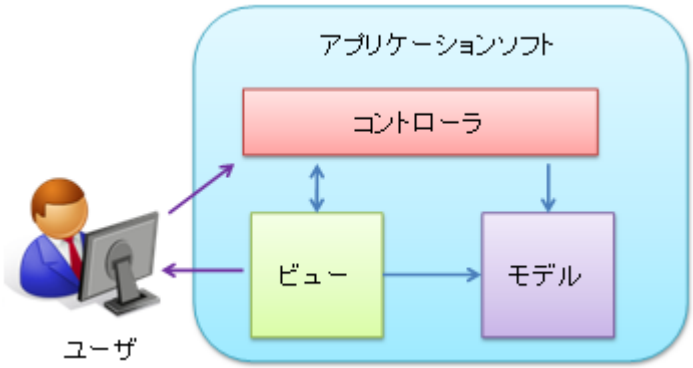
学習目標

今回の学習モジュールを修了すると，Java言語でSQLデータベースを操作するGUI(グラフィカル・ユーザ・インターフェース)のアプリケーションを作成することができるようになります．具体的には以下のことをできるようにしましょう．

- MVCと呼ばれるアーキテクチャがどのようなものか説明できる
- Javaプログラミングでウィンドウを表示させるアプリケーションはどのように作成すればよいか説明できる
- Javaプログラミングで，Frameクラスを拡張してテキストフィールド，ボタン等を配置したインタフェースを作成できる
- Frameクラスを拡張したクラスで，ボタンをクリックしたときの動作を記述できる

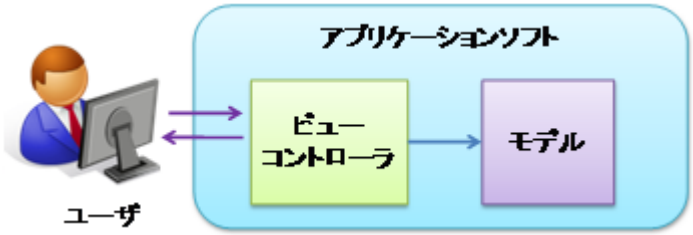
MVCアーキテクチャ

MVCはGUIのアプリケーションソフトの構造の基本的な指針と言えるものです．オブジェクト指向開発でのデザインパターンとしても採用されています．古くはSmallTalkというオブジェクト指向プログラミング言語でGUIのアプリケーションが開発された頃に提唱された考え方です．
MVCのMはModel，VはView，CはControllerを表しています．モデルはアプリケーションで扱う対象を表したものです．ビューはモデルをユーザに表示するウィンドウの部分，コントローラはユーザからの入力を受け取りビューとモデルを制御する部分です．これらの関係は以下の図のようになります．



アプリケーションソフトは，モデル，ビュー，コントローラに分けて実装します．アプリケーション内の矢印は，オブジェクト指向で言うメッセージを送る(メソッドを呼び出す)ことを表しています．つまり，ビューがモデルにメッセージを送り，その結果得た値を表示したり，コントローラがモデルにメッセージを送り，モデルを操作したりします．しかし，モデルから，コントローラやビューにメッセージが送られることはありません．コントローラとビューの間ではお互いにメッセージを送ることがあり得ます．

情報科学演習１の最後にCOMETシミュレータをJavaで作成しましたが，CPUやメモリの構造や動きを表すクラスがモデル，それをユーザに表示するFrameクラスがビューに対応します．COMETシミュレータの場合は，コントローラの役割もFrameクラスが行っていましたので，ビューとコントローラは同じでした．つまり，以下の図のようになります．



