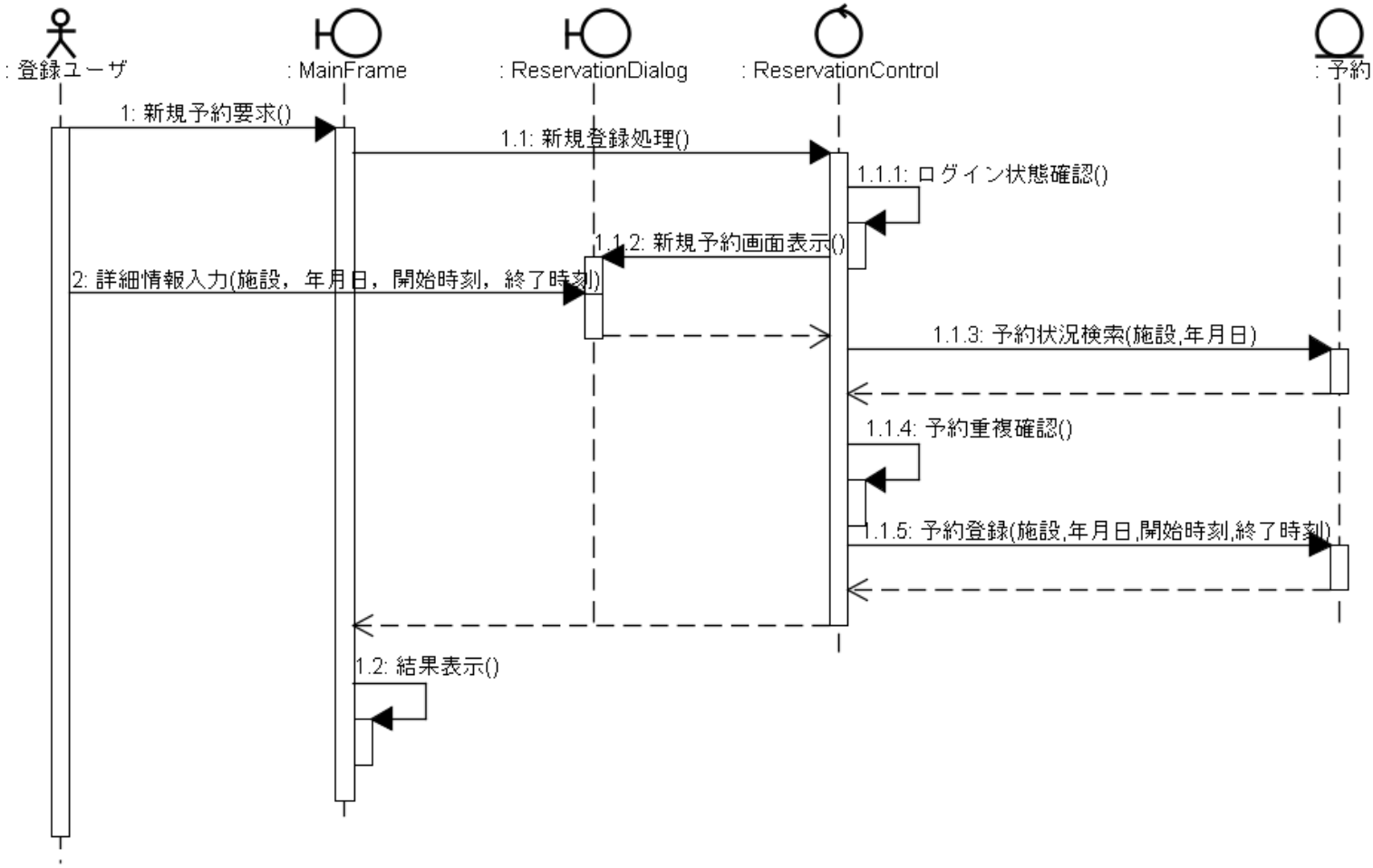
イントロダクション

利用者用サブシステムのバージョン3を作成し，完成させます．今回のバージョンでは，新規の予約を登録する機能を実装します．余力がある学生は，自分の予約をキャンセルする機能を実装しましょう．

新規予約機能の実装方針

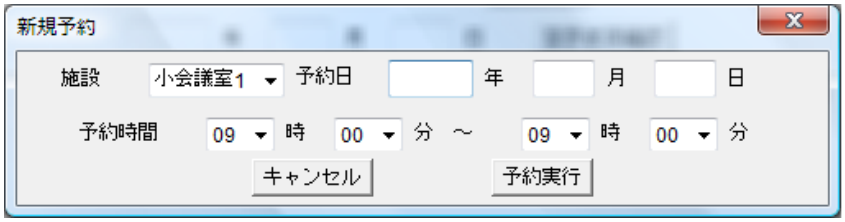
新規予約機能のシーケンス図は以下のようになっていました.



