

REPORT

협동분산시스템

최종 보고서 (Final report)



과목명 | 협동분산시스템

담당교수 | 임민규 교수님

학과 | 컴퓨터공학부

소속 | 7 팀

팀원 | 201715007 안지호

201714154 오병현

201714150 김동진

201714151 박민기

201714158 허승희

주제 | 경매 프로그램

목 차

1. Team member information and role	3
2. Project goal	4
3. Database	5
3.1 데이터베이스 접근 방법	5
3.2 생성 결과	6
3.3 데이터베이스 구성 요소	6
4. Class Diagram	8
4.1 Client Side	8
4.2 Server Side	9
5. Project design	10
5.1 시나리오 및 CM API	10
5.1.1 회원 가입	10
5.1.2 로그인	16
5.1.3 경매 목록	26
5.1.4 입찰	31
5.1.5 등록	43
5.1.6 갱신	52
5.1.7 낙찰	54
6 Project result	60
7 Video and Github URL	60

1. Team member information and role

프로젝트 역할 분담은 크게, GUI 구현과 서버 및 클라이언트에 필요한 기능 구현으로 분할하였다.

본 프로젝트는 경매 프로그램을 제작하는 것이며, 서버, 클라이언트가 제공하는 기능은 2장 Project goal 부분에서 자세하게 소개하였다. 팀원 정보 및 역할 분담 내용은 아래와 같다.

- 안지호(조장) (17%) : 회원가입 및 로그인 구현
- 오병현 (20%) : 경매 목록 리스트 출력 구현
- 김동진 (20%) : 경매 목록 등록 구현
- 박민기 (20%) : 입찰 프로세스 및 경매 목록 갱신 구현
- 허승희 (23%) : 경매 진행 상황 및 낙찰 알림 구현, 영상 녹화
- 공통 : GUI 구현

GUI 를 제외한 나머지 기능들에 대해서는 서버와 클라이언트에서 동시에 구현해야 되는 부분이 있기 때문에, 조원 간 클라이언트 부분과 서버 부분 분배를 따로 하지는 않았다.

2. Project goal

본 프로젝트의 목표는 CM 을 이용하여 Client - Server 기반으로 하는 경매 프로그램을 구축하는 것이다.

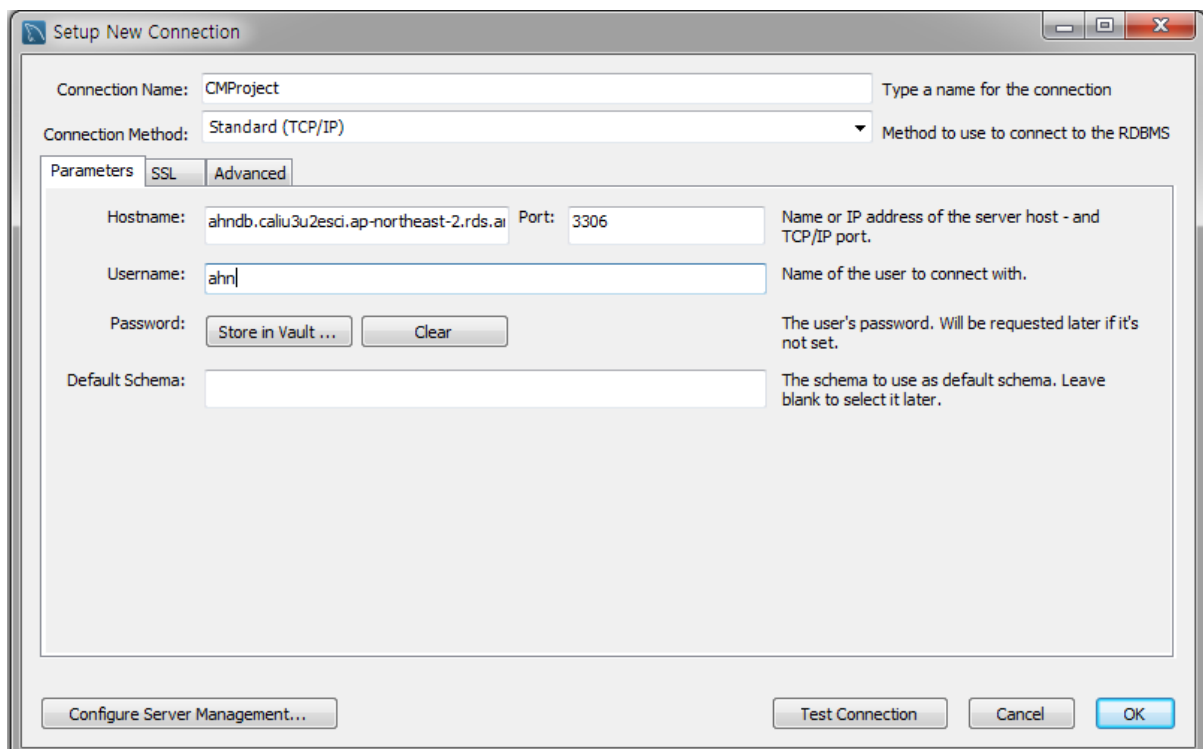
먼저 Server 에서는 크게 세 가지 기능을 지원한다. 첫 번째로 회원 관리이다. 경매 프로그램을 이용하기 전 사용자가 회원가입을 할 때, 각 입력 정보에 대한 유효성 검사를 한다. 유효성 검사를 마친 후 클라이언트 측에서 회원정보가 전송되면 Server 에서 저장하고 관리한다. 이후에 사용자가 로그인을 할 때, 인증과정을 거친다. 두 번째로 경매 목록 관리 기능을 지원한다. 사용자가 입찰 or 등록을 하거나 물품이 낙찰될 경우, 경매 목록을 업데이트 하고, 이 정보를 클라이언트로 전송을 해준다. 세 번째로 메시지 알림 기능이다. 경매 물품이 낙찰될 경우, 낙찰자에게 해당 물품이 낙찰되었다는 정보를 알림으로 전송한다. 낙찰이 되지 않고 종료될 경우, 물품 등록자에게 물품이 반환되었음을 알린다.

Client 에서는 GUI 를 통해 회원가입, 로그인 서비스 및 사용자가 경매에 참여할 수 있도록 여러가지 경매 서비스를 제공해준다. 첫 번째로 회원가입을 할 때, 사용자가 서버로 전송할 회원정보를 입력 할 수 있도록 입력 form 을 제공해준다. 두 번째로 로그인 기능이다. 사용자는 회원가입한 정보로 로그인을 하여 경매장에 입장할 수 있다. 세 번째로 경매장에 입장하면 경매목록을 사용자에게 출력해준다. 여기서 사용자는 원하는 경매 물품에 대해 등록된 물품 설명서를 확인할 수 있으며, 입찰을 하거나 새로운 경매 물품을 등록할 수 있다. 사용자가 물품 입찰을 원할 경우, 해당 물품의 현재 최고 입찰가를 초과하는 금액으로 입찰을 요청할 수 있다. 등록을 원할 경우, 사용자는 등록할 물품에 대한 이름, 설명, 마감일, 입찰 시작가에 대한 정보를 입력하여 물품을 등록할 수 있다.

3. Database

본 프로젝트에서는 조원들이 하나의 공통된 데이터베이스를 사용하기 때문에 아마존에서 제공하는 RDS 데이터베이스를 사용하여, 공통으로 데이터베이스에 접근하여, 변경된 내용을 참조할 수 있도록 하였다. 사용한 데이터베이스의 접근 방법 및 구성 요소 등 자세한 내용은 아래에 기재하였다.

3.1 데이터베이스 접근 방법



MySQL Workbench 프로그램을 실행하여, 초기 화면에서 Setup New Connection 아이콘을 클릭하면 위와 같은 창이 클릭한다. 이 때, Connection Name은 원하는 이름으로 입력을 해주고, Hostname, Username, Password 필드는 아래와 같이 입력한다.

Hostname : ahndb.caliu3u2esci.ap-northeast-2.rds.amazonaws.com (아마존 RDS 주소)

Username : ahn

Password : guqqnstks123(영어로 협분산123)

3.2 생성 결과

MySQL Connections



위와 같이 정상적으로 데이터베이스가 생성된 사실을 확인할 수 있다.

3.3 데이터베이스 구성 요소

기존 CM 프로젝트의 덤프 파일을 사용하므로, 현재 우리 프로젝트에서 사용하는 테이블에 대한 구성만 설명하였다.

① user_table




Table Name:

Schema: **cmdb**





Charset/Collation:

utf8

utf8_bin

Engine: **InnoDB**

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 seqNum	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 userName	VARCHAR(80)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 password	VARCHAR(80)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 creationTime	DATETIME(1)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

기존 CM Project 에서 사용하는 테이블로 구성은 아래와 같다.

- 1) seqNum (일련번호) : 기본키, Not Null, Auto Increment
- 2) userName (클라이언트 id) : Not Null
- 3) password (비밀번호)
- 4) creationTime (생성일시)

② item




Table Name:

Schema: **cmdb**

Charset/Collation:









utf8

utf8_bin

Engine:

InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
 no	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
 name	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 start_price	BIGINT(20)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 due_date	DATETIME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
 description	VARCHAR(150)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 now_price	BIGINT(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 status	VARCHAR(20)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
 bid_winner	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

현 프로젝트에서 기존에 존재한 cmdb schema 에 새로 추가한 테이블이다. 구성은 아래와 같다.

1) no (물건 번호) : 기본키, Auto Increment, Not Null

2) name (물건 명) : Not Null

3) start_price (경매 시작가) : Not Null

4) due_date (경매 마감 기한) : Not Null

5) description (물건 설명)

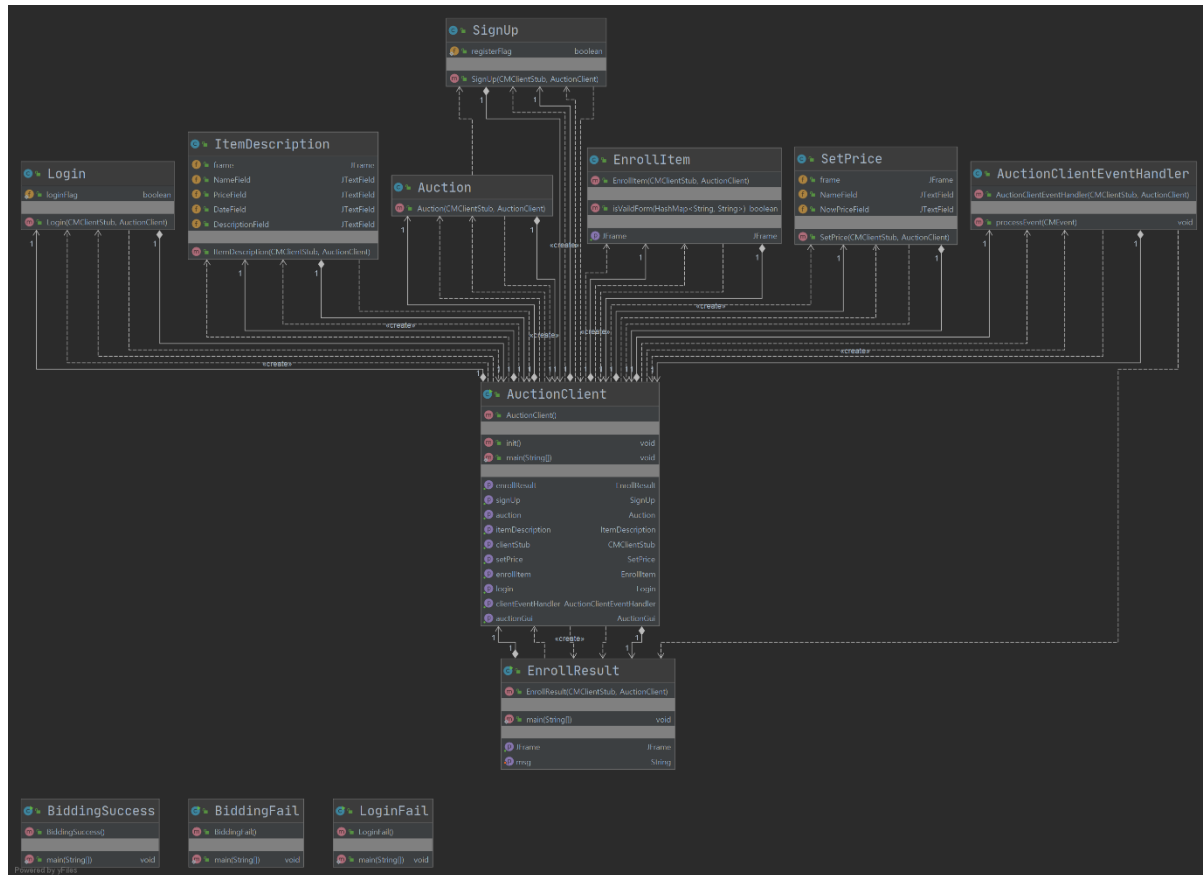
6) now_price (경매 현재가)

7) status (경매 진행 상태) : 't' 또는 'f' 중 하나의 값을 가지며, 't'는 낙찰(경매 종료), 'f'는 경매 진행 중을 의미한다.

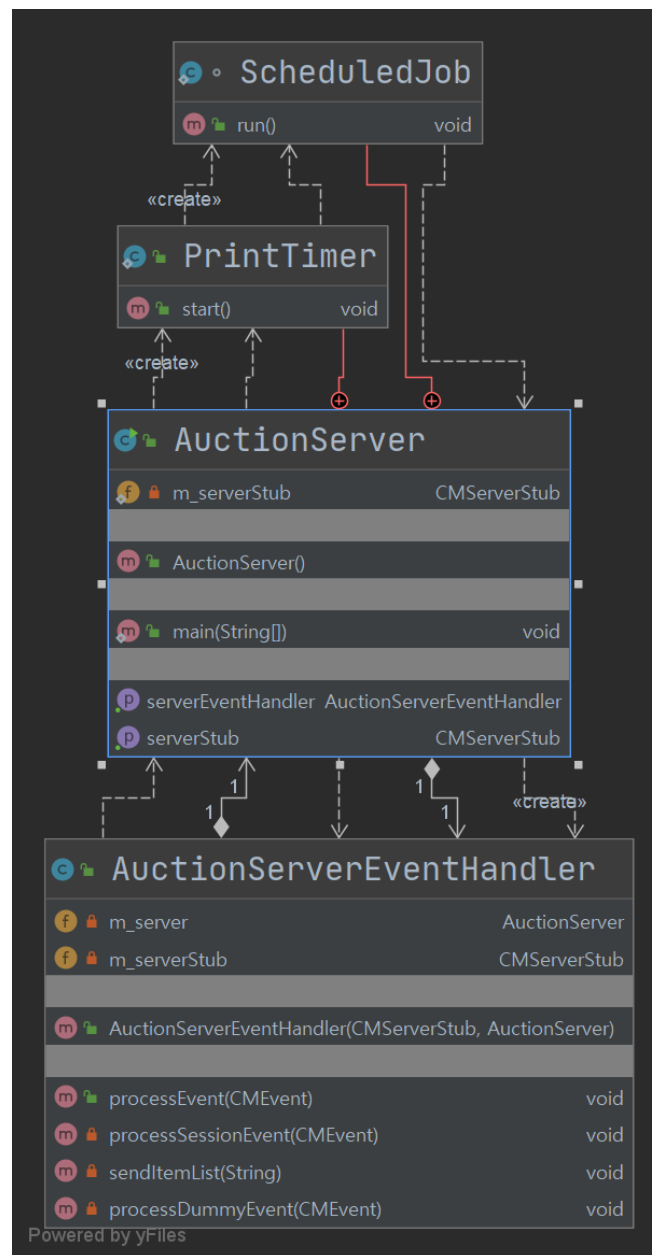
8) bid_winner (낙찰자) : 최초에서 경매 물품을 등록한 client id로 지정되며, 경매 진행 중에는 최고 입찰가를 적용한 client id로 지속적으로 업데이트 된다. 경매가 종료되면, 즉 낙찰 상태인 경우 bid_winner는 낙찰자를 의미하게 된다.