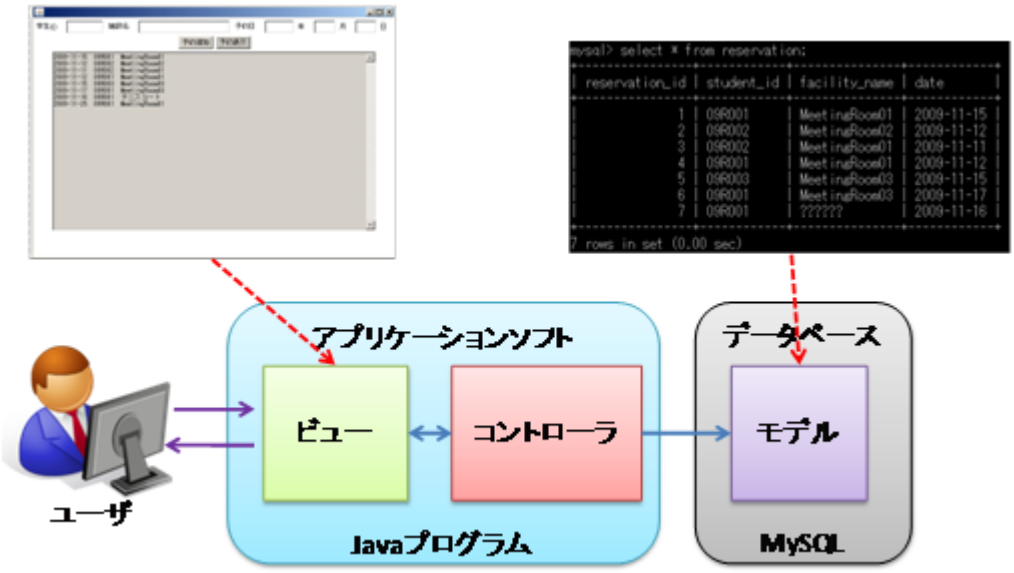
このようにビューとコントローラを同じクラスに実装する場合も少なくありません．実際，JavaでGUIのアプリケーションを作成しようとすると，FrameクラスやDialogクラスなどをビューとします．ユーザからの入力もそうしたFrameやDialogクラス上のテキストフィールドやボタンに行われるわけですから，単純なプログラムで入力の処理も同じクラスに実装したくなるのも自然です．また，逆に複雑なプログラムでビューとコントローラのやりとりが頻繁に生じ，同じクラスとして実装する方が有利な場合もあります．
いずれにせよ，モデルとビュー・コントローラを分割することで，プログラムコードはかなり整理整頓されます．あるモデルに対して複数のビューを使い分けることも可能になります．また，情報科学演習１の話になりますが，WCASL II を思い出してください．WCASL IIにはCPUやメモリのモデルを表すクラスと，アセンブラプログラミングを理解するためのCASLのビューとCPUの動作を理解するためのCOMETのビューがあり，それらを切り替えることができました．また，モデルとビューが分割さ

れていればモデルのテストも容易になります．つまり，まずモデルを作成し，動作をテストした上でビューとコントローラを作成してアプリケーションの全体を仕上げるといったことができるわけです．

例題プログラムの概要

では，MVCアーキテクチャに沿って実際にプログラムを作成してみましょう．第8回の授業で作成した予約テーブルの操作をするテストプログラムをGUIで作成してみます．
作成しようとするプログラムは以下のような画面で，学生ID,施設名，予約年月日を入力し，「予約追加」ボタンをクリックすると，予約情報がMySQLのreservationテーブルに追加され，「予約表示」ボタンをクリックすると，reservationテーブルに登録されているレコードがすべて表示されるといったものです．

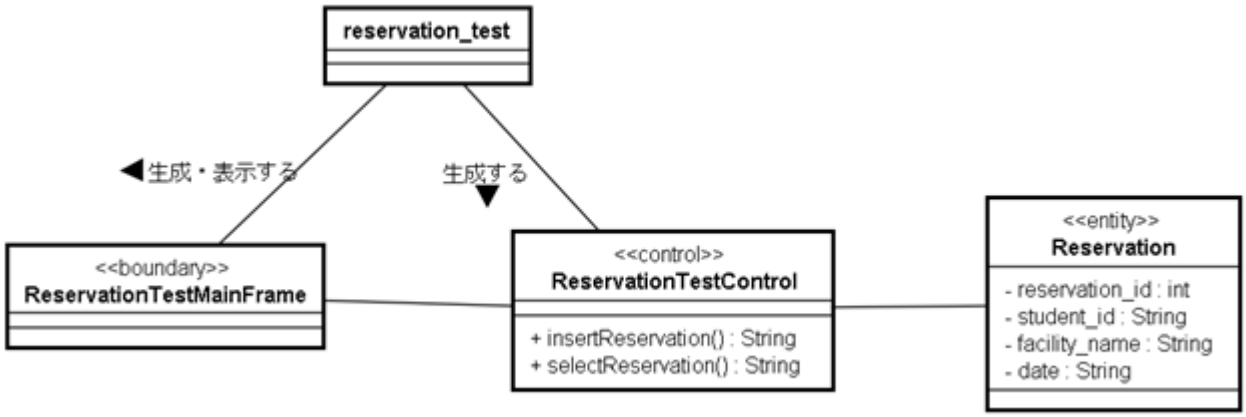
このプログラムでは，上の画面がビューになります．この画面は，JavaのFrameクラスを拡張したクラスにラベルやテキストフィールド，ボタンなどを配置して作成します．一方，SQLデータベースに保存されているデータがモデルになります．ビューから受け取った情報をもとにSQLデータベースを直接操作する部分をコントローラとして位置づけます．したがって，今回作成しようとするプログラムは以下のような構成になります．