ユーザがMainFrameの新規予約ボタンをクリックすると，MainFrameはReservationControlに予約の新規登録処理を依頼します．ReservationControlはReservationDialogを生成して表示し，新規予約の詳細な情報を受け付けます．ユーザが予約に必要な情報(施設，年月日，開始時刻，終了時刻)を入力すると，予約テーブルから施設と年月日でレコードを検索します．その結果を調べて既存の予約と重複しないかを確認します．シーケンス図には明示していませんが，「3ヶ月先まで予約」という期間の条件を満たしているかどうかの確認もします．それらの結果，条件を満たしている場合は，新規予約テーブルに新規の予約レコードを追加します．重複している場合は，その旨のメッセージを結果とします．
予約情報を入力するためのReservationDialogの実装，既存の予約レコードとの重複確認処理，期間の条件の確認がポイントになります．

ReservationDialogの実装

以下にReservationDialogのウィンドウを示します．年月日は入力しますが，施設と時刻は選択できるようにします．



予約期間の条件の確認

予約期間は，2日後から3ヶ月先となっています．これらを確認するためにCalendarクラスを使って日付のインスタンスを生

成して比較することで確認を行うこととします．

既存の予約レコードとの重複確認

2つの時間帯が重なるパターンは以下の4パターンあります．このような場合は同じ施設を予約することはできません．

この図において，上に書かれている青の帯が既存の予約の時間帯，下の赤の帯が新規の時間帯だとします．左側の2パターンは，新規予約の開始時刻が既存の予約の開始時刻と終了時刻の間にある場合であることがわかります．一方，右側の2パターンは，既存の予約の開始時刻が新規予約の開始時刻と終了時刻の間にある場合です．したがって，以下のいずれか一方でも満たせば時間帯が重なっていることになります．

- 既存の開始時刻＜新規の開始時刻 かつ 新規の開始時刻＜既存の終了時刻
- 新規の開始時刻＜既存の開始時刻 かつ 既存の開始時刻＜新規の終了時刻

時分選択ボックスの実装

予約の開始時刻と終了時刻の範囲は，施設によって決まっています．早いものでも9時開始ですし，遅くとも21時までです．したがって，その範囲のみ指定できれる方が望ましいと言えます．また，分は30分刻みで指定することになっていますので，分も00分と30分のみが指定できる方がよいでしょう．そこで，時刻の入力のために，時と分を指定するための選択ボックスを，Choiceクラスを継承して作成しておきます．

時刻の選択ボックス

時刻を指定するための選択ボックスChoiceHourのソースリストを以下に示します．

```
package client_system;

import java.awt.*;

public class ChoiceHour extends Choice{

    ChoiceHour(){
        //時刻の範囲の初期値は 9時～21時とする
        resetRange(9,21);
    }

    // 指定できる時刻の範囲を設定するメソッド
    public void resetRange( int start, int end){
        // 範囲に含まれる時刻の数を求める
        int number = end - start +1;

        // 選択ボックスの内容をクリアする
        removeAll();

        // 指定できる範囲の時刻を設定する
        while (start<=end){
            String h = String.valueOf(start);
            //一桁の場合，前に0を付ける
            if ( h.length()==1){
                h = "0" + h;
            }
            // 選択ボックスに追加 (こちらは文字列)
            add(h);
            // startを1増やす
            start++;
        }
    }

    // 最後に設定されている時刻を返す
    public String getLast(){
        return getItem( getItemCount()-1 );
    }
}
```

Choiceを継承して定義しています．メソッドとして，時刻の範囲を設定するresetRangeと最後の時刻を返すgetLastがあります．コンストラクタではresetRangeを使って範囲の初期値を設定しています．resetRangeは整数型の開始時刻(start)と終了時刻(end)の2つの引数が持ちます．まず，startとendか

ら範囲の数をnumberに求め，選択ボックスの内容をクリアしています．その後，while文でstartを1ずつ増やしながら，選択ボックスに追加(add)しています．その際に時刻が一桁の場合は前に0を付ける処理を行っています． getLastは，項目の数から1を引いた場所にある項目を返しています．これによって，最後の項目(時刻)が文字列で返ることになります．これは，時刻の範囲の最初の値だけを変更する際に，resetRangeを用いると，最後の値ももう一度指定する必要があるので，現在の最後の値を取り出すために用いる．

