FPT ACADEMY INTERNATIONAL

FPT – APTECH COMPUTER EDUCATION

Centre Name: ACE-THUDUC-1-FPT

Address : 62 Street 36, Van Phuc Residential Area, Hiep Binh Phuoc Ward, Thu Duc City

Karaoke

Suppervisor	**Mr.Pham Cong Danh**	
Batch	**T5.2308.M0**	
Group	**Group 04**	
No	**Full name**	**Student code**
1	Dang To Nhan	Student1501054
2	Phan Tran Dang Chi	Student1501058
3	Nguyen Hoang Anh	Student1501270
4	Nguyen Hoang Minh Ngoc	Student1501214
5	Phan Van Duy	Student1470114

Mont: 07 Year 2024

This is to certify that

Mr.	**DANG TO NHAN**
Mr.	**PHAN TRAN DANG CHI**
Mr.	**NGUYEN HOANG ANH**
Mr.	**NGUYEN HOANG MINH NGOC**
Mr.	**PHAN VAN DUY**

Have successfully Designed & Developed

Karaoke Database Management

Submitted by:

Mr. PHAM CONG DANH

Date Of Issue:

10/07/2024

Authorized Signature:

NGUYEN HOANG MINH NGOC

CONTENT

REVIEW 1

ACKNOWLEDGE

We extend our deepest gratitude and appreciation to all those who have contributed to the successful completion of our Semester 2 project as part of the Advance Diploma of Software Engineering program. As a collaborative effort, this endeavor involved the dedicated efforts of our team of four students who have strived tirelessly to deliver a robust and innovative project.

First and foremost, we would like to express our heartfelt thanks to our esteemed teacher, Mr. Danh, whose guidance and mentorship have been instrumental throughout the development of our project. His wealth of knowledge, unwavering support, and constructive feedback have been invaluable, shaping not only our technical skills but also fostering a deep understanding of the software engineering principles.

The foundation of our project lies in the integration of various technologies, and we want to acknowledge the pivotal role played by SQLServer, CSS, and JavaFx. These tools have been the building blocks of our project, enabling us to create a dynamic and feature-rich application. The hands-on experience gained in utilizing these technologies has not only enriched our technical prowess but also provided a practical insight into real-world software development.

The collaborative nature of this project has allowed us to cultivate effective teamwork and communication skills. Working as a cohesive unit, we learned the importance of coordination, task delegation, and problem-solving. These invaluable skills extend beyond the realm of software engineering and are applicable to various aspects of our academic and professional lives.

Furthermore, we want to express our gratitude to our fellow team members for their commitment and diligence. Each member has brought a unique set of skills and perspectives, contributing to the overall success of the project.

In conclusion, this project has been a journey of growth, learning, and camaraderie. We are sincerely thankful to everyone who has been part of this endeavor, shaping it into a rewarding experience that will undoubtedly influence our future endeavors in the field of software engineering.

INTRODUCTION

Greetings! We are excited to present our project in the realm of software engineering. As part of our Advanced Diploma studies, our team of five diligent students has undertaken the development of a Program Karaoke Database Management. The primary objective is to provide visitors with a user-friendly platform for Karaoke Database Management In an easy.

We aim to provide a visually appealing and technologically robust karaoke database management system. The emphasis lies in creating a seamless user experience, allowing users to easily navigate and manage a diverse array of karaoke tracks and gain a deeper understanding of the associated details. With intuitive management features, our goal is to simplify the database management process, offering users a convenient and efficient means to organize and maintain their song lists. We invite you to join us on this technological journey as we unveil a karaoke database management system that not only showcases our technical expertise but also enhances the karaoke experience for every user.

PROBLEM DEFINITION

Karaoke bar management software has functions that help managers and employees easily use it at work. The software will provide managers and employees with a separate account, able to log in and change passwords when needed.

Room management is an important issue in karaoke bars. Good room management will help businesses grasp important issues when hotels book rooms. A karaoke bar will fail and encounter errors if it does not know which customers have booked rooms in advance, or which customers have booked which room types at what time... Reservation management includes the following functions:

In addition to renting rooms to sing Karaoke, Karaoke shops also sell food items to serve customers in need. Product management helps employees and managers easily manage information about the shop's products, avoiding loss and errors in inspection that cause loss of revenue... There are 8 Functions in product management are: add product, delete product, update product, find product, add product type, delete product type, update product type, find product type.

Payment is an important function, requiring high accuracy to help calculate the amount of money guests must pay when renting a room, when managing the communication system to handle many issues at the same time such as guest entry and exit times, furniture, etc. Knowing the food used by customers and the type of room used by customers... will be extremely difficult and can lead to errors. The payment function of this karaoke management software will take the customer's room information and the product the customer ordered entered by the staff to calculate the final amount, saving time and effort for the user. After charging, employees can produce invoices for customers who need invoices. Payment management includes two functions: billing and invoicing.

To be able to manage customers' old invoices when problems arise, they need to be used such as checking for loss, customers need to check invoices again, need to update when there are errors... Ksing karaoke system Has invoice management function. Invoice management includes 3 functions: find invoices, update invoices, delete invoices.

Customer management helps employees and managers control necessary customer information such as phone numbers and names when customers come to make a reservation or make a reservation. Customer management includes 4 functions: Find customers, delete customer information, update customer information, add customers.

In addition to the general functions for management staff, there is a function to help manage information for your employees, which is employee management. Employee management helps managers control employee information such as: full name, gender, address, rights, shift, login name, number of employees... as well as Add accounts for employees who do not have accounts. Employee management includes 4 functions: add employee accounts, delete employee accounts, update employee information, and find employees.

Statistical management is also a very important function in businesses. The software has a statistical management function that helps users to calculate how much revenue they have in a day, thereby knowing whether the company's profits are good or not. Statistics management includes one function: find revenue (display revenue of the selected day).

USER REQUEST SPECIFICATIONS

1. Manage

Input:

1. **Forms and Text Fields:** Employees can input information through forms and text fields. This includes filling out employee details, account information, product specifications, customer data, or room details.
2. **Buttons and Links:** Employees interact with buttons and links to navigate through different sections of the management system, submit forms, or trigger specific actions.
3. **Dropdowns and Selections:** Employees can make selections from dropdown menus or choose options from lists to customize their management tasks.
4. **Checkboxes and Radio Buttons:** Users can provide input by selecting checkboxes or radio buttons to indicate preferences or choices in the management processes.
5. **Uploads:** If applicable, employees may upload files, images, or documents using file input fields for tasks such as adding employee photos or product images.

Output:

1. **Displaying Results:** The webpage outputs information based on employee input, such as displaying employee details, updated account information, product lists, customer profiles, or room availability.
2. **Error Messages:** If there are issues with the input, the webpage may output error messages to guide employees in correcting their input.
3. **Confirmation Messages:** Employees may receive confirmation messages for successful actions, such as submitting a form or completing a management task.
4. **Visual Changes:** The webpage may undergo visual changes to reflect the processed information, such as updating employee lists, product images, customer details, or room statuses dynamically.

2. ADMIN

Input

1. **Login Credentials:** The system requires input in the form of a username and password for authentication during the login process.
2. **Configuration Settings:** Administrators may input or modify configuration settings, such as system preferences, security parameters, or feature toggles.

3. **Data Upload:** Administrators might upload data, such as user information, song details, or other relevant content to the system.

Process :

1. **Authentication:** The system processes login credentials to authenticate administrators and grant access to the admin panel.
2. **Authorization:** Once authenticated, the system checks for appropriate authorization levels to determine the scope of actions an administrator can perform.
3. **Configuration Processing:** Changes made to configuration settings are processed by the system to implement the updated preferences or rules.
4. **Data Management:** The system processes uploaded data, ensuring proper validation,storage, and integration into the database.

Output:

1. **Admin Dashboard:** After successful login, the system displays an admin dashboard, providing an overview of system status, analytics, or key metrics.
2. **Error Messages:** In case of incorrect login credentials or invalid input during configuration, the system displays error messages to guide administrators.
3. **Activity Logs:** The system generates logs to record administrator activities, helping to track and audit performed actions.
4. **System Status Updates:** If configuration settings are modified, the system updates to reflect changes in system behavior or functionality.
5. **Confirmation Messages:** After successful actions, such as data uploads or configuration updates, the system displays confirmation messages to inform administrators.

Additional Aspects:

1. **Role Management:** The system allows administrators to manage user roles and permissions, defining who can access certain features or perform specific actions.
2. **Security Measures:** Input validation, secure authentication protocols, and encryption are crucial components to ensure the security of admin interactions.
3. **Notification System:** The system may include a notification system to alert administrators about critical events, updates, or potential issues.
4. **Backup and Recovery:** Administrators may have the capability to initiate backup processes and recovery procedures in case of data loss or system failures.

Hardware / Software requirement

Hardware:

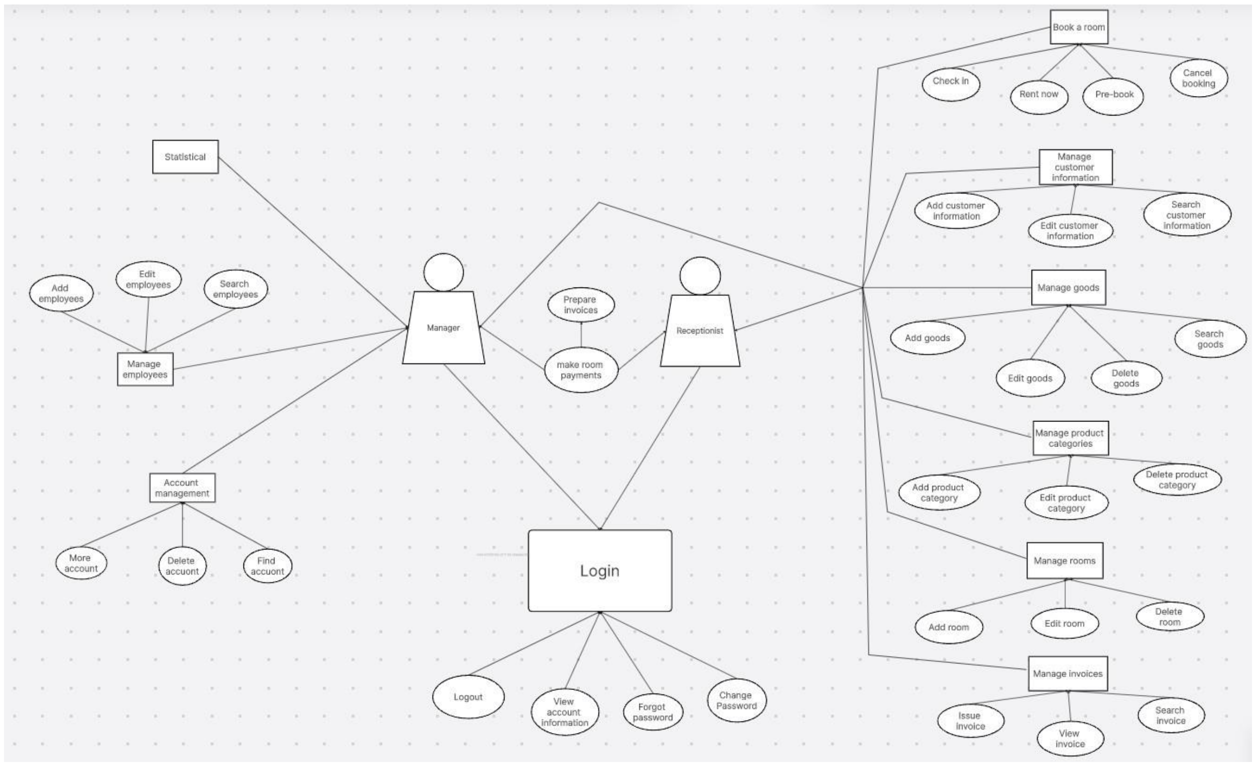
- **Computer with network connection**
- **Operating system: Windows 7, 10**
- **Processor: 2.4 GHz Dual Core Processor or 3 GHz**
- **RAM: 2GB**
- **Graphics card: NVIDIA GeForce 8800/AMD Radeon HD 5670 or equivalent graphics (512 MB)**
- **Free hard disk space: 1GB**

Task Sheet Review 1

Project: Karaoke management			**Date of preparation of activity plan:**			
#	**Task**	**Prepared by**	**Start date**	**Actual Days**	**Team member name**	**status**
1	Acknowledgement	DANG TO NHAN NGUYEN HOANG ANH PHAN TRAN DANG CHI NGUYEN HOANG MINH NGOC PHAN VAN DUY	27/05/2024	31/05/2024	DANG TO NHAN NGUYEN HOANG ANH PHAN TRAN DANG CHI NGUYEN HOANG MINH NGOC PHAN VAN DUY	completed
2	Introduction		27/05/2024	31/05/2024		completed
3	Problem definition		27/05/2024	31/05/2024		completed
4	User request parameters		27/05/2024	31/05/2024		completed
5	Scope of work		27/05/2024	31/05/2024		completed
6	Hardware/software requirement		27/05/2024	31/05/2024		completed
7	Task Sheet		27/05/2024	31/05/2024		Completed
Review			Signature of instructor			
			Mr.Pham Cong Danh			

REVIEW 2

I .USECASE



1 . Use Case Admin

1.1 Login

Use case name: Login		
Actors: Manage		
Description: Start the software to display the login interface		
Pre-condition: Have a new or manager account		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter the correct account and password and click log in 3. Click forgot password	2. Enter the system, display the main interface 4. Switch to the forgot password interface
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not enter or enter incorrect account and password and click log in	2. Show notification "Do not empty or enter incorrect account name or password"

1.2 Forgot password

Use case name: Forgot password		
Actors: Manage		
Description: Click forgot password in the login interface to display. Used to change account password for people who have forgotten their password.		
Pre-condition: Must have an account first		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter the correct account name and press search	2. Displays the account's security question and requires the answer to be entered
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter an incorrect answer or do not enter an answer 3. Enter the new password in incorrect format or not entered at all	2. Displays a message that the answer is incorrect, cannot be empty 4. Displays a message that the password is not in the correct format and cannot be empty
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not enter the account name or enter it incorrectly, press search	2. Display message cannot be empty, account does not exist

1.3 More staff

Use case name: More staff		
Actors: Manage		
Description: Use to add new user		
Pre-condition: Must be a manager account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter the correct account name and press search 3. Enter complete employee information and enter it in the correct format	2. Displays the account's security question and requires the answer to be entered 4. Displays a notification that the addition was successful and employee information is displayed in the employee list
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing employee information or incorrect format (First name capitalized, no numbers or special characters, phone number cannot be duplicate and must be a number)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

1.4 Edit staff

Use case name: Edit staff		
Actors: Manage		
Description: Used to edit information for an employee		
Pre-condition: Must be a manager account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the employee list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again a
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the employee list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the employee list

1.5 More account

Use case name: More account		
Actors: Manage		
Description: Used to add information for an employee		
Pre-condition: Must be a manager account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter missing employee account information or incorrect format (Login name must have no spaces, no special characters; Password must have at least 1 lowercase letter, 1 uppercase letter, 1 special character, No spaces and minimum 8 characters; Reply cannot be blank)	2. Display an incorrect input message and do not leave that information empty
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the employee list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**

	1. Do not select the account to delete but press the delete button	2. Display a notification to select a row in the table

1.6 More account

Use case name: More account		
Actors: Manage		
Description: Used to add information for an employee		
Pre-condition: Must be a manager account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter missing employee account information or incorrect format (Login name must have no spaces, no special characters; Password must have at least 1 lowercase letter, 1 uppercase letter, 1 special character, No spaces and minimum 8 characters; Reply cannot be blank)	2. Display an incorrect input message and do not leave that information empty
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the employee list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry

Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select the account to delete but press the delete button	2. Display a notification to select a row in the table

1.7 Change Password

Use case name: Change Password		
Actors: Employees		
Description: Used to change the account password for an employee		
Pre-condition: Must be the account of a manager or receptionist. Click on the forgot password function at the login screen		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter your username and press the search button	2. Display security questions
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. nput missing password change information (Password must have at least 1 lowercase letter, 1 uppercase letter, 1 special character, no spaces and at least 8 characters; Answer cannot be empty)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**

	1. Don't enter your username, just press the search button	2. Displays a message that the login name cannot be empty

1.8 Add products

Use case name: Add products		
Actors: Employees		
Description: Used to add information to a product		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing product information or incorrect format (Product name does not have special characters; Unit price can only be entered in numbers)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

1.9 Product updates

Use case name: Product updates		
Actors: Employees		
Description: Used to update information for a product		
Pre-condition: Must be a manager or receptionist account.Login successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the product list

1.10 Delete product

Use case name: Delete product		
Actors: Employees		
Description: Used to delete account information for a product		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and click delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

1.11 Add product type

Use case name: Add product type		
Actors: Employees		
Description: Used to add information to a product type		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor’s actions:**	**System’s responses:**
	N/A	N/A
Alternativeflow	**Actor’s actions:**	**System’s responses:**
	1. Enter missing product type information or incorrect format (Product type name does not have special characters)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor’s actions:**	**System’s responses:**
	N/A	N/A

1.12 Update product type

Use case name: Update product type		
Actors: Employees		
Description: Used to update information for a product type		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product type list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product type list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the product type list

1.13 Delete product type

Use case name: Delete product type		
Actors: Employees		
Description: Used to delete account information for a product type		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product type list and click delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select the type of product to delete but press the delete button	2. Display a notification to select a row in the table

1.14 Add customers

Use case name: Add customers		
Actors: Employees		
Description: Used to add information for a customer		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on customer management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing customer information or incorrect format (Customer name cannot enter numbers and special characters; Phone number must be 9-10 digits, no characters) 3. Enter complete information but the phone number is the same	2. Display an incorrect input message and do not leave that information empty 4. Display notification that the phone number already exists
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

1.15 Update customers

Use case name: Update customers		
Actors: Employees		
Description: Used to update information for a customer		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on customer management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the customer list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the customer list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the customer list

1.16 More room

Use case name: More room		
Actors: Employees		
Description: Used to add information to a room		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on room management		
Post-condition: N/A		
Basic flow	**Actor’s actions:**	**System’s responses:**
	1. Enter complete room information	2. Display a notification of successful addition and update the list again
Alternativeflow	**Actor’s actions:**	**System’s responses:**
	1. Input missing room information	2. Display notice that room information cannot be left empty
Exceptions	**Actor’s actions:**	**System’s responses:**
	N/A	N/A

1.17 Room update

Use case name: Room update		
Actors: Employees		
Description: Used to update information for a room		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on room management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the room list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the room list

