

SAMPLE PROJECT

BÀI 3: THỰC HIỆN DỰ ÁN 1

- ⊙ Kết thúc bài học này bạn có khả năng
 - ⊙ Nắm vững hơn kiến thức lập trình Jdbc
 - ⊙ Xây dựng thư viện tiện ích JdbcHelper
 - ⊙ Lập trình CSDL với mô hình DAO



Ôn lại kiến thức lập trình JDBC

 Statement

 ResultSet

 PreparedStatement

 CallableStatement

Xây dựng thư viện tiện ích để đơn giản hóa lập trình Jdbc

Tổ chức theo mô hình DAO

 Xây dựng model

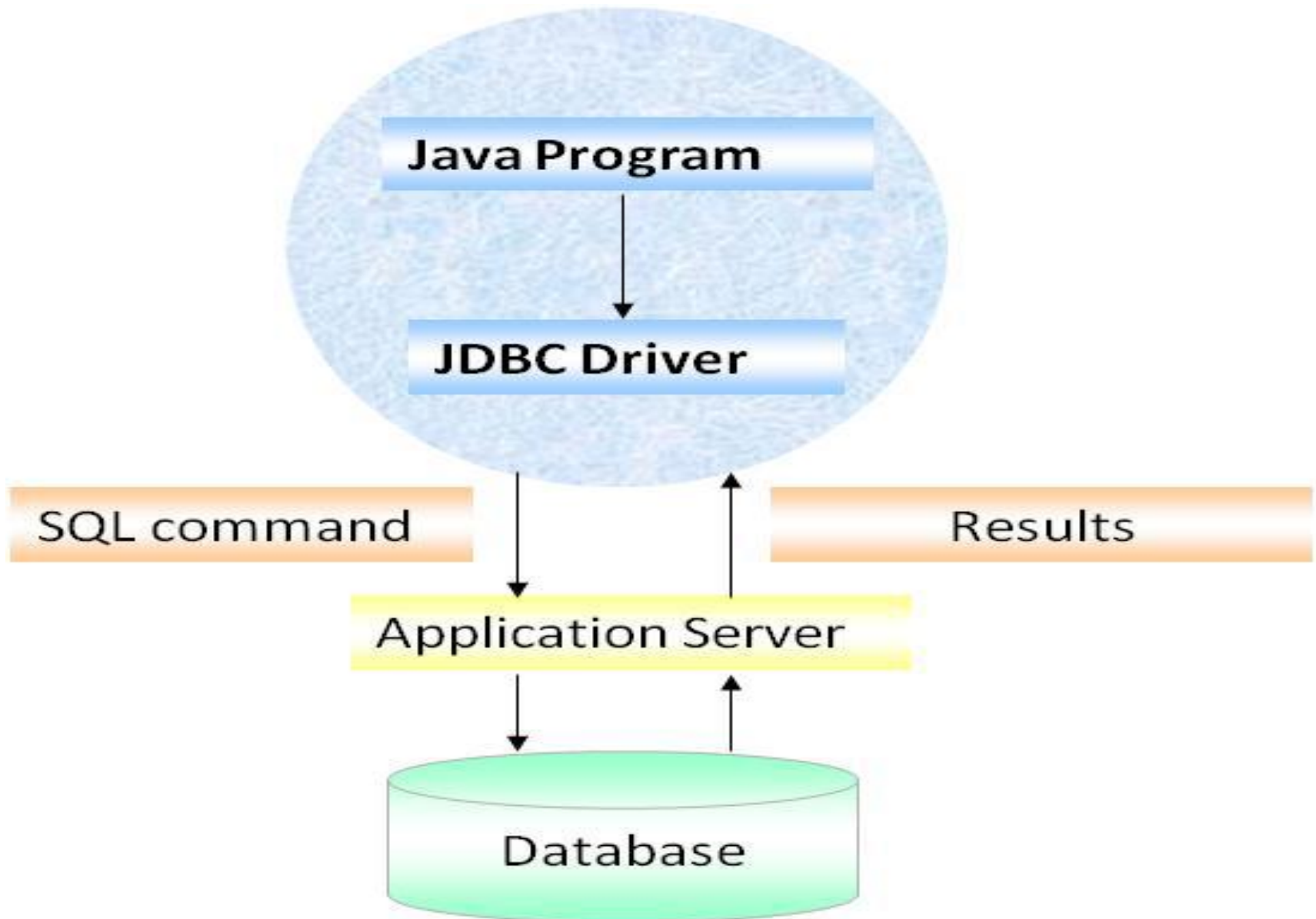
 Xây dựng DAO

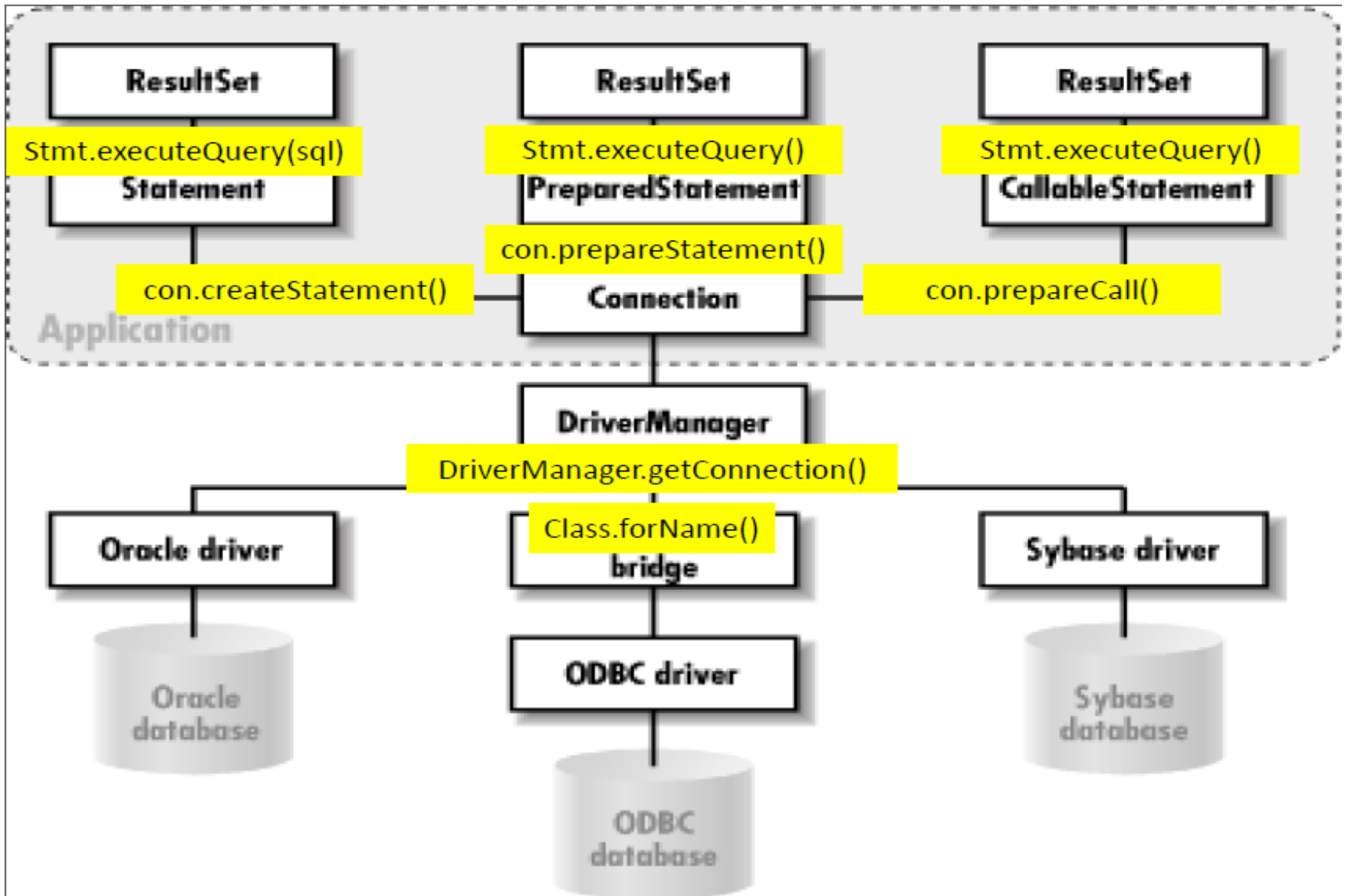
 Sử dụng DAO làm việc với CSDL



JDBC Review

MÔ HÌNH ỨNG DỤNG LÀM VIỆC VỚI CSDL





```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class Test {

    private static String driver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    private static String url = "jdbc:sqlserver://localhost;Database=PolyPro";
    private static String user = "sa";
    private static String password = "123456";

    public static void insert() {
        ...
    }

    public static void select() {
        ...
    }
}
```

