클라이언트 서버 프로그래밍

**RMI 수강신청 프로젝트 보고서**

학과: 융합소프트웨어학부 응용소프트웨어전공

학번: 60201976

이름: 장채은

목차

1. **서론**2
2. 프로젝트 목적2
3. 프로젝트 순서3
4. **본론2**
5. 공통코드3
6. 로그인/회원가입7
7. 학생 정보13
8. 강의 정보20
9. 수강 신청27
10. 프로젝트 실행 결과34
11. 결론**40**
12. 서론
13. 프로젝트 목적

목적을 설명하기에 앞서 프로젝트에서 사용한 기술 및 간단한 개념을 설명하고 넘어가고자 한다. Client는 사용자 서비스 화면을 제공하며, 사용자가 작성한 정보를 서버에 전달하여 정보를 받는 역할을 한다. Server에서는 Data와 Client 연결을 해준다. 또한 프로그램이 예외가 발생하여 생기는 에러 상황을 방지해주고, Client에서 요청한 서비스를 제공한다. 클라이언트와 서버가 소통하는 프로그램의 경우 하나의 컴퓨터 안에서 서비스를 제공하고 받을 수 있지만, 서비스를 제공을 하는 Server와 제공을 받는 Client의 경우 다른 환경(위치 등) 일 수 있기 때문에 클라이언트 서버프로그래밍 기본 개념 및 기술을 사용하기 위해서는 Client와 Server, Data로 각각의 프로젝트 파일로 생성해야 한다. RMI는 순수 자바 기반의 분산 프로그래밍(여러 개로 나누는)을 지원하는 라이브러리로 각각의 Client와 Server의 소통이 가능하도록 도와준다.

따라서, 수강신청 프로젝트는 Client와 Server, Data가 서로 요청과 응답을 받으면서 Client와 Server간의 상호관계(데이터 요청 및 제공 등)를 알아보는 목적을 갖고 RMI 라이브러리를 바탕으로 한 수강신청 프로젝트를 진행한다.

1. 프로젝트 순서

프로젝트는 로그인, 회원가입, 학생 정보 기능, 강의 정보 기능, 수강신청 기능으로 이루어져 있다. 첫 번째로, 수강신청 프로그램의 모든 서비스 기능은 Sequence Diagram을 통해 시각적으로 설명한다. 두 번째로, 프로젝트 Source Code 확인을 통해 서비스의 유지 보수 및 기능 & 기술에 대해 자세히 알아보며, 세 번째로 프로젝트 실행 결과를 CMD 화면으로 보여준다. 이후 마지막으로, 프로젝트를 하면서 알게 된 점, 아쉬웠던 점을 작성하며 보고서를 마무리하고자 한다.

1. 본론

본격적인 기능 설명에 들어가기에 앞서, 사용자가 서비스를 사용할 때, 명확한 이유의 예외 결과를 알려주는 것이 중요하다고 생각했기 때문에 필자는 수강신청 프로그램에서 예외 처리에 힘을 쏟았다는 점을 강조한다.

서비스는 크게 로그인/회원 가입, 학생 정보, 강의 정보, 수강 신청을 구현했다. 서비스에 대해 자세히 설명하기 전에 공통적으로 사용하는 코드부터 설명하고자 한다.

1. 공통 코드

* Course.java, Student.java Reservation.java는 Client, Service, Data에서 동일한 코드를 사용한다.
* Server의 성공/실패 여부를 Enumeration으로 받아오기 때문에 동일한 예외처리 Class를 갖고 있다.
* Log도 Log File 생성 method 이외의 Class는 Client, Server, Data 모두 동일하다
* 각각의 method는 Log에 학번을 넣어 File에 저장하는 코드를 항상 갖고 있다.

🡪 ***LOG***.info(logUser);

Course.java [Client/Server/Data]

텍스트이(가) 표시된 사진

자동 생성된 설명

Reservation.java [Client/Server/Data]

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Reservation의 경우, Client에 보여지는 console 내용과 data file에 저장할 값이 달라야 하므로 toString을 Client와 Data에서 달리 작성하였다. 위이 코드가 Client며, 아래 코드가 Data Code이다.

Student.java [Client/Server/Data]

텍스트이(가) 표시된 사진

자동 생성된 설명

Student.java의 경우 Data 프로젝트 내의 소스 파일과 Client의 파일이 살짝 다르다. Student의 비밀번호는 File에 저장되어 toString()으로 불러오면 안 되기 때문에 Data에서는 허용하되, Client, Server에서는 허용되지 않도록 아래처럼 변경해주었다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Data에서는 File을 초기화 해준 다음, 다시 List 값을 복사하여 File에 새로 저장하기 때문에 password 부분이 필요하다. 따라서, Client와 Server에서는 기존의 Student.java toString() 함수에서 this.password 부분을 제외시켜준 것이다.

StringReturnException.java [Client/Server]

Enum으로 된 Exception Class이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

StringReturnResponse<T>.java [Client/Server]

Server에서 서비스 성공/실패 여부를 다음 Class로 된 객체를 통해 return을 해준다. 이에, Client와 Server 모두 위 Class를 저장하고 있어야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Constructor를 두개 만들어서 실패한 경우 StringReturnException, 성공한 경우 Result 값을 return 하여 Client에서 확인하도록 설계했다.

LogFormat.java [Client/Server/Data]

텍스트이(가) 표시된 사진

자동 생성된 설명

Log에는 학번, 시간, Class의 이름과 Method의 이름이 필요하다. 따라서 Log Formatter를 상속받아 위와 같이 코드를 변경하여 사용하고자 한다.

NullDataException.java

Student, Course List의 File이 없는 경우 처리하도록 생성한 NullException이다. String ReturnException을 사용하기 때문에 따로 이 Exception을 상속받아 사용하는 코드는 많이 없지만, 상속받은 Exception 코드를 경험해보고자 지우지 않고 코드에 적용하여 사용했다.

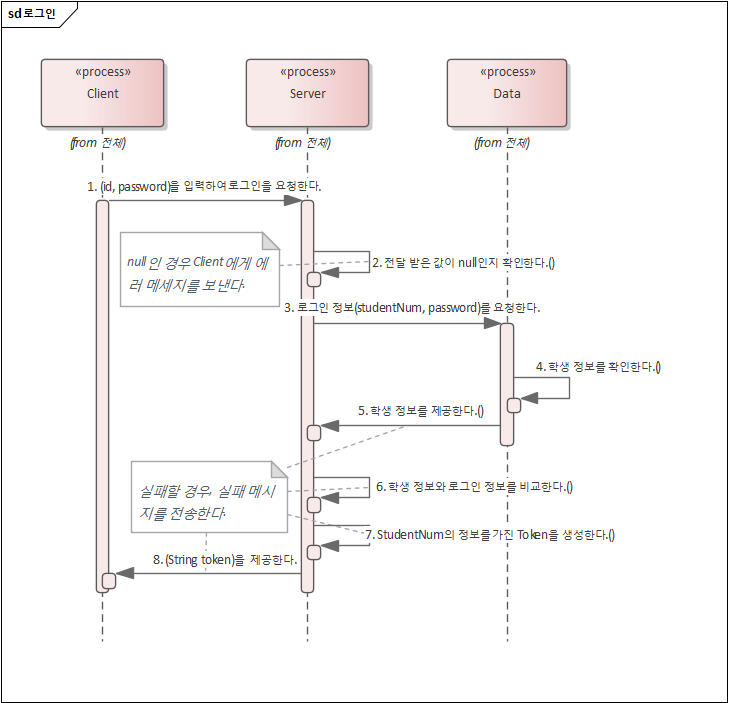
텍스트이(가) 표시된 사진

자동 생성된 설명

1. 로그인/회원가입

회원 가입과 로그인의 경우 사용자가 직접 자신의 정보를 작성하여 Client가 서버에 전달하기 때문에 값이 전달되지 않는 경우 데이터에 값을 작성하기 이전에 예외처리를 진행해야 한다. 로그인, 회원가입 이후, 수강신청 서비스를 제공받을 수 있도록 설계를 해두어 각 서비스에 접근하여 로그인한 사용자와 접근하려는 사용자가 일치하는 지 확인하는 Token도 생성해준다.

로그인



로그인 EA는 다음과 같다. EA 시나리오 순으로 아래에 코드를 작성할 것이다.

