BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC GIA ĐỊNH**

**KHOA: CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**TIỂU LUẬN**

**XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ HOÁ ĐƠN TIỀN ĐIỆN CỦA KHÁCH HÀNG**

**MÔN:** **[MẪU THIẾT KẾ CHO PHẦN MỀM](javascript:" \o "" \t "https://sinhvien.giadinh.edu.vn/)**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **KỸ THUẬT PHẦN MỀM**

Giảng viên hướng dẫn: **LÊ HUỲNH PHƯỚC**

Sinh viên thực hiện: **ĐẶNG NGUYỄN THANH SANG**

MSSV: **22150535**

Lớp: **221407**

TP. Hồ Chí Minh, ngày 1 tháng 8 năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC GIA ĐỊNH**

**KHOA: CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**TIỂU LUẬN**

**XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ HOÁ ĐƠN TIỀN ĐIỆN CỦA KHÁCH HÀNG**

**MÔN:** **[MẪU THIẾT KẾ CHO PHẦN MỀM](javascript:" \o "" \t "https://sinhvien.giadinh.edu.vn/)**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **KỸ THUẬT PHẦN MỀM**

Giảng viên hướng dẫn: **LÊ HUỲNH PHƯỚC**

Sinh viên thực hiện: **ĐẶNG NGUYỄN THANH SANG**

MSSV: **22150535**

Lớp: **221407**

TP. Hồ Chí Minh, ngày 1 tháng 8 năm 2024

**Khoa/Viện: Công Nghệ Thông Tin**

**NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN**

**TIỂU LUẬN MÔN: [MẪU THIẾT KẾ CHO PHẦN MỀM](javascript:" \o "" \t "https://sinhvien.giadinh.edu.vn/)**

1. **Họ và tên sinh viên:** Đặng Nguyễn Thanh Sang
2. **Tên đề tài:** “XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ HOÁ ĐƠN TIỀN ĐIỆN CỦA KHÁCH HÀNG”.
3. **Nhận xét**:

***Những kết quả đạt được:***

***Những hạn chế:***

1. **Điểm đánh giá** *(theo thang điểm 10, làm tròn đến 0.5):*

Sinh viên: Đặng Nguyễn Thanh Sang

Điểm số: ……….…… Điểm chữ:

|  |  |
| --- | --- |
|  | *TP. HCM, ngày 1 tháng 8 năm 20….*  **Giảng viên chấm thi**  *(Ký và ghi rõ họ tên)* |

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn sâu sắc và được bày tỏ lòng biết ơn đến với quý Thầy Cô của khoa Công Nghệ Thông Tin tại Trường Đại học Gia Định và đặc biệt đối với thầy **Lê Huỳnh Phước** đã tận tình hướng dẫn, đồng hành và chỉ bảo giúp em hoàn thành tốt bài báo cáo này cũng như truyền đạt những kiến thức bổ ích, rất quan trọng đối với quá trình học tập tại trường và đi làm của em sau này. Trong thời gian nghiên cứu đề tài tiểu luận vừa qua, em đã có nhiều cố gắng trong suốt quá trình thực hiện tiểu luận, tích cực trao đổi thông tin, sưu tầm, tham khảo tài liệu và học hỏi thêm kiến thức từ các Thầy Cô và các học viên khác, không chỉ nhận được những kiến thức đầy bổ ích về chuyên môn mà còn ở những lĩnh vực khác. Những trải nghiệm quý báu đó không chỉ giúp em hoàn thành tốt bài báo cáo mà còn là hành trang quan trọng theo em trong suốt thời gian học tập và giúp em có thể tự tin bước vào đời làm việc sau này. Do kiến thức và kinh nghiệm còn hạn chế cho nên bài báo cáo của em còn nhiều thiếu sót, kính mong được sự đánh giá, góp ý của thầy. Cuối cùng, xin kính chúc thầy luôn luôn khỏe mạnh, vui vẻ và đạt được nhiều thành công cao trong công tác giảng dạy. Chúc trường Đại học Gia Định sẽ luôn là nền tảng vững chắc cho nhiều thế hệ sinh viên tiếp bước trên con đường học tập.

**Em xin chân thành cảm ơn!**

Đặng Nguyễn Thanh Sang

**LỜI NÓI ĐẦU**

Với sự phát triển không ngừng, công nghệ thông tin ngày nay đã len lỏi vào hầu hết mọi lĩnh vực trong đời sống. Trong đó, công nghệ phần mềm ngày càng được ứng dụng nhiều hơn trong việc quản lý cuộc sống thường ngày của chúng ta.Nắm bắt được nhu cầu ứng dụng CNTT để quản lý trong cuộc sống, áp dụng những gì đã học trong học phần Lập trình Java, em dưới sự giúp đỡ của thầy Lê Huỳnh Phước đã quyết định tìm hiểu và xây dựng một phần mềm đơn giản và gần gũi với thực tế là Phần mềm quản lý biên lai thu tiền điện.Phần mềm hỗ trợ quản lý biên lai thu tiền điện được nhóm em xây dựng nhằm hỗ trợ người quản lý có thể dễ dàng cập nhập tình hình tiêu thụ điện năng và quản lý hoá đơn tiền điện tránh việc phải tính toán thủ công mất nhiều thời gian cũng như khó khăn trong việc lưu trữ và xử lý thông tin.Bài tập lớn này được hoàn thành dưới sự đóng góp của các thành viên trong nhóm và sự hỗ trợ của giảng viên bộ môn Lập trình Java. Xong trong quá trình nghiên cứu và hoàn thành bài tập lớn không tránh khỏi thiếu sót, rất nhận được đóng góp của thầy để đề tài của chúng em được đầy đủ và hoàn chỉnh hơn.Chúng em xin chân thành cảm ơn!

**MỤC LỤC**

**[CHƯƠNG 1 : LẬP KẾ HOẠCH THỰC HIỆN PHẦN MỀM 1](#_Toc20755)**

[1.1. Lập kế hoạch 1](#_Toc23653)

[1.1.1. Khởi tạo 1](#_Toc14153)

[1.1.2. Phân tích tính khả thi 1](#_Toc26882)

**[CHƯƠNG 2 : LÝ THUYẾT VỀ MẪU THIẾT KẾ 2](#_Toc8667)**

[2.1. Định nghĩa 2](#_Toc3506)

[2.2. Design Patterns hỗ trợ developers như thế nào 2](#_Toc17853)

[2.2.1. Tăng tốc độ phát triển phần mềm 2](#_Toc11993)

[2.2.2. Code tường minh giúp dễ dàng teamwork 2](#_Toc29688)

[2.2.3. Tái sử dụng code 3](#_Toc18102)

[2.2.4. Hạn chế lỗi tiềm ẩn, dễ dàng nâng cấp 3](#_Toc15838)

[2.3. Phân loại Design Patterns 3](#_Toc18732)

[2.3.1. Observer pattern 3](#_Toc22559)

[2.3.2. Singleton 4](#_Toc20262)

[2.3.3. MVC 4](#_Toc6946)

[2.3.3.1. Model 5](#_Toc3561)

[2.3.3.2. View 6](#_Toc13204)

[2.3.3.3. Controller 6](#_Toc516)

**[CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ CHƯƠNG TRÌNH 7](#_Toc27663)**

[3.1. Phân tích chương trình 7](#_Toc12999)

[3.2. Thiết kế chương trình 7](#_Toc320)

[3.2.1. Lớp Customer (Khách hàng) 7](#_Toc23142)

[3.2.2. Lớp CustomerDao (Data Access Object cho Khách hàng) 7](#_Toc15350)

[3.2.3. Lớp CustomerController (Bộ điều khiển Khách hàng) 8](#_Toc24621)

[3.2.4. Lớp CustomerView (Giao diện Khách hàng) 8](#_Toc10500)

[3.2.5. Lớp DbConnect (Kết nối CSDL) 8](#_Toc15738)

[3.2.6. Lớp Invoice (Hóa đơn) 9](#_Toc17629)

[3.2.7. Lớp InvoiceDao (Data Access Object cho Hóa đơn) 9](#_Toc27473)

[3.2.8. Lớp InvoiceController (Bộ điều khiển Hóa đơn) 9](#_Toc24909)

[3.2.9. Lớp InvoiceView (Giao diện Hóa đơn) 10](#_Toc9671)

[3.2.10. Lớp MainView (Giao diện chính) 10](#_Toc32189)

[3.2.11. Lớp MainViewController (Bộ điều khiển giao diện chính) 10](#_Toc31645)

[3.3. Sơ đồ lớp (Class Diagram) 11](#_Toc20200)

[3.4. Sơ đồ cơ sở dữ liệu (MySQL) 12](#_Toc5045)

**[CHƯƠNG 4 : CÀI ĐẶT THỰC NGHIỆM VÀ KẾT QUẢ CỦA CHƯƠNG TRÌNH 13](#_Toc22766)**

**[CHƯƠNG 5 : KẾT LUẬN VÀ ĐÁNH GIÁ 23](#_Toc8022)**

[5.1. Kết luận 23](#_Toc22296)

[5.2. Đánh giá và cải tiến trong tương lai 23](#_Toc26448)

**[TÀI LIỆU THAM KHẢO 24](#_Toc18143)**

**[PHỤ LỤC 25](#_Toc4991)**

# CHƯƠNG 1 : LẬP KẾ HOẠCH THỰC HIỆN PHẦN MỀM

## 1.1. Lập kế hoạch

### 1.1.1. Khởi tạo

Phần mềm quản lý hoá đơn tiền điện giúp nâng cao hiệu quả quản lý và tránh được rủi ro sai số do tính toán cũng như tiết kiệm nhiều chi phí cho việc quản lý, tránh được rủi ro như trường hợp “Vụ ghi nhầm tiền điện tăng 30 lần: ‘Sai sót rất đáng tiếc’” trên trang Báo Tuổi Trẻ. Nhóm sinh viên chúng em cùng nhau xây dựng phần mềm quản lý biên lai thu tiền điện để hỗ trợ người quản lý và nhân viên của họ quản lý tiền điện một cách hiện đại chuyên nghiệp. Do số lượng các hộ đăng ký sử dụng điện lớn, cùng với phép tính tiền điện theo bậc phức tạp, nếu quản lý thủ công sẽ tốn nhiều công sức và nhiều khi không đảm bảo được chính xác. Việc dùng phần mềm trong quản lý sẽ giúp đỡ rất nhiều trong việc quản lý kinh doanh.Mục tiêu là hệ thống giúp quản lý dễ dàng các thông tin khách hàng tiêu thụ điện và cách tính tiền điện và in hoá đơn tiêu thụ một cách chính xác, nhanh chóng. Hệ thống dễ sử dụng và tiết kiệm thời gian, giao diện phù hợp người dùng.

### 1.1.2. Phân tích tính khả thi

Phần mềm được phát triển dựa trên ngôn ngữ lập trình Java và hệ quản trị cơ sở dữ liệu MySQL là những mã nguồn mở miễn phí hỗ trợ mạnh mẽ, ít tốn tài nguyên và giúp tiết kiệm chi phí. Được hỗ trợ đa nền tảng và độc lập với nền nên có thể chạy trên nhiều hệ điều hành khác nhau. Phần mềm sau khi hoàn thành sẽ được người quản lý tiền điện, nhân viên ghi số điện, kế toán, người thu đơn lẻ,… sử dụng dễ dàng và hiệu quả.