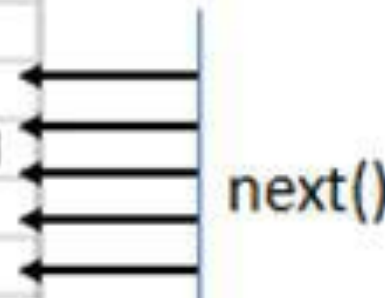
```
try {  
    //1. Nạp driver (sqljdbc4.jar)  
    Class.forName(driver);  
    //2. Mở kết nối đến CSDL Java  
    Connection connection = DriverManager.getConnection(url, user, password);  
    //3. Tạo Statement và thực thi câu lệnh SQL  
    Statement statement = connection.createStatement();  
    String sql = "INSERT INTO NhanVien (MaNV, MatKhau, HoTen, VaiTro) "  
        + "VALUES ('TeoNV', N'123456', N'Nguyễn Văn Tèo', 1)";  
    statement.executeUpdate(sql);  
    //4. Đóng kết nối đến CSDL  
    connection.close();  
}  
catch (Exception e) {  
    throw new RuntimeException(e);  
}
```


TRUY VẤN DỮ LIỆU VỚI STATEMENT

```
try {
    //1. Nạp driver (sqljdbc4.jar)
    Class.forName(driver);
    //2. Mở kết nối đến CSDL Java
    Connection connection = DriverManager.getConnection(url, user, password);
    //3. Tạo Statement và thực thi câu lệnh SQL
    Statement statement = connection.createStatement();
    String sql = "SELECT * FROM NhanVien";
    ResultSet resultSet = statement.executeQuery(sql);
    //4. Đọc dữ liệu từ CSDL
    while(resultSet.next()){
        String name = resultSet.getString("MaNV");
        boolean role = resultSet.getBoolean("VaiTro");
    }
    //5. Đóng kết nối đến CSDL
    connection.close();
}
catch (Exception e) {
    throw new RuntimeException(e);
}
```

- ❑ Khi ResultSet được tạo ra thì con trỏ sẽ trở vào vị trí trước bản ghi đầu tiên (before first).
- ❑ Gọi `ResultSet.next()` để chuyển con trỏ đến và đọc dữ liệu của bản ghi tiếp theo

1	Name	Age	Place
2	Harry	34	Florida
3	Samson	19	London
4	Johny	25	Ottawa
5	Carol	45	Auckland
6	Christina	23	Sydney
7	Mary	9	Rome



The diagram illustrates the movement of a cursor through a ResultSet. A vertical blue line represents the cursor's position. Five horizontal arrows point from this line to the right, indicating the sequence of calls to the `next()` method. The arrows start at the row containing 'Johny' (row 4) and end at the row containing 'Mary' (row 7), showing the cursor moving from row 4 to row 5, then to row 6, and finally to row 7.

```

Class.forName(driver);
Connection connection = DriverManager.getConnection(url, user, password);
String sql = "INSERT INTO NhanVien (MaNV, MatKhu, HoTen, VaiTro) "
            + "VALUES (?, ?, ?, ?)";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(1, "TeoNV");
statement.setString(2, "123456");
statement.setString(3, "Nguyễn Văn Tèo");
statement.setBoolean(4, true);
statement.executeUpdate();
connection.close();
  
```

```

Class.forName(driver);
Connection connection = DriverManager.getConnection(url, user, password);
String sql = "SELECT * FROM NhanVien WHERE MaNV=?";
PreparedStatement statement = connection.prepareStatement(sql);
ResultSet resultSet = statement.executeQuery();
while(resultSet.next()){
    String name = resultSet.getString("MaNV");
    boolean role = resultSet.getBoolean("VaiTro");
}
connection.close();
  
```

LÀM VIỆC VỚI CALLABLESTATEMENT

```

Class.forName(driver);
Connection connection = DriverManager.getConnection(url, user, password);
String sql = "{call sp_InsertNhanVien(?, ?, ?, ?)}";
CallableStatement statement = connection.prepareCall(sql);
statement.setString(1, "TeoNV");
statement.setString(2, "123456");
statement.setString(3, "Nguyễn Văn Tèo");
statement.setBoolean(4, true);
statement.executeUpdate();
connection.close();
  