モデルはMySQL上にあります．Javaプログラムでは，ビューとコントローラを実装し，コントローラからMySQL上のモデルを操作します．

例題プログラムのクラス図

例題プログラムのクラス構成をクラス図で表すと，以下のようになります．このクラス図内のクラスは実装するJavaプログラムのクラスと対応しています．このようなクラス図が設計レベルのクラス図になります．



ビューがReservationTestMainFrameクラス，コントローラがReservationTestControlクラスです．モデルはクラス図上ではReservationクラスとして表現されていますが，これは実際にはMySQL上のテーブルです．Javaプログラムはmainメソッドから動作が開始されます．そのmainメソッドを持たせるのが，reservation_testクラスです．mainメソッドでは，ReservationTestMainFrameとResevationTestControlのインスタンスを生成し，ReservationTestMainFrameを表示します．それ以降は，ユーザとのやりとりはReservationTestMainFrameが担当することになります．このようにアプリケーションソフトとユーザなどのアクターとの間を取り持つクラスはboundaryというステレオタイプで表されます．

例題プログラムversion1

プログラムを理解し，動作を確認しつつプログラミングをするために，例題プログラムを3段階に分けて作成しましょう．
最初に新しいプロジェクトを作成します．プロジェクト名はdb_win_applicationとしておきましょう．プロジェクトの作成方法は，第8回と同じですので，ここでは説明しません．外部jarファイルの追加でmysql-connector-javaを追加しておくのを忘れないようにしましょう．

ReservationTestControlクラス

次に3つのクラスを作成します．
1つめはコントローラのReservationTestControlです．クラスを新規作成すると，以下のようなプログラムリストが生成されますが，version 1ではこれはこのままにしておきます．

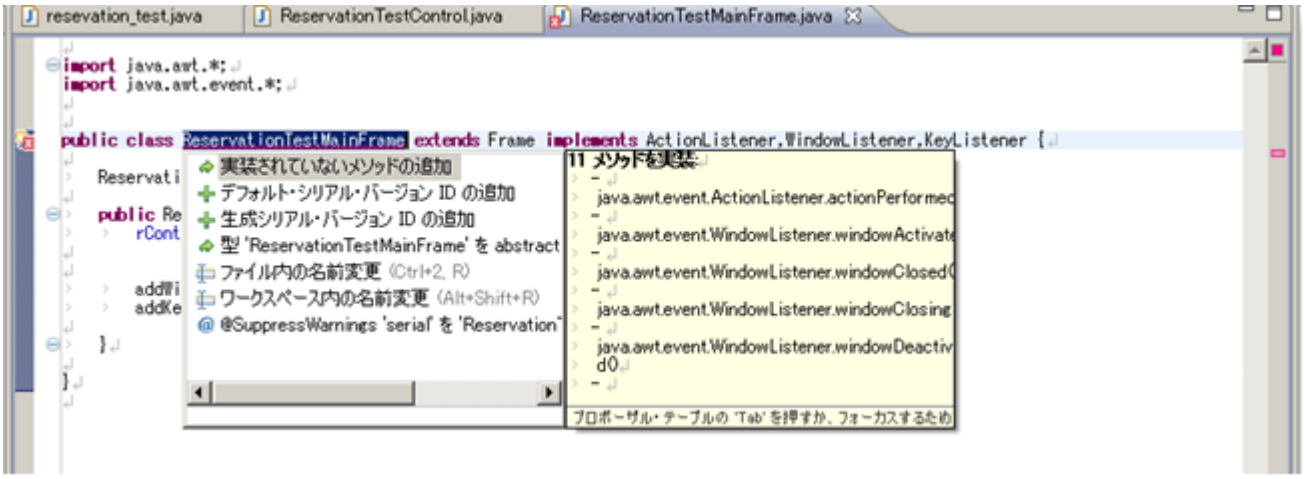
```
public class ReservationTestControl {

}
```

ReservationTestMainFrameクラス

次に，ビューとなるReservationTestMainFrameクラスです．クラスを新規作成した後，以下のように入力しましょう．

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

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

    ReservationTestControl rControl;

    public ReservationTestMainFrame( ReservationTestControl rc){
        rControl = rc;


        addWindowListener(this);
        addKeyListener(this);

