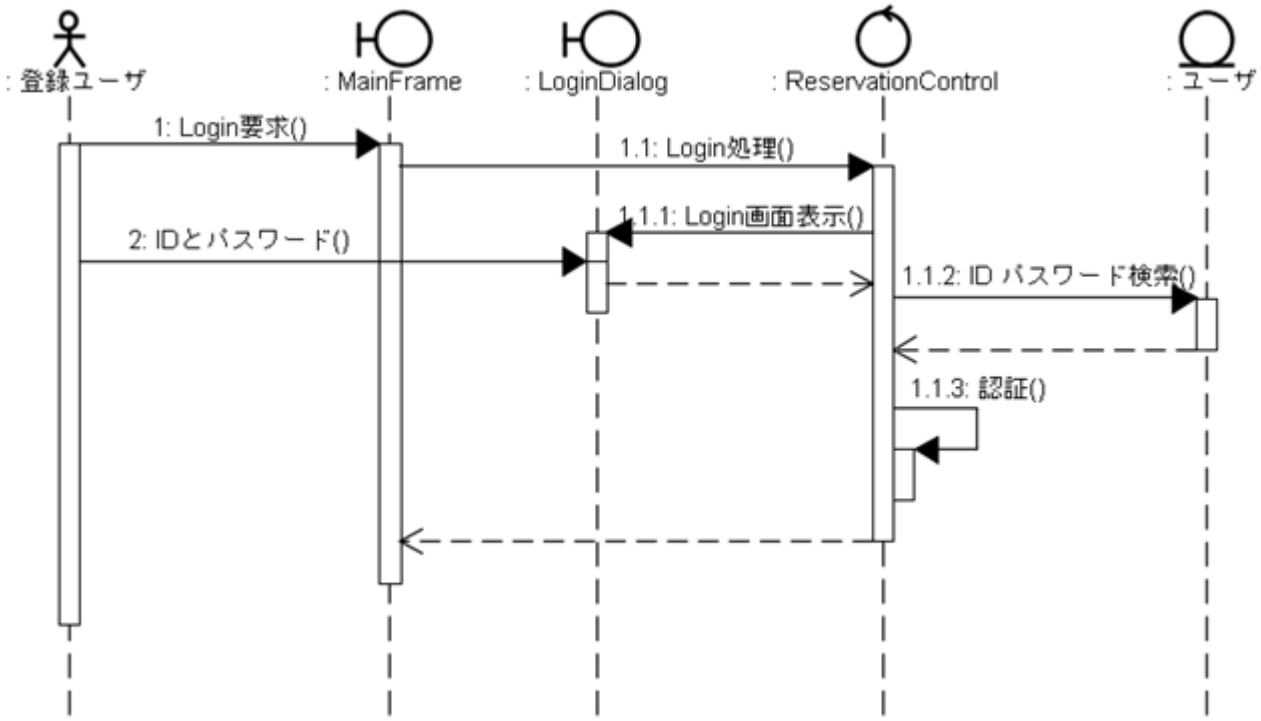
イントロダクション

利用者用サブシステムのバージョン2を作成します．今回のバージョンでは，ログインの処理を実装します．余力がある学生は，ログイン後に使用可能となる自分の予約状況を確認する機能を実装しましょう．

ログイン処理の実装方針

ログイン処理のシーケンス図は以下のようになっていました.



ユーザがMainFrameのログインボタンをクリックすると，MainFrameはReservationControlにログイン処理を依頼します．ReservationControlはログインダイアログ(LoginDialog)を生成して表示し，ユーザからIDとパスワードを受け付けます．入力されたユーザIDのレコードをデータベースのユーザテーブルから検索し，パスワードを照合します．パスワードは本来，暗号化しておくべきですが，今回のプログラムでは，実習なので簡単にするために暗号化はしていません．パスワードが一致した場合は，ユーザ名を保存し，ログイン状態に設定して，ログインボタンはログアウトボタンに変更します．
一方，ログアウトボタンをクリックすると，ログアウトの処理を行います．これはログイン状態からログアウト状態(ログインしていない状態)に設定し，ログアウトボタンをログインボタンに変更します．
これらの処理を行うにあたり，MainFrameはボタンが，ログインボタンになっているか，ログアウトボタンになっているかは調べずに，そのボタンがクリックされたら，ログイン処理とログアウト処理の両方を行うReservationControlのloginLogoutメソッドを呼び出すことにします．loginLogoutメソッドには，引数としてMainFrameを持たせます．これは，以下の理由で必要になります．