1.18 Delete room

Use case name: Delete room		
Actors: Employees		
Description: Used to delete account information for a room		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on room management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the room list and press delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select the room to delete but press the delete button	2. Display a notification to select a row in the table

1.19 Finding customers

Use case name: Finding customers		
Actors: Employees		
Description: Used to search for information of customers who need to make a reservation		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on booking		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter the phone number the customer needs to make a reservation and press the search button	2. Display customer booking information
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter the customer phone number that has not been added to the customer list and press the search button	2. isplay interface without customers
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not enter the phone number of the customer coming to make the reservation and press the search button	2. Display interface without customers

1.20 Add food

Use case name: Add food		
Actors: Employees		
Description: Used to add food to a customer's bill		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter complete food information	2. Display the notification of successfully adding food and update the invoice detail list
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing food information or incorrect format (Quantity must be a positive integer greater than 0)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**
	1. Choose the same foods	2. Display duplicate food notification

1.21 Delete food

Use case name: Delete food		
Actors: Employees		
Description: Used to delete food for customer invoices		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor’s actions:**	**System’s responses:**
	1. Select a line in the invoice details list and click delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor’s actions:**	**System’s responses:**
	N/A	N/A
Exceptions	**Actor’s actions:**	**System’s responses:**
	1. Do not select the room to delete but press the delete button	2. Display a notification to select a line in the invoice detail table

1.22 Food updates

Use case name: Food updates		
Actors: Employees		
Description: Used to update food items for customer invoices		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the room list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the invoice detail list

1.23 Pay

Use case name: Pay		
Actors: Employees		
Description: Used to pay customer orders		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select 1 room to pay for in the booking list	2. Display successful payment interface
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select a room in the booking list and click pay	2. Displays a message that no room has been selected for payment

1.24 Invoice

Use case name: Invoice		
Actors: Employees		
Description: Used to issue customer invoices		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select 1 invoice to export in the invoice list table 3.Select the path, enter the file name and click export invoice	2. Display file selection interface (where invoices are stored) 4. Display exported invoice excel file (including invoice information and invoice details)
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

1.25 Statistical

Use case name: Statistical		
Actors: Employees		
Description: Used for revenue statistics		
Pre-condition: Must be a manager account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a time period before the current date	2. Display statistical results (including daily sales statistics and statistics of all products sold)
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select the current date period	2. Do not display statistical results
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

2. Use case for Employees

2.1 Change Password

Use case name: Change Password		
Actors: Employees		
Description: Used to change the account password for an employee		
Pre-condition: Must be the account of a manager or receptionist. Click on the forgot password function at the login screen		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter your username and press the search button	2. Display security questions
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. nput missing password change information (Password must have at least 1 lowercase letter, 1 uppercase letter, 1 special character, no spaces and at least 8 characters; Answer cannot be empty)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't enter your username, just press the search button	2. Displays a message that the login name cannot be empty

2.2 Add products

Use case name: Add products		
Actors: Employees		
Description: Used to add information to a product		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing product information or incorrect format (Product name does not have special characters; Unit price can only be entered in numbers)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

2.3 Product updates

Use case name: Product updates		
Actors: Employees		
Description: Used to update information for a product		
Pre-condition: Must be a manager or receptionist account.Login successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the product list

2.4 Delete product

Use case name: Delete product		
Actors: Employees		
Description: Used to delete account information for a product		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and click delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

2.5 Add product type

Use case name: Add product type		
Actors: Employees		
Description: Used to add information to a product type		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing product type information or incorrect format (Product type name does not have special characters)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**

	N/A	N/A

2.6 Update product type

Use case name: Update product type		
Actors: Employees		
Description: Used to update information for a product type		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on product management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product type list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product type list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the product type list

2.7 Delete product type

Use case name: Delete product type		
Actors: Employees		
Description: Used to delete account information for a product type		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on employee management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the product type list and click delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select the type of product to delete but press the delete button	2. Display a notification to select a row in the table

2.8 Add customers

Use case name: Add customers		
Actors: Employees		
Description: Used to add information for a customer		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on customer management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing customer information or incorrect format (Customer name cannot enter numbers and special characters; Phone number must be 9-10 digits, no characters) 3. Enter complete information but the phone number is the same	2. Display an incorrect input message and do not leave that information empty 4. Display notification that the phone number already exists
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

2.9 Update customers

Use case name: Update customers		
Actors: Employees		
Description: Used to update information for a customer		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on customer management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the customer list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Select a line in the customer list and enter the information that needs to be corrected in the wrong format	2. Displays a message stating that the input format is incorrect, requiring re-entry
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the customer list

2.10 More room

Use case name: More room		
Actors: Employees		
Description: Used to add information to a room		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on room management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter complete room information	2. Display a notification of successful addition and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Input missing room information	2. Display notice that room information cannot be left empty
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

2.11 Room update

Use case name: Room update		
Actors: Employees		
Description: Used to update information for a room		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on room management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the room list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the room list

2.12 Delete room

Use case name: Delete room		
Actors: Employees		
Description: Used to delete account information for a room		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on room management		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the room list and press delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select the room to delete but press the delete button	2. Display a notification to select a row in the table

2.13 Finding customers

Use case name: Finding customers		
Actors: Employees		
Description: Used to search for information of customers who need to make a reservation		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on booking		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter the phone number the customer needs to make a reservation and press the search button	2. Display customer booking information
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter the customer phone number that has not been added to the customer list and press the search button	2. isplay interface without customers
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not enter the phone number of the customer coming to make the reservation and press the search button	2. Display interface without customers

2.14 Add food

Use case name: Add food		
Actors: Employees		
Description: Used to add food to a customer's bill		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Enter complete food information	2. Display the notification of successfully adding food and update the invoice detail list
Alternativeflow	**Actor's actions:**	**System's responses:**
	1. Enter missing food information or incorrect format (Quantity must be a positive integer greater than 0)	2. Display an incorrect input message and do not leave that information empty
Exceptions	**Actor's actions:**	**System's responses:**
	1. Choose the same foods	2. Display duplicate food notification

2.15 Delete food

Use case name: Delete food		
Actors: Employees		
Description: Used to delete food for customer invoices		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the invoice details list and click delete	2. Display a notification of successful deletion and update of the list
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select the room to delete but press the delete button	2. Display a notification to select a line in the invoice detail table

2.16 Food updates

Use case name: Food updates		
Actors: Employees		
Description: Used to update food items for customer invoices		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select a line in the room list and enter the information to be edited in the correct format and click update	2. Display a successful update message and update the list again
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Don't select any line in the list and click update	2. Display a message asking to select a line in the invoice detail list

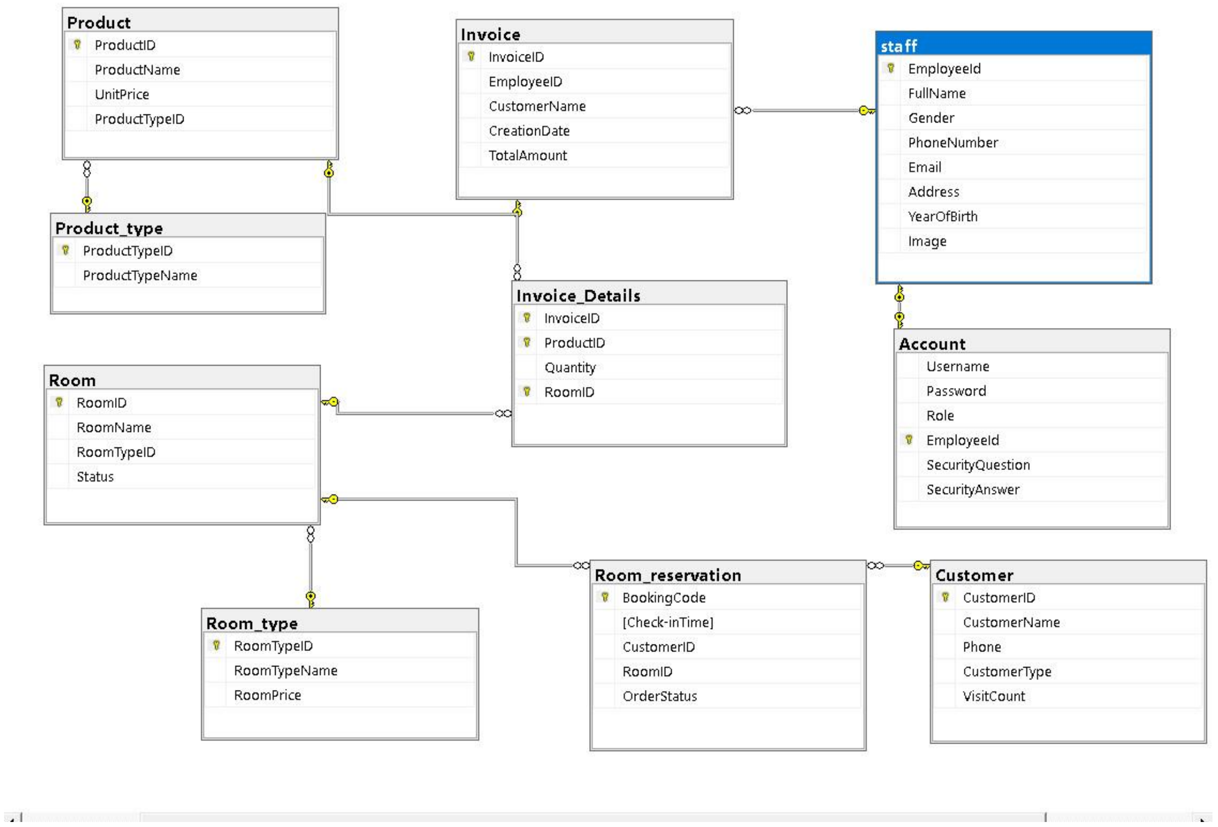
2.17 Pay

Use case name: Pay		
Actors: Employees		
Description: Used to pay customer orders		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select 1 room to pay for in the booking list	2. Display successful payment interface
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	1. Do not select a room in the booking list and click pay	2. Displays a message that no room has been selected for payment

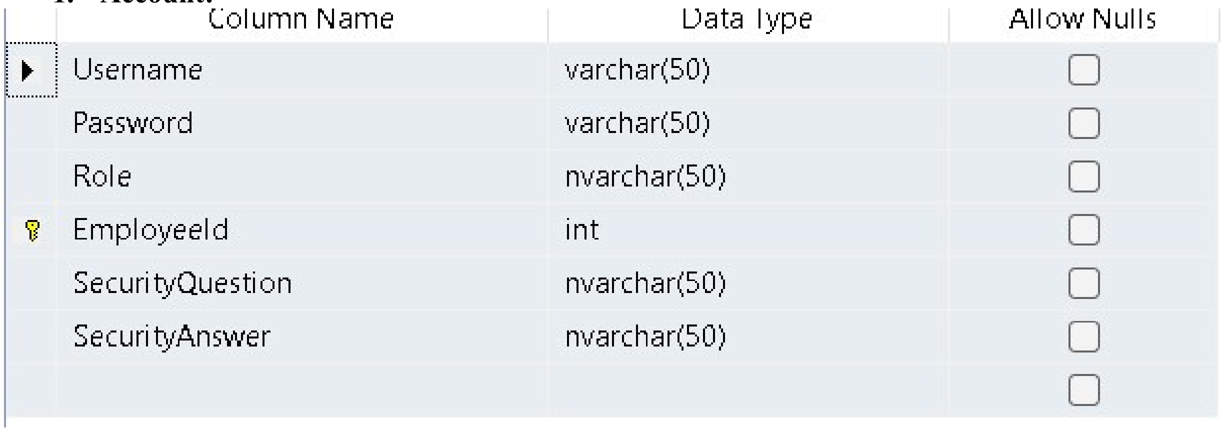
2.18 Invoice

Use case name: Invoice		
Actors: Employees		
Description: Used to issue customer invoices		
Pre-condition: Must be a manager or receptionist account. Log in successfully and click on payment		
Post-condition: N/A		
Basic flow	**Actor's actions:**	**System's responses:**
	1. Select 1 invoice to export in the invoice list table 3.Select the path, enter the file name and click export invoice	2. Display file selection interface (where invoices are stored) 4. Display exported invoice excel file (including invoice information and invoice details)
Alternativeflow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

ENTITY RELATIONSHIP DIAGRAM (ERD)

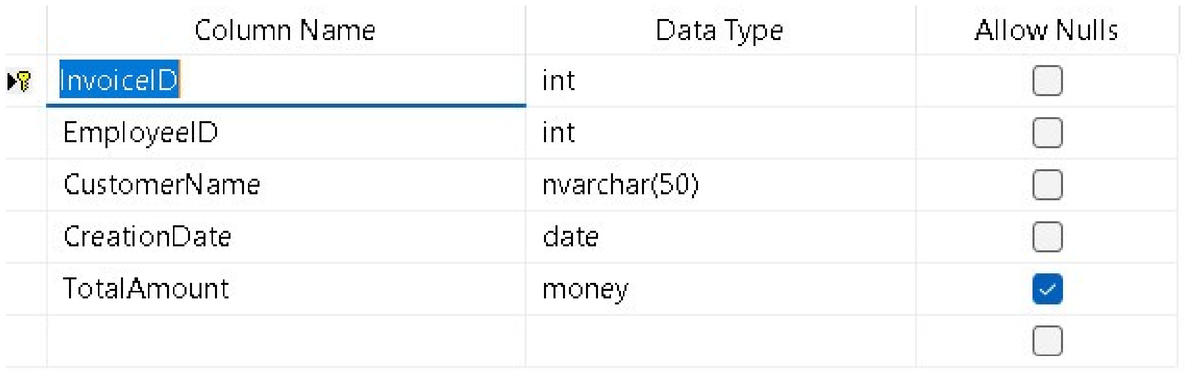


1. **Account:**



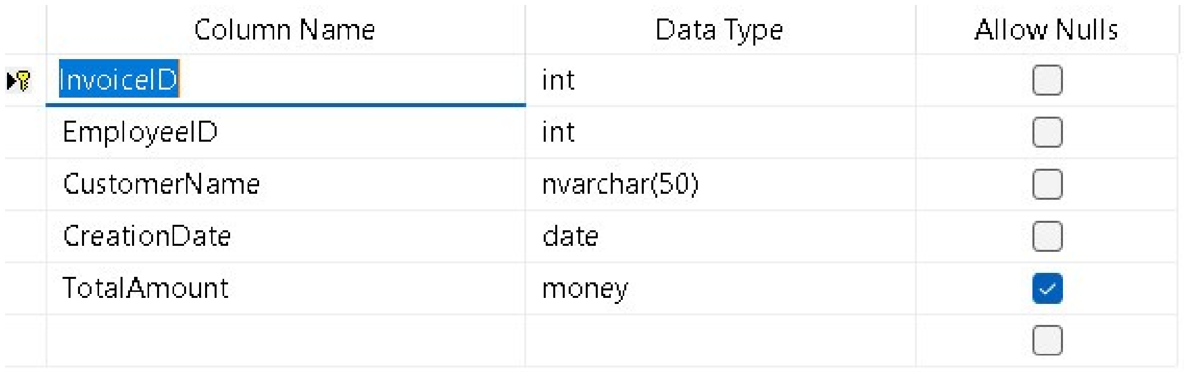
Column Name	Data Type	Allow Nulls
Username	varchar(50)	☐
Password	varchar(50)	☐
Role	nvarchar(50)	☐
EmployeeId	int	☐
SecurityQuestion	nvarchar(50)	☐
SecurityAnswer	nvarchar(50)	☐
		☐

2. Customer



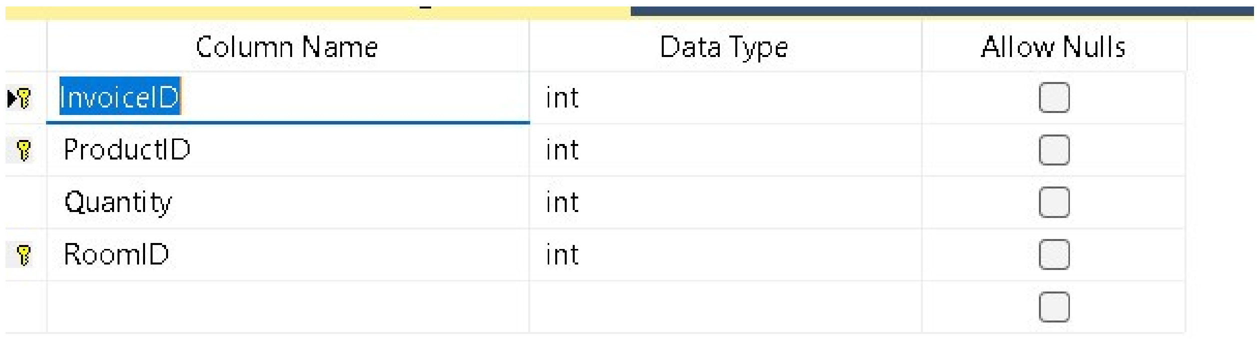
Column Name	Data Type	Allow Nulls
InvoiceID	int	☐
EmployeeID	int	☐
CustomerName	nvarchar(50)	☐
CreationDate	date	☐
TotalAmount	money	✓
		☐

3. Invoice



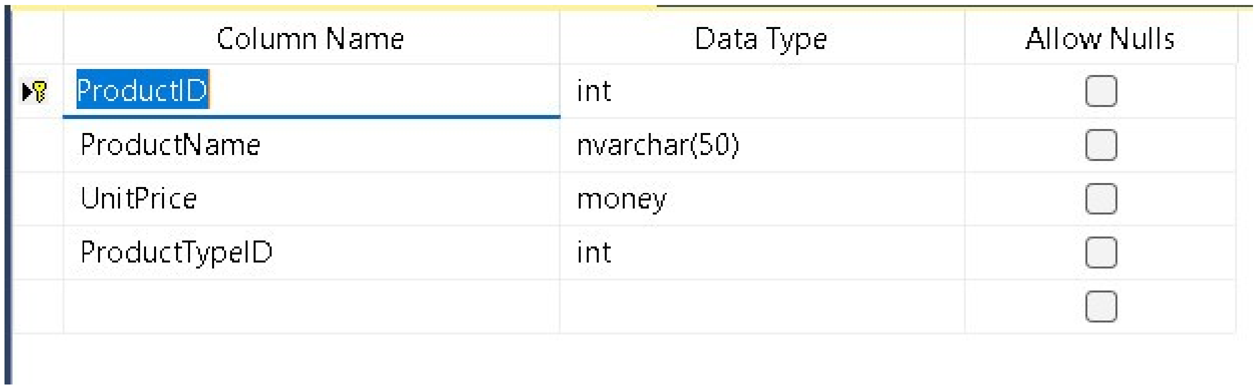
Column Name	Data Type	Allow Nulls
InvoiceID	int	☐
EmployeeID	int	☐
CustomerName	nvarchar(50)	☐
CreationDate	date	☐
TotalAmount	money	✓
		☐

