Always be there

-Design-

Student No	
Name	
E-Mail	

[Revision history]

Revision date	Version #	Description	Author
2017/05/23	1.00	초기 버전	
2017/06/20	1.01	클래스 관계 설정	
2017/06/22	1.02	기능 수정	

= Contents =

1. Introduction

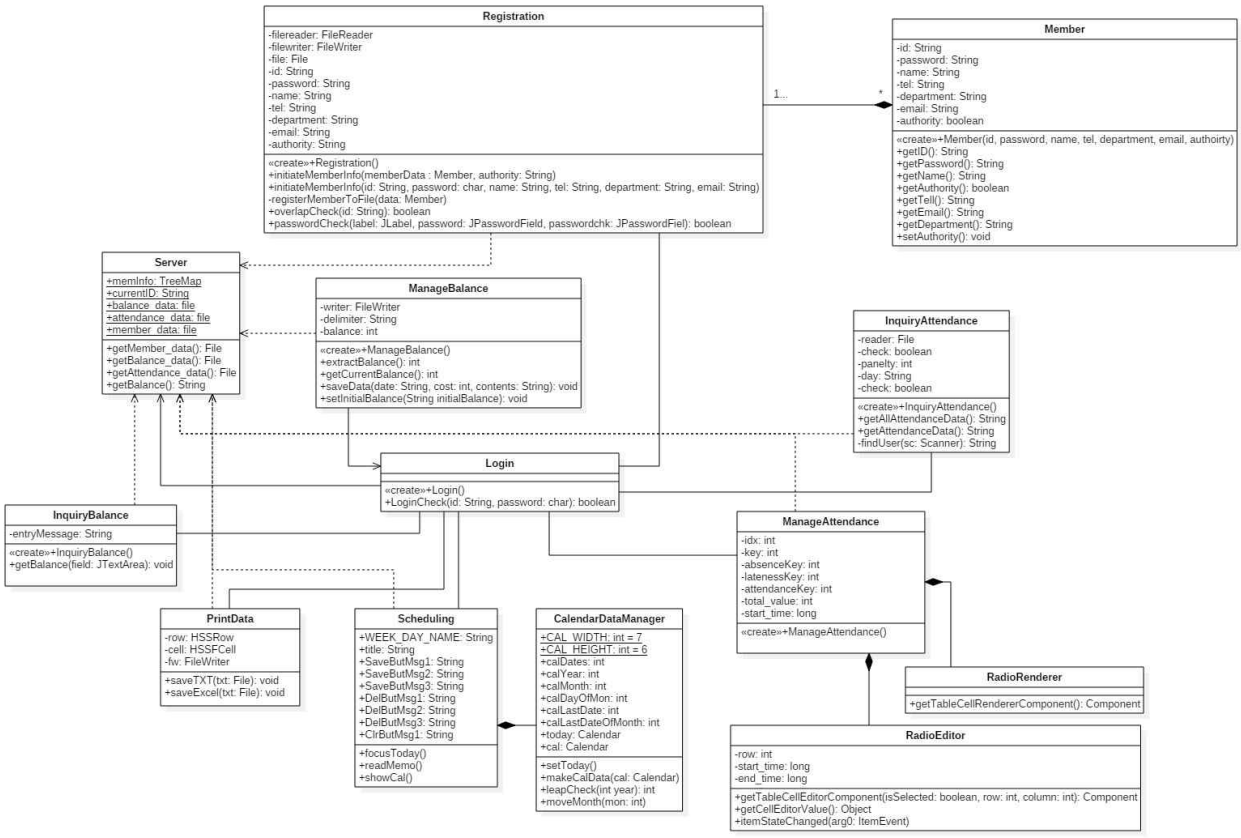
옛날부터 사람들은 자기가 가지고 있는 정보들을 나누기 위해 모임을 가졌다. 순수하게 모여서 친목을 쌓고 서로서로에게 많은 도움이 되었다. 현대에 와서도 모임은 동아리, 동호회, 단체 등으로 바뀌어 규모가 더 커지고 있다. 그림1의 조사 결과를 보면 모 대학교의 학생들 중 절반 이상이 동아리 활동을 해봤다고 답했다. 이렇듯 모임은 많은 사람들이 참여하고 있음을 알 수 있다. 그러나 이렇게 모임들이 많아지면서 악의적으로 사용하는 사람들이 생겨나기 시작했다. 회비를 횡령하거나 특정인에게 특혜를 주거나 보이지 않는 곳에서 일어나지 말아야할 일들이 일어난다. 그래서 이러한 악용을 막고 모임 주최자들이 효율적인 관리를 하도록 도와주고 싶었다. 그리하여 주기적으로 모임을 가지는 크고 작은 단체 혹은 학교 동아리를 위한 종합 회원 관리 프로그램 "Always be there"을 개발하게 되었다.

이 프로그램을 만들어서 이루고자 하는 첫 번째 목표는 바로 효율적인 모임 진행이다. 회원이 많은 모임은 모두에게 정확한 정보를 보내기가 힘들다. 그래서 게시판을 이용해서 공지사항을 올리게 한다. 또 질의응답 기능과 각각 회원들이 모임에 참석할 수 있는 시간을 분석하여 모두가 참여할 수 있는 날을 계산하여 효율적이게 모임 관리를 할 수 있도록 할 것이다. 그 다음으론 청렴한 자금관리이다. 제대로 관리를 안하게 되면 나중엔 어디에 얼마가 쓰였는지 알 수가 없어진다. 그래서 서버에 자금 사용에 대한 이력을 저장하고 관리하게 하여 정확한 자금관리를 도와주는 게 두 번째 목표이다.
출석관리를 해야 하는 학원이나 대학교 동아리가 주 타겟이 될 것이다. 그러나 꼭 출석을 하지 않는 모임이나 단체여도 자금 관리나 일정 공지든 여러 면으로 기능이 많기 때문에 작은 규모에서 중간 규모까지의 고객들도 타겟이 될 것이다.

아래는 Analysis에 이은 이 System개발의 세 번째 단계인 Design에 관한 내용으로써, 실 제 System 구현에 직접적으로 관여하는 모든 요소들의 윤곽을 확정하고 구체적으로 디자인 해 나가는 내용을 다루고 있다. 본 문서의 모든 세부 사항은 직접적인 구현 시 소스코드상 에서의 용례와 완벽히 일치함을 목적으로 한다.

2. Class diagram

아래의 그림은 시스템의 Class Diagram을 표현한 그림이다.



아래의 표는 위의 Class Diagram에서 표현한 Class들의 대한 설명이다.