分の選択ボックス

分を指定するための選択ボックスChoiceMinuteのソースリストを以下に示します．

```
package client_system;

import java.awt.*;

public class ChoiceMinute extends Choice{

    public ChoiceMinute(){
        // 00分と30分のみを登録
        add("00");
        add("30");
    }

    //選択されている分を整数で返す
    public int getMinute(int index){
        if ( index==0){
            return 0;
        } else {
            return 30;
        }
    }
}
```

これもChoiceを継承して定義しています．また，コンストラクタでは00分と30分のみを登録しています．メソッドは現在選択されている分の値を整数で返すgetMinuteのみです．

ReservationDialogの実装

ReservationDialogという名称で新規クラスを作成して，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，WindouListener，ItemListenerを指定し，ボタンをクリックしたとき，ウィンドウを閉じるとき，および選択ボックスの項目が変更された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．次にclass ReservationDialog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，第13回のLoginDialogクラスを作成したときとほぼ同じですが，今回は，ItemListenerが追加されています．

ReservationDialogの変数

ReservationDialogでは以下の変数を定義しておきます．

```
    boolean canceled;        //キャンセルされたら true 予約実行ボタンのときは false

    // パネル
    Panel panelNorth;
    Panel panelMid;
    Panel panelSouth;

    // 入力用コンポーネント
    ChoiceFacility choiceFacility; // 施設選択用ボックス
    TextField tfYear,tfMonth,tfDay; // 年月日のテキストフィールド
    ChoiceHour startHour;          // 予約開始時刻 時の選択用ボックス
    ChoiceMinute startMinute;      // 予約開始時刻 分の選択用ボックス
    ChoiceHour endHour;            // 予約終了時刻 時の選択用ボックス
    ChoiceMinute endMinute;        // 予約終了時刻 分の選択用ボックス

    // ボタン
    Button buttonOK;
    Button buttonCancel;
```

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，ユーザが新規登録を実行したのか，操作をキャンセルしたのかがわかるようにします．考え方はLoginDialogと同じです．

コンポーネントを上部，中央，下部に分けて配置するために3つのパネルを用意しています．これもLoginDialogと同じです．入力用のコンポーネントは，施設選択用ボックスと，年月日のテキストフィールド，開始時刻と終了時刻を指定するための選択ボックスがあります．

ReservationDialogのコンストラクタ

以下にReservationDialogのコンストラクタを示します．

```
    public ReservationDialog(Frame owner) {
```

```
    //基底クラス(Dialog)のコンストラクタを呼び出す
    super(owner, "新規予約", true);

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

    //施設選択ボックスの生成
    choiceFacility = new ChoiceFacility();
    //テキストフィールドの生成｜年月日
    tfYear = new TextField("",4);
    tfMonth = new TextField("",2);
    tfDay = new TextField("",2);
    //開始時刻｜時分選択ボックスの生成
    startHour = new ChoiceHour();
    startMinute = new ChoiceMinute();
    //終了時刻｜自分選択ボックスの生成
    endHour = new ChoiceHour();
    endMinute = new ChoiceMinute();

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

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

    //上部パネルに施設選択ボックス，年月日入力欄を追加
    panelNorth.add( new Label("施設｜"));
    panelNorth.add(choiceFacility);
    panelNorth.add( new Label("予約日 "));
    panelNorth.add(tfYear);
    panelNorth.add(new Label("年"));
    panelNorth.add(tfMonth);
    panelNorth.add(new Label("月"));
    panelNorth.add(tfDay);
    panelNorth.add(new Label("日｜"));

    //中央パネルに予約｜開始時刻，終了時刻入力用選択ボックスを追加
    panelMid.add( new Label("予約時間｜"));
    panelMid.add( startHour);
    panelMid.add( new Label("時"));
    panelMid.add( startMinute);
    panelMid.add( new Label("分｜～｜"));
    panelMid.add( endHour);
    panelMid.add( new Label("時"));
    panelMid.add( endMinute);
    panelMid.add( new Label("分"));

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

    // ReservationDialog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);
```

```
    // ウイ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);
    //施設選択ボックス，時・分選択ボックスそれぞれに項目リスナを追加
    choiceFacility.addItemListener(this);
    startHour.addItemListener(this);
    startMinute.addItemListener(this);
    endHour.addItemListener(this);
    endMinute.addItemListener(this);

    // 選択されている施設によって，時刻の範囲を設定する．
    resetTimeRange();

    // 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, 500, 120);
    setResizable(false);
}
```

注釈を読むことで，どのようなことをやっているかがわかると思います．多くは，これまでの画面を扱うクラスのコンストラクタと同じです．それらと比較すると，今回は以下の点がポイントとなります．