4. Class Diagram

4.1 Client Side



4.2 Server Side



5. Project design

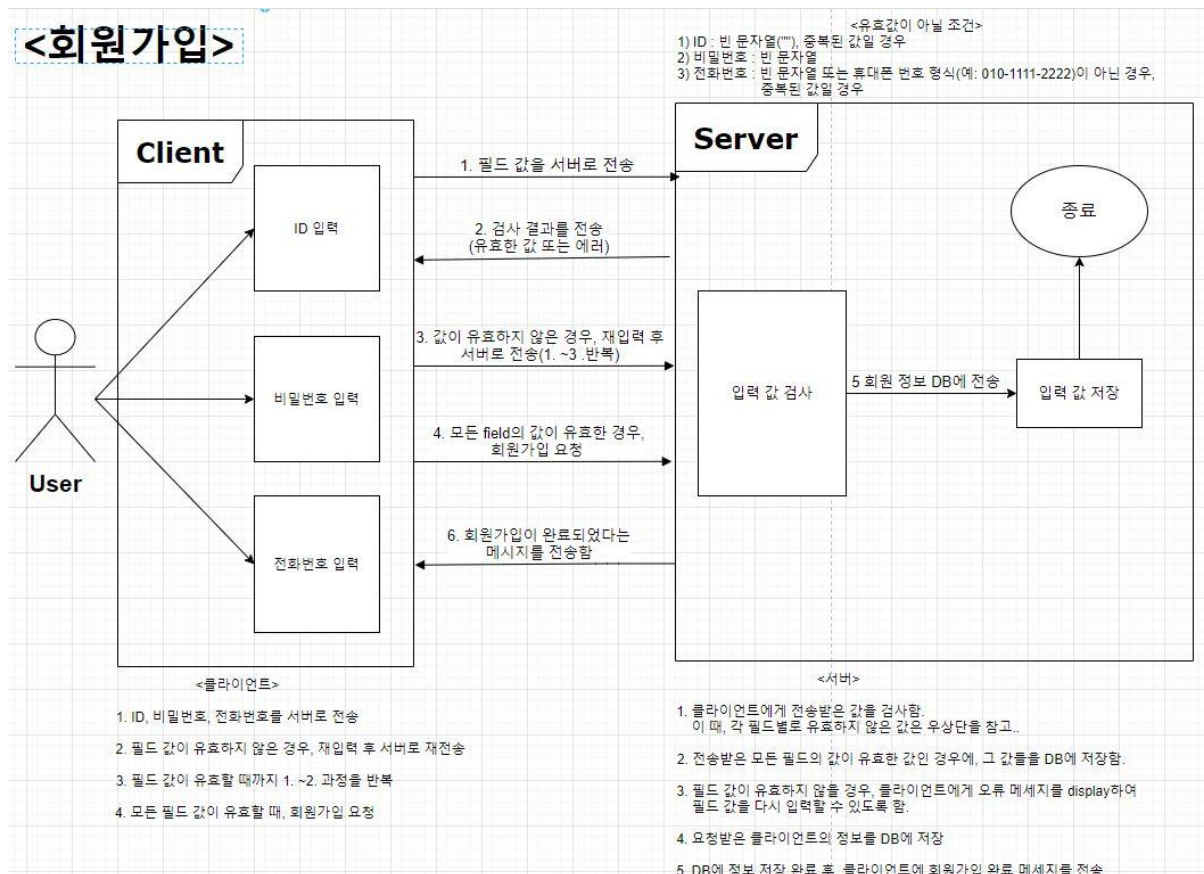
5.1 시나리오 및 CM API

크게 회원 가입, 경매 목록 출력, 입찰, 경매 물품 등록의 네 개의 파트로 나누었으며, 각각에 대한 시나리오 다이어그램 및 사용할 CM API의 활용 계획을 기재하였다.

5.1.1 회원 가입

중간 보고서 작성 시와 다르게 구현의 간편성 추구와, 서버의 걸리는 부하를 최소화하기 위하여, System design을 일정 부분 변경하였다. 자세한 내용은 아래부터 시작된다.

<기존 시나리오>

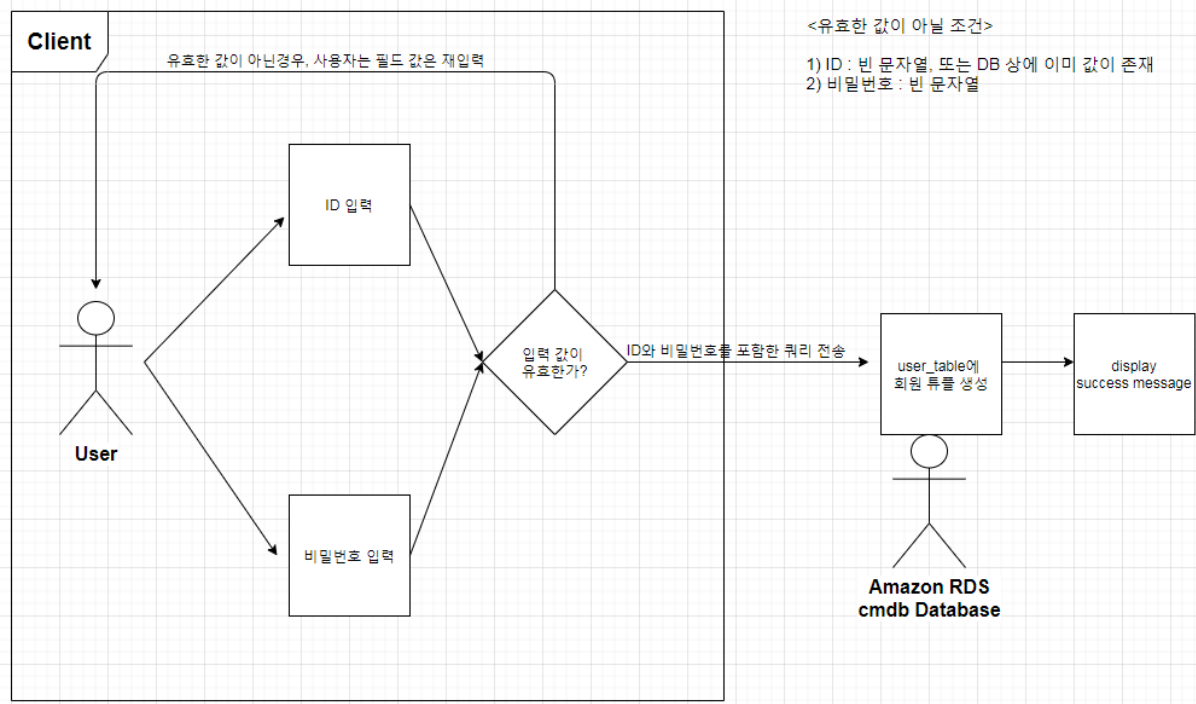


기존 시나리오에서는 사용자가 ID, 비밀번호, 전화번호 3 가지 필드 값을 GUI 상에 입력하고 서버에 전송하면, 서버에서 입력 값을 검사한 후, 입력 값이 유효하지 않을 경우, 서버에서 에러 메시지를 보내고, 다시 클라이언트에서 필드 값을 서버로 보내는 ping pong message 를

반복적으로 보내는 내용이 있었다. 이러한 경우, 클라이언트-서버 간 communication delay 가 지속적으로 발생할 수 있고, 서버에서 입력 값을 검사하는 역할을 수행하게 돼, 부하가 가중될 수 있는 문제점이 발생한다. 따라서 아래와 같이 수정된 시나리오를 구성 및 구현하였다.

<수정 시나리오>

<회원가입>



첫 번째, 필드 종류를 CM DB user_table 테이블의 속성 구성을 참고하여, ID와 비밀번호 두 개로 간소화하였다. 두 번째, 입력 값 검사하는 부분 중, 빈 문자열을 검사하는 역할은 Client에게 할당하였고, ID 중복 여부 검사 부분만 DB에 할당함으로써, Server의 부하를 최소화시켰다.

<사용한 CM API>

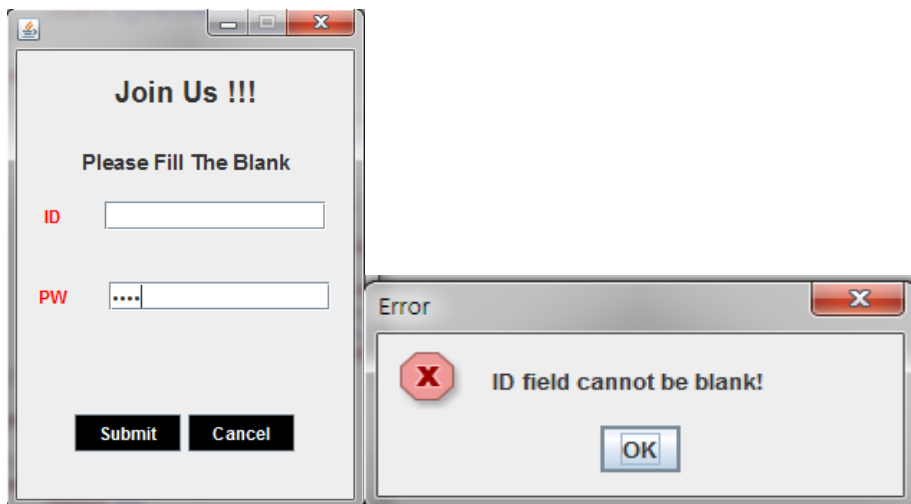
위 수정된 시나리오에서 Client에서 Amazon RDS Database로 회원가입 요청 쿼리를 전송하기 위해 아래의 CM API를 사용하였다.

- CMClientStub.registerUser(String id, String password)

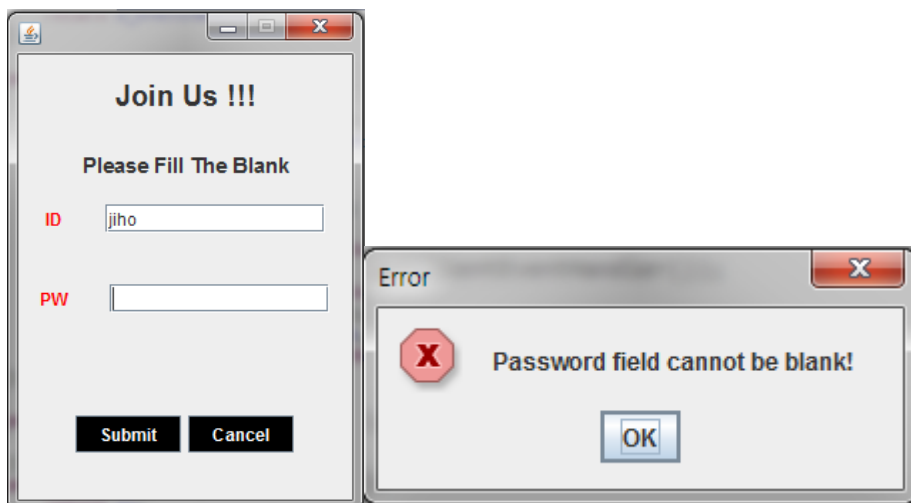
<실행 결과>

실행 결과는 입력 값이 유효하지 않은 경우와 회원 가입이 성공한 경우 크게 두 가지로 분류되며, 각 case 에 따른 결과 화면은 아래와 같다.

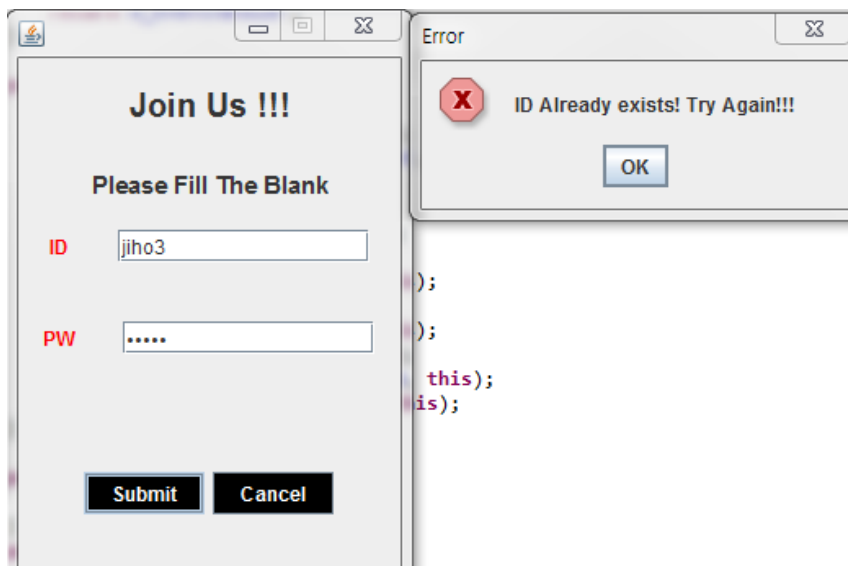
① case 1 : id 가 빈 값인 경우



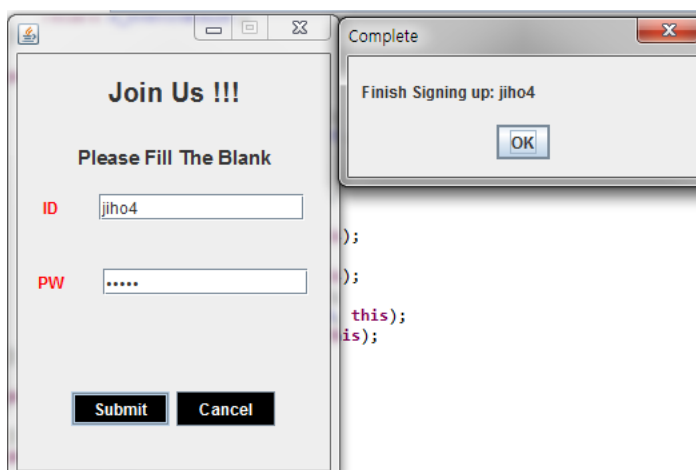
② case 2 : password 가 빈 값인 경우



③ case 3 : id 가 중복된 경우



④ case 4 : 성공적으로 회원가입을 완료한 경우



45	jiho4	*7958CC36194002FCA2D0BB76A4FAFB6A2FC...	2020-06-16 00:19:53.0
----	-------	---	-----------------------

<코드 소개>

회원 가입을 처리하는 파일은 SignUp.java 파일이며, 주요 기능에 대해 코드 설명을 하면 아래와 같다.