- ・LoginDialogを生成する際に親ウィンドウを指定するため
- ・MainFrame上のログイン・ログアウトボタンのラベルを変更するため

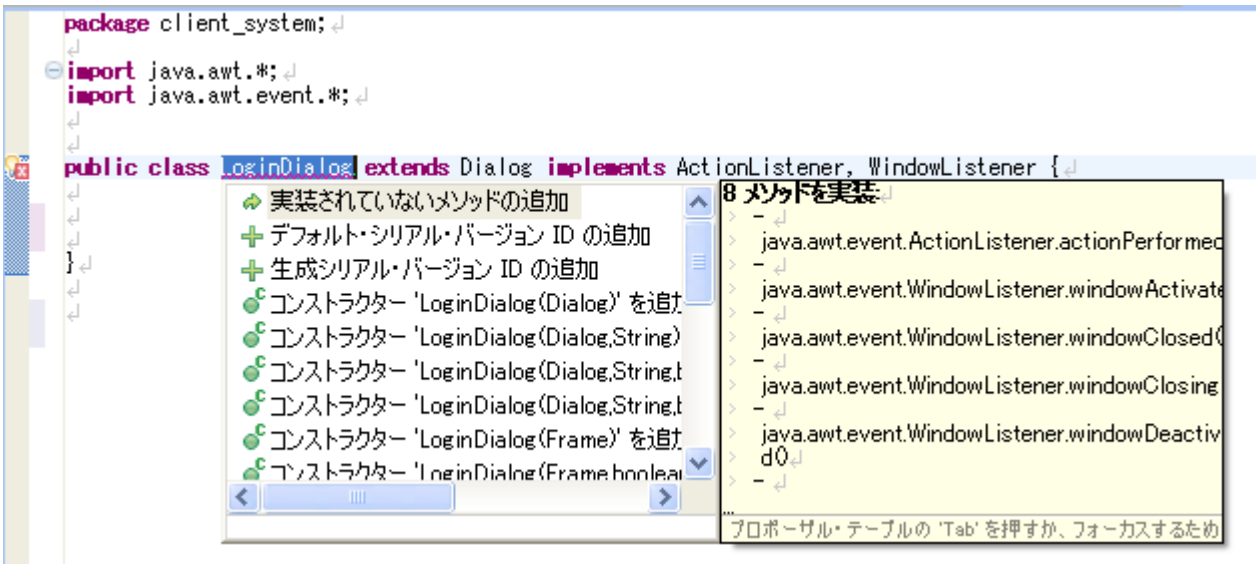
loginLogoutメソッドは今がログイン状態なのか，ログアウト状態なのかを判断して，上で述べたログイン処理とログアウト処理を実施します．

LoginDialogの実装

LoginDialogという名称で新規クラスを作成して，以下のようにjava.awt.*とjava.awt.event.*をインポートしておき，extendsでDialogを指定し，Dialogクラスを継承させます．また，implementsでActionListenerとWindouListenerを指定し，ボタンをクリックしたときと，ウィンドウを閉じるときのイベントに対する処理ができるようにしておきます．

```
package client_system;

import java.awt.*;
import java.awt.event.*;


public class LoginDialog extends Dialog implements ActionListener, WindowListener {



}
```

この状態で，以下のように，class LoginDialogの定義の左側の警告アイコンをクリックして，必要なイベントリスナのメソッドを実装します． この辺の方法は，MainFrameのイベントリスナを実装したときと同じです．



LoginDialogの変数

LoginDialogに定義する変数は以下の通りです．

```
boolean canceled;  // キャンセルのときはtrue OKおしたら false

TextField tfUserID;   //ユーザIDを入力するテキストフィールド
TextField tfPassword;  //パスワードを入力するテキストフィールド

Button buttonOK;     //OKボタン
Button buttonCancel;  //キャンセルボタン

Panel panelNorth;     //上部パネル
Panel panelMid;       //中央のパネル
Panel panelSouth;     //下部パネル
```

boolean型のcanceledは，OKボタンがクリックされたときはfalse，キャンセルボタンがクリックされた場合，ウィンドウが閉じられた場合はtrueを入れておきます．これによって，ユーザIDとパスワー

ドが入力されても，OKボタンが押されていない場合は，認証をしないようにします．
ユーザIDとパスワードを入力するためのテキストフィールド，OKボタンとキャンセルボタンの変数を定義します．また，上部にユーザID欄，中央にパスワード入力欄，下部に2つのボタンを配置するために，3つのパネルも定義しておきます．

