HeaderText, String ContentText)

* 각 예외처리에서 경고창을 출력하는 경우가 많아 코드 리팩토링을 위해 생성한 메서드
* 입력 매개변수로 HeaderText와 ContentText를 입력하면 자동으로 입력한 문구가 들어간 경고창을 출력한다.

**- login Package**



* 로그인 동작과 관련하여 필요한 기능을 구현한 파일을 담은 패키지

**loginuserAct.java**

: 사용자가 로그인을 시도했을 때 각 상황에 맞는 동작을 구현한 클래스

**public** **static** **void** userupdate(String id)

* 사용자가 로그인을 성공했을 때 호출되는 메서드
* 입력 매개변수로 로그인 당시 사용자가 입력했던 member\_id를 받고, 아래의 쿼리를 실행하여 현재 접속된 사용자의 member\_code를 받아온다.

**SELECT member\_code FROM MEMBERS WHERE member\_id = ?**

* 클래스에 전역변수로 선언해둔 member\_code를 받아온 값으로 할당해준다.
* 이를 통해 다른 패키지에서 loginuserAct클래스의 member\_code를 참조하여 사용자 인식을 할 수 있다.
* Member\_code를 받는 것 외에도 생년월일을 토대로 사용자의 나이를 반환하는 기능을 구현한 view를 사용하여 member\_age를 받아온다. 이는 계정관리 페이지에서 사용되는 값이다.

**public** **static** **int** loginCheck(String id, String pw)

* 사용자가 로그인을 시도했을 때 호출되는 메서드
* 로그인을 시도한 id와 pw를 아래의 쿼리문에 대입하고, DB에 존재하는 사용자라면 메서드 내 생성해둔 변수 check에 1이 할당된다.
* 만약 존재하지 않는 사용자라면 check값은 0이된다.
* check값을 return해주고, 이 메서드를 start패키지에서 로그인 동작 제어에 활용한다.

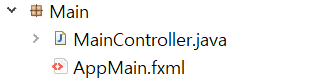
**public** **static** **int** whoIsMember()

* 현재 접속된 사용자의 member\_code를 반환해주는 메서드

**public** **static** **int** memberAge()

* 현재 접속된 사용자의 나이를 반환해주는 메서드

**- main Package**



* 로그인 이후 첫 등장하는 앱의 메인 실행화면과 관련된 기능을 구현한 패키지

**MainController.java**

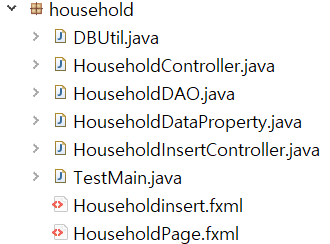
: 로그인 이후 등장하는 메인 페이지의 기능 및 동작을 구현한 클래스

**void** selectToggleButton(ActionEvent event)

* 하단의 토글버튼 동작을 구현한 메서드

AppMain.fxml : 로그인 이후 등장하는 앱의 첫 화면

**- household Package**

****

* 가계부 페이지의 기능을 구현한 패키지

HouseholdPage.fxml : 가계부 페이지의 화면

<BorderPane fx:controller="household.HouseholdController">

- fx:controller를 지정해 주어 BorderPane의 컨트롤을 자바 Class에서 컨트롤을 할 수 있게 된다.

<ToggleButton onAction="#household\_menu">

- onAction을 통해 자바에서 메서드를 만들어 기능을 부여할 수 있게 된다. (#필요)

<ToggleGroup fx:id="toggleGroup" />

- ToggleButton의 속성 중 하나로 다른 ToggleButton과 유기적으로 묶어 사용할 수 있다.

- 다른 ToggleButton에 toggleGroup="$group" 속성을 부여해 주어야 한다.

<DatePicker fx:id="start\_date" >

- fxml에 생성된 DatePicker의 값을 받아오기 위해 자바 Class에서 호출하기 위한 fx:id 속성 부여

<TableView fx:id="table\_view" onMouseClicked="#click\_tableview"

- TableView 목록 중 마우스 클릭으로 데이터를 가져오기 위해 onMouseClicked 속성을 부여

- <TableColumn fx:id="type\_column" text="Type" />으로 열 클래스 생성

HouseholdInsert.fxml : 가계부 페이지의 입력 창 화면

<ComboBox fx:id="combo1\_menu">

- ComboBox의 값을 받아오기 위해 자바 Class에서 호출하기 위한 fx:id 속성 부여

- <FXCollections fx:factory="observableArrayList">을 통해 목록<String fx:value="수입" />을 생성한다.

**HouseholdDataProperty.java**

: TableView의 데이터 DTO 기능을 구현한 클래스

**private** IntegerProperty Household\_Code;

* Property는 자바 FX의 데이터 타입

**public** HouseholdDataProperty()

- 생성자를 통해 한 행의 열 객체를 생성한다.

- this.Household\_Code = new SimpleIntegerProperty();

**public** **final** IntegerProperty Household\_CodeProperty()

- 자바 FX 타입의 데이터 getter

- return this.Household\_Code;

**public** **final** **int** getHousehold\_Code()

- TableView 데이터의 getter

- return this.Household\_CodeProperty().get();

**public** **final** **void** setHousehold\_Type(**final** String Household\_Type)

- TableView 데이터의 getter

- this.Household\_CodeProperty().set(Household\_Code);

**HouseholdController.java**

: 가계부 페이지 Fxml을 제어할 클래스

**public** **void** initialize(URL location, ResourceBundle resources)

- 페이지 실행 시 먼저 실행되는 메서드

- TableView의 열 타입을 지정해준 뒤 자바의 LocalDate.now() 이용하여 한 달간 사용 내역 출력한다.

- HouseholdDAO.searchHousehold()를 호출한다.

**private** **void** search\_act(ActionEvent actionEvent)

- 조회 버튼에 대한 기능 메서드

- HouseholdDAO.searchHousehold()를 호출한다.

- @FXML private DatePicker start\_date;를 통해 start\_date.getValue()로 값을 받는다.

**private** **void** insertmenu\_act(ActionEvent actionEvent)

- 추가 버튼에 대한 기능 메서드

- FXMLLoader.load(getClass().getResource("householdinsert.fxml")); 통해 입력창을 새로운 Scene에 넣어 새로운 Stage에 올려 창을 띄운다.

**private** HouseholdDataProperty click\_tableview(MouseEvent event)

- TableView의 행 클릭 이벤트에 대한 기능 메서드

- table\_view.getSelectionModel().getSelectedItem();로 선택된 행 데이터를 가져온다.

- HouseholdDataProperty 클래스에 저장한다.

**private** **void** delete\_act(ActionEvent actionEvent)

- 삭제 버튼에 대한 기능 메서드

- HouseholdDAO.deleteHousehold()를 호출한다.

