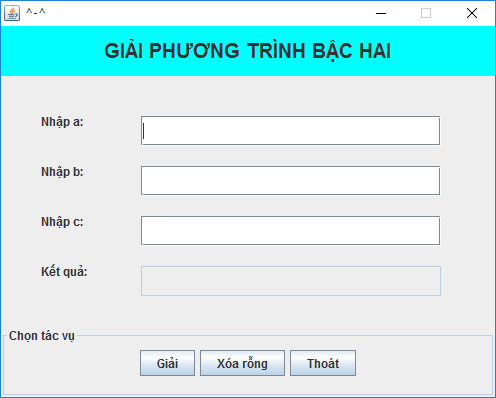
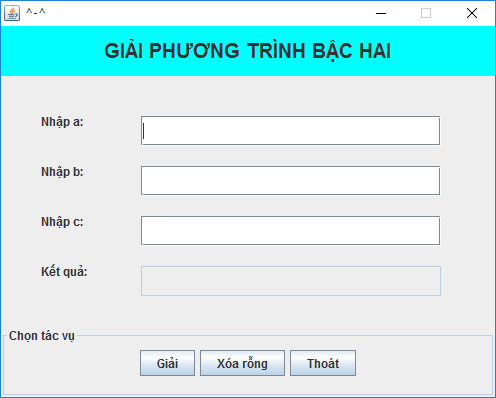
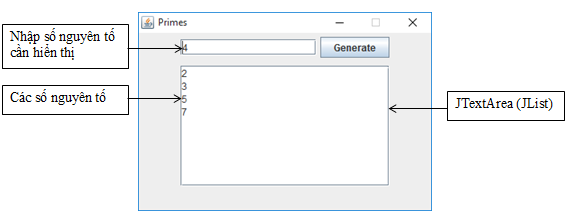
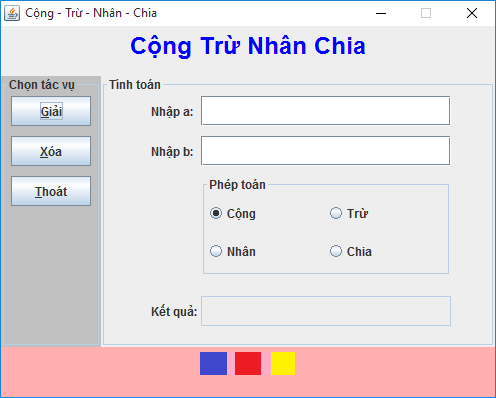
**BÀI TẬP THỰC HÀNH TUẦN 1**

*Nội dung thực hành:*

* *Swing container components*
* *Event Handling*
* *Swing Components*
* *Layout Manager*

1. Thiết kế và hiện thực giao diện sau:
2. Viết một chương trình hiển thị N số nguyên tố, trong đó N là một giá trị nhập trong 1 JTextField. Khi người sử dụng click vào JButton “Generate”, N số nguyên tố đầu tiên được ghi vào một JTextArea (JList) như hình bên dưới. Một số nguyên tố là một số nguyên lớn hơn 1, chỉ chia hết cho chính nó và 1.

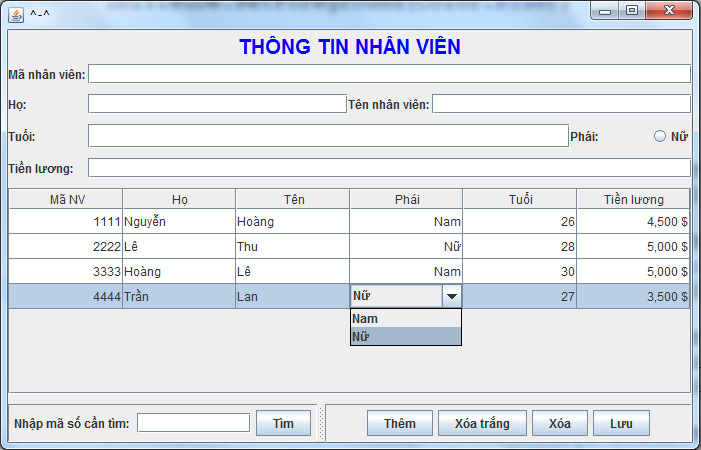


1. Thiết kế và hiện thực giao diện sau *(bài tập làm ở nhà)*:

**BÀI TẬP THỰC HÀNH TUẦN 2**

*Nội dung thực hành: Swing Advanced Controls*

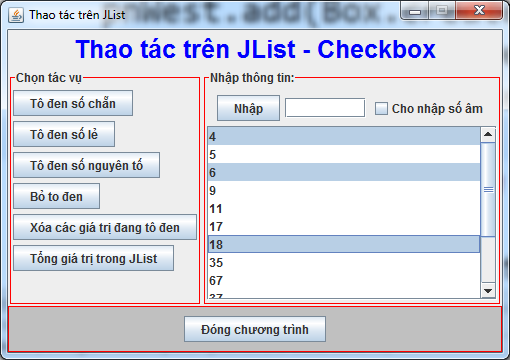
* *Thực hành đưa dữ liệu lên JTable (JList), xử lý dữ liệu.*
* *Thực hành TableColumn.*
* *Thực hành TableCellRenderer.*
* *Thực hành TableCellEditor*

