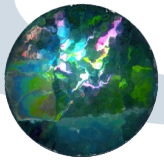
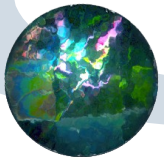
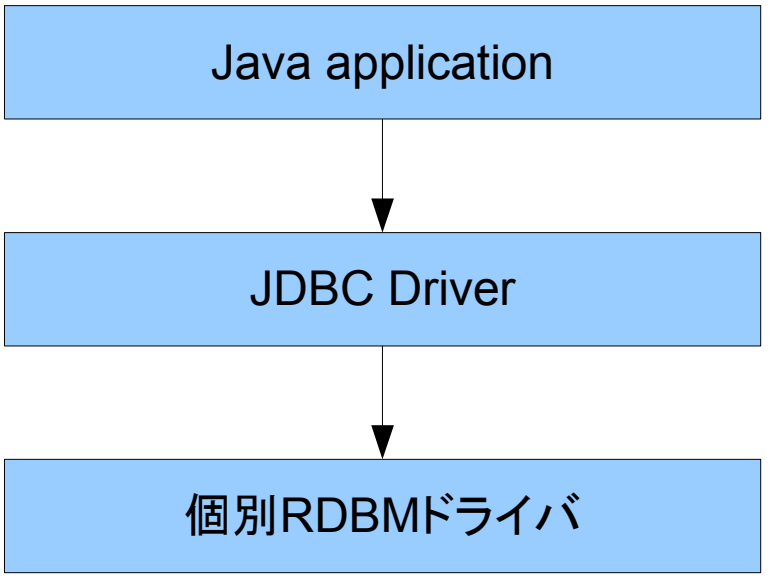


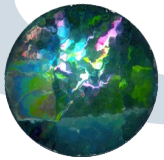
データベースへの接続



Javaとデータベース

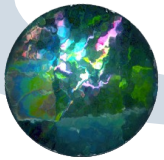
- 関係データベースには様々な種類がある
- RDBMごとにドライバが必要になる
- Javaからの共通のAPI
  - RDBM変更が容易
  - 保守性向上





DBへの接続

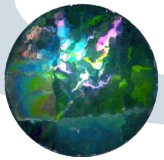
- DBを文字列として指定する
  - jdbc:postgresql://ホスト名:ポート/DB名
- ドライバのロード
  - Class.forName(ドライバ名)
- 接続
  - java.sql.DriverManager.getConnection(url,ユーザ、パスワード)



DBへの接続

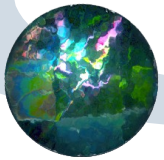
- Class.forName("org.postgresql.Driver");
  - ドライバインスタンスの生成
  - java.sql.DriverManagerへの登録
- java.sql.DriverManager.getConnection(url, owner,password);
  - DBへの接続

```
String url = "jdbc:postgresql://localhost:5432/dbname”;
Class.forName("org.postgresql.Driver");
java.sql.Connection con
        = java.sql.DriverManager.getConnection(url, owner,password);
```



注意

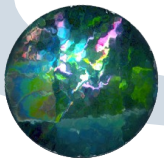
- 一旦接続できるとRDBMに依存しないAPIが使える
- 終了時には接続を切断すること
  - java.sql.Connection.close()
- Dbのエラーへの対応
  - 例外処理による対応
  - java.sql.SQLException



簡単なデータベースの例

ID	name	valid
1	Tom Brown	TRUE
2	Kim Snow	TRUE
3	Al Tompson	FALSE
4	Ann Kay	TRUE

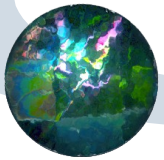
```
create table staffs(
   id       integer primary key,
   name text,
   valid   bool
);
```



検索

- 検索SQL文の作成
  - String sql=”select * from staffs”;
- java.sql.Statementの生成
- 検索結果はjava.sql.ResultSetに戻る

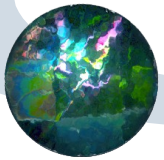
```
private ResultSet query(String sql) throws SQLException {
    ResultSet resultSet = null;
    Statement select = con.createStatement();
    resultSet = select.executeQuery(sql);
    return resultSet;
}
```