- addItemListenerで，5つの選択ボックスにthis(つまりReservationDialog)を，項目が変更されたイベントのリスナとして登録しています．
- メソッドresetTimeRange()を呼び出して，予約の開始時刻と終了時刻の指定できる範囲を設定しています．

予約の開始時刻と終了時刻を設定するメソッド

コンストラクタで使用されているresetTimeRangeのソースリストを以下に示します．

```
private void resetTimeRange() {
    // 選択されている施設によって，時刻の範囲を設定する．
    if ( choiceFacility.getSelectedIndex()==0){
        // 最初の施設(小ホールのとき)の設定
        startHour.resetRange(10, 20);
        endHour.resetRange(10, 21);
    } else {
        // 小ホール以外の設定
        startHour.resetRange(9, 19);
        endHour.resetRange(9, 20);
    }
}
```

施設の選択ボックスchoiceFacilityの先頭には「小ホール」が設定されていますから，choiceFacilityで先頭の項目が設定されている場合(getSelectedIndex()の値が0のとき)は，開始時刻の指定範囲を10時から20時，終了時刻の指定範囲を10時から21時に設定します．それ以外の場合は，開始時刻の指定範囲を9時から19時，終了時刻の指定範囲を9時から20時に設定しています．

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){
        setVisible(false);
        dispose();
    } else if ( arg0.getSource() == buttonOK){
        canceled = false;
        setVisible(false);
        dispose();
    }
}
```

これはLoginDialogのときと同じです．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．これもLoginDialogと同じです．

```
@Override
public void windowClosing(WindowEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ
    setVisible(false);
    dispose();
}
```

ReservationDialogでは，これらに加えてitemStateChangedにも以下のようなコードを追加します．

```
@Override
public void itemStateChanged(ItemEvent arg0) {
    // TODO 自動生成されたメソッド・スタブ
    if ( arg0.getSource()==choiceFacility){
        // 施設が変更されたら，施設に応じた範囲を設定
        resetTimeRange();
    } else if ( arg0.getSource()==startHour){
        //開始時刻が変更されたら，終了時刻入力欄の時を開始時刻に合わせる
        int start = Integer.parseInt( startHour.getSelectedItem());
        endHour.resetRange(start, Integer.parseInt( endHour.getLast()));
    } else if ( arg0.getSource()==endHour){
        //終了時刻が変更され，最後の時刻の場合，分は 00分に設定
        if ( endHour.getSelectedIndex()==endHour.getItemCount()-1){
            endMinute.select(0);
        }
    } if( arg0.getSource()==endMinute){
        //終了時刻(分)が変更され，時が最後の場合，分は 00分に設定
        if ( endHour.getSelectedIndex()==endHour.getItemCount()-1){
            endMinute.select(0);
        }
    }
}
```

最初のif文では選択されている項目が変更されたコンポーネントがchoiceFacility(施設選択ボックス)であった場合，resetTimeRangeを呼び出して，施設に応じた時刻の範囲に設定します．2つめのif文は開始時刻の時が変更された場合に，終了時刻の範囲を開始時刻から，施設の閉館時刻に設定しています．3つめのif文では，終了時刻の時(endHour)が閉館時刻に設定された場合に，分(endMinute)の方は30分にはならないはずなので，selectメソッドを呼び出して00分に設定しています．selectの引数0は0番目の項目，つまり先頭の項目を意味します．4つめのif文も同様で，逆に終了時刻の分(endMinute)が変更された場合に，終了時刻の時(endHour)を調べて閉館時刻なら，

分(endMinute)を00分に設定しています．以上のような処理で，開館時間以外の時間帯が設定されないようにしています．

ReservationControlにおける実装

予約日の期間の条件をチェックするメソッド

checkReservationDateは，新規予約をする際に，予約日が2日後から3ヶ月先の間になっているかをチェックするメソッドです．

```
//// 予約日が2日後～3ヶ月先の条件に入っているか確認するメソッド
///| 期間の条件を満たしていたら true 入っていなかったら false を返す
private boolean checkReservationDate( int y, int m, int d){
    // 予約日
    Calendar dateR = Calendar.getInstance();
    dateR.set( y, m-1, d); // 月から1引かなければならないことに注意！

    // 今日の1日後
    Calendar date1 = Calendar.getInstance();
    date1.add(Calendar.DATE, 1);