① JAVA 와 데이터베이스 간 연결

```
// MySQL driver
private final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
// Database address
private final String DB_URL = "jdbc:mysql://ahndb.caliu3u2esci.ap-northeast-2.rds.amazonaws.com:3306/cmdb?useSSL=false&autoReconnection=true"
// User name & password (We use database in Amazon RDS, so we fixed user name and password)
private final String USER_NAME = "ahn";
private final String PASSWORD = "guqqnstks123";

// objects for creating and executing some queries
private Connection conn = null;
private Statement state = null;

// code for connecting to MySQL Server to use select query
try {
    Class.forName(JDBC_DRIVER);
    conn = DriverManager.getConnection(DB_URL, USER_NAME, PASSWORD);
    System.out.println(conn);
    state = conn.createStatement();
} catch (Exception e) {
    System.out.println(e);
}
```

Java 와 데이터베이스 연동을 위해, JDBC Driver, DB URL, DB user name, DB password 정의하고, getConnection() 함수를 통해 연동하게 된다. 또한 쿼리 처리를 위해 필요한 Connection 및 Statement 객체를 정의하였다.

② 필드 값 검사(Error part)

```
String newUserID = id.getText();    // id
String newUserPassword = "";        // password

// code for reading input password
char[] secret_pw = pwd.getPassword();
String errorMessage = "";

for(char cha : secret_pw) {
    Character.toString(cha);
    newUserPassword += (newUserPassword.equals("")) ? cha + "" : "" + cha + "";
}
```

GUI 상에 입력된 id 및 password 필드 값을 읽어들이는 부분이다.

case 1) id 필드 값이 빈 값일 경우

```
// case that id field is blank
if(newUserID.length() == 0) {
    errorMessage = "ID field cannot be blank!";
    JOptionPane.showMessageDialog(null, errorMessage, "Error", JOptionPane.ERROR_MESSAGE);
}
```

Case 2) password 필드 값이 빈 값일 경우

```
// case that password field is blank
else if(newUserPassword.length() == 0) {
    errorMessage = "Password field cannot be blank!";
    JOptionPane.showMessageDialog(null, errorMessage, "Error", JOptionPane.ERROR_MESSAGE);
}
```

id 필드 값과, password 필드 값이 빈 값이 아닌 경우는 중복된 id 인지 체크하기 위하여 아래 코드를 수행한다.

```
// case that user fills both id and password field
else if(newUserID.length() != 0 && newUserPassword.length() != 0) {

    ResultSet rs;
    try {
        // set MySQL Query
        PreparedStatement pstmt = conn.prepareStatement("SELECT * FROM user_table WHERE userName=?");
        pstmt.setString(1, newUserID);

        // execute |
        rs = pstmt.executeQuery();

        // get result
        rs.last();
        int rowCount = rs.getRow();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Case 3) 중복 된 id 가 존재하는 경우

```
// If newUserId value is already exists (in cmdb), it's impossible to submit
if (rowCount != 0) {
    errorMessage = "ID Already exists! Try Again!!!";
    JOptionPane.showMessageDialog(null, errorMessage, "Error", JOptionPane.ERROR_MESSAGE);
}
```

Case 4) 중복 된 id 가 없는 경우, user_table 에 새로운 튜플을 추가 (지면 관계 상 다음 장에 표시)

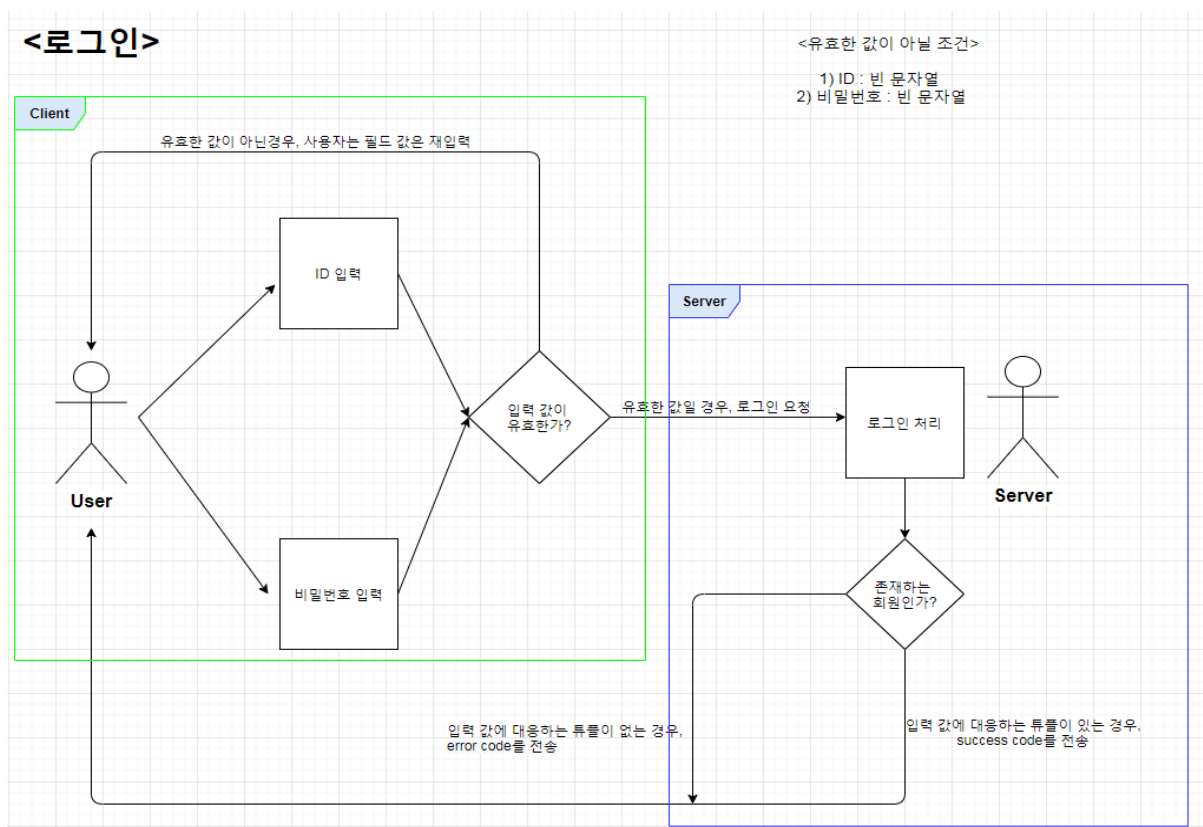
```
// If newUserId value doesn't exist, it's possible to submit new user information in cmdb
else {
    // register user
    m_clientStub.registerUser(newUserID, newUserPassword);
    try {
        Thread.sleep(3000);
    } catch (InterruptedException e1) {
        e1.printStackTrace();
    }
}

String successMessage = "Finish Signing up: " + newUserID;
JOptionPane.showMessageDialog(null, successMessage, "Complete", JOptionPane.PLAIN_MESSAGE);
}
```

5.1.2 로그인

중간 보고서에서는 로그인 부분이 빠져 있어서, 기말 보고서에 본 내용을 추가시켰다.

<시나리오>



Client 는 먼저 로그인 GUI 에서 ID 와 비밀번호를 입력한 후, 확인 버튼을 누른다. 그 다음 자체적으로 ID 와 비밀번호 중 비어있는 값이 있는지 검사한다. 만약 둘 중에 어느 하나라도 빈

값이 존재할 경우, 재입력을 받도록 한다. ID 필드와 비밀번호 필드 모두 비어있지 않은 경우, Client 는 Server 측으로 CM login 요청을 수행한다. Server에서는 Amazon RDS Database 에 있는 cmdb.user_table 테이블에 있는 tuple 을 검색하여, 입력 값에 대응하는 tuple 이 있는지 검사한다. 만일 입력 값에 대응되는 회원이 존재할 경우, client 측으로 성공 코드를 전송하고, 대응되는 회원이 존재하지 않는 경우, 마찬가지로 client 측으로 실패 코드를 전송하게 된다.

<사용한 CM API>

① Login.java (클라이언트 로그인 GUI 및 로그인 요청을 수행하는 코드)

- CMClientStub.loginCM()

② AuctionServerEventHandler.java (경매 CM 서버 이벤트 핸들러 코드)

- CMServerStub.getCMInfo()
- CMInfo.getConfigurationInfo()
- CMInfo.getInteractionInfo()
- CMInteractionInfo.getMyself()
- CMUser.getCurrentSession()
- CMUser.getCurrentGroup()
- CMInfo.isLoginScheme()
- CMSessionEvent.getUserName, getPassword()
- CMSessionEvent.getSender()
- CMDDBManager.authenticateUser()
- CMDummyEvent()
- CMDummyEvent.setHandlerSession()
- CMDummyEvent.setHandlerGroup()
- CMDummyEvent.setDummyInfo()

- CMStub.send()

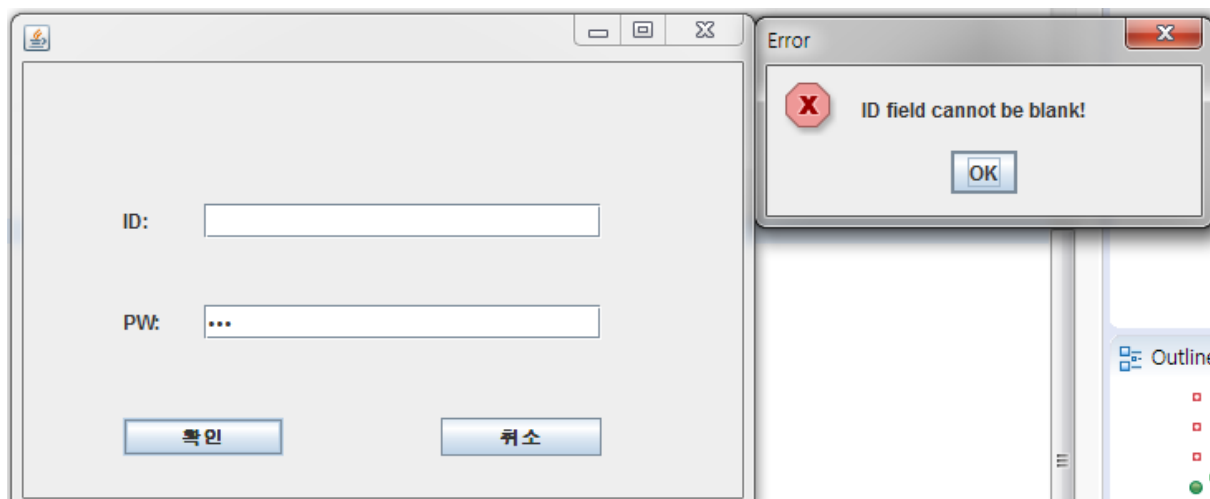
③ AuctionClientEventHandler.java (경매 CM 클라이언트 이벤트 핸들러 코드)

- CMDummyEvent.getDummyInfo()

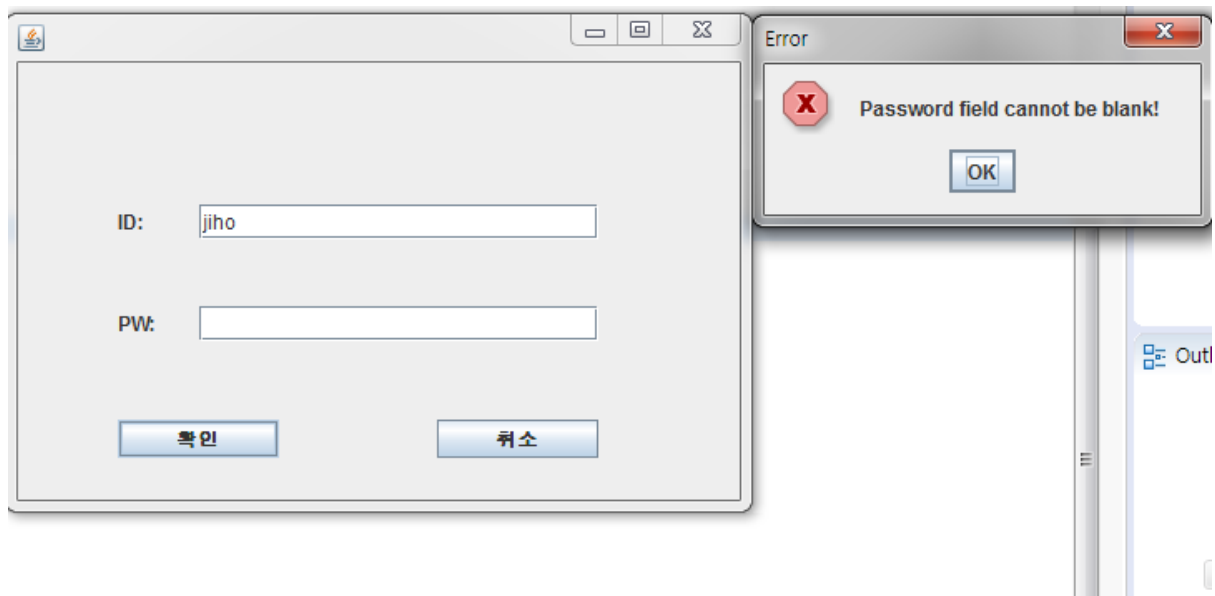
<실행 결과>

실행 결과는 입력 값이 유효하지 않은 경우, 필드 값에 대응되는 회원이 존재하지 않는 경우, 로그인에 성공한 경우 크게 세 가지로 분류되며, 각 case 에 따른 결과 화면은 아래와 같다.

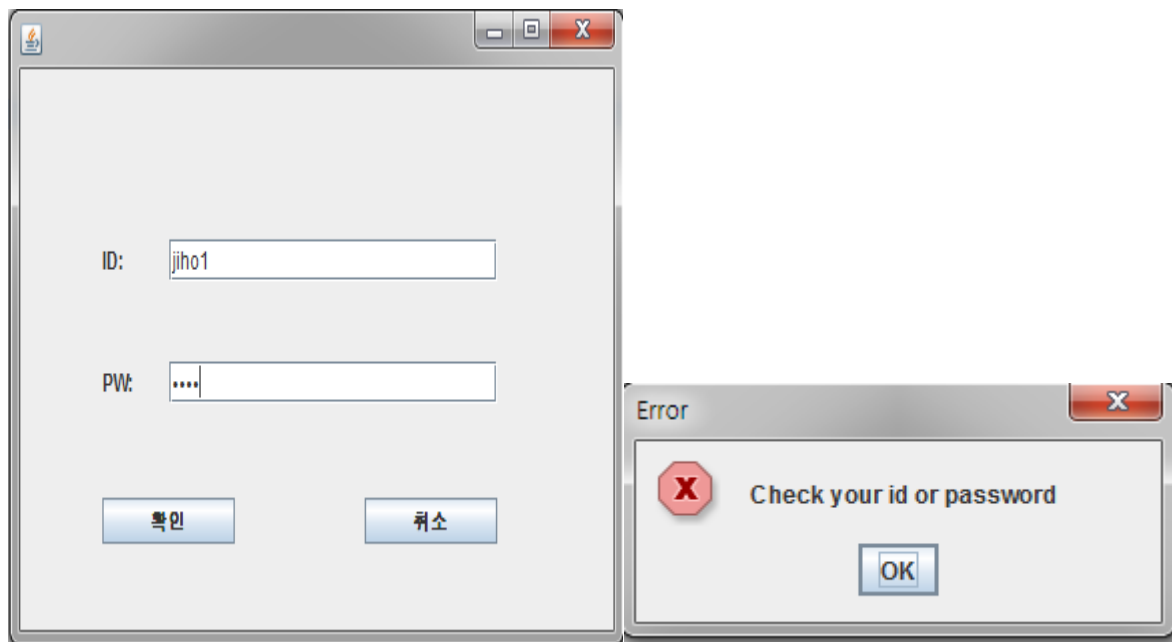
① case 1 : id 가 빈 값인 경우



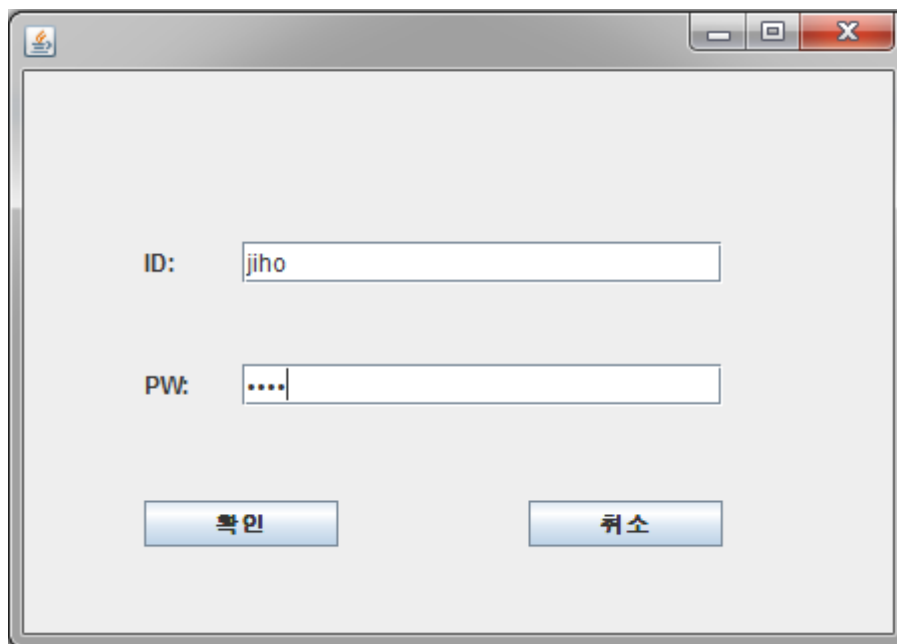
② case 2 : password 가 빈 값인 경우(지면 관계 상 다음 장에 기재함)



- ③ case 3 : id 또는 password 중 어느 하나 또는 모든 값이 DB 에 존재하지 않는 경우
(회원 정보가 존재하지 않는 경우)



- ④ case 4 : 입력 값에 대응되는 회원 정보가 DB 상에 존재하는 경우(성공, 다음 장에 기재)



ID:

PW:



Auction List

물품 명	최고 입찰가	마감날짜	입찰
iPhone10	9000	2020-06-15 00:5...	<input type="button" value="입찰"/>
iPad	5000	2020-06-16 22:0...	<input type="button" value="입찰"/>
iPad2	3000	2020-06-15 00:5...	<input type="button" value="입찰"/>
iPad3	45000	2020-06-15 12:0...	<input type="button" value="입찰"/>
iPhone7	3000	2020-06-16 23:0...	<input type="button" value="입찰"/>
iPhoneSE	23000	2020-06-16 13:0...	<input type="button" value="입찰"/>
iPad5	1000	2020-06-15 23:2...	<input type="button" value="입찰"/>
iPad6	1000	2020-06-15 23:2...	<input type="button" value="입찰"/>
iPad4	1000	2020-06-15 23:2...	<input type="button" value="입찰"/>
iMac	1000	2020-06-15 23:2...	<input type="button" value="입찰"/>
...

위 그림과 같이 물품 등록, 관심 물품 입찰 및 물품 설명서를 확인할 수 있는 창으로 넘어가게 된다.

~~** 오류 사항 : 예를 들어 123 이라는 id 로 회원가입을 해서, 123 id 와 password 를 입력한 경우, password 가 틀려서 팝업 된 예러 메시지를 보고 다시 로그인을 시도할 수 있다. 이 때, 다시 로그인 버튼을 클릭하여 123 이라는 id 로 회원가입을 시도할 시,~~

<코드 소개>

로그인을 처리하는 파일은,

Login.java, AuctionServerEventHandler.java, AuctionClientEventHandler.java 파일이며, 각 파일 별 사용하는 주요기능에 대해 코드 설명을 하면 아래와 같다. (지면 관계 상 다음 장에서 소개하겠다.)

① Login.java