LoginDialogのコンストラクタ

以下にLoginDialogのコンストラクタを示します．

```
public LoginDialog(Frame arg0) {

    //基底クラス(Dialog)のコンストラクタを呼び出す
    super(arg0,"Login",true);

    //キャンセルは初期値ではtrueとしておく
    canceled = true;

    // テキストフィールドの生成
    tfUserID = new TextField("",10);
    tfPassword = new TextField("",10);
    // パスワードを入力するテキストフィールドは入力した文字が * になるようにする
    tfPassword.setEchoChar('*');

    // ボタンの生成
    buttonOK = new Button("OK");
    buttonCancel = new Button("キャンセル");

    // パネルの生成
    panelNorth = new Panel();
    panelMid = new Panel();
    panelSouth = new Panel();

    // 上部パネルに，ユーザIDテキストフィールドを追加
    panelNorth.add( new Label("ユーザID"));
    panelNorth.add(tfUserID);

    // 中央パネルに，パスワードテキストフィールドを追加
    panelMid.add( new Label("パスワード"));
    panelMid.add(tfPassword);

    // 下部パネルに2つのボタンを追加
    panelSouth.add(buttonCancel);
    panelSouth.add(buttonOK);

    // LoginDialogをBorderLayoutに設定し，3つのパネルを追加
    setLayout( new BorderLayout());
    add( panelNorth,BorderLayout.NORTH);
    add( panelMid, BorderLayout.CENTER);
    add( panelSouth, BorderLayout.SOUTH);

    // ウィンドウリスナを追加
    addWindowListener(this);
    // ボタンにアクションリスナを追加
    buttonOK.addActionListener(this);
    buttonCancel.addActionListener(this);

    // 大きさの設定，ウィンドウのサイズ変更不可の設定
    this.setBounds( 100, 100, 350, 150);
    setResizable(false);
}
```

注釈を細かく入れておきましたので，どのようなことをやっているかは注釈を読んでください．これまでのMainFrameのコンストラクタに比較して，以下の点はポイントとなります．

- 最初に，基底クラス(親クラス)のコンストラクタを明示的に呼び出しています．
- パスワードを入力するためのテキストフィールドは，tfPassword.setEchoChar('*')によって表示される文字が*になるようにしています．
- 最後の部分で，ウィンドウのサイズを指定し，ユーザがウィンドウのサイズ変更をできないように設定しています．

メインウィンドウ(MainFrame)では，サイズ変更をできないように設定していないのに，ログインダイアログ(LoginDialog)では設定していることには特に意味はありません．

イベントリスナでの処理

actionPerformedに以下の処理を追加します．

```
@Override
public void actionPerformed(ActionEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ
    if ( arg0.getSource() == buttonCancel){
        canceled = true;
    } else if ( arg0.getSource() == buttonOK){
        canceled = false;
    }
    setVisible(false);
    dispose();
}
```

If文で判断して，OKボタンがクリックされたときはcanceledをfalseに，キャンセルボタンがクリックされたときはcanceledをtrueに設定します．その後，setVisible(false)でLoginDialogを非表示にし，dispose()で破棄して終了します．

また，windowClosing()にも，以下のように，キャンセルボタンをクリックしたときと同様の処理を追加します．

```
@Override
public void windowClosing(WindowEvent e) {
    // TODO 自動生成されたメソッド・スタブ
    setVisible(false);
    canceled = true;
    dispose();
}
```

ReservationControlにおける実装

変数とコンストラクタ

ReservationControlクラスの変数の定義の下に，以下のような新しい変数とコンストラクタのコードを追加します．

```
//この予約システムのユーザIDとログイン状態
String reservation_userid;
private boolean flagLogin; //ログインしていればtrue

ReservationControl(){
    flagLogin = false;
}
```

ここで追加した変数は，ログインをしたユーザのユーザID reservation_useridとログインの状態flagLoginです．コンストラクタではflagLoginをfalse(ログインしていない状態)に設定しておきます．