- HouseholdDAO.searchHousehold()로 바로 반영된 결과를 출력한다.

- NullPointerException의 예외 처리(TableView 선택 안 함)로 자바 FX 에러창을 띄운다.

**private** **void** household\_menu(ActionEvent actionEvent)

**private** **void** start\_menu(ActionEvent actionEvent)

**private** **void** account\_menu(ActionEvent actionEvent)

- 가계부 페이지 하단의 각 페이지 버튼에 대한 기능 메서드

- Platform.runLater()를 이용하여 기존 창을 닫는다. [stage.close();]

- FXMLLoader.load(getClass().getResource("fxml 경로")); 통해 입력창을 새로운 Scene에 넣어 새로운 Stage에 올려 창을 띄움

**HouseholdInsertController.java**

: 가계부 페이지의 입력 Fxml을 제어할 클래스

**public** **void** initialize(URL location, ResourceBundle resources)

- 페이지 실행 시 먼저 실행되는 메서드

- combo1\_menu.setValue("수입"); 초기값을 설정해 주어 오류를 잡아주었다.

**private** **void** insert\_act(ActionEvent actionEvent)

- 입력 버튼에 대한 기능 메서드

- HouseholdDAO.insertHousehold()를 호출한다.

- Platform.runLater()를 이용하여 기존 창을 닫는다. [stage.close();]

**HouseholdDAO.java**

: DB에 접근하는 클래스

**public** **static** ObservableList<HouseholdDataProperty> searchHousehold()

- DB에 select에 대한 결과를 가져오는 메서드

- loginuserAct.whoIsMember()로 로그인 시 사용자 Member\_code를 가져온다.

- select에 대한 SQL 문을 작성한다.

- DBUtil.dbExecuteQuery()를 호출한다.

- while를 사용해 DB에 출력된 모든 행을 담는다.

while(rsSHH.next()) {

HouseholdDataProperty sHH = new HouseholdDataProperty();

sHH.setHousehold\_Type(rsSHH.getString("HOUSEHOLD\_TYPE"));

sHH.setHousehold\_Category(rsSHH.getString("HOUSEHOLD\_CATEGORY"));

sHH.setHousehold\_Date(rsSHH.getDate("HOUSEHOLD\_DATE"));

sHH.setHousehold\_Money(rsSHH.getInt("HOUSEHOLD\_MONEY"));

sHH.setHousehold\_Memo(rsSHH.getString("HOUSEHOLD\_MEMO"));

empData.add(sHH);

**public** **static** **void** insertHousehold()

- DB에 insert에 대한 결과를 가져오는 메서드

- 위의 searchHousehold()과 과정이 동일하며 DBUtil.dbExecuteUpdate(StmtIHH);의 호출과 void라는 차이가 있다.

- SQLException의 예외 처리(DB와 형식이 안 맞음)로 자바 FX 에러창을 띄운다.

**public** **static** **void** deleteHousehold(HouseholdDataProperty click\_tv)

- DB에 delete에 대한 결과를 가져오는 메서드

- 위의 insertHousehold()과 과정이 동일하다.

**DBUtil.java**

: Java에서 DB에 접속하기 위한 클래스

**public** **static** **void** dbConnect()

- private으로 제한된 정보(DB 접속 소스) 들에 접근하기 위한 메서드

- SQL 문 실행 메서드에서 호출한다.

**public** **static** ResultSet dbExecuteQuery(String sqlStmt)

- select SQL 문 실행에 호출되는 메서드

- dbConnect() 호출로 DB에 접속하게 된다.

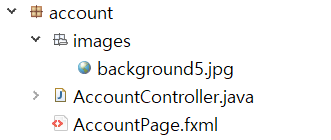
- TableView에 출력을 해야 하기에 ResultSet을 return 한다.

**public** **static** **void** dbExecuteUpdate(String sqlStmt)

- insert, delete SQL 실행에 호출되는 메서드

- dbConnect() 호출로 DB에 접속하게 된다.

**- account Package**



* 계정관리 페이지의 기능을 구현한 패키지

**AccountController.java**

: 계정관리 페이지의 기능 및 동작을 구현한 클래스

**void** selectToggleButton(ActionEvent event)

* 하단의 토글버튼 동작을 구현한 메서드

AppMain.fxml : 로그인 이후 등장하는 앱의 첫 화면

**- chart Package**

텍스트이(가) 표시된 사진

자동 생성된 설명

* 통계 페이지의 기능을 구현한 패키지

**ChartController.java**

: 통계 페이지의 기능 및 동작을 구현한 클래스

**public** **void** initialize(URL location, ResourceBundle resources)

* 통계 페이지의 화면을 구현한 메서드
* 항목별 지출 비율을 나타내는 차트로 원형 차트를 이용하였다.
* 원형 차트는 사용자가 데이터를 입력하기 전에는 기본으로 최근 30일 간의 통계를 나타내도록 하였으며, 30일 간의 수치라는 것을 사용자가 알 수 있도록 날짜 입력 란에 현재 날짜와 30일 전의 날짜를 보여주었다.
* 원형 차트는 사용자가 원하는 기간을 설정하고 통계를 조회할 수 있도록 구현하였다.
* 원형 차트는 2개의 입력 매개변수가 있으며 항목의 이름과 항목에 대한 수치를 받는다.
* 수치에 들어가는 값은 작성한 SQL문을 실행하여 얻은 데이터 값이다.
* 월별 지출 추이를 나타내는 차트로 영역 차트를 이용하였다.
* 영역 차트는 최근 6개월의 월별 지출액으로 나타냈으며, Initialize 메서드 내에서 구현하여 통계 페이지에 들어올 때 기본으로 화면에 나타난다.
* 영역 차트의 수치 값은 작성한 SQL문을 실행하여 얻은 결과를 사용했다.

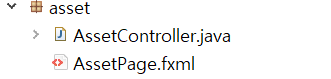
**public** **void** search\_act(ActionEvent actionEvent)

* 조회 버튼 클릭 시 작동하는 메서드
* DB와 연결을 해주고 로그인한 사용자를 특정할 수 있는 메서드를 호출한다.
* 그 후 사용자가 지정하는 시작날짜와 마지막날짜의 입력 값을 담은 변수를 설정하였다.
* 앞서 만든 변수를 SQL문과 결합하여 문장을 완성하였다.