1. [Client: Client.java] – main

텍스트이(가) 표시된 사진

자동 생성된 설명

프로그램을 실행하면 Log를 실행해줘야 하며, 그 이후 login/sign-up이 될 때까지 loop문을 돌리면 된다.

1. [Client: Client.java] – logConfigurationMethod

텍스트이(가) 표시된 사진

자동 생성된 설명

로그 File을 생성하는 Method로 (1)- 에서의 LogFormat을 File에 저장하도록 한다. Console이 아닌, File에 저장할 것이므로 ConsoleHandler가 아닌 FileHandler를 사용해야한다.

1. [Client: Client.java] – login

텍스트이(가) 표시된 사진

자동 생성된 설명

로그인의 경우 학번과 비밀번호를 Server에게 전달한 후, login이 되었다면 학번을 Return 해주고 아니면 null을 return 해준다. 연결된 server에서 return result는 token이므로, token을 저장해준다. Token은 server에 연결될 경우 모든 method parameter로 전달해 줘야하기 때문에 필수적으로 생성이 필요하다.

1. [Server: Server.java] – login

텍스트이(가) 표시된 사진

자동 생성된 설명

Client에서 받은 정보가 null인지 확인하며, Login이 되었다면 token을 생성해준다. 이 과정 속에서 문제가 발생하게 되면 바로 Client에 에러 메시지를 전송한다. 이 때, 에러 메시지는 앞서 설명한 Exception class 자체를 return 한다.

1. [Server: AES128.java] – AES128, encrypt

텍스트이(가) 표시된 사진

자동 생성된 설명

Token 생성의 경우 Spring의 JWT와 완전히 동일한 메커니즘으로 생성을 하지 못했다. 하지만, Token 내에 User의 정보(학번)가 저장되어 있기 때문에 암호화를 통해 Token을 생성하여 Client에게 전달해준다.

Token과 학번의 정보는 암호화된 코드들로 비교할 예정이기 때문에 복호화가 필요가 없다. 복호화 함수를 사용하는 경우, 데이터의 보안의 문제가 발생할 수 있다고 생각이 들어 암호화로만 사용을 하고자 했다.

1. [Server: Server.java] – createToken

텍스트이(가) 표시된 사진

자동 생성된 설명

Login의 extract method로, AES128를 사용하여 학번 정보를 암호화한다. 이는 학번이 암호화된 Token으로 생성되는 것이다.

1. [Data: Server.java] – checkLogin

텍스트이(가) 표시된 사진

자동 생성된 설명

Data에서 전달받은 학번과 비밀번호를 가진 학생이 있는지 확인한다.

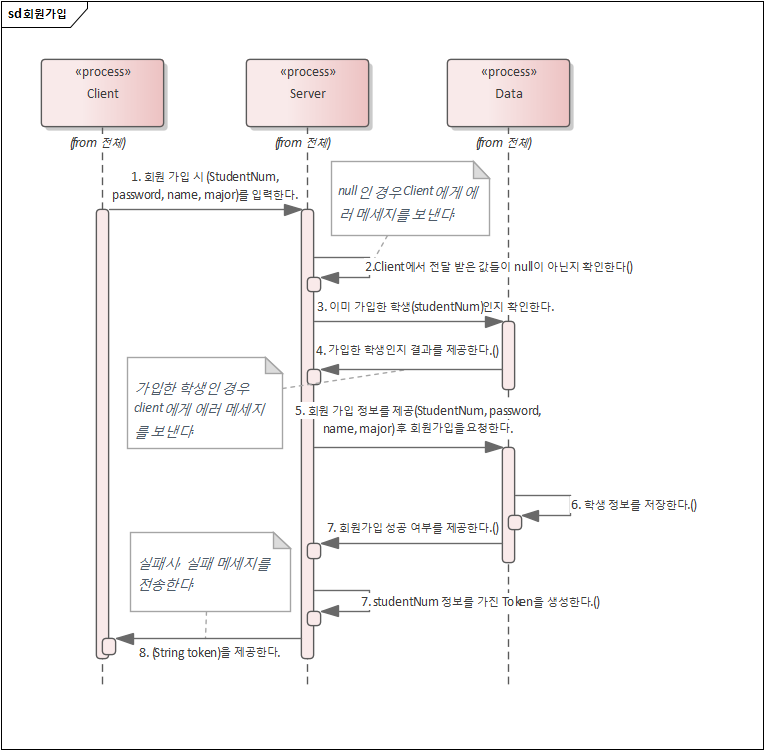
1. [Data: StudentList.java] – checkStudent

텍스트이(가) 표시된 사진

자동 생성된 설명

StudentList에 저장된 정보와 입력 받은 데이터를 확인하여 로그인 여부를 Server에 전달한다.

회원가입



회원가입 EA는 다음과 같다. EA 시나리오 순으로 아래에 코드를 작성할 것이다.

1. [Server: Server.java] – signUP

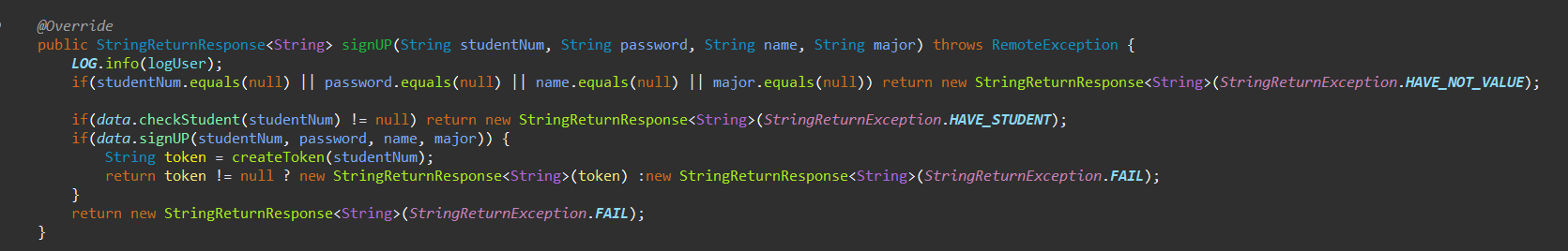
텍스트이(가) 표시된 사진

자동 생성된 설명

회원가입의 경우 학번과 비밀번호, 이름, 전공을 Server에게 전달한 후, 회원가입이 되었다면 암호화된 학번을 Return 해주고 아니면 null을 return 해준다.

Log 및 Token과 관련된 내용은 로그인 부분에서 언급했으므로 넘어가도록 한다.

1. [Server: Server.java] – signUP



Client에서 받은 정보가 null인지 확인하며, 회원가입이 되었다면 token을 생성해준다. 이 과정 속에서 문제가 발생하게 되면 바로 Client에 에러 메시지를 전송한다. 이 때, 에러 메시지는 앞서 설명한 Exception class 자체를 return 한다.

로그인과 달리, 회원가입이므로 이미 가입되어 있는 학번은 다시 가입할 수 없다. 따라서, Data에 학번 중복조회가 필요하다. 이 또한 문제가 발생하게 되면 바로 Client에 에러 메시지를 전송한다.

Token 생성 또한 위 로그인 단계에서 작성했기 때문에 따로 코드를 첨부하지 않고 넘어간다.

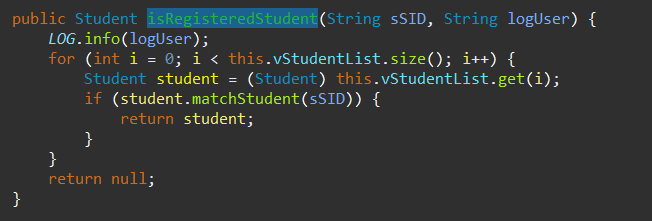
1. [Data: Data.java] – checkStudent

텍스트이(가) 표시된 사진

자동 생성된 설명

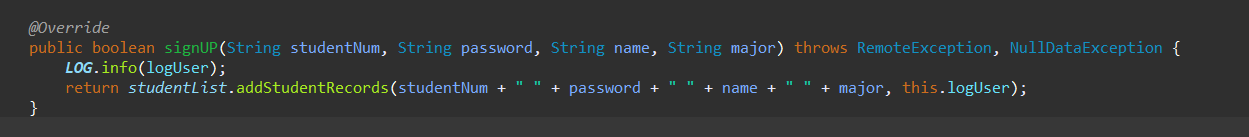
전달받은 데이터가 학생 데이터에 저장이 되어 있는지 아닌지를 확인하는 메소드이다.