1. Thiết kế và hiện thực giao diện sau:

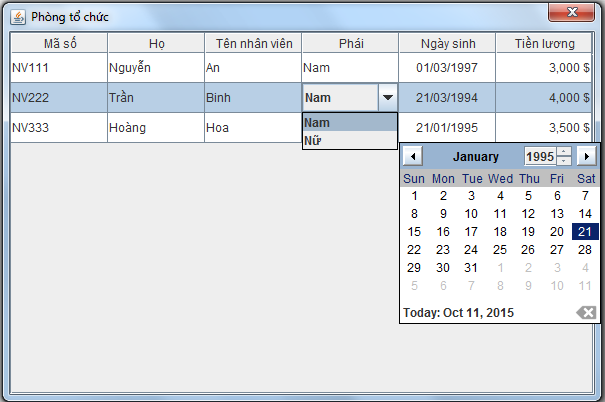
* Chương trình cho phép người sử dụng cập nhật dữ liệu trực tiếp trên bảng. Không cho phép cập nhật mã nhân viên.

**Customize: Sao chép project sang 1 project mới. Hiển thị dữ liệu trong JList theo dạng cột.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1111 | Nguyễn | Hoàng | Nam | 26 (Tuổi) | 4,500.0 $ |
| 2222 | Lê | Thu | Nữ | 28 (Tuổi) | 5,000.0 $ |
| 3333 | Hoàng | Lê | Nam | 30 (Tuổi) | 5,000.0 $ |
| 4444 | Trần | Lan | Nữ | 27 (Tuổi) | 3,500.0 $ |

1. Thiết kế và hiện thực giao diện sau *(bài tập làm ở nhà):*
2. Thiết kế và hiện thực giao diện sau *(bài tập làm ở nhà):*

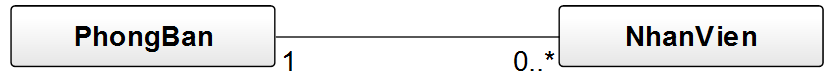
* Download JDatePicker : <http://sourceforge.net/projects/jdatepicker/files/Releases/1.3.x/>

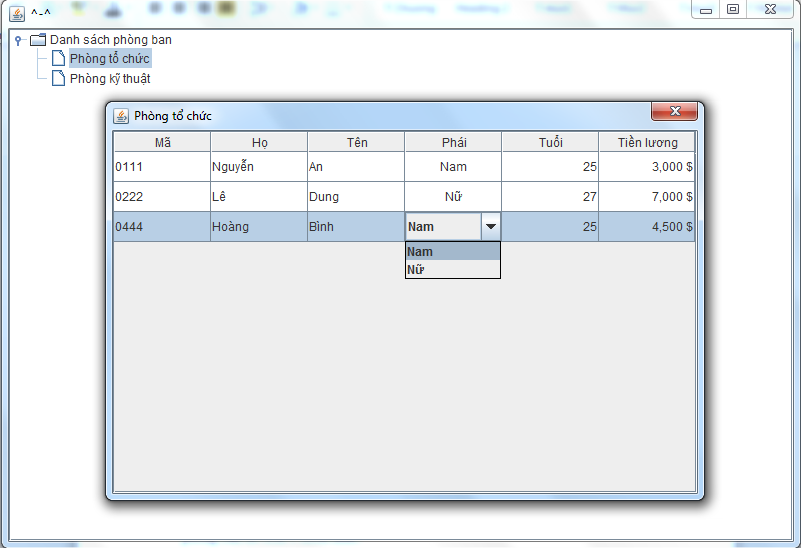


**BÀI TẬP THỰC HÀNH TUẦN 3**

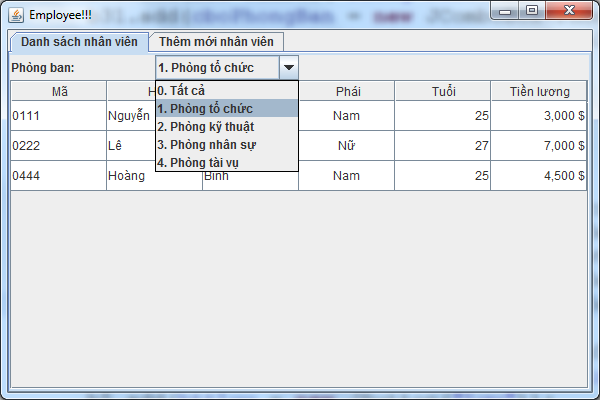
*Nội dung thực hành:*

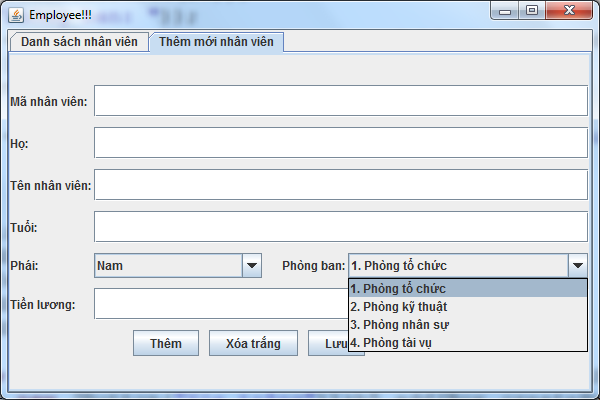
* *Thực hành JTree*
* *Thực hành JTabbedPane*

Cho class diagram:

1. Thiết kế và hiện thực giao diện sau:

* Form chính liệt kê danh sách các phòng ban.
* Khi người dùng double click vào 1 phòng ban, sẽ liệt kê danh sách các nhân viên của phòng ban đó trên 1 form khác.