**void** selectToggleButton(ActionEvent event)

* 하단 메뉴 버튼들의 동작을 구현한 메서드

**- asset Package**



* 총 수입과 지출 그리고 잔여 자산을 확인할 수 있도록 구현한 패키지

**AssetController.java**

: 자산 관리 페이지의 기능 및 동작을 구현한 클래스

**public** **void** initialize(URL arg0, ResourceBundle arg1)

* 계정 관리 페이지 이동시 값이 들어간 상태의 화면을 구현한 메서드
* 현재 사용자를 인식하기 위해 loginuserAct 클래스 파일의 whoIsMember메서드를 호출하여 member\_code를 받아온다.
* 이후 작성한 SQL문을 실행하여 사용자의 가계부 활동 기록을 바탕으로 총 지출, 총 수입을 화면에 출력한다.
* 총 자산은 수입에서 지출을 뺀 값으로 나타냈다.
* 상단의 날짜는 사용자가 가계부에 기록한 가장 과거의 날짜부터 실행 시점의 날짜를 출력한다.

AssetPage.fxml : 자산 관리 페이지 화면

**3. 기능 요구사항**

1) Database

- Oracle DB (11G OR 18CEx 0R Cloud)

- View를 사용

- 새로운 유저 생성

- 새로운 테이블 생성

2) JAVA

- java 8

- DAO[data access object] pattern

- JDBC

- ResultSet 사용

- PreparedStatement 사용

- JAVA FX