```
loginFlag = false;
//CMSessionEvent loginAckEvent = null;
String userID = id.getText();
String userPassword = "";
char[] secret_pw = pwd.getPassword();
String errorMessage = "";

for(char cha : secret_pw) {
    Character.toString(cha);
    userPassword += (userPassword.equals("")) ? cha + "" : "" + cha + "";
}
```

Login GUI 에서 id 와 password 필드 값을 읽어 들이는 부분이다.

```
if(userID.length() == 0) {
    errorMessage = "ID field cannot be blank!";
    JOptionPane.showMessageDialog(null, errorMessage, "Error", JOptionPane.ERROR_MESSAGE);
}

else if(userPassword.length() == 0) {
    errorMessage = "Password field cannot be blank!";
    JOptionPane.showMessageDialog(null, errorMessage, "Error", JOptionPane.ERROR_MESSAGE);
}
```

id 와 password 필드 값의 유효성을 검사하는 부분이다. 둘 중에 어느 하나라도 빈 값인 경우, 에러 메시지 팝업 창을 띄어주게 한다.

```
else if(userID.length() != 0 && userPassword.length() != 0) {  
    //System.out.println(m_clientStub);  
  
    boolean bRequestResult = m_clientStub.loginCM(userID, userPassword);  
    if(bRequestResult)  
    {  
        System.out.println("successfully sent the login request.\n");  
    }  
    else  
    {  
        System.out.println("failed the login request!\n");  
    }  
}
```

Id 와 password 필드 값이 유효한 경우, 해당 값을 가지는 회원이 있을 경우 CM Server 로 로그인을 시켜주는 부분이다.

② AuctionServerEventHandler.java

```

private void processSessionEvent(CMEvent cme)
{
    CMConfigurationInfo confInfo = m_serverStub.getCMInfo().getConfigurationInfo();
    CMSessionEvent se = (CMSessionEvent) cme;
    switch(se.getID())
    {
        case CMSessionEvent.LOGIN:
            System.out.println("~~~~~CMSessionEvent.LOGIN");
            System.out.println("[ "+se.getUserName()+" ] requests login.");
            if(confInfo.isLoginScheme())
            {
                // user authentication...
                // CM DB must be used in the following authentication..
                boolean ret = CMDBManager.authenticateUser(se.getUserName(), se.getPassword(),
                    m_serverStub.getCMInfo());
                if(!ret)
                {
                    CMInteractionInfo interInfo = m_serverStub.getCMInfo().getInteractionInfo();
                    CMUser myself = interInfo.getMyself();
                    CMDummyEvent dummy = new CMDummyEvent();
                    dummy.setHandlerSession(myself.getCurrentSession());
                    dummy.setHandlerGroup(myself.getCurrentGroup());

                    dummy.setDummyInfo("LoginFailed#");
                    m_serverStub.send(dummy, se.getSender());
                }
            }
            else
            {
                CMInteractionInfo interInfo = m_serverStub.getCMInfo().getInteractionInfo();
                CMUser myself = interInfo.getMyself();
                System.out.println("[ "+se.getUserName()+" ] authentication succeeded.");
                sendItemList(se.getSender());
            }
    }
    break;
}

```

Login.java 파일에서 loginCM() 함수를 호출하여 로그인 요청을 하면, 로그인에 관련된 처리를 수행하는 부분이다. 먼저 현재 CM Default Server 에 대한 CM 및 설정 파일 정보를 추출하여, 이 정보와, CMSessionEvent.Login 이벤트로 넘어온 id 및 password 필드 값을 참고하여 CMDBManager 클래스의 authenticateUser() 함수를 호출한다. 이 결과 값에 의해, 만약 해당 id와 비밀번호 값을 만족하는 유저가 없는 경우, 더미 이벤트를 생성하여 "LoginFailed#" 이라는 메시지를 첨부하여, CMClientEventHandler.java(클라이언트 측) 파일로 전송한다. 만일 해당 값을 만족하는 유저가 실제로 존재하는 경우, 현재 서버 상에 올라와있는 경매 물품을 확인하고, 경매 물품 등록을 할 수 있는 AuctionGui.java 파일을 호출하기 위해, sendItemList() 함수를 호출한다.

③ AuctionClientEventHandler.java

```
private void processDummyEvent(CMEvent cme)
{
    CMDummyEvent due = (CMDummyEvent) cme;

    String due_tmp = due.getDummyInfo();
    System.out.println("receive dummy msg: "+due.getDummyInfo()+"\n");
    String[] tmp = due_tmp.split("#");

    if(tmp.length == 0 ) {
        return;
    }
    else if(tmp[0].equals("LoginFailed")) {
        String errorMessage = "Check your id or password";
        JOptionPane.showMessageDialog(null, errorMessage, "Error", JOptionPane.ERROR_MESSAGE);
        Login lg = m_client.getLogin();
        lg.setVisible(true);
    }
}
```

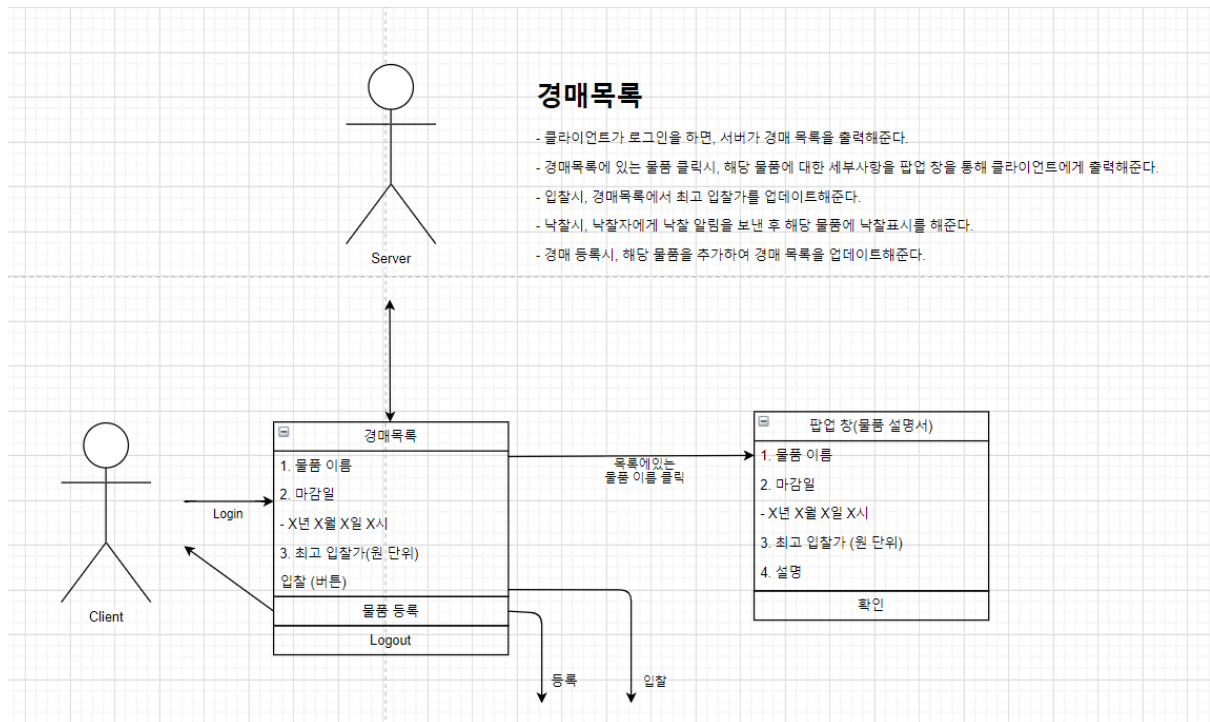
AuctionServerEventHandler.java 파일에서 전송된 더미 이벤트를 처리하는 코드이다. 우선 getDummyInfo() 함수를 통해 서버로부터 전송된 메시지의 내용을 확인한다. 메시지 형식은 "이벤트 이름#기타 1#기타 2#..."의 형식이다. 이 메시지 내용이 구분자 "#"를 기준으로 파싱되어 String 배열 tmp 에 저장된다. 이 때, tmp[0]의 내용이 이벤트 이름이 되는 것이다.

만일 'tmp[0] == LoginFailed' 조건을 만족하는 경우 login GUI 를 통해 입력한 id 또는 password 값을 확인하라는 에러 메시지 팝업 창을 띄어 준다. 그 다음 다시 로그인을 시도할 수 있도록 login GUI 화면을 띄어 준다.


```
else {  
    int rowLength = tmp.length / 4;  
    String[][] input = new String[rowLength][4];  
    for (int i = 0; i < rowLength; i++) {  
        for (int j = 0; j < 4; j++) {  
            input[i][j] = tmp[4 * i + j];  
        }  
    }  
    m_client.getLogin().dispose();  
    m_client.getAuctionGui().setContent(input, rowLength);  
    m_client.getAuctionGui().initialize();  
    m_client.getAuctionGui().frame.setVisible(true);  
}
```

Else 에 해당되는 이벤트는 로그인을 정상적으로 처리했다는 내용에 해당되는데, 이 경우 AuctionGui class 를 호출하는 동작을 수행한다.

5.1.3 경매 목록



(사진상에 있는 시나리오는 기존 시나리오 입니다.)

<시나리오 (최종)>

- (1) 클라이언트가 로그인을 하면, 서버가 경매 목록을 데이터베이스로부터 불러와서 전송해준다.
- (2) 클라이언트가 전송 받은 경매 목록을 GUI 상으로 띄워준다.
- (3) 경매 목록에 있는 물품 명 클릭시, 마찬가지로 해당 물품에 대한 세부사항을 서버로부터 받아와 팝업 창을 통해 클라이언트에게 보여준다.
- (4) 입찰, 낙찰, 등록 후 업데이트된 경매 목록 상황을 반영해준다.

<CM API (기준)>

- (1) CMapEventHandler 를 이용
- (2) queryGetUsers() 사용하여 경매를 등록한 사람의 물품 정보를 가져온다.
- (3) getName()과 getDate()를 통해 물품의 이름과 마감일을 가져온다.

(4) CMDummyEvent 클래스와 send 함수를 이용하여 경매목록에 대한 정보를 서버에서 클라이언트로 전송한다.

(5) 경매정보를 나타내는 CMUserEvent 인스턴스에 있는 value 들을 이용하기 위해 getEventField()를 사용한다.

<CM API (최종)>

(1) CMClientEventHandler 와 CMServerEventHandler 를 이용하여 클라이언트와 서버간에 경매목록 관련 이벤트를 수신한다.

(2) CMDBManager 의 sendSelectQuery() 를 사용하여 경매목록이 저장된 데이터베이스로부터 경매물품 정보들을 불러온다.

(3) CMDummyEvent 클래스와 send 함수를 이용하여 경매목록에 대한 정보를 서버에서 클라이언트로 전송한다.

<동작과정 및 소스코드>

1. 사용자 로그인 후



물품명	최고 입찰가	마감날짜	입찰
iPhone10	10000	2020-06-15 00:5...	낙찰
iPad	5000	2020-06-16 22:0...	낙찰
iPad2	3000	2020-06-15 00:5...	낙찰
iPad3	45000	2020-06-15 12:0...	낙찰
iPhone7	3000	2020-06-16 23:0...	낙찰
iPhoneSE	23000	2020-06-16 13:0...	낙찰
iPad5	1000	2020-06-15 23:2...	낙찰
iPad6	1000	2020-06-15 23:2...	낙찰
iPad4	1000	2020-06-15 23:2...	낙찰
iMac	1000	2020-06-15 23:2...	낙찰

로그아웃 목록 갱신 물품 등록

소스 코드 (CMServerEventHandler.java)

```
private void processSessionEvent(CMEvent cme)
{
    CMConfigurationInfo confInfo = m_serverStub.getCMInfo().getConfigurationInfo();
    CMSessionEvent se = (CMSessionEvent) cme;
    switch(se.getID())
    {
        case CMSessionEvent.LOGIN:

    else
    {
        CMInteractionInfo interInfo = m_serverStub.getCMInfo().getInteractionInfo();
        CMUser myself = interInfo.getMyself();
        System.out.println "["+se.getUserName()+"] authentication succeeded.");
        m_serverStub.replyEvent(se, 1);

        sendItemList(se.getSender());
    }
}
```

사용자가 로그인을 하면, 서버가 인증절차를 거친다. 만약 인증에 성공하면, 서버는 해당 클라이언트로 응답이벤트를 보내고, sendItemList() 함수를 호출하여 경매목록 또한 보내준다. sendItemList() 의 구현은 아래와 같다.