Class Name	Explanation
Login	시스템을 실행하고 로그인을 할 때 아이디와 비밀번호를 검사하는 클래스이다. - LoginCheck(id : String, password : char) : 아이디와 비밀번호 입력란에 입력한 정보들을 가지고 와서 등록된 회원인지, 등록된 회원이면 비밀번호가 맞는지 확인하는 메소드이다.
Registration	시스템 사용을 위해 회원등록을 할 때 사용되고 로그인 한 후 회원 정보와 로그인한 회원의 아이디를 "Server"클래스의 멤버 변수에 저장하는 클래스이다.

	- initiateMemberInfo(memberData : Member, authority : String) : 등록된 회원을 "Server" 클래스에 있는 "memInfo"에 저장하거나 지금 등록한 회원을 추가 할 때 쓰이는 메소드이다. - registerMemberToFile(data : Member) : 지금 등록한 회원을 "member.txt"파일에 저장하는 메소드이다. - overlapCheck(id : String) : 회원등록 화면에서 입력한 아이디(학번)가 이미 등록된 것은 아닌지 검사하는 메소드이다. - passwordCheck(label : JLabel...) : 회원등록 창에서 비밀번호, 비밀번호 확인란에 입력한 비밀번호가 같은지 확인하고 틀리다면 JLabel에 메시지를 띄워주는 메소드이다.
Member	회원들의 정보가 저장이 되는 클래스이다. - getID() : 아이디 값을 반환하는 메소드이다. - getPassword() : 비밀번호 값을 반환하는 메소드이다. 로그인할 때 쓰인다. - getName() : 이름 값을 반환하는 메소드이다. "InquiryAttendance" 기능을 수행할 때 사용되는 메소드이다. - getAuthority() : 현재 로그인한 사용자가 관리자인지 일반 회원인지를 판단하는 메소드이다. 반환값으로 boolean을 가진다.
Server	이 시스템은 서버를 지원하지 않는다. 회원들의 정보, 출석 정보 등은 원래 서버의 데이터베이스에 저장을 해야하지만 지원하지 않으므로 서버의 기능을 하는 "Server"라는 클래스를 만들었다. 이 클래스에서 현재 로그인한 사용자가 누구인지 알 수 있고 데이터베이스에서 불러올 데이터들이 여기에 변수로 선언되어 있다. - balance_data, attendance_data, member_data : 이 변수들 모두 시스템이 사용할 정보들이 담겨있는 파일을 저장하고 있는 변수들이다.

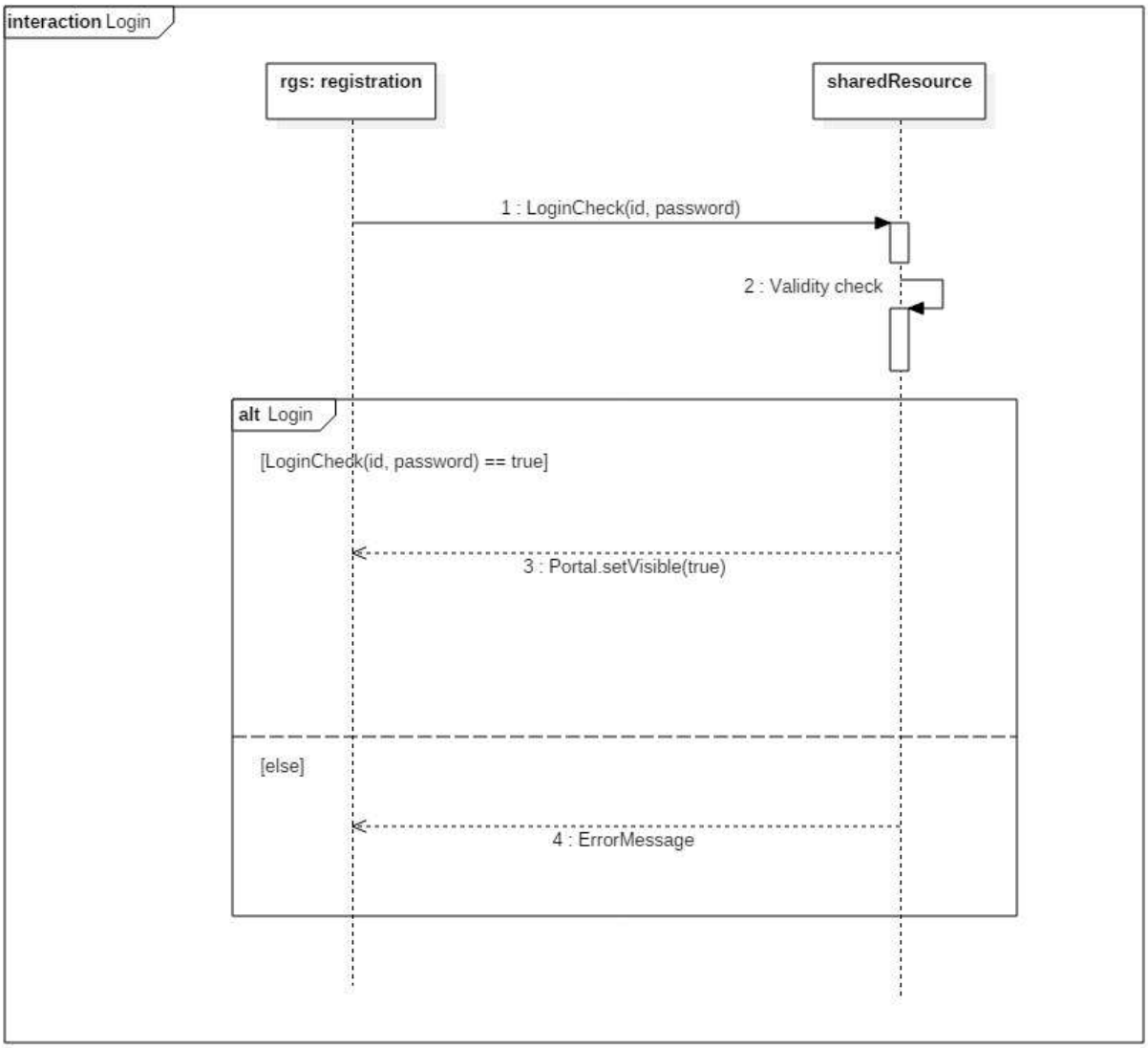
	- memInfo : Java의 Collection Frame Work 중 하나인 "TreeMap"이라는 자료형을 써서 회원 정보들을 저장하였다. - currentID : 현재 로그인한 회원의 ID를 저장해서 필요할 때 해당 회원의 정보를 확인할 수 있다. - Operation 들은 전부 위의 변수들을 반환하는 형태이고 직접 코드를 봤을 때 이해할 수 있음으로 언급하지 않겠다.
InquiryAttendance	로그인한 사용자의 권한에 따라 출석상태를 출력해주는 메소드이다. 로그인 되어 있는 사용자가 회원 권한이면 자신의 출석현황을 출력해주고 관리자 권한이면 등록된 모든 회원들의 출석 상태를 보여준다. - getAllAttendanceData() : 관리자 권한이 가진 사용자는 모든 회원의 출석 정보를 확인할 수 있다. 모든 데이터를 출력해주는 메소드이다. - getAttendanceData() : 권한에 따라서 출석현황을 출력해 주는 메소드이다. - findUser(sc : scanner) : 현재 로그인한 사용자가 회원권한일 경우 출석현황이 저장되어 있는 데이터에서 로그인한 사용자의 정보만 추출해 오는 메소드이다.
InquiryBalance	회원이나 관리자에 상관없이 현재 자금상황을 보여주는 기능이다. - getBalance(field : JTextArea) : 자금 데이터 파일을 열어서 저장되어 있는 정보들을 JTextArea에 출력해준다.
ManageAttendance	출석을 관리하는 클래스이다. 관리자만 사용할 수 있으며 기능을 실행함과 동시에 타이머가 실행된다. 타이머가 시작한 기준으로 10분이 넘어가면 지각, 20분이 되면 결석이 되어서 지각은 2000원, 결석은 5000원의 벌금이 부과된다.
RadioRenderer	"ManageAttendance" 클래스는 JTable 형식으로 만들어진다. JTable 안에 JRadioButton이 들어가게 되는데 이것을 테이블에서 보여주게 하는 것이 해당 클래스이다.
RadioEditor	위의 클래스와 비슷한 명목이다. "RadioRenderer"는 테이블에 JRadioButton을 보여주는 클래스이고 그 것이