```

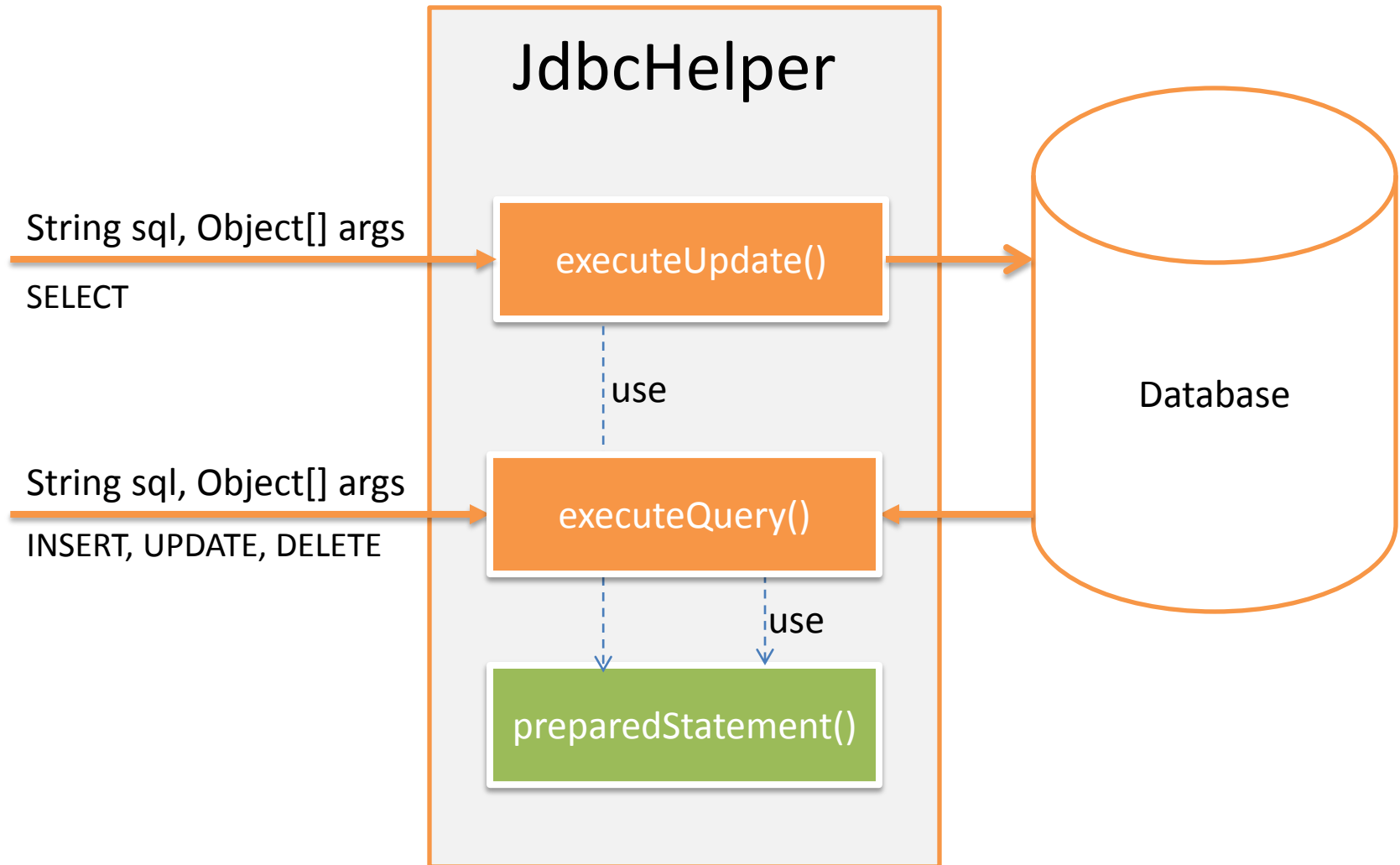
```

Class.forName(driver);
Connection connection = DriverManager.getConnection(url, user, password);
String sql = "{call sp_SelectNhanVien()}";
CallableStatement statement = connection.prepareCall(sql);
ResultSet resultSet = statement.executeQuery(sql);
while(resultSet.next()){
    String name = resultSet.getString("MaNV");
    boolean role = resultSet.getBoolean("VaiTro");
}
connection.close();
  
```

- ❑ statement.setObject(index, value) có thể thay thế cho statement.setXYZ(index, value) với điều kiện value phải có kiểu là XYZ
- ❑ Vì vậy có thể viết
 - ❖ statement.setObject(1, "TeoNV");
 - ❖ statement.setObject(2, "123456");
 - ❖ statement.setObject(3, "Nguyễn Văn Tèo");
 - ❖ statement.setObject(4, true);