```
// 클라이언트에게 경매 목록 리스트 보내주기
private void sendItemList(String clientName){
    CMDummyEvent due = new CMDummyEvent();
    due.setHandlerSession(m_serverStub.getMyself().getCurrentSession());
    due.setHandlerGroup(m_serverStub.getMyself().getCurrentGroup());
    String tmp="";
    try{
        ResultSet rs = CMDBManager.sendSelectQuery("SELECT name,now_price,due_date,status FROM item", m_serverStub.getCMInfo());
        while(rs.next()) {
            tmp += rs.getString(1)+"#";
            tmp += rs.getString(2)+"#";
            tmp += rs.getString(3)+"#";
            if(rs.getString(4).equals("t"))
                tmp += "낙찰"+"#";
            else tmp += "입찰"+"#";
        }
    }catch(SQLException e) {
    }
    tmp=tmp.substring(0, tmp.length()-1);
    System.out.println(tmp);
    due.setDummyInfo(tmp);
    m_serverStub.send(due, clientName);
}
```

먼저 CMDBManager 의 sendSelectedQuery() 함수에 Query 문을 인자로 주어서 원하는 ResultSet 을 받는다. 실제 GUI 에서 목록으로 보여주는 값은 물품 명, 현재가, 마감기한, 낙찰여부 이기 때문에 그에 해당하는 모든 목록을 불러온다. 그리고 불러온 ResultSet 으로부터 DummyEvent 를 만든다. 이 때 DummyEvent 의 형식은 "<물품 명>#<현재가>#<마감기한>#<낙찰 여부>" 이며, 물품 목록이 하나 이상인 경우에도 마찬가지로

"<물품명>#<현재가>#<마감기한>#<낙찰여부>#<물품명>#<현재가>#<마감기한>#<낙찰여부>#...#<물품명>#<현재가>#<마감기한>#<낙찰여부>" 와 같이 연결시켜서 DummyEvent 를 만든다. 마지막으로 해당 DummyEvent 를 클라이언트에게 send 를 이용하여 전송한다.

소스 코드 (CMClientEventHandler.java)

```
private void processDummyEvent(CMEvent cme)
{
    CMDummyEvent due = (CMDummyEvent) cme;

    String due_tmp = due.getDummyInfo();
    System.out.println("receive dummy msg: "+due.getDummyInfo()+"\n");
    String[] tmp = due_tmp.split("#");
```

서버가 DummyEvent 로 제작하여 전송한 경매목록을 클라이언트가 받게 되면, 먼저 #을 기준으로 Parsing 과정을 거친다.

```
else {

    int rowLength = tmp.length / 4;

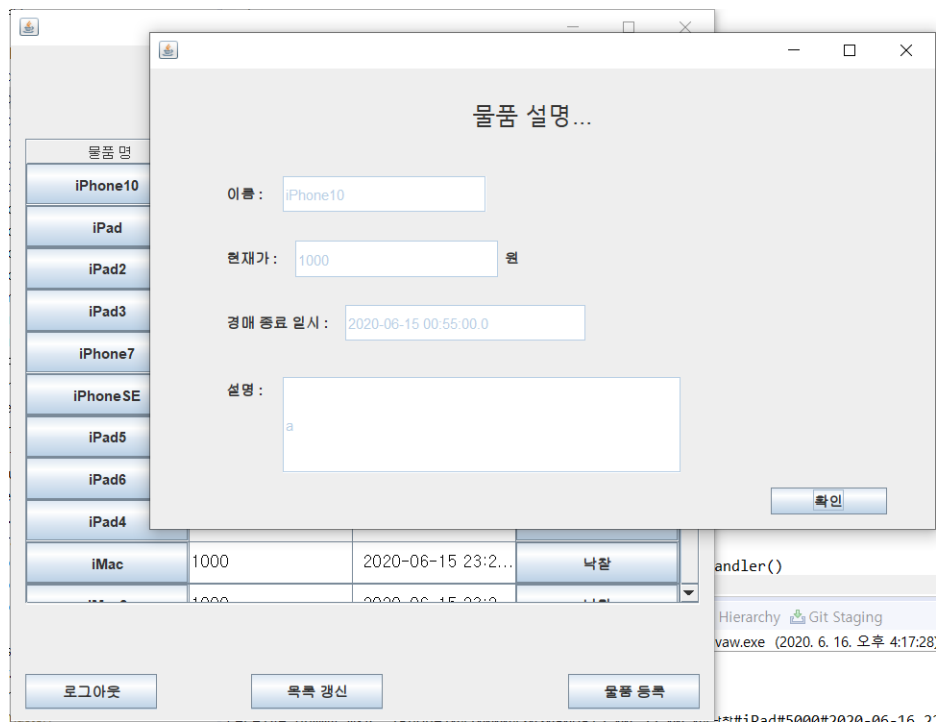
    String[][] input = new String[rowLength][4];

    for (int i = 0; i < rowLength; i++) {
        for (int j = 0; j < 4; j++) {
            input[i][j] = tmp[4 * i + j];
        }
    }

    m_client.getAuctionGui().setContent(input, rowLength);
    m_client.getAuctionGui().initialize();
    m_client.getAuctionGui().frame.setVisible(true);
}
```

Parsing 후 경매목록을 GUI 상에 보여주기 위해서는 Parsing 한 결과를 2 차원 배열로 만들어 주어야 한다. 이때 컬럼은 물품명, 현재가, 마감기한, 낙찰여부순이다. input 이라는 2 차원 배열을 형성하게 되면 그 배열을 토대로 AuctionGUI.java 에서 테이블을 set 한 후 시각화해준다.

2. 물품명 클릭시



소스 코드(AuctionGui.java)

```

} else if(clicked && fieldcol == 0){

    int selected_row = m_client.getAuctionGui().table.getSelectedRow();

    CMInteractionInfo interInfo = m_clientStub.getCMInfo().getInteractionInfo();
    CMUser myself = interInfo.getMyself();

    CMDummyEvent due = new CMDummyEvent();

    String tmp = "ItemDescription#" + selected_row;

    due.setDummyInfo(tmp);

    due.setHandlerSession(myself.getCurrentSession());
    due.setHandlerGroup(myself.getCurrentGroup());

    m_clientStub.send(due, "SERVER");

```

물품 명 버튼을 클릭할 경우, 클릭된 행을 토대로 물품 정보를 요청하는 DummyEvent 를 만들어서 서버에게 send 하여 준다. 이때 DummyEvent 의 형식은 "ItemDescription#<행>" 이다.

소스 코드(CMClientEventHandler.java)

```

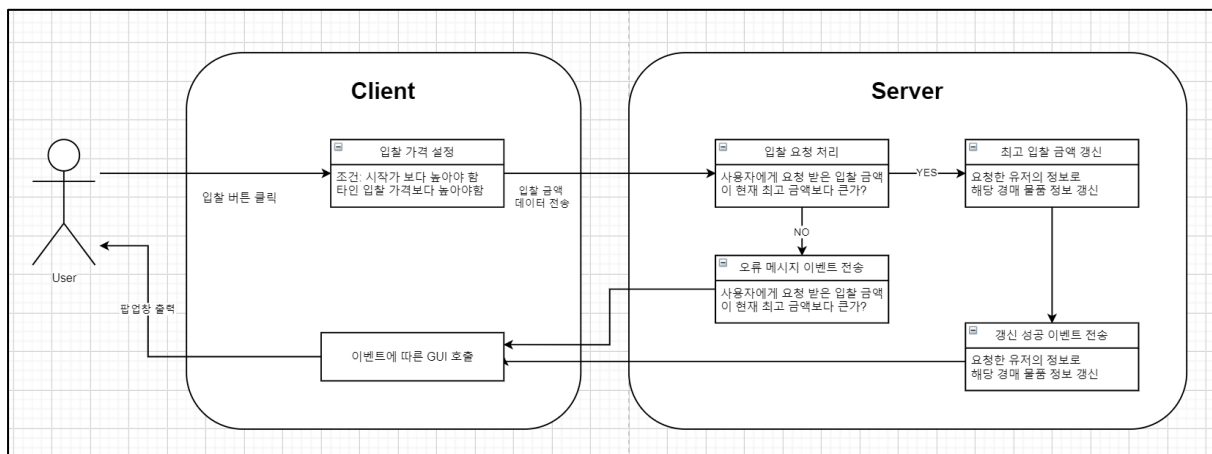
else if(data[0].equals("ItemDescription")) {
    CMDummyEvent ID = new CMDummyEvent();
    ID.setHandlerSession(myself.getCurrentSession());
    ID.setHandlerGroup(myself.getCurrentGroup());
    String tmp="ItemDescription#";
    try{
        ResultSet rs = CMDBManager.sendSelectQuery("SELECT name,start_price,due_date,description FROM item WHERE no="
                                                    + (Integer.parseInt(data[1])+1), m_serverStub.getCMInfo());
        while(rs.next()) {
            tmp += (rs.getString(1)+"#");
            System.out.println(rs.getString(1));
            tmp += (rs.getString(2)+"#");
            tmp += (rs.getString(3)+"#");
            tmp += rs.getString(4);
        }
        System.out.println(tmp);
    }catch(SQLException e) {
    }
    ID.setDummyInfo(tmp);
    m_serverStub.send(ID, due.getSender());
}

```

클라이언트의 processDummyEvent()에서 도착한 Dummy 를 공통의 방법으로 Parsing 을 했을 때, 첫번째 요소가 "ItemDescription" 인 경우 즉, 물품 정보 요청인 경우 실행되는 코드이다. 서버는 경매목록 출력과 동일한 방식으로 데이터베이스로부터 물품정보를 받아서 Dummy 를 제작 후 요청한 클라이언트에게 send 한다. 클라이언트가 Dummy 를 받으면, Gui 의 TextField 에 Dummy 의 내용을 입력하여 사용자에게 띄워준다.

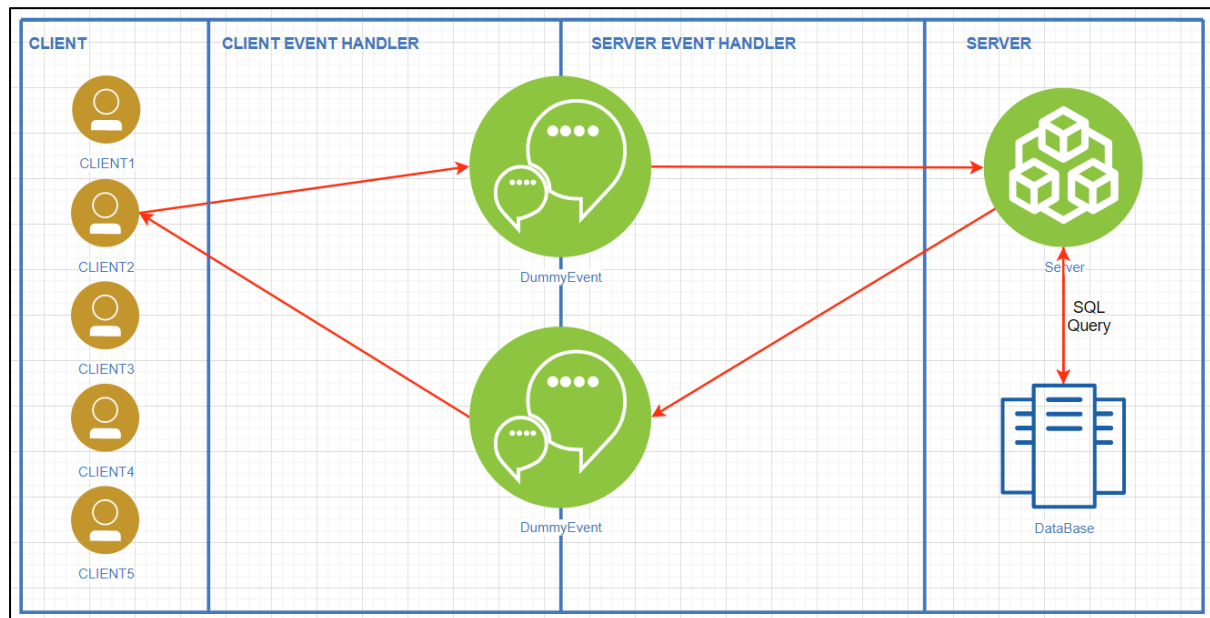
5.1.4 입찰

<기존 입찰 시나리오>



시나리오 : client 가 입찰을 원할경우,
 유저는 경매 리스트 창에서 입찰 버튼을 누른다.
 입찰 금액은 서버로 전송이 되며,
 현재 물품 최고가 보다 크면, 갱신을 한뒤 성공 이벤트를 client 에게 전송한다.
 만약, 물품 최고가 보다 작으면, 실패 이벤트를 client 에게 전송하다.
 메시지를 받은 client 은 팝업창을 출력한다.

<실제 구현 시나리오> - 변동사항 : DB 와 Server 와의 연결



<기존 CM API>

- (1) CMClientStub()을 이용하여 Client 측에서 Server 와 연결한다.
- (2) CMDDBInfo()을 이용하여 물품 등록 DB 와 연결하여 입찰 물품을 등록한다.
- (3) CMEventSynchronizer()을 이용하여 물품 목록 및 최고 입찰가를 최신화한다.
- (4) CMDDBManager()을 이용하여 물품 목록을 관리한다.
- (5) CMConfigurationInfo()을 이용하여 물품의 입찰 날짜를 지정한다.

(6) CMEventInfo()을 이용하여 최고 금액 입찰자에게 낙찰 알림 서비스를 제공한다.

<실제 구현 CM API>

(1) 입찰 버튼 클릭 시 해당 물품의 정보를 가져오는 경우

- CMInteractionInfo()
- CMdummyEvent()
- CMUser
- CMDBManager()

(2) 입찰가 설정 후 입찰을 하는 경우

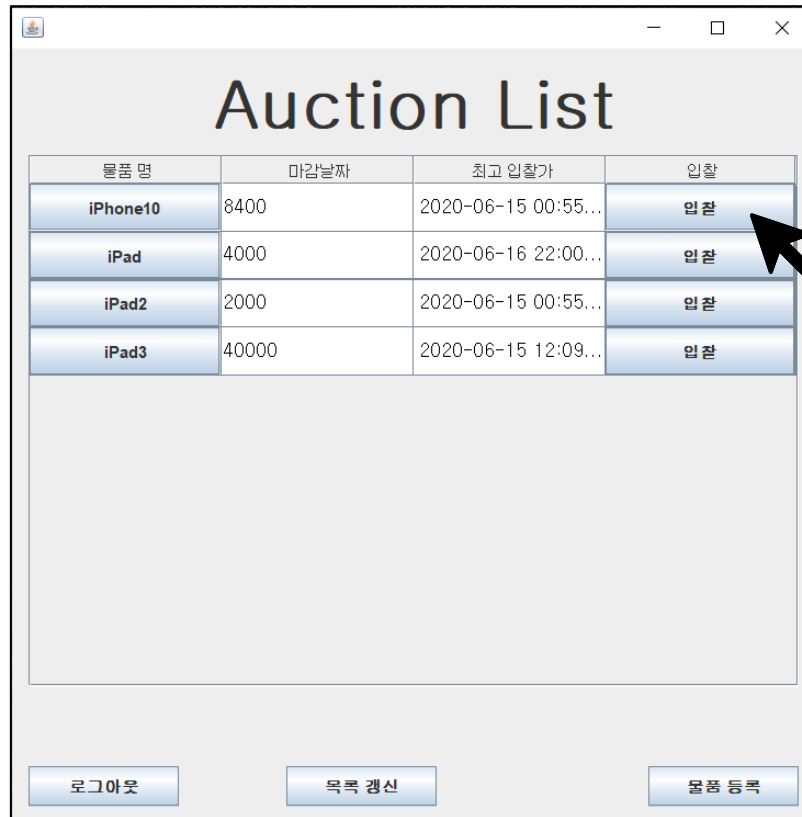
- CMinteractionInfo()
- CMdummyEvent()
- CMUser
- CMDBManager()

<실행 코드 및 프로세스>

(1) 입찰 버튼 클릭 시 해당 물품의 정보를 가져오는 경우

1). 입찰을 원하는 경매 물품에 대하여 입찰 버튼을 클릭한다.

- AuctionGUI.java -> getCellEditorValue() object



2). client 측에서 server에게 dummyEvent를 요청한다.

- 해당 dummyEvent는 "itemInfo" 라는 topic을 가지고, 해당 물품의 행 정보를 함께 전달한다.
- dummyEvent 내에서 여러 정보들은 '#'을 구분자로 하여 구분한다.