4. Invoice_Details



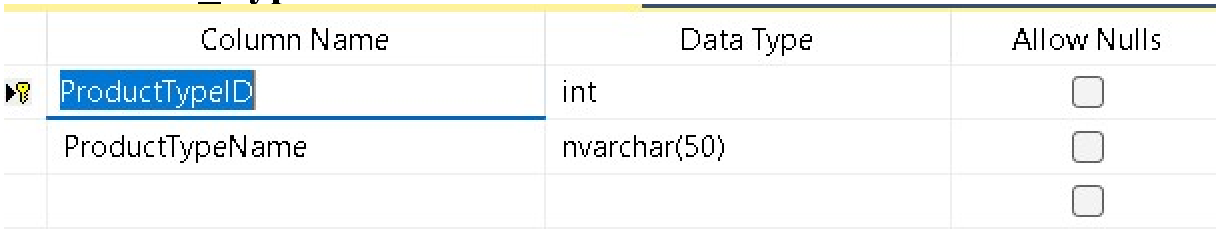
Column Name	Data Type	Allow Nulls
InvoiceID	int	☐
ProductID	int	☐
Quantity	int	☐
RoomID	int	☐
		☐

5. Product



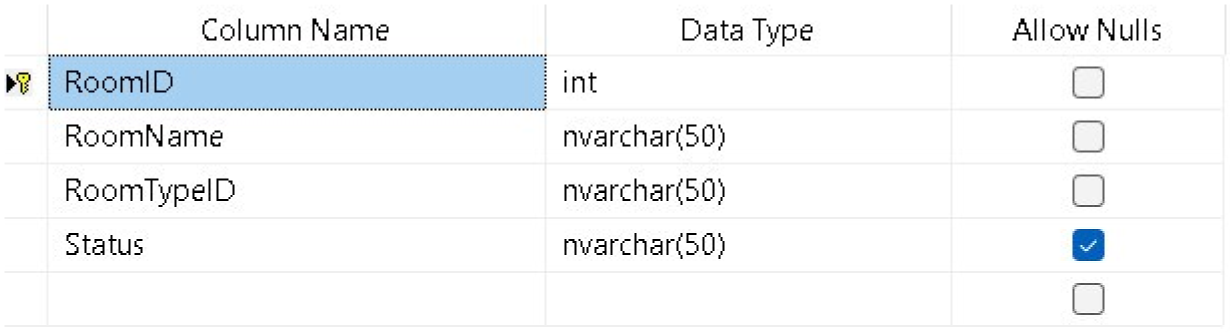
Column Name	Data Type	Allow Nulls
ProductID	int	☐
ProductName	nvarchar(50)	☐
UnitPrice	money	☐
ProductTypeID	int	☐
		☐

6. Product_Type



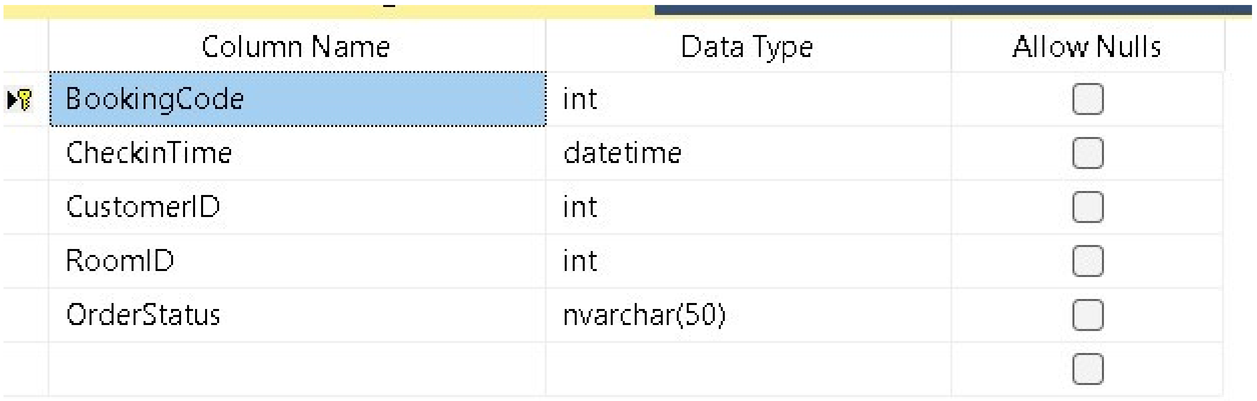
Column Name	Data Type	Allow Nulls
ProductTypeID	int	☐
ProductTypeName	nvarchar(50)	☐
		☐

7. Room



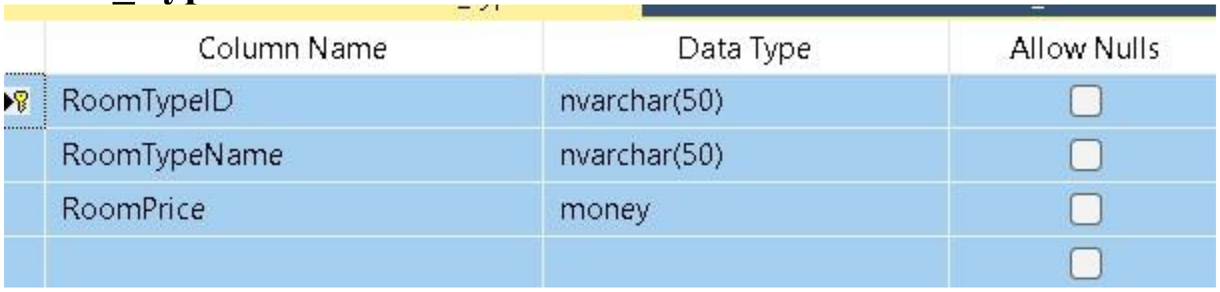
Column Name	Data Type	Allow Nulls
RoomID	int	☐
RoomName	nvarchar(50)	☐
RoomTypeID	nvarchar(50)	☐
Status	nvarchar(50)	✓
		☐

8. Room_reservation



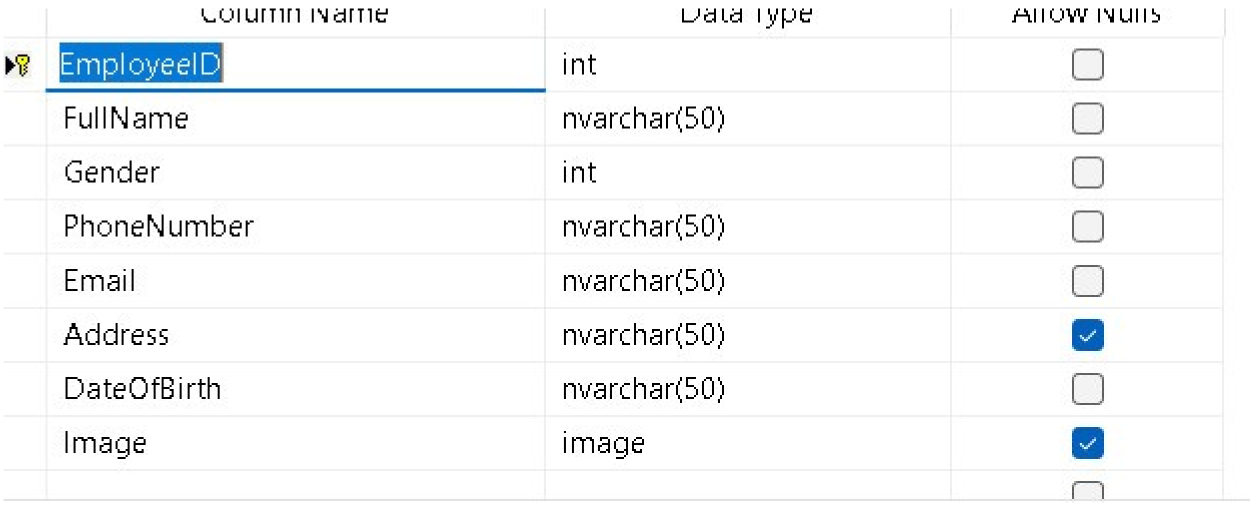
Column Name	Data Type	Allow Nulls
BookingCode	int	☐
CheckinTime	datetime	☐
CustomerID	int	☐
RoomID	int	☐
OrderStatus	nvarchar(50)	☐
		☐

9. Room_Type



Column Name	Data Type	Allow Nulls
RoomTypeID	nvarchar(50)	☐
RoomTypeName	nvarchar(50)	☐
RoomPrice	money	☐
		☐

10. Staff



Column Name	Data Type	Allow Nulls
EmployeeID	int	☐
FullName	nvarchar(50)	☐
Gender	int	☐
PhoneNumber	nvarchar(50)	☐
Email	nvarchar(50)	☐
Address	nvarchar(50)	✓
DateOfBirth	nvarchar(50)	☐
Image	image	✓
		☐

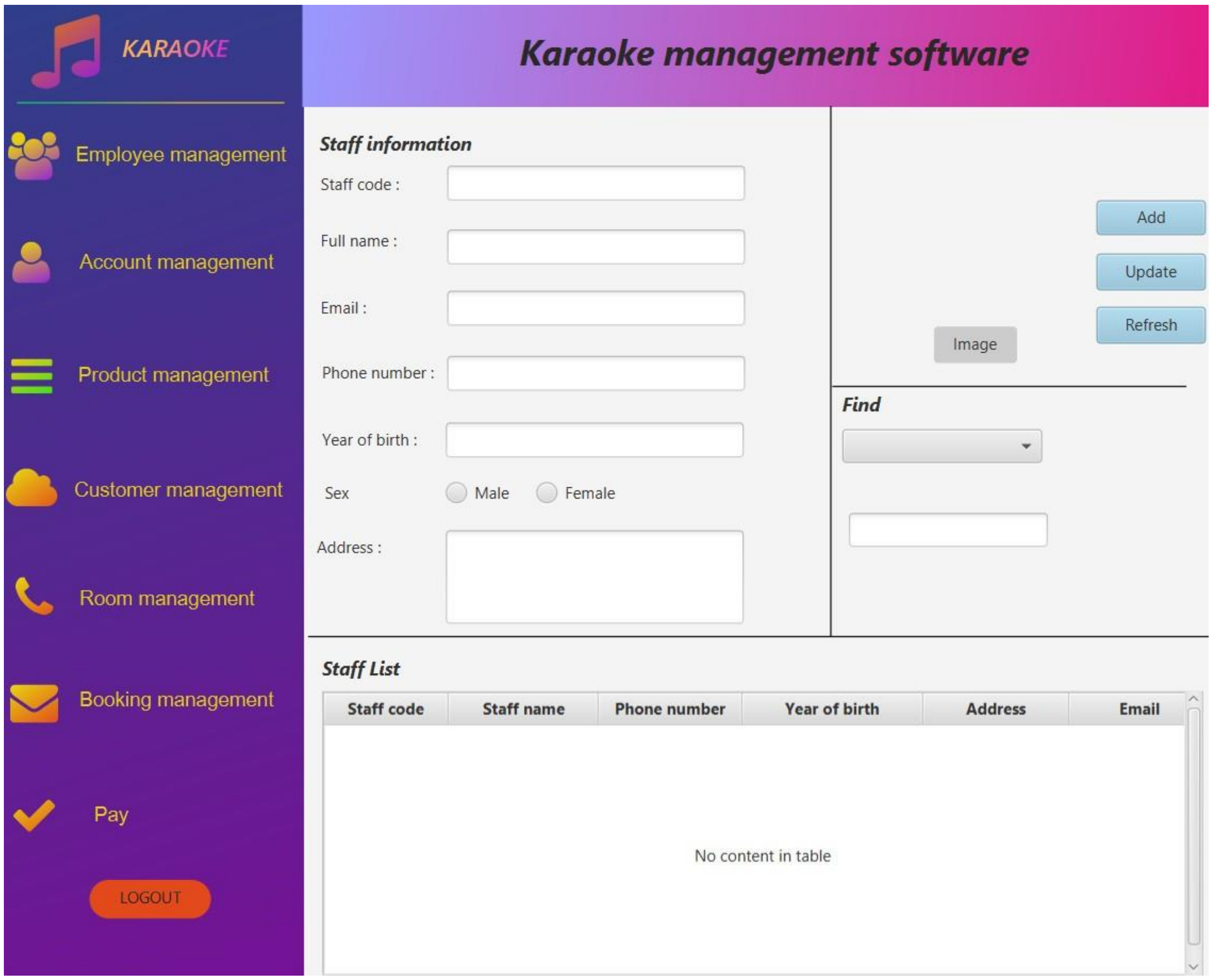
TASK SHEET REVIEW 2

Project:Karaoke management			**Date of preparation of activity plan:**			
#	**Task**	**Prepared by**	**Start date**	**days**	**Team member name**	**status**
1	Use case	NGUYEN HOANG MINH NGOC	31/05/2024	07/06/2024	NGUYEN HOANG MINH NGOC	completed
2	ENTITY RELATIONSHIP DIAGRAM (ERD)	DANG TO NHAN NGUYEN HOANG ANH PHAN TRAN DANG CHI NGUYEN HOANG MINH NGOC PHAN VAN DUY	31/05/2024	07/06/2024	DANG TO NHAN NGUYEN HOANG ANH PHAN TRAN DANG CHI NGUYEN HOANG MINH NGOC PHAN VAN DUY	completed
Review			Signature of instructor			
			Mr.Pham Cong Danh			

REVIEW 3

GUI DESIGN

1. Personnel management



Staff code	Staff name	Phone number	Year of birth	Address	Email

No content in table

1: Click the “Employee Management” button to display the employee management interface

2: Fields used to enter employee information.

3: Table displaying Staff list

4: Enter employee information , press the "Add" button to add employee

5: Select the employee to update in the Staff list table , enter the information to be edited (2) and press the "Update" button to update the employee

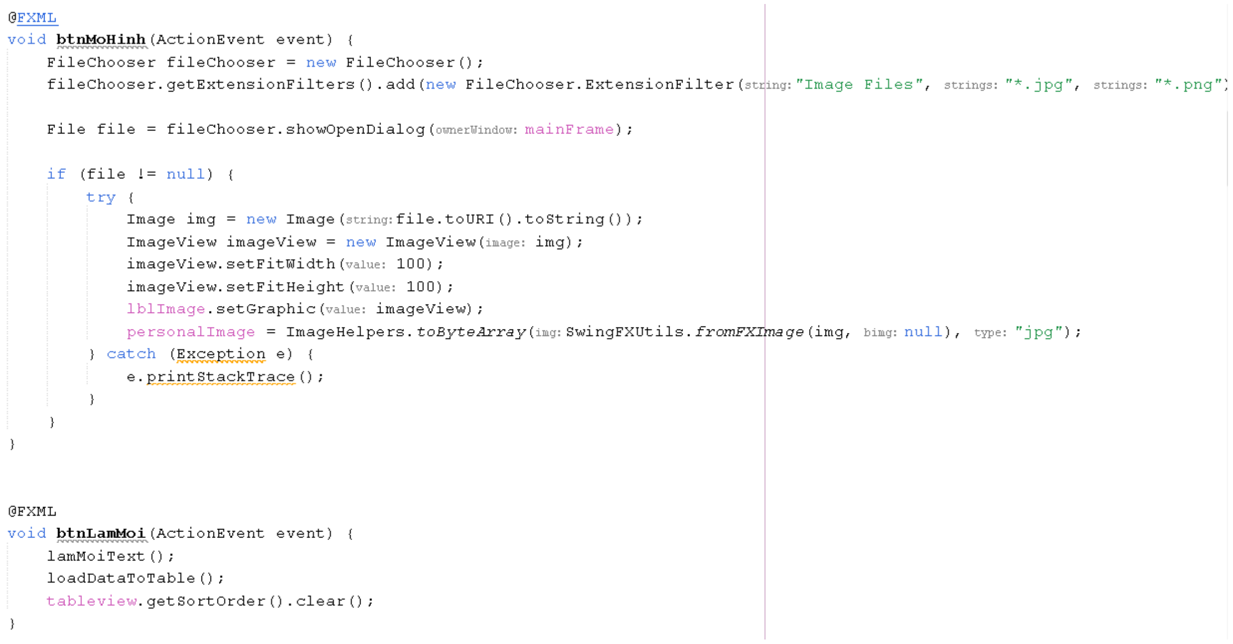
6: Click the “Refresh” button to refresh the input fields and Staff list

7: Click the "Open photo" button to select a photo for the employee (if desired)

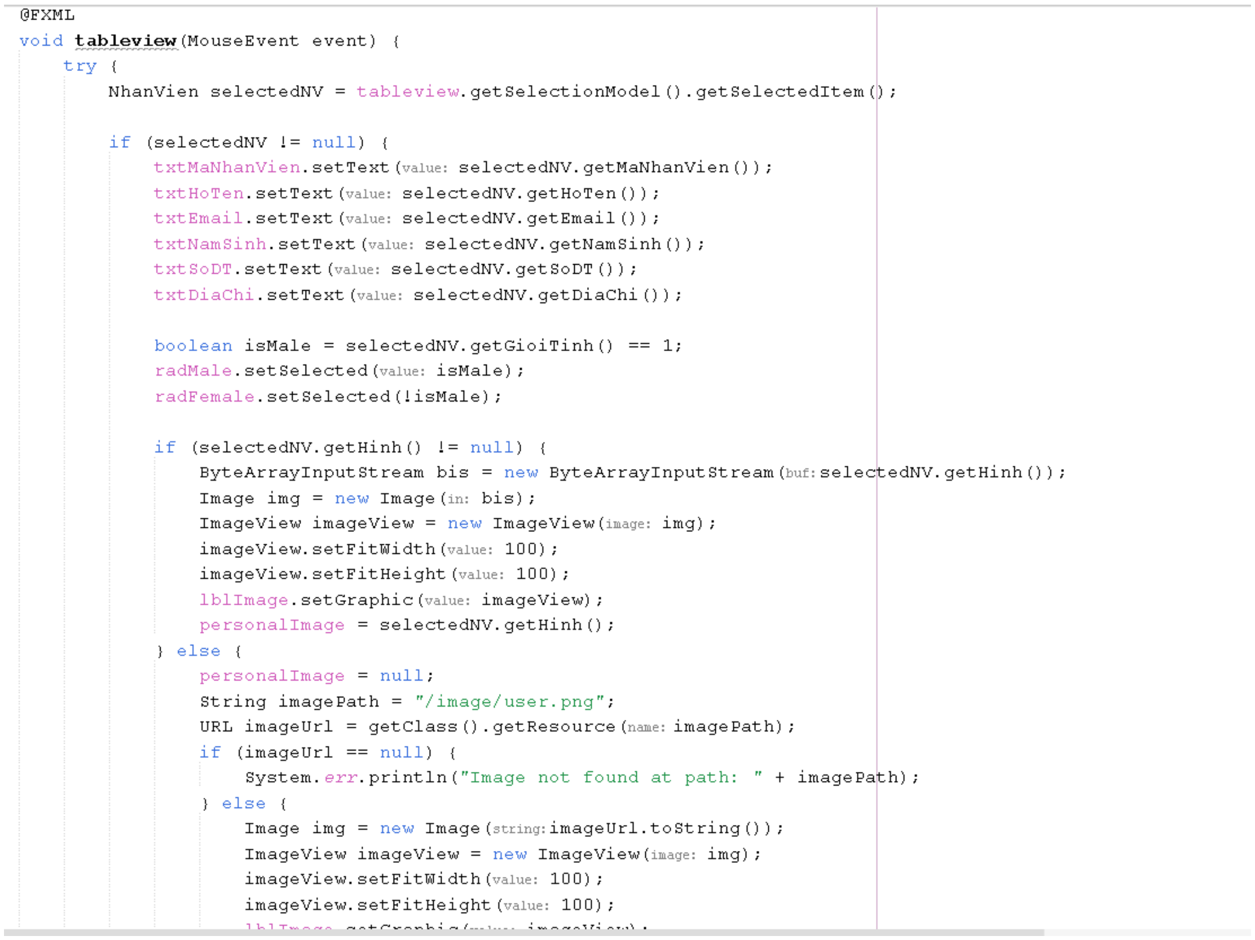
8: Select the employee code you want to search and press the "Search" button, employee information will be displayed in the employee list