Xây dựng JdbcHelper

XÂY DỰNG THƯ VIỆN JDBCHELPER



```

/*
 * Nạp driver
 */
static{
    ...
}
/**
 * Xây dựng PreparedStatement
 * @param sql là câu lệnh SQL chứa có thể chứa tham số. Nó có thể là một lời gọi thủ tục lưu
 * @param args là danh sách các giá trị được cung cấp cho các tham số trong câu lệnh sql
 * @return PreparedStatement tạo được
 * @throws java.sql.SQLException lỗi sai cú pháp
 */
public static PreparedStatement prepareStatement(String sql, Object...args) throws SQLException{
    ...
}
/**
 * Thực hiện câu lệnh SQL thao tác (INSERT, UPDATE, DELETE) hoặc thủ tục lưu thao tác dữ liệu
 * @param sql là câu lệnh SQL chứa có thể chứa tham số. Nó có thể là một lời gọi thủ tục lưu
 * @param args là danh sách các giá trị được cung cấp cho các tham số trong câu lệnh sql
 */
public static void executeUpdate(String sql, Object...args) {
    ...
}
/**
 * Thực hiện câu lệnh SQL truy vấn (SELECT) hoặc thủ tục lưu truy vấn dữ liệu
 * @param sql là câu lệnh SQL chứa có thể chứa tham số. Nó có thể là một lời gọi thủ tục lưu
 * @param args là danh sách các giá trị được cung cấp cho các tham số trong câu lệnh sql
 */
public static ResultSet executeQuery(String sql, Object...args) {
    ...
}

```


❑ Nạp driver

```
try {  
    Class.forName(driver);  
}  
catch (ClassNotFoundException ex) {  
    throw new RuntimeException(ex);  
}
```

❑ preparedStatement(String sql, Object...args)

```
Connection connection = DriverManager.getConnection(dburl, username, password);  
PreparedStatement pstmt = null;  
if(sql.trim().startsWith("{")){  
    pstmt = connection.prepareCall(sql);  
}  
else{  
    pstmt = connection.prepareStatement(sql);  
}  
for(int i=0;i<args.length;i++){  
    pstmt.setObject(i + 1, args[i]);  
}  
return pstmt;
```

❑ executeUpdate(String sql, Object...args)

```
try {
    PreparedStatement stmt = prepareStatement(sql, args);
    try {
        stmt.executeUpdate();
    }
    finally{
        stmt.getConnection().close();
    }
}
catch (SQLException e) {
    throw new RuntimeException(e);
}
```

❑ executeQuery(String sql, Object...args)

```
try {
    PreparedStatement stmt = prepareStatement(sql, args);
    return stmt.executeQuery();
}
catch (SQLException e) {
    throw new RuntimeException(e);
}
```

❑ Thao tác – INSERT, UPDATE, DELETE

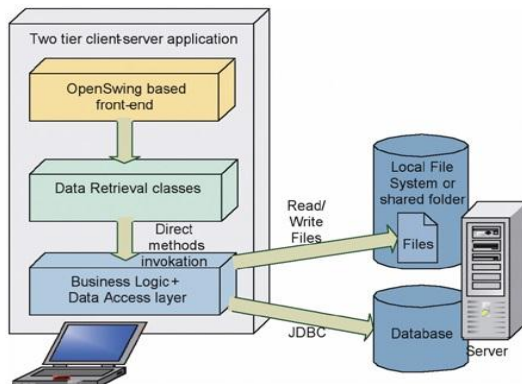
String sql="INSERT INTO NhanVien (MaNV, MatKhau, HoTen, VaiTro) VALUES (?, ?, ?, ?)";

***JdbcHelper.executeUpdate**(sql, "TeoNV", "123456", "Nguyễn Văn Tèo", true);*

❑ Truy vấn - SELECT

*String sql="SELECT * FROM NhanVien WHERE MaNV=?";*

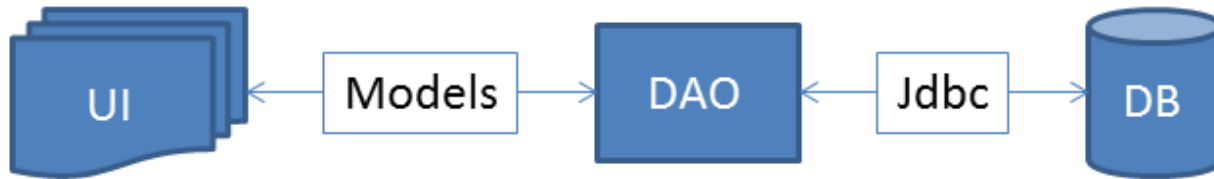
*ResultSet rs = **JdbcHelper.executeQuery**(sql, "TeoNV");*