ログインログアウトの処理をするメソッド

ログインとログアウトの処理を行う以下のようなメソッドを追加します．loginLogoutメソッドはMainFrameを引数として，結果を文字列で返します．

```
////// ログイン・ログアウトボタンの処理
public String loginLogout( MainFrame frame){
    String res="";  //結果を入れる変数

    if ( flagLogin){    //ログアウトを行う処理
        flagLogin = false;
        frame.buttonLog.setLabel(" ログイン ");
    } else {            //ログインを行う処理

        //ログインダイアログの生成と表示
        LoginDialog ld = new LoginDialog(frame);
        ld.setVisible(true);
        ld.setModalityType(Dialog. ModalityType.APPLICATION_MODAL);

        //IDとパスワードの入力がキャンセルされたら，空文字列を結果として終了
        if ( ld.canceled){
            return "";
        }

        // ユーザIDとパスワードが入力された場合の処理
        // ユーザIDは他の機能のときに使用するのでメンバー変数に代入
        reservation_userid = ld.tfUserID.getText();
        // パスワードはここでしか使わないので，ローカル変数に代入
        String password = ld.tfPassword.getText();

        //(1) MySQLを使用する準備
        connectDB();

        //(2) MySQLの操作(SELECT文の実行)
        try {
            // userの情報を取得するクエリ
            String sql = "SELECT * FROM db_reservation.user WHERE user_id ='" + reservation_userid + "';";
            // クエリーを実行して結果セットを取得
            ResultSet rs = sqlStmt.executeQuery(sql);
            // 検索結果に対してパスワードチェック
            if (rs.next()){
                String password_from_db = rs.getString("password");
                if ( password_from_db.equals(password)){ //認証成功：データベースのIDとパスワードに一致
                    flagLogin = true;
                  frame.buttonLog.setLabel("ログアウト");
                  res = "";
                } else { // 認証失敗：パスワードが不一致
                    res = "ログインできません．ID パスワードが□違います．";
                }
            } else { //認証失敗；ユーザIDがデータベースに存在しない
                res = "ログインできません．ID パスワードが□違います．";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        //(3) MySQへの接続切断
        closeDB();
    }
    return res;
}
```

最初にメンバー変数flagLoginを調べてログイン状態かどうかを判断します．flagLoginがtrueのときは，ログイン状態なので，ログアウトを行う処理を実施します．具体的にはログオン状態を表すflagLoginにfalseを入れて，buttonLogのラベルを「ログイン」に変更します．

flagLoginがfalseのとき，つまり上のソースリストでは else 以降の場合，ログインをしていない状態なので，ログインの処理を行います．おおまかな流れとしては，ログインダイアログ(LoginDialog)の生成と表示，ユーザ情報のデータベースへの問い合わせ，パスワードの確認の順になります．

最初にLoginDialogを生成し，setVisibleで可視化しています．LoginDialogに対して，APPLICATION_MODALに設定します．APPLICATION_MODALに設定するとLoginDialogの親となるMainFrameへの操作がブロックされます．つまり，LoginDialogへの操作を終了しないと，MainFrameへの操作はできなくなります．LoginDialogへの操作が終了すると，次のif文の処理の前に戻ります．If文ではログインダイアログでOKボタンが押されたのか，キャンセルされたのかを調べます．キャンセルされた場合は，空文字列を結果として返し，何も実行しません．
ログインダイアログからユーザIDとパスワードが入力された場合は，そのユーザIDをメンバー変数に保存しておきます．パスワードの方はメソッド内のローカル変数に入れておきます．次に，MySQLへSELECT文でユーザIDが一致するレコードを検索します．レコードが得られた場合，つ

まり，rs.next()という次のレコードを取り出す処理が成功した場合は，そのレコードのpasswordのカラムの値を取り出し，ログインダイアログに入力されたパスワードと比較します．等しい場合は認証成功で，ログイン状態を表すflagLoginをtrueにします．また，buttonLogのラベルを「ログアウト」に変更します．結果は，特にメッセージを表示しないので，空文字列とします．
パスワードが一致しなかった場合とユーザIDが等しいレコードが得られなかった場合(rs.next()が失敗した場合)は，「ログインできません． ID パスワードが違います」という結果を返します．処理としては，ユーザIDが存在しなかったのか，パスワードが違ったのかは判別できるのですが，不正にアクセスしようとするユーザのことを考慮すると，どちらに該当するのかがわからない方がよいため，どちらも同じメッセージを返すという実装方法をとっています．

MainFrameにおける実装

MainFrameには，buttonLogがクリックされたときに，reservationControlのloginLogoutメソッドを呼び出すコードを追加します．以下の赤い四角で囲んだ部分です．課題11bを実施した方は，buttonVacancyをクリックしたときのために，else if 文がもう一つ追加されていることと思います．

```
@Override
public void actionPerformed(ActionEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ
    String result = new String();
    textMessage.setText("");
    if ( arg0.getSource() == buttonVacancy){ //// 空き状況確認ボタン
        result = reservationControl.getReservationOn( choiceFacility.getSelectedItem(),
                tfYear.getText(), tfMonth.getText(), tfDay.getText());
    } else if (arg0.getSource() == buttonLog){
        result = reservationControl.loginLogout(this);
    }
    textMessage.setText(result);
}
```