9: Click the “Log out” button to exit the application

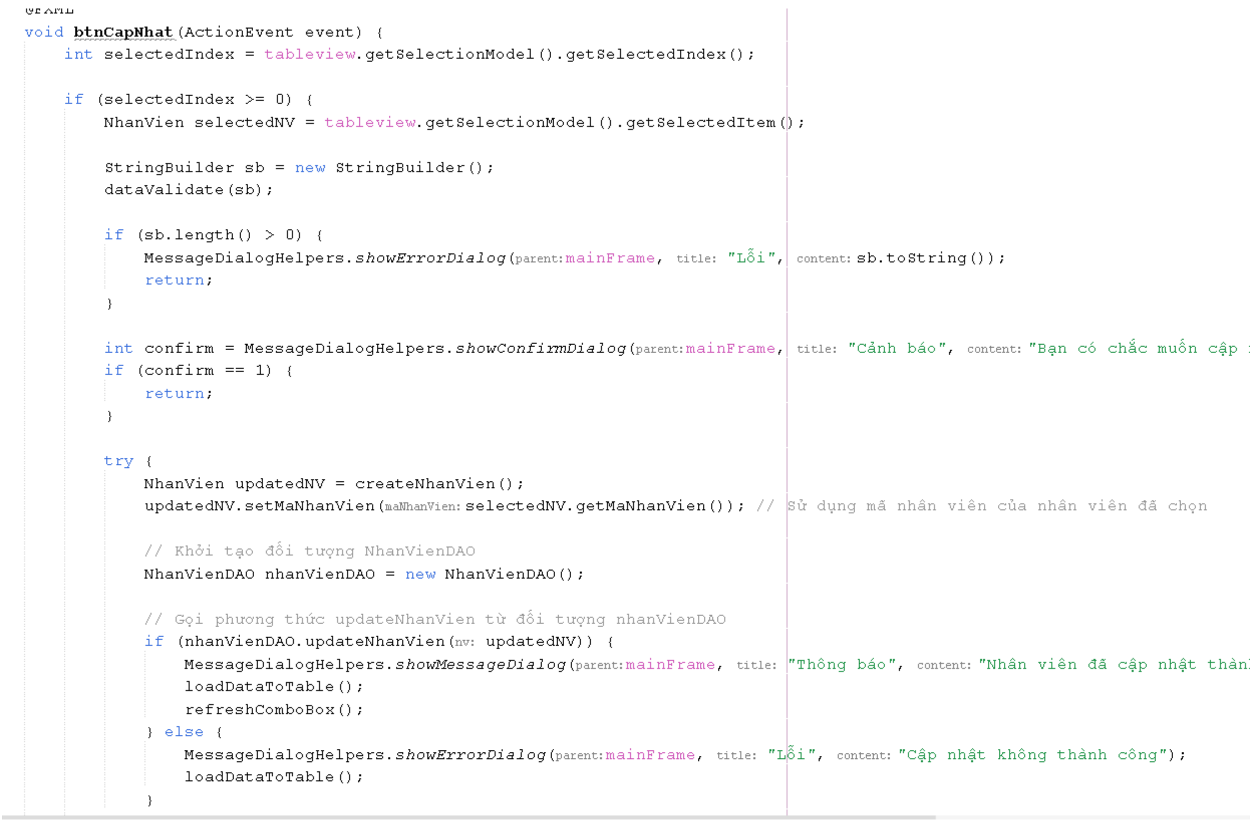
- **Validate:**
  - Full name : staff name cannot be blank , Full name cannot contain special characters
  - Email: Email cannot be blank , Email format is wrong.
  - Phone Number: Phone Number cannot be blank , The phone number is in the wrong format, must have 9-10 digits, no special characters.
  - DateOfBirth : afterbirth cannot be blank , There are 4 digits, no letters or special characters
  - Address : address cannot be blank



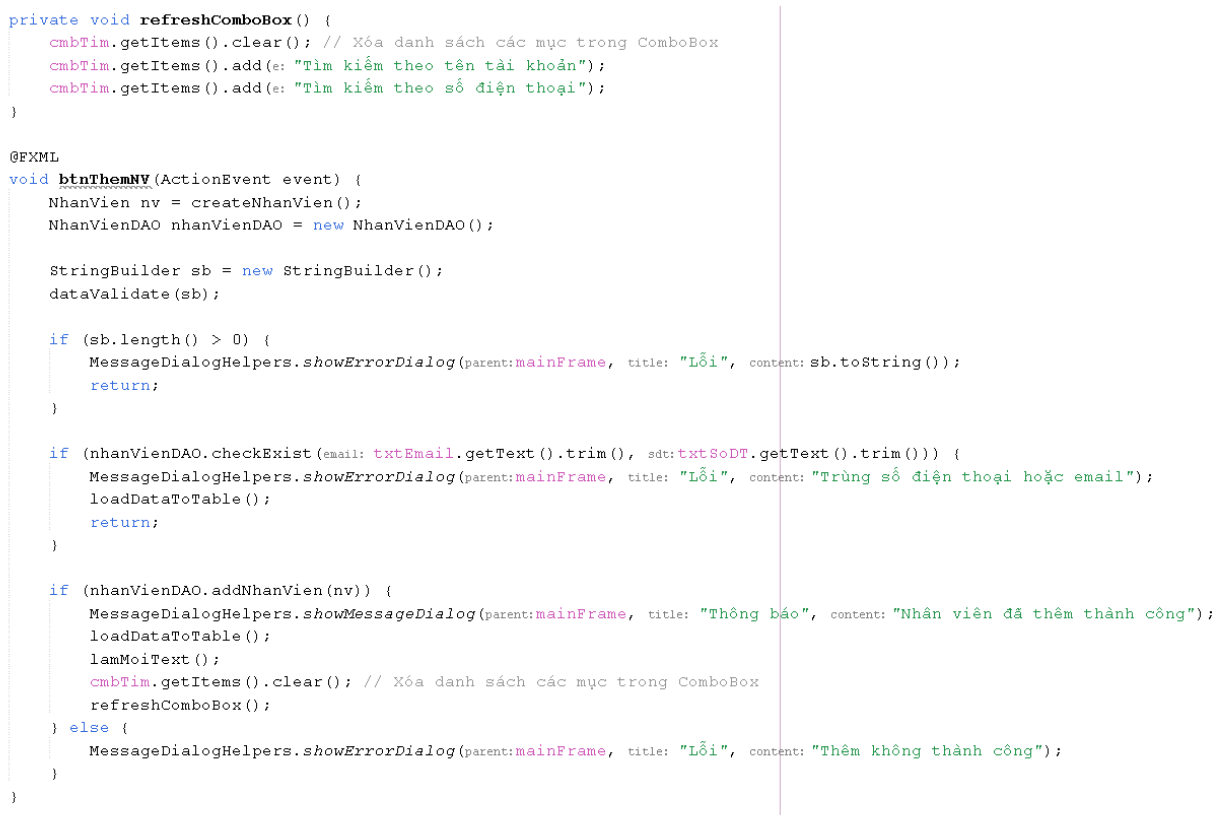
```
@FXML
void btnMoHinh(ActionEvent event) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter(string: "Image Files", strings: "*.jpg", strings: "*.png")

    File file = fileChooser.showOpenDialog(ownerWindow: mainFrame);

    if (file != null) {
        try {
            Image img = new Image(string: file.toURI().toString());
            ImageView imageView = new ImageView(image: img);
            imageView.setFitWidth(value: 100);
            imageView.setFitHeight(value: 100);
            lblImage.setGraphic(value: imageView);
            personalImage = ImageHelpers.toByteArray(img: SwingFXUtils.fromFXImage(img, bimg: null), type: "jpg");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


@FXML
void btnLamMoi(ActionEvent event) {
    lamMoiText();
    loadDataToTable();
    tableview.getSortOrder().clear();
}
```



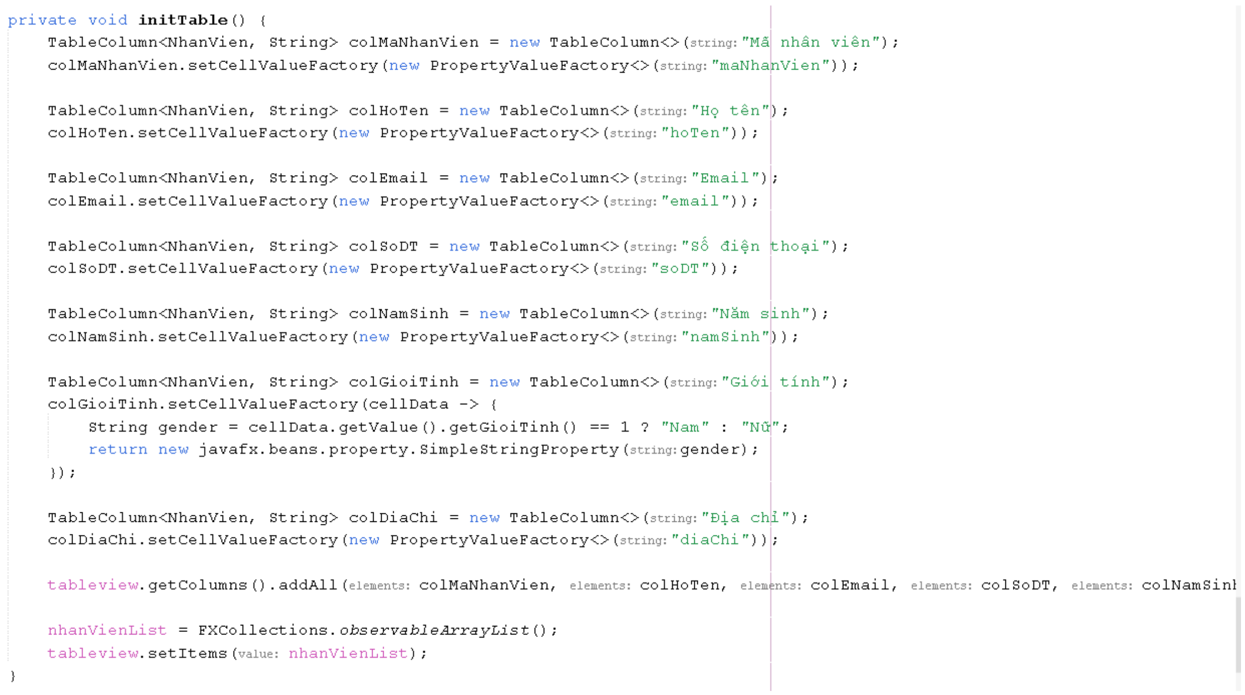
```
@FXML
void tableview(MouseEvent event) {
    try {
        NhanVien selectedNV = tableview.getSelectionModel().getSelectedItem();

        if (selectedNV != null) {
            txtMaNhanVien.setText(value: selectedNV.getMaNhanVien());
            txtHoTen.setText(value: selectedNV.getHoTen());
            txtEmail.setText(value: selectedNV.getEmail());
            txtNamSinh.setText(value: selectedNV.getNamSinh());
            txtSoDT.setText(value: selectedNV.getSoDT());
            txtDiaChi.setText(value: selectedNV.getDiaChi());

            boolean isMale = selectedNV.getGioiTinh() == 1;
            radMale.setSelected(value: isMale);
            radFemale.setSelected(!isMale);

            if (selectedNV.getHinh() != null) {
                ByteArrayInputStream bis = new ByteArrayInputStream(buf:selectedNV.getHinh());
                Image img = new Image(in: bis);
                ImageView imageView = new ImageView(image: img);
                imageView.setFitWidth(value: 100);
                imageView.setFitHeight(value: 100);
                lblImage.setGraphic(value: imageView);
                personalImage = selectedNV.getHinh();
            } else {
                personalImage = null;
                String imagePath = "/image/user.png";
                URL imageUrl = getClass().getResource(name: imagePath);
                if (imageUrl == null) {
                    System.err.println("Image not found at path: " + imagePath);
                } else {
                    Image img = new Image(string:imageUrl.toString());
                    ImageView imageView = new ImageView(image: img);
                    imageView.setFitWidth(value: 100);
                    imageView.setFitHeight(value: 100);
```



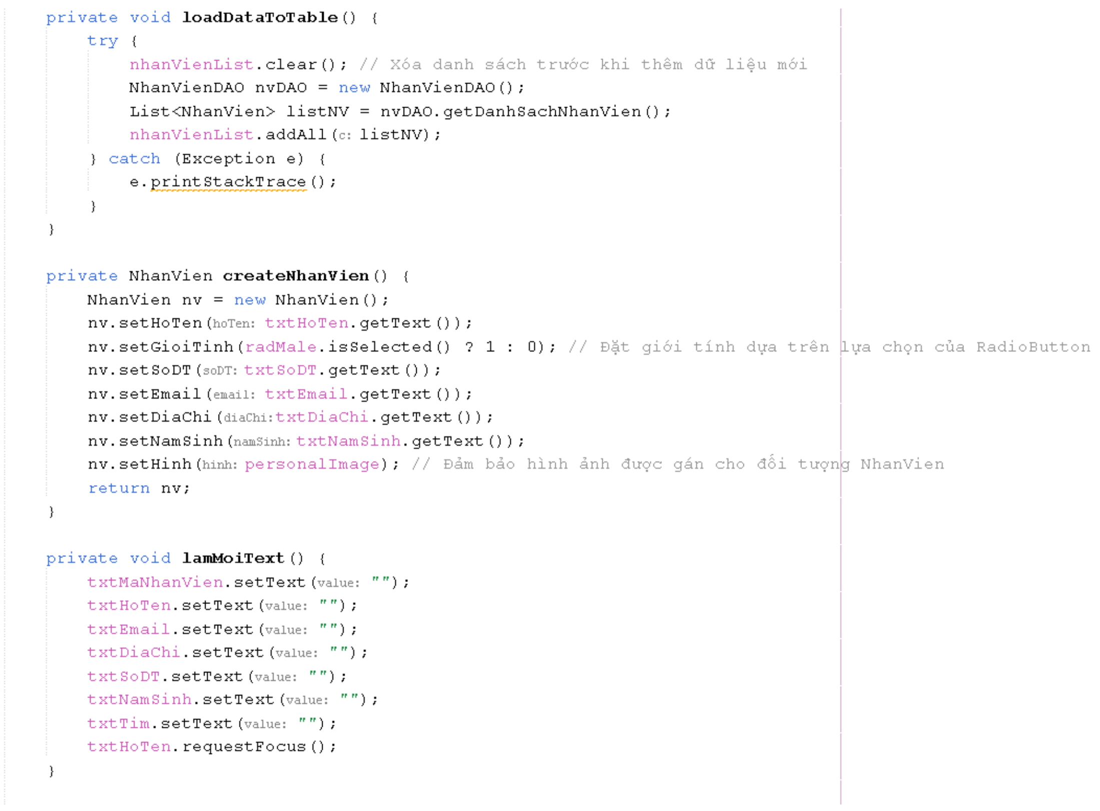
```
@FXML
void btnCapNhat(ActionEvent event) {
    int selectedIndex = tableview.getSelectionModel().getSelectedIndex();

    if (selectedIndex >= 0) {
        NhanVien selectedNV = tableview.getSelectionModel().getSelectedItem();

        StringBuilder sb = new StringBuilder();
        dataValidate(sb);

        if (sb.length() > 0) {
            MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Lỗi", content: sb.toString());
            return;
        }

        int confirm = MessageDialogHelpers.showConfirmDialog(parent:mainFrame, title: "Cảnh báo", content: "Bạn có chắc muốn cập :
        if (confirm == 1) {
            return;
        }

        try {
            NhanVien updatedNV = createNhanVien();
            updatedNV.setMaNhanVien(maNhanVien: selectedNV.getMaNhanVien()); // Sử dụng mã nhân viên của nhân viên đã chọn

            // Khởi tạo đối tượng NhanVienDAO
            NhanVienDAO nhanVienDAO = new NhanVienDAO();

            // Gọi phương thức updateNhanVien từ đối tượng nhanVienDAO
            if (nhanVienDAO.updateNhanVien(nv: updatedNV)) {
                MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Thông báo", content: "Nhân viên đã cập nhật thàn
                loadDataToTable();
                refreshComboBox();
            } else {
                MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Lỗi", content: "Cập nhật không thành công");
                loadDataToTable();
            }
```