	변경이 되거나 편집이 일어나면 처리를 해주는 클래스가 해당 클래스이다.
ManageBalance	해당 클래스는 관리자만 사용할 수 있는 클래스이다. 모임에서 발생한 수입과 지출에 대해서 상세하게 기록해서 저장할 수 있는 클래스이다. - extractBalance() : "Server" 클래스로부터 자금현황 데이터를 가져와서 현재 금액의 데이터만 추출하는 메소드이다. - getCurrentBalance() : extractBalance()가 현재금액을 추출해서 클래스 멤버 변수에 저장하면 그것을 반환해주는 메소드이다. - saveData(date : String, cost : int, contents : String) : 입력한 날짜와 수입 or 지출 비용, 거기에 따른 설명을 매개변수로 받아와서 저장하는 메소드이다. - setInitialBalance(initialBalance : String) : 초기 자금을 설정하는 메소드이다.
Scheduling	관리자가 모임 날짜를 정할 때 사용하는 기능이다. 해당 기능을 선택하면 달력 모양의 새로운 창이하나 뜨고 원하는 날짜에 클릭 후 자세한 정보를 입력하고 저장하면 된다. 회원들은 시스템 메뉴에 "Home"버튼을 눌러서 일정을 확인할 수 있다. - focusToday() : 해당 기능이 실행한 시점을 기준으로 현재 날짜를 출력하는 메소드이다. - readMemo() : 저장되어 있는 일정이 있으면 불러와서 해당 날짜에 저장한다. - showCal() : 달력을 보여주는 메소드이다.
CalendarDataManager	"Scheduling" 클래스는 달력을 보여주는 클래스이다. 해당 클래스는 달력을 어떻게 보여주는지에 대해 구현해놓은 클래스이다. - setToday() : 현재 날짜를 설정하는 메소드이다. Scheduling 클래스에서 focusToday()로 출력을 한다. - makeCalData(cal : Calendar) : 일정을 저장하는 메소드이다.

	- leapCheck(int year) : 윤년인지 체크하는 메소드이다. - moveMonth(mon : int) : 달력을 넘기면 넘긴 달로 다시 달력을 설정 해주는 메소드이다.

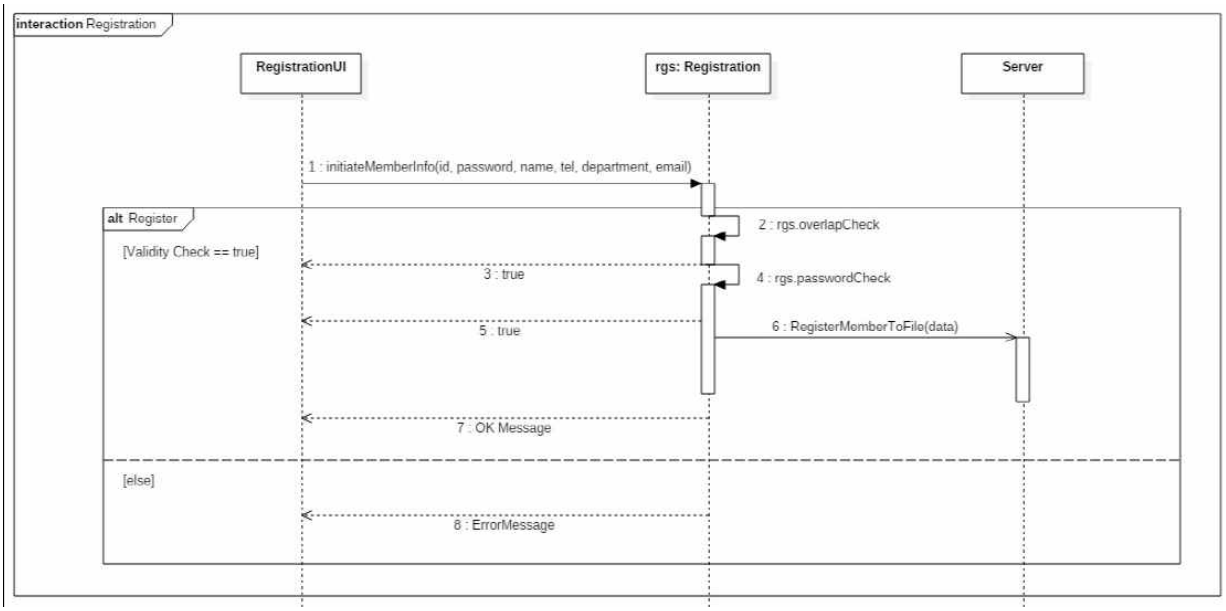
3. Sequence diagram

아래에 나오는 그림들은 Conceptualization에서 표현한 기능들을 Sequence Diagram으로 표현한 그림들이다.