1. Thiết kế và hiện thực giao diện sau *(bài tập làm ở nhà):*



**BÀI TẬP THỰC HÀNH TUẦN 4**

*Nội dung thực hành:*

* *Thực hành IO*
* *Thực hành Regular Expression*

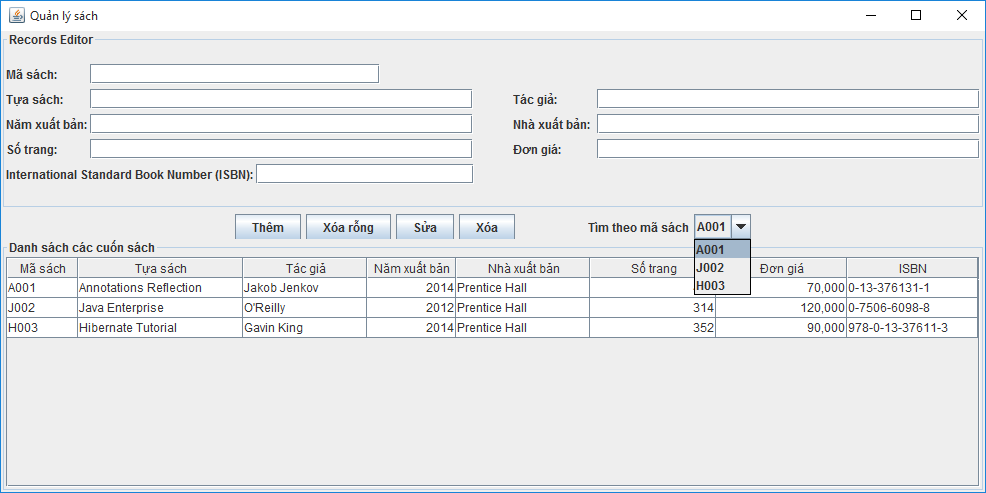
**Bài 1: Bài toán:**

Cho tập tin *DanhMucSach.txt*, chứa dữ liệu về thông tin các *cuốn sách* dạng văn bản. Dòng đầu tiên là tiêu đề cột, các dòng kế tiếp là giá trị dữ liệu tương ứng cho từng cột. Mỗi giá trị trong dòng ngăn cách bằng dấu “;”.

|  |
| --- |
| MaSach;TuaSach;TacGia;NamXuatBan;NhaXuatBan;SoTrang;DonGia;ISBN  A001;Annotations Reflection;Jakob Jenkov;2014;Prentice Hall;420;70000;0-13-376131-1  J002;Java Enterprise;O'Reilly;2012;Prentice Hall;314;120000;0-7506-6098-8  H003;Hibernate Tutorial;Gavin King;2014;Prentice Hall;352;90000;978-0-13-37611-3 |

**Yêu cầu:**

Viết chương trình đọc dữ liệu từ tập tin, đưa lên trên JTable của giao diện, thao tác trên giao diện và cập nhật dữ liệu lại file.



1. Thiết kế giao diện trên.
2. Hiện thực giao diện với các yêu cầu sau:

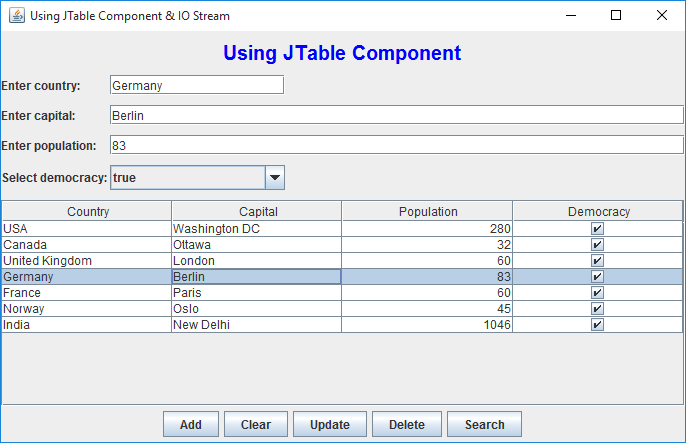
* Tạo file *data/ DanhMucSach.txt*  trong project. Đọc dữ liệu từ file lên JTable.
* Chương trình không cho phép người sử dụng sửa dữ liệu trực tiếp trên JTable. Khi người sử dụng click chuột chọn 1 dòng trên JTable, dữ liệu sẽ hiện thị trên các component.
* Nút *Sửa*: Chỉnh sửa giá trị dữ liệu trên các component của cuốn sách được chọn.
* Nút *Thêm*: Thêm 1 cuốn sách mới.
  + Mã sách không được rỗng, không được trùng.
  + Mã sách phải theo qui ước sau: Có ký tự đầu là ký tự đầu của tựa sách, theo sau là 3 ký số.
  + Tựa sách, và tác giả không được để trống, có thể gồm nhiều từ ngăn cách bởi khoảng trắng. Không chứa ký số hoặc các ký tự đặc biệt khác, ngoại trừ ký tự ‘.
  + ISBN có mẫu dạng X-X-X-X (hoặc X-X-X-X-X). Trong đó, X gồm các ký số, ít nhất là 1 ký số.
* Nút *Xóa rỗng*: Xóa rỗng nội dung trong các component*.*
* Nút *Xóa*: Xóa dòng dữ liệu được chọn. Hỏi nhắc người dùng trước khi xóa.
* Chức năng *Tìm theo mã sách*: Tìm kiếm cuốn sách dựa theo mã sách. Khi người dùng chọn 1 cuốn sách trong JComboBox, hiển thị thông tin trên các component, tô đen dòng vừa tìm trên JTable và scroll tới nó.