ResultSetからの読み出し

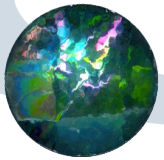
- 検索結果が複数あることに注意
  - ResultSet.next()で順に取り出し
- ResultSet.getクラス名(フィールド名)
- ResultSet.getクラス名(フィールド番号)
- 例

```
ResultSet r;
while(r.next()){
    Integer id=r.getInteger(“id”);
    String name=r.getString(“name”);
    Boolean valid=new Boolean(r.getBoolean(“valid”));
}
```

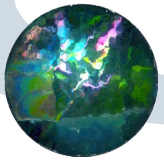


挿入、更新

- 挿入・更新SQL文の生成
  - String sql=”insert into staffs (id,name,valid) values ('5','Sam Crow','true')”;
  - String sql=”update staffs set valid='false' where id='5';

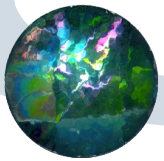


```
private int insert(String sql) throws SQLException {
    int count = 0;
    Statement stm;
    /* 自動コミットをオフにする */
    con.setAutoCommit(false);
    stm = con.createStatement();
    count = stm.executeUpdate(sql);
    if (count != 0) {
        con.commit();
    } else {
        con.rollback();
    }
    return count;
}
```



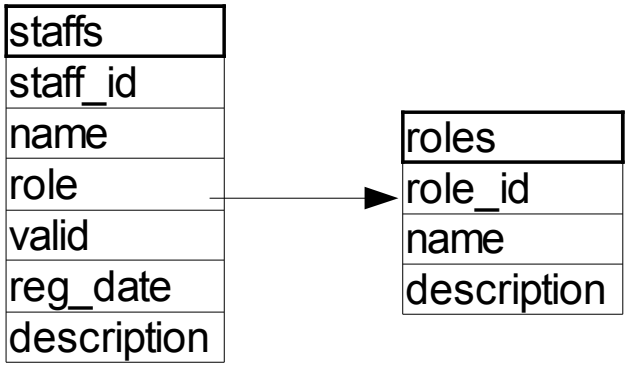
DB更新時の注意

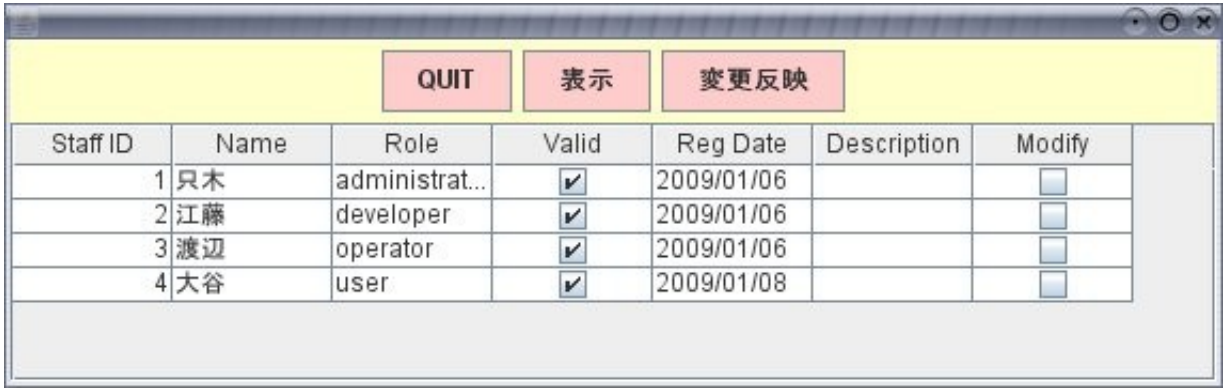
- DB更新に失敗する可能性に注意
  - 複数テーブルや他のデータを連携させている場合には、特に注意が必要
- DBには仮の変更を可能とする機能がある
- 自動更新の解除
  - java.sql.Connection.setAutoCommit(false)
- 更新の実行・キャンセル
  - java.sql.Connection.commit()
  - java.sql.Connection.rollback()