1. [Data: StudentList.java] – isRegisterdStudent



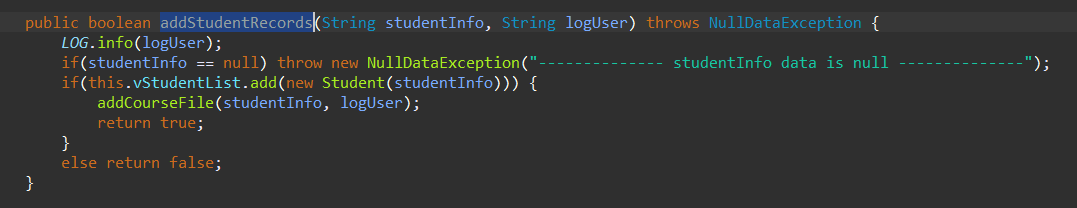
학번을 비교하여 이미 학생 정보가 저장되어 있다면 null을 return 한다.

1. [Data: Server.java] – signUP



Server로부터 정보를 받아와 File 및 ArrayList에 저장하고자 한다. 실제로 저장하는 method는 studentList 내부에 저장되어 있다. 위 코드는 Server와 StudentList를 연결해주는 역할을 한다.

1. [Data: StudentList.java] – addStudentRecords



ArrayList에 회원가입 한 학생 정보를 저장하고, 실제 File에도 저장을 해줘야한다. 실제 File에 저장하기 위해 extract method인 addCourseFile을 생성했다. List와 File에 모두 저장이 되었다면 return true를 해준다.

1. [Data: StudentList.java] – addCourseFile

텍스트이(가) 표시된 사진

자동 생성된 설명

BufferedWriter를 사용하여 기존에 저장된 TextFile에 정보를 붙여 저장해 둔다.

로그인과 회원가입이 완료되면 학번을 main에 return을 해준다. return의 값이 null 이 아니라면 수강 신청 서비스로 화면을 이동하며 Client, Server, Data에 어떤 사용자가 사용했는지 log로 저장하기 위한 User 설정을 login/회원가입에서 return 한 학번으로 저장한다. 이를 Server와 Data에 한 번에 전달해준다.

[Client: Client.java] – main

텍스트이(가) 표시된 사진

자동 생성된 설명

어디에서든 사용할 수 있도록 속성을 저장한 후, Server에 학번 정보를 넘겨준다.

[Server: Server.java] – sendServerStudentForLog

텍스트이(가) 표시된 사진

자동 생성된 설명

Client와 마찬가지로 어디서든 사용할 수 있는 속성을 저장하고, Data에 학번 정보를 넘겨준다.

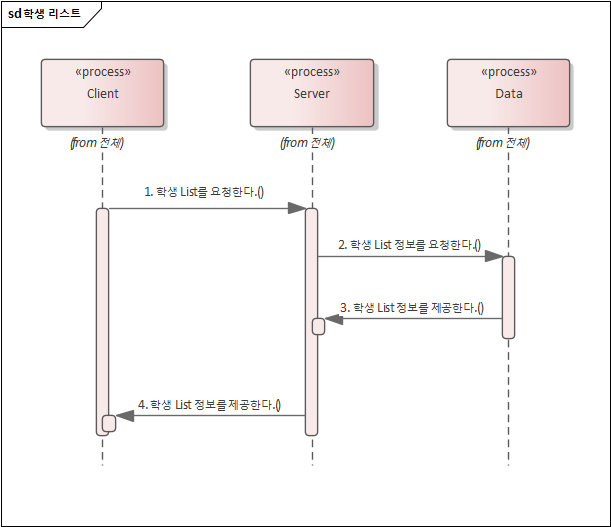
[Data: Data.java] – sendServerStudentForLog

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 학생 정보

학생 리스트



학생 리스트를 불러오는 메소드를 호출하게 되면, 따로 Token 값을 전달해주더라도 token과 사용자가 일치하는지 확인하지 않는다. 그 이유는 따로 값을 바꾸는 기능을 수행하는 것이 아니라, 데이터를 확인만 하기 때문이다. 데이터에 접근해 값을 추가하거나 지우는 경우는 회원가입과 로그인을 한 User만 가능하지만, 학생 및 강좌, 수강신청 되어있는지 확인하는 User는 따로 인증 없이 가능한 서비스들이 많기 때문이다.

1. [Client: Client.java] – showData

텍스트이(가) 표시된 사진

자동 생성된 설명

리스트로 불러와서 Console에 작성하는 List의 종류들이 많기 때문에 특정 객체를 선정하지 않고, Generic 타입으로 선언하여 코드의 중복을 줄인다.

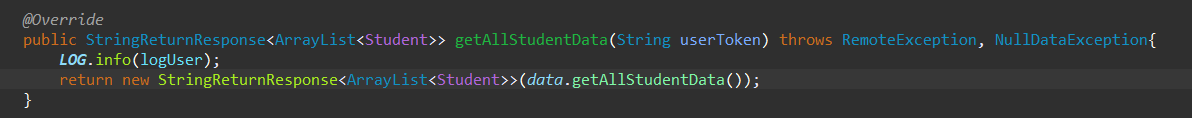
1. [Client: Client.java] – registerCourseMenu

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자가 console로 원하는 서비스를 입력하면 그 서비스에 맞게 서버와 연결해주는 함수이다. 그 중에서도 학생 정보 리스트를 불러와야 하기 때문에 Client에서 호출하는 Server 메소드 중 server.getAllStudentData 함수를 호출한다.

1. [Server: Server.java] – getAllStudentData



Server에서는 NullException 처리가 되어 있으므로 따로 Exception 처리를 해두지 않았다. Data에서 값이 없는 것이 확인되면 에러가 발생한다.

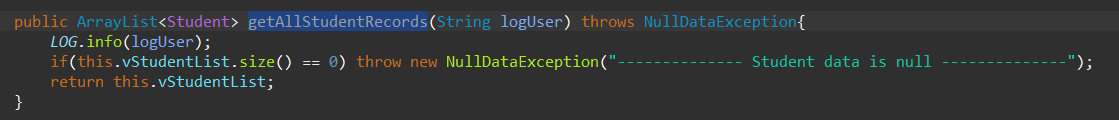
1. [Data: Data.java] – getAllStudentData

텍스트이(가) 표시된 사진

자동 생성된 설명

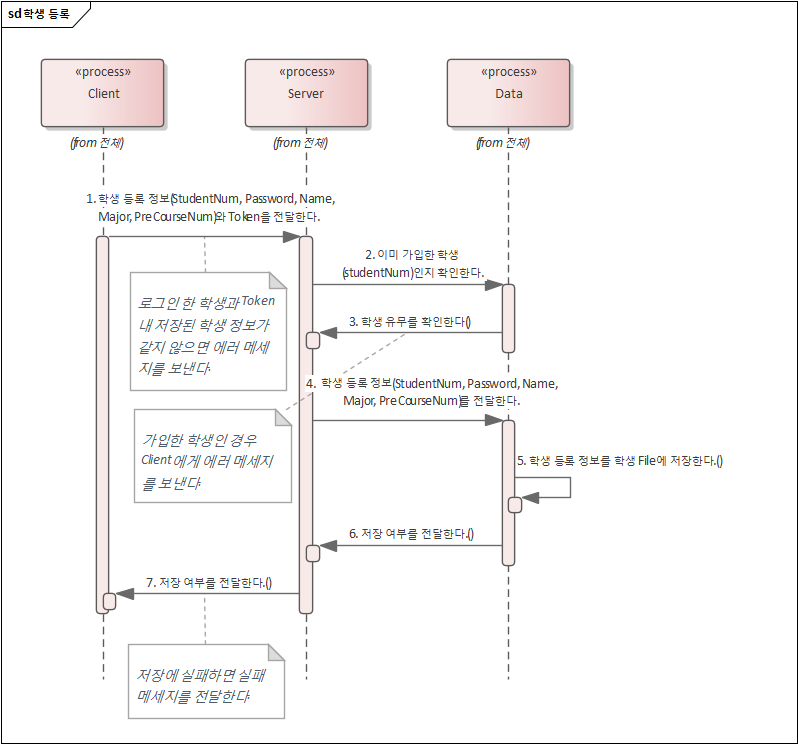
저장된 학생 List에서 값을 불러와서 Server로 return, Server는 Client로 return 해준다.