    }

}
```

まず，awt.*とawt.event.*をインポートしておきます．次にextendsを使ってFrameクラスを継承するようにし，implementsを使ってActionListener，WindowListener，KeyListenerを指定します．コントローラのReservationTestControlにメッセージを送れるようにするために，変数rControlを設けておきます．コンストラクタReservationTestMainFrameでは引数としてReservationTestControlクラスのrcを持たせ，これを使ってrControlにコントローラのインスタンスを入れておきます．コンストラクタの最後には，WindowListenerとKeyListenerとしてthis(つまりReservationTestMainFrame)を追加する処理を記述します．
ここまでできたら，左に表示された警告のアイコンをクリックします．すると，以下の図のように「実装されていないメソッドの追加」が出てきますから，これをダブルクリックしてメソッドを自動的に追加します．



ここで追加したメソッドがいろいろなイベント(ボタンをクリックしたり，ウィンドウのアイコンをクリックしたりといった操作)によって動作するメソッドになります．
生成されたメソッドの中のwindowClosingというメソッドの中に，System.exit(0)というプログラムを終了させる命令を記述しましょう．以下の図の赤い四角の部分です．

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

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

    ReservationTestControl rControl;

    public ReservationTestMainFrame( ReservationTestControl rc){
        rControl = rc;


        addWindowListener(this);
        addKeyListener(this);

    }

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

    }

    @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
        System.exit(0);
    }
```

windowClosingは，ウィンドウの右上の×のアイコンをクリックしたときに実行されますので，そのときに，プログラムを終了させるようにするわけです．

reservation_testクラス

最後に，reservation_testクラスは，以下のように記述します．

```
public class resevation test {

    public static void main ( String argv[]){
        ReservationTestControl reservationControl = new ReservationTestControl();
        ReservationTestMainFrame mainFrame = new ReservationTestMainFrame(reservationControl);
        mainFrame.setBounds( 5,5,620,500);
        mainFrame.setVisible(true);
    }
}
```

mainメソッドの中で，ReservationTestControlクラスのインスタンスを生成し，それを引数としてReservationTestMainFrameクラスのインスタンスを生成してmainFrameに入れています．次にmainFrameに対してsetBoundsで表示場所とサイズを指定し，setVisibleで可視化しています．

実行例

以上により，単にウィンドウが表示され，右上の×をクリックすると閉じるだけのアプリケーションが完成しました．実行例は以下のようになります．

例題プログラム version 2

次に，インターフェースを実装してversion2とします．このバージョンにおいても，ReservationTestControlクラスは空のままです．

ReservationTestMainFrameクラス

「例題の概要」で述べたようなインターフェースを実装するために，以下のようなコードを追加します．赤の四角が追加した部分です．

```
public class ReservationTestMainFrame extends Frame implements ActionListener,WindowListener,KeyListener {

    ReservationTestControl rControl;

    TextField tfStudentID,tfFacilityName;    // 学生IDと施設のテキストフィールド
    TextField tfYear,tfMonth,tfDay;          // 年月日のテキストフィールド

    Button buttonInsert;      // 新規予約ボタン
    Button buttonSelect;      // レコードの検ボタン

    TextArea textMessage;                    // 検索結果・メッセージ欄

    public ReservationTestMainFrame( ReservationTestControl rc){
        rControl = rc;

        // 各フィールドの生成
        tfStudentID = new TextField("",6);
        tfFacilityName = new TextField("",20);
        tfYear = new TextField("",4);
        tfMonth = new TextField("",2);
        tfDay = new TextField("",2);

        // ボタンの生成
        buttonInsert = new Button("予約追加");
        buttonSelect = new Button("予約表示");

        // テキストエリアの生成と編集不可の設定
        textMessage = new TextArea( 20, 80);
        textMessage.setEditable(false);

        // ウィンドウへの登録
        setLayout( new FlowLayout());
        add( new Label("学生ID"));
        add( tfStudentID);
        add( new Label(" 施設名"));
        add( tfFacilityName);
        add( new Label(" 予約日"));
        add( tfYear);
        add( new Label("年"));
        add( tfMonth);
        add( new Label("月"));
        add( tfDay);
        add( new Label("日"));

        add( buttonInsert);
        add( buttonSelect);

        add(textMessage);

        buttonInsert.addActionListener(this);
        buttonSelect.addActionListener(this);

        addWindowListener(this);
        addKeyListener(this);

    }
```

最初の赤い四角はクラスで使用するコンポーネントの変数を定義している部分です．学生ID，施設名，年，月，日のためのテキストフィールド，「予約追加」と「予約表示」の2つのボタン，結果を表示するためのテキストエリアの変数を定義します．