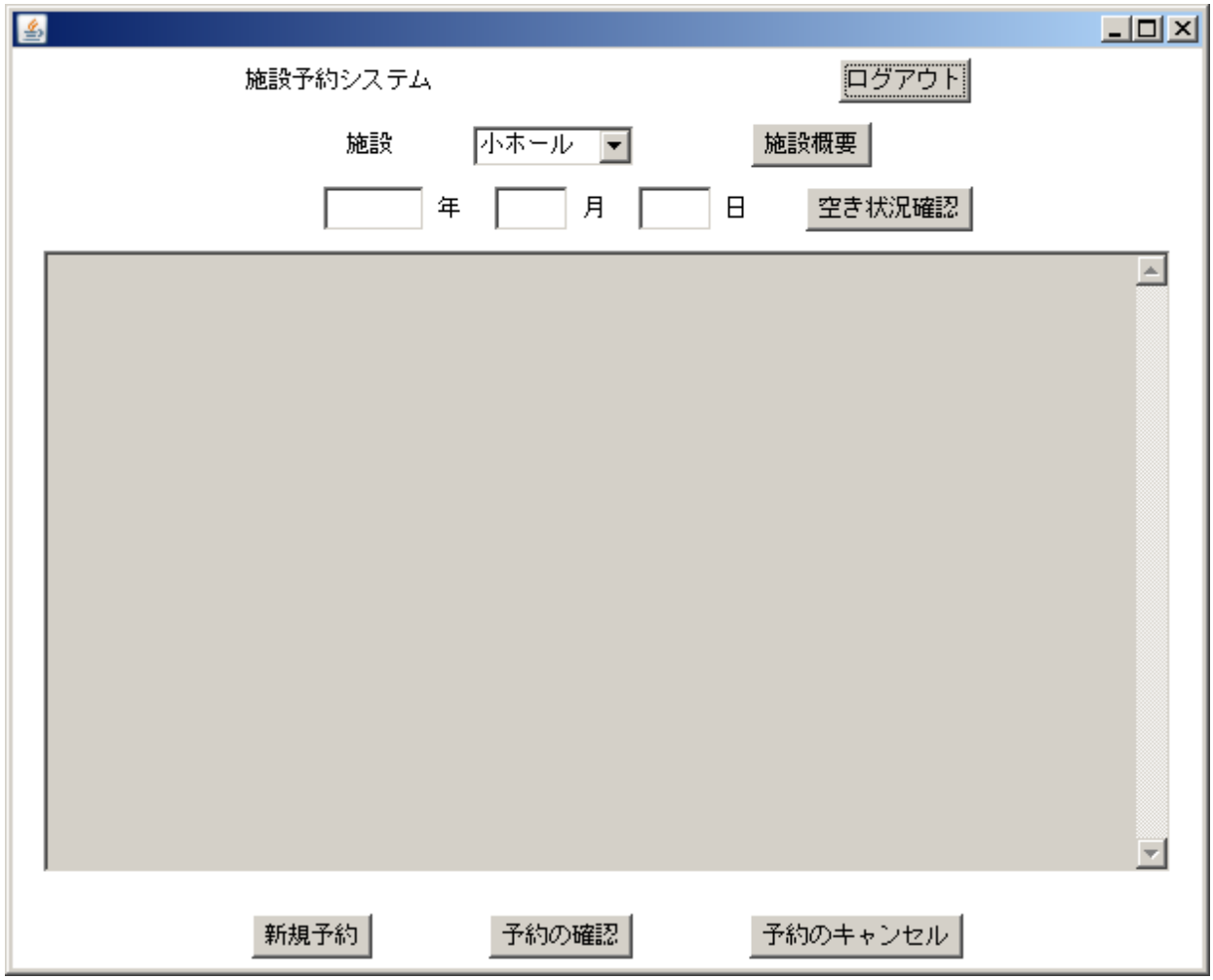
メソッドloginLogoutはMainFrameの引数をとりますが，自分自身を引数として与えるのでthisを指定しています．結果は resultに入れています．

実行例

以上の実装ができたら，実行してみましょう． MySQLを起動していないと正しく動作しませんので，注意してください．

以下はログインボタンをクリックし，ログインダイアログが表示され，IDとパスワードを入力した画面です．

ここで，OKをクリックすると，以下のようにログインボタンがログアウトボタンに変わります．



ログアウトボタンをクリックすると，ログアウトボタンがログインボタンに変わることも確認しておきましょう．また，ログインダイアログで誤ったユーザIDとパスワードを入力した場合は，以下のようにメッセージが表示されます．


施設予約システム
ログイン
施設
小ホール
施設概要
年
月
日
空き状況確認
ログインできません. ID パスワードが　違います.
新規予約
予約の確認
予約のキャンセル

プログラムリスト

version 2のプログラムリストをまとめて示しておきます.