```
CMInteractionInfo interInfo = m_clientStub.getCMInfo().getInteractionInfo();
CMUser myself = interInfo.getMyself();
CMDummyEvent due2 = new CMDummyEvent();

String tmp2 = "itemInfo#" + selected_row;
due2.setDummyInfo(tmp2);

due2.setHandlerSession(myself.getCurrentSession());
due2.setHandlerGroup(myself.getCurrentGroup());
m_clientStub.send(due2, "SERVER");
```

3). sever측에서 topic이 "itemInfo"인 dummyEvent를 수신한다.

- serverEventHandler

```

private void processDummyEvent(CMEvent cme){
    CMInteractionInfo interInfo = m_serverStub.getCMInfo().getInteractionInfo();
    CMUser myself = interInfo.getMyself();

    CMDummyEvent due = (CMDummyEvent) cme;
    String[] data = due.getDummyInfo().split("#");

    ...

else if(data[0].equals("itemInfo")){
    CMDummyEvent ID = new CMDummyEvent();
    ID.setHandlerSession(myself.getCurrentSession());
    ID.setHandlerGroup(myself.getCurrentGroup());
    String tmp2="itemInfo#";

    try{
        ResultSet rs = CMDBManager.sendSelectQuery("SELECT name, now_price FROM item WHERE
no="+Integer.parseInt(data[1])+1, m_serverStub.getCMInfo());
        while(rs.next()) {
            tmp2 += (rs.getString(1)+"#");
            tmp2 += (rs.getString(2));
        }
    }catch(SQLException e) {}
    ID.setDummyInfo(tmp2);
    m_serverStub.send(ID, due.getSender());
}

...

```

4. server에서 dummyEvent로 받은 해당 열 정보를 통해 해당 열의 물품의 이름, 현재 입찰가 정보를 DB에 요청한다. (SQL Query문 사용)

```

ResultSet rs = CMDBManager.sendSelectQuery("SELECT name, now_price FROM item WHERE
no="+Integer.parseInt(data[1])+1, m_serverStub.getCMInfo());

```

5. DB로부터 받은 정보를 client에게 다시 dummyEvent를 통하여 전달한다.

6. client는 해당 정보를 다시 받아 GUI 상에 나타낸다.

```
else if(tmp[0].equals("itemInfo")){  
    m_client.getSetPrice().NameField.setText(tmp[1]);  
    m_client.getSetPrice().NowPriceField.setText(tmp[2]);  
    m_client.getSetPrice().frame.setVisible(true);  
}
```

(2) 입찰가 설정 후 입찰을 하는 경우

- 1). client가 입찰가를 설정한 이후 확인 버튼을 클릭한다.
- 2). 클릭 시 Client는 Server에게 dummyEvent를 발생시킨다.
 - 해당 dummyEvent는 "setPriceInfo" 라는 topic을 갖는다.
 - 해당 물품의 행 정보와 Client의 ID를 함께 전달한다.

- dummyEvent 내에서 여러 정보들은 '#'을 구분자로 하여 구분한다.

```
public void actionPerformed(ActionEvent e) {
    int selected_row = m_client.getAuctionGui().table.getSelectedRow();

    CMInteractionInfo interInfo = m_clientStub.getCMInfo().getInteractionInfo();
    CMUser myself = interInfo.getMyself();
    CMDummyEvent due3 = new CMDummyEvent();

    String tmp3 = "setPriceInfo#" + selected_row
+"#" + Integer.parseInt(SetPriceField.getText()) + "#" +
m_client.m_clientStub.getMyself().getName();
    due3.setDummyInfo(tmp3);

    due3.setHandlerSession(myself.getCurrentSession());
    due3.setHandlerGroup(myself.getCurrentGroup());
    m_clientStub.send(due3, "SERVER");
    frame.dispose();
}
```

3). Sever에서 "setPriceInfo"라는 topic을 가진 dummyEvent를 수신한다.

```
else if(data[0].equals("setPriceInfo")){
    CMDummyEvent ID = new CMDummyEvent();
    ID.setHandlerSession(myself.getCurrentSession());
    ID.setHandlerGroup(myself.getCurrentGroup());

    CMDummyEvent setResult = new CMDummyEvent();
    setResult.setHandlerSession(myself.getCurrentSession());
    setResult.setHandlerGroup(myself.getCurrentGroup());

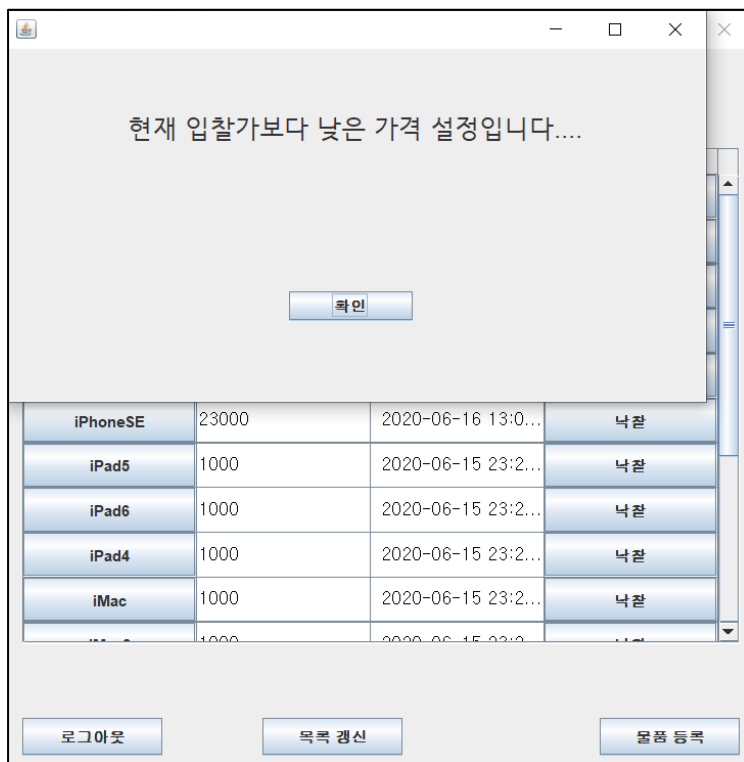
    int nowPrice = -1;
    System.out.println(data[3]);
    String query = String.format("UPDATE item SET now_price = %d, bid_winner = %s WHERE
no= %d", Integer.parseInt(data[2]), data[3], (Integer.parseInt(data[1])+1));

    try{
        ResultSet rs = CMDBManager.sendSelectQuery("SELECT now_price FROM item WHERE
no="+(Integer.parseInt(data[1])+1), m_serverStub.getCMInfo());
        while(rs.next()) {
```

```
        nowPrice = Integer.parseInt(rs.getString(1));
    }
} catch (SQLException e) {}

if (nowPrice == -1) {
    System.out.println("DB 접속 오류 예외처리");
    setResult.setDummyInfo("setPriceResult#1");
    sendItemList(due.getSender());
}
else if (Integer.parseInt(data[2]) > nowPrice) {
    System.out.println("success!!!");
    setResult.setDummyInfo("setPriceResult#2");
    CMDBManager.sendUpdateQuery(query, m_serverStub.getCMInfo());
    sendItemList(due.getSender());
}
else {
    System.out.println("현재 가격보다 작습니다.");
    setResult.setDummyInfo("setPriceResult#3");
    sendItemList(due.getSender());
}
m_serverStub.send(setResult, due.getSender());
}
```

4). 가격이 현재 입찰가보다 낮거나 같은 경우 예외 처리한다.



setPriceResult.java

```
import kr.ac.konkuk.ccslab.cm.stub.CMClientStub;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Font;

import javax.swing.JButton;

public class setPriceResult {
    private AuctionClient m_client;
    private CMClientStub m_clientStub;

    private JLabel setPriceResultLabel;
    JFrame frame;
```

```
public setPriceResult(CMClientStub clientStub, AuctionClient client) {
    m_client = client;
    m_clientStub = clientStub;
    initialize();
}

public void setMsg(String s) {
    setPriceResultLabel.setText(s);
}

public JFrame getJFrame() {
    return frame;
}

private void initialize() {
    frame = new JFrame();
    frame.setBounds(100, 100, 562, 314);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    setPriceResultLabel = new JLabel("입찰을 완료했습니다!");
    setPriceResultLabel.setFont(new Font("함초롬돋움", Font.PLAIN, 20));
    setPriceResultLabel.setBounds(95, 20, 354, 80);
    frame.getContentPane().add(setPriceResultLabel);

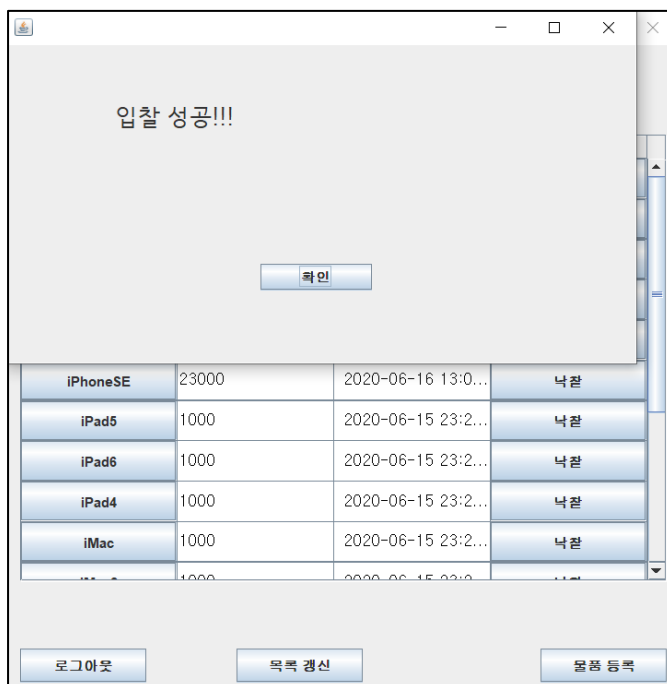
    JButton ConfirmButton = new JButton("확인");
    ConfirmButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            frame.dispose();
        }
    });
    ConfirmButton.setBounds(221, 190, 97, 23);
    frame.getContentPane().add(ConfirmButton);
}
}
```


5). 현재 입찰가보다 높은 금액을 입력하는 경우 DB에 update를 요청한다.

- bid_winner(낙찰자명)을 Client의 ID로 갱신한다.
- now_price(현재입찰가)를 Client가 입력한 값으로 갱신한다.

```
String query = String.format("UPDATE item SET now_price = %d, bid_winner = %s WHERE  
no= %d", Integer.parseInt(data[2]), data[3], (Integer.parseInt(data[1])+1));
```

6). Client는 갱신된 물품 목록을 GUI로 나타낸다.





물품명	마감날짜	최고 입찰가	입찰
iPhone10	9000	2020-06-15 00:5...	입찰
iPad	5000	2020-06-16 22:0...	입찰
iPad2	3000	2020-06-15 00:5...	입찰
iPad3	45000	2020-06-15 12:0...	입찰
iPhone7	2000	2020-06-16 23:0...	입찰
iPhoneSE	23000	2020-06-16 13:0...	입찰
iPad5	1000	2020-06-15 23:2...	입찰
iPad6	1000	2020-06-15 23:2...	입찰
iPad4	1000	2020-06-15 23:2...	입찰
iMac	1000	2020-06-15 23:2...	입찰

로그아웃 목록 갱신 물품 등록

client에서의 입찰 성공 여부 dummyEvent 수신