    // 今日の3ヶ月後(90日後)
    Calendar date2 = Calendar.getInstance();
    date2.add(Calendar.DATE, 90);

    if ( dateR.after(date1) && dateR.before(date2)){
        return true;
    }
    return false;
}
```

まず， Calendarクラスのインスタンスを dateRに引数の年月日を設定します．日付を設定する際に，月は1引いておく仕様になっていますので，注意が必要です．次に1日後をdate1に設定します．Calendarのインスタンスを生成すると，デフォールトで今日の日付が設定されています．そこにaddメソッドで日数を加算します．Calendar.DATEは加算するのが日であることを示しており，第2引数で加算する値を指定しています．同様にして，3ヶ月後（90日後）をdate2に設定します．最後にif文ではafterとbeforeメソッドを使って，dateRがdate1とdate2の間にある場合は，trueを返し，そうでない場合はfalseを返します．

新規予約の処理をするメソッド

新規予約の処理を行う以下のようなメソッドを追加します．makeReservationメソッドはMainFrameを引数として，結果を文字列で返します．

```
////// 新規予約の登録
public String makeReservation(MainFrame frame){

    String res="";      //結果を入れる変数

    if ( flagLogin){ // ログインしていた場合
        //新規予約画面作成
        ReservationDialog rd = new ReservationDialog(frame);

        // 新規予約画面の予約日に，メイン画面に設定されている年月日を設定する
        rd.tfYear.setText(frame.tfYear.getText());
```

```
        rd.tfMonth.setText(frame.tfMonth.getText());
        rd.tfDay.setText(frame.tfDay.getText());

        // 新規予約画面を可視化
        rd.setVisible(true);
        if ( rd.canceled){
            return res;
        }
        try {
            //新規予約画面から年月日を取得
            String ryear_str = rd.tfYear.getText();
            String rmonth_str = rd.tfMonth.getText();
            String rday_str = rd.tfDay.getText();

            // 年月日が数字かどうかををチェックする処理
            int ryear = Integer.parseInt( ryear_str);
            int rmonth = Integer.parseInt( rmonth_str);
            int rday = Integer.parseInt( rday_str);

            if ( checkReservationDate( ryear, rmonth, rday)){  // 期間の条件を満たしている場合
                // 新規予約画面から施設名，開始時刻，終了時刻を取得
                String facility = rd.choiceFacility.getSelectedItem();
                String st = rd.startHour.getSelectedItem()+":" + rd.startMinute.getSelectedItem() +":00";
                String et = rd.endHour.getSelectedItem() + ":" + rd.endMinute.getSelectedItem() +":00";

                if( st.equals(et)){    //開始時刻と終了時刻が等しい
                    res = "開始時刻と終了時刻が同じです";
                } else {
                    //(1) MySQLを使用する準備
                    connectDB();

                    try {
                        // 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;
                        }
                        //(2) MySQLの操作(SELECT文の実行)
                        String rdate = ryear_str + "-" + rmonth_str + "-" + rday_str;
                          // 指定した施設の指定した予約日の予約情報を取得するクエリ
                          String sql = "SELECT * FROM db_reservation.reservation WHERE facility_name ='" + facility +
                          "' AND date = '" + rdate + "' ;";
                          // クエリーを実行して結果のセットを取得
                          ResultSet rs = sqlStmt.executeQuery(sql);
                          // 検索結果に対して重なりチェックの処理
                          boolean ng = false;  //重なりチェックの結果の初期値(重なっていない=false)を設定
                          // 取得したレコード一つ一つに対して確認
                          while(rs.next()){
                              //レコードの開始時刻と終了時刻をそれぞれstartとendに設定
                              String start = rs.getString("start_time");
                              String end = rs.getString("end_time");

                              if ( (start.compareTo(st)<0 && st.compareTo(end)<0) ||   //レコードの開始時刻< 新規の開始時刻 | AND | 新規の開始時刻<レコードの終了時刻
                                   (st.compareTo(start)<0 && start.compareTo(et)<0)){   //新規の開始時刻<レコードの開始時刻 | AND | レコードの開始時刻< 新規の終了時刻
```

```
                        // 重複有りの場合に ng をtrueに設定
                        ng = true; break;
                    }
                }
                /// 重なりチェックの処理！ここまで ///////