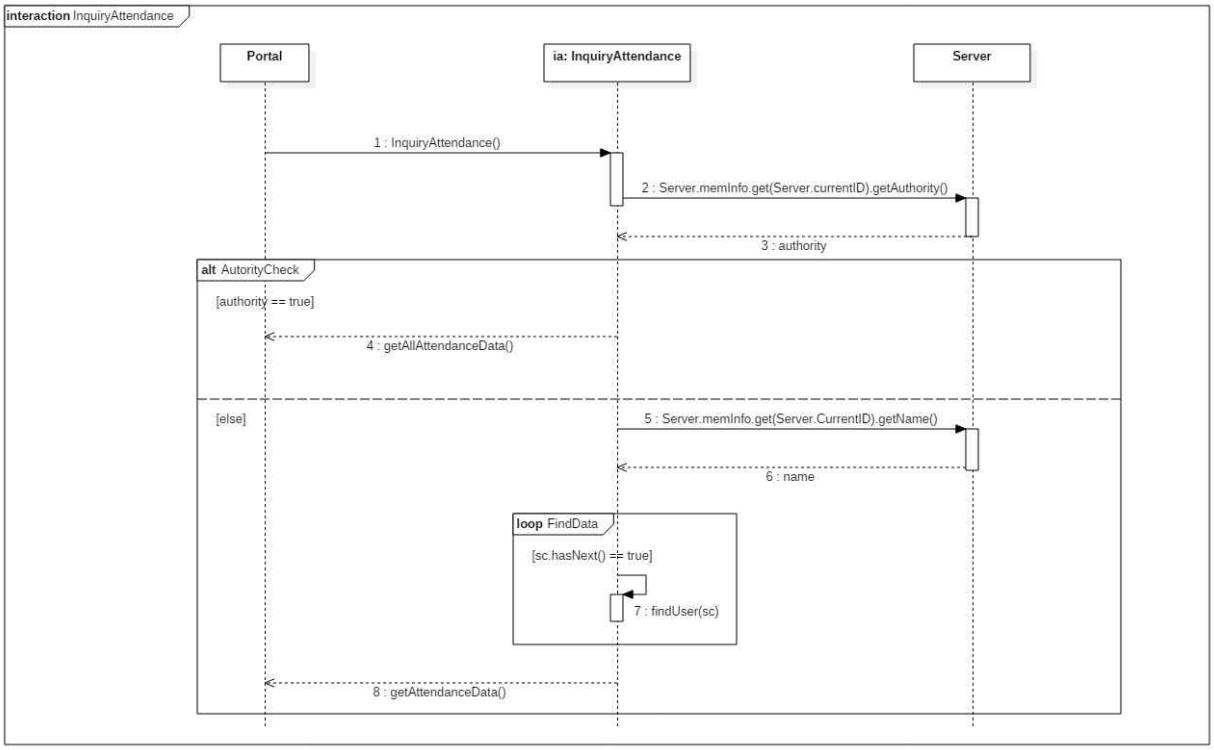
위의 그림은 시스템 실행 후 로그인 기능을 수행할 때를 표현한 Sequence Diagram이다. 아이디, 패스워드를 입력해서 LoginCheck 메소드를 부른다. 그 다음 true를 반환하면 "Portal" GUI Class를 실행한다.

아래의 그림은 시스템의 기능 중 "회원등록"에 대한 Sequence Diagram이다.



시스템 실행 후 "Register" 버튼을 누르면 "RegistrationUI"라는 GUI 클래스가 setVisible(true)가 되고 회원등록 화면이 나온다. 각 입력란에 맞게 입력을 다하게 되면 "Registration" 클래스가 initiateMemberInfo() 메소드로 입력한 정보를 받고 잘못된 입력이 있는지 확인한 후 true이면 RegisterMemberToFile() 메소드를 불러서 회원정보를 저장한다. 그리고 사용자에게 OK메세지를 보여준다. 하지만 만약 false라면 Error 메시지를 보여준다.

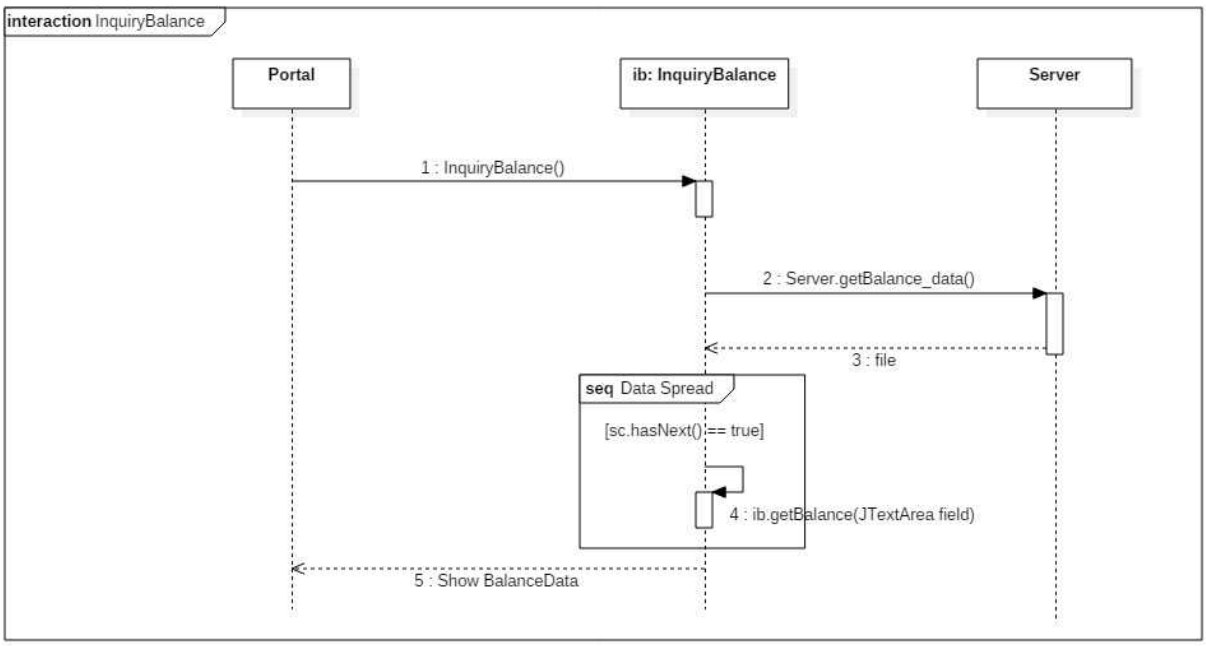
아래의 그림은 시스템의 기능 중 "출석현황 조회"에 대한 Sequence Diagram이다.