1. [Data: StudentList.java] – getAllStudentRecords



List에 데이터 값이 없으면 exception 처리를 해주고 아니면 값을 return 해주도록 설계하였다.

학생 등록



학생 등록 시, Login/회원가입을 통해 생성한 Token 값을 Server에 전달을 해준다. 이 때, 서버는 logUser로 저장한 학번을 암호화하고 암호화된 Token 값과 서로 비교해서 같지 않으면 에러 메시지를 전달해준다. 이후, List를 요청하는 서비스를 진행하면 된다. 참고로 Login/회원가입을 제외한 Client에서 호출하는 모든 Server 메소드는 token 값을 parameter로 전달해줘야 한다.

학생 등록은 회원가입과 비슷한 절차로 진행된다. 회원 가입과 다른 딱 한 가지는 선 이수 과목을 입력해준다는 점이다. 이에, 깊은 설명은 아래 코드를 통해 설명하고자 한다.

1. [Client: Client.java] – registerCourseMenu

(3)--b와 동일한 동일한 코드이므로 코드 사진은 생략한다. Method 내에서 addStudent로 이동하면 된다.

1. [Client: Client.java] – addStudent

텍스트, 모니터, 스크린샷, 닫기이(가) 표시된 사진

자동 생성된 설명

학번, 비밀번호, 이름, 학과, 선이수 과목을 입력하여 Server에 값을 제공하고, List로 결과 값을 받는다. 이때, 중요한 것은 Token 값을 무조건 넣어줘야한다는 점이다.

1. [Server: Server.java] – addStudent

텍스트이(가) 표시된 사진

자동 생성된 설명

앞서 설명했듯, userToken과 logUser의 학번이 일치하는지 확인하는 작업을 하고, 이미 저장되어 있는 학번인지 확인한다. 이 때, 학번이 일치하지 않거나, 이미 저장되어 있는 학번이라면 Server가 Client에게 exception을 제공한다.

1. [Server: Server.java] – checkAccessUser

텍스트이(가) 표시된 사진

자동 생성된 설명

UserToken의 경우 암호화를 통해 사용자의 학번을 저장해둔 것이다. 이 때, 받아온 token의 값과 이미 저장된 logUser(학번)의 값이 동일하다면 true를 제공한다.

1. [Data: Data.java] – checkStudent

& [Data: StudentList.java] - isRegisteredStudent

(2) - – c와 중복된 내용으로 넘어가고자 한다

1. [Data: Data.java] – addStudent

텍스트이(가) 표시된 사진

자동 생성된 설명

회원가입과 마찬가지로 File에 저장하는 것은 같은데, 다른 메소드를 사용하는 이유는 받아오는 학생의 정보 때문이다. 회원가입의 경우 정보를 각각의 속성으로 불러오는데, 학생 등록의 경우 하나의 String으로 불러온다. 하나로 합칠 수도 있었지만, 각각의 경우의 수가 있다는 것을 확인하고자 따로 생성하게 되었다.

1. [Data: StudentList.java] – addStudentRecords

텍스트이(가) 표시된 사진

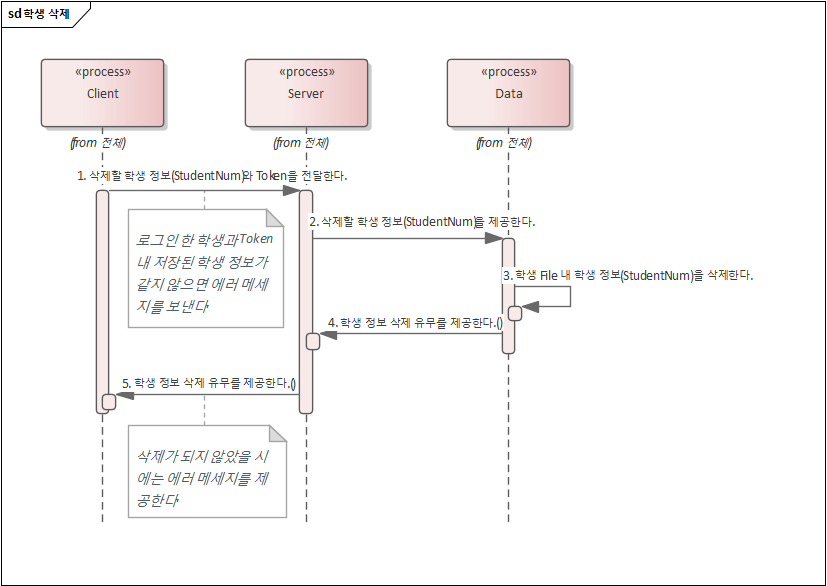
자동 생성된 설명

학생 정보가 없다면 예외가 발생하게 된다. 또한, List에만 저장하는 것이 아닌 File에도 저장해야 하기 때문에 addCourseFile Method를 실행한다. 따로 extract method로 뺀 이유는 공통으로 사용할 경우가 많아 코드의 중복을 피하기 위함이다. addCourseFile의 경우 회원가입 서비스 때에도 사용했던 method이다.

1. [Data: StudentList.java] – addCourseFile

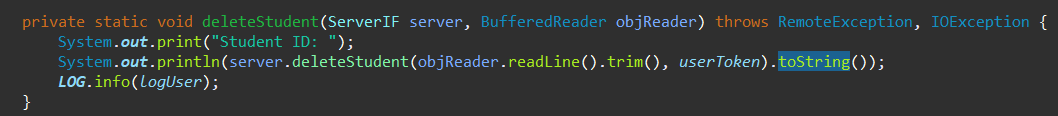
(2) - - f에서 설명했기 때문에 중복되어 설명은 생략한다.

학생 삭제



학생 삭제의 경우, 수업시간에서 List 내부만 삭제했던 것과 달리 File 내에서도 적용을 하고자 했다. 이에, File 내부의 정보를 삭제할 때, Filewriter를 사용했다. 하지만 데이터 한 줄만 삭제할 수 없기 때문에 정보와 일치하는 List의 값을 삭제하고, File에 새로 덮어 씌우는 형식으로 진행했는데, 완전히 효율적인 방법은 아니라고 생각한다.

1. [Client: Client.java] – deleteStudent



삭제하고자 하는 학생의 학번 정보를 받아, token 값과 함께 Server로 전달한다. Result의 값을 받아서, 성공했는지, 실패했는지를 console에 작성한다.

1. [Server: Server.java] – deleteStudent

텍스트이(가) 표시된 사진

자동 생성된 설명

여기서는 token 확인 이외에 따로 예외 처리할 필요 없이 데이터에 삭제하고자 하는 정보를 제공한다.

1. [Server: Server.java] – checkAccessUser

(3) - - d에서 공통된 코드 설명을 했으므로, 생략한다.

1. [Data: Data.java] – deleteStudent

텍스트이(가) 표시된 사진

자동 생성된 설명

학생 삭제는 studentList.deleteStudentRecords에서 진행한다. 위 코드는 studentList와 Server를 연결해주는 중간 지점 역할을 한다.

1. [Data: StudentList.java] – deleteStudentRecords

텍스트이(가) 표시된 사진

자동 생성된 설명

List 안에 잇는 학생의 정보와 parameter로 받은 학생 정보가 일치하면 list의 값을 삭제하고 deleteStudentFile method를 실행한다. 하지만 같은 정보가 없다면 false를 return 한다.

1. [Data: StudentList.java] – deleteStudentFile

텍스트이(가) 표시된 사진

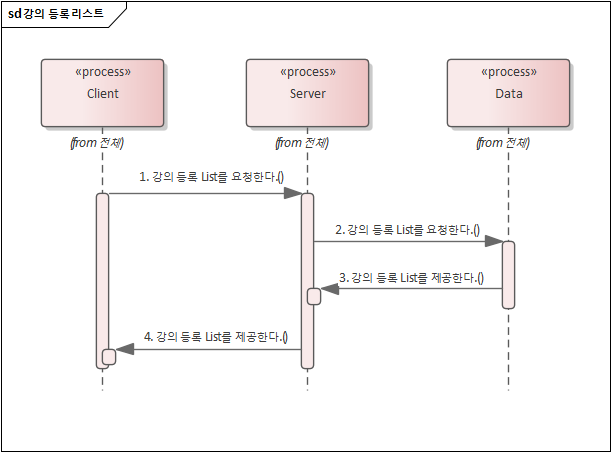
자동 생성된 설명

BufferedWriter를 사용하여 File 안에 있는 값을 모두 지운 다음, vStudentList에 있는 정보를 모두 File에 저장한다.