- JAVA FX Scene Builder

- FXML 레이아웃 사용

3) 코드 작성

- 패키지 2개 이상 사용

- 모든 java 소스에 주석 작성

- 코드작성자 표시

4) UI

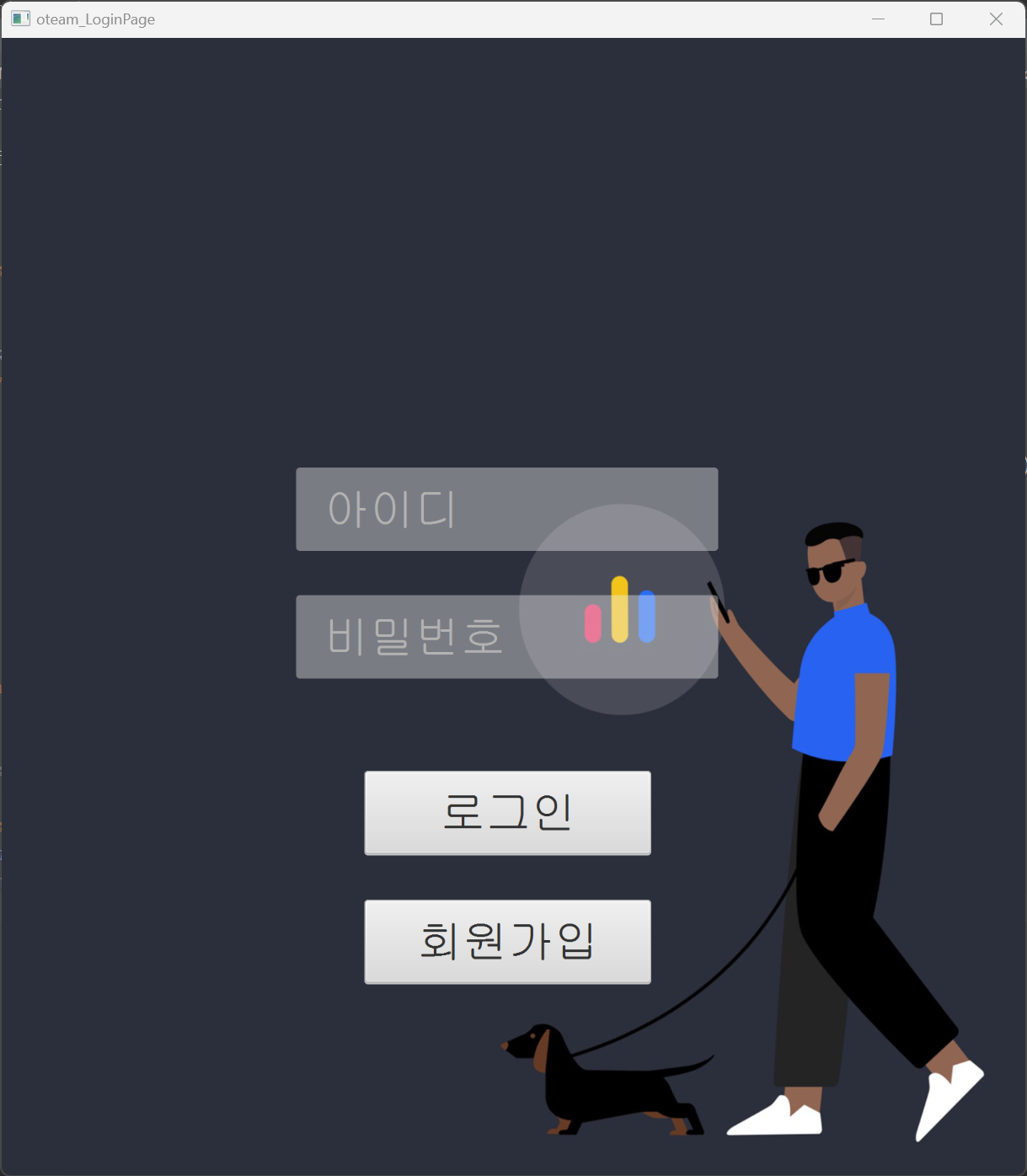
- 로그인, 회원가입 구현

- 추가 기능 페이지 구현

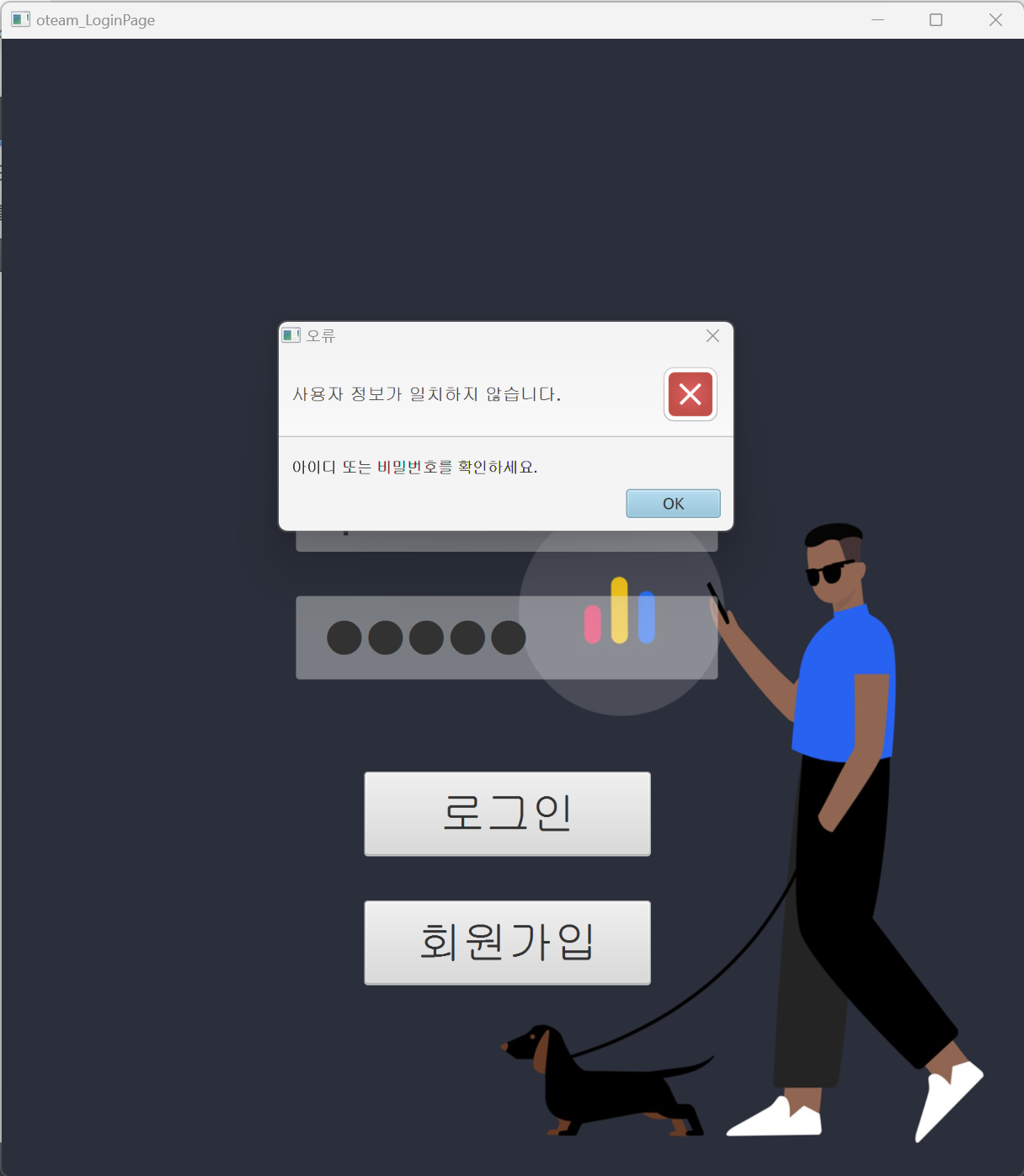
**4. 실행 결과**



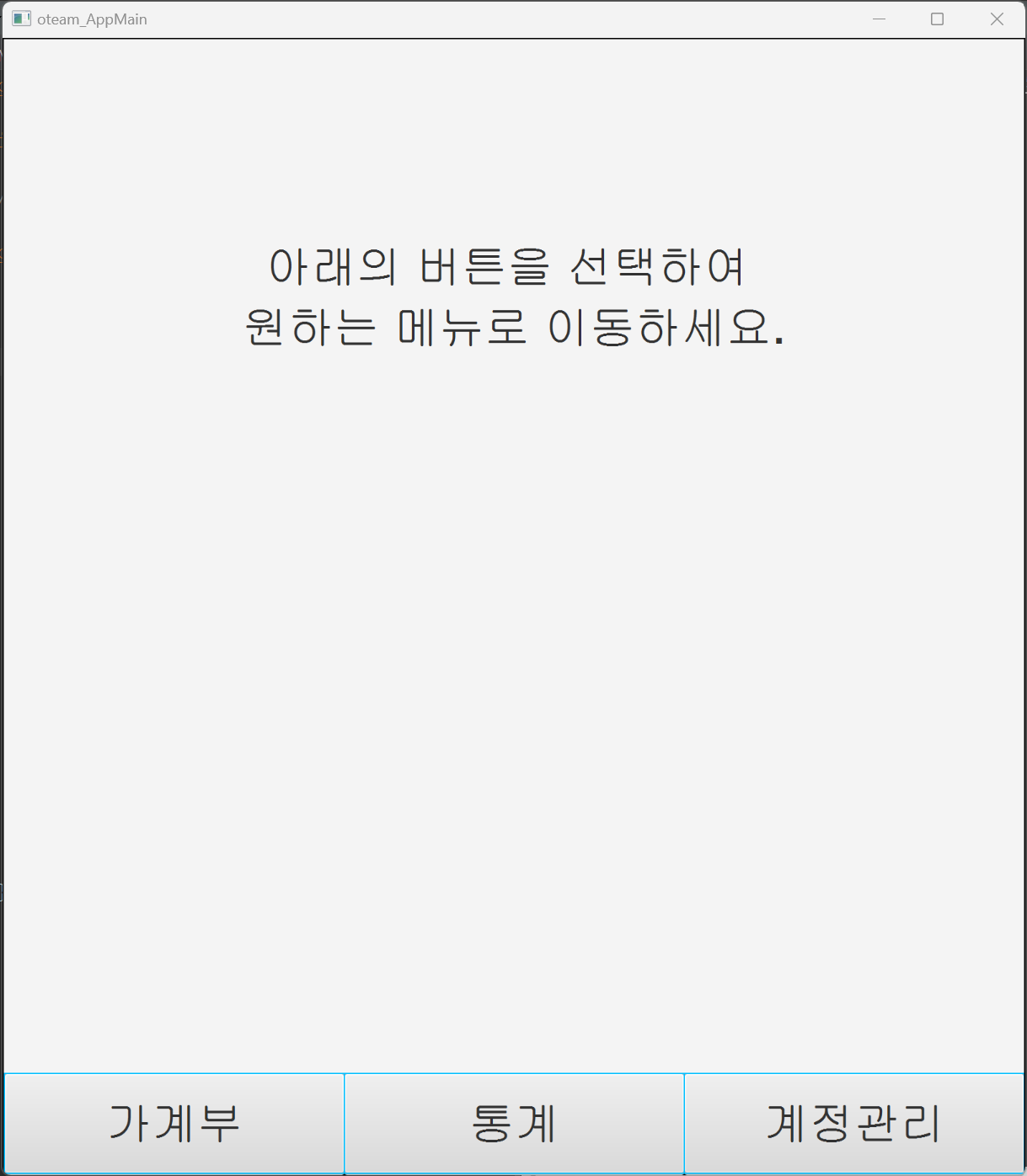
1) 시작화면

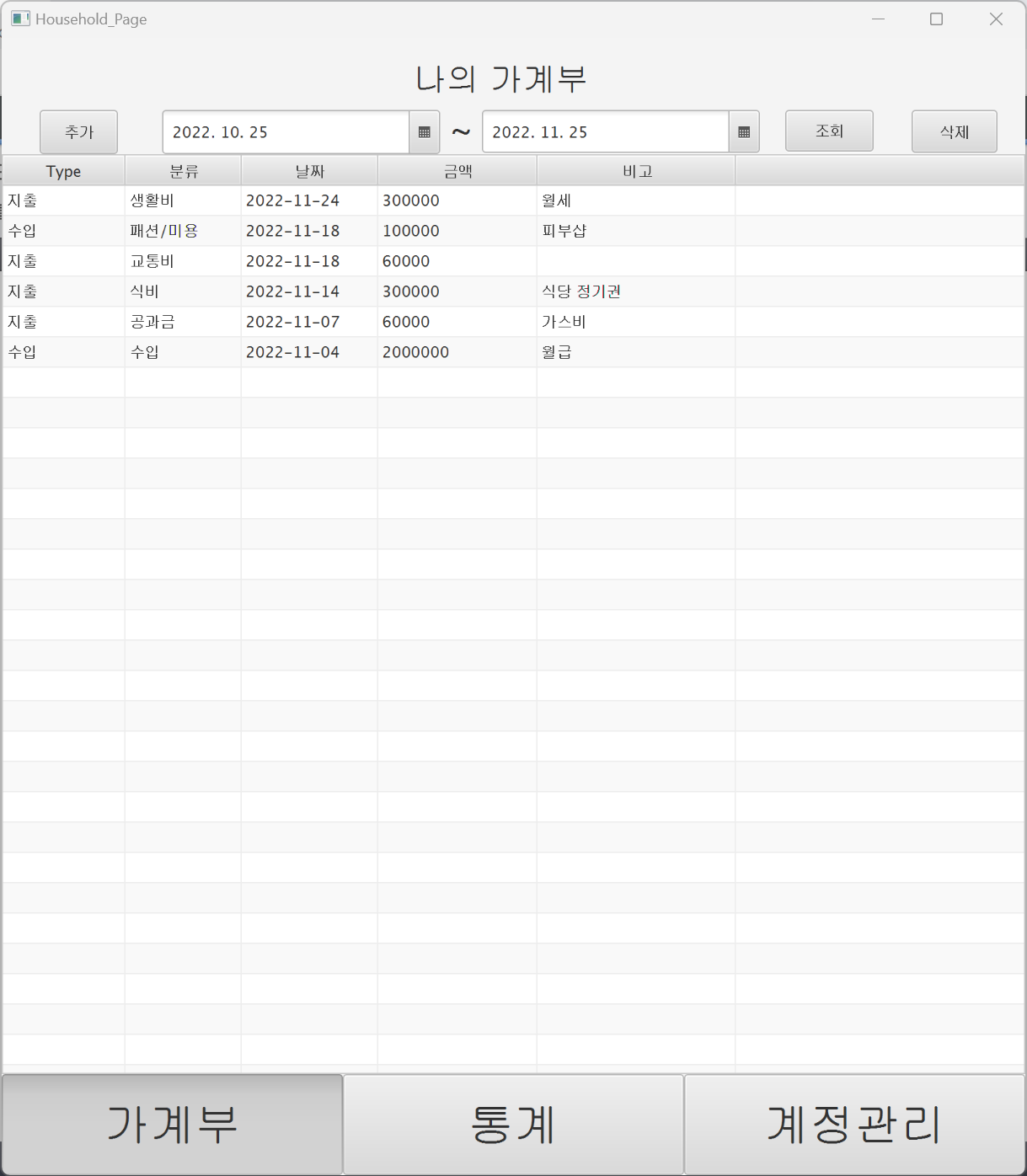
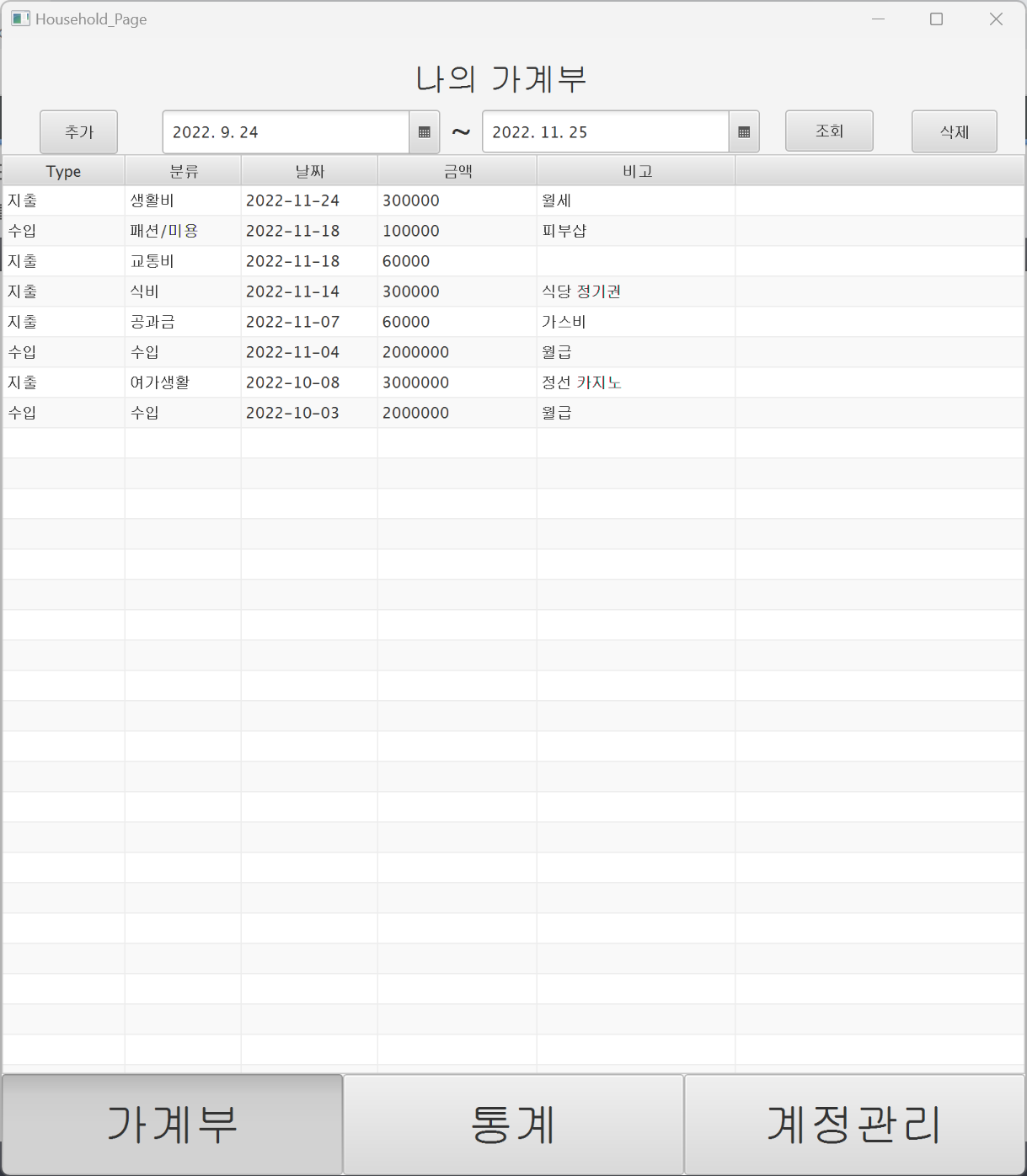
2) 로그인 화면 3) 회원가입 화면