```
if(tmp.length == 0 ) {
    return;
}else if(tmp[0].equals("setPriceResult")){
    setPriceResult sr = m_client.getSetPriceResult();
    switch (tmp[1]){
        case "1":
            sr.setMsg("DB 등록 오류");
            break;
        case "2":
            sr.setMsg("입찰 성공!!!");
            break;
        case "3":
            sr.setMsg("현재 입찰가보다 낮은 가격 설정입니다. 입찰 실패했습니다.");
            break;
    }
    sr.frame.setVisible(true);
}
```

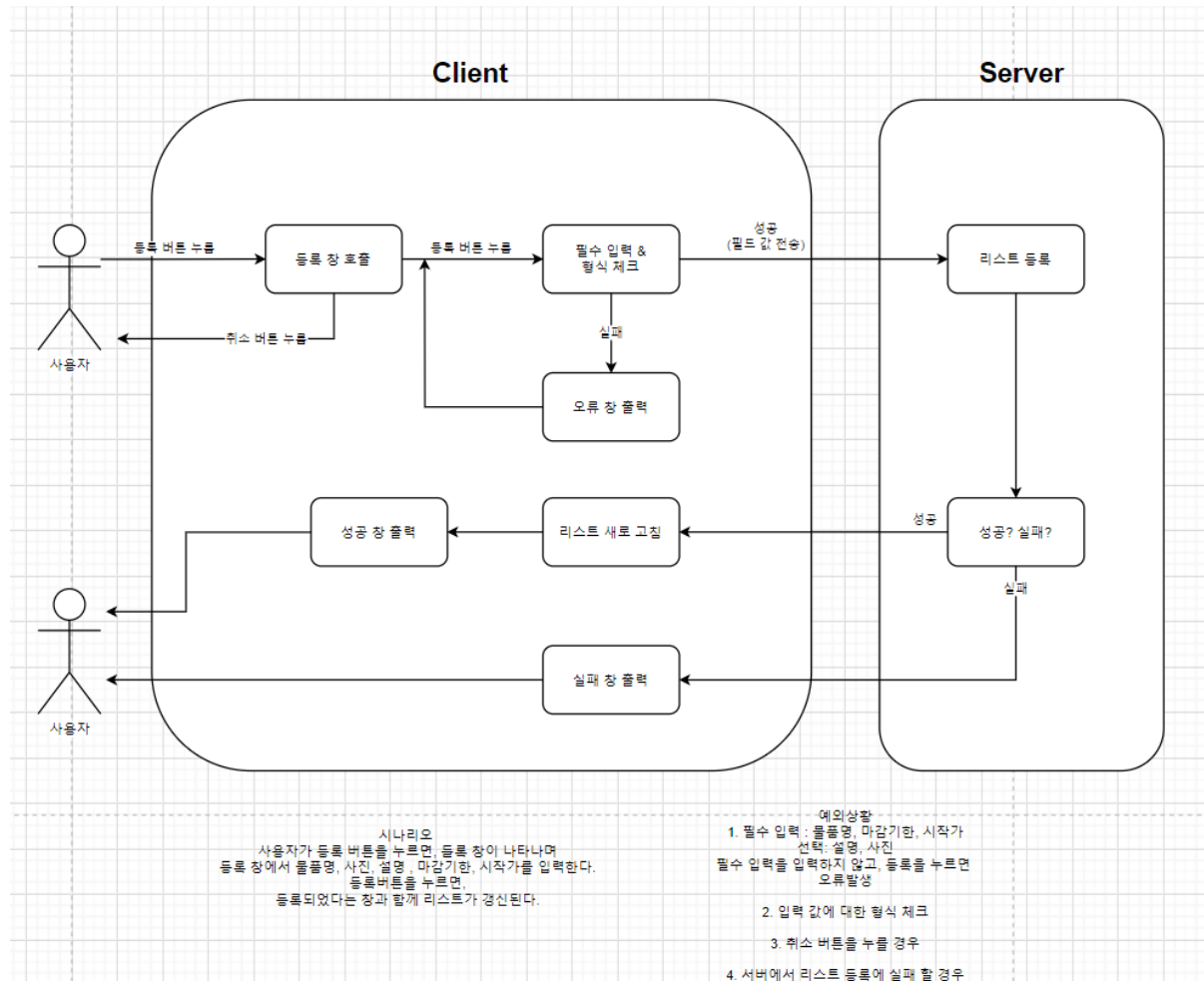
```

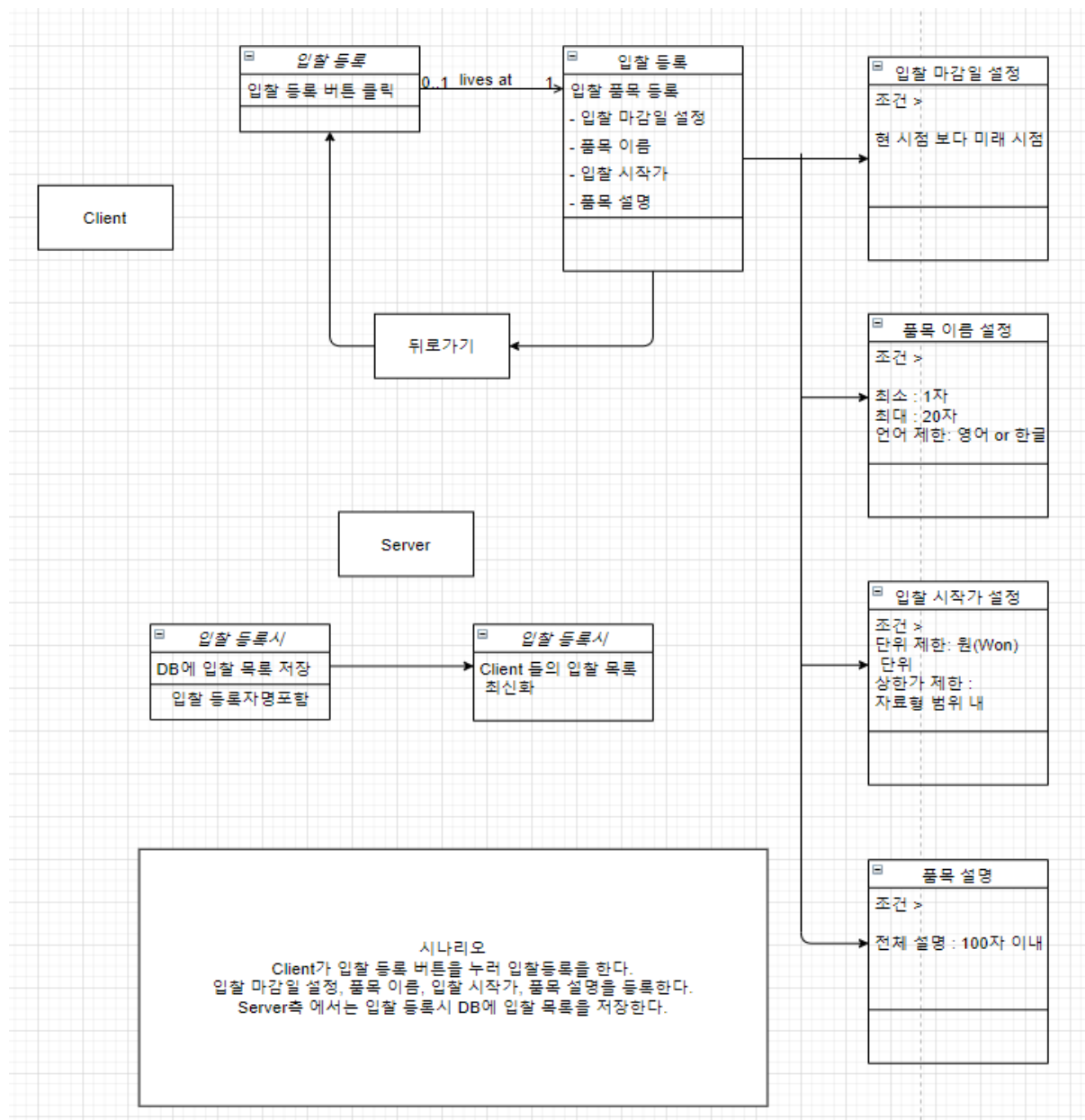
m_client.getSetPrice().getJFrame().dispose();
}

```

5.1.5 등록

<설계>






CM API

- (1) 서버와 클라이언트가 메시지를 전송할 때, CM user event 을 사용한다. 이때, CM 이벤트 핸들러에 이벤트를 등록시켜 비동기식으로 통신을 한다.
- (2) 서버가 경매 리스트를 갱신, 등록할 때, MySQL 서버와 관련된 CM DB API(CMDBManager) 을 이용하여 DB 를 갱신한다. 예로 들어, queryInsertUser(), queryGetUsers(), queryDeleteUser(), queryUpdateUser() 등이 있다.

<실행 결과>

1. 물품 등록 메인 화면



물품명	최고 입찰가	마감날짜	입찰
iPhone10	9000	2020-06-15 00:55:0...	입찰
iPad	5000	2020-06-16 22:00:0...	입찰
iPad2	3000	2020-06-15 00:55:0...	입찰
iPad3	45000	2020-06-15 12:09:0...	입찰
iPhone7	3000	2020-06-16 23:00:0...	입찰
iPhoneSE	23000	2020-06-16 13:00:0...	입찰
iPad5	1000	2020-06-15 23:20:0...	입찰
iPad6	1000	2020-06-15 23:21:0...	입찰
iPad4	1000	2020-06-15 23:22:0...	입찰
iMac	1000	2020-06-15 23:26:0...	입찰

로그아웃 목록 갱신 **물품 등록**

물품 등록을 하려면, 경매 리스트 메인 화면에서 우측 하단에 '물품 등록'을 선택한다.

2. 물품 등록을 눌렀을 때 화면

물품 등록

이름 :

입찰 시작가 : 원

마감일 설정 : 년 월 일 시

설명 :

물품 등록을 하려면, 사용자는 물품의 이름, 입찰 시작 가, 마감일 설정, 설명을 모두 입력하여 야 한다. 각 입력 필드의 제약사항은 위의 설계 그림에 명시되어 있다.

3. 물품 등록에서 물품 정보를 정상적으로 입력한 예시

물품 등록

이름 :

입찰 시작가 : 원

마감일 설정 : 년 월 일 시

설명 :

정상적으로 입력을 마친 창의 예시이다.

4. 정상적으로 입력했을 때, 리스트에 등록된 상태

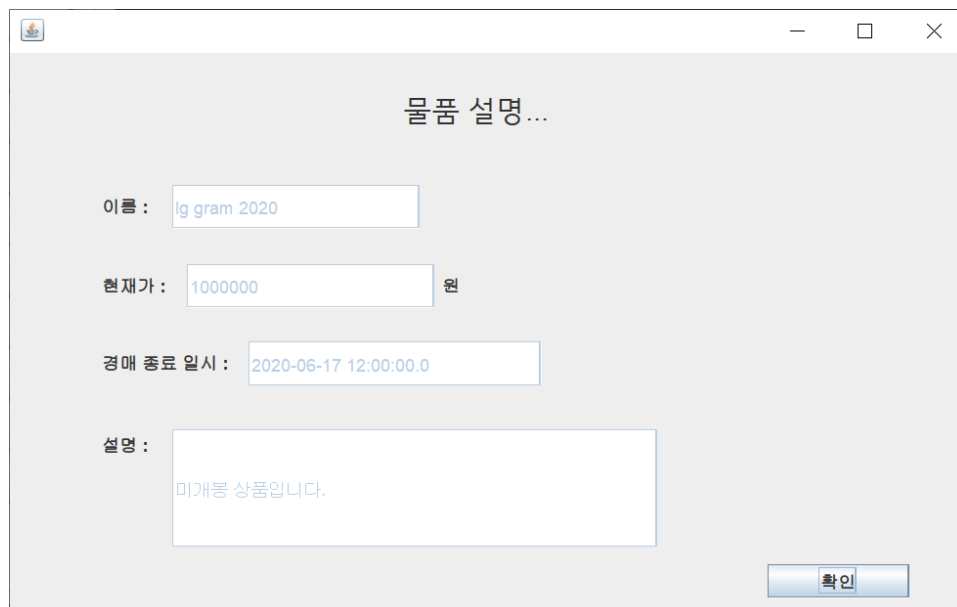


물품명	최고 입찰가	마감날짜	입찰
iPad3	45000	2020-06-15 12:09:0...	입찰
iPhone7	3000	2020-06-16 23:00:0...	입찰
iPhoneSE	23000	2020-06-16 13:00:0...	입찰
iPad5	1000	2020-06-15 23:20:0...	입찰
iPad6	1000	2020-06-15 23:21:0...	입찰
iPad4	1000	2020-06-15 23:22:0...	입찰
iMac	1000	2020-06-15 23:26:0...	입찰
iMac2	1000	2020-06-15 23:27:0...	입찰
gram	3444	2020-06-16 01:00:0...	입찰
lg gram 2020	1000000	2020-06-17 12:00:0...	입찰

로그아웃 목록 갱신 물품 등록

전의 그림에서 정상적으로 등록되었을 때, 맨 하단에 방금 등록한 물품이 리스트에 보이는 것을 확인할 수 있다.

5. 등록된 상품의 상세 정보가 정상 출력되는 화면



물품 설명...

이름 : lg gram 2020

현재가 : 1000000 원

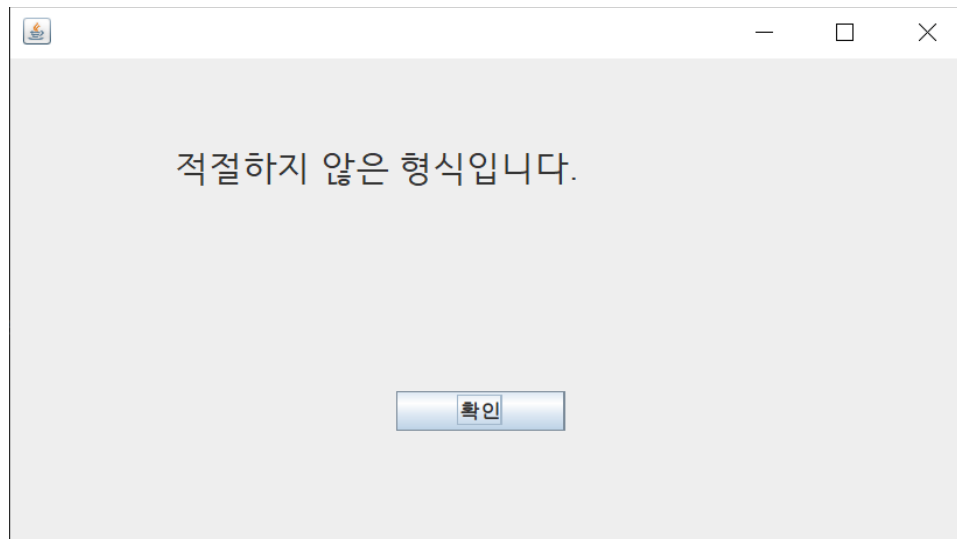
경매 종료 일시 : 2020-06-17 12:00:00.0

설명 :
미개봉 상품입니다.

확인

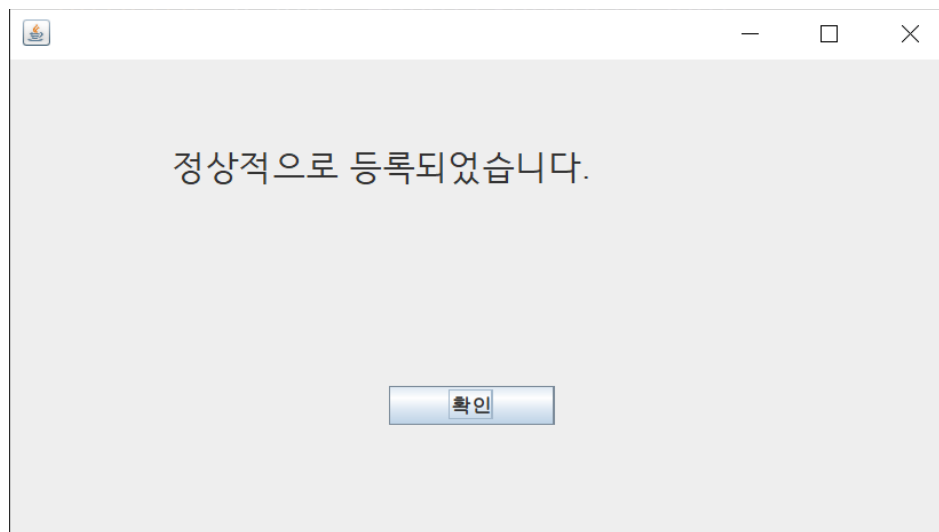
물품명을 더블 클릭을 할 경우, 등록된 해당 물품의 상세 정보가 정상적으로 등록되었다는 사실을 확인할 수 있다.

6. 물품 등록 시, 입력사항 형식이 맞지 않을 때 출력되는 에러메시지



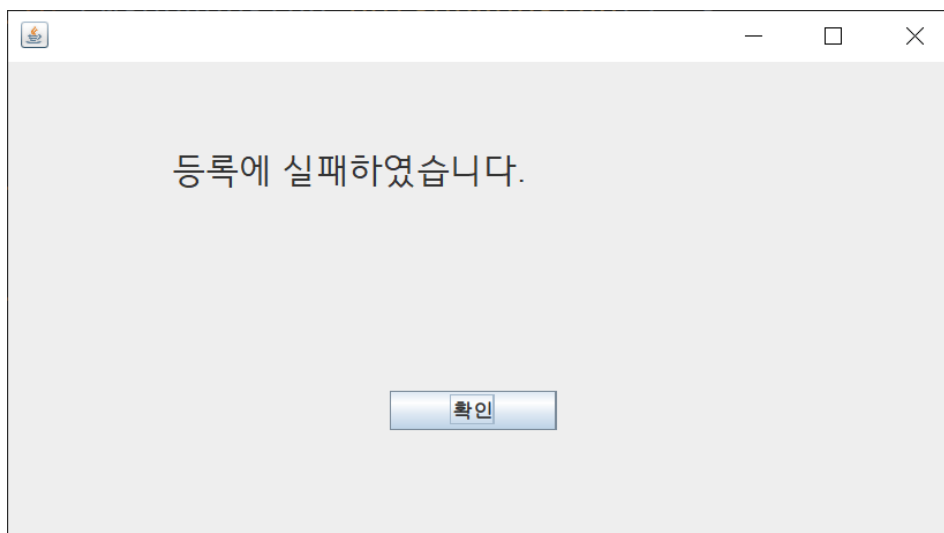
만약, 위에 적합한 형식이 아닐 경우, 오류 메시지를 출력한다.

7. 물품이 정상 등록되었을 때, 메시지 화면



만약, 정상적으로 등록이 된 경우, 정상적으로 등록되었다는 메시지를 출력한다.

8. 등록에 실패하였을 때, 오류 메시지 화면



외부적인 이유(DB 접속 불량, 서버 통신 불량) 등으로, 등록이 실패한 경우, 등록에 실패하였다는 메시지를 출력한다.