                if (!ng){ //重なっていない場合
                    //(2) MySQLの操作(INSERT文の実行)
                    sql = "INSERT INTO db_reservation.reservation (date,start_time,end_time,user_id,facility_name) VALUES ( '"
                        + rdate +"', '"  + st +"','" + et + "','" + reservation_userid +"','" + facility +"');";
                    int rs_int = sqlStmt.executeUpdate(sql);
                    res ="予約されました";
                } else { //重なっていた場合
                    res = "既にある予約に重なっています";
                }
            }catch (Exception e) {
                e.printStackTrace();
            }
            //(3) MySQLへの接続切断
            closeDB();
        }
    } else {
        res = "予約日が無効です．";
    }
} catch(NumberFormatException e){
    res ="予約日には数字を指定してください";
}
} else { // ログインしていない場合
    res = "ログインしてください";
}
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，つまりログイン状態のときには，新規予約の処理を行います．一方，falseのときは「ログインしてください」という文字列を結果として設定します．
ログイン状態の時は，まず，ReservationDialogを生成します．次に生成したReservationDialogの年月日のテキストフィールドに，MainFrameの年月日の値をそれぞれ設定します．これは，ManFrameで年月日が入力されている場合，ある日の空き状況を確認し，その後，予約を取ろうとしていることが考えられるので，自動的にその日が入るようにしている処理です．その上で，ReservationDialogを可視化します． これによって，ReservationDialogへの操作が可能になります．
ReservationDialogへの操作が終了後，canceledがtrueの場合(キャンセルされた場合)は，空文字列を結果として返して終了します．次にReservationDialogに設定された年月日を取り出し，整数かどうかを確認します．その後，checkReservationDateメソッドを呼び出して2日後から3ヶ月先の条件を満たしているかどうかを確認します．満たしていない場合は，「予約日が無効です」という文字列を結果に設定します．条件を満たしている場合は，ReservationDialogから施設名と予約の開始時刻，終了時刻を取り出し，開始時刻と終了時刻が同じになっていないかを調べます．同じ場合は，「開始時刻と終了時刻が同じです」という文字列を結果に設定して終了します．
以上の確認をパスしたら，MySQLの操作で，指定された施設の指定された予約日の予約レコードを検索します．検索の結果得た予約レコード一つ一つに対して時間帯が重なっていないかを確認します．その上で，重なっていない場合は，新規に予約レコードを挿入します．

MainFrameにおける実装

MainFrameのActionPerformedに，buttonReservationがクリックされたときに，reservationControlのmakeReservationメソッドを呼び出すコードを追加します．loginLogoutメソッドのときと同様なので，具体的には示しません．

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．

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.

ChoiceHourクラス

```
package client_system;

import java.awt.*;

public class ChoiceHour extends Choice{

        ChoiceHour(){
                //時刻の範囲の初期値は 9時～21時とする
                resetRange(9,21);
        }

        // 指定できる時刻の範囲を設定するメソッド
        public void resetRange( int start, int end){
                // 範囲に含まれる時刻の数を求める
                int number = end - start +1;

                // 選択ボックスの内容をクリアする
                removeAll();

                // 指定できる範囲の時刻を設定する
                while (start<=end){
                        String h = String.valueOf(start);
                        //一桁の場合, 前に0を付ける
                        if ( h.length()==1){
                                h = "0" + h;
                        }
                        // 選択ボックスに追加 (こちらは文字列)
                        add(h);
                        // startを1増やす
                        start++;
                }
        }

        // 最後に設定されている時刻を返す
        public String getLast(){
                return getItem( getItemCount()-1 );
        }
}
```

ChoiceMinuteクラス

```
package client_system;

import java.awt.*;

public class ChoiceMinute extends Choice{

        public ChoiceMinute(){
                // 00分と30分のみを登録
                add("00");
                add("30");
        }

        //選択されている分を整数で返す
        public int getMinute(int index){
                if ( index==0){
                        return 0;
                } else {
                        return 30;
                }
        }
```

```
}
```