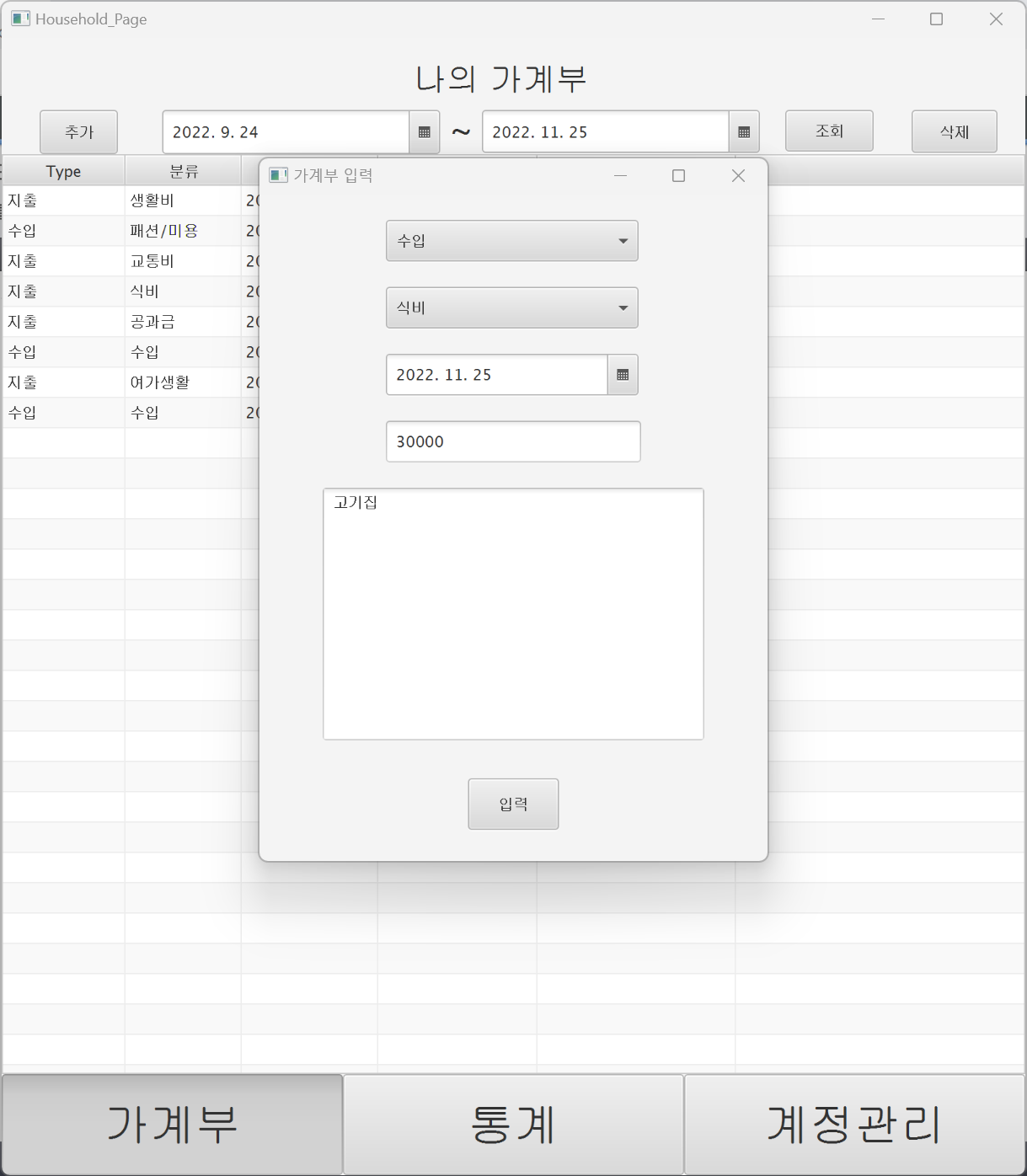
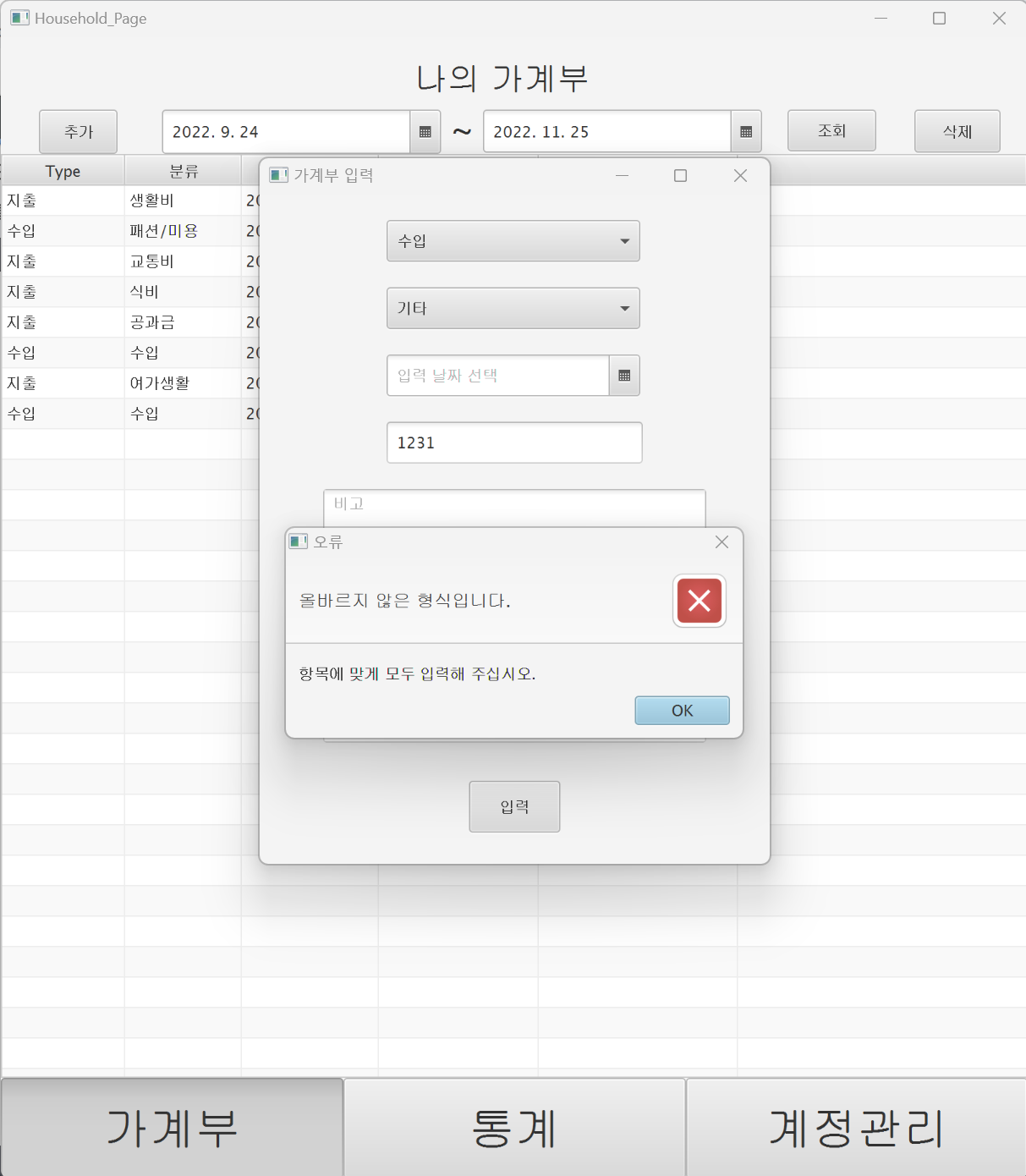
4) 아이디, 패스워드 누락 및 잘못 입력 시 오류 창