메인 메뉴(Portal)에서 "InquiryAttendance" 기능을 선택하게 되면 해당 클래스를 실행한다. 클래스 내부에서 현재 로그인한 회원의 authority(권한)을 체크하고 만약에 true이면 관리자 이므로 "Server" 클래스에서 출석현황 데이터를 불러와서 getAllAttendanceData() 메소드를 통해 모든 회원들의 현황을 출력한다.

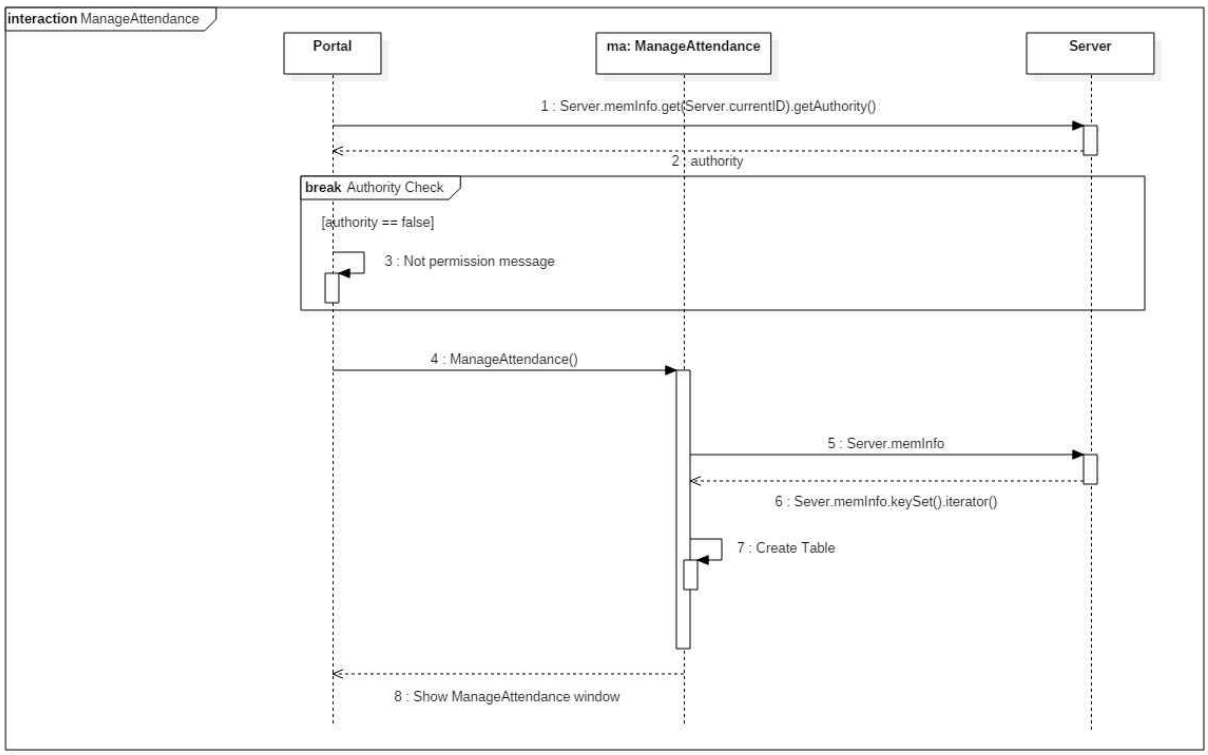
반대로 회원일 경우에는 해당 회원의 정보만 출력해야 하므로 데이터에서 추출해낸다. 그리고 getAttendanceData() 메소드를 통해 해당 로그인한 멤버의 데이터만 출력한다.

아래의 그림은 시스템의 기능 중 "자금현황 조회"에 대한 Sequence Diagram이다.



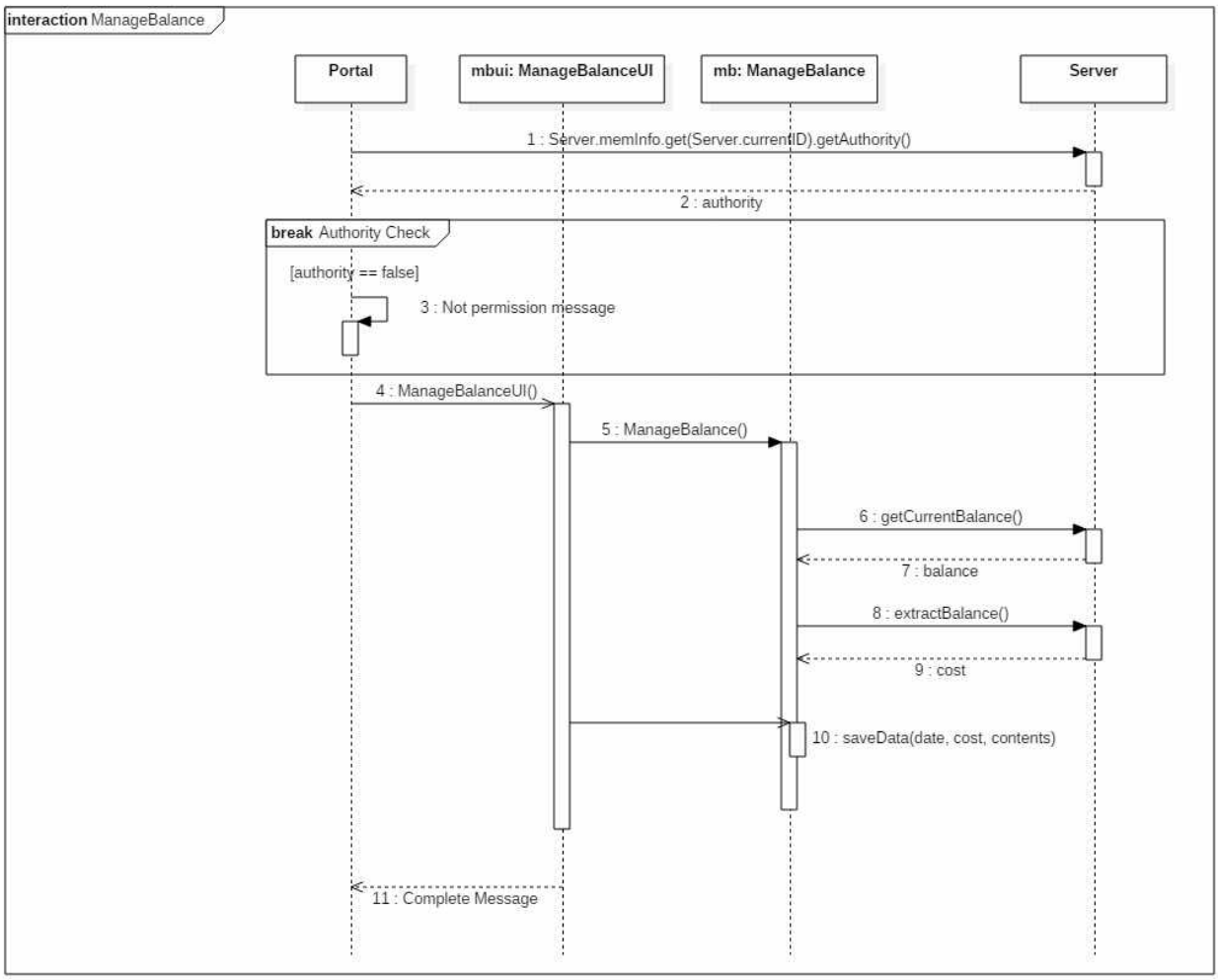
메인 메뉴(Portal)에서 "InquiryBalance" 기능을 선택하게 되면 해당 클래스를 실행한다. 현재 사용자의 권한은 체크하지 않고 "Server" 클래스로부터 자금현황이 들어있는 파일을 열어서 데이터를 가져온 후 getBalance() 메소드를 통해 출력한다.