# CHƯƠNG 2 : LÝ THUYẾT VỀ MẪU THIẾT KẾ

## 2.1. Định nghĩa

Mẫu thiết kế (Design pattern) là các giải pháp tổng thể đã được tối ưu hóa, được tái sử dụng cho các vấn đề phổ biến trong thiết kế phần mềm mà chúng ta thường gặp phải hàng ngày. Design patterns là một kỹ thuật trong lập trình hướng đối tượng, không phải là ngôn ngữ cụ thể nào cả, nó là một kỹ thuật lập trình. Nó có thể thực hiện được ở phần lớn các ngôn ngữ lập trình, chẳng hạn như Java, C#, thậm chí là Javascript hay bất kỳ ngôn ngữ lập trình nào khác. Nó khá quan trọng và mọi lập trình viên muốn giỏi đều phải biết, nó cung cấp cho chúng ta các “mẫu thiết kế”, giải pháp để giải quyết các vấn đề chung, thường gặp trong lập trình. Các vấn đề mà chúng ta gặp phải có thể sẽ tự nghĩ ra cách giải quyết nhưng có thể nó chưa phải là tối ưu. Design Pattern giúp chúng ta giải quyết vấn đề một cách tối ưu nhất, cung cấp các giải pháp trong lập trình OOP. Tuy nhiên, nếu là người mới tìm hiểu về lập trình, chưa nắm rõ kiến thức về OOP thì giờ chưa phải lúc để tìm hiểu sâu về Design Pattern. Muốn học Design Pattern, chúng ta cần phải có kiến thức vững chắc về lập trình, đặc biệt là lập trình OOP. Hiểu và áp dụng được kiến thức về OOP sẽ giúp chúng ta tiếp cận và nâng cao trình độ với Design Pattern dễ dàng và hiệu quả hơn.

## 2.2. Design Patterns hỗ trợ developers như thế nào

### 2.2.1. Tăng tốc độ phát triển phần mềm

Trong quá trình phát triển ứng dụng, việc sử dụng design patterns cho phép các developers có một công cụ để giải quyết các vấn đề thông dụng trong thiết kế phần mềm. Kể cả khi không gặp phải những vấn đề đó, việc nắm vững design patterns cũng rất hữu ích khi nó giúp developers thấy được cách giải quyết vấn đề thông qua việc ứng dụng các nguyên tắc thiết kế hướng đối tượng.

### 2.2.2. Code tường minh giúp dễ dàng teamwork

Bên cạnh đó, design patterns định nghĩa một ngôn ngữ chung mà developers có thể dùng để giao tiếp hiệu quả hơn. Chẳng hạn, chỉ cần nêu tên một pattern, tất cả những đồng đội trong nhóm đều sẽ hình dung được cấu trúc, ý tưởng đằng sau đó và cách ứng dụng nó. Tối ưu thời gian phát triển ý tưởng hơn vì hạn chế được thời gian giải thích.

### 2.2.3. Tái sử dụng code

Về phía dự án phần mềm, design patterns giúp developers có thể dễ dàng tái sử dụng và mở rộng code với các giải pháp tối ưu đã được kiểm chứng để giải quyết những vấn đề thông thường trong phát triển phần mềm. Do đó, khi gặp vấn đề trong xây dựng phần mềm, developers có thể coi design patterns như là kim chỉ nam để giúp mình giải quyết những vấn đề thay vì tự đi tìm kiếm giải pháp (mà có thể chưa có sự kiểm chứng kỹ càng).

### 2.2.4. Hạn chế lỗi tiềm ẩn, dễ dàng nâng cấp

Ngoài ra, việc sử dụng lại các design patterns còn giúp developers tránh các vấn đề tiềm ẩn có thể sẽ gây ra những lỗi lớn trong tương lai, cùng với đó, điều đó cũng giúp dự án dễ nâng cấp và bảo trì trong tương lai hơn.

## 2.3. Phân loại Design Patterns

Nhóm chúng em sẽ đưa ra ba mẫu thiết kế mà nhóm chúng em thấy rằng rất quan trọng và thường được sử dụng trong việc phát triển. Đó là :

* Observer
* MVC
* Singleton

### 2.3.1. Observer pattern

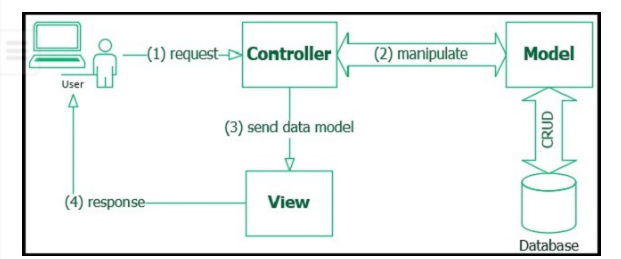
Observer pattern là một mẫu thiết kế phần mềm mà một đối tượng, gọi là subject, duy trì một danh sách các thành phần phụ thuộc nó, gọi là observer, và thông báo tới chúng một cách tự động về bất cứ thay đổi nào, thường thì bằng cách gọi 1 hàm của chúng. Observer Pattern được áp dụng khi: Sự thay đổi trạng thái ở 1 đối tượng có thể được thông báo đến các đối tượng khác mà không phải giữ chúng liên kết quá chặt chẽ.

### 2.3.2. Singleton

Singleton pattern trong Java là gì ? Singleton là 1 trong 5 design pattern của nhóm Creational Design Pattern Singleton đảm bảo chỉ duy nhất một thể hiện (instance) được tạo ra và nó sẽ cung cấp cho bạn một method để có thể truy xuất được thể hiện duy nhất đó mọi lúc mọi nơi trong chương trình. Một số nguyên tắc về Singleton Pattern trong Java. Singleton hạn chế việc khởi tạo đối tượng từ lớp và đảm bảo chỉ có một đối tượng của lớp tồn tại trong máy ảo JVM. Lớp singleton phải cung cấp một điểm truy cập mở để có thể lấy ra đối tượng của lớp. Singleton pattern được sử dụng để ghi log, đối tượng driver, lưu cache và xử lý threadpool. Singleton design pattern cũng được sử dụng trong các pattern khác như Abstract Factory, Builder, Prototype, Facade, etc Singleton design pattern được sử dụng trong các lớp core Java (ví dụ, java.lang.Runtime, java.awt.Desktop).

### 2.3.3. MVC

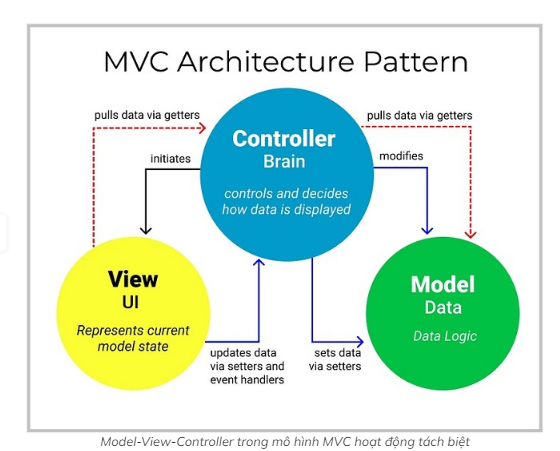
MVC là mô hình phần mềm dùng để tạo lập giao diện người dùng trên máy tính. Mô hình MVC bao gồm ba bộ phận chính là Model, View và Controller. Mỗi thành phần có chức năng, nhiệm vụ khác nhau nhưng giữa chúng có sự tương tác qua lại, hỗ trợ nhau. Trong đó Model có chức năng quản lý và xử lý dữ liệu View có nhiệm vụ hiển thị dữ liệu cho người dùng. Controller có chức năng điều khiển tương tác giữa Model và View, mô hình MVC trong Java giúp lập trình viên dễ dàng tách biệt giữa cách thức dữ liệu nội hàm với dữ liệu hiển thị. Sự tương tác qua lại giữa ba thành tố Model, View, Controller tạo nên hiệu quả tốt nhất cho việc lập trình. Mẫu MVC cũng là mẫu được chúng em sử dụng trong đề tài “Xây dựng chương trình quản lý hoá đơn tiền điện của khách hàng”.



**Hình 2.1.Sơ đồ mô hình MVC**

#### 2.3.3.1. Model

Một trong những thành phần quan trọng nhất của mô hình MVC trong Java. Đây là bộ phận làm nhiệm vụ quản lý dữ liệu. Model có chức năng vận chuyển thông tin từ nội hàm để hiển thị đến người dùng thông qua màn hình và xử lý các thông tin để người dùng dễ dàng tiếp cận nhất. Model hoàn toàn độc lập với các thành phần còn lại trong MVC và nó chứa các tác vụ cần thiết nhất cho quá trình lập trình.



**Hình 2.2.Mô hình kiến trúc MVC**

#### 2.3.3.2. View

Thành phần tiếp theo chúng ta sẽ nhắc đến ở mô hình MVC trong Java, đó là View. Đối với người dùng thì View có vai trò thiết yếu. Nó thực hiện nhiệm vụ tạo tương tác với người dùng và hiển thị các kết quả từ tầng Controller. Đồng thời, View cũng thực hiện việc tiếp nhận các hoạt động, yêu cầu của người dùng để chuyển đến Controller xử lý.

#### 2.3.3.3. Controller

Khi nhắc tới các thành phần ở mô hình MVC nhất định không thể bỏ qua Controller. Nếu không có thành phần này thì mọi hoạt động của Model hay View đều không còn giá trị Controller thực hiện chức năng kết nối tương tác giữa View và Model. Nó định nghĩa các lệnh và thực hiện xử lý các lệnh trong hệ thống. Controller đối chiếu hành động của người dùng từ View và tương tác với Model để chuyển tải thông tin cần thiết đến người dùng.

# CHƯƠNG 3 : PHÂN TÍCH VÀ THIẾT KẾ CHƯƠNG TRÌNH

## 3.1. Phân tích chương trình

Chương trình “Quản lý hóa đơn tiền điện” được sử dụng theo mẫu MVC được xây dựng trong ứng dụng vscode và kết nối với database trong MySQL nhằm mục đích có thể lưu trữ, thêm, sửa hoặc xóa khách hàng đó. Không những vậy chúng em còn thêm chức năng in hóa đơn và tìm kiếm hóa đơn.

## 3.2. Thiết kế chương trình

### 3.2.1. Lớp Customer (Khách hàng)

Thuộc tính:

* `String customerId`
* `String quota`
* `String nationality`
* `String customerName`
* `String customerType`

Phương thức:

* `toString() : String`

### 3.2.2. Lớp CustomerDao (Data Access Object cho Khách hàng)

Thuộc tính:

* `List<String> allCustomerIds`
* `List<Customer> allCustomers`

Phương thức:

* `addCustomer(Customer) : void`
* `getQuotaByCusId(String) : String`
* `deleteCustomer(Customer) : void`
* `isExistCustomer(String) : Boolean`
* `updateCustomer(Customer) : void`

### 3.2.3. Lớp CustomerController (Bộ điều khiển Khách hàng)

Phương thức:

* `updateCustomerTable() : void`
* `showFrm() : void`

### 3.2.4. Lớp CustomerView (Giao diện Khách hàng)

Thuộc tính:

* `ActionListener btnRefreshListener`
* `String keyword`
* `ActionListener btnDeleteListener`
* `ActionListener btnSearchListener`
* `Boolean foreign`
* `boolean Vietnamese`
* `ActionListener btnUpdateListener`
* `ActionListener btnAddListener`
* `Customer customer`
* `List<Customer> dataToTable`

Phương thức:

* `tbCustomersMouseClicked(MouseEvent) : void`
* `addFilterByCustomerListener(ActionListener) : void`
* `status() : void`
* `clear() : void`
* `initComponents() : void`

### 3.2.5. Lớp DbConnect (Kết nối CSDL)

Thuộc tính:

* `DbConnect instance`

Phương thức:

* `createPreparedStatement(String, Object[]) : PreparedStatement`
* `executeUpdate(String, Object[]) : int`
* `executeQuery(String, Object[]) : ResultSet`

### 3.2.6. Lớp Invoice (Hóa đơn)

Thuộc tính:

* `double total`
* `LocalDate invoiceDate`
* `int quantity`
* `String customerId`
* `double unitPrice`

Phương thức:

* `toString() : String`
* `main(String[]) : void`

### 3.2.7. Lớp InvoiceDao (Data Access Object cho Hóa đơn)

Thuộc tính:

* `List<Invoice> allInvoices`

Phương thức:

* `updateInvoice(Invoice) : void`
* `addInvoice(Invoice) : void`
* `deleteInvoice(Invoice) : void`
* `getInvoiceByCusId(Invoice) : Map<String, String>`
* `searchInvoices(String, Integer, Integer) : List<Invoice>`

### 3.2.8. Lớp InvoiceController (Bộ điều khiển Hóa đơn)

Phương thức:

* exportInvoice(String, String, String, String, String) : void
* readFile(String) : String
* calculateTotalAmount(int, double, String) : double
* showFrm() : void