1. 강의 정보

강의 정보 서비스 구현도 파일의 위치와 안에 들어갈 정보들만 달라졌을 뿐, 학생 정보와 구조는 완전히 같다. 따라서 코드 및 설명을 간단하게 작성하고자 한다.

강의 리스트



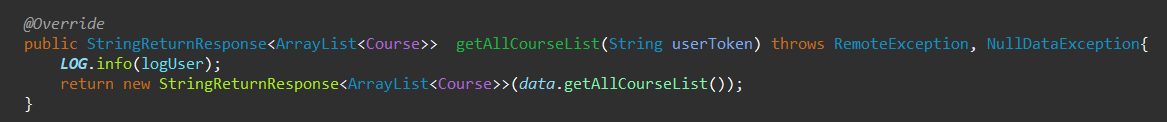
강의 리스트를 불러오는 메소드를 호출하게 되면, 학생 리스트와 같이 따로 Token 값을 전달해주더라도 token과 사용자가 일치하는지 확인하지 않는다. 그 이유는 앞 학생 리스트와 같은 이유기 때문에 생략한다.

1. [Client: Client.java] – registerCourseMenu & showData

registerCourseMenu의 경우 (3)--b와, showData의 경우 (3) – 1 – a와 동일한 코드이므로 코드 사진은 생략한다.

강의 정보 리스트를 불러와야 하기 때문에 Client에서 호출하는 Server 메소드 중 server.getAllCourseList 함수를 호출한다.

1. [Server: Server.java] – getAllCourseList



Server에서는 NullException 처리가 되어 있으므로 따로 Exception 처리를 해두지 않았다. Data에서 값이 없는 것이 확인되면 에러가 발생한다.

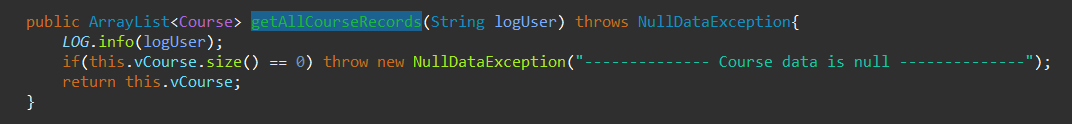
1. [Data: Data.java] – getAllCourseList

텍스트이(가) 표시된 사진

자동 생성된 설명

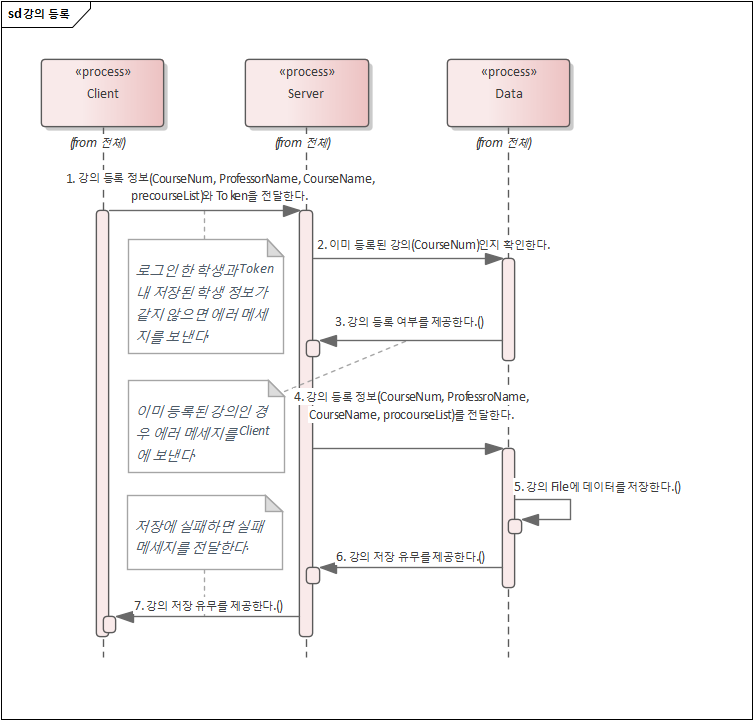
저장된 강좌 List에서 값을 불러와서 Server로 return, Server는 Client로 return 해준다.

1. [Data: CourseList.java] – getAllCourseRecords



List에 데이터 값이 없으면 exception 처리를 해주고 아니면 값을 return 해주도록 설계했다.

강의 등록



강의 등록은 다음과 같은 순서로 진행된다. 사용자가 입력한 강의 정보와 token을 Server에 전달해준다. 이후, token과 이미 등록된 강의인지 아닌지 확인하고, 등록되지 않은 강의라면 강의를 등록할 정보를 Data에 전달해준다. Data에서는 courseList 뿐만 아니라 File에서도 값을 저장해준다. 각각의 메소드들이 진행될 때, 원하는 결과물이 도출되지 않는다면 에러 메시지를 return 해준다.

1. [Client: Client.java] – registerCourseMenu

registerCourseMenu의 경우 (3)--b와 동일하기 때문에 생략한다. Method 내에서 addCourse로 이동하면 된다.

1. [Client: Client.java] – addCourse

텍스트이(가) 표시된 사진

자동 생성된 설명

강의 번호, 강의 이름, 교수 이름, 선이수 과목을 입력하여 Server에 값을 제공하고, List로 결과 값을 받는다. 이때, 중요한 것은 Token 값을 무조건 넣어줘야한다는 점이다.

1. [Server: Server.java] – addCourse

텍스트이(가) 표시된 사진

자동 생성된 설명

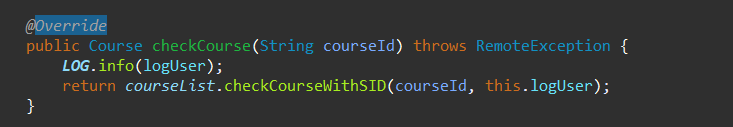
앞서 설명했듯, userToken과 logUser의 학번이 일치하는지 확인하는 작업을 하고, 이미 저장되어 있는 강의번호인지 확인한다. 이 때, token과 학번이 일치하지 않거나, 이미 저장되어 있는 강의 번호라면 Server가 Client에게 exception을 제공한다.

성공적으로 실행이 되었다면 직접 데이터에 추가하는 addCourse로 이동한다.

1. [Data: Data.java] – checkAccessUser

(3) - - d에서 공통된 코드 설명을 했으므로, 생략한다.

1. [Data: Data.java] – checkCourse



위 코드는 courseList와 Server를 연결해주는 중간 지점 역할을 한다.

1. [Data: CourseList.java] – checkCoursewithSID

텍스트이(가) 표시된 사진

자동 생성된 설명

Course가 이미 저장되어 있었는지 확인하는 코드이다. 강의 등록 시, 중복되어 강의가 입력되면 안 되기 때문이다.

1. [Data: Data.java] – addCourse

텍스트이(가) 표시된 사진

자동 생성된 설명

학생 삭제는 courseList.addCourseRecords에서 진행한다. 위 코드는 courseList와 Server를 연결해주는 중간 지점 역할을 한다.

1. [Data: CourseList.java] – addCourseRecords

텍스트이(가) 표시된 사진

자동 생성된 설명

List에서 저장하기 위한 method이다. 이 후, List에 저장이 되었다면, addCourseFile 로 이동하여 file에도 저장을 해줘야한다.