아래의 그림은 시스템의 기능 중 "ManageAttendance" 대한 Sequence Diagram이다.



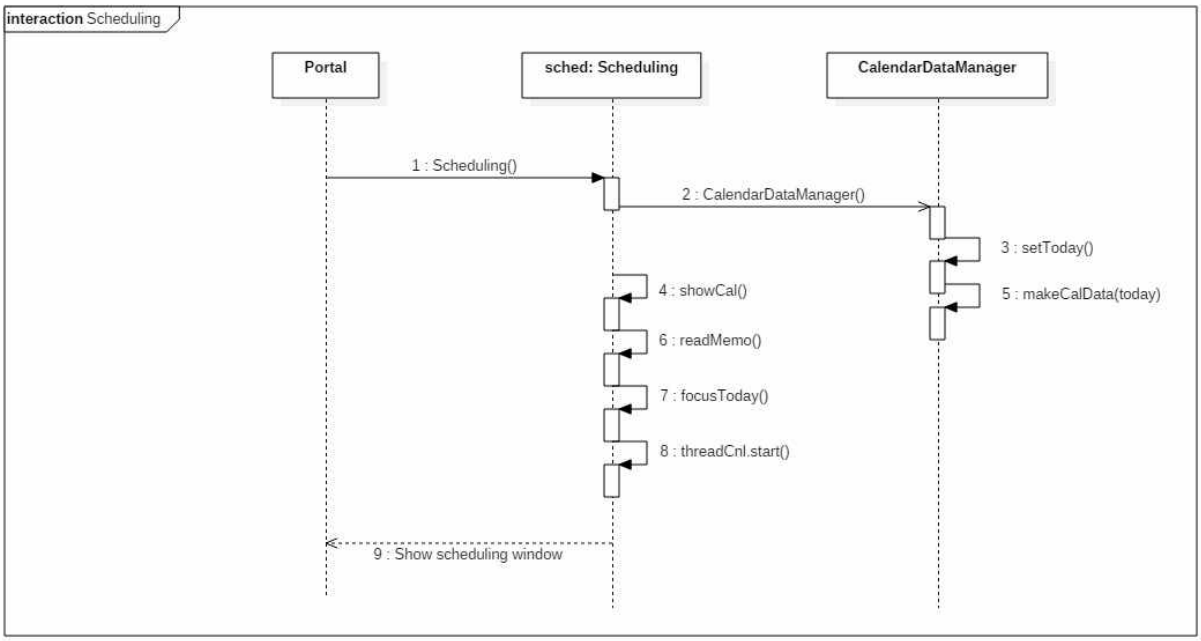
메인 메뉴(Portal)에서 "ManageAttendance" 메뉴를 선택하면 해당 클래스가 실행된다. 생성자가 만들어지기 전에 우선적으로 로그인한 사용자의 권한을 체크한다.
authority가 "true"이면 관리자 이므로 해당 기능을 실행한다. 반대로 "false"이면 일반 멤버이기 때문에 Error 메시지를 출력하고 기능을 실행시키지 않는다.
생성자가 만들어지고 "Server" 클래스로부터 회원등록이 되어있는 회원 데이터를 가지고와서 "Table"을 만든다. 그리고 새창을 띄워서 관리자에게 보여준다.

아래의 그림은 시스템의 기능 중 "ManageBalance" 대한 Sequence Diagram이다.