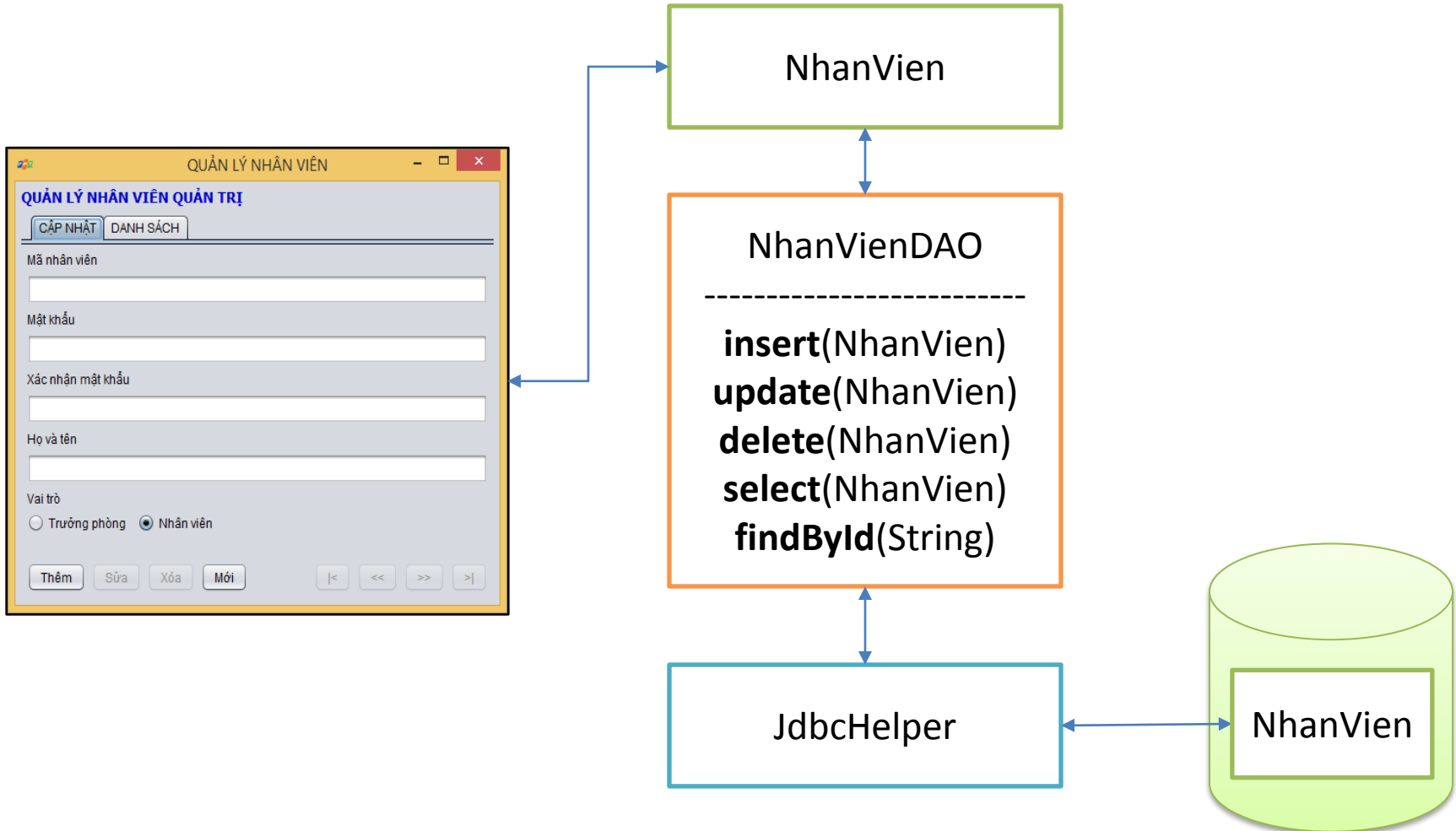
SAMPLE PROJECT

BÀI 3: THỰC HIỆN DỰ ÁN 1

Lập trình CSDL theo mô hình DAO



- ❑ UI: giao diện
- ❑ DB: cơ sở dữ liệu
- ❑ JdbcHelper: các hàm tiện ích làm việc với CSDL thông qua câu lệnh sql hoặc lời gọi thủ tục lưu
- ❑ Model: các lớp mô tả dữ liệu theo cấu trúc các bảng
- ❑ DAO (Data Access Object): là các lớp thao tác và truy vấn dữ liệu. Nó có nhiệm vụ chuyển đổi Model sang SQL và ngược lại.