### 3.2.9. Lớp InvoiceView (Giao diện Hóa đơn)

Thuộc tính:

* `Invoice invoice`
* `ActionListener btnExportAction`
* `Optional<String> cusIdSearch`
* `Optional<integer> month`
* `ActionListener btnDeleteAction`
* `ActionListener btnAddAction`
* `ActionListener btnSearchAction`
* `List<String> cbCusId`
* `ActionListener btnRefreshAction`
* `int year`
* `List<Invoice> dataToTable`

Phương thức:

* `initComponents() : void`
* `clear() : void`
* `btnInvoicesMouseClicked(MouseEvent) : void`

### 3.2.10. Lớp MainView (Giao diện chính)

Thuộc tính:

* `ActionListener btnInvoicesActionListener`
* `ActionListener btnCustomersActionListener`

Phương thức:

* `initComponents() : void`

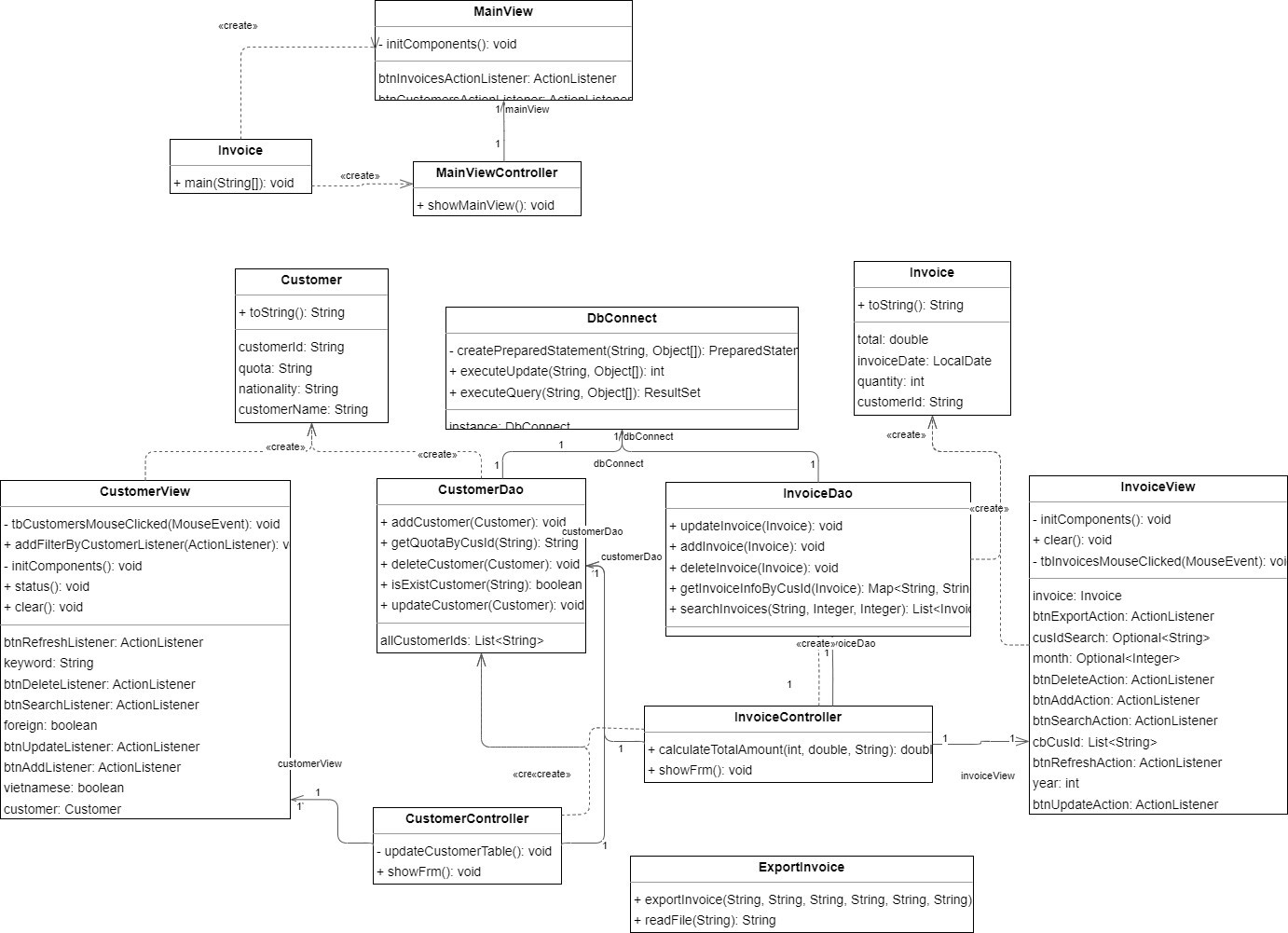
### 3.2.11. Lớp MainViewController (Bộ điều khiển giao diện chính)

Phương thức:

* `showMainView() : void`

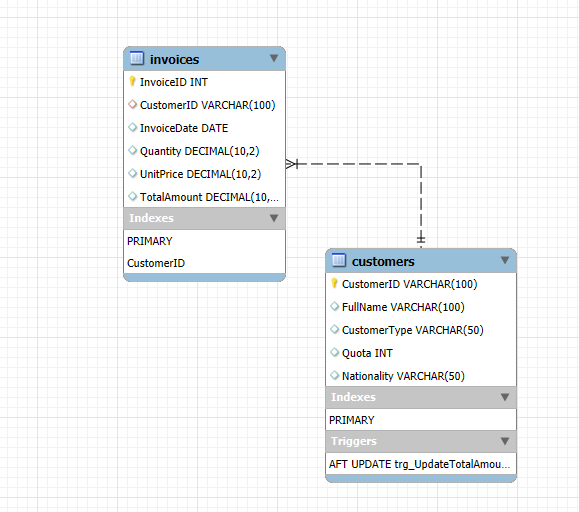
Mối quan hệ giữa các lớp: CustomerView tạo ra CustomerDao, CustomerController và sử dụng DbConnect để thực hiện các thao tác với cơ sở dữ liệu. InvoiceView tạo ra InvoiceDao, InvoiceController và sử dụng DbConnect để thực hiện các thao tác với cơ sở dữ liệu. CustomerController và InvoiceController sử dụng CustomerDao và InvoiceDao để quản lý dữ liệu khách hàng và hóa đơn. MainViewController tạo ra MainView để hiển thị giao diện chính của ứng dụng. Các lớp Controller và View kết hợp với nhau để thực hiện các thao tác trên giao diện người dùng và cập nhật dữ liệu trong cơ sở dữ liệu thông qua các lớp Dao.

## 3.3. Sơ đồ lớp (Class Diagram)



**Hình 3.1.Sơ đồ lớp**

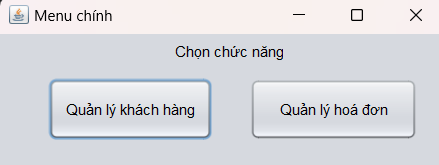
## 3.4. Sơ đồ cơ sở dữ liệu (MySQL)



**Hình 3.2.Sơ đồ cơ sở dữ liệu**

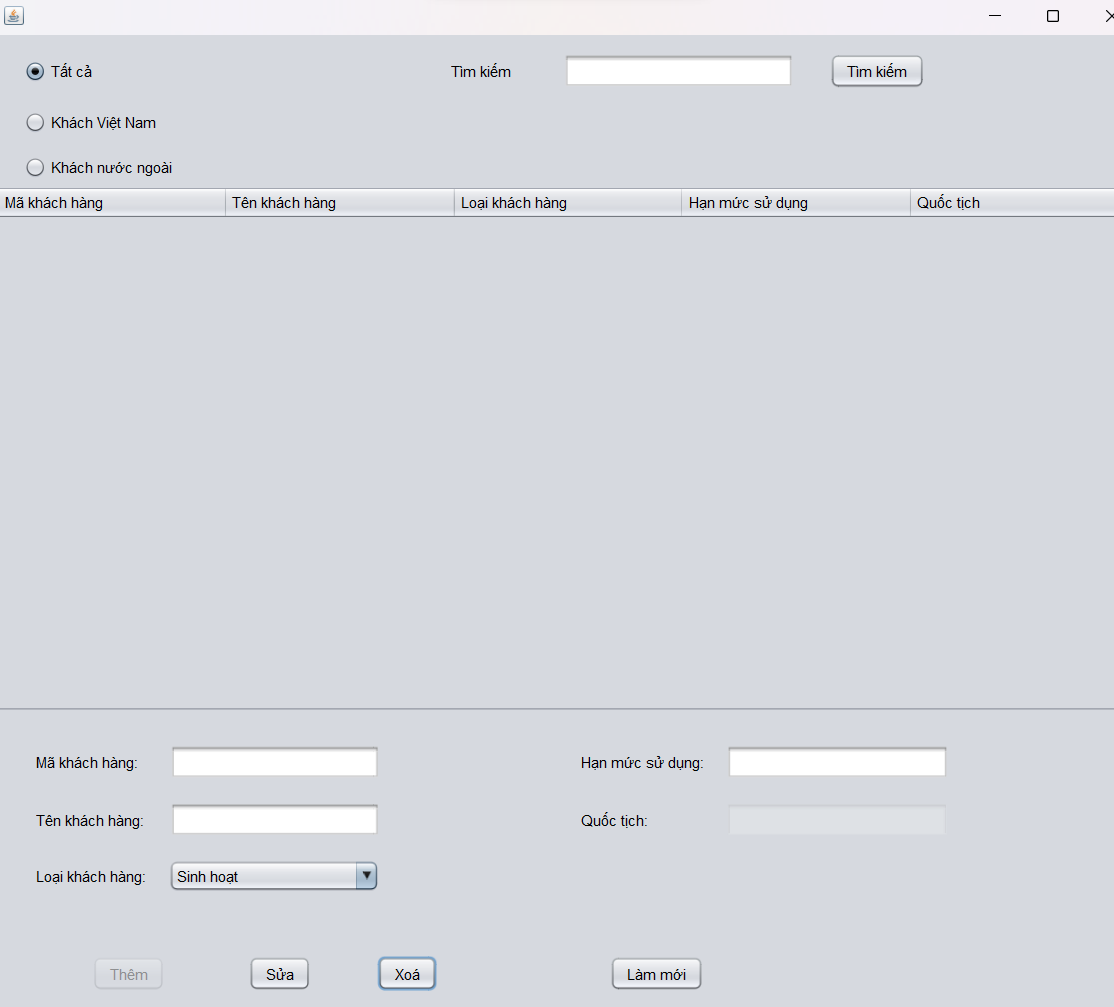
# CHƯƠNG 4 : CÀI ĐẶT THỰC NGHIỆM VÀ KẾT QUẢ CỦA CHƯƠNG TRÌNH

Sau khi hoàn thiện về mặt ý tưởng cũng như thực nghiệm thì chúng em đã cho ra một chương trình gần như hoàn thiện so với yêu cầu mà giảng viên đưa ra. Đầy đủ chức năng thêm, sửa, xóa, cũng như là phân chia User Việt Nam và nước ngoài. Dưới đây sẽ là giao diện và các chức năng của chương trình được thể hiện một cách thực tế.