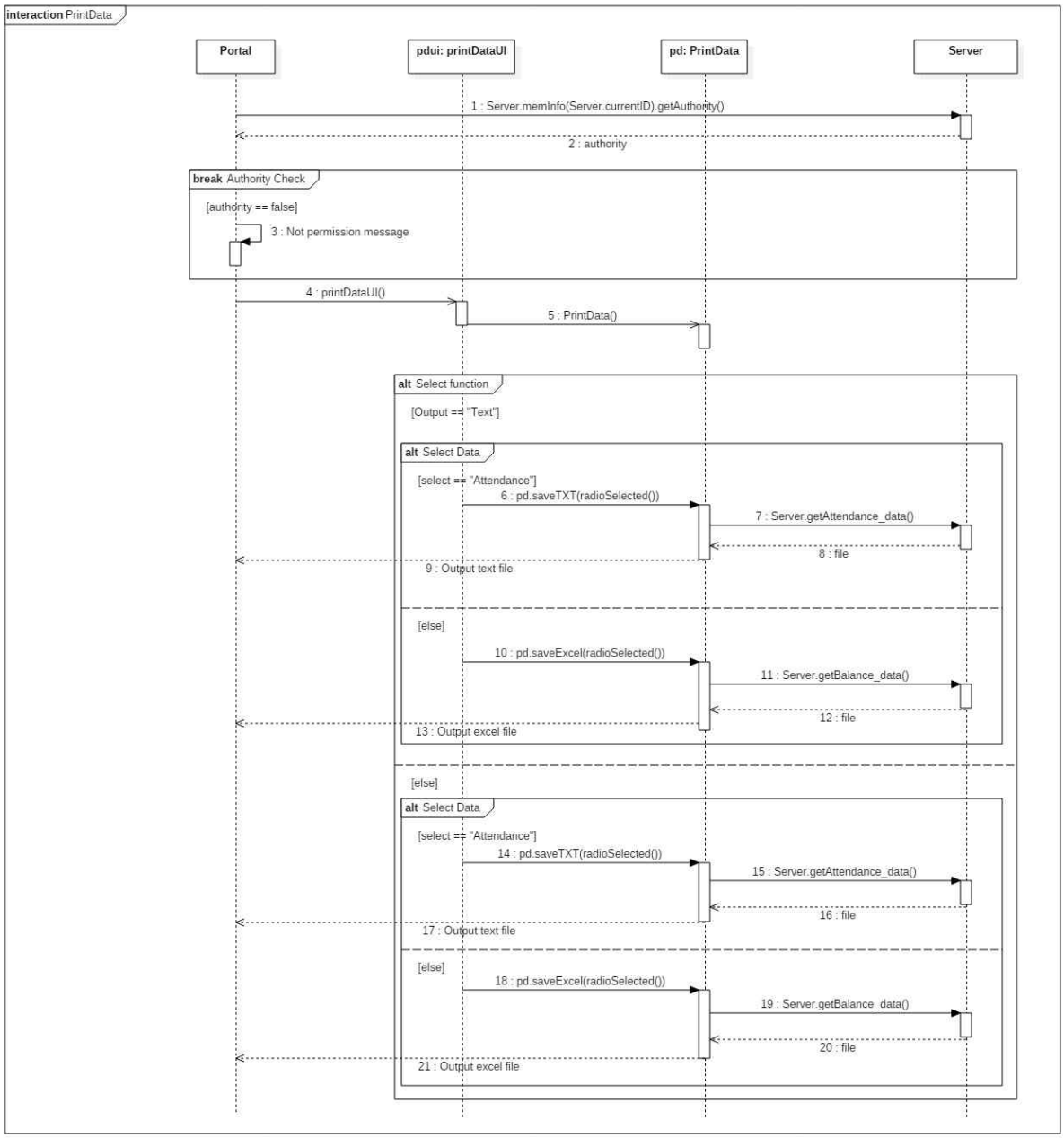
메인 메뉴(Portal)에서 "ManageBalance" 메뉴를 선택하면 해당 클래스가 실행된다. 생성자가 만들어지기 전에 우선적으로 로그인한 사용자의 권한을 체크한다.
authority가 "true"이면 관리자 이므로 해당 기능을 실행한다. 반대로 "false"이면 일반 멤버이기 때문에 Error 메시지를 출력하고 기능을 실행시키지 않는다.
생성자가 만들어지고 "Server" 클래스로부터 "Balance Data"를 가져오게 된다. 여기서 최근 남은 금액을 알아야 하기 때문에 extractBalance()를 통해 추출해낸다. getCurrentBalance() 메소드를 통하여 "balance" 변수에 저장해 둔다. 그 다음 SaveData() 메소드를 통하여 저장을 한다.

아래의 그림은 시스템의 기능 중 "Scheduling" 대한 Sequence Diagram이다.



메인 메뉴(Portal)에서 "Scheduling" 메뉴를 선택하면 해당 클래스가 실행된다. "CalendarManager" 클래스에서는 날짜와 관련된 정보들을 생성하고, "Scheduling" 클래스에선 만들어진 정보와 이전에 저장한 정보들을 합쳐서 UI로 만들어낸다. showCal() 메소드는 달력 형태의 UI를 띄워주는 역할을 하고, readMemo()는 과거에 저장한 일정들을 불러와서 달력에 기록한다. focusToday()는 현재 날짜와 시간 정보를 나타내주고 이것을 Thread로 실시간 반영을 해준다.

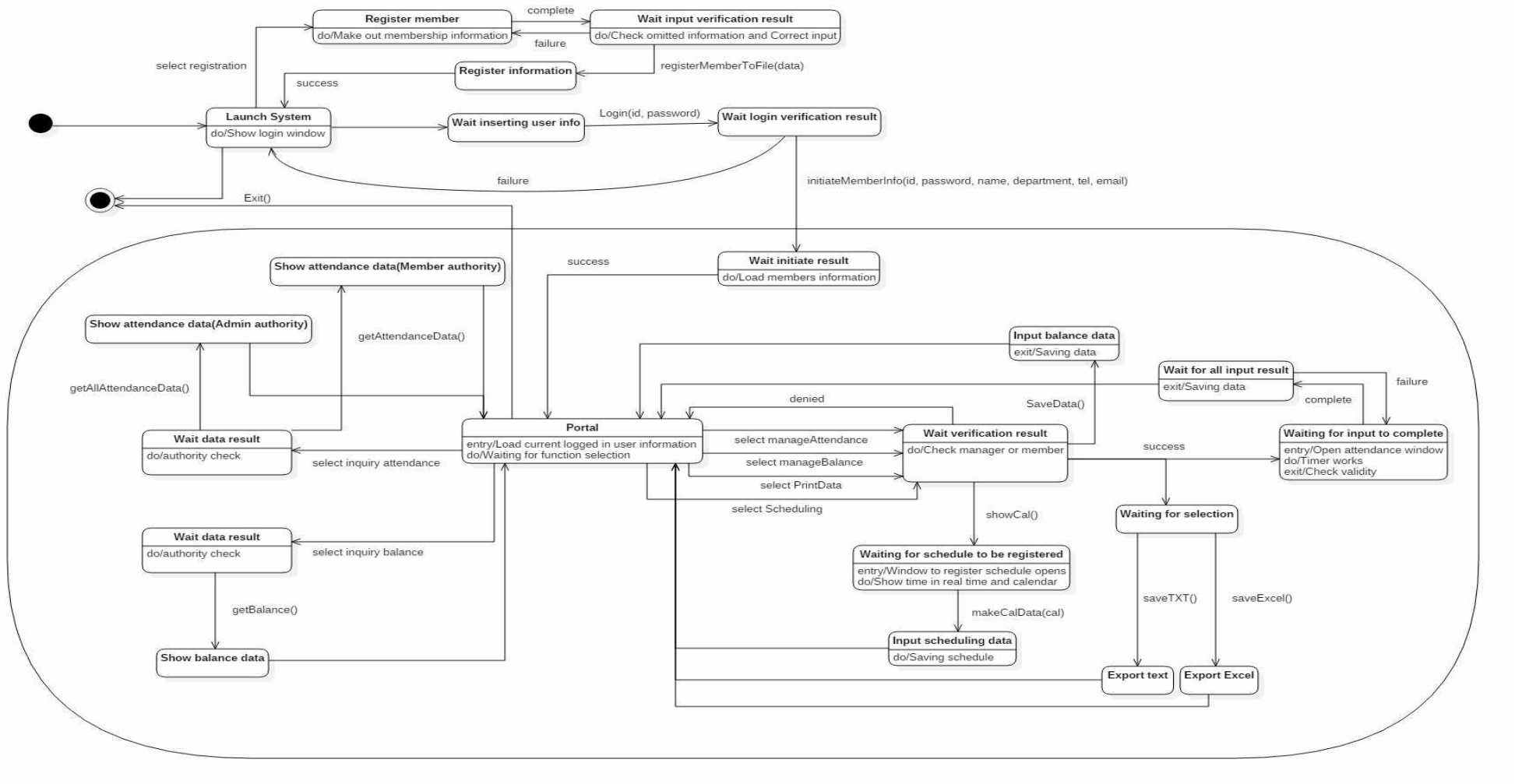
아래의 그림은 시스템의 기능 중 "PrintData" 대한 Sequence Diagram이다.