下の赤い四角は，コンストラクタの中での初期化の処理です．まず，テキストフィールドを生成し，それぞれの変数に入れます．TextFieldの第1引数はあらかじめ設定しておく文字列なので，すべて空文字にしています．第2引数はテキストフィールドのサイズで，それぞれのフィールドに対して適宜指定しています．次に，ボタンを生成しています．Buttonに与える引数でボタンに表示されるラベルを指定します．ここでは「予約追加」と「予約表示」になっています．その後の2行はテキストエリアの生成と設定です．TextAreaには行数と文字数を引数に与えて大きさを設定します．また，今回のプログラムでは編集する必要がないので，setEditableで編集不可に設定しています．
このようにして生成したコンポーネントをウィンドウ上に配置します．まず，配置のレイアウトはフローレイアウトを指定しています．次に，テキストフィールドの説明となるラベルを生成して追加し，テキストフィールドを追加しています．さらに2つのボタンとテキストエリアも追加します．最後に，2つのボタンに対してthis(つまりReservationTestMainFrame)をActionListenerとして追加しています．
以上のようなコンポーネントの定義，レイアウトマネージャ，イベントリスナーを使ったJavaプログラミングについてはプログラミング4で学びましたので，必要に応じて復習しておきましょう．

reservation_testクラス

ReservationTestMainFrameのレイアウトマネージャをフローレイアウトにしたので，reservation_testクラスのmainメソッドの中で，表示されるウィンドウのサイズを調整して，ある程度見栄えを整えます．赤の部分を修正しました．

```
public class resevation_test {

    public static void main ( String argv[]){
        ReservationTestControl reservationControl = new ReservationTestControl();
        ReservationTestMainFrame mainFrame = new ReservationTestMainFrame(reservationControl);
        mainFrame.setBounds( 5,5,655,455);
        mainFrame.setVisible(true);
    }
}
```

実行例

version 2の実行例を以下に示します．この段階ではボタンをクリックしても何も起きません．右上の×で閉じることだけができます．

例題プログラム version 3

version 3 でMySQLへの操作と結果の表示を作成して完成します．reservation_testクラスはversion 2と同じです．

ReservationTestControlクラス

いよいよ，ReservationTestControlクラスにコードを追加します．以下のようになります．

```
import java.sql.*;

public class ReservationTestControl {
    String userid = "puser";    //ユーザID
    String password = "1234";   //パスワード
    Connection sqlCon;          //MySQLへの接続
    Statement sqlStmt;          //SQLコマンド実行のためのインスタンス

    public ReservationTestControl(){

    }
    public String insertReservation( String student_id, String facility_name, String y, String m, String d ){
        String res ="";
        // 年月日が数字かどうかををチェック
        try {
            int ryear = Integer.parseInt( y);
            int rmonth = Integer.parseInt( m);
            int rday = Integer.parseInt( d);
        } catch(NumberFormatException e){
            res ="年月日には数字を指定してください";
            return res;
        }
        // 月と日が一桁だったら，前に0をつける処理
        if (m.length()==1) {
            m = "0" + m;
        }
        if ( d.length()==1){
            d = "0" + d;
        }
        String date = y + "-" + m + "-" + d;

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

        //(2) MySQLの操作(INSERT文の実行)
        try {
              // 予約情報を挿入するクエリ
              String sql = "INSERT INTO practice.reservation ( student_id, facility_name, date ) VALUES ('";
              sql += student_id + "','" + facility_name + "','" + date + "');";
              // クエリーを実行して結果セットを取得

              int rs_int = sqlStmt.executeUpdate(sql);
              res = rs_int+ "行登録しました";
        } catch (Exception e) {
            res = "エラーが生じたため，登録できませんでした";
        }

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

        return res;
    }
    public String selectReservation(){
        String res = "";

        //(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 practice.reservation;";
              // クエリーを実行して結果セットを取得
              ResultSet rs = sqlStmt.executeQuery(sql);
              // 検索結果からレコードを１つずつ取り出し，3つのカラムの値を表示
              while(rs.next()){
                  String student = rs.getString("student_id");
                  String facility_name = rs.getString("facility_name");
                  String date = rs.getString("date");
                  res += date + "  " + student + "  " + facility_name +"¥n" ;
              }
        } catch (Exception e) {
            res = "データ検索においてエラーが生じました";
        }
```

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

    return res;
}

/////  MySQLを操作する準備を行うメソッド
private void connectDB(){
    try{
        // ドライバクラスをロード
        Class.forName("org.gjt.mm.mysql.Driver"); // MySQLの指定