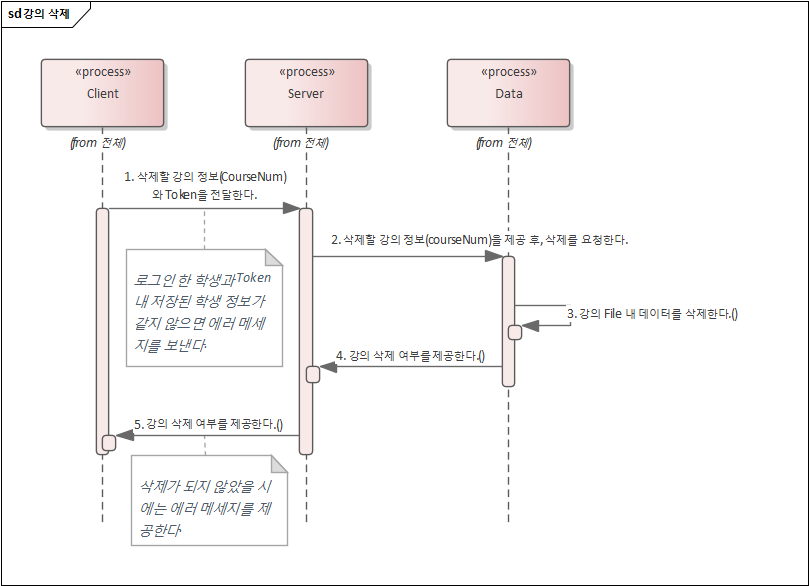
1. [Data: CourseList.java] – addCourseFile

텍스트이(가) 표시된 사진

자동 생성된 설명

BufferedWriter를 사용하여 기존에 저장된 TextFile에 강의 정보를 붙여 저장해둔다.

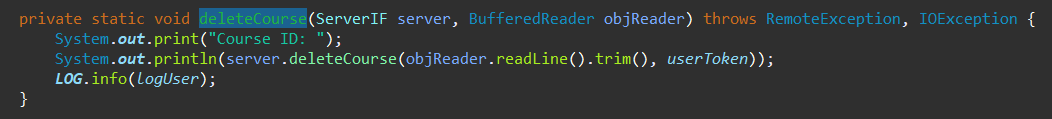
강의 삭제



1. [Client: Client.java] – registerCourseMenu

registerCourseMenu의 경우 (3)--b와 동일하기 때문에 생략한다. Method 내에서 deleteCourse로 이동하면 된다.

1. [Client: Client.java] – deleteCourse



사용자로부터 삭제하고 싶은 강의 번호를 받고 로그인/회원 가입시 받은 Token과 함께 Server로 전달해준다.

1. [Server: Server.java] – deleteCourse

텍스트이(가) 표시된 사진

자동 생성된 설명

Token과 LogUser 간의 정보를 확인하는 코드와 실제 삭제하는 코드를 작성하여 문제가 발생하면 Client로 예외 메시지를 넘겨준다.

1. [Server: Server.java] – checkAccessUser

(3) - - d에서 공통된 코드 설명을 했으므로, 생략한다.

1. [Data: Data.java] – deleteCourse

텍스트이(가) 표시된 사진

자동 생성된 설명

강의 삭제는 courseList.deleteCourseRecords에서 진행한다. 위 코드는 courseList 와 Server를 연결해주는 중간 지점 역할을 한다.

1. [Data: CouseList.java] – deleteCourseRecords

텍스트이(가) 표시된 사진

자동 생성된 설명

List 안에 있는 강의의 정보와 parameter로 받은 강의 정보가 일치하면 list의 값을 삭제하고 deleteCoursetFile method를 실행한다. 위 method는 List 안에 있는 정보를 삭제해주는 method이고, File에 있는 정보를 삭제하기 위해서는 deleteCourseFile 호출이 필요하다.

1. [Data: CouseList.java] – deleteCourseFile

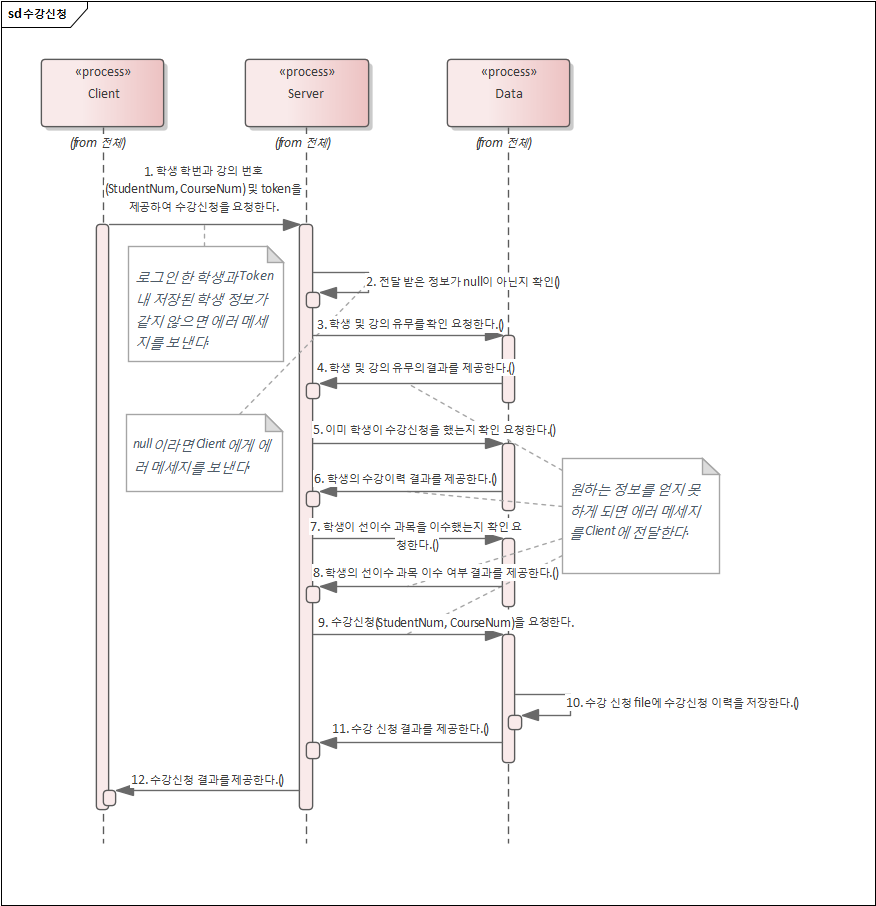
텍스트이(가) 표시된 사진

자동 생성된 설명

BufferedWriter를 사용하여 File 안에 있는 값을 모두 지운 다음, vCourseList에 있는 정보를 모두 File에 저장한다.

1. 수강 신청

수강 신청



수강신청을 설계할 때에는 많은 예외 처리를 생각하게 되었다. 매 서비스마다 Token 비교하는 것을 제외하고도, 학생, 강좌가 실제로 있는 정보인지 있다면 이미 강의를 수강하지는 않았는지 혹은 선 이수 과목을 이수했는지를 예외 처리했다. 모두 다 false로만 Client에게 제공하지 않고, 왜 에러가 났는지에 대한 이유도 같이 적어서 보냈다.

1. [Client: Client.java] – registerCourseMenu

registerCourseMenu의 경우 (3)--b와 동일하기 때문에 생략한다. Method 내에서 deleteCourse로 이동하면 된다.

1. [Client: Client.java] - makeReservation

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자의 학번과 강의 번호를 받아 Server에게 수강신청 서비스를 요청한다.

1. [Server: Server.java] - addReservation

텍스트, 모니터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

위 코드를 확인해보면 많은 예외 처리 상황을 볼 수 있다. checkAccessUser와 checkStudent, checkCourse는 이미 위에서 확인할 수 있으므로 생략하도록 한다. for문은 강좌를 수강했는지 확인하는 코드이다.

1. [Server: Server.java] – checkPrecourse

텍스트이(가) 표시된 사진

자동 생성된 설명

학생이 수강한 과목과 선 이수 과목을 서로 비교하여 선 이수 과목을 모두 수강했는지 확인하는 함수이다.

1. [Data: Data.java] – checkReservation

텍스트, 실내, 스크린샷, 닫기이(가) 표시된 사진

자동 생성된 설명

checkReservation은 reservationList.isRegisteredStudent에서 진행한다. 위 코드는 reservationList 와 Server를 연결해주는 중간 지점 역할을 한다.

1. [Data: ReservationList.java] – isRegisteredStudent

텍스트이(가) 표시된 사진

자동 생성된 설명

이미 수강신청을 하였다면, 또 다시 중복되어 문제가 발생할 수 있으므로 이를 확인해준다.

1. [Data: Data.java] – addReservation

텍스트이(가) 표시된 사진

자동 생성된 설명

addReservation은 reservationList.deleteCourseRecords에서 진행한다. 위 코드는 reservationList 와 Server를 연결해주는 중간 지점 역할을 한다.