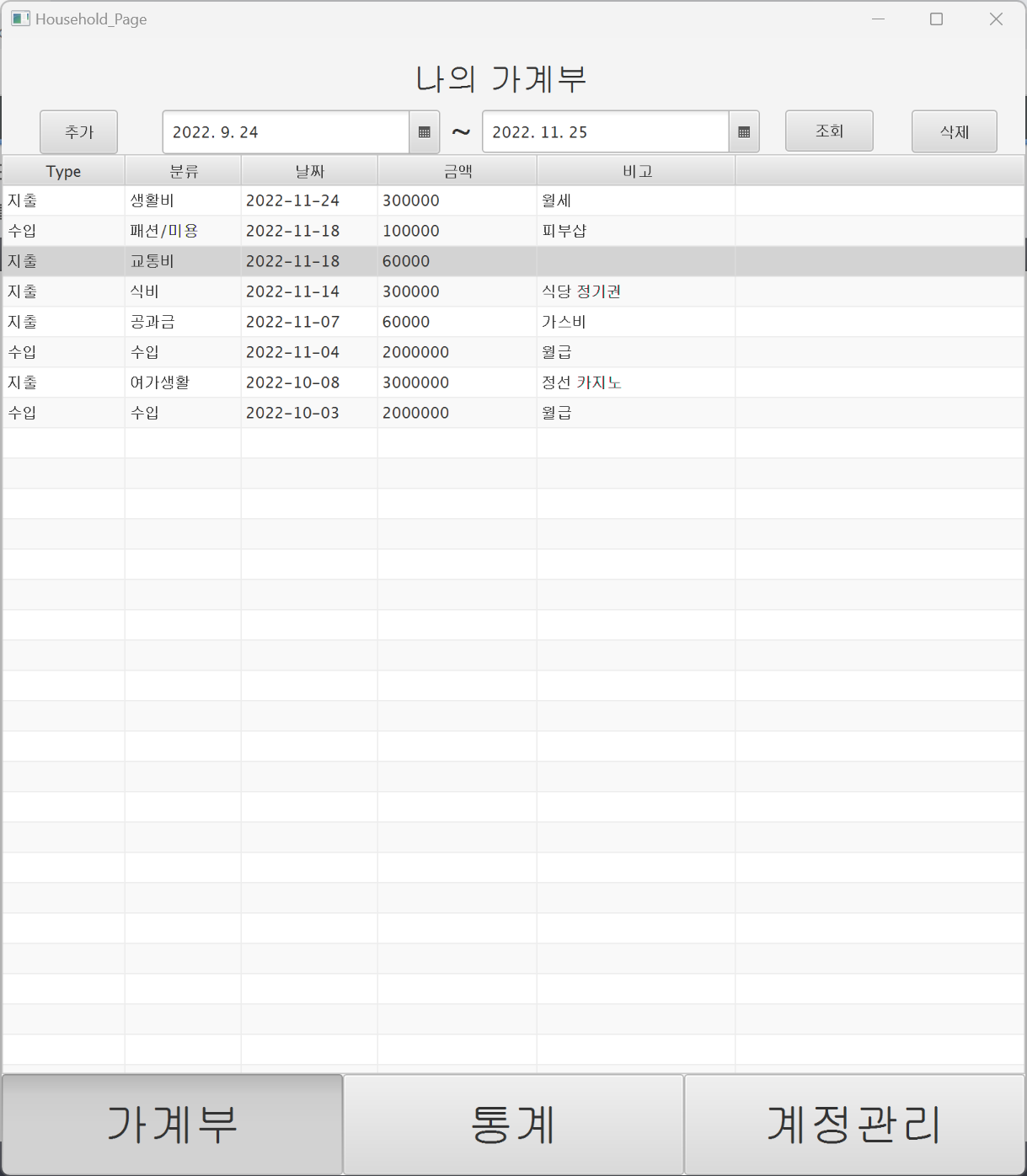
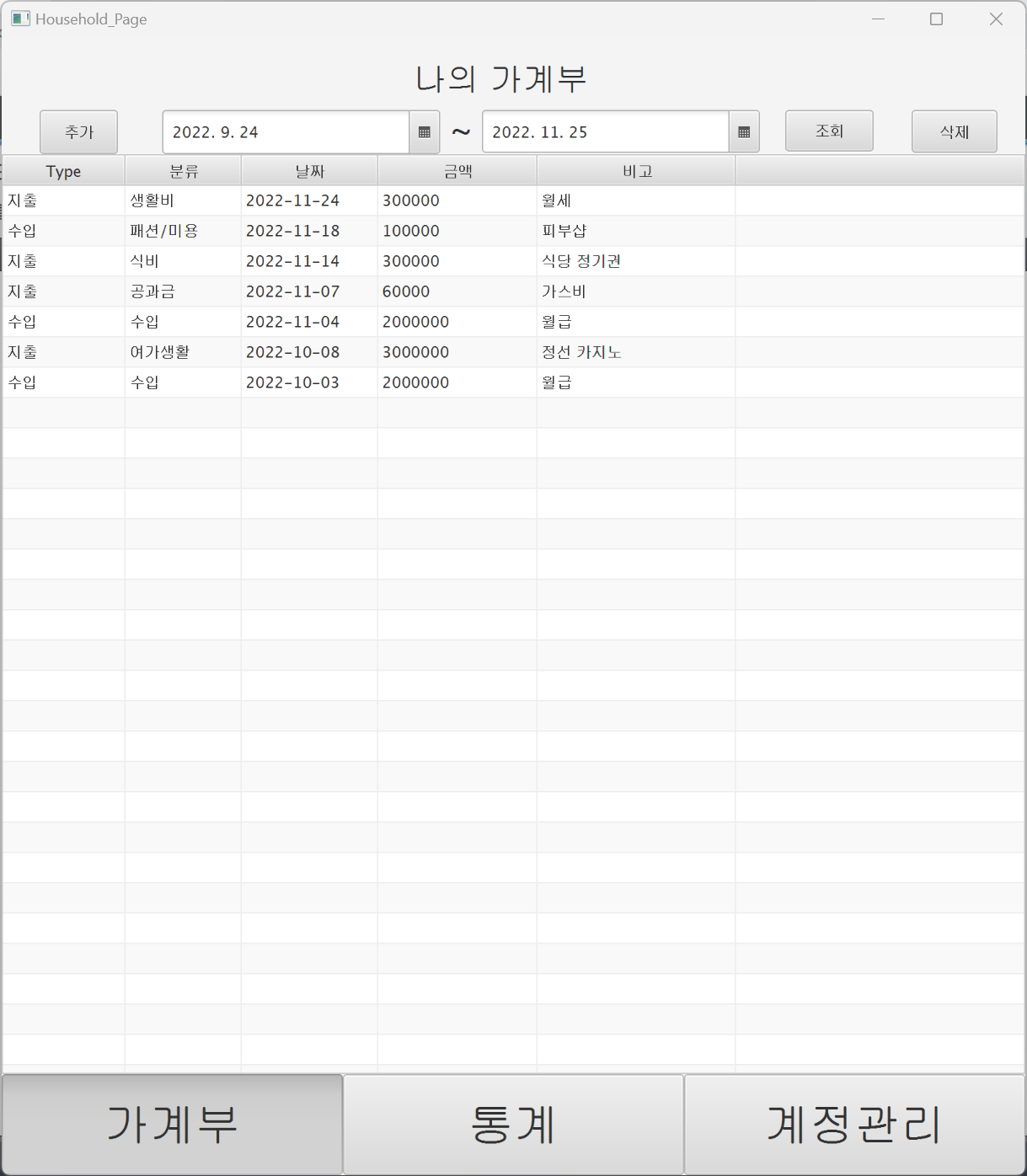
5) 회원 가입 및 오류 화면 6) 로그인 후 Main 화면