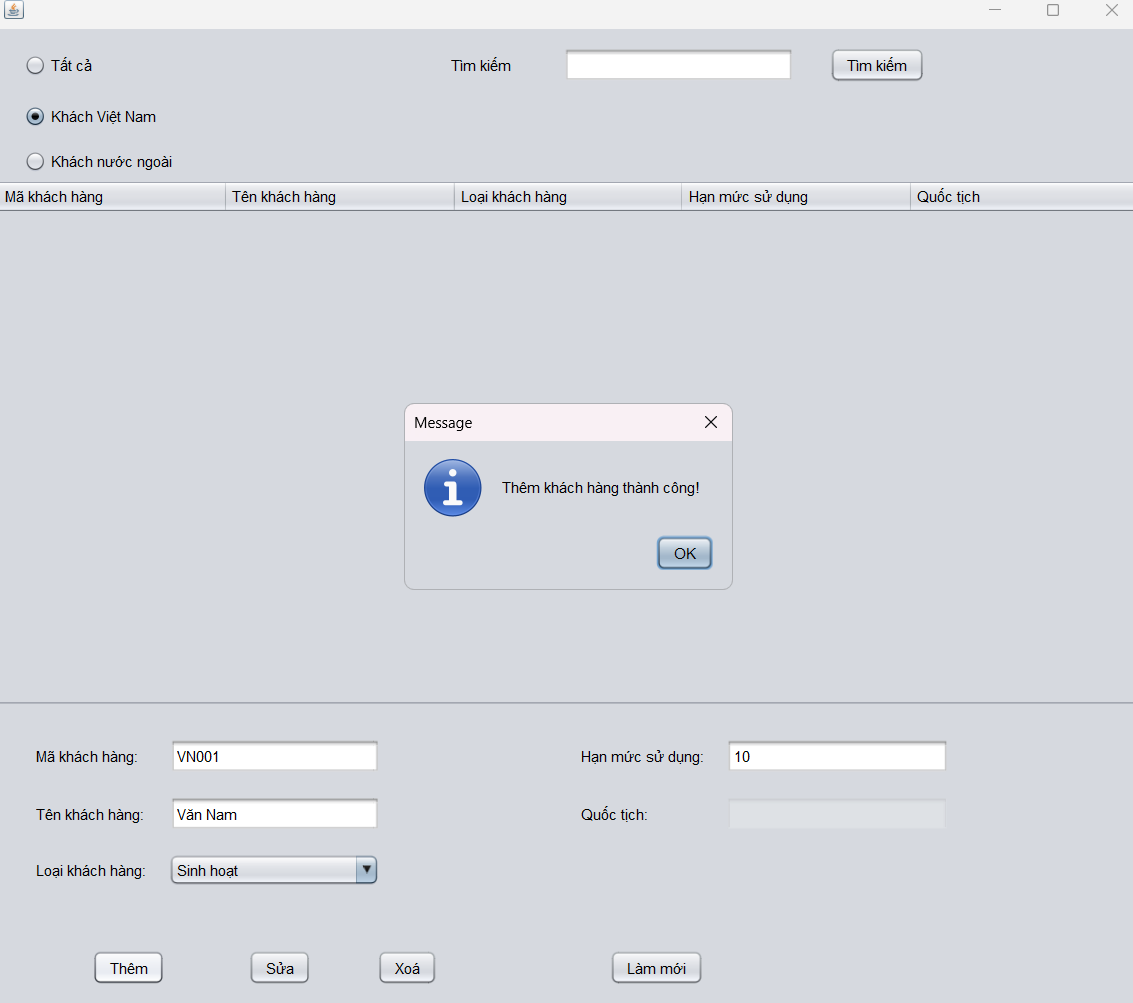


**Hình 4.1.Giao diện Menu chính**

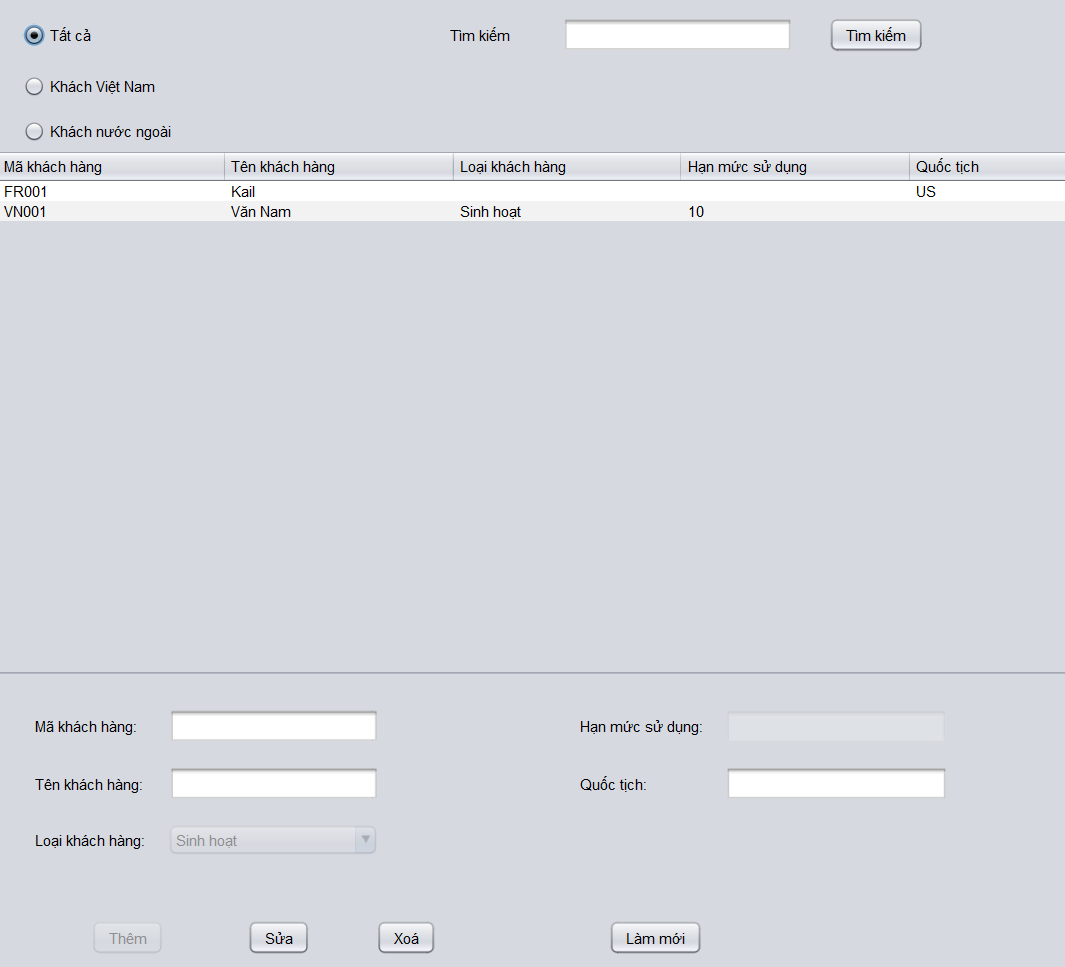
Sau khi nhấp vào “Run”, chương trình sẽ hiển thị hai option để ta lựa chọn đó là ‘Quản Lý Khách Hàng’ và ‘Quản Lý Hóa Đơn’. Chức năng đầu tiên chúng em sẽ thể hiện ra là chức năng quản lý khách hàng. Dưới đây là giao diện cơ bản mà chúng em thiết kế để thực hiện chức năng quản lý khách hàng. Ở góc trái chúng em đã thiết kế để quản lý có thể lựa chọn loại khách hàng mong muốn như khách Việt Nam hoặc Khách nước ngoài. Khách Việt Nam: sẽ phải đăng ký những mục như mã, tên, hạn mức sử dụng và loại khách hàng. Khách nước ngoài: Cũng như khách Việt Nam nhưng sẽ phải đăng ký quốc tịch. Chúng em cũng đã thêm chức năng tìm kiếm bằng ID khách hàng sau khi hoàn thành đăng ký.



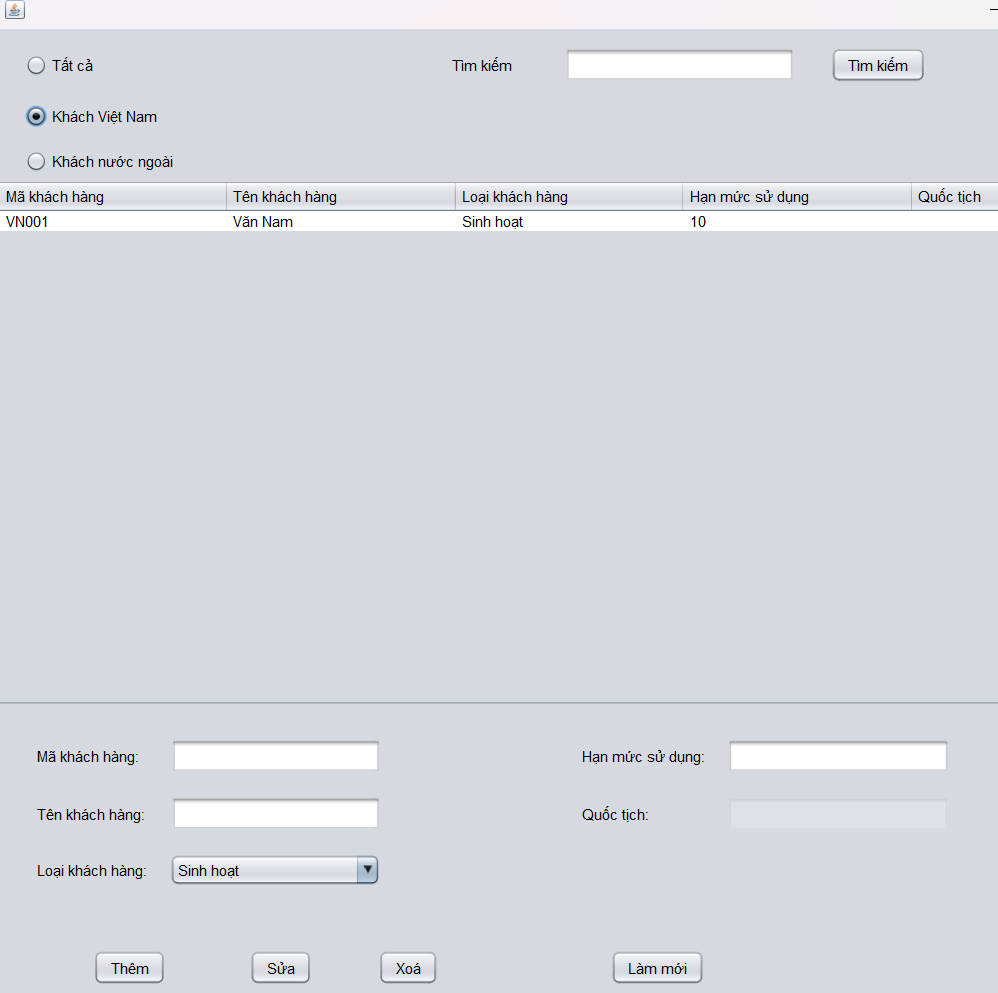
**Hình 4.2.Giao diện UI quản lý khách hàng**



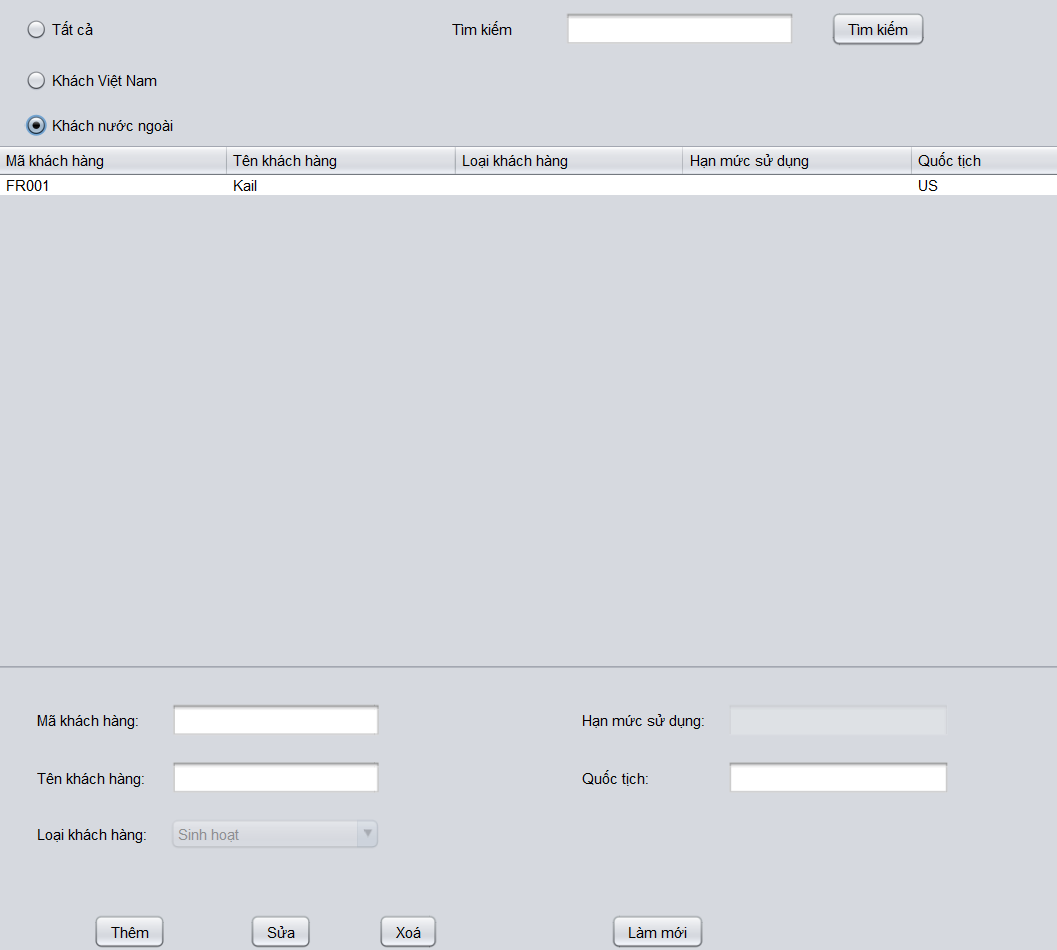
**Hình 4.3.Hiển thị sau khi thêm khách hàng thành công**