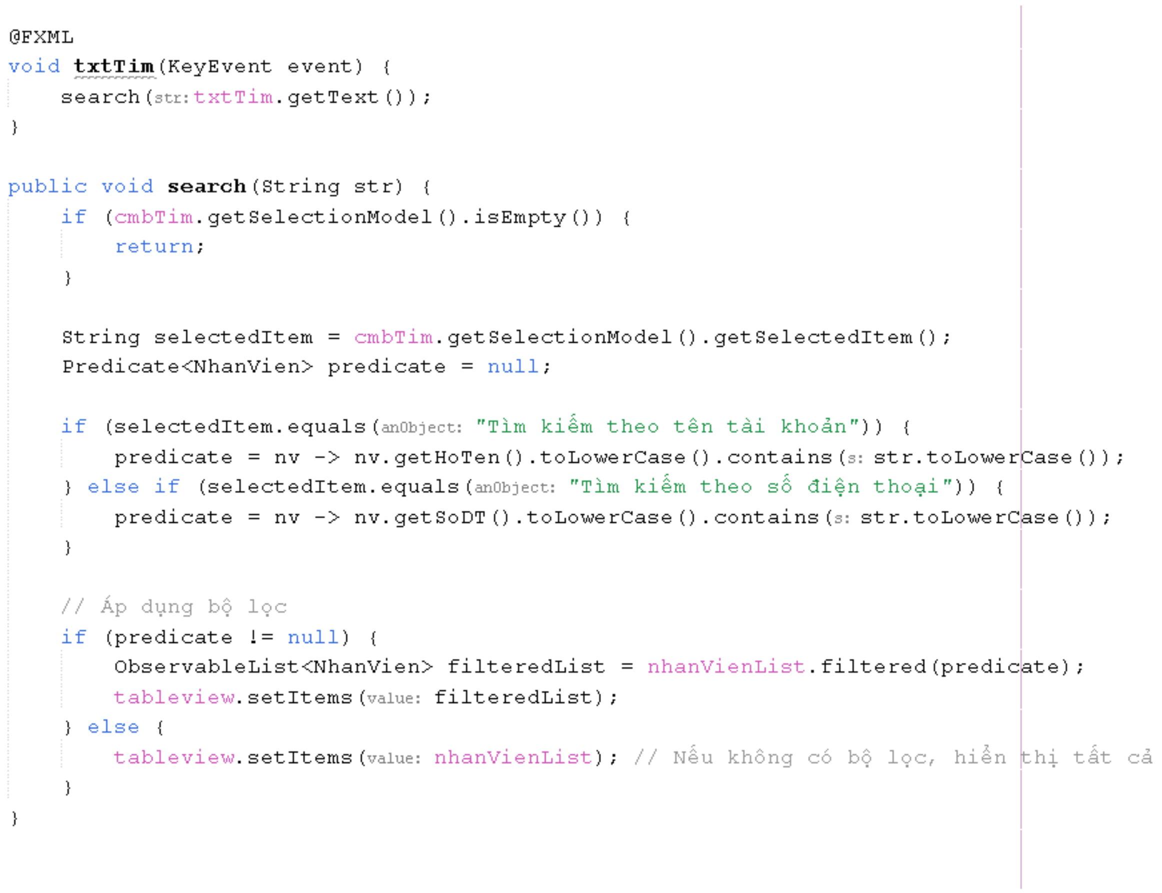
```
private void refreshComboBox() {
    cmbTim.getItems().clear(); // Xóa danh sách các mục trong ComboBox
    cmbTim.getItems().add(e: "Tìm kiếm theo tên tài khoản");
    cmbTim.getItems().add(e: "Tìm kiếm theo số điện thoại");
}

@FXML
void btnThemNV(ActionEvent event) {
    NhanVien nv = createNhanVien();
    NhanVienDAO nhanVienDAO = new NhanVienDAO();

    StringBuilder sb = new StringBuilder();
    dataValidate(sb);

    if (sb.length() > 0) {
        MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Lỗi", content: sb.toString());
        return;
    }

    if (nhanVienDAO.checkExist(email: txtEmail.getText().trim(), sdt:txtSoDT.getText().trim())) {
        MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Lỗi", content: "Trùng số điện thoại hoặc email");
        loadDataToTable();
        return;
    }

    if (nhanVienDAO.addNhanVien(nv)) {
        MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Thông báo", content: "Nhân viên đã thêm thành công");
        loadDataToTable();
        lamMoiText();
        cmbTim.getItems().clear(); // Xóa danh sách các mục trong ComboBox
        refreshComboBox();
    } else {
        MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Lỗi", content: "Thêm không thành công");
    }
}
```



```
private void initTable() {
    TableColumn<NhanVien, String> colMaNhanVien = new TableColumn<>(string: "Mã nhân viên");
    colMaNhanVien.setCellValueFactory(new PropertyValueFactory<>(string: "maNhanVien"));

    TableColumn<NhanVien, String> colHoTen = new TableColumn<>(string: "Họ tên");
    colHoTen.setCellValueFactory(new PropertyValueFactory<>(string: "hoTen"));

    TableColumn<NhanVien, String> colEmail = new TableColumn<>(string: "Email");
    colEmail.setCellValueFactory(new PropertyValueFactory<>(string: "email"));

    TableColumn<NhanVien, String> colSoDT = new TableColumn<>(string: "Số điện thoại");
    colSoDT.setCellValueFactory(new PropertyValueFactory<>(string: "soDT"));

    TableColumn<NhanVien, String> colNamSinh = new TableColumn<>(string: "Năm sinh");
    colNamSinh.setCellValueFactory(new PropertyValueFactory<>(string: "namSinh"));

    TableColumn<NhanVien, String> colGioiTinh = new TableColumn<>(string: "Giới tính");
    colGioiTinh.setCellValueFactory(cellData -> {
        String gender = cellData.getValue().getGioiTinh() == 1 ? "Nam" : "Nữ";
        return new javafx.beans.property.SimpleStringProperty(string: gender);
    });

    TableColumn<NhanVien, String> colDiaChi = new TableColumn<>(string: "Địa chỉ");
    colDiaChi.setCellValueFactory(new PropertyValueFactory<>(string: "diaChi"));

    tableview.getColumns().addAll(elements: colMaNhanVien, elements: colHoTen, elements: colEmail, elements: colSoDT, elements: colNamSin

    nhanVienList = FXCollections.observableArrayList();
    tableview.setItems(value: nhanVienList);
}
```

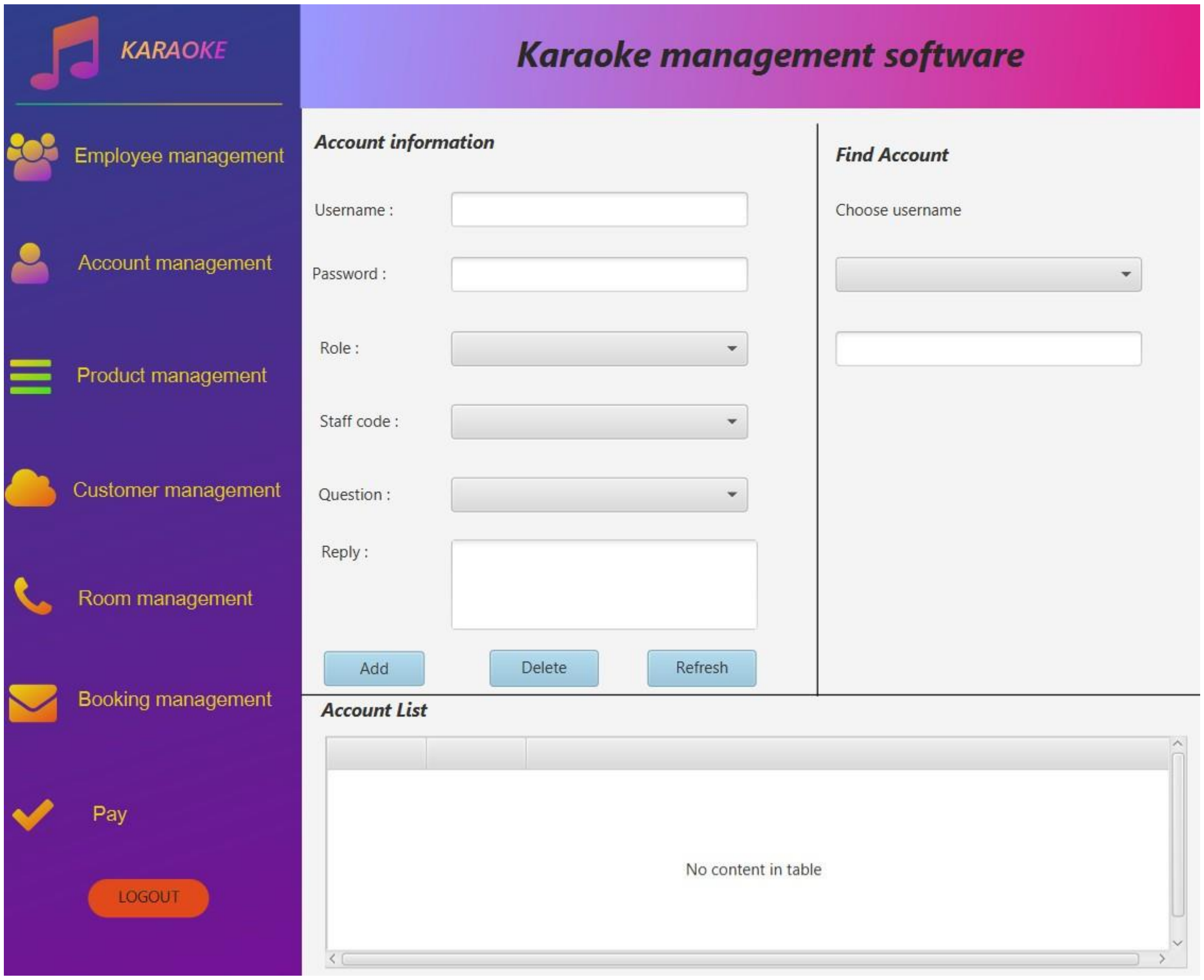


```java
private void loadDataToTable() {
    try {
        nhanVienList.clear(); // Xóa danh sách trước khi thêm dữ liệu mới
        NhanVienDAO nvDAO = new NhanVienDAO();
        List<NhanVien> listNV = nvDAO.getDanhSachNhanVien();
        nhanVienList.addAll(c: listNV);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private NhanVien createNhanVien() {
    NhanVien nv = new NhanVien();
    nv.setHoTen(hoTen: txtHoTen.getText());
    nv.setGioiTinh(radMale.isSelected() ? 1 : 0); // Đặt giới tính dựa trên lựa chọn của RadioButton
    nv.setSoDT(soDT: txtSoDT.getText());
    nv.setEmail(email: txtEmail.getText());
    nv.setDiaChi(diaChi:txtDiaChi.getText());
    nv.setNamSinh(namSinh: txtNamSinh.getText());
    nv.setHinh(hinh: personalImage); // Đảm bảo hình ảnh được gán cho đối tượng NhanVien
    return nv;
}

private void lamMoiText() {
    txtMaNhanVien.setText(value: "");
    txtHoTen.setText(value: "");
    txtEmail.setText(value: "");
    txtDiaChi.setText(value: "");
    txtSoDT.setText(value: "");
    txtNamSinh.setText(value: "");
    txtTim.setText(value: "");
    txtHoTen.requestFocus();
}
```



```
@FXML
void txtTim(KeyEvent event) {
    search(str: txtTim.getText());
}

public void search(String str) {
    if (cmbTim.getSelectionModel().isEmpty()) {
        return;
    }

    String selectedItem = cmbTim.getSelectionModel().getSelectedItem();
    Predicate<NhanVien> predicate = null;

    if (selectedItem.equals(anObject: "Tìm kiếm theo tên tài khoản")) {
        predicate = nv -> nv.getHoTen().toLowerCase().contains(s: str.toLowerCase());
    } else if (selectedItem.equals(anObject: "Tìm kiếm theo số điện thoại")) {
        predicate = nv -> nv.getSoDT().toLowerCase().contains(s: str.toLowerCase());
    }

    // Áp dụng bộ lọc
    if (predicate != null) {
        ObservableList<NhanVien> filteredList = nhanVienList.filtered(predicate);
        tableview.setItems(value: filteredList);
    } else {
        tableview.setItems(value: nhanVienList); // Nếu không có bộ lọc, hiển thị tất cả
    }
}
```

2. Account management



1: Click the Account Management button to display the account management interface

2: Fields used to enter account information.

3: The table displays the accounts list

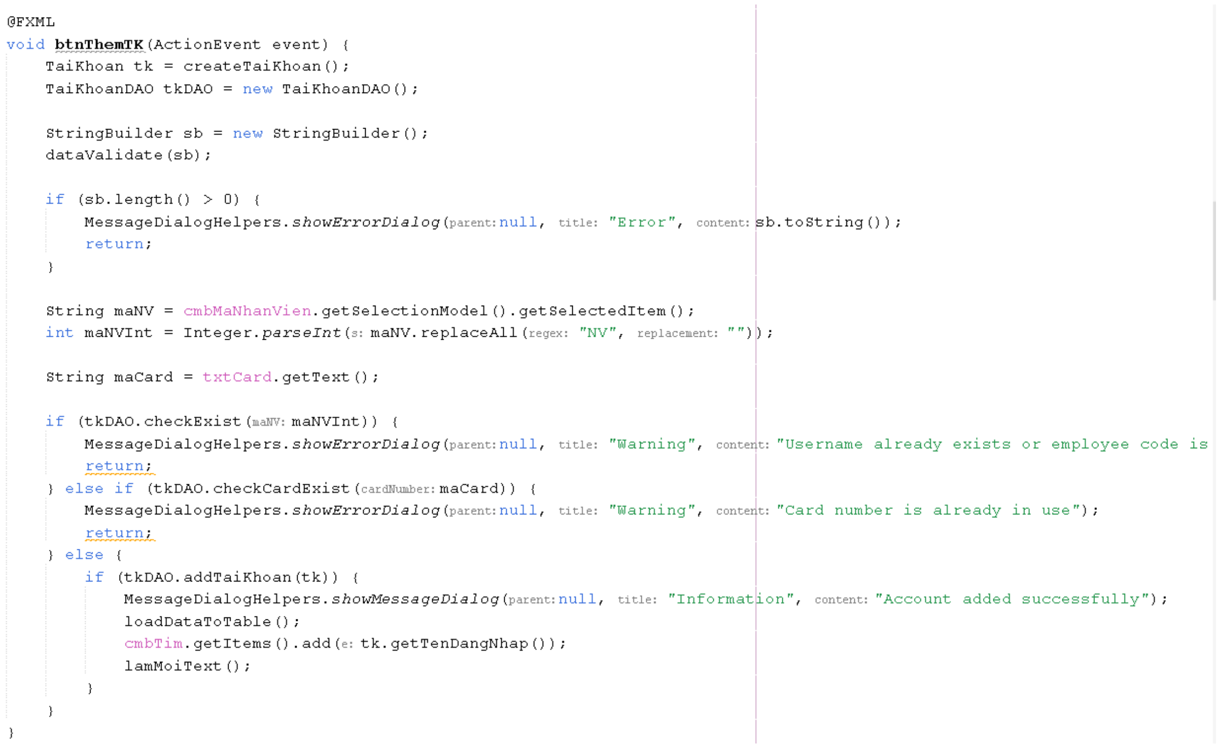
4: Enter employee account information , press the "Add" button to add an employee account

5: Select the employee account to delete in the employee account list table , press the "Delete" button to delete the employee account

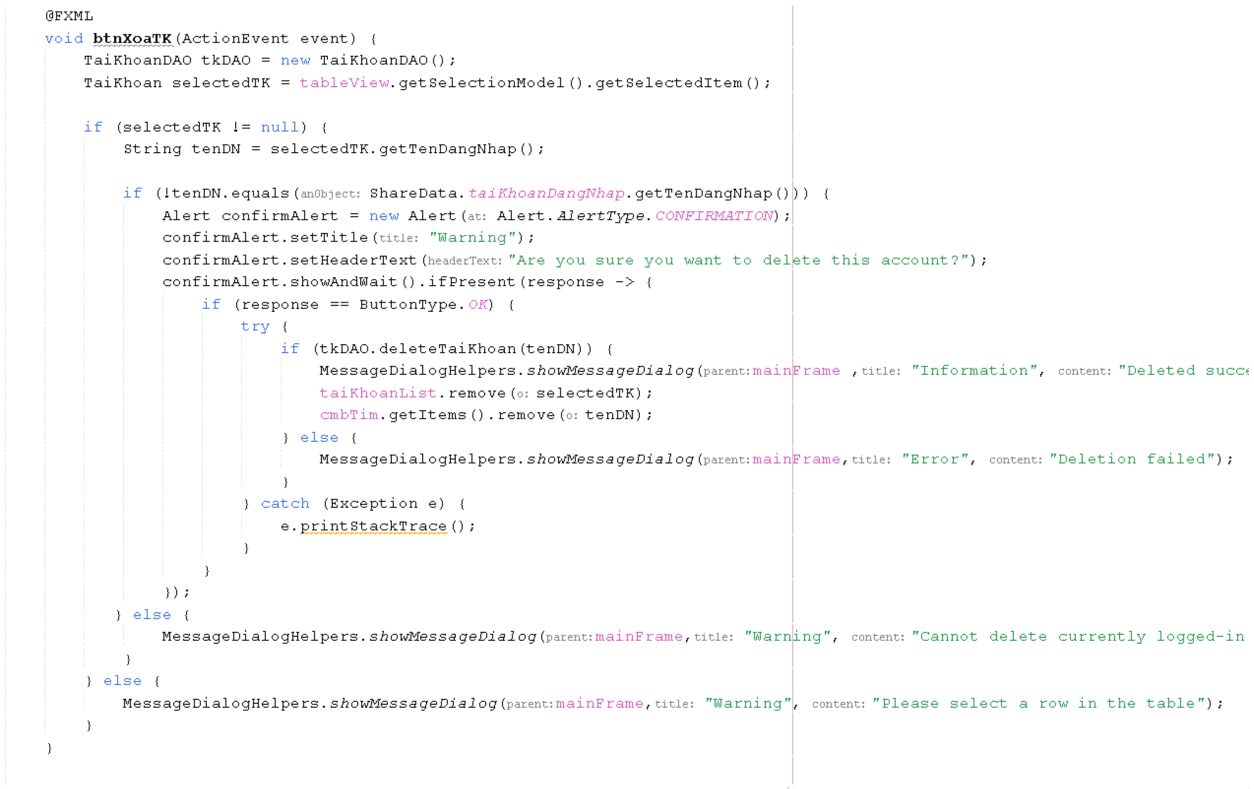
6: Click the “Refresh” button to refresh the input fields and account list

7: Select the login name in the combo box and press the Text Field, the employee account information will be displayed in the employee account list