        // データベースへのURLを作成
        String url = "jdbc:mysql://localhost?useUnicode=true&characterEncoding=SJIS";
        // ユーザIDとパスワードを設定して接続
        sqlCon = DriverManager.getConnection(url,userid, 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();
    }
}

////// MySQLへの接続を解除するメソッド
private void closeDB(){
    try{
        sqlStmt.close();
        sqlCon.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}
```

黄緑の四角の部分は第7回授業で作成した例題プログラムと同じコードなので，ここでは説明をしません．
insertReservationメソッドは，引数で指定された値を１つの予約情報のレコードとしてテーブルreservationに追加します．赤い四角の部分は，年月日が整数になっているかをチェックし，月と日を2桁のしています．整数かどうかのチェックは，Integer.parseIntを実行して例外が生じなければ整数であり，例外が生じた場合は整数ではないので，そうしたメッセージを返すようにしています．実際には，年月日としての値の適切さも調べる方がよいのですが，この例題プログラムでは，整数かどうかの確認だけにしています．
selectReservationメソッドは，テーブルreservationのすべてのレコードを1レコード1行となるような文字列として返します．第7回の例題プログラムとほぼ同じです．ただし，赤線の部分は第7回の例題プログラムではコンソールに出力していましたが，このプログラムでは表示されるはずの文字列を改行区切りでつないで，１つの文字列変数に入れています．

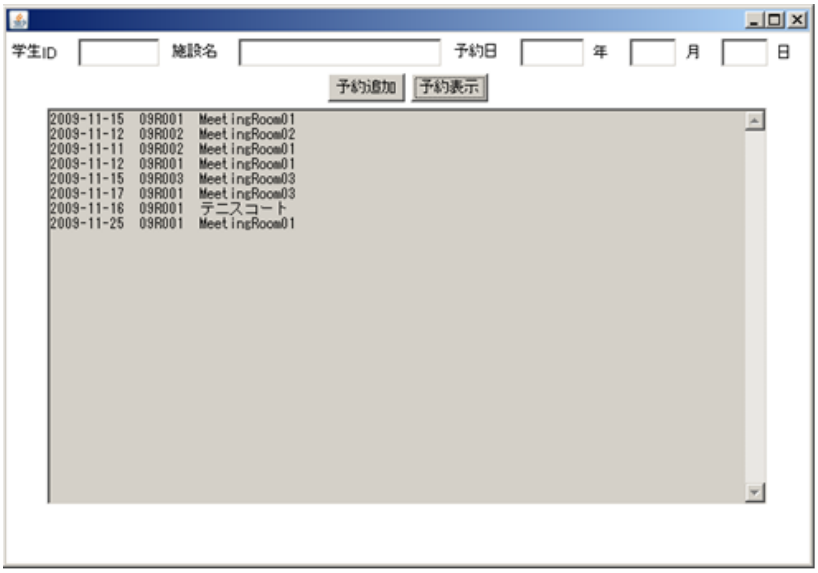
ReservationTestMainFrameクラス

自動生成されたactionPerformedというメソッドの中に，「予約追加」ボタンがクリックされたとき，「予約表示」ボタンがクリックされたときの処理を追加します．それぞれ，コントローラrControlに対してinsertReservation，selectReservationを呼び出しています．どちらの処理でも結果がresultに返されるので，resultをテキストエリアに設定することで表示します．

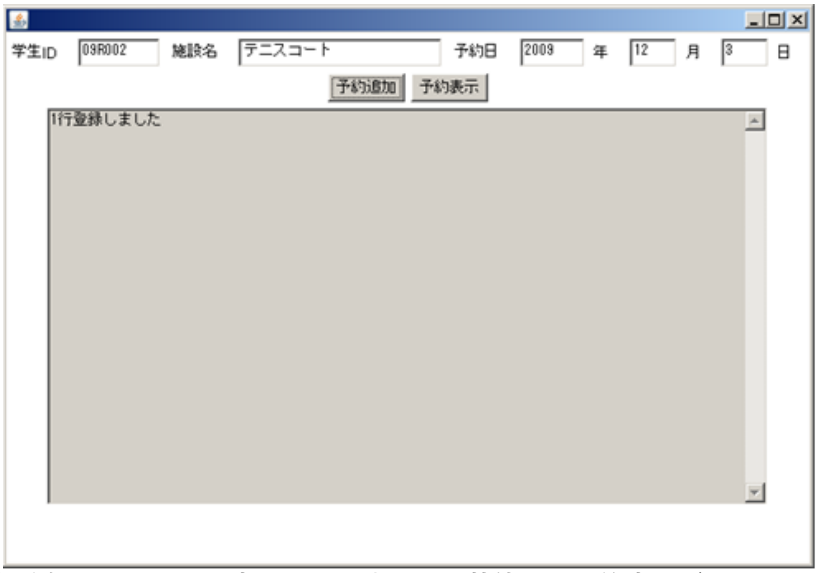
```
@Override
public void actionPerformed(ActionEvent e) {
    //結果を受け取る変数とメッセージエリアの初期化
    String result = "";
    textMessage.setText("");

