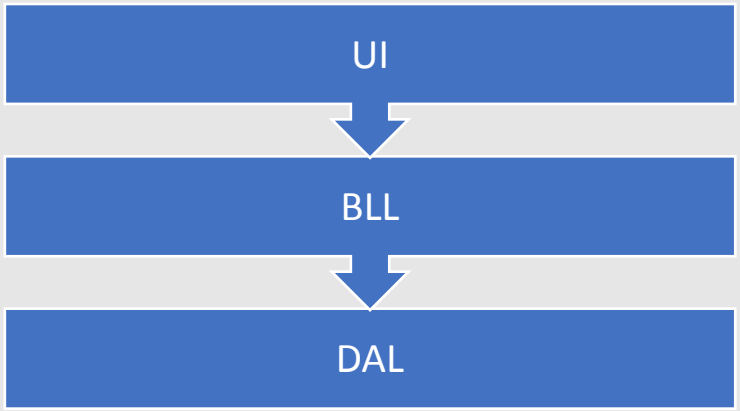
JAVA JDBC

Thực hành 3

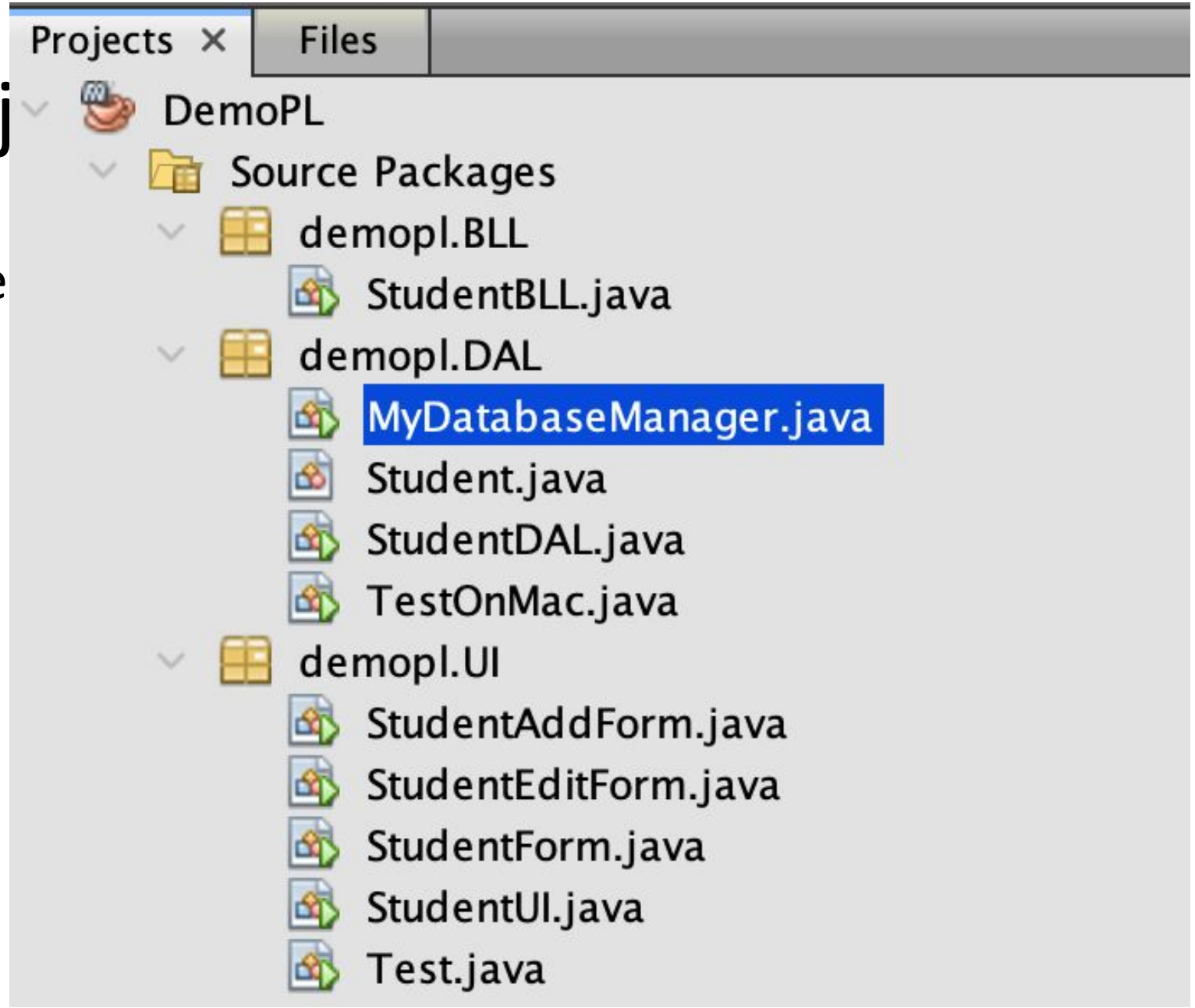
Nội dung chính

- Viết 3 Layer
  - DAL: Truy xuất CSDL
  - BLL: xử lý dữ liệu
  - UI: hiển thị dữ liệu

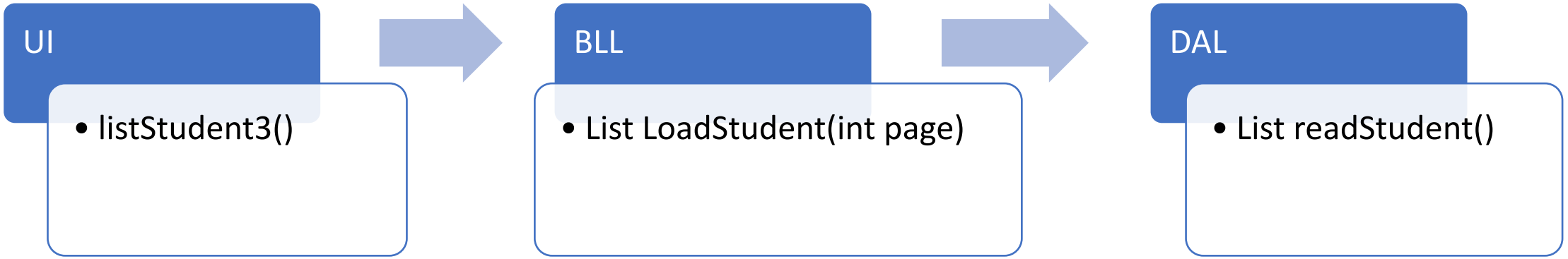


Cấu trúc của proj

- Mỗi layer là một package
  - DAL
  - BLL
  - UI



A. Xử lý hiển thị danh sách



A.1. Lớp DAL: StudentDAL

- Hàm lấy danh sách sinh viên, tương tự khi viết 2 layer.
- Trả về mảng

```java
//2 layer
public ArrayList readStudent() throws SQLException
{
    String query = "SELECT * FROM Person WHERE EnrollmentDate >0";
    ResultSet rs = StudentDAL.doReadQuery(query);
    ArrayList list = new ArrayList();

    if (rs != null) {
        int i = 1;

        while (rs.next()) {
            Student s = new Student();
            s.setPersonId(rs.getInt("PersonID"));
            s.setFirstName(rs.getString("FirstName"));
            s.setLastName(rs.getString("LastName"));
            list.add(s);
        }
    }
    return list;
}
```

A.2. Lớp BLL: StudentBLL

- Viết hàm LoadStudent hiển thị dữ liệu theo từng trang.
- Gọi DAL để lấy dữ liệu sau đó phân trang
- Có thể trả về kiểu dữ liệu gần với lớp UI hơn.

```
public List LoadStudents(int page) throws SQLException
{
    int numofrecords = 30;
    ArrayList list = stdDal.readStudent();
    int size = list.size();
    int from, to;
    from = (page - 1) * numofrecords;
    to = page * numofrecords;

    return list.subList(from, Math.min(to, size));
}
```