ReservationDialogクラス

```
package client_system;

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

public class ReservationDialog extends Dialog implements ActionListener,
WindowListener,ItemListener{

        boolean canceled;               //キャンセルされたら true 予約実行ボタンのときは
false

        // パネル
        Panel panelNorth;
        Panel panelMid;
        Panel panelSouth;

        // 入力用コンポーネント
        ChoiceFacility choiceFacility;  // 施設選択用ボックス
        TextField tfYear,tfMonth,tfDay; // 年月日のテキストフィールド
        ChoiceHour startHour;                   // 予約開始時刻 時の選択用ボックス
        ChoiceMinute startMinute;               // 予約開始時刻 分の選択用ボックス
        ChoiceHour endHour;                             // 予約終了時刻　時の選択用
ボックス
        ChoiceMinute endMinute;                 // 予約終了時刻 分の選択用ボックス

        // ボタン
        Button buttonOK;
        Button buttonCancel;

        public ReservationDialog(Frame owner) {

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

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

                //施設選択ボックスの生成
                choiceFacility = new ChoiceFacility();
                //テキストフィールドの生成 年月日
                tfYear = new TextField("",4);
                tfMonth = new TextField("",2);
                tfDay = new TextField("",2);
                //開始時刻 時分選択ボックスの生成
                startHour = new ChoiceHour();
                startMinute = new ChoiceMinute();
                //終了時刻 自分選択ボックスの生成
                endHour = new ChoiceHour();
                endMinute = new ChoiceMinute();

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

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

                //上部パネルに施設選択ボックス，年月日入力欄を追加
                panelNorth.add( new Label("施設 "));
                panelNorth.add(choiceFacility);
                panelNorth.add( new Label("予約日 "));
                panelNorth.add(tfYear);
                panelNorth.add(new Label("年"));
                panelNorth.add(tfMonth);
                panelNorth.add(new Label("月"));
                panelNorth.add(tfDay);
                panelNorth.add(new Label("日 "));

                //中央パネルに予約 開始時刻，終了時刻入力用選択ボックスを追加
                panelMid.add( new Label("予約時間 "));
```

```
                panelMid.add( startHour);
                panelMid.add( new Label("時"));
                panelMid.add( startMinute);
                panelMid.add( new Label("分 ～ "));
                panelMid.add( endHour);
                panelMid.add( new Label("時"));
                panelMid.add( endMinute);
                panelMid.add( new Label("分"));

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

                // ReservationDialog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);
                //施設選択ボックス，時・分選択ボックスそれぞれに項目リスナを追加
                choiceFacility.addItemListener(this);
                startHour.addItemListener(this);
                startMinute.addItemListener(this);
                endHour.addItemListener(this);
                endMinute.addItemListener(this);

                // 選択されている施設によって，時刻の範囲を設定する.
                resetTimeRange();

                // 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, 500, 120);
                setResizable(false);
        }

        private void resetTimeRange() {
                // 選択されている施設によって，時刻の範囲を設定する.
                if ( choiceFacility.getSelectedIndex()==0){
                        // 最初の施設(小ホールのとき)の設定
                        startHour.resetRange(10, 20);
                        endHour.resetRange(10, 21);
                } else {
                        // 小ホール以外の設定
                        startHour.resetRange(9, 19);
                        endHour.resetRange(9, 20);
                }
        }


        @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){
                        setVisible(false);
                        dispose();
                } else if ( arg0.getSource() == buttonOK){
                        canceled = false;
                        setVisible(false);
                        dispose();
                }
        }

        @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
                setVisible(false);
                dispose();
        }
```

```
        @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 itemStateChanged(ItemEvent arg0) {
                // TODO 自動生成されたメソッド・スタブ
                if ( arg0.getSource()==choiceFacility){
                        // 施設が変更されたら，施設に応じた範囲を設定
                        resetTimeRange();
                } else if ( arg0.getSource()==startHour){
                        //開始時刻が変更されたら，終了時刻入力欄の時を開始時刻に合わせる
                        int start = Integer.parseInt(
startHour.getSelectedItem());
                        endHour.resetRange(start, Integer.parseInt(
endHour.getLast()));
                } else if ( arg0.getSource()==endHour){
                        //終了時刻が変更され，最後の時刻の場合，分は 00分に設定
                        if ( endHour.getSelectedIndex()==endHour.getItemCount()-
1){
                                endMinute.select(0);
                        }
                } if( arg0.getSource()==endMinute){
                        //終了時刻（分）が変更され，時が最後の場合，分は 00分に設定
                        if ( endHour.getSelectedIndex()==endHour.getItemCount()-
1){
                                endMinute.select(0);
                        }
                }
        }
}
```