*Ghi chú:*

1. *Thao tác Add, Update, Delete phải cập nhật lại file, cập nhật dữ liệu trên các component, hiển thị lại dữ liệu trên JTable, JComboBox.*
2. *Chương trình phải thông báo các message cụ thể nếu người dùng gõ dữ liệu không phù hợp.*

**Bài 2:** *(Bài tập làm ở nhà)*

Viết chương trình thao tác trên giao diện và cập nhật dữ liệu lại file. Sử dụng luồng đối tượng để đọc ghi dữ liệu trên file *(ObjectInputStream / ObjectOutputStream).*



* Khi người dùng click chuột chọn 1 dòng trong JTable, sẽ hiển thị thông tin của dòng được chọn trên các JTextFields.
* Add: Thêm 1 Country vào file
  + Không được trùng tên nước
  + Tên nước, và thủ đô có thể gồm nhiều từ ngăn cách bằng khoảng trắng, không chứa ký số hoặc các ký tự đặc biệt khác.
* Clear: Xóa rỗng các JTextFields, JComboBox có giá trị mặc định là false.
* Update: Có thể sửa dữ liệu trực tiếp trên các ô của Jtable, hoặc có thể chỉnh sửa dữ liệu trên các component của dòng được chọn.
* Delete: Xóa dòng dữ liệu được chọn. Hỏi nhắc người dùng trước khi xóa.
* Search: Tìm theo tên country, nhập dữ liệu từ hộp thoại input dialog.
* Tìm thấy nạp vào các JTextField, chỉ hiển thị dòng tìm thấy trên JTable. Nếu người dùng click nút clear, sẽ hiển thị lại toàn bộ thông tin lại trên JTable.
* Thông báo “Not found” nếu không tìm thấy.

*Ghi chú:*

1. *Thao tác Add, Update, Delete phải cập nhật lại file, cập nhật dữ liệu trên các component, hiển thị lại dữ liệu trên JTable*
2. *Chương trình phải thông báo các message cụ thể nếu người dùng gõ dữ liệu không phù hợp.*

**BÀI TẬP THỰC HÀNH TUẦN 5**

*Nội dung thực hành*

* *Thực hành Java Reflection*

**Bài 1:** Chuyển từ 1 class bất kỳ sang interface.

*Ví dụ, ta có class* ***Test\_Mixed*** *như sau:*

|  |
| --- |
| **public** **class** Test\_Mixed{    **private** **static** **final** **int** ***i*** = 32;  **public** **static** **final** **char** ***c*** = 'd';  **public** **final** **float** f = 3.1f;  **public** **static** **final** **double** ***bar*** = 3.14;  **public** **static** **final** **int**[] ***a*** = {3, 42, 9};  **public** **static** **final** **double**[] ***d*** = {3.34, 4.2, 9.7};  **public** **static** **final** String ***s*** = "testing\ntesting";  **public** **static** **final** **Double** od = **new** **Double**(4.2);  **public** **static** **final** **Character** oc = **new** **Character**('d');  **private** **int** x;    **public** Test\_Mixed() {  }  **public** Test\_Mixed(**int** x) {  **this**.x = x;  }  **private** **void** privateMethod() {}  **public** **static** **void** staticMethod(){}  **public** **void** nullMethod() {  }  **public** **int** simpleMethod(**double** d) {  **return** (**int**) d;  }  **public** **int** complexMethod(**int** i, **int** j, **double** d) {  **return** 0;  }  **public** **int** complexMethod(**char** c) **throws** IllegalAccessException, ArithmeticException {  **if**(c == 'd')  **throw** **new** IllegalAccessException("D-exception!");  **if**(c == 'x')  **throw** **new** ArithmeticException("A-exception");  **return** 0;  }  } |

Chuyển sang interface như sau:

|  |
| --- |
| **interface** ITest\_Mixed{  **int** ***i*** = 32;  **char** ***c*** = 'd';  **float** ***f*** = 3.1f;  **double** ***bar*** = 3.14;  **int**[] ***a*** = {3, 42, 9};  **double**[] ***d*** = {3.34, 4.2, 9.7};  String ***s*** = "testing\ntesting";  **Double** od = **new** **Double**(4.2);  **Character** oc = **new** **Character**('d');  **int** simpleMethod(**double** par1);  **int** complexMethod(**int** par1,**int** par2,**double** par3);  **int** complexMethod(**char** par1) **throws** IllegalAccessException, ArithmeticException;  **void** nullMethod();  } |

**Bài 2**:

Thực thi phương thức bất kỳ của một lớp.