A.3. Lớp UI: StudentForm

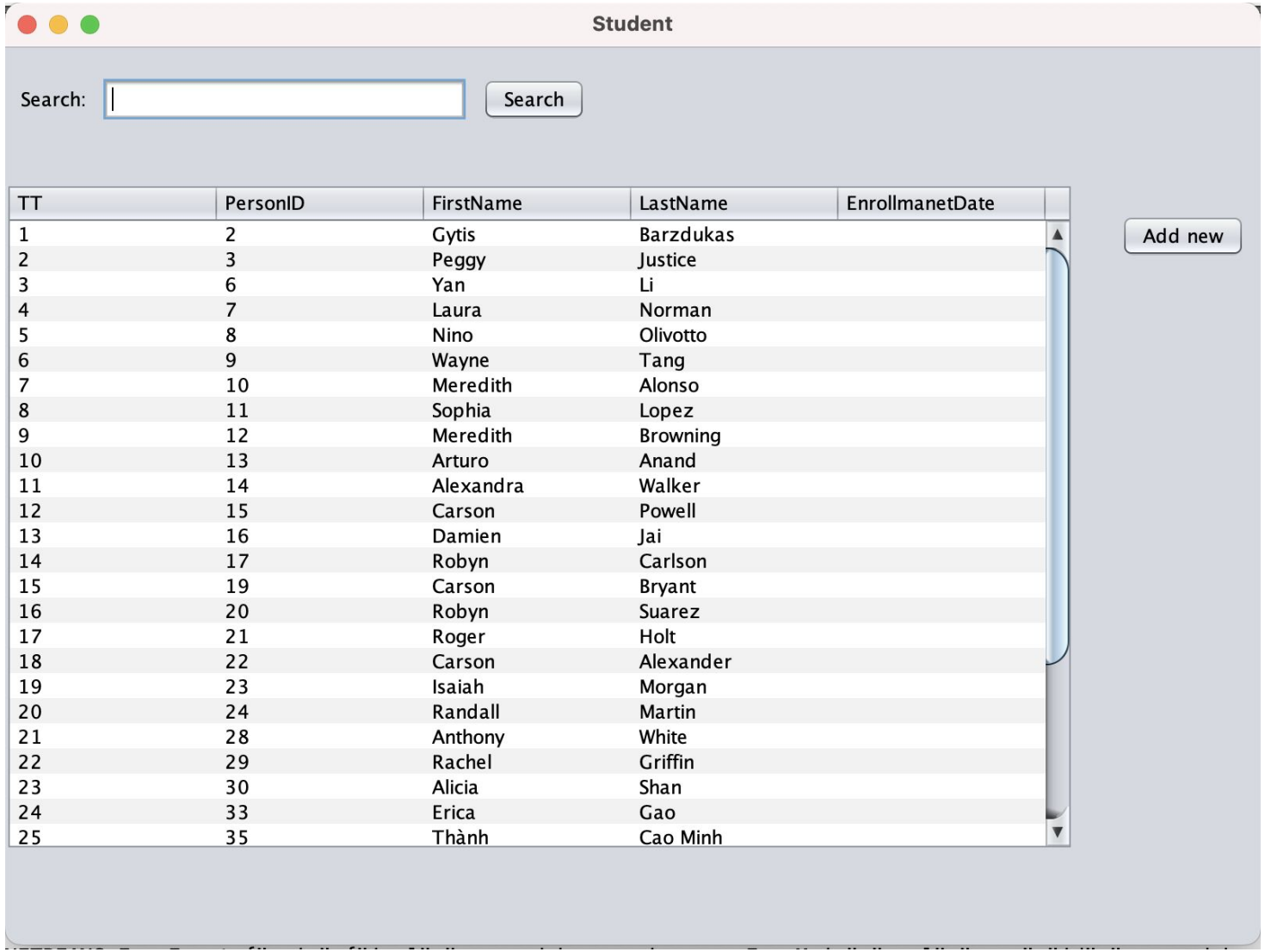
- Gọi lớp BLL để hiển thị dữ liệu

```
//for 3layer`
private void listStudent3() throws SQLException {
    List list = std.LoadStudents(1);
    DefaultTableModel model = convertStudent(list);
    jTable1.setModel(model);
    lbStatus.setText("Num of rows: " + list.size());
}

private DefaultTableModel convertStudent(List list) {
    String[] columnNames = {"TT", "PersonID", "FirstName", "LastName", "EnrollmanetDate"};
    Object[][] data = new Object[list.size()][5];
    for (int i = 0; i < list.size(); i++) {
        Student s = (Student) list.get(i);
        data[i][0] = i + 1;
        data[i][1] = s.getPersonId();
        data[i][2] = s.getFirstName();
        data[i][3] = s.getLastName();
        data[i][4] = s.getEnrollmentDate();
    }
    DefaultTableModel model = new DefaultTableModel(data, columnNames);
    return model;
}
```