1. [Data: ReservationList.java] - checkReservation

텍스트이(가) 표시된 사진

자동 생성된 설명

받아온 값이 없다면 예외처리를 진행하고, list에 값을 저장한 후, File 내에도 값을 저장하기 위해 addReservationFile로 이동한다.

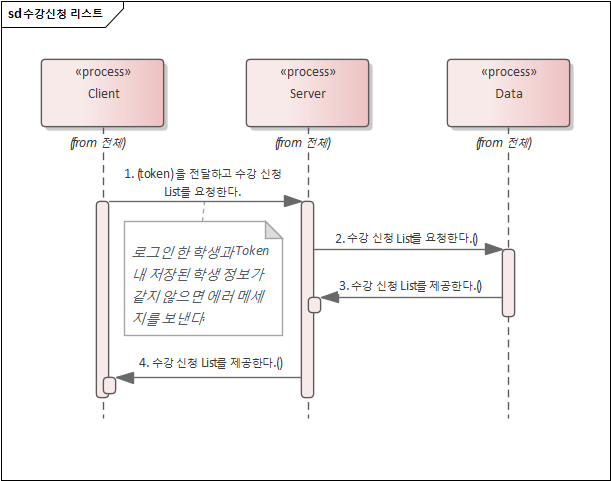
1. [Data: ReservationList.java] - addReservationFile

텍스트이(가) 표시된 사진

자동 생성된 설명

BufferedWriter를 통해 File에 값을 추가한다.

수강 신청 리스트

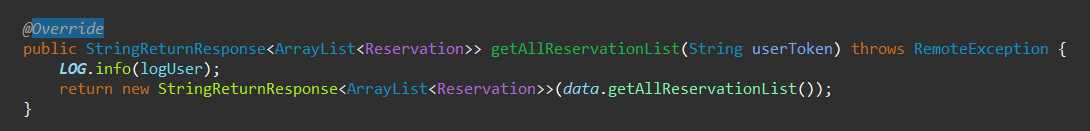


수강 신청 리스트는 학생, 강의 리스트와 메커니즘이 같다. 하지만, 수강 신청은 개인적인 정보가 들어가기 때문에 Token으로 받아오는 코드를 추가했다. 자세한 설명은 아래 코드 설명하면서 진행할 할 예정이다.

1. [Client: Client.java] – registerCourseMenu

registerCourseMenu의 경우 (3)--b와 동일하기 때문에 생략한다.

1. [Server: Server.java] – getAllReservationList



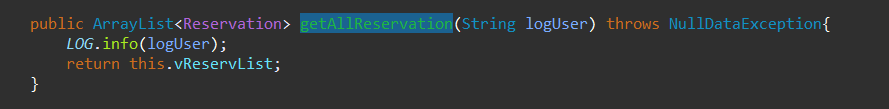
Server에서는 NullException 처리가 되어 있으므로 따로 Exception 처리를 해두지 않았다. Data에서 값이 없는 것이 확인되면 에러가 발생한다.

1. [Data: Data.java] – getAllReservationList

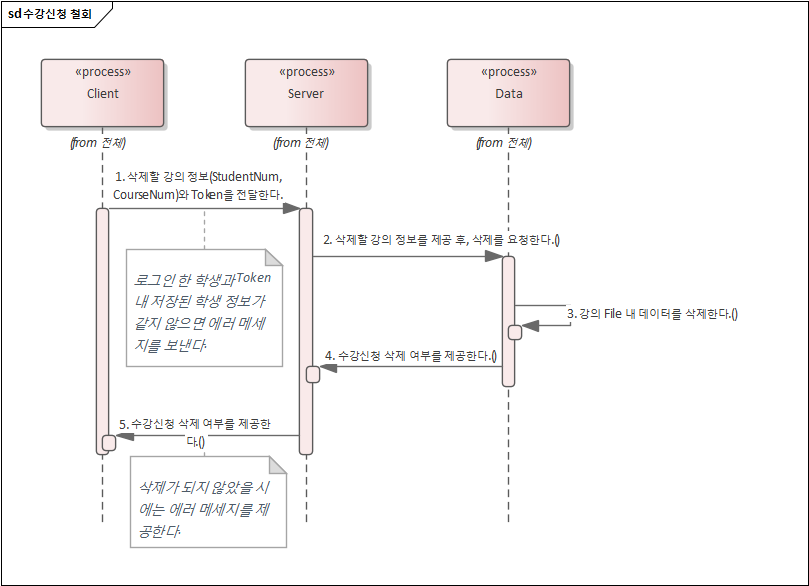
텍스트이(가) 표시된 사진

자동 생성된 설명

1. [Data: Data.java] – getAllReservation



수강신청 삭제



수강신청 삭제는 학생 삭제, 강의 삭제와 비슷하다. 하지만, 수강신청 삭제는 input 값이 학번, 강의 번호 각각 한 개가 아닌 두 개를 다 비교해야 한다는 점이 다르다.

1. [Client: Client.java] – registerCourseMenu

registerCourseMenu의 경우 (3)--b와 동일하기 때문에 생략한다.

1. [Client: Client.java] – deleteReservation

텍스트이(가) 표시된 사진

자동 생성된 설명

학생과 과목 정보를 받아 Server에 전달한다.

1. [Server: Server.java] – deleteReservation

텍스트이(가) 표시된 사진

자동 생성된 설명

Token과 logUser의 비교 후, 삭제할 정보를 Data에 전달한다.

1. [Data: Data.java] – deleteReservation

텍스트이(가) 표시된 사진

자동 생성된 설명

1. [Data: ReservationList.java] – deleteReservationRecords

텍스트이(가) 표시된 사진

자동 생성된 설명

List에 저장된 값과 User가 전달한 값이 같으면 삭제를 하고 deleteStudentFile을 호출하여 File에서도 삭제가 가능하도록 한다.

1. [Data: ReservationList.java] – deleteReservationFile

텍스트이(가) 표시된 사진

자동 생성된 설명

BufferedWriter를 사용하여 File 안에 있는 값을 모두 지운 다음, vReservationList에 있는 정보를 모두 File에 저장한다.

1. 프로젝트 실행 결과

* 초기 세팅

<Data>

텍스트이(가) 표시된 사진

자동 생성된 설명

<Server>

텍스트이(가) 표시된 사진

자동 생성된 설명

<Client>

텍스트이(가) 표시된 사진

자동 생성된 설명

* SignUp/Login

|  |  |
| --- | --- |
| SignUp | Login |
| * 성공   텍스트이(가) 표시된 사진  자동 생성된 설명   * Exception 처리   <이미 가입한 경우>    - File 저장 | * 성공      * Exception 처리   <정보 잘못 입력한 경우>    <null 값 입력한 경우> |

* 학생 정보

|  |  |  |
| --- | --- | --- |
| 학생 리스트 | 학생 등록 | 학생 삭제 |
| * 성공 | * 성공      * Exception 처리   <이미 가입한 경우>     * File 저장 | * 성공     60201977 삭제 확인     * Exception 처리   <이미 삭제된 경우 🡪 60201977>     * File 삭제 확인 |

* 강의 정보

|  |  |  |
| --- | --- | --- |
| 강의 리스트 | 강의 등록 | 강의 삭제 |
| * 성공 | * 성공        * Exception 처리   <이미 개설된 강좌 번호>     * File 저장 | * 성공      * Exception 처리   <이미 삭제된 경우>     * File 삭제 확인 |

* 수강신청 정보

|  |  |  |
| --- | --- | --- |
| 수강신청 리스트 | 수강 신청 등록 | 수강 내역 삭제 |
| * 성공 | * 성공        * Exception 처리   <값이 없는 경우>    <이미 수강신청 한 경우>    <강좌 번호가 없는 경우>    <학생 정보가 없는 경우>    <이미 수강한 과목일 경우>    <선 이수 과목을 이수하지 않은 경우>     * File 저장 | * 성공      * Exception 처리   <이미 삭제된 경우>         * File 삭제 확인 |