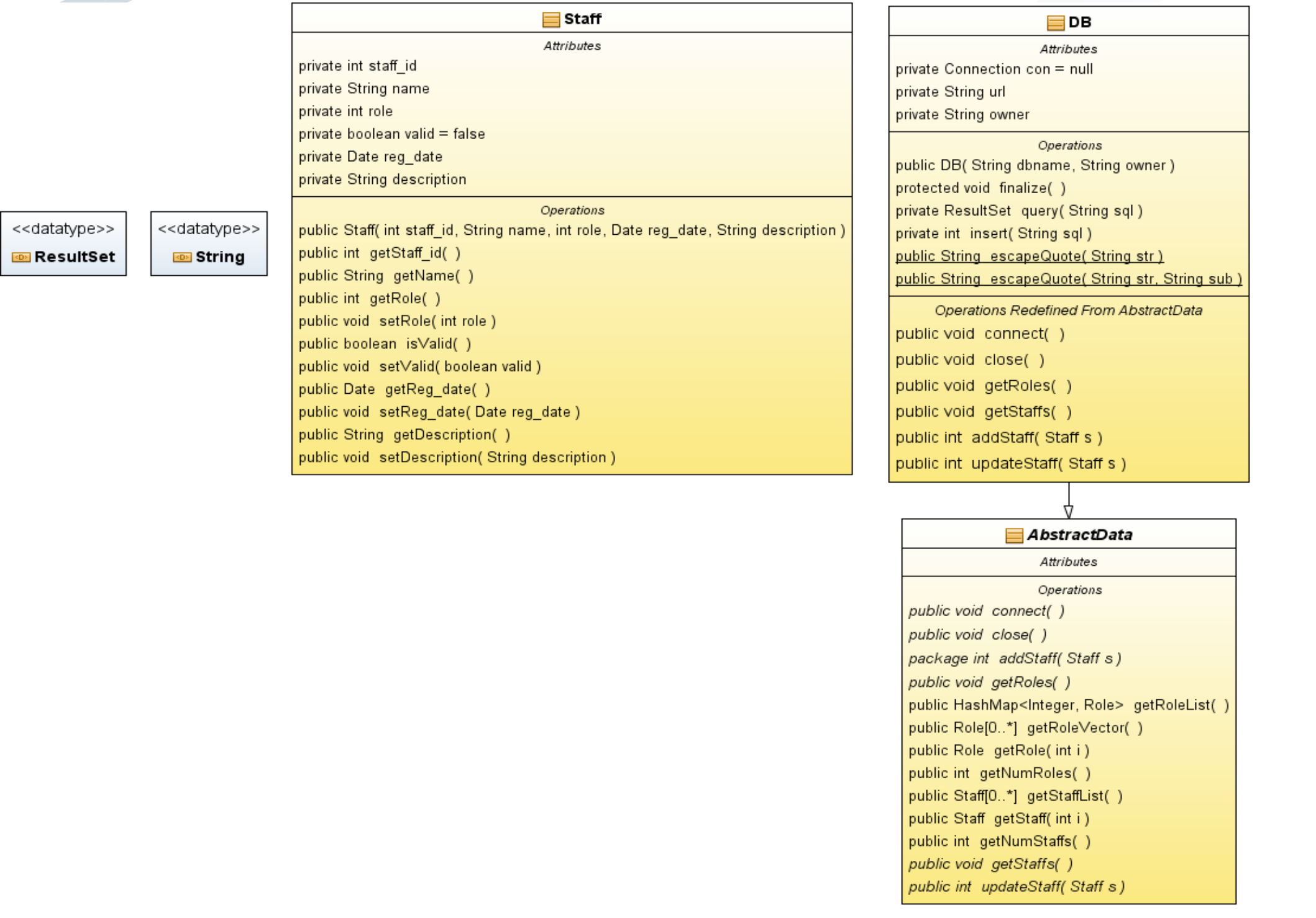


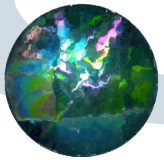
例題

- GUIからデータ検索、変更



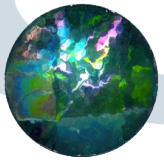



Staff
Attributes
private int staff_id
private String name
private int role
private boolean valid = false
private Date reg_date
private String description
Operations
public Staff(int staff_id, String name, int role, Date reg_date, String description)
public int getStaff_id( )
public String getName( )
public int getRole( )
public void setRole(int role)
public boolean isValid( )
public void setValid(boolean valid)
public Date getReg_date( )
public void setReg_date(Date reg_date)
public String getDescription( )
public void setDescription(String description)
DB
Attributes
private Connection con = null
private String url
private String owner
Operations
public DB(String dbname, String owner)
protected void finalize( )
private ResultSet query(String sql)
private int insert(String sql)
public String escapeQuote(String str)
public String escapeQuote(String str, String sub)
Operations Redefined From AbstractData
public void connect( )
public void close( )
public void getRoles( )
public void getStaffs( )
public int addStaff(Staff s)
public int updateStaff(Staff s)
AbstractData
Attributes
Operations
public void connect( )
public void close( )
package int addStaff(Staff s)
public void getRoles( )
public HashMap<Integer, Role> getRoleList( )
public Role[0..*] getRoleVector( )
public Role getRole(int i)
public int getNumRoles( )
public Staff[0..*] getStaffList( )
public Staff getStaff(int i)
public int getNumStaffs( )
public void getStaffs( )
public int updateStaff(Staff s)
<<datatype>>
ResultSet
<<datatype>>
String