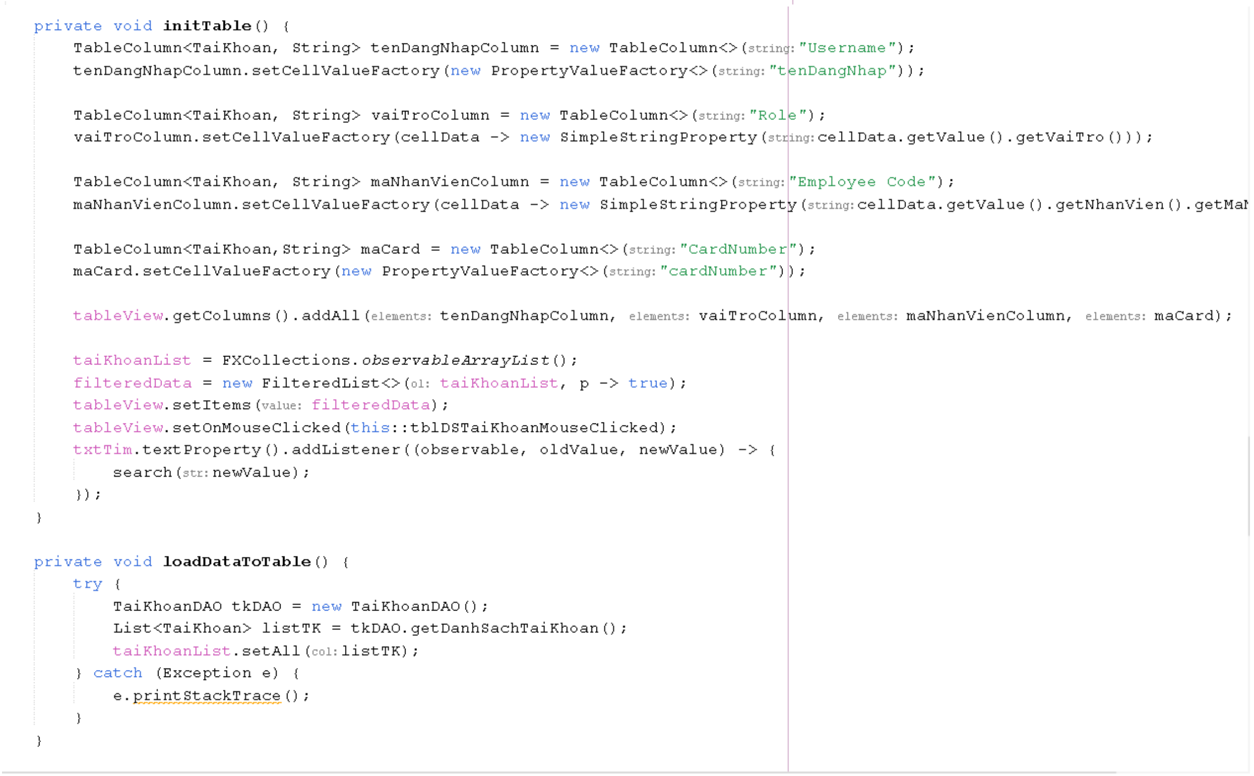
- **Validate:**
  - User Name : username cannot blank, The login name must not contain spaces or special characters
  - Password : password cannot blank ,Password must have at least 1 lowercase letter, 1 uppercase letter, 1 special character, no spaces and at least 8 characters
  - Role : cannot blank



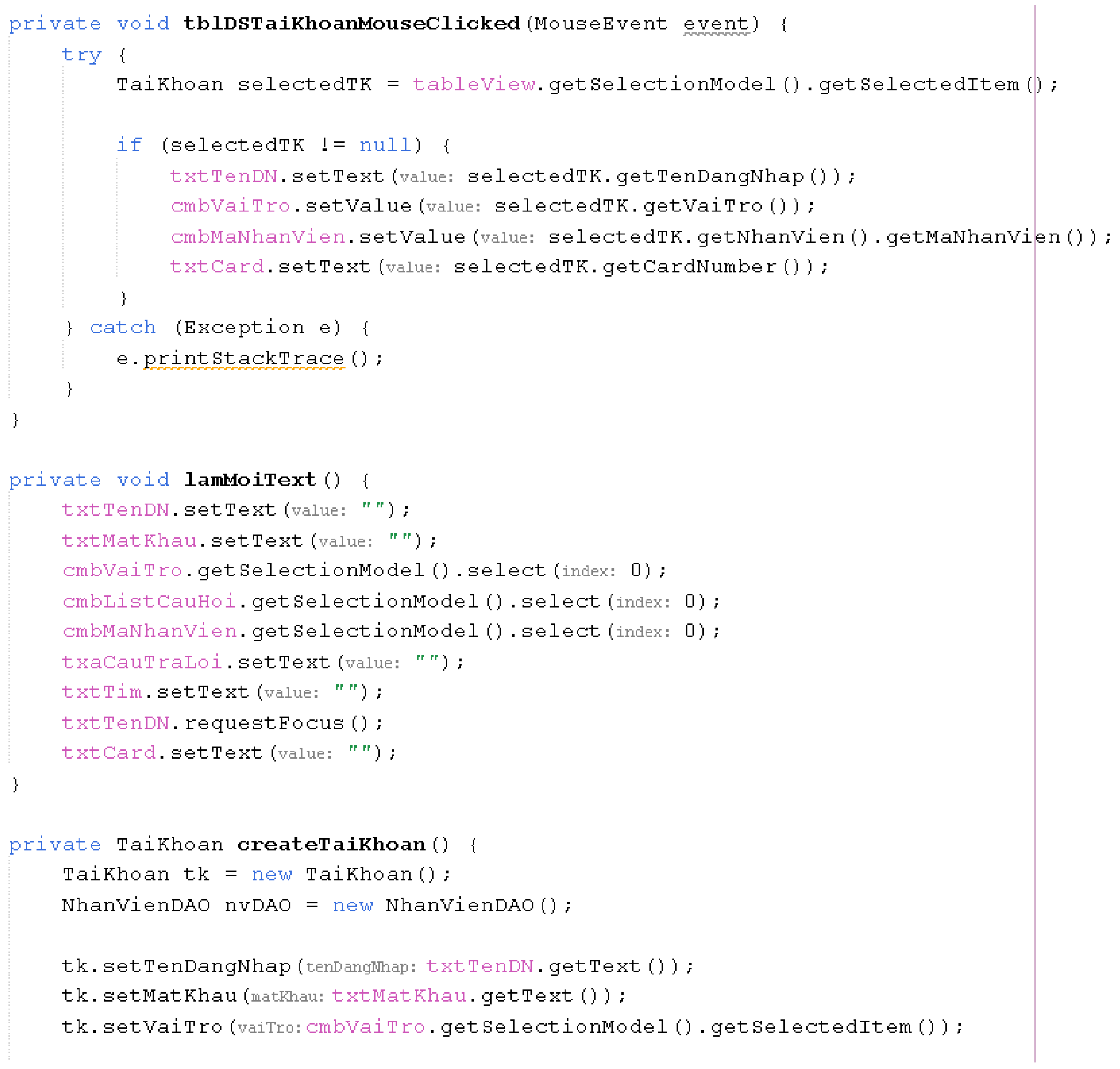
```
@FXML
void btnThemTK(ActionEvent event) {
    TaiKhoan tk = createTaiKhoan();
    TaiKhoanDAO tkDAO = new TaiKhoanDAO();

    StringBuilder sb = new StringBuilder();
    dataValidate(sb);

    if (sb.length() > 0) {
        MessageDialogHelpers.showErrorDialog(parent: null, title: "Error", content: sb.toString());
        return;
    }

    String maNV = cmbMaNhanVien.getSelectionModel().getSelectedItem();
    int maNVInt = Integer.parseInt(s: maNV.replaceAll(regex: "NV", replacement: ""));

    String maCard = txtCard.getText();

    if (tkDAO.checkExist(maNV: maNVInt)) {
        MessageDialogHelpers.showErrorDialog(parent: null, title: "Warning", content: "Username already exists or employee code is
        return;
    } else if (tkDAO.checkCardExist(cardNumber: maCard)) {
        MessageDialogHelpers.showErrorDialog(parent: null, title: "Warning", content: "Card number is already in use");
        return;
    } else {
        if (tkDAO.addTaiKhoan(tk)) {
            MessageDialogHelpers.showMessageDialog(parent: null, title: "Information", content: "Account added successfully");
            loadDataToTable();
            cmbTim.getItems().add(e: tk.getTenDangNhap());
            lamMoiText();
        }
    }
}
```



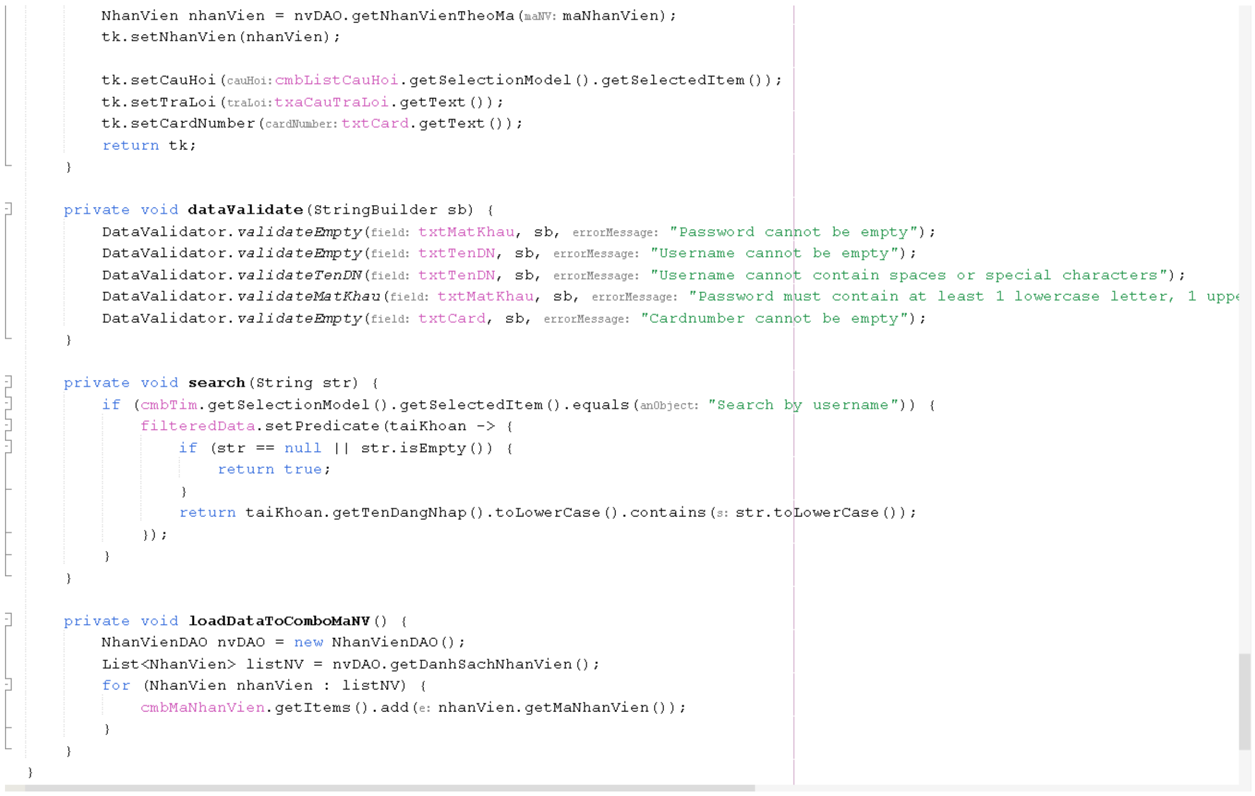
```
@FXML
void btnXoaTK(ActionEvent event) {
    TaiKhoanDAO tkDAO = new TaiKhoanDAO();
    TaiKhoan selectedTK = tableView.getSelectionModel().getSelectedItem();

    if (selectedTK != null) {
        String tenDN = selectedTK.getTenDangNhap();

        if (!tenDN.equals(anObject: ShareData.taiKhoanDangNhap.getTenDangNhap())) {
            Alert confirmAlert = new Alert(at: Alert.AlertType.CONFIRMATION);
            confirmAlert.setTitle(title: "Warning");
            confirmAlert.setHeaderText(headerText: "Are you sure you want to delete this account?");
            confirmAlert.showAndWait().ifPresent(response -> {
                if (response == ButtonType.OK) {
                    try {
                        if (tkDAO.deleteTaiKhoan(tenDN)) {
                            MessageDialogHelpers.showMessageDialog(parent:mainFrame ,title: "Information", content: "Deleted succe
                            taiKhoanList.remove(o: selectedTK);
                            cmbTim.getItems().remove(o: tenDN);
                        } else {
                            MessageDialogHelpers.showMessageDialog(parent:mainFrame,title: "Error", content: "Deletion failed");
                        }
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
        } else {
            MessageDialogHelpers.showMessageDialog(parent:mainFrame,title: "Warning", content: "Cannot delete currently logged-in
        }
    } else {
        MessageDialogHelpers.showMessageDialog(parent:mainFrame,title: "Warning", content: "Please select a row in the table");
    }
}
```



```
private void initTable() {
    TableColumn<TaiKhoan, String> tenDangNhapColumn = new TableColumn<>(string:"Username");
    tenDangNhapColumn.setCellValueFactory(new PropertyValueFactory<>(string:"tenDangNhap"));

    TableColumn<TaiKhoan, String> vaiTroColumn = new TableColumn<>(string:"Role");
    vaiTroColumn.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getVaiTro()));

    TableColumn<TaiKhoan, String> maNhanVienColumn = new TableColumn<>(string:"Employee Code");
    maNhanVienColumn.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getNhanVien().getMaN

    TableColumn<TaiKhoan,String> maCard = new TableColumn<>(string:"CardNumber");
    maCard.setCellValueFactory(new PropertyValueFactory<>(string:"cardNumber"));

    tableView.getColumns().addAll(elements: tenDangNhapColumn, elements: vaiTroColumn, elements: maNhanVienColumn, elements: maCard);

    taiKhoanList = FXCollections.observableArrayList();
    filteredData = new FilteredList<>(ol: taiKhoanList, p -> true);
    tableView.setItems(value: filteredData);
    tableView.setOnMouseClicked(this::tblDSTaiKhoanMouseClicked);
    txtTim.textProperty().addListener((observable, oldValue, newValue) -> {
        search(str:newValue);
    });
}

private void loadDataToTable() {
    try {
        TaiKhoanDAO tkDAO = new TaiKhoanDAO();
        List<TaiKhoan> listTK = tkDAO.getDanhSachTaiKhoan();
        taiKhoanList.setAll(col:listTK);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

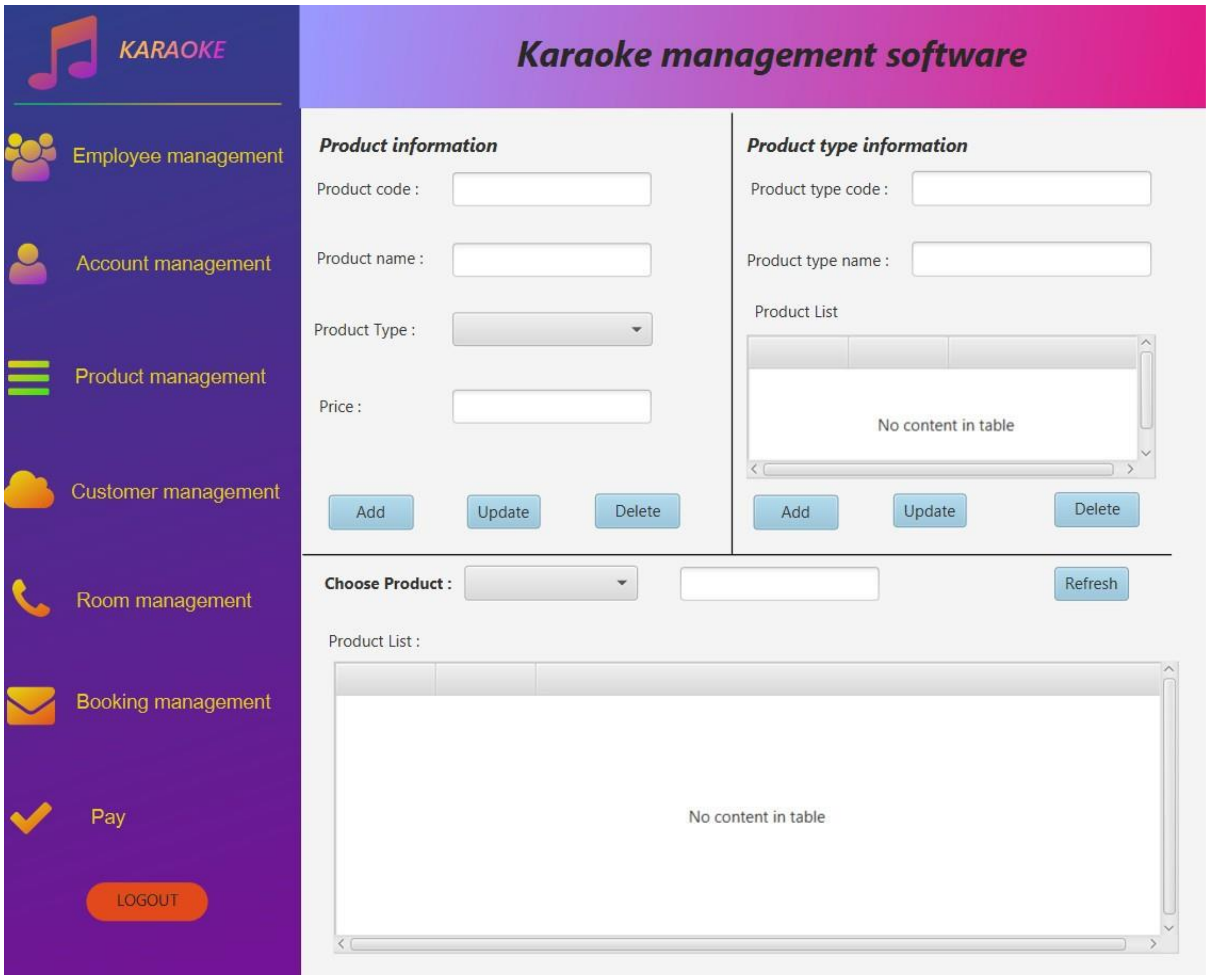