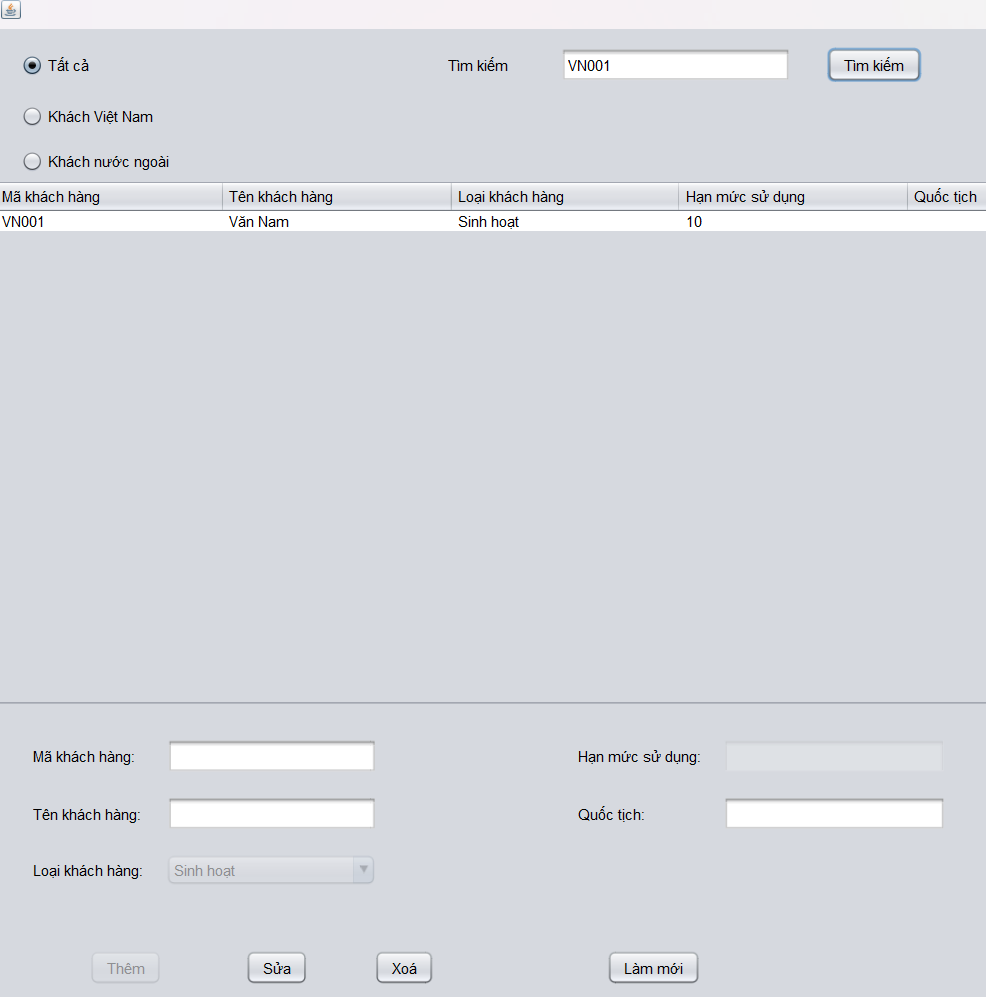
**Hình 4.4.Hiển thị sau khi đã đăng ký khách hàng thành công**



**Hình 4.5.Hiển thị lọc ra khách hàng Việt Nam**

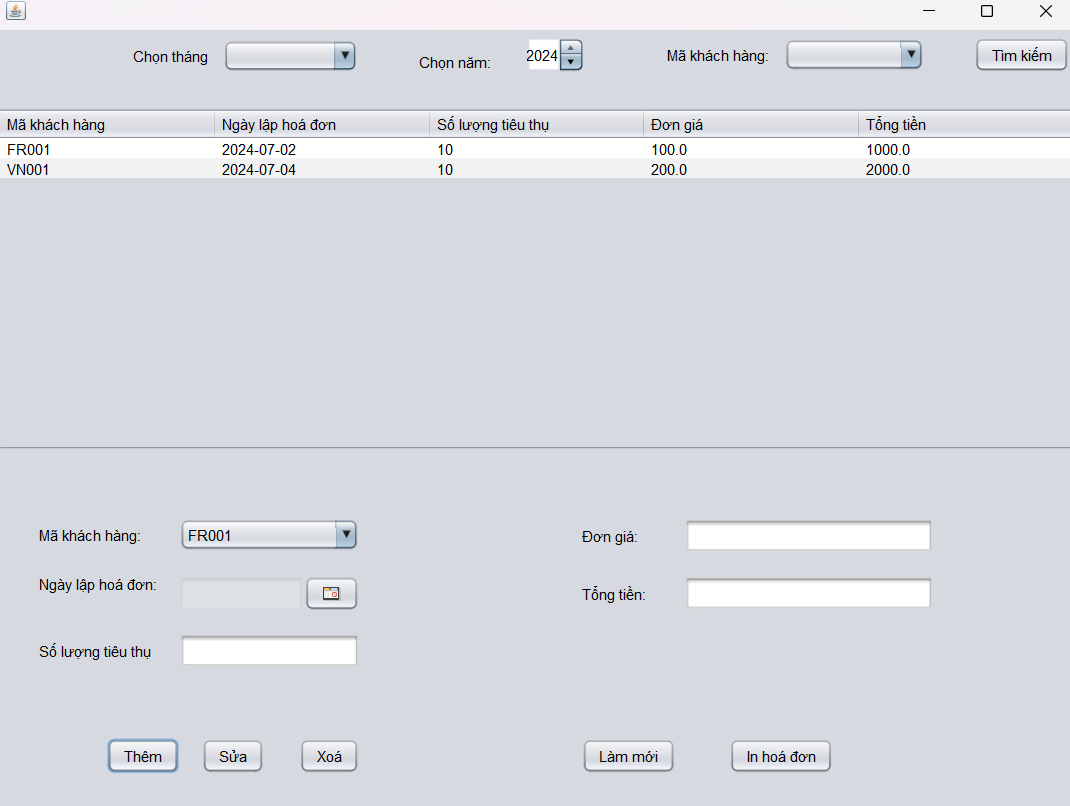


**Hình 4.6.Hiển thị lọc ra khách hàng nước ngoài**

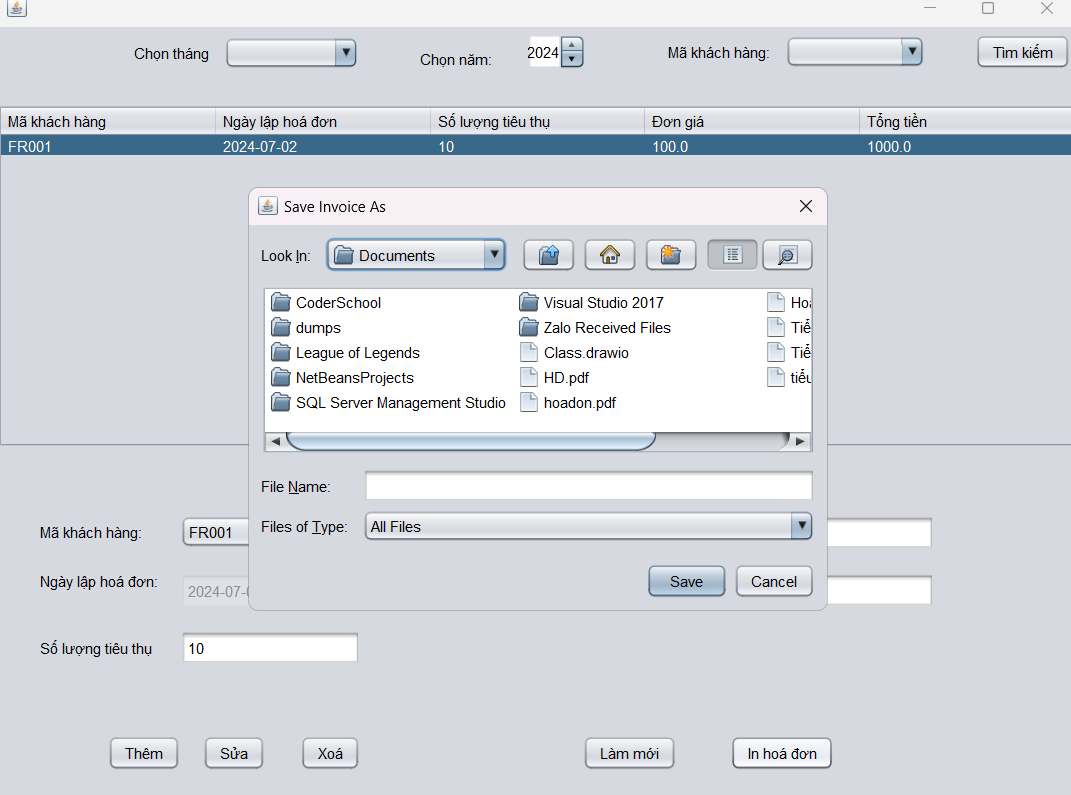


**Hình 4.7.Chức năng tìm kiếm khách hàng theo ID**

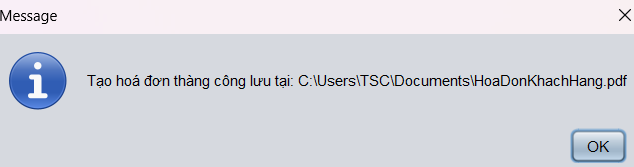
Chức năng in hóa đơn: Người sử dụng phải lựa chọn mã khách hàng cần in hóa đơn, các ngày tháng năm và số lượng tiêu thụ. Sau khi thêm hóa đơn, thông tin khách hàng kèm số lượng tiêu thụ sẽ được hiển thị. Cuối cùng khi in hóa đơn, người dùng sẽ được đưa đến nơi được chọn để lưu trữ tệp tin in hóa đơn. Kết quả sẽ được hiển thị dưới dạng PDF.