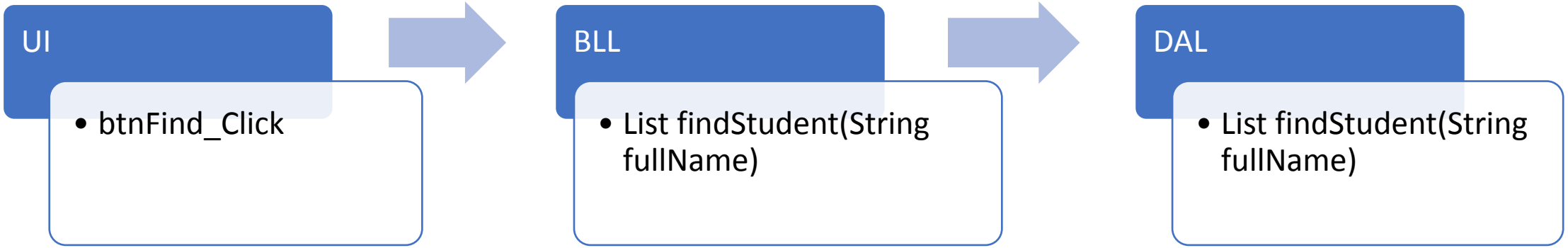
A.3. Lớp UI: StudentForm

- Dùng JTable để hiển thị data



TT	PersonID	FirstName	LastName	EnrollmanetDate
1	2	Gytis	Barzdukas	
2	3	Peggy	Justice	
3	6	Yan	Li	
4	7	Laura	Norman	
5	8	Nino	Olivotto	
6	9	Wayne	Tang	
7	10	Meredith	Alonso	
8	11	Sophia	Lopez	
9	12	Meredith	Browning	
10	13	Arturo	Anand	
11	14	Alexandra	Walker	
12	15	Carson	Powell	
13	16	Damien	Jai	
14	17	Robyn	Carlson	
15	19	Carson	Bryant	
16	20	Robyn	Suarez	
17	21	Roger	Holt	
18	22	Carson	Alexander	
19	23	Isaiah	Morgan	
20	24	Randall	Martin	
21	28	Anthony	White	
22	29	Rachel	Griffin	
23	30	Alicia	Shan	
24	33	Erica	Gao	
25	35	Thành	Cao Minh	