```
private void tblDSTaiKhoanMouseClicked(MouseEvent event) {
    try {
        TaiKhoan selectedTK = tableView.getSelectionModel().getSelectedItem();

        if (selectedTK != null) {
            txtTenDN.setText(value: selectedTK.getTenDangNhap());
            cmbVaiTro.setValue(value: selectedTK.getVaiTro());
            cmbMaNhanVien.setValue(value: selectedTK.getNhanVien().getMaNhanVien());
            txtCard.setText(value: selectedTK.getCardNumber());
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void lamMoiText() {
    txtTenDN.setText(value: "");
    txtMatKhau.setText(value: "");
    cmbVaiTro.getSelectionModel().select(index: 0);
    cmbListCauHoi.getSelectionModel().select(index: 0);
    cmbMaNhanVien.getSelectionModel().select(index: 0);
    txaCauTraLoi.setText(value: "");
    txtTim.setText(value: "");
    txtTenDN.requestFocus();
    txtCard.setText(value: "");
}

private TaiKhoan createTaiKhoan() {
    TaiKhoan tk = new TaiKhoan();
    NhanVienDAO nvDAO = new NhanVienDAO();

    tk.setTenDangNhap(tenDangNhap: txtTenDN.getText());
    tk.setMatKhau(matKhau: txtMatKhau.getText());
    tk.setVaiTro(vaiTro: cmbVaiTro.getSelectionModel().getSelectedItem());
```



```
        NhanVien nhanVien = nvDAO.getNhanVienTheoMa(maNV: maNhanVien);
        tk.setNhanVien(nhanVien);

        tk.setCauHoi(cauHoi:cmbListCauHoi.getSelectionModel().getSelectedItem());
        tk.setTraLoi(traLoi:txaCauTraLoi.getText());
        tk.setCardNumber(cardNumber:txtCard.getText());
        return tk;
    }

    private void dataValidate(StringBuilder sb) {
        DataValidator.validateEmpty(field: txtMatKhau, sb, errorMessage: "Password cannot be empty");
        DataValidator.validateEmpty(field: txtTenDN, sb, errorMessage: "Username cannot be empty");
        DataValidator.validateTenDN(field: txtTenDN, sb, errorMessage: "Username cannot contain spaces or special characters");
        DataValidator.validateMatKhau(field: txtMatKhau, sb, errorMessage: "Password must contain at least 1 lowercase letter, 1 uppe
        DataValidator.validateEmpty(field: txtCard, sb, errorMessage: "Cardnumber cannot be empty");
    }

    private void search(String str) {
        if (cmbTim.getSelectionModel().getSelectedItem().equals(anObject: "Search by username")) {
            filteredData.setPredicate(taiKhoan -> {
                if (str == null || str.isEmpty()) {
                    return true;
                }
                return taiKhoan.getTenDangNhap().toLowerCase().contains(s: str.toLowerCase());
            });
        }
    }

    private void loadDataToComboMaNV() {
        NhanVienDAO nvDAO = new NhanVienDAO();
        List<NhanVien> listNV = nvDAO.getDanhSachNhanVien();
        for (NhanVien nhanVien : listNV) {
            cmbMaNhanVien.getItems().add(e: nhanVien.getMaNhanVien());
        }
    }
}
```

3. Product Management



1: Click the "Product Management" button to display the product management interface

2: Fields used to enter product information.

3: Table displaying product list

4: Enter product information , press the "Add" button to add the product

5: Select a product to update in the product list , enter the information to edit and press the "Update" button to update the product.

6: Select a product to delete in the product list , press the "Delete" button to delete the product

7: Fields used to enter product type information.

8: Table displaying the list of product types

9: Enter product type information , press the "Add" button to add the product type

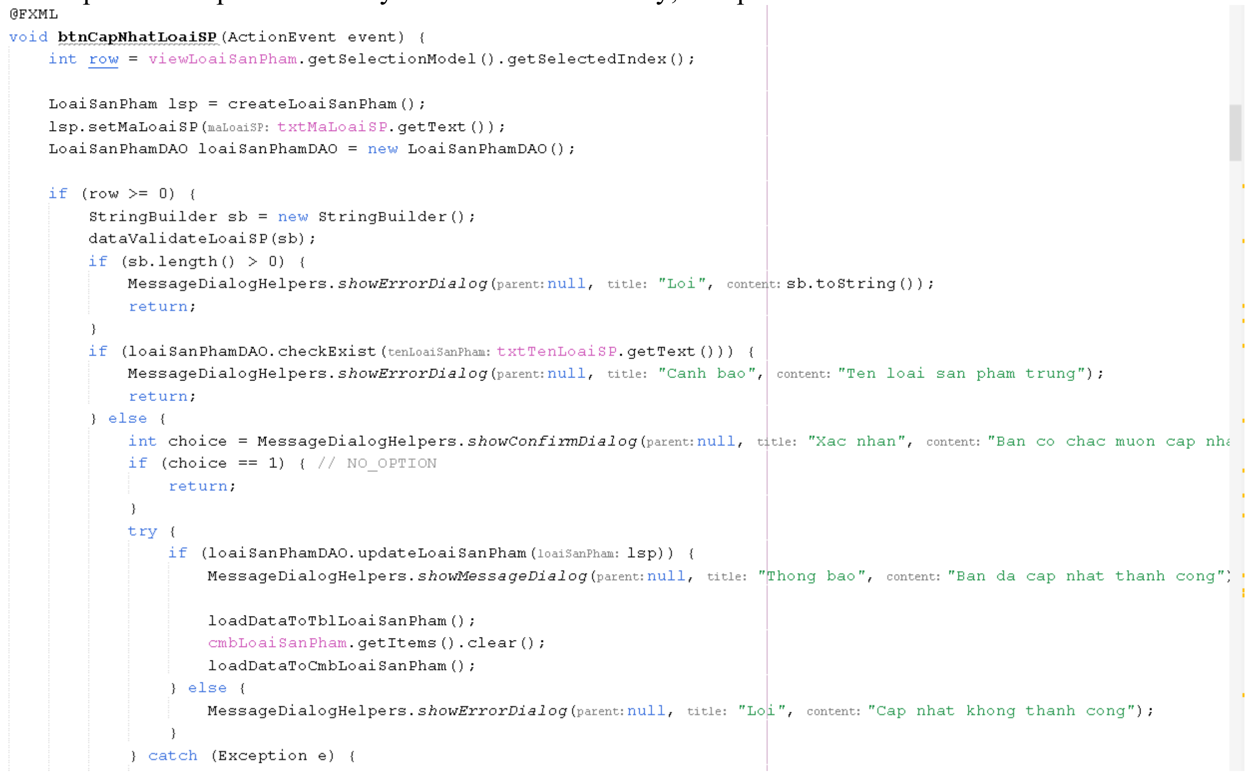
10: Select a product type to update in the product type list , enter the information to edit and click the "Update" button to update the product type.

11: Select a product type to delete in the product type list , press the "Delete" button to delete the product type

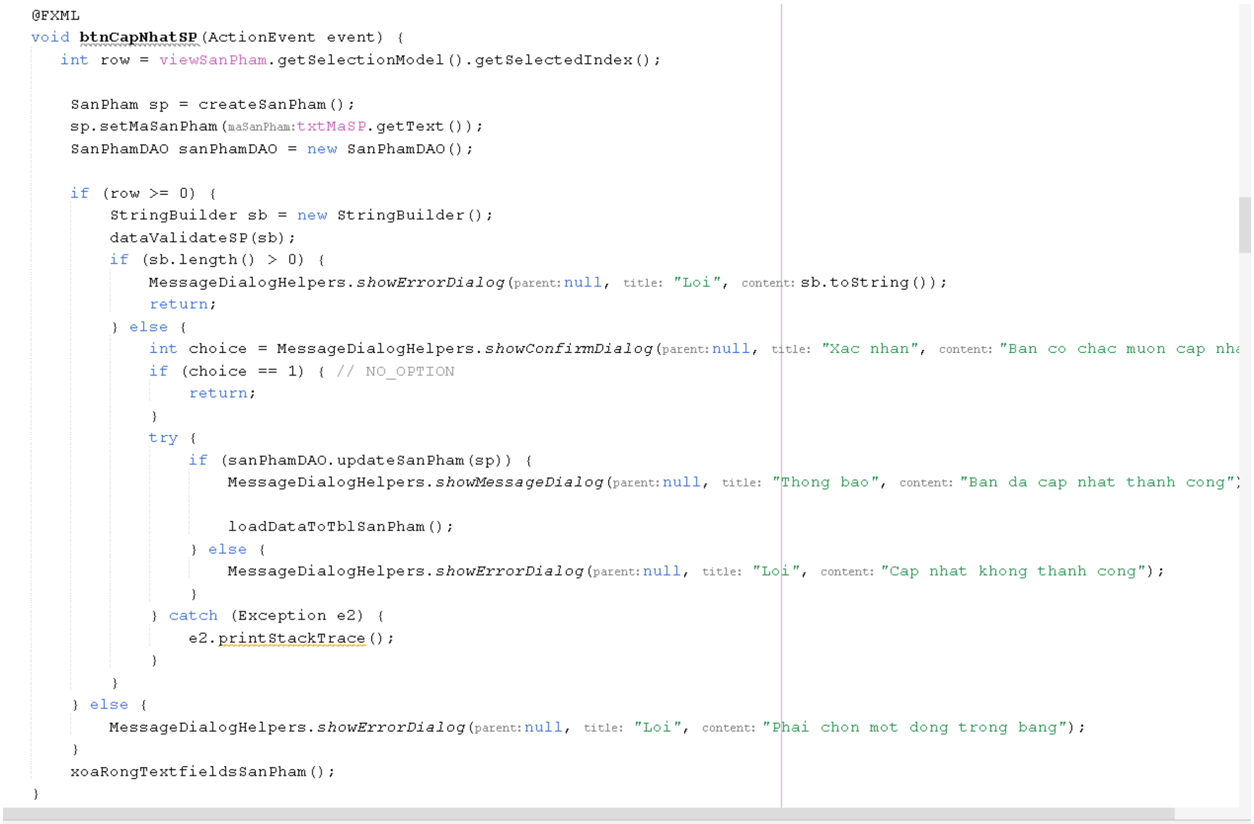
12: Click the “Refresh” button to refresh the input fields and product list

13: Select the product name in the combo box and press the Text Field , product information will be displayed in the product list

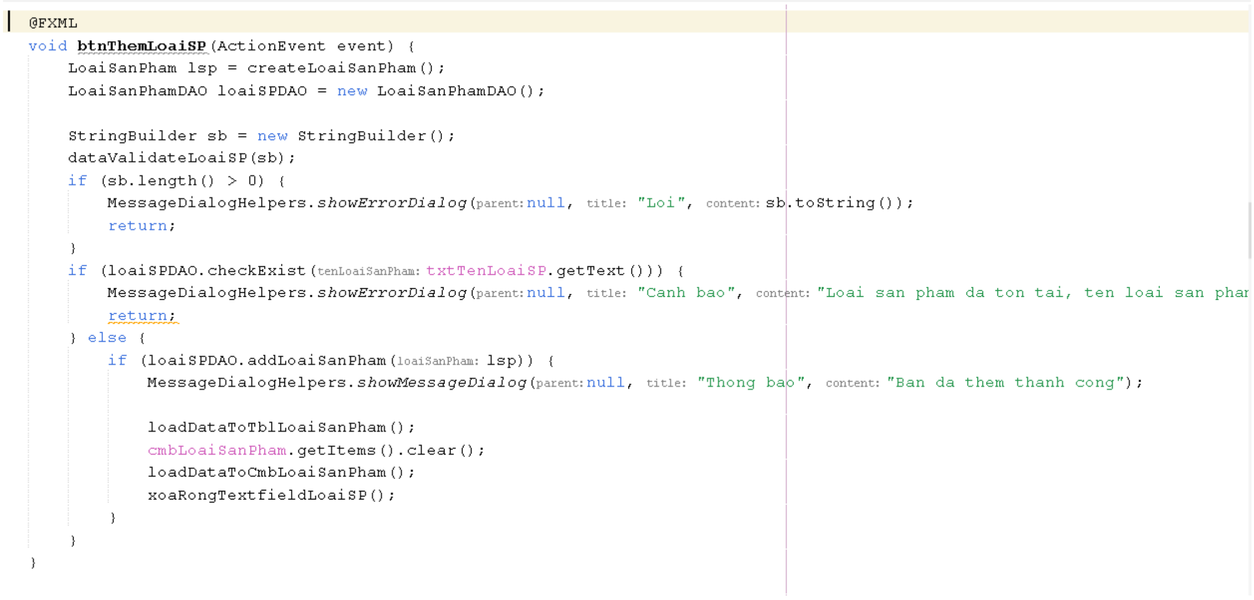
- **Validate:**
  - Product Name: Product type name cannot be blank
  - Product_Type: Product type name is wrong. Do not enter numbers and special characters, Product name cannot be blank
  - price: Unit prices can only be entered numerically,Unit price cannot be blank



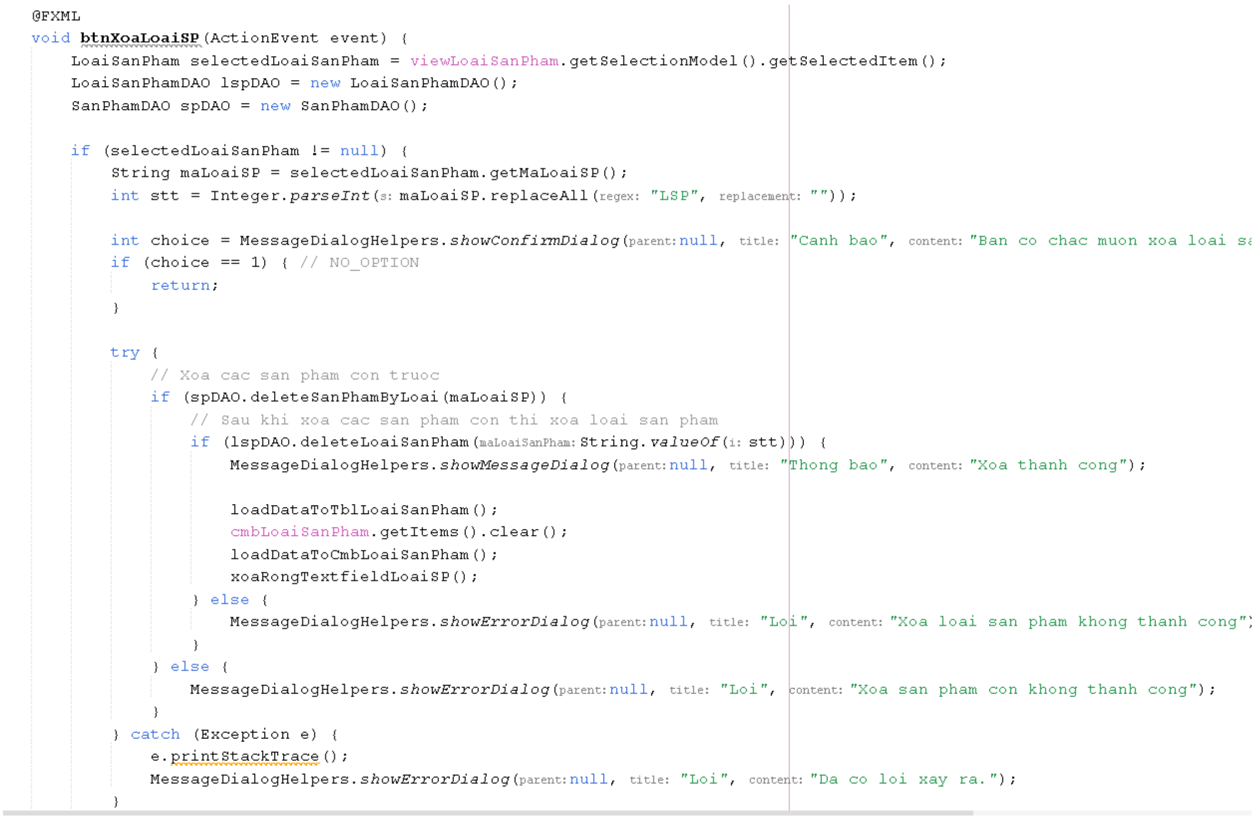
```
@FXML
void btnCapNhatLoaiSP(ActionEvent event) {
    int row = viewLoaiSanPham.getSelectionModel().getSelectedIndex();

    LoaiSanPham lsp = createLoaiSanPham();
    lsp.setMaLoaiSP(maLoaiSP: txtMaLoaiSP.getText());
    LoaiSanPhamDAO loaiSanPhamDAO = new LoaiSanPhamDAO();

    if (row >= 0) {
        StringBuilder sb = new StringBuilder();
        dataValidateLoaiSP(sb);
        if (sb.length() > 0) {
            MessageDialogHelpers.showErrorDialog(parent: null, title: "Loi", content: sb.toString());
            return;
        }
        if (loaiSanPhamDAO.checkExist(tenLoaiSanPham: txtTenLoaiSP.getText())) {
            MessageDialogHelpers.showErrorDialog(parent: null, title: "Canh bao", content: "Ten loai san pham trung");
            return;
        } else {
            int choice = MessageDialogHelpers.showConfirmDialog(parent: null, title: "Xac nhan", content: "Ban co chac muon cap nha
            if (choice == 1) { // NO_OPTION
                return;
            }
            try {
                if (loaiSanPhamDAO.updateLoaiSanPham(loaiSanPham: lsp)) {
                    MessageDialogHelpers.showMessageDialog(parent: null, title: "Thong bao", content: "Ban da cap nhat thanh cong")

                    loadDataToTblLoaiSanPham();
                    cmbLoaiSanPham.getItems().clear();
                    loadDataToCmbLoaiSanPham();
                } else {
                    MessageDialogHelpers.showErrorDialog(parent: null, title: "Loi", content: "Cap nhat khong thanh cong");
                }
            } catch (Exception e) {
```



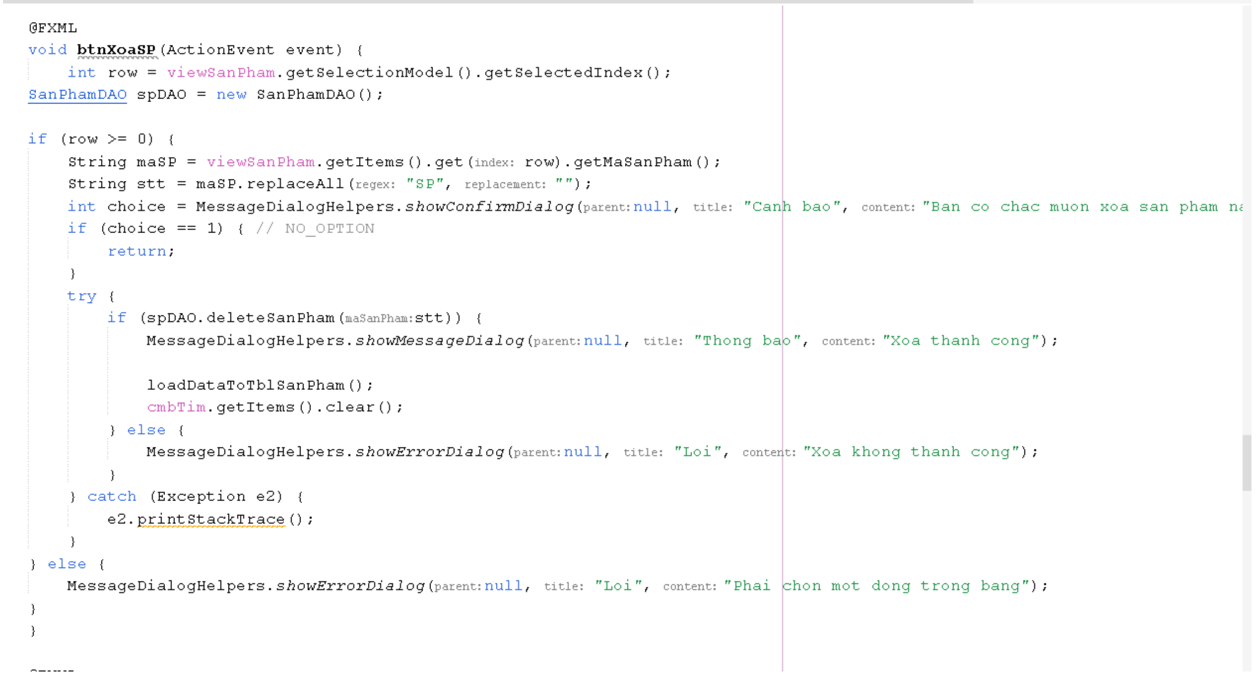
```
@FXML
void btnCapNhatSP(ActionEvent event) {
    int row = viewSanPham.getSelectionModel().getSelectedIndex();