LoginDialogクラス

```
package client_system;

import java.awt.*;
import java.awt.event.*;


public class LoginDialog extends Dialog implements ActionListener, WindowListener
{

        boolean canceled;       // キャンセルのときはtrue OKおしたら false

        TextField tfUserID;             //ユーザIDを入力するテキストフィールド
        TextField tfPassword;   //パスワードを入力するテキストフィールド

        Button buttonOK;                //OKボタン
        Button buttonCancel;    //キャンセルボタン

        Panel panelNorth;               //上部パネル
        Panel panelMid;                 //中央のパネル
        Panel panelSouth;               //下部パネル

        public LoginDialog(Frame arg0) {

                //基底クラス(Dialog)のコンストラクタを呼び出す
                super(arg0,"Login",true);

                //キャンセルは初期値ではtrueとしておく
                canceled = true;

                // テキストフィールドの生成
                tfUserID = new TextField("",10);
                tfPassword = new TextField("",10);
                // パスワードを入力するテキストフィールドは入力した文字が * になるようにする
                tfPassword.setEchoChar('*');

                // ボタンの生成
                buttonOK = new Button("OK");
                buttonCancel = new Button("キャンセル");

                // パネルの生成
                panelNorth = new Panel();
                panelMid = new Panel();
                panelSouth = new Panel();

                // 上部パネルに，ユーザIDテキストフィールドを追加
                panelNorth.add( new Label("ユーザID"));
                panelNorth.add(tfUserID);

                // 中央パネルに，パスワードテキストフィールドを追加
                panelMid.add( new Label("パスワード"));
                panelMid.add(tfPassword);

                // 下部パネルに2つのボタンを追加
                panelSouth.add(buttonCancel);
                panelSouth.add(buttonOK);

                // LoginDialogをBorderLayoutに設定し，3つのパネルを追加
                setLayout( new BorderLayout());
                add( panelNorth,BorderLayout.NORTH);
                add( panelMid, BorderLayout.CENTER);
                add( panelSouth, BorderLayout.SOUTH);

                // ウィンドウリスナを追加
                addWindowListener(this);
                // ボタンにアクションリスナを追加
                buttonOK.addActionListener(this);
                buttonCancel.addActionListener(this);
```

```
                // 大きさの設定，ウィンドウのサイズ変更不可の設定
                this.setBounds( 100, 100, 350, 150);
                setResizable(false);
        }



        @Override
        public void actionPerformed(ActionEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ
                if ( arg0.getSource() == buttonCancel){
                        canceled = true;
                } else if ( arg0.getSource() == buttonOK){
                        canceled = false;
                }
                setVisible(false);
                dispose();
        }

        @Override
        public void windowActivated(WindowEvent e) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowClosed(WindowEvent e) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowClosing(WindowEvent e) {
                // TODO 自動生成されたメソッド・スタブ
                setVisible(false);
                canceled = true;
                dispose();
        }

        @Override
        public void windowDeactivated(WindowEvent e) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowDeiconified(WindowEvent e) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowIconified(WindowEvent e) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowOpened(WindowEvent e) {
                // TODO 自動生成されたメソッド・スタブ

        }
}
```