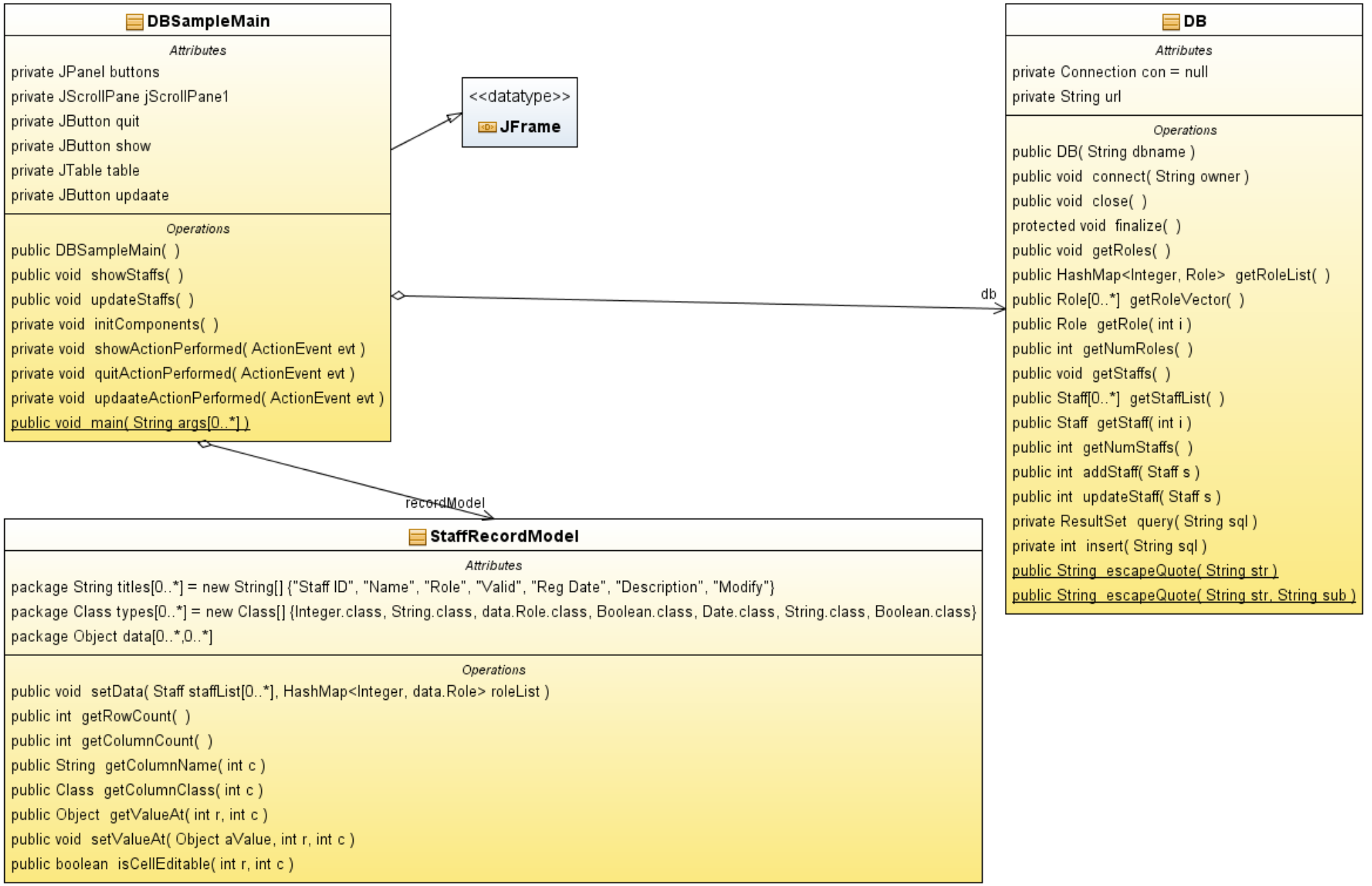


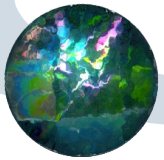
パッケージdata

- Staff：テーブルstaffsに対応したクラス
- Role：テーブルrolesに対応したクラス
- DB
  - Abstractデータクラスの派生
  - データベース接続全般を行うクラス
  - 検索
  - 挿入
  - 変更



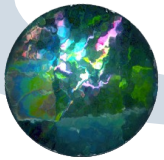
- AbstractDataクラス
  - データの保持方法に依存しないデータ操作
  - DB以外のデータ保持方法への対応


DBSampleMain
Attributes
private JPanel buttons
private JScrollPane jScrollPane1
private JButton quit
private JButton show
private JTable table
private JButton updaate
Operations
public DBSampleMain( )
public void showStaffs( )
public void updateStaffs( )
private void initComponents( )
private void showActionPerformed(ActionEvent evt)
private void quitActionPerformed(ActionEvent evt)
private void updaateActionPerformed(ActionEvent evt)
public void main(String args[0..*])
<<datatype>>
JFrame
db
DB
Attributes
private Connection con = null
private String url
Operations
public DB(String dbname)
public void connect(String owner)
public void close( )
protected void finalize( )
public void getRoles( )
public HashMap<Integer, Role> getRoleList( )
public Role[0..*] getRoleVector( )
public Role getRole(int i)
public int getNumRoles( )
public void getStaffs( )
public Staff[0..*] getStaffList( )
public Staff getStaff(int i)
public int getNumStaffs( )