- ❑ Lớp NhanVien được sử dụng để mô hình hóa dữ liệu của sinh viên.
- ❑ Đối tượng NhanVien được sử dụng để giao tiếp với NhanVienDAO

```
public class NhanVien {  
    private String maNV;  
    private String matKhau;  
    private String hoTen;  
    private boolean vaiTro = false;  
  
    @Override  
    public String toString() {  
        return this.hoTen;  
    }  
  
    getters/setters  
}
```



```

public class NhanVienDAO {
    public void insert(NhanVien model) {
        ...
    }
    public void update(NhanVien model) {
        ...
    }
    public void delete(String MaNV) {
        ...
    }
    public List<NhanVien> select() {
        ...
    }
    public NhanVien findById(String manv) {
        ...
    }
    private List<NhanVien> select(String sql, Object...args) {
        ...
    }
    private NhanVien readFromResultSet(ResultSet rs) throws SQLException {
        ...
    }
}

```

2 phương thức private
phục vụ các phương
thức public

❑ Ngoài **findById()** và **select()** có thể bổ sung các phương thức truy vấn khác

```

public void insert(NhanVien model) {
    String sql="INSERT INTO NhanVien (MaNV, MatKhau, HoTen, VaiTro) VALUES (?, ?, ?, ?)";
    Jdbc.executeUpdate(sql,
        model.getMaNV(), model.getMatKhau(),
        model.getHoTen(), model.getVaiTro());
}

public void update(NhanVien model) {
    String sql="UPDATE NhanVien SET MatKhau=?, HoTen=?, VaiTro=? WHERE MaNV=?";
    Jdbc.executeUpdate(sql,
        model.getMatKhau(), model.getHoTen(),
        model.getVaiTro(),model.getMaNV());
}

public void delete(String MaNV) {
    String sql="DELETE FROM NhanVien WHERE MaNV=?";
    Jdbc.executeUpdate(sql, MaNV);
}

public List<NhanVien> select(){
    String sql="SELECT * FROM NhanVien";
    return select(sql);
}

public NhanVien findById(String manv) {
    String sql="SELECT * FROM NhanVien WHERE MaNV=?";
    List<NhanVien> list = select(sql, manv);
    return list.size() > 0 ? list.get(0) : null;
}

```

```

private List<NhanVien> select(String sql, Object...args){
    List<NhanVien> list=new ArrayList<>();
    try {
        ResultSet rs = null;
        try {
            rs = Jdbc.executeQuery(sql, args);
            while(rs.next()){
                NhanVien model=readFromResultSet(rs);
                list.add(model);
            }
        }
        finally{
            rs.getStatement().getConnection().close();
        }
    }
    catch (SQLException ex) {
        throw new RuntimeException(ex);
    }
    return list;
}

private NhanVien readFromResultSet(ResultSet rs) throws SQLException{
    NhanVien model=new NhanVien();
    model.setMaNV(rs.getString("MaNV"));
    model.setMatKhau(rs.getString("MatKhau"));
    model.setHoTen(rs.getString("HoTen"));
    model.setVaiTro(rs.getBoolean("VaiTro"));
    return model;
}

```

❑ Tạo model

```
NhanVien model = new NhanVien();  
model.setMaNV("TeoNV");  
model.setMatKhau("123456");  
model.setHoTen("Nguyễn Văn Tèo");  
model.setVaiTro(true);
```

❑ Sử dụng DAO để làm việc với CSDL

```
NhanVienDAO dao = new NhanVienDAO();  
dao.insert(model);  
NhanVien teo = dao.findById("TeoNV");
```

- ❑ ThongKeDAO là DAO gồm các chức năng truy vấn dữ liệu cho việc tổng hợp – thống kê.
- ❑ DAO này gọi đến các stored procedure đã được xây dựng trước đó để lấy dữ liệu cung cấp cho giao diện.