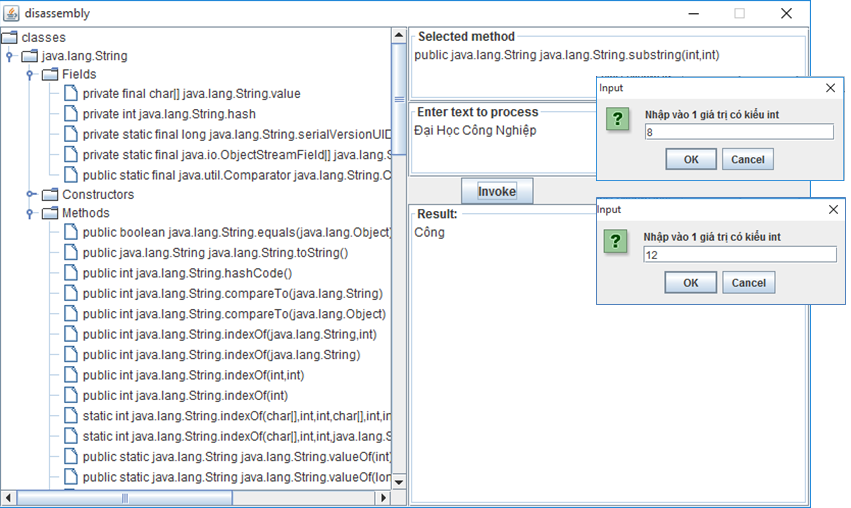
1. Thiết kế giao diện:

* Thêm bất kỳ class nào vào cây, với nút gốc là “classes”.
* Mỗi class sẽ chứa 3 nút con gồm các fields, constrcutors và các methods của class.
* Đọc cấu trúc của lớp, nạp tương ứng lên cây như trong hình.

1. Khi người dùng chọn bất kỳ 1 phương thức, click nút Invoke, sẽ tự động hiển thị lần lượt các JOptionPane tương ứng với số tham số của phương thức để người dùng nhập giá trị tương ứng với kiểu của từng tham số.

Nhập đủ tham số, sẽ thực thi phương thức và hiển thị kết quả.

*Ví dụ: Thêm lớp String vào tree, thực thi phương thức substring với 2 tham số kiểu int:*



**Bài 3:** *(Bài tập làm ở nhà)*

Thay vì dùng JOptionPane để nhập liệu cho các tham số. Ta dùng các JLabel và JTextFields thay thế. Khi người dùng chọn 1 phương thức, sẽ tự động hiển thị số component tương ứng với số tham số của phương phương thức được chọn.

Người dùng nhập giá trị tương ứng cho từng tham số. Click nút Invoke, thực thi phương thức và hiển thị kết quả.

**BÀI TẬP THỰC HÀNH TUẦN 6**

*Nội dung thực hành*

* *Thực hành Annotations*

**Bài 1:** Object/Relational Mapping *(mapping java class to table)*. Đọc class file, phát sinh tự động câu lệnh SQL để tạo các bảng tạo CSDL tương ứng.

**Primitive types:**

|  |  |  |
| --- | --- | --- |
| **Mapping type** | **Java type** | **ANSI SQL Type** |
| integer | int or java.lang.Integer | INTEGER |
| long | long or java.lang.Long | BIGINT |
| short | short or java.lang.Short | SMALLINT |
| float | float or java.lang.Float | FLOAT |
| double | double or java.lang.Double | DOUBLE |
| big\_decimal | java.math.BigDecimal | NUMERIC |
| character | java.lang.String | CHAR(1) |
| string | java.lang.String | VARCHAR |
| byte | byte or java.lang.Byte | TINYINT |
| boolean | boolean or java.lang.Boolean | BIT |
| yes/no | boolean or java.lang.Boolean | CHAR(1) ('Y' or 'N') |
| true/false | boolean or java.lang.Boolean | CHAR(1) ('T' or 'F') |

**Date and time types:**

|  |  |  |
| --- | --- | --- |
| **Mapping type** | **Java type** | **ANSI SQL Type** |
| date | java.util.Date or java.sql.Date | DATE |
| time | java.util.Date or java.sql.Time | TIME |
| timestamp | java.util.Date or java.sql.Timestamp | TIMESTAMP |
| calendar | java.util.Calendar | TIMESTAMP |
| calendar\_date | java.util.Calendar | DATE |

*Ví dụ:*

Java class

|  |
| --- |
| **public** **class** Employee {  **private** **int** id;  **private** String firstName;  **private** String lastName;  **private** **long** salary;  **public** Employee(**int** id, String firstName, String lastName, **long** salary) {  **this**.id = id;  **this**.firstName = firstName;  **this**.lastName = lastName;  **this**.salary = salary;  }  **public** **int** getId() {  **return** id;  }  **public** String getFirstName() {  **return** firstName;  }  **public** String getLastName() {  **return** lastName;  }  **public** **long** getSalary() {  **return** salary;  }  **public** **void** setSalary(**long** salary) {  **this**.salary = salary;  }  } |

Java class with annotations

|  |
| --- |
| @TableName  **public** **class** Employee {  @TableColumn(constraint = @Constraints(primaryKey = **true**),  type = SQLType.***INTEGER***)  **private** **int** id;  @TableColumn(constraint = @Constraints(allowNull = **false**),  name = "first\_name", type = SQLType.***NVARCHAR***, value = 30)  **private** String firstName;  @TableColumn(name = "last\_name", type = SQLType.***NVARCHAR***, value = 40)  **private** String lastName;  @TableColumn(type = SQLType.***BITINT***)  **private** **long** salary;    **public** Employee() {  **this**(0,"","",0l);  }  **public** Employee(**int** id, String firstName, String lastName, **long** salary) {  **this**.id = id;  **this**.firstName = firstName;  **this**.lastName = lastName;  **this**.salary = salary;  }  **public** **int** getId() {  **return** id;  }  **public** String getFirstName() {  **return** firstName;  }  **public** String getLastName() {  **return** lastName;  }  **public** **long** getSalary() {  **return** salary;  }  **public** **void** setSalary(**long** salary) {  **this**.salary = salary;  }  } |