public int addStaff(Staff s)
public int updateStaff(Staff s)
private ResultSet query(String sql)
private int insert(String sql)
public String escapeQuote(String str)
public String escapeQuote(String str, String sub)
recordModel
StaffRecordModel
Attributes
package String titles[0..*] = new String[] {"Staff ID", "Name", "Role", "Valid", "Reg Date", "Description", "Modify"}
package Class types[0..*] = new Class[] {Integer.class, String.class, data.Role.class, Boolean.class, Date.class, String.class, Boolean.class}
package Object data[0..*,0..*]
Operations
public void setData(Staff staffList[0..*], HashMap<Integer, data.Role> roleList)
public int getRowCount( )
public int getColumnCount( )
public String getColumnName(int c)
public Class getColumnClass(int c)
public Object getValueAt(int r, int c)
public void setValueAt(Object aValue, int r, int c)
public boolean isCellEditable(int r, int c)



パッケージgui

- DBSampleMain
  - javax.swing.JFrameの継承
  - 操作ボタンと検索結果の表
- StaffRecordModel
  - javax.swing.table.AbstractTableModelの継承
  - javax.swing.JTableにデータを設定する
  - データ編集の可否を設定



javax.swing.JTableの利用

- パネルに表を作成するクラス
- データの表示方法はjavax.swing.table.AbstractTableModelの派生クラスで定義
- データ編集方法はjavax.swing.DefaultCellEditorの派生クラスで定義
    - 一覧プルダウンを設定できる