7) 가계부 페이지  
초기 한달 이용 내역 및 사용자 정의 기간 내역 화면

8) 입력 창 및 입력 형식 오류 및 누락 시 오류 창

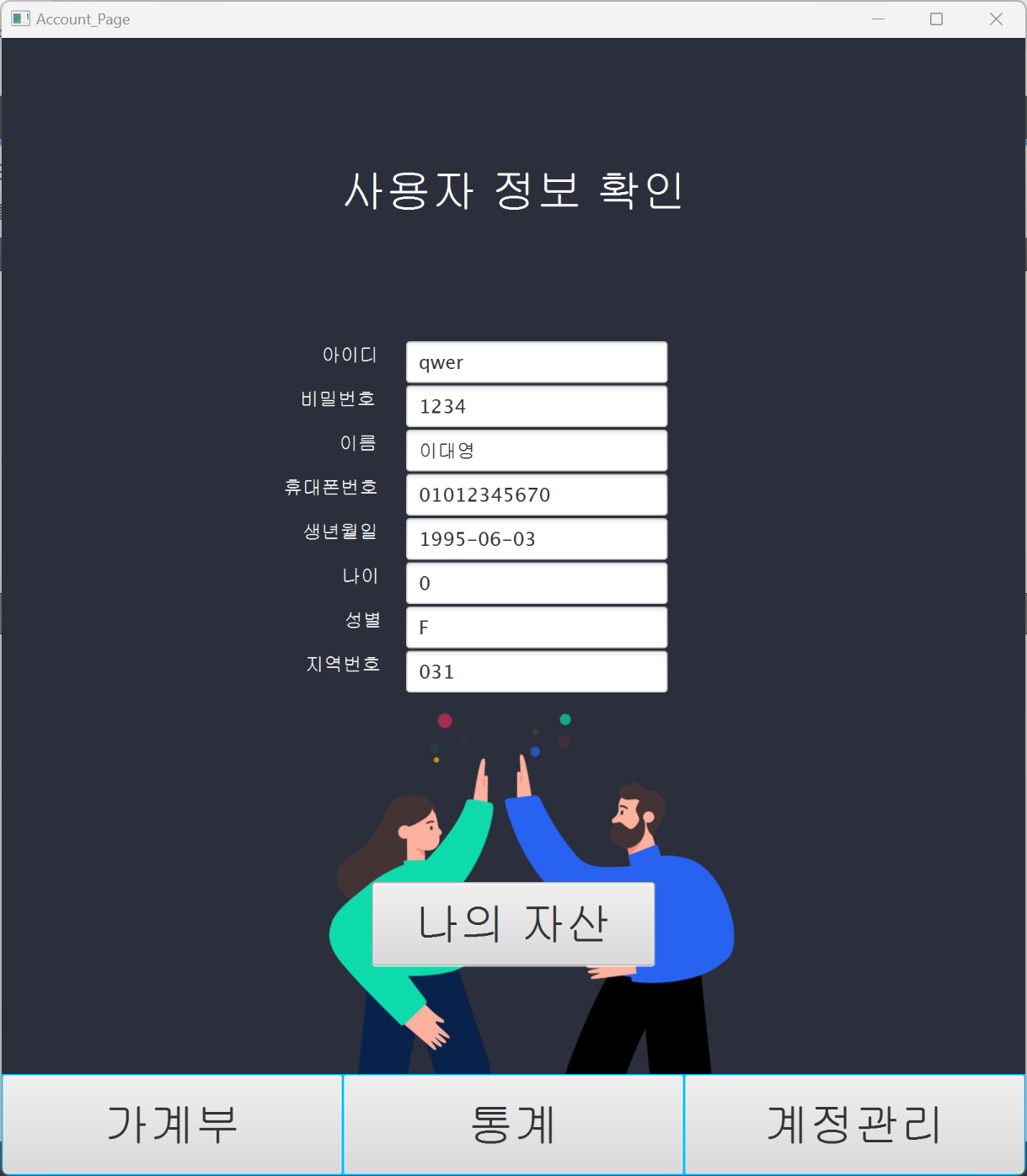
 

9) TableView에서 내역 선택 후 삭제하는 화면

10) 통계 페이지

초기 한달 이용 내역 및 사용자 정의 기간 내역 화면

11) 계정 관리 페이지 12) 나의 자산 페이지