    // ボタンごとの処理
    if ( e.getSource()==buttonInsert){///予約追加ボタンが押された時の処理
        result = rControl.insertReservation( tfStudentID.getText(), tfFacilityName.getText(), tfYear.getText(), tfMonth.getText(), tfDay.getText());
    } else if ( e.getSource() == buttonSelect){///予約表示ボタンが押された時の処理
        result = rControl.selectReservation();
    }

    //結果をメッセージエリアに設定
    textMessage.setText(result);
}
```

実行例

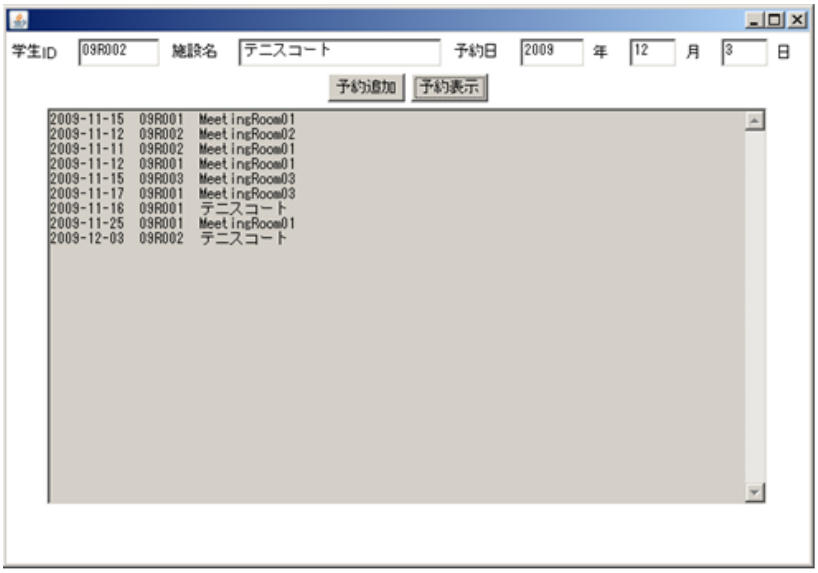
以下は，version 3で予約表示ボタンをクリックした際の実行例です．version 3を実行する際には，MySQLを起動していないと正しく動作しませんので，注意してください．



この状態で，以下のように学生ID，施設名，年月日を入力して予約追加ボタンをクリックしました.



すると，実行結果として「1行登録しました」と表示されます．その状態で，予約表示ボタンをクリックすると，以下のような画面になります.



プログラムリスト

version 3のプログラムリストをまとめて示しておきます.

ReservationTestControlクラス

コントローラとなるクラスです.

```
import java.sql.*;


public class ReservationTestControl {

        String userid = "puser";        //ユーザID
        String password = "1234";       //パスワード
        Connection sqlCon;                      //MySQLへの接続
        Statement sqlStmt;                      //SQLコマンド実行のためのインスタンス


        public ReservationTestControl(){

        }

        public String insertReservation( String student_id, String facility_name,
String y, String m, String d ){

                String res ="";

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

                // 月と日が一桁だったら，前に0をつける処理
                if (m.length()==1) {
                        m = "0" + m;
                }
                if ( d.length()==1){
                        d = "0" + d;
                }
                String date = y + "-" + m + "-" + d;


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

                //(2) MySQLの操作(INSERT文の実行)
                try {
                      // 予約情報を挿入するクエリ
                      String sql = "INSERT INTO practice.reservation (
student_id, facility_name, date ) VALUES ('";
                      sql += student_id + "','" + facility_name + "','" + date +
"');";
                      // クエリーを実行して結果セットを取得

                      int rs_int = sqlStmt.executeUpdate(sql);
                      res = rs_int+ "行登録しました";
                } catch (Exception e) {
                        res = "エラーが生じたため，登録できませんでした";
                }

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

                return res;
        }
```

```
        public String selectReservation(){

                String res = "";