RDBMS table:

|  |
| --- |
| Create table Employee (  id INTEGER primary key,  first\_name NVARCHAR(30) not null,  last\_name NVARCHAR(40),  salary BITINT  ); |

**Bài 2:** *(Bài tập làm ở nhà)*

Mở rộng bài 1 cho Association Mappings.

|  |
| --- |
| **Mapping type** |
| Many-to-One |
| One-to-One |
| One-to-Many |
| Many-to-Many |

**BÀI TẬP THỰC HÀNH TUẦN 7**

*Nội dung thực hành: Thực hành XML*

*Parsing XML document:*

* *Dùng SAX Parser*
* *Dùng DOM Parser*

Cho tài liệu XML có cấu trúc như sau:

|  |
| --- |
| <?xml version=*"1.0"* ?>  <productList>  <product id=*"TV1025"*>  <productName>TV Led</productName>  <manufacture>Sony</manufacture>  <description>TV led voi cong nghe</description>  <suplier>  <suplierName>Thiên Hòa</suplierName>  <country>Việt Nam</country>  <website>www.thienhoa.com</website>  </suplier>  <price>500.0</price>  </product>  <product id=*"Phone1002"*>  <productName>DT NOKIA lumia 630</productName>  <manufacture>nokia</manufacture>  <description>Điện thoại cảm ứng</description>  <suplier>  <suplierName>FBT</suplierName>  <country>Việt Nam</country>  <website>www.fbt.com</website>  </suplier>  <price>800.0</price>  </product>  <product id=*"IPhone5S"*>  <productName>Điện thoại IPhne 5S</productName>  <manufacture>Apple</manufacture>  <description>Điện thoại cảm ứng</description>  <suplier>  <suplierName>Thế giới di động</suplierName>  <country>Việt Nam</country>  <website>www.diDong.com</website>  </suplier>  <price>1200.0</price>  </product>  </productList> |

**Bài 1.** Đọc tập tin XML dùng DOM Parser

**Bài 2.** Dùng DOM parser sửa nội dung tập tin XML

* Thêm 1 element
* Cập nhật thuộc tính (đã có) của một element.
* Cập nhật giá trị (đã có) của một element.
* Xóa một thành phần

**Bài 3.** Đọc tập tin XML dùng SAX Parser

**Bài 4.** Sinh viên thêm vào vài sản phẩm nữa để thử

* Dùng SAX duyệt qua và in cấu trúc của file XML
* Dùng  StAX tìm sản phẩm có giá >600$.
* Dùng StAX thêm vào 1 sản phẩm mới
* Dùng DOM phân tích tài liệu và hiển thị các đối tượng lên 1 JTable. KHi người dung chọn 1 phần tử trong JTable sẽ hiển thị thong tin chi tiết lên các JTextField. Thêm vào các nút Thêm, Lưu, Xóa, Sửa và Tìm kiếm với các chức năng tương ứng.

**BÀI TẬP THỰC HÀNH TUẦN 8**

*Nội dung thực hành:*

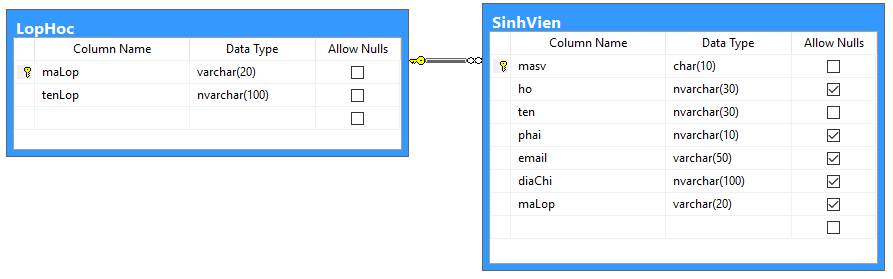
* *Kết nối CSDL*
* *Các câu lệnh xử lý dữ liệu JDBC : CRUD*
* *JDBC ResultSet*

**Bài 1:**

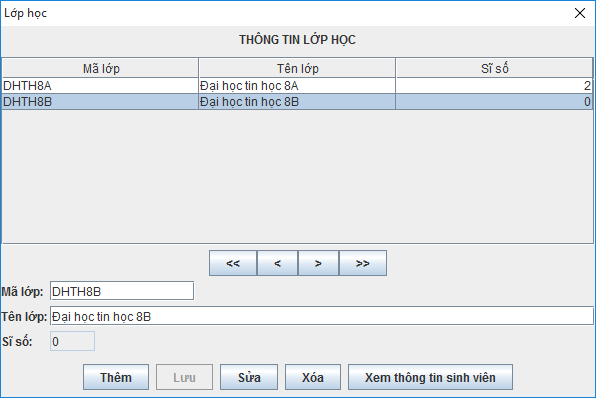
Làm các demontraction về kết nối CSDL và thực thi các phương thức của đối tượng Statement, PrepareStatement: executeQuery, executeUpdate, executeBatch

**Bài 2:**

Dùng hệ quản trị cơ sở dữ liệu SQL Server, tạo CSDL với các bảng sau:

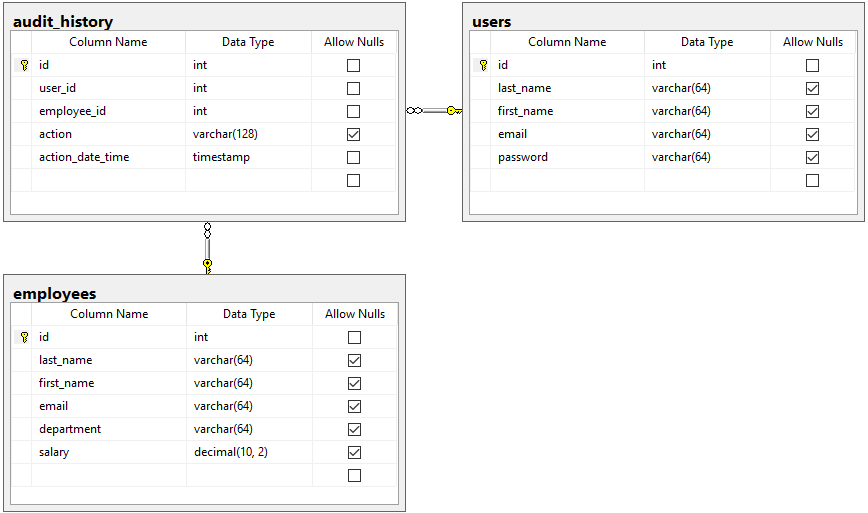


Tạo 1 project bằng ngôn ngữ lập trình java thực hiện các công việc sau:

* Tạo kết nối với CSDL khi JFrame được nạp lên.
* Đóng kết nối với CSDL khi người dùng đóng JFrame.
* Khi người dùng click chọn nút “Xem thông tin sinh viên”. Một cửa sổ tương tự xuất hiện hiển thị các sinh viên thuộc lớp hiện tại đang được chọn.

**Bài 3:** *(Bài tập về nhà)*

* Hệ quản trị CSDL SQL Server: *Chuẩn bị CSDL theo mô hình sau:*

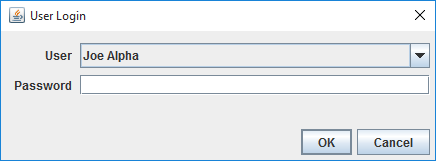


* Nhập liệu:

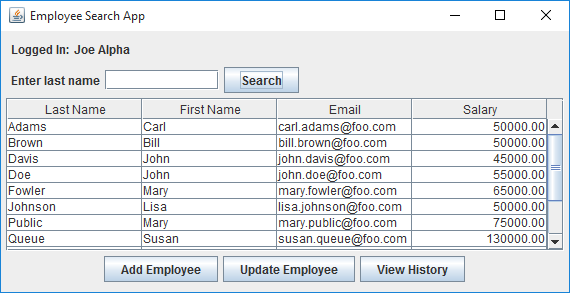
|  |
| --- |
| INSERT INTO employees VALUES (1,'Doe','John','john.doe@foo.com', 'HR', 55000.00);  INSERT INTO employees VALUES (2,'Public','Mary','mary.public@foo.com', 'Engineering', 75000.00);  INSERT INTO employees VALUES (3,'Queue','Susan','susan.queue@foo.com', 'Legal', 130000.00);  INSERT INTO employees VALUES (4,'Williams','David','david.williams@foo.com', 'HR', 120000.00);  INSERT INTO employees VALUES (5,'Johnson','Lisa','lisa.johnson@foo.com', 'Engineering', 50000.00);  INSERT INTO employees VALUES (6,'Smith','Paul','paul.smith@foo.com', 'Legal', 100000.00);  INSERT INTO employees VALUES (7,'Adams','Carl','carl.adams@foo.com', 'HR', 50000.00);  INSERT INTO employees VALUES (8,'Brown','Bill','bill.brown@foo.com', 'Engineering', 50000.00);  INSERT INTO employees VALUES (9,'Thomas','Susan','susan.thomas@foo.com', 'Legal', 80000.00);  INSERT INTO employees VALUES (10,'Davis','John','john.davis@foo.com', 'HR', 45000.00);  INSERT INTO employees VALUES (11,'Fowler','Mary','mary.fowler@foo.com', 'Engineering', 65000.00);  INSERT INTO employees VALUES (12,'Waters','David','david.waters@foo.com', 'Legal', 90000.00);  INSERT INTO users (id,last\_name,first\_name,email) VALUES (1,'Alpha','Joe','joe.alpha@foo.com');  INSERT INTO users (id,last\_name,first\_name,email) VALUES (2,'Beta','Jane','jane.beta@foo.com');  INSERT INTO users (id,last\_name,first\_name,email) VALUES (3,'Zeta','Becky','becky.zeta@foo.com'); |

Tạo 1 project bằng ngôn ngữ lập trình java thực hiện các công việc sau:

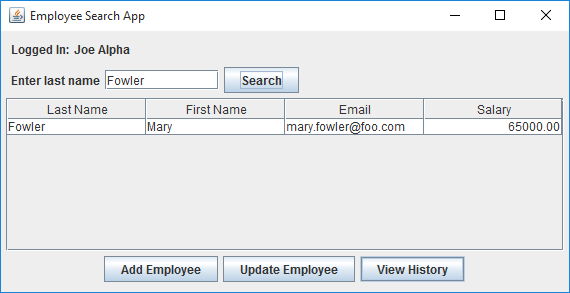
* Khi chạy chương trình: yêu cầu xác định usename và password



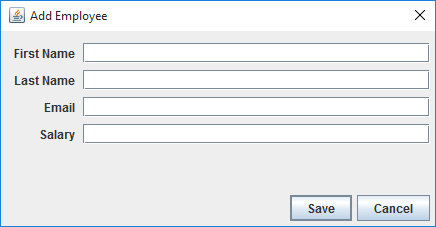
* Sau khi đăng nhập thành công:



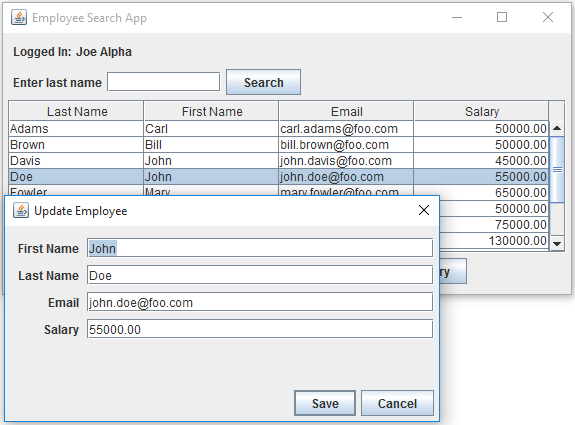
* Với nút “Search”:



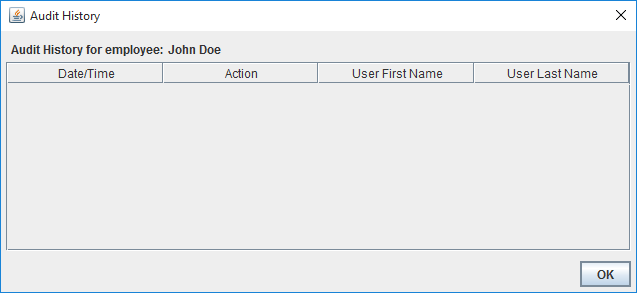
* Khi click nút “Add Employee”:



* Khi click nút “Update Employee”:



* Khi click nút “View History”:

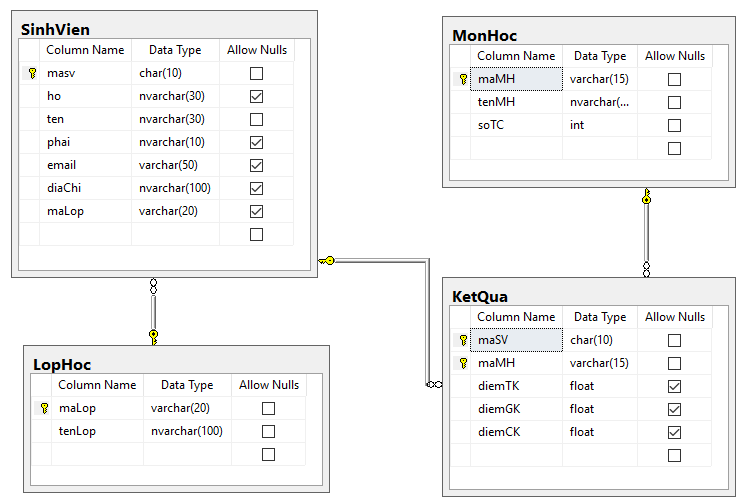


**BÀI TẬP THỰC HÀNH TUẦN 9**

*Nội dung thực hành:*

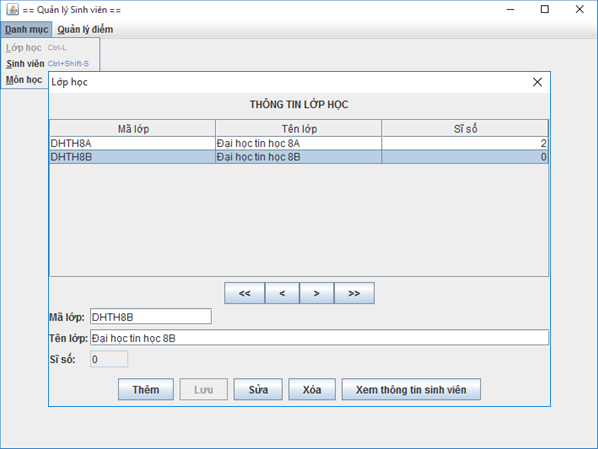
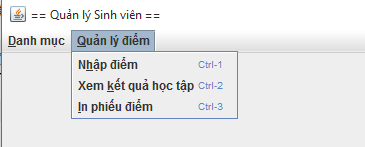
* *Swing Menu Components*
* *Using callable Statements*
* *Call Stored Procedures that take parameters.*
  + *No parameter*
  + *IN parameters*
  + *INOUT parameters*
  + *OUT parameters*
  + *Return a result set*

Mở rộng bài tập 2 của tuần 8. CSDL với các bảng sau:



Tạo 1 project bằng ngôn ngữ lập trình java thực hiện các công việc sau:

* Tạo kết nối với CSDL khi JFrame chính được nạp lên.
* Đóng kết nối với CSDL khi người dùng đóng JFrame này.
* Java Application, với giao diện chính sau:



**BÀI TẬP THỰC HÀNH TUẦN 10**

*Nội dung thực hành:*

* *Using Metadata*

**Bài 1:**

Viết chương trình cho phép kết nối vào 1 database của MS SQL server sau đó lấy danh sách các bảng hiển thị lên 1 JTree. Khi người dùng chọn 1 phần tử trên Jtree: nếu là Table thì hiển thị cấu trúc của nó đồng thời hiển thị dữ liệu. Nếu người dụng chọn 1 StoredProcedure thì hiển thị nội dung của câu sql tạo storedproc này. Viết 1 Form cho phép người dùng gõ vào 1 câu SQL và cho thực thi câu SQL này.

**Bài 2:** Làm bài kiểm tra thực hành.