DB.java

```
package data;

import java.sql.*;
import java.util.HashMap;

/**
 *
 * @author tadaki
 */
public class DB extends AbstractData {

    private Connection con = null;
    private String url;
    private String owner;

    /**
     * DBの指定
     * @param dbname DBの名前
     * @param owner  DBの所有者
     */
    public DB(String dbname, String owner) {
        url = "jdbc:postgresql://localhost:5432/" + dbname;
        this.owner = owner;
    }

    /**
     * データ源への接続
     * @throws SQLException
     * @throws ClassNotFoundException
     */
    public void connect() throws SQLException, ClassNotFoundException {
        Class.forName("org.postgresql.Driver");
        con = DriverManager.getConnection(url, owner, "");
    }

    /**
     * データ源を閉じる
     * @throws SQLException
     */
    public void close() throws SQLException {
        if (con == null) {
            return;
        }
        con.close();
```

```
    }

    @Override
    protected void finalize() throws Throwable {
        super.finalize();
        close();
    }

    /**********************************************************************
     * 検索
     */

    /**
     * Role一覧を取得
     * @throws SQLException
     */
    public void getRoles() throws SQLException {
        if (con == null) {
            return;
        }
        ResultSet r = query("select * from roles");
        roles = new HashMap<Integer, Role>();

        while (r.next()) {
            int id = r.getInt("role_id");
            Role role = new Role(id, r.getString("name"),
                    r.getString("description"));
            roles.put(id, role);
        }

    }

    /**
     * Staff一覧を取得
     * @throws SQLException
     */
    public void getStaffs() throws SQLException {
        if (con == null) {
            return;
        }
        ResultSet r = query("select * from staffs");
        staffs = new HashMap<Integer, Staff>();

        while (r.next()) {
```

```
            int id = r.getInt("staff_id");
            Staff staff = new Staff(id, r.getString("name"),
 r.getInt("role"),
                    new
 java.util.Date(r.getTimestamp("reg_date").getTime()),
                    r.getString("description"));
            staff.setValid(r.getBoolean("valid"));
            staffs.put(id, staff);
        }

    }


 /**************************************************************************
     * 挿入
     */
    /**
     * Staffを追加
     * @param s 追加するStaff
     * @return
     * @throws SQLException
     */
    public int addStaff(Staff s) throws SQLException {
        StringBuffer b = new StringBuffer();
        b.append("insert into Staffs ");
        b.append("(name,role,valid,reg_date,description) ");
        b.append("values (");
        b.append("'" + s.getName() + "',");
        b.append("'" + String.valueOf(s.getReg_date()) + "',");
        b.append("'" + String.valueOf(s.isValid()) + "',");
        b.append("'NOW',");
        b.append("'" + s.getDescription() + "'");
        b.append(")");
        int k = insert(b.toString());
        return k;
    }

    /**
     * Staffの情報を更新する
     * @param s
     * @return 更新した数
     * @throws SQLException
     */
    public int updateStaff(Staff s) throws SQLException {
```

```
        StringBuffer b = new StringBuffer();
        b.append("update Staffs set ");
        b.append("role='" + String.valueOf(s.getRole()) + "',");
        b.append("valid='" + String.valueOf(s.isValid()) + "',");
        if (s.getDescription() != null) {
            b.append("description='" + s.getDescription() + "',");
        } else {
            b.append("description='',");
        }
        b.append("reg_date='NOW' ");
        b.append("where staff_id='" + String.valueOf(s.getStaff_id() +
"'"));
        System.out.println(b.toString());
        int k = insert(b.toString());
        System.out.println(k);
        return k;
    }


/*****************************************************************************
     * ツール
     */

    private ResultSet query(String sql) throws SQLException {
        if (con == null) {
            return null;
        }

        ResultSet resultSet = null;
        Statement select = con.createStatement();
        resultSet = select.executeQuery(sql);
        return resultSet;
    }

    private int insert(String sql) throws SQLException {
        if (con == null) {
            return 0;
        }
        int count = 0;
        Statement stm;

        /**
         * 自動コミットをオフにする
         */
```

```
            con.setAutoCommit(false);
            stm = con.createStatement();
            count = stm.executeUpdate(sql);

            if (count != 0) {
                con.commit();
            } else {
                con.rollback();
            }

            return count;
        }

        /**
         * SQL文字列の無害化
         * @param str
         * @return
         */
        static public String escapeQuote(String str) {
            return escapeQuote(str, "'");
        }

        /**
         * SQL文字列の無害化
         * @param str
         * @param sub
         * @return
         */
        static public String escapeQuote(String str, String sub) {
            String ret = null;
            java.util.regex.Pattern p = java.util.regex.Pattern.compile("'");
            java.util.regex.Matcher m = p.matcher(str);
            ret = m.replaceAll(sub);
            return ret;
        }
    }
```