                //(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 practice.reservation;";
                      // クエリーを実行して結果セットを取得
                      ResultSet rs = sqlStmt.executeQuery(sql);
                      // 検索結果からレコードを１つずつ取り出し，3つのカラムの値を表示
                      while(rs.next()){
                          String student = rs.getString("student_id");
                          String facility_name = rs.getString("facility_name");
                          String date = rs.getString("date");
                          res += date + "   " + student + "   " + facility_name
+"\n" ;
                      }
                } catch (Exception e) {
                        res = "データ検索においてエラーが生じました";
                }

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

                return res;
        }



        /////  MySQLを操作する準備を行うメソッド
        private void connectDB(){
                  try{
                      // ドライバクラスをロード
                      Class.forName("org.gjt.mm.mysql.Driver"); // MySQLの指定

                      // データベースへのURLを作成
                      String url = "jdbc:mysql://localhost?
useUnicode=true&characterEncoding=SJIS";
                      // ユーザIDとパスワードを設定して接続
                      sqlCon = DriverManager.getConnection(url,userid, 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();
                    }
                }

        ////// MySQLへの接続を解除するメソッド
        private void closeDB(){
                try{
                        sqlStmt.close();
                        sqlCon.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
        }

}
```

ReservatoinTestMainFrameクラス

ビューとなるクラスです.

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


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

        ReservationTestControl rControl;
```

```
        TextField tfStudentID,tfFacilityName;   // 学生IDと施設のテキストフィールド
        TextField tfYear,tfMonth,tfDay;                 // 年月日のテキストフィールド

        Button buttonInsert;            // 新規予約ボタン
        Button buttonSelect;            // レコードの検ボタン

        TextArea textMessage;                                   // 検索結果・メッセージ欄


        public ReservationTestMainFrame( ReservationTestControl rc){
                rControl = rc;

                // 各フィールドの生成
                tfStudentID = new TextField("",6);
                tfFacilityName = new TextField("",20);
                tfYear = new TextField("",4);
                tfMonth = new TextField("",2);
                tfDay = new TextField("",2);

                // ボタンの生成
                buttonInsert = new Button("予約追加");
                buttonSelect = new Button("予約表示");

                // テキストエリアの生成と編集不可の設定
                textMessage = new TextArea( 20, 80);
                textMessage.setEditable(false);

                // ウィンドウへの登録
                setLayout( new FlowLayout());
                add( new Label("学生ID"));
                add( tfStudentID);
                add( new Label(" 施設名"));
                add( tfFacilityName);
                add( new Label(" 予約日"));
                add( tfYear);
                add( new Label("年"));
                add( tfMonth);
                add( new Label("月"));
                add( tfDay);
                add( new Label("日"));

                add( buttonInsert);
                add( buttonSelect);

                add(textMessage);

                buttonInsert.addActionListener(this);
                buttonSelect.addActionListener(this);

                addWindowListener(this);
                addKeyListener(this);

        }

        @Override
        public void actionPerformed(ActionEvent e) {

                //結果を受け取る変数とメッセージエリアの初期化
                String result = "";
                textMessage.setText("");

                // ボタンごとの処理
                if ( e.getSource()==buttonInsert){///予約追加ボタンが押された時の処理
                        result = rControl.insertReservation(
tfStudentID.getText(), tfFacilityName.getText(), tfYear.getText(),
tfMonth.getText(), tfDay.getText());
                } else if ( e.getSource() == buttonSelect){///予約表示ボタンが押された時の処理
                        result = rControl.selectReservation();
                }

                //結果をメッセージエリアに設定
                textMessage.setText(result);

        }

        @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
                System.exit(0);
        }

        @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

        }

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

        }

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

        }

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

        }

}
```

reservation_testクラス

mainメソッドを実装するクラスです.

```
public class resevation_test {

        public static void main ( String argv[]){
                ReservationTestControl reservationControl = new
ReservationTestControl();
                ReservationTestMainFrame mainFrame = new
ReservationTestMainFrame(reservationControl);
                mainFrame.setBounds( 5,5,655,455);
                mainFrame.setVisible(true);
        }
}
```

まとめ

このモジュールでは以下のことを学び，MySQLを操作するGUIプログラムを作成しました.

- MVCアーキテクチャ
- Javaによるウィンドウを表示させるアプリケーションの作成方法
- Frameクラスを拡張してテキストフィールド，ボタン等を配置したインターフェースの作成方法
- Frameクラスを拡張したクラスで，ボタンをクリックしたときの動作を記述する方法