7.4 ReservationControlクラス

```
package client_system;

import java.sql.*;
import java.util.*;
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) MySQ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) MySQへの接続切断
                        closeDB();
                }
                return res;
        }

        ////// 新規予約の登録
        public String makeReservation(MainFrame frame){

                String res="";          //結果を入れる変数

                if ( flagLogin){ // ログインしていた場合
                        //新規予約画面作成
                        ReservationDialog rd = new ReservationDialog(frame);

                        // 新規予約画面の予約日に，メイン画面に設定されている年月日を設定す
る
                        rd.tfYear.setText(frame.tfYear.getText());
                        rd.tfMonth.setText(frame.tfMonth.getText());
                        rd.tfDay.setText(frame.tfDay.getText());

                        // 新規予約画面を可視化
                        rd.setVisible(true);
                        if ( rd.canceled){
                                return res;
                        }
                        try {
```

```
                                    //新規予約画面から年月日を取得
                                    String ryear_str = rd.tfYear.getText();
                                    String rmonth_str = rd.tfMonth.getText();
                                    String rday_str = rd.tfDay.getText();

                                    // 年月日が数字かどうかををチェックする処理
                                    int ryear = Integer.parseInt( ryear_str);
                                    int rmonth = Integer.parseInt( rmonth_str);
                                    int rday = Integer.parseInt( rday_str);

                                    if ( checkReservationDate( ryear, rmonth, rday)){
// 期間の条件を満たしている場合
                                            // 新規予約画面から施設名，開始時刻，終了時刻を取
得
                                            String facility =
rd.choiceFacility.getSelectedItem();
                                            String st =
rd.startHour.getSelectedItem()+":" + rd.startMinute.getSelectedItem() +":00";
                                            String et = rd.endHour.getSelectedItem()
+ ":" + rd.endMinute.getSelectedItem() +":00";

                                            if( st.equals(et)){             //開始時刻
と終了時刻が等しい
                                                    res = "開始時刻と終了時刻が同じです";
                                            } else {
                                                    //(1) MySQLを使用する準備
                                                    connectDB();


                                                    try {
                                                      // 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;
                                                      }
                                                      //(2) MySQLの操作(SELECT文
の実行)
                                                      String rdate = ryear_str
+ "-" + rmonth_str + "-" + rday_str;
                                                    // 指定した施設の指定した予約日
の予約情報を取得するクエリ
                                                    String sql = "SELECT *
FROM db_reservation.reservation WHERE facility_name ='" + facility +
"' ;";
                                                    // クエリーを実行して結果のセッ
トを取得
                                                    ResultSet rs =
sqlStmt.executeQuery(sql);
                                                    // 検索結果に対して重なりチェッ
クの処理
                                                    boolean ng = false;
//重なりチェックの結果の初期値（重なっていない=false）を設定
                                                        // 取得したレコード一つ一つ
に対して確認
                                                    while(rs.next()){
                                                                    //レコード
の開始時刻と終了時刻をそれぞれstartとendに設定
                                                            String start =
rs.getString("start_time");
                                                            String end =
rs.getString("end_time");

                                                            if (
(start.compareTo(st)<0 && st.compareTo(end)<0) ||           //レコードの開始時刻＜新規
の開始時刻 AND 新規の開始時刻＜レコードの終了時刻
(st.compareTo(start)<0 && start.compareTo(et)<0)){          //新規の開始時刻＜レ
コードの開始時刻 AND レコードの開始時刻＜新規の開始時刻
// 重複有りの場合に ng をtrueに設定
                                                                    ng =
true; break;
                                                            }
                                                    }
                                                        /// 重なりチェックの処理
```

```
ここまで  ///////

                                                        if (!ng){ //重なっていない場合
                                                                //(2)
MySQLの操作(INSERT文の実行)
                                                            sql = "INSERT INTO
db_reservation.reservation (date,start_time,end_time,user_id,facility_name) VALUES
( '"
                                                                    + rdate +"',
'"  + st +"','" + et + "','" + reservation_userid +"','" + facility +"');";
                                                            int rs_int =
sqlStmt.executeUpdate(sql);
                                                            res ="予約されました";
                                                        } else {  //重なっていた場合
                                                            res = "既にある予約に重
なっています";
                                                        }
                                                    }catch (Exception e) {
                                                            e.printStackTrace();
                                                    }
                                                    //(3) MySQへの接続切断
                                                    closeDB();
                                            }
                                    } else {
                                            res = "予約日が無効です. ";
                                    }
                            } catch(NumberFormatException e){
                                    res ="予約日には数字を指定してください";
                            }
                    } else { // ログインしていない場合
                            res = "ログインしてください";
                    }
                    return res;
            }

            //// 予約日が2日後～3ヶ月先の条件に入っているか確認するメソッド
            /// 期間の条件を満たしていたら true 入っていなかったら false を返す
            private boolean checkReservationDate( int y, int m, int d){
                    // 予約日
                    Calendar dateR = Calendar.getInstance();
                    dateR.set( y, m-1, d);  // 月から1引かなければならないことに注意！

                    // 今日の１日後
                    Calendar date1 = Calendar.getInstance();
                    date1.add(Calendar.DATE, 1);

                    // 今日の３ヶ月後（90日後）
                    Calendar date2 = Calendar.getInstance();
                    date2.add(Calendar.DATE, 90);

                    if ( dateR.after(date1) && dateR.before(date2)){
                            return true;
                    }
                    return false;
            }
}
```

7.5 MainFrameクラスのActionPerformed

これまでの課題でBを実施している方は，以下のリストに加えて，他のボタンをクリックしたときの処理が追加されていると思います．

```
            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);
                    } else if ( arg0.getSource() == buttonReservation){     /// 新規予
約ボタン
                            result = reservationControl.makeReservation(this);
                    }
                    textMessage.setText(result);
            }
```