B. Xử lý tìm kiếm



B. Xử lý tìm kiếm

- StudentDAL

```java
public List findStudents(String fullName) throws SQLException
{
    String query = "SELECT * FROM Person WHERE concat(FirstName, ' ', LastName)  LIKE ?";
    PreparedStatement p = StudentDAL.getConnection().prepareStatement(query);
    p.setString(1, "%" + fullName + "%");
    ResultSet rs = p.executeQuery();
    List list = new ArrayList();

    if (rs != null) {
        int i = 1;

        while (rs.next()) {
            Student s = new Student();
            s.setPersonId(rs.getInt("PersonID"));
            s.setFirstName(rs.getString("FirstName"));
            s.setLastName(rs.getString("LastName"));
            list.add(s);
        }
    }
    return list;
}
```

B. Xử lý tìm kiếm

- StudentBLL

```
public List findStudent(String fullname) throws SQLException {
    List list = new ArrayList();

    list = stdDal.findStudents(fullname);

    return list;

}
```

B. Xử lý tìm kiếm

- UI: StudentForm

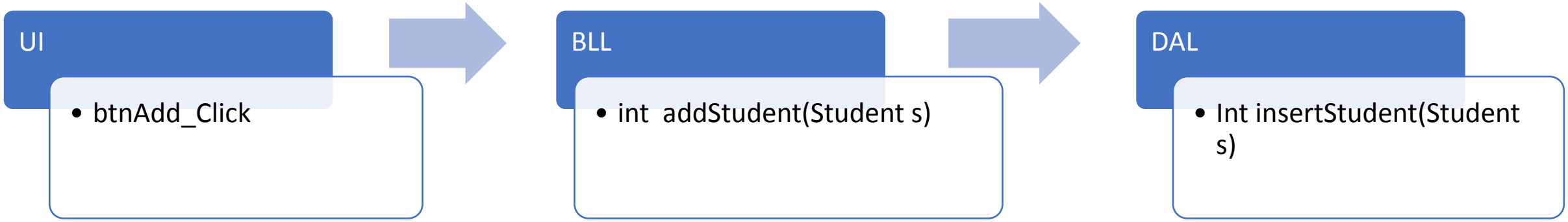
```
public void btnFind_Click(ActionEvent e) {
    try {

        String fullname = jtxtFind.getText();
        if(fullname.isBlank() == false){
        List list = std.findStudent(fullname);
        DefaultTableModel model = convertStudent(list);
        jTable1.setModel(model);
        lbStatus.setText("Num of rows: " + list.size());
        }
        else
        {
            JOptionPane.showMessageDialog(this, "fullname is empty", "Message", JOptionPane.ERROR_MESSAGE);
        }

    } catch (SQLException ex) {
        Logger.getLogger(StudentForm.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

C. Xử lý thêm mới



C. Xử lý thêm mới

- Lớp StudentDAL

```
public int insertStudent(Student s) throws SQLException {
    String query = "Insert Person (FirstName, LastName, EnrollmentDate) VALUES (?, ?, ?)";
    PreparedStatement p = StudentDAL.getConnection().prepareStatement(query);
    p.setString(1, s.getFirstName());
    p.setString(2, s.getLastName());
    p.setString(3, s.getEnrollmentDate().toString());
    int result = p.executeUpdate();
    return result;
}
```

C. Xử lý thêm mới

- Lớp StudentBLL
- Lớp StudentFormAdd

```java
public int addStudent(Student s) throws SQLException {
    int result = stdDal.insertStudent(s);
    return result;
}
```

```java
public void btnAdd_Click(ActionEvent e) {
    Student s = new Student();
    s.setFirstName(jtxtFirstName.getText());
    s.setLastName(jtxtLastName.getText());
    Date date = Date.valueOf(jtxtEnrollmentDate.getText());
    s.setEnrollmentDate(date);

    try {
        if (std.addStudent(s) > 0) {
            JOptionPane.showMessageDialog(this, "Complete add student", "Message", JOptionPane.INFORMATION_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "Error add student", "Message", JOptionPane.ERROR_MESSAGE);
        }
    } catch (SQLException ex) {
        Logger.getLogger(StudentAddForm.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```