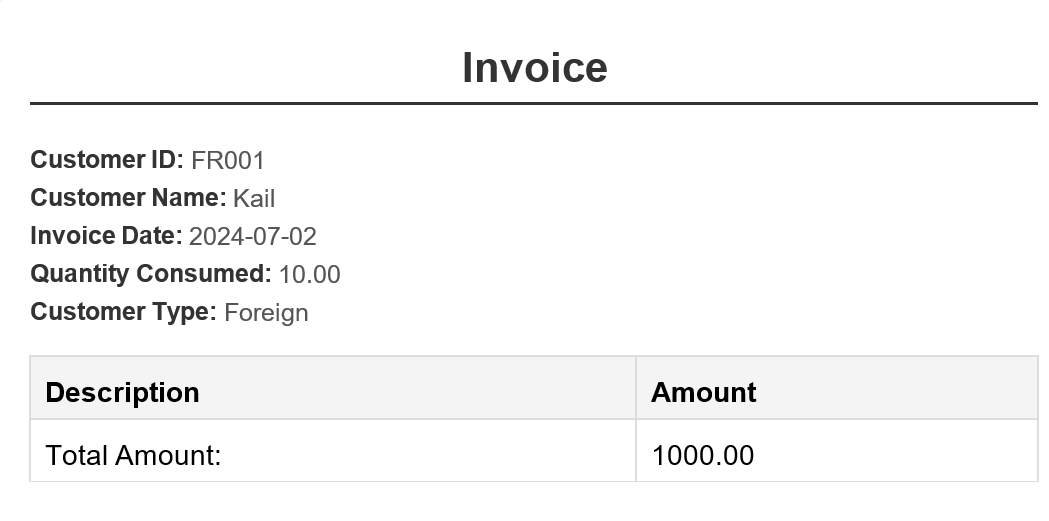
**Hình 4.8.Chức năng in hóa đơn khách hàng**



**Hình 4.9.Lưu trữ file hóa đơn**



**Hình 4.10.Hiển thị thông báo khi hóa đơn được xuất thành công**



**Hình 4.11.Hóa đơn khi được in thành công**

# CHƯƠNG 5 : KẾT LUẬN VÀ ĐÁNH GIÁ

## 5.1. Kết luận

Sau khi hoàn thành chương trình “Quản lý hóa đơn tiền điện” chúng em đã nhận ra vẫn còn nhiều thứ cần phải cải tiến và sửa chữa ở nhiều chỗ, nhưng về mặt lý thuyết nhóm chúng em đã hoàn thành 95% yêu cầu được đề ra và không thể nào tránh khỏi những sai sót, em hi vọng sau bài luận này thầy Lê Huỳnh Phước có thể góp ý để chúng em có thể hoàn thiện và cải thiện phần code tốt hơn về mặt logic để chương trình có thể tối ưu nhất.

## **5.2. Đánh giá và cải tiến trong tương lai**

Chương trình quản lý tiền điện là chương trình khá phức tạp đối với kiến thức và kỹ năng của chúng em hiện tại, nhưng chúng em đã cố gắng để hoàn thành sau vài tuần lên kế hoạch cho chương trình. Chúng em vẫn còn nhiều thứ để cải tiến trong tương lai chẳng hạn như tạo thêm frontend cho chương trình, cải tiến chương trình tối ưu nhất có thể, tính tổng hóa đơn của nhiều loại khách hàng trong tháng, trong năm.

# TÀI LIỆU THAM KHẢO

[1] Refactoring.Guru. [https://refactoring.guru/design-patterns](https://refactoring.guru/design-patterns" \t "_blank).

[2] Design Patterns for Dummies, PhD.Steve Holzner.

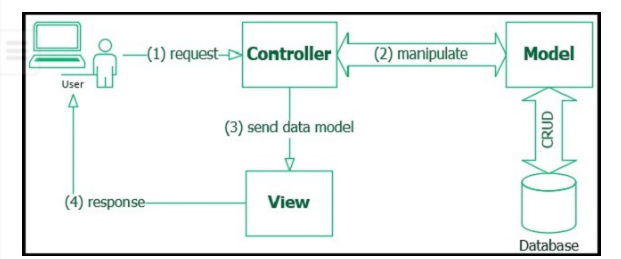
[3] Head First, Eric Freeman.

[4] Gang of Four Design Patterns 4.0.

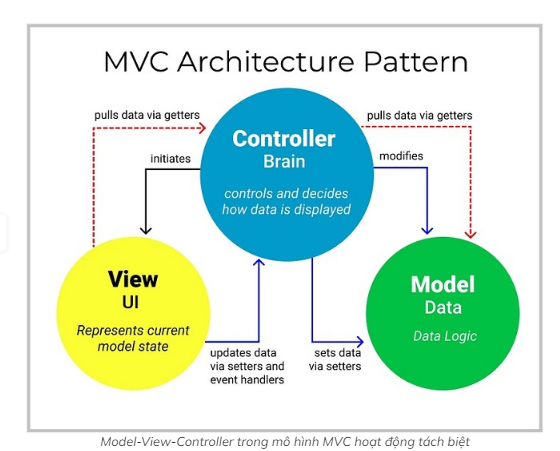
[5] Dive into Design Patterns.