- ❖ **getNguoiHoc()**

- Tổng hợp người học từng năm

- ❖ **getBangDiem()**

- Bảng điểm của khóa

- ❖ **getDiemTheoChuyenDe()**

- Tổng hợp điểm theo chuyên đề

- ❖ **getDoanhThu()**

- Tổng hợp doanh thu theo chuyên đề

ThongKeDAO

**getNguoiHoc()
getBangDiem()
getDiemTheoChuyenDe()
getDoanhThu()**

```
public class ThongKeDAO {  
    public List<Object[]> getNguoiHoc() {  
        ...  
    }  
  
    public List<Object[]> getBangDiem(Integer makh) {  
        ...  
    }  
  
    public List<Object[]> getDiemTheoChuyenDe() {  
        ...  
    }  
  
    public List<Object[]> getDoanhThu(int nam) {  
        ...  
    }  
}
```

```
List<Object[]> list=new ArrayList<>();
try {
    ResultSet rs = null;
    try {
        String sql="{call sp_ThongKeNguoiHoc}";
        rs = Jdbc.executeQuery(sql);
        while(rs.next()) {
            Object[] model={
                rs.getInt("Nam"),
                rs.getInt("SoLuong"),
                rs.getDate("DauTien"),
                rs.getDate("CuoiCung")
            };
            list.add(model);
        }
    }
    finally{
        rs.getStatement().getConnection().close();
    }
}
catch (SQLException ex) {
    throw new RuntimeException(ex);
}
return list;
```

THONGKEDAO.GETBANGDIEM()

```
List<Object[]> list=new ArrayList<>();
try{
    ResultSet rs = null;
    try {
        String sql="{call sp_BangDiem (?)}";
        rs = Jdbc.executeQuery(sql, makh);
        while(rs.next()){
            double diem = rs.getDouble("Diem");
            String xepLoai = "Xuất sắc";
            if(diem < 0){xepLoai = "Chưa nhập"; }
            else if(diem < 3){ xepLoai = "Kém"; }
            else if(diem < 5){ xepLoai = "Yếu"; }
            else if(diem < 6.5){ xepLoai = "Trung bình"; }
            else if(diem < 7.5){ xepLoai = "Khá"; }
            else if(diem < 9){ xepLoai = "Giỏi"; }
            Object[] model={ rs.getString("MaNH"),
                rs.getString("HoTen"), diem, xepLoai };
            list.add(model);
        }
    }
    finally{
        rs.getStatement().getConnection().close();
    }
}
catch(SQLException ex){
    throw new RuntimeException(ex);
}
return list;
```



```

List<Object[]> list=new ArrayList<>();
try {
    ResultSet rs = null;
    try {
        String sql="{call sp_ThongKeDiem}";
        rs = Jdbc.executeQuery(sql);
        while(rs.next()){
            Object[] model={
                rs.getString("ChuyenDe"),
                rs.getInt("SoHV"),
                rs.getDouble("ThapNhat"),
                rs.getDouble("CaoNhat"),
                rs.getDouble("TrungBinh")
            };
            list.add(model);
        }
    }
    finally{
        rs.getStatement().getConnection().close();
    }
}
catch (SQLException ex) {
    throw new RuntimeException(ex);
}
return list;

```

THONGKEDAO.GETDOANHTHU()

```
List<Object[]> list=new ArrayList<>();
try {
    ResultSet rs = null;
    try {
        String sql="{call sp_ThongKeDoanhThu (?)}";
        rs = Jdbc.executeQuery(sql, nam);
        while(rs.next()){
            Object[] model={
                rs.getString("ChuyenDe"),
                rs.getInt("SoKH"),
                rs.getInt("SoHV"),
                rs.getDouble("DoanhThu"),
                rs.getDouble("ThapNhat"),
                rs.getDouble("CaoNhat"),
                rs.getDouble("TrungBinh")
            };
            list.add(model);
        }
    }
    finally{
        rs.getStatement().getConnection().close();
    }
}
catch (SQLException e) {
    throw new RuntimeException(e);
}
return list;
```

- ☑ Ôn lại kiến thức lập trình JDBC
 - ☑ Statement
 - ☑ ResultSet
 - ☑ PreparedStatement
 - ☑ CallableStatement
- ☑ Xây dựng thư viện tiện ích để đơn giản hóa lập trình Jdbc
- ☑ Tổ chức theo mô hình DAO
 - ☑ Xây dựng model
 - ☑ Xây dựng DAO
 - ☑ Sử dụng DAO làm việc với CSDL





Cảm ơn