메인 메뉴(Portal)에서 "PrintData" 메뉴를 선택하면 해당 클래스가 실행된다. 생성자가 만들어 지기전에 authority를 체크해서 관리자이면 해당 기능을 실행한다. 실행되면 간단한 UI를 볼 수 있다. 라디오 버튼으로 "출석", "자금" 데이터를 선택하고 "Text" 또는 "Excel" 버튼을 클릭하여 출력한다.

4. State machine diagram

아래의 그림은 ABT시스템의 State machine diagram을 표현한 그림이다.



아래는 위의 State Machine Diagram에 나온 각 State들에 대해서 간단하게 설명한 표이다.

Status	Explanation
Launch System	시스템이 실행된 상태이다.
Register Member	시스템이 실행된 상태에서 "Register" 버튼을 누르면 회원등록을 하는 창이 나와서 입력하는 상태를 말한다.
Wait input verification result	회원등록 창에서 잘못된 입력이 있는지 결과를 기다리는 상태이다. 잘못된 입력이 있는 경우 "Register Member" 상태로 돌아가고 정확히 입력 했으면 "Register information" 상태로 간다.
Register information	회원등록 창에서 입력한 정보를 토대로 회원등록한 사용자들의 정보가 들어있는 파일에 저장한다.
Wait inserting user info	로그인하기 위해 아이디와 패스워드를 입력하는 상태이다.
Wait login verification result	아이디와 패스워드 입력란에 입력한 정보가 맞는지 검사하는 상태이다. true이면 시스템에 접근 가능하고 false이면 다시 "Launch System" 상태로 돌아간다.
Wait initiate result	시스템 내부에 접근하기 전에 현재 로그인한 사용자의 정보들을 초기화하고 필요한 데이터를 준비하기 위한 상태이다.
Portal	시스템 내부에 들어온 사용자는 이 상태에서 모든 기능을 시작하고 시스템을 끝낼 수 있다.
Wait data result	"InquiryAttendance"와 "InquiryBalance" 기능을 실행하였을 때 각각 메소드를 통해 데이터를 가져오는 상태를 말한다.
Show attendance data, Show balance data	출석현황과 자금현황을 보여주는 상태이다. "Show attendance data" 같은 경우엔 관리자일 경우 "getAllAttedanceData()" 메소드가 실행되어서 모든 회원들의 출석 현황을 출력한다.
Wait verification result	권한을 검사해서 관리자이면 다음으로 넘어가게 하는 상태이다.
Input balance data	자금의 수입과 지출에 대한 정보를 입력하는 상태이다.

Waiting for input to complete	"ManageAttendance"에 관한 상태이다. 모임이 있는 당일에 회원들 개개인 마다 출석하였거나 지각, 결석을 체크하는 상태이다. 모두다 입력을 하여야 다음 상태로 넘어갈 수 있다.
Wait for all input result	"Waiting for input to complete"에서 모든 회원에 대해 출석을 완료하였으면 true 아니면 다시 되돌아간다.
Wait for selection	"PrintData" 기능에 관한 상태이다. 텍스트 파일로 출력을 받을지 엑셀 파일로 출력을 받을지 기다리는 상태이다.
Export text, Export Excel	회원 출석 현황을 출력할건지 아니면 자금 관리 데이터를 출력할건지 선택 하고, 텍스트로 출력 하거나 엑셀 파일로 출력하는 상태이다.
Waiting for schedule to be registered	"Scheduling"에 대한 상태이다. 관리자가 일정을 계획하는 상태이다.
Input scheduling data	계획한 일정을 파일에 저장하는 상태이다.

5. Implementation requirements

ABT 시스템을 구동하기 위해 필요한 요구사항은 아래와 같다.

1) Hardware Requirements

CPU	INTEL Pentium IV 이상
RAM	1GByte 이상
HDD or SSD	10GByte 이상의 여유공간
Network	연결 안 되어 있어도 됨

2) Software Requirments

Operating System	Window 7 이상
Implementation Language	Java (version 1.8.0 이상)

3) Nonfunctional requirements

해당 시스템은 서버가 없으며 서버의 역할을 하는 "Server" 클래스가 있다. 그리고 데이터베이스 없으므로 데이터베이스의 기능을 하는 변수들을 "Server"클래스 안에 멤버변수로 선언해 두었다. 그리고 데이터베이스 역할을 하는 텍스트 파일(.txt)이 구현한 Java Project 폴더에 안에 있다. 각 파일에는 모두 예시로서 데이터가 들어가 있는데 예시에 들어간 데이터처럼 입력을 하여야한다. 다르게 입력을 할 경우 오류가 날 수 있다. 아니면 시스템에 접속하여 기능을 실행하면 자동적으로 입력이 되지만 수동적으로 입력을 할 경우 유의를 해야 한다.

6. Glossary

용어사전에 대해선 다음의 표와 같다.

Terms	Description
Class Diagram	객체지향형 시스템 설계에서, 시스템의 논리 설계를 위한 클래스들의 존재와 그들의 관계를 도식으로 정의한 것. 단일 클래스 다이어그램은 시스템 클래스 구조를 보여줌.
State Machine Diagram	상태 다이어그램은 컴퓨터 과학 및 관련 분야에서 시스템의 동작을 설명하는 데 사용되는 다이어그램의 유형이다. 상태 다이어그램은 설명 된 시스템이 한정된 수의 상태로 구성되어야 한다. 상태 다이어그램에는 여러 가지 형태가 존재하며, 약간 의미가 다를 수 있다.
JLabel	객체지향언어인 JAVA 언어의 GUI 구현에 쓰이는 클래스 중 하나로써 현실세계로 예로 들면 이름표라고 할 수 있다.
boolean	컴퓨터 언어에 있는 데이터 유형으로써 true와 false 두 가지 결과를 가진다.
TreeMap	이진 검색트리의 형태로, 키와 값이 쌍으로 이루어진 데이터를 저장한다.
메소드	멤버 함수라고도 하며, 객체지향 프로그래밍 언어에서 클래스 혹은 객체에 소속된 서브루틴을 가리킨다.

7. References

- **SPARX SYSTEMS – State Machine Diagram Practice**
(http://www.sparxsystems.com.au/resources/uml2_tutorial/uml2_statediagram.html)