# PHỤ LỤC

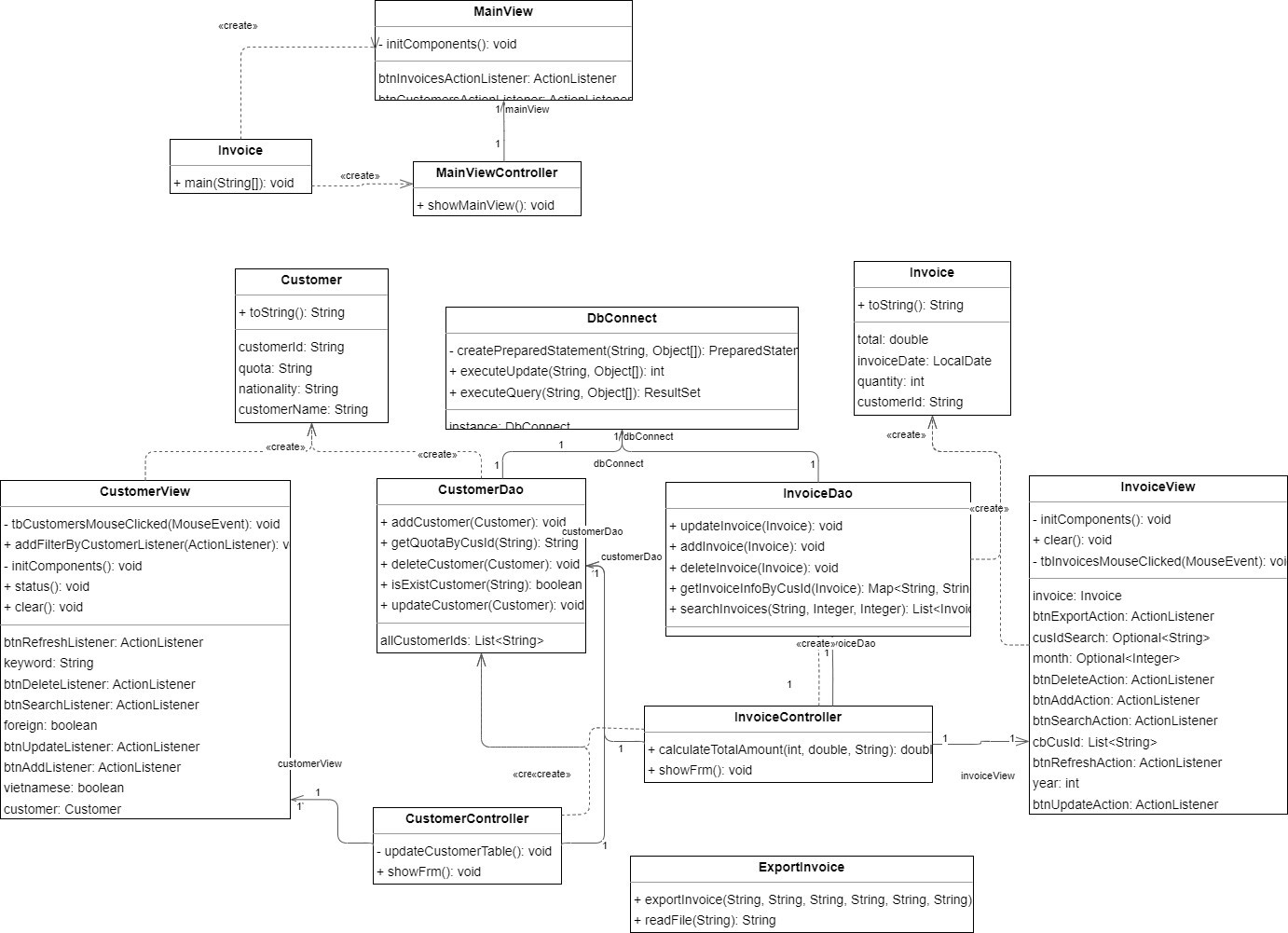
Phụ lục 1: Sơ đồ mô hình MVC



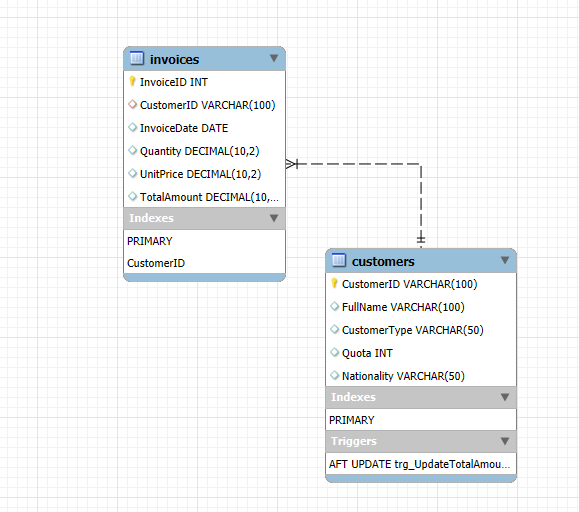
Phụ lục 2: Mô hình kiến trúc MVC



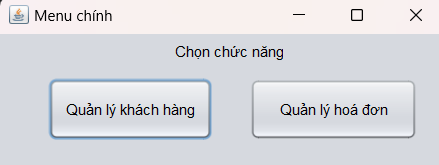
Phụ lục 3: Sơ đồ lớp



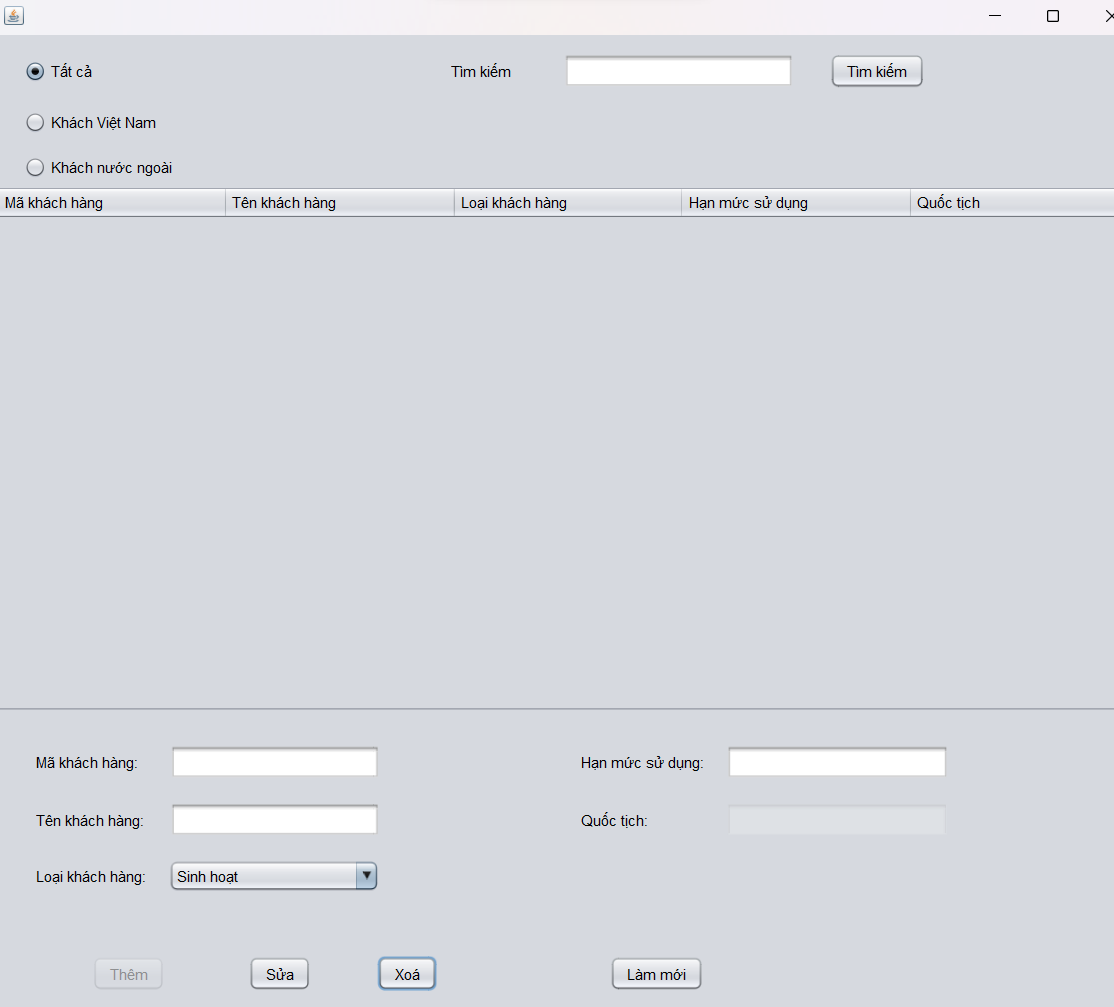
Phụ lục 4: Sơ đồ cơ sở dữ liệu



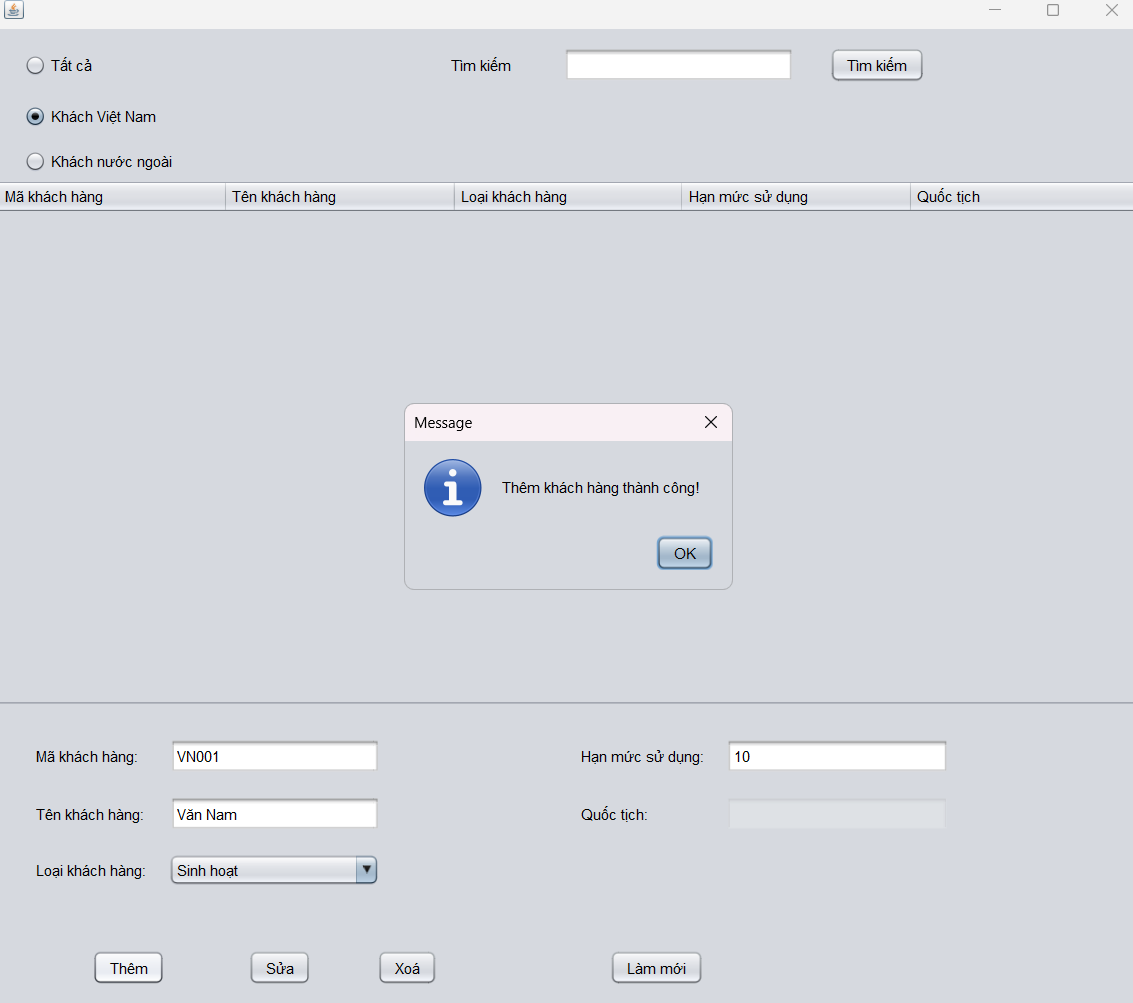
Phụ lục 5: Giao diện Menu chính



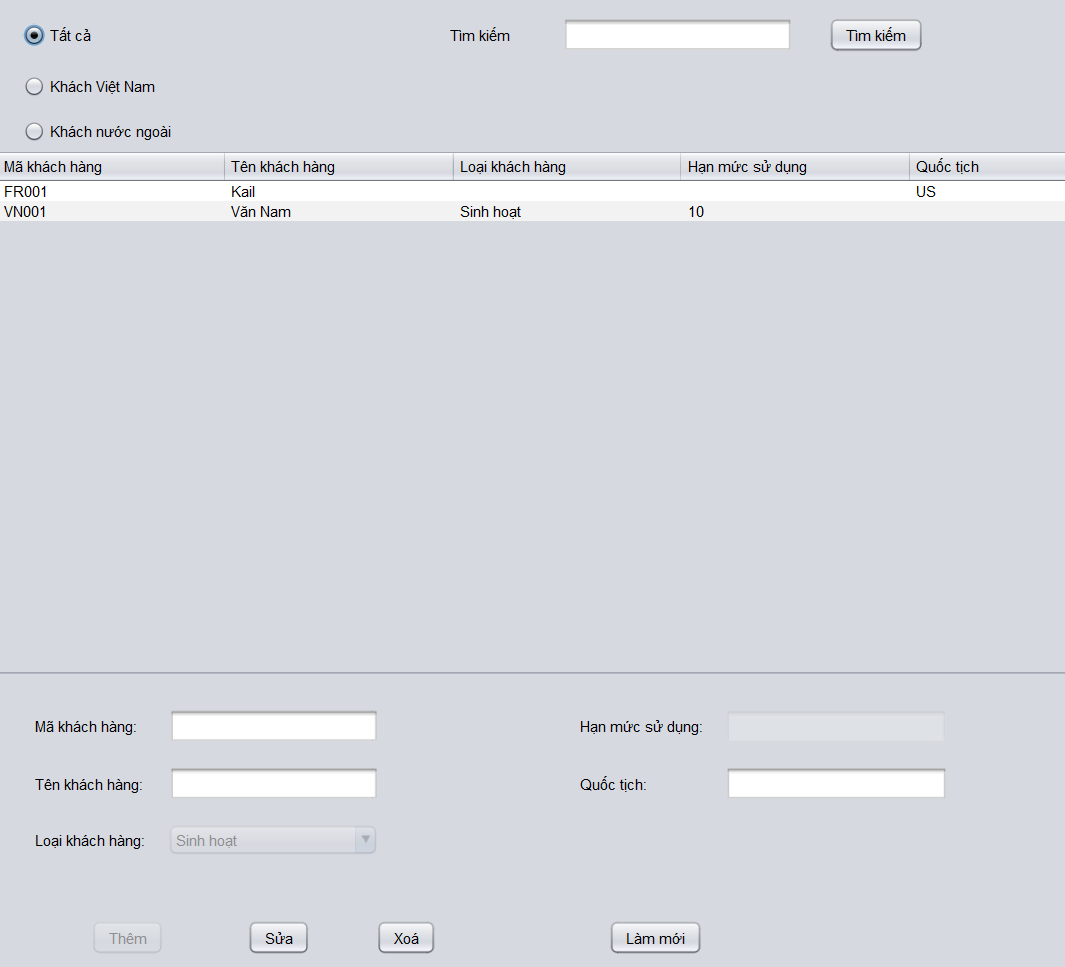
Phụ lục 6: Giao diện UI quản lý khách hàng



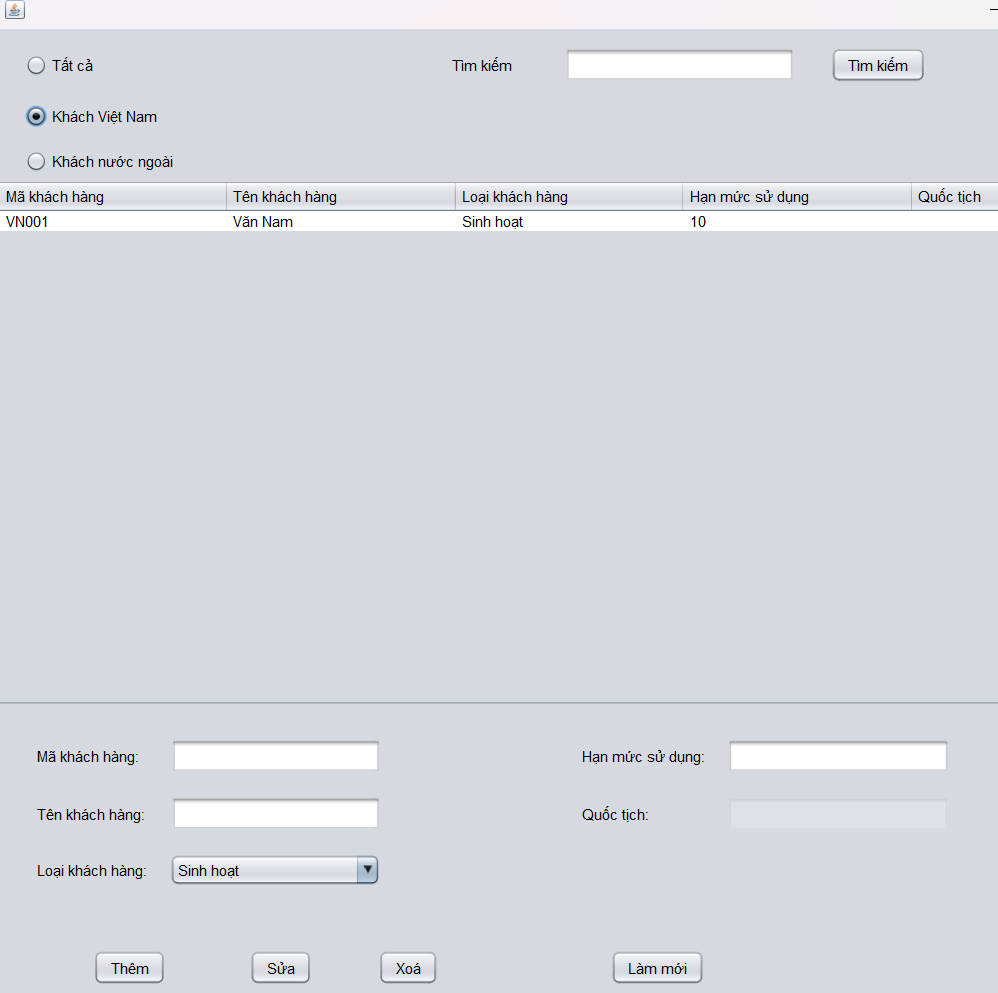
Phụ lục 7: Hiển thị sau khi thêm khách hàng thành công