ReservationControlクラス

```
package client_system;

import java.sql.*;
import java.awt.*;

public class ReservationControl {

        //MySQLデータベース接続のための変数
        Connection sqlCon;
        Statement sqlStmt;
        String sql_userid = "reservation_user"; //ユーザID
```

```
        String sql_password = "pass0004";       //パスワード

        //この予約システムのユーザIDとログイン状態
        String reservation_userid;
        private boolean flagLogin;      //ログインしていればtrue


        ReservationControl(){
                flagLogin = false;
        }


        //データベースの操作準備
        private void connectDB(){
                try{
              // ドライバクラスをロード
              Class.forName("org.gjt.mm.mysql.Driver"); // MySQLの場合

              // データベースへ接続
              String url = "jdbc:mysql://localhost?
useUnicode=true&characterEncoding=SJIS";
             sqlCon = DriverManager.getConnection(url, sql_userid, sql_password);

              // ステートメントオブジェクトを生成
              sqlStmt = sqlCon.createStatement();
                } catch (Exception e) {
                      e.printStackTrace();
            }
        }

        //データベースの切断
        private void closeDB(){
                try{
                        sqlStmt.close();
                        sqlCon.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

        ////指定した日，施設の 空き状況(というか予約状況)
        public String getReservationOn( String facility, String ryear_str, String
rmonth_str, String rday_str){

                String res = "";

                // 年月日が数字かどうかををチェックする処理
                try {
                        int ryear = Integer.parseInt( ryear_str);
                        int rmonth = Integer.parseInt( rmonth_str);
                        int rday = Integer.parseInt( rday_str);
                } catch(NumberFormatException e){
                        res ="年月日には数字を指定してください";
                        return res;
                }

                res =   facility + "  予約状況\n\n";

                // 月と日が一桁だったら，前に0をつける処理
                if (rmonth_str.length()==1) {
                        rmonth_str = "0" + rmonth_str;
                }
                if ( rday_str.length()==1){
                        rday_str = "0" + rday_str;
                }
                //SQLで検索するための年月日のフォーマットの文字列を作成する処理
                String rdate = ryear_str + "-" + rmonth_str + "-" + rday_str;

                //(1) MySQLを使用する準備
                connectDB();

                //(2) MySQLの操作(SELECT文の実行)
                try {
                      // 予約情報を取得するクエリ
                      String sql = "SELECT * FROM db_reservation.reservation
WHERE date ='" + rdate + "' AND facility_name = '"+ facility +"' ORDER BY
start_time;";
                      // クエリーを実行して結果セットを取得
                      ResultSet rs = sqlStmt.executeQuery(sql);
                      // 検索結果から予約状況を作成
                      boolean exist = false;
                      while(rs.next()){
                                String start = rs.getString("start_time");
                                String end = rs.getString("end_time");
```

```
                                res += "    " + start + " -- " + end + "\n";
                                exist = true;
                        }
                        if ( !exist){     //予約が１つも存在しない場合の処理
                            res = "予約はありません";
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }

                //(3) MySQLへの接続切断
                closeDB();

                return res;
        }

        ////// ログイン・ログアウトボタンの処理
        public String loginLogout( MainFrame frame){
                String res="";  //結果を入れる変数

                if ( flagLogin){        //ログアウトを行う処理
                        flagLogin = false;
                        frame.buttonLog.setLabel(" ログイン ");
                } else {                        //ログインを行う処理

                        //ログインダイアログの生成と表示
                        LoginDialog ld = new LoginDialog(frame);
                        ld.setVisible(true);

ld.setModalityType(Dialog.ModalityType.APPLICATION_MODAL);

                        //IDとパスワードの入力がキャンセルされたら，空文字列を結果として終了
                        if ( ld.canceled){
                                return "";
                        }

                        // ユーザIDとパスワードが入力された場合の処理
                        // ユーザIDは他の機能のときに使用するのでメンバー変数に代入
                        reservation_userid = ld.tfUserID.getText();
                        // パスワードはここでしか使わないので，ローカル変数に代入
                        String password = ld.tfPassword.getText();

                        //(1) MySQLを使用する準備
                        connectDB();

                        //(2) MySQLの操作(SELECT文の実行)
                        try {
                                // userの情報を取得するクエリ
                                String sql = "SELECT * FROM db_reservation.user
WHERE user_id ='" + reservation_userid + "';";
                                // クエリーを実行して結果セットを取得
                                ResultSet rs = sqlStmt.executeQuery(sql);
                                // 検索結果に対してパスワードチェック
                                if (rs.next()){
                                    String password_from_db =
rs.getString("password");
                                    if ( password_from_db.equals(password)){ //認証
成功：データベースのIDとパスワードに一致
                                            flagLogin = true;
                                                frame.buttonLog.setLabel("ログアウ
ト");
                                                res = "";
                                    } else {      //      認証失敗：パスワードが不一致
                                            res = "ログインできません．ID パスワードが
違います．";
                                    }
                                } else {  //認証失敗；ユーザIDがデータベースに存在しない
                                    res = "ログインできません．ID パスワードが 違いま
す．";
                                }
                        } catch (Exception e) {
                                e.printStackTrace();
                        }

                        //(3) MySQLへの接続切断
                        closeDB();
                }
                return res;
        }

}
```