```

if(res == 1) {
    this.sendItemList(due.getSender());
    ackMsg.setDummyInfo("EnrollItemAck#1");
}
else {
    ackMsg.setDummyInfo("EnrollItemAck#0");
}
m_serverStub.send(ackMsg, due.getSender());

```

3. AuctionClientEventHandler.java

결과 상태에 대한 더비 이벤트 메시지를 클라이언트는 다시 받게 되고, 해당 메시지를 파싱을 한다. 메시지에 있는 데이터에 따라, 사용자는 GUI 메시지를 볼 수 있게 된다.

```

else if(tmp[0].equals("EnrollItemAck")) {
    EnrollResult er = m_client.getEnrollResult();
    switch (tmp[1]) {
        case "0":
            er.setMsg("등록에 실패하였습니다.");
            break;
        case "1":
            er.setMsg("정상적으로 등록되었습니다.");
            break;
        default:
            er.setMsg("등록에 실패하였습니다.");
            break;
    }
    er.frame.setVisible(true);
    m_client.getEnrollItem().getJFrame().dispose();
}

```

5.1.6 갱신

<기존 시나리오>

존재하지 않음

<구현 시나리오>

해당 부분에 대한 설계가 빠져 있어 추가 구현하였다.

<사용 CM API>

- CMInteractionInfo
- CMDummyEvent
- CMUser
- CMDBManager

<프로세스>

1. 목록 갱신 버튼을 클릭하여 기존의 AuctionGUI 창을 dispose()한다.

2. DB item table 정보를 다시 읽어와 목록을 최신화한다.



<구현 코드>

AuctionGUI.java

```

JButton btnNewButton_2 = new JButton("목록 갱신");
btnNewButton_2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ///목록 갱신하기
        CMInteractionInfo interInfo = m_clientStub.getCMInfo().getInteractionInfo();
        CMUser myself = interInfo.getMyself();

        CMDummyEvent due = new CMDummyEvent();

        String tmp = "updateList";

        due.setDummyInfo(tmp);

        due.setHandlerSession(myself.getCurrentSession());
        due.setHandlerGroup(myself.getCurrentGroup());

        m_clientStub.send(due, "SERVER");
    }
});

```

```

        frame.dispose();
    }
});

```

AuctionServerEventHandler.java

```

else if(data[0].equals("updateList")){
    sendItemList(due.getSender());
}

```

```

private void sendItemList(String clientName){
    CMDummyEvent due = new CMDummyEvent();
    due.setHandlerSession(m_serverStub.getMyself().getCurrentSession());
    due.setHandlerGroup(m_serverStub.getMyself().getCurrentGroup());
    String tmp="";
    try{
        ResultSet rs = CMDBManager.sendSelectQuery("SELECT name,now_price,due_date FROM
item", m_serverStub.getCMInfo());
        while(rs.next()) {
            tmp += rs.getString(1)+"#";
            tmp += rs.getString(2)+"#";
            tmp += rs.getString(3)+"#";
            tmp += "입찰"+"#";
        }
    }catch(SQLException e) {
    }
    tmp=tmp.substring(0, tmp.length()-1);
    System.out.println(tmp);
    due.setDummyInfo(tmp);
    m_serverStub.send(due, clientName);
}

```

5.1.7 낙찰

<기존 시나리오>

존재하지 않음

<구현 시나리오>

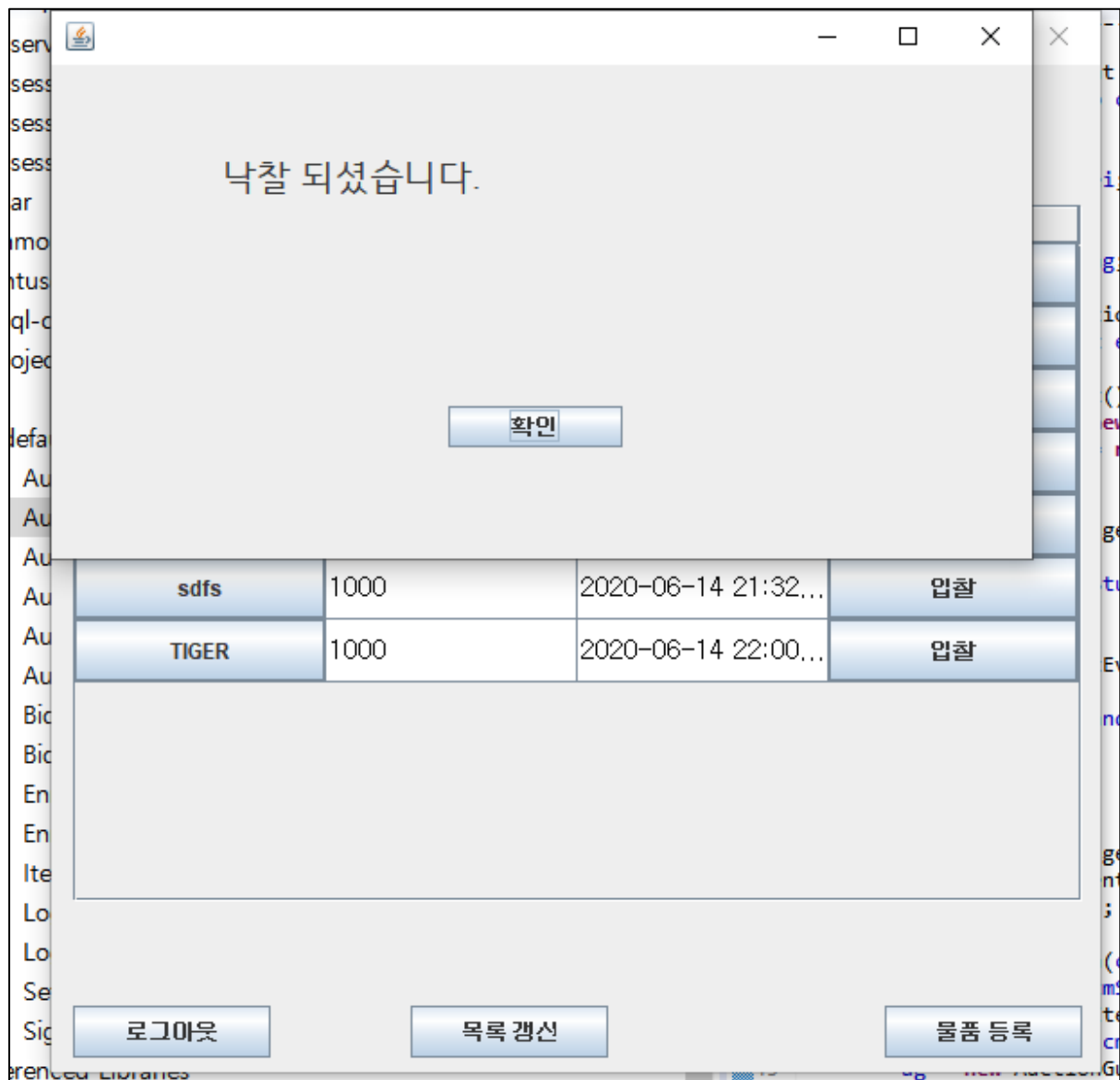
해당 부분에 대한 설계가 빠져 있어 추가 구현하였다.

<사용 CM API>

- CMInteractionInfo
- CMDummyEvent
- CMUser
- CMDBManager

<프로세스>

1. PrintTimer 함수는 AuctionServer.java를 실행한지 30초 후에 시작이 되고 매 30초마다 DB로 부터 item의 정보를 가지고 온다.
2. 현재 시각과 DB에 저장되어 있는 item의 마감시간의 차를 구하고 만약 그 차이가 0보다 큰 경우 시간이 지났으므로 해당 물품에 입찰하여 최고 입찰자인 클라이언트에게 Alert라는 더미 이벤트를 전송한다.
3. ActionClientEventHandler.java에서Alert 더미 이벤트를 받은 낙찰자에 "낙찰 되었습니다." 라는 GUI 창을 띄운다.



<구현 코드>

AuctionServer.java

```

public static class PrintTimer {
    public void start() {
        ScheduledJob job = new ScheduledJob();
        Timer jobScheduler = new Timer();
        jobScheduler.scheduleAtFixedRate(job, 30000, 30000);
    }
    //jobScheduler.cancel();
}

static class ScheduledJob extends TimerTask {
    public void run() {

        CMInteractionInfo interInfo = m_serverStub.getCMInfo().getInteractionInfo();
        CMUser myself = interInfo.getMyself();
        CMDummyEvent due2 = new CMDummyEvent();
        due2.setHandlerSession(myself.getCurrentSession());
        due2.setHandlerGroup(myself.getCurrentGroup());
        String tmp="";
        try{
            ResultSet rs = CMDBManager.sendSelectQuery("SELECT due_date, status, bid_winner FROM item WHERE status = 'f'", m_serverStub.getCMInfo());
            while(rs.next()) {
                tmp += rs.getString(1)+"#";
                tmp += rs.getString(2)+"#";
                tmp += rs.getString(3)+"#";
            }

            String[] tmp2 = tmp.split("#");
            System.out.println(tmp.length);
            int rowLength = tmp2.length / 3;

            String[][] input = new String[rowLength][3];

```

JobScheduler.scheduleAtFixedRate를 사용해 AuctionServer.java를 실행하고 최초 30초후에 job을 실행한다. 그리고 매 30초마다 실행한다. (매 3초마다 진행상태 었을 경우 DB Connection이 너무 많ㄴ다는 에러가 발생해서 30초로 바꿨습니다.)

CMDBManager.sendSelectQuery를 통해 ,status(낙찰 상태)가 'f'인 due_date(마감 날짜), status(낙찰 상태), bid_winner(최고가 입찰자)를 불러온다.

```
for(int i=0;i<rowLength;i++) {
    for(int j=0;j<3;j++) {
        input[i][j]=tmp2[3*i+j];
    }
}

for(int i=0;i<rowLength;i++) {

    String from1 = "";
    String from2 = "";

    // 현재시간
    SimpleDateFormat format1 = new SimpleDateFormat ( "yyyy-MM-dd HH:mm:ss");
    String format_time1 = format1.format (System.currentTimeMillis());

    from1 = format_time1;
    SimpleDateFormat transFormat1 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Date to1 = transFormat1.parse(from1);

    // 마감시간
    from2 = input[i][0];
    SimpleDateFormat transFormat2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Date to2 = transFormat2.parse(from2);

    long diff = to1.getTime() - to2.getTime();
    long min = diff/ 6000;

    System.out.println("현재시간"+to1.getTime());
    System.out.println("반품시간"+to2.getTime());
    System.out.println("현재시간 - 반품시간"+min);
}
```

SendSelectQuery를 통해 받은 데이터를 “#”으로 split하여 input 배열에 넣는다.

매 행마다 현재 시간과 item의 마감시의 차를 구한다.

```

        if (min > 0) {
            if (input[i][1].equals("f")) {
                String query = String.format("UPDATE item SET status = 't' WHERE bid_winner = '%s'", input[i][2]);
                CMDBManager.sendUpdateQuery(query, m_serverStub.getCMInfo());
                CMDummyEvent due2 = new CMDummyEvent();
                due2.setHandlerSession(myself.getCurrentSession());
                due2.setHandlerGroup(myself.getCurrentGroup());
                due2.setDummyInfo("Alert");
                m_serverStub.send(due2, input[i][2]);
            }
        }
    }

    } catch (SQLException e) {
    } catch (ParseException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    tmp=tmp.substring(0, tmp.length()-1);
    System.out.println(tmp);
    //due2.setDummyInfo(tmp);
    //m_serverStub.send(due2, se.getSender());
}
}

public static void main(String[] args) {
    AuctionServer server = new AuctionServer();
    CMServerStub cmStub = server.getServerStub();
    cmStub.setAppEventHandler(server.getServerEventHandler());
    cmStub.startCM();

    PrintTimer pt = new PrintTimer();
    pt.start();
}
}

```

만약 마감시간의 차이 min이 0보다 크게 된다면 CMDBManager.sendUpdateQuery를 통해 해당 item의 status를 "t"로 바꿔주고 (낙찰이 되었음을 표시하기 위해) due2.setDummyInfo("Alert")를 해당 item의 낙찰자인 bid_winner(input[i][2])에게 전송한다.

AuctionClientEventHandler.java

```

} else if(tmp[0].equals("Alert")) {
    EnrollResult er = m_client.getEnrollResult();
    er.setMsg("낙찰 되었습니다.");
    er.frame.setVisible(true);
    m_client.getEnrollItem().getJFrame().dispose();
}
}

```

해당 item의 낙찰자인 bid_winner는 더미 이벤트를 처리하는 AuctionClientEventHandler.java에서 "Alert" 알림이 왔을 경우에 해당하는 로직에 따라 낙찰자의 CM 화면에 "낙찰 되었습니다."라는 메시지가 보이게 된다.

6 Project result

: 사용자가 회원가입을 한 후, 로그인에서 인증과정을 거친 후 로그인을 하게된다. 로그인을 하면, 경매목록 창으로 넘어가게 되고, 경매 물품에 대한 입찰, 등록을 할 수 있게 된다. 경매목록에 나타나 있는 물품의 이름을 클릭시 해당 물품에 대한 상세정보를 얻을 수 있으며, 입찰 버튼을 클릭 시 그 물품에 대해 입찰을 할 수 있다. 이때 입찰가는 현재 입찰가보다 더 높은 금액으로 설정해야 할 수 있다. 입찰이 완료되고, 마감시간이 되면, 낙찰이 된다. 이때 가장 최고금액을 입력한 사용자는 낙찰알림을 받게되고, 그 물품은 목록에서 낙찰표시가 된다. 낙찰표시된 물품은 입찰을 수행할 수 없다. 사용자가 새로운 물품을 등록하는 물품 등록 버튼을 클릭하면, 물품 정보를 입력하는 폼이 주어지고, 그 내용을 토대로 등록을 하게 된다.

7 Video and Github URL

Video : <https://youtu.be/UJA0yVQEA3Y>

Github : https://github.com/kdje0506/CM_Project