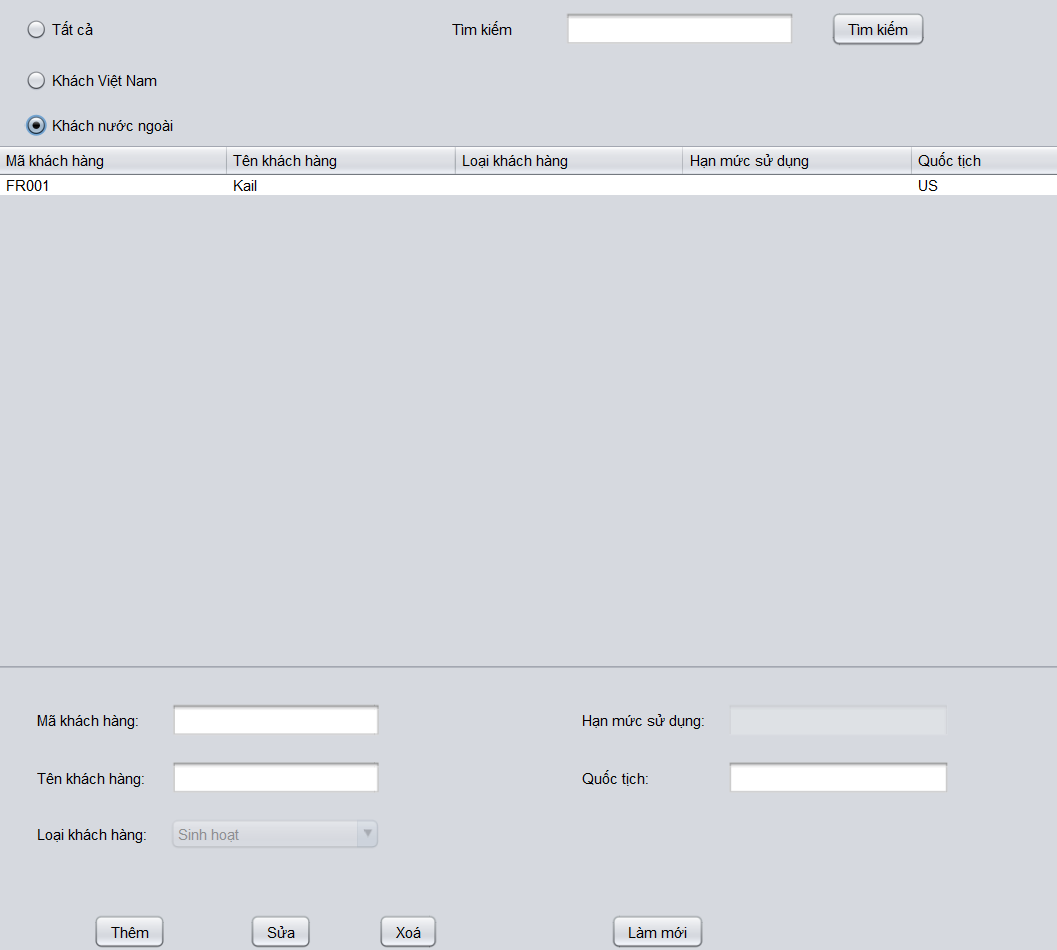
Phụ lục 8: Hiển thị sau khi đã đăng ký khách hàng thành công



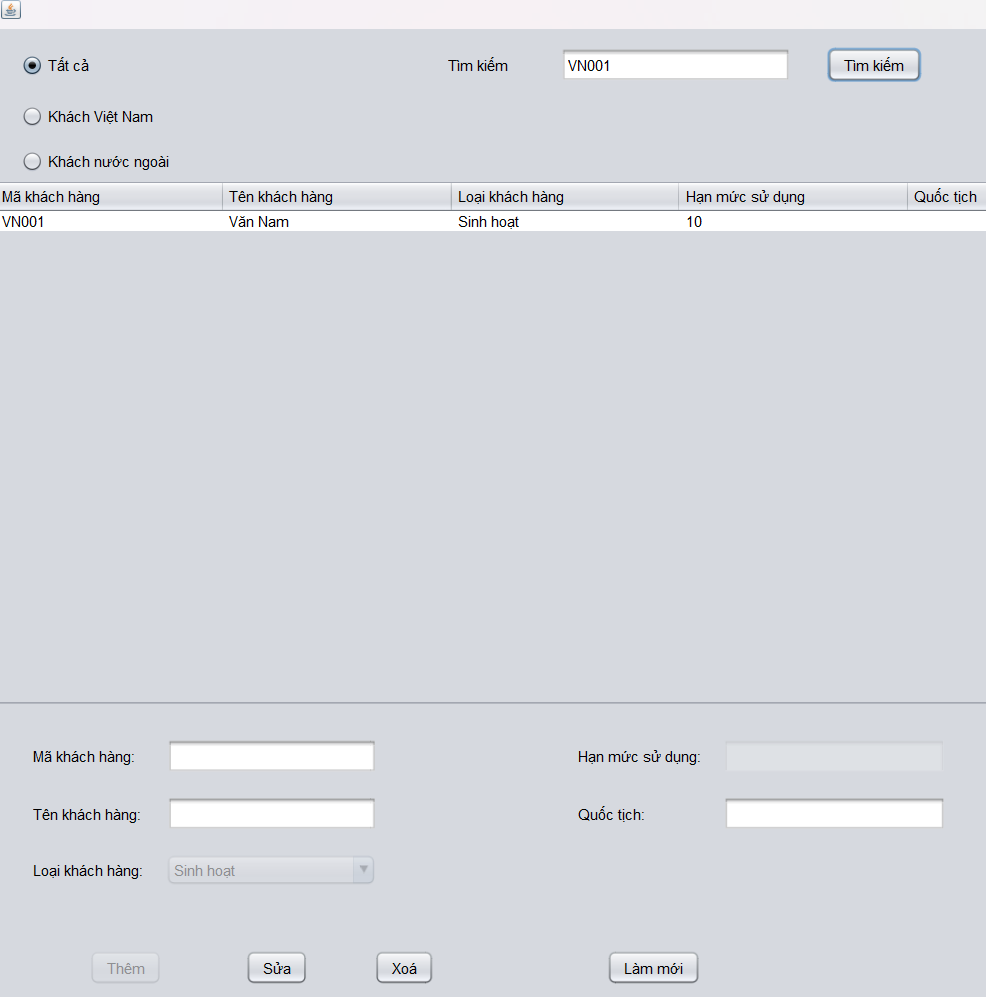
Phụ lục 9: Hiển thị lọc ra khách hàng Việt Nam



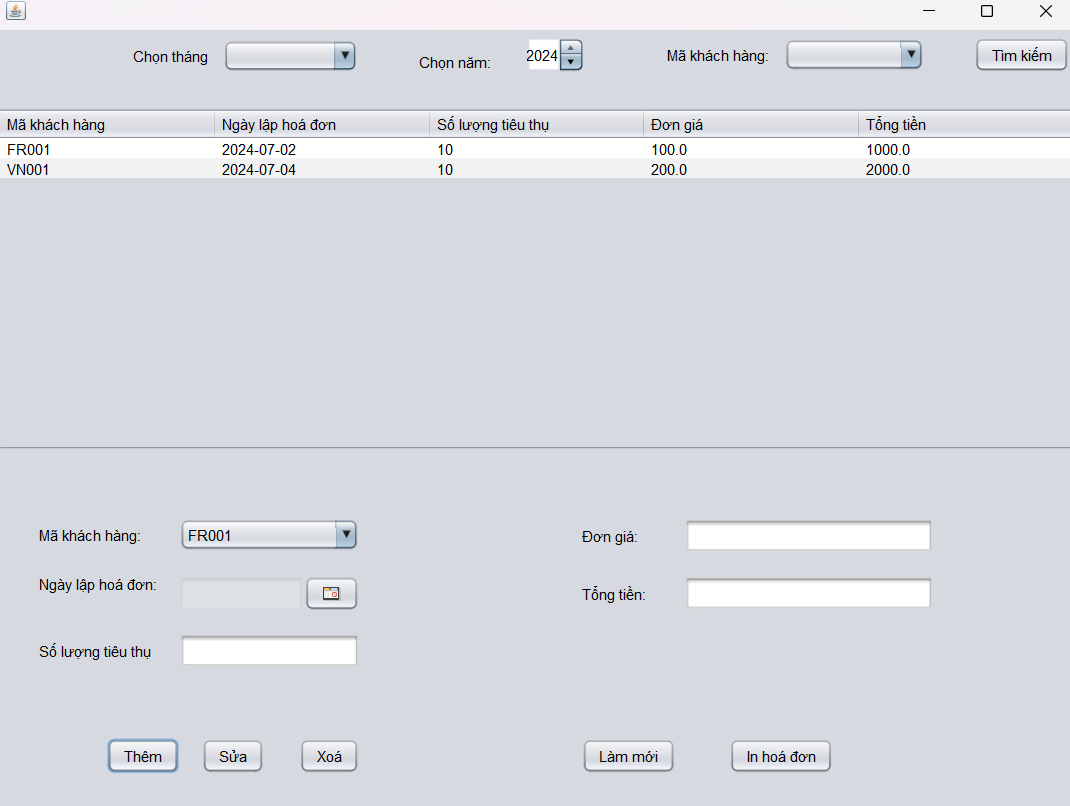
Phụ lục 10: Hiển thị lọc ra khách hàng nước ngoài



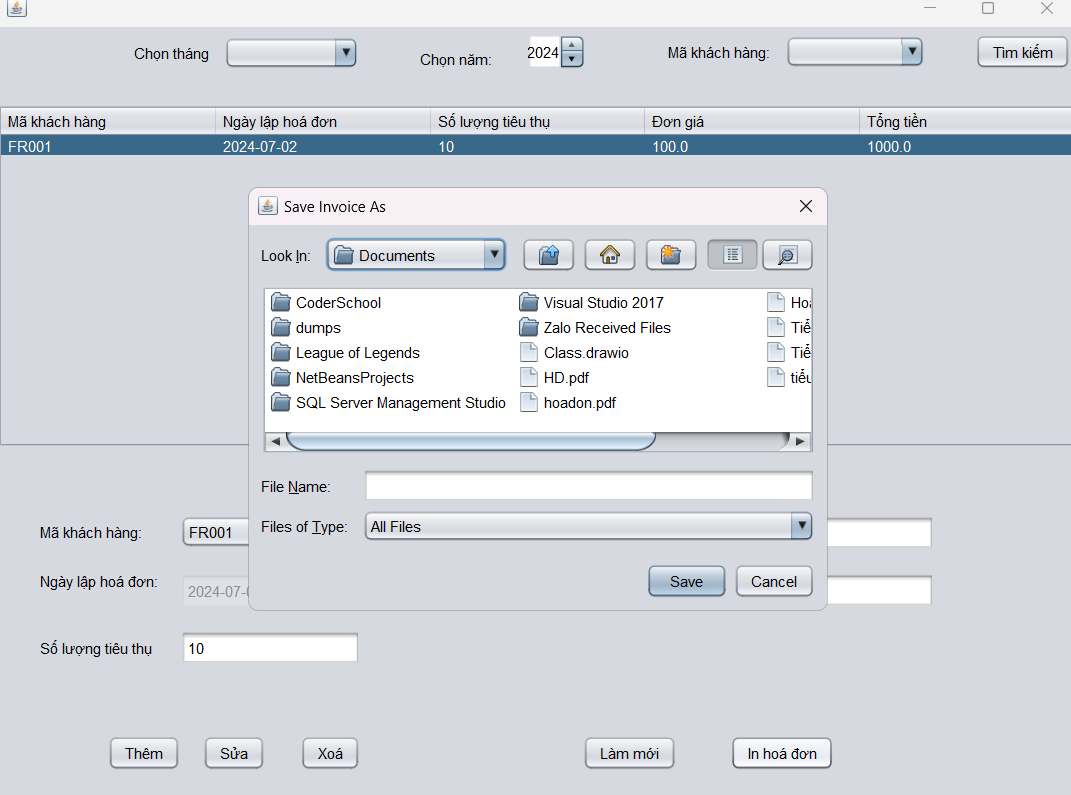
Phụ lục 11: Chức năng tìm kiếm khách hàng theo ID



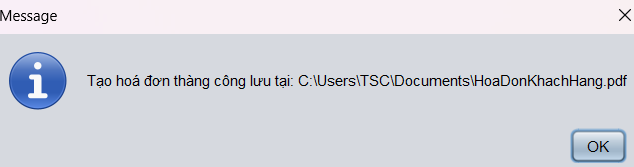
Phụ lục 12: Chức năng in hóa đơn khách hàng



Phụ lục 13: Lưu trữ file hóa đơn



Phụ lục 14: Hiển thị thông báo khi hóa đơn được xuất thành công



Phụ lục 15: Hóa đơn khi được in thành công