MainFrameクラス

```
package client_system;

import java.awt.*;
import java.awt.event.*;


public class MainFrame extends Frame implements
ActionListener,WindowListener,KeyListener{

        ReservationControl reservationControl;

        Panel panelNorth;                       //上部パネル
        Panel panelNorthSub1;           //上部パネルの上
        Panel panelNorthSub2;           //上部パネルの中央
        Panel panelNorthSub3;           //上部パネルの下
        Panel panelMid;                         //中央パネル
        Panel panelSouth;                       //下部パネル

        Button buttonLog;                                       // ログイン ・ ログアウト ボ
タン
        Button buttonExplanation;               // 施設概要 説明ボタン
        Button buttonVacancy;                   //  空き状況確認
        Button buttonReservation;               // 新規予約ボタン
        Button buttonConfirm;                   // 予約の確認
        Button buttonCancel;                    // 予約のキャンセルボタン

        ChoiceFacility choiceFacility;                  // 施設選択用選択ボックス
        TextField tfYear,tfMonth,tfDay; // 年月日のテキストフィールド
        TextArea textMessage;                   // 結果表示用メッセージ欄

        public MainFrame( ReservationControl rc){

                reservationControl = rc;

                // ボタンの生成
                buttonLog = new Button(" ログイン ");
                buttonExplanation = new Button("施設概要");
                buttonVacancy = new Button("空き状況確認");
                buttonReservation = new Button("新規予約");
                buttonConfirm = new Button("予約の確認");
                buttonCancel = new Button("予約のキャンセル");

                // 設備チョイスボックスの生成
                choiceFacility = new ChoiceFacility();
                tfYear = new TextField("",4);
                tfMonth = new TextField("",2);
                tfDay = new TextField("",2);

                // 上中下のパネルを使うため，レイアウトマネージャーにBorderLayoutを設定
                setLayout( new BorderLayout());


                // 上部パネルの上パネルに　予約システム というラベルと [ログイン]ボタンを追加
                panelNorthSub1 = new Panel();
                panelNorthSub1.add(new Label("施設予約システム                  "));
                panelNorthSub1.add(buttonLog);

                // 上部パネルの中央パネルに　　施設　[施設名選択]チョイス　　[概要説明]ボタン
を追加
                panelNorthSub2 = new Panel();
                panelNorthSub2.add(new Label("施設  "));
                panelNorthSub2.add( choiceFacility);
                panelNorthSub2.add(new Label("   "));
                panelNorthSub2.add( buttonExplanation);

                // 上部パネルのしたパネルに年月日入力欄と 空き状況確認ボタンを追加
                panelNorthSub3 = new Panel();
                panelNorthSub3.add(new Label("  "));
                panelNorthSub3.add(tfYear);
                panelNorthSub3.add(new Label("年"));
                panelNorthSub3.add(tfMonth);
                panelNorthSub3.add(new Label("月"));
                panelNorthSub3.add(tfDay);
                panelNorthSub3.add(new Label("日 "));
                panelNorthSub3.add( buttonVacancy);
```

```
                // 上部パネルに３つのパネルを追加
                panelNorth = new Panel(new BorderLayout());
                panelNorth.add(panelNorthSub1, BorderLayout.NORTH);
                panelNorth.add(panelNorthSub2, BorderLayout.CENTER);
                panelNorth.add(panelNorthSub3, BorderLayout.SOUTH);
                // メイン画面(MainFrame)に上部パネルを追加
                add(panelNorth,BorderLayout.NORTH);


                // 中央パネルにテキストメッセージ欄を設定
                panelMid = new Panel();
                textMessage = new TextArea( 20, 80);
                textMessage.setEditable(false);
                panelMid.add(textMessage);
                // メイン画面(MainFrame)に中央パネルを追加
                add( panelMid,BorderLayout.CENTER);


                // 下部パネルにボタンを設定
                panelSouth = new Panel();
                panelSouth.add(buttonReservation);
                panelSouth.add(new Label("   "));
                panelSouth.add(buttonConfirm);
                panelSouth.add(new Label("   "));
                panelSouth.add(buttonCancel);
                // メイン画面(MainFrame)に下部パネルを追加
                add( panelSouth,BorderLayout.SOUTH);

                //ボタンのアクションリスナの追加
                buttonLog.addActionListener(this);
                buttonExplanation.addActionListener(this);
                buttonVacancy.addActionListener(this);
                buttonReservation.addActionListener(this);
                buttonConfirm.addActionListener(this);
                buttonCancel.addActionListener(this);

                addWindowListener(this);
                addKeyListener(this);
        }


        @Override
        public void actionPerformed(ActionEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ
                String result = new String();
                textMessage.setText("");
                if ( arg0.getSource() == buttonVacancy){ //// 空き状況確認ボタン
                        result = reservationControl.getReservationOn(
choiceFacility.getSelectedItem(),
                                                        tfYear.getText(),
tfMonth.getText(), tfDay.getText());
                } else if (arg0.getSource() == buttonLog){
                        result = reservationControl.loginLogout(this);
                }
                textMessage.setText(result);
        }

        @Override
        public void windowActivated(WindowEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowClosed(WindowEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowClosing(WindowEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ
                System.exit(0);

        }

        @Override
        public void windowDeactivated(WindowEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowDeiconified(WindowEvent arg0) {
```

```
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowIconified(WindowEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void windowOpened(WindowEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void keyPressed(KeyEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void keyReleased(KeyEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

        @Override
        public void keyTyped(KeyEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ

        }

}
```