* Log File 생성 예시

<Client.log>

START LOG

2022-11-06 20:30 [Client: showInitialView] [사용자 정보] 사용자 정보 없음

2022-11-06 20:31 [Client: login] [사용자 정보] 60201976

2022-11-06 20:31 [Client: showMenu] [사용자 정보] 60201976

2022-11-06 20:31 [Client: showData] [사용자 정보] 60201976

2022-11-06 20:31 [Client: showMenu] [사용자 정보] 60201976

2022-11-06 20:31 [Client: showData] [사용자 정보] 60201976

2022-11-06 20:31 [Client: showMenu] [사용자 정보] 60201976

2022-11-06 20:31 [Client: showInitialView] [사용자 정보] 사용자 정보 없음

2022-11-06 20:31 [Client: login] [사용자 정보] 20130095

2022-11-06 20:31 [Client: showMenu] [사용자 정보] 20130095

2022-11-06 20:32 [Client: addCourse] [사용자 정보] 20130095

2022-11-06 20:32 [Client: showMenu] [사용자 정보] 20130095

2022-11-06 20:32 [Client: makeReservation] [사용자 정보] 20130095

2022-11-06 20:32 [Client: showMenu] [사용자 정보] 20130095

2022-11-06 20:32 [Client: makeReservation] [사용자 정보] 20130095

2022-11-06 20:32 [Client: showMenu] [사용자 정보] 20130095

2022-11-06 20:32 [Client: showData] [사용자 정보] 20130095

2022-11-06 20:32 [Client: showMenu] [사용자 정보] 20130095

2022-11-06 20:33 [Client: deleteReservation] [사용자 정보] 20130095

2022-11-06 20:33 [Client: showMenu] [사용자 정보] 20130095

2022-11-06 20:33 [Client: deleteReservation] [사용자 정보] 20130095

2022-11-06 20:33 [Client: showMenu] [사용자 정보] 20130095

2022-11-06 20:33 [Client: showInitialView] [사용자 정보] 사용자 정보 없음

<Server.log>

START LOG

2022-11-06 20:31 [Server: login] [사용자 정보] null

2022-11-06 20:31 [Server: sendSeverStudentForLog] [사용자 정보] 60201976

2022-11-06 20:31 [Server: getAllStudentData] [사용자 정보] 60201976

2022-11-06 20:31 [Server: getAllCourseList] [사용자 정보] 60201976

2022-11-06 20:31 [Server: login] [사용자 정보] 60201976

2022-11-06 20:31 [Server: sendSeverStudentForLog] [사용자 정보] 20130095

2022-11-06 20:32 [Server: addCourse] [사용자 정보] 20130095

2022-11-06 20:32 [Server: addReservation] [사용자 정보] 20130095

2022-11-06 20:32 [Server: checkPrecourse] [사용자 정보] 20130095

2022-11-06 20:32 [Server: addReservation] [사용자 정보] 20130095

2022-11-06 20:32 [Server: checkPrecourse] [사용자 정보] 20130095

2022-11-06 20:32 [Server: getAllReservationList] [사용자 정보] 20130095

2022-11-06 20:33 [Server: deleteReservation] [사용자 정보] 20130095

2022-11-06 20:33 [Server: deleteReservation] [사용자 정보] 20130095

<Data.log>

START LOG

2022-11-06 20:31 [Data: checkLogin] [사용자 정보] null

2022-11-06 20:31 [StudentList: checkStudent] [사용자 정보] null

2022-11-06 20:31 [Data: sendServerStudentForLog] [사용자 정보] 60201976

2022-11-06 20:31 [Data: getAllStudentData] [사용자 정보] 60201976

2022-11-06 20:31 [StudentList: getAllStudentRecords] [사용자 정보] 60201976

2022-11-06 20:31 [Data: getAllCourseList] [사용자 정보] 60201976

2022-11-06 20:31 [CourseList: getAllCourseRecords] [사용자 정보] 60201976

2022-11-06 20:31 [Data: checkLogin] [사용자 정보] 60201976

2022-11-06 20:31 [StudentList: checkStudent] [사용자 정보] 60201976

2022-11-06 20:31 [Data: sendServerStudentForLog] [사용자 정보] 20130095

2022-11-06 20:32 [Data: checkCourse] [사용자 정보] 20130095

2022-11-06 20:32 [CourseList: checkCourseWithSID] [사용자 정보] 20130095

2022-11-06 20:32 [Data: addCourse] [사용자 정보] 20130095

2022-11-06 20:32 [CourseList: addCourseRecords] [사용자 정보] 20130095

2022-11-06 20:32 [CourseList: addCourseFile] [사용자 정보] 20130095

2022-11-06 20:32 [Data: checkStudent] [사용자 정보] 20130095

2022-11-06 20:32 [StudentList: isRegisteredStudent] [사용자 정보] 20130095

2022-11-06 20:32 [Data: checkCourse] [사용자 정보] 20130095

2022-11-06 20:32 [CourseList: checkCourseWithSID] [사용자 정보] 20130095

2022-11-06 20:32 [Data: checkStudent] [사용자 정보] 20130095

2022-11-06 20:32 [StudentList: isRegisteredStudent] [사용자 정보] 20130095

2022-11-06 20:32 [Data: checkCourse] [사용자 정보] 20130095

2022-11-06 20:32 [CourseList: checkCourseWithSID] [사용자 정보] 20130095

2022-11-06 20:32 [Data: checkReservation] [사용자 정보] 20130095

2022-11-06 20:32 [ReservationList: isRegisteredStudent] [사용자 정보] 20130095

2022-11-06 20:32 [Data: addReservation] [사용자 정보] 20130095

2022-11-06 20:32 [ReservationList: addReservationRecords] [사용자 정보] 20130095

2022-11-06 20:32 [ReservationList: addReservationFile] [사용자 정보] 20130095

2022-11-06 20:32 [Data: getAllReservationList] [사용자 정보] 20130095

2022-11-06 20:32 [ReservationList: getAllReservation] [사용자 정보] 20130095

2022-11-06 20:33 [Data: deleteReservation] [사용자 정보] 20130095

2022-11-06 20:33 [ReservationList: deleteReservationRecords] [사용자 정보] 20130095

2022-11-06 20:33 [Data: deleteReservation] [사용자 정보] 20130095

2022-11-06 20:33 [ReservationList: deleteReservationRecords] [사용자 정보] 20130095

1. 결론

이번 프로젝트에서 가장 아쉬웠던 점은 Token 생성 기능이었다. 프로젝트에서 Spring의 Json Web Token과 동일한 구조의 Token을 생성하고 싶었지만, 완전한 기능을 만들지 못했다는 점이 아쉬웠다. 하지만, 이번 프로젝트에서 생성한 Token은 암호화를 도와주는 AES를 활용하여 외부에 studentNum이 노출되지 않으면서도 복호화 시, 학생의 정보를 확인할 수 있도록 하여 JWT와 최대한 비슷하게 구현했다.

또한, RMI로만 실습을 진행하고 grpc를 사용해보지 못했던 점이 아쉬웠다. 현업에서는 RMI보다 grpc를 더욱더 선호하고 많이 사용한다고 하기에, 앞으로 RMI를 완전히 익히면 grpc도 사용해보고자 한다.

그리고, File에서 원하는 정보를 삭제할 때, 파일에 있는 모든 데이터를 지우고 다시 다 작성하는 부분이 비효율적이라는 생각이 들었다. BufferedWriter로 만들어 아직까지 비효율적인 문제점을 해결하지 못했지만 앞으로 공부를 하며 해결해나가고자 한다.

RMI 수강신청 프로젝트는 프로젝트 개발 및 EA, 보고서까지 작성하면서 Client와 Server 간의 정보 요청 및 제공의 메커니즘을 명확하게 알게 된 좋은 프로젝트였다. Client와 Server의 통신에 관심을 갖고 인터넷을 검색하더라도 글로만 작성되어 있는 경우가 더러 있어 쉽게 이해할 수 없는 개념들도 많았지만 이번 프로젝트를 통해 많은 것들을 알게 되었다. 또한, 하나의 프로젝트 생성으로 연결된 것이 아닌 RMI Registry를 통해 연결되어 사용자의 요청 서비스가 화면에 보이게 된다는 점이 신기했다.

교수님 과의 멘토링 활동은 method 이름을 명확하게 사용해야하는 점과 같은 피드백을 다시 되새기며 프로젝트에 임해 여태까지 홀로 공부하면서 작성한 다른 코드보다 훨씬 더 가독성이 좋다라는 생각을 갖게 되는 시간이었다.