     SanPham sp = createSanPham();
     sp.setMaSanPham(maSanPham:txtMaSP.getText());
     SanPhamDAO sanPhamDAO = new SanPhamDAO();

     if (row >= 0) {
         StringBuilder sb = new StringBuilder();
         dataValidateSP(sb);
         if (sb.length() > 0) {
             MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: sb.toString());
             return;
         } else {
             int choice = MessageDialogHelpers.showConfirmDialog(parent:null, title: "Xac nhan", content: "Ban co chac muon cap nha
             if (choice == 1) { // NO_OPTION
                 return;
             }
             try {
                 if (sanPhamDAO.updateSanPham(sp)) {
                     MessageDialogHelpers.showMessageDialog(parent:null, title: "Thong bao", content: "Ban da cap nhat thanh cong")

                     loadDataToTblSanPham();
                 } else {
                     MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: "Cap nhat khong thanh cong");
                 }
             } catch (Exception e2) {
                 e2.printStackTrace();
             }
         }
     } else {
         MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: "Phai chon mot dong trong bang");
     }
     xoaRongTextfieldsSanPham();
}
```



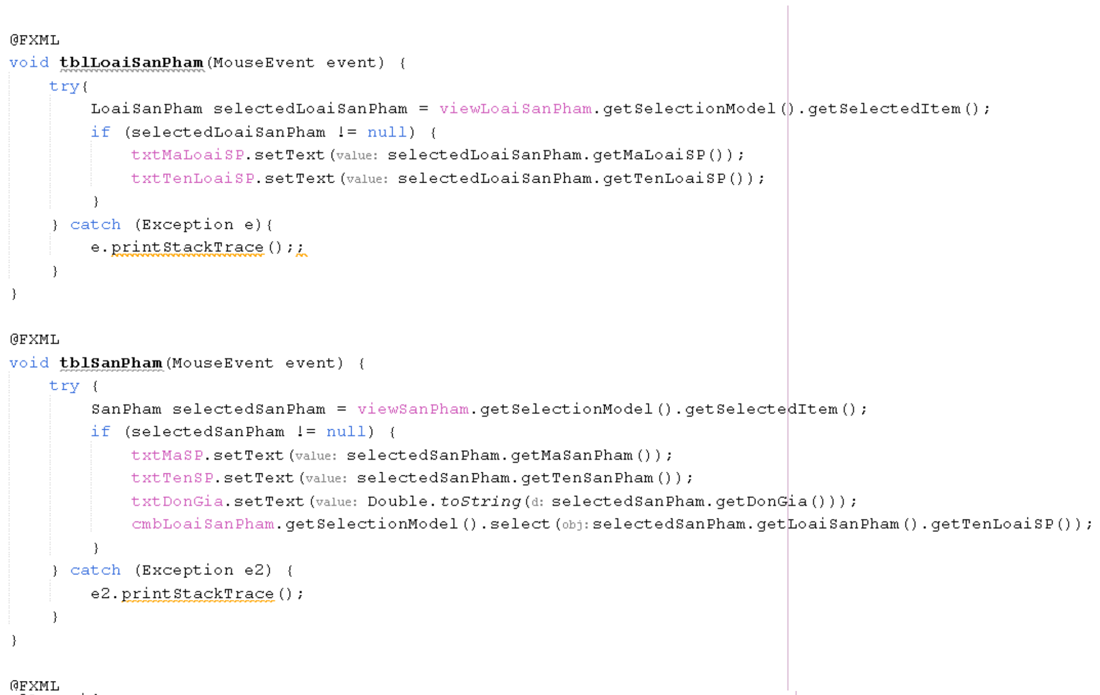
```
@FXML
void btnThemLoaiSP(ActionEvent event) {
    LoaiSanPham lsp = createLoaiSanPham();
    LoaiSanPhamDAO loaiSPDAO = new LoaiSanPhamDAO();

    StringBuilder sb = new StringBuilder();
    dataValidateLoaiSP(sb);
    if (sb.length() > 0) {
        MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: sb.toString());
        return;
    }
    if (loaiSPDAO.checkExist(tenLoaiSanPham: txtTenLoaiSP.getText())) {
        MessageDialogHelpers.showErrorDialog(parent:null, title: "Canh bao", content: "Loai san pham da ton tai, ten loai san phar
        return;
    } else {
        if (loaiSPDAO.addLoaiSanPham(loaiSanPham: lsp)) {
            MessageDialogHelpers.showMessageDialog(parent:null, title: "Thong bao", content: "Ban da them thanh cong");

            loadDataToTblLoaiSanPham();
            cmbLoaiSanPham.getItems().clear();
            loadDataToCmbLoaiSanPham();
            xoaRongTextfieldLoaiSP();
        }
    }
}
```



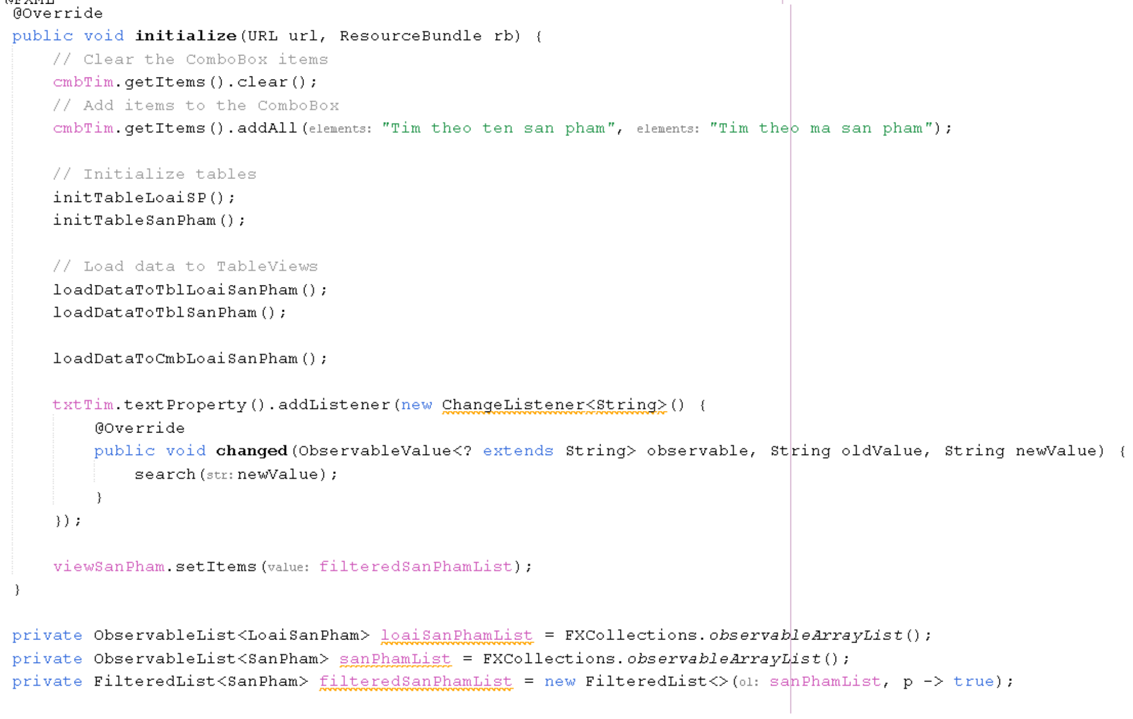
```
@FXML
void btnXoaLoaiSP(ActionEvent event) {
    LoaiSanPham selectedLoaiSanPham = viewLoaiSanPham.getSelectionModel().getSelectedItem();
    LoaiSanPhamDAO lspDAO = new LoaiSanPhamDAO();
    SanPhamDAO spDAO = new SanPhamDAO();

    if (selectedLoaiSanPham != null) {
        String maLoaiSP = selectedLoaiSanPham.getMaLoaiSP();
        int stt = Integer.parseInt(s: maLoaiSP.replaceAll(regex: "LSP", replacement: ""));

        int choice = MessageDialogHelpers.showConfirmDialog(parent:null, title: "Canh bao", content: "Ban co chac muon xoa loai sa
        if (choice == 1) { // NO_OPTION
            return;
        }

        try {
            // Xoa cac san pham con truoc
            if (spDAO.deleteSanPhamByLoai(maLoaiSP)) {
                // Sau khi xoa cac san pham con thi xoa loai san pham
                if (lspDAO.deleteLoaiSanPham(maLoaiSanPham: String.valueOf(i: stt))) {
                    MessageDialogHelpers.showMessageDialog(parent:null, title: "Thong bao", content: "Xoa thanh cong");

                    loadDataToTblLoaiSanPham();
                    cmbLoaiSanPham.getItems().clear();
                    loadDataToCmbLoaiSanPham();
                    xoaRongTextfieldLoaiSP();
                } else {
                    MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: "Xoa loai san pham khong thanh cong")
                }
            } else {
                MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: "Xoa san pham con khong thanh cong");
            }
        } catch (Exception e) {
            e.printStackTrace();
            MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: "Da co loi xay ra.");
        }
```



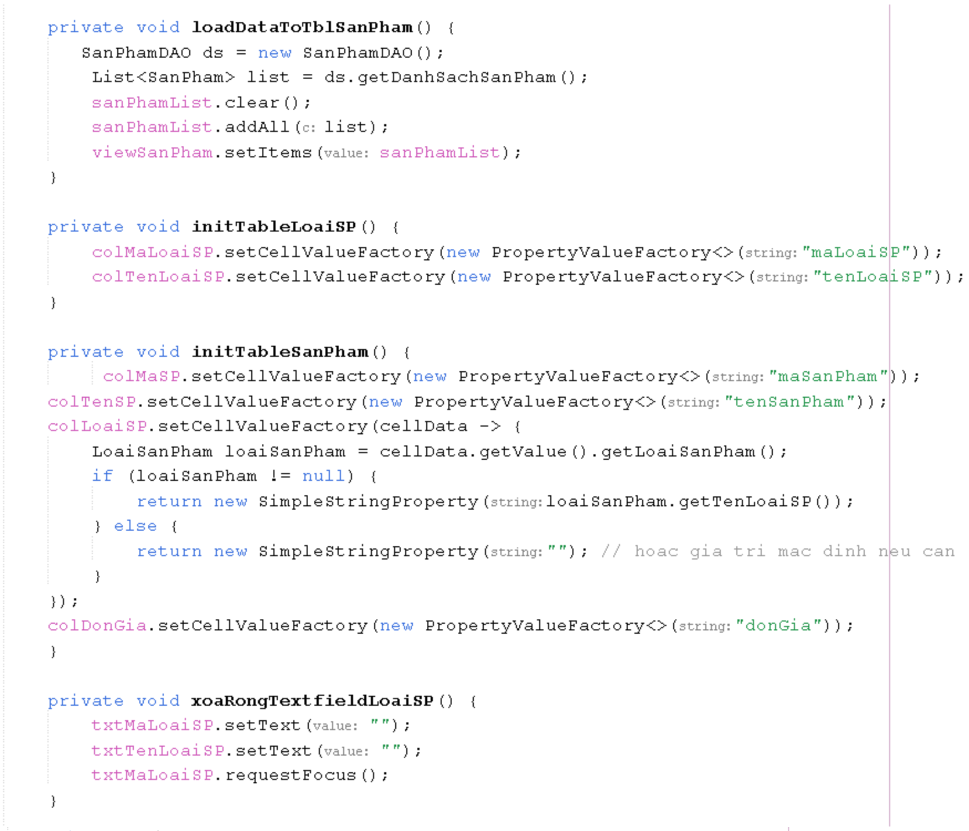
```
@FXML
void btnXoaSP(ActionEvent event) {
    int row = viewSanPham.getSelectionModel().getSelectedIndex();
SanPhamDAO spDAO = new SanPhamDAO();

if (row >= 0) {
    String maSP = viewSanPham.getItems().get(index: row).getMaSanPham();
    String stt = maSP.replaceAll(regex: "SP", replacement: "");
    int choice = MessageDialogHelpers.showConfirmDialog(parent:null, title: "Canh bao", content: "Ban co chac muon xoa san pham na
    if (choice == 1) { // NO_OPTION
        return;
    }
    try {
        if (spDAO.deleteSanPham(maSanPham:stt)) {
            MessageDialogHelpers.showMessageDialog(parent:null, title: "Thong bao", content: "Xoa thanh cong");

            loadDataToTblSanPham();
            cmbTim.getItems().clear();
        } else {
            MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: "Xoa khong thanh cong");
        }
    } catch (Exception e2) {
        e2.printStackTrace();
    }
} else {
    MessageDialogHelpers.showErrorDialog(parent:null, title: "Loi", content: "Phai chon mot dong trong bang");
}
}
```



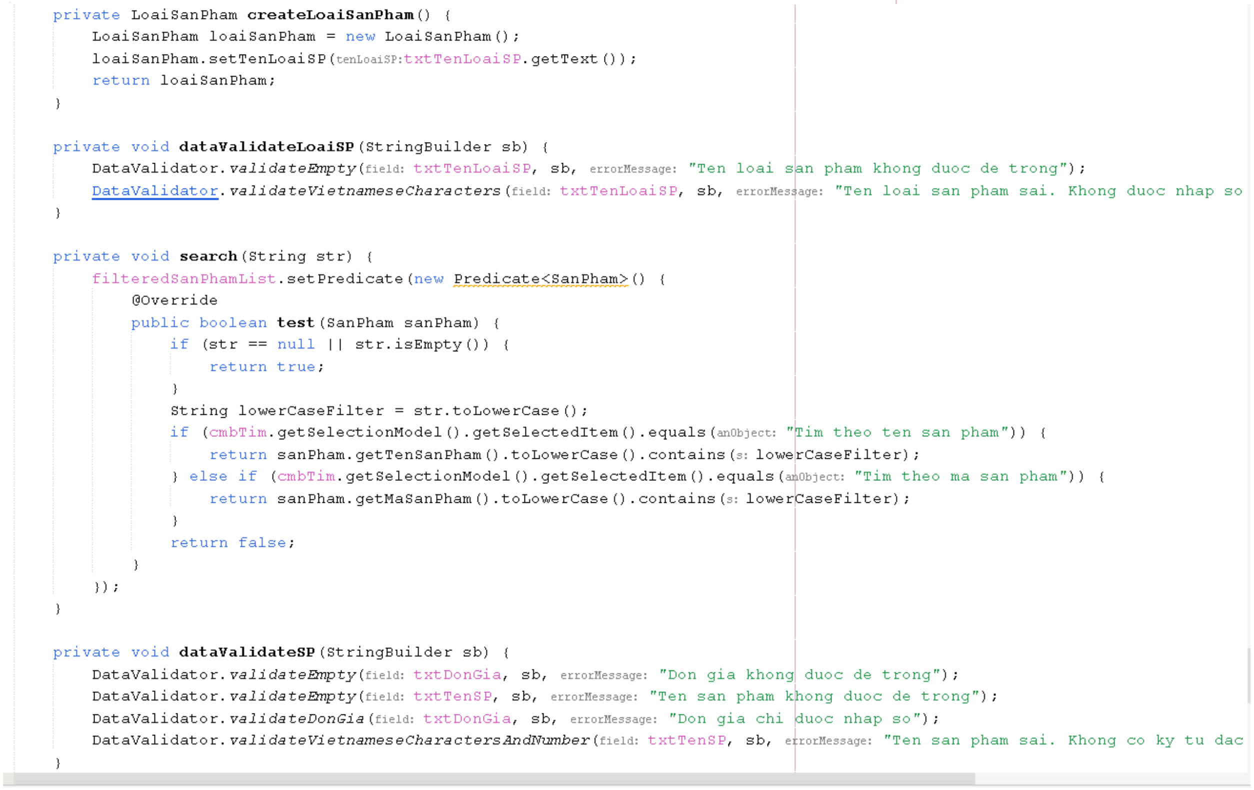
```java
@FXML
void tblLoaiSanPham(MouseEvent event) {
    try{
        LoaiSanPham selectedLoaiSanPham = viewLoaiSanPham.getSelectionModel().getSelectedItem();
        if (selectedLoaiSanPham != null) {
            txtMaLoaiSP.setText(value: selectedLoaiSanPham.getMaLoaiSP());
            txtTenLoaiSP.setText(value: selectedLoaiSanPham.getTenLoaiSP());
        }
    } catch (Exception e){
        e.printStackTrace();;
    }
}

@FXML
void tblSanPham(MouseEvent event) {
    try {
        SanPham selectedSanPham = viewSanPham.getSelectionModel().getSelectedItem();
        if (selectedSanPham != null) {
            txtMaSP.setText(value: selectedSanPham.getMaSanPham());
            txtTenSP.setText(value: selectedSanPham.getTenSanPham());
            txtDonGia.setText(value: Double.toString(d: selectedSanPham.getDonGia()));
            cmbLoaiSanPham.getSelectionModel().select(obj:selectedSanPham.getLoaiSanPham().getTenLoaiSP());
        }
    } catch (Exception e2) {
        e2.printStackTrace();
    }
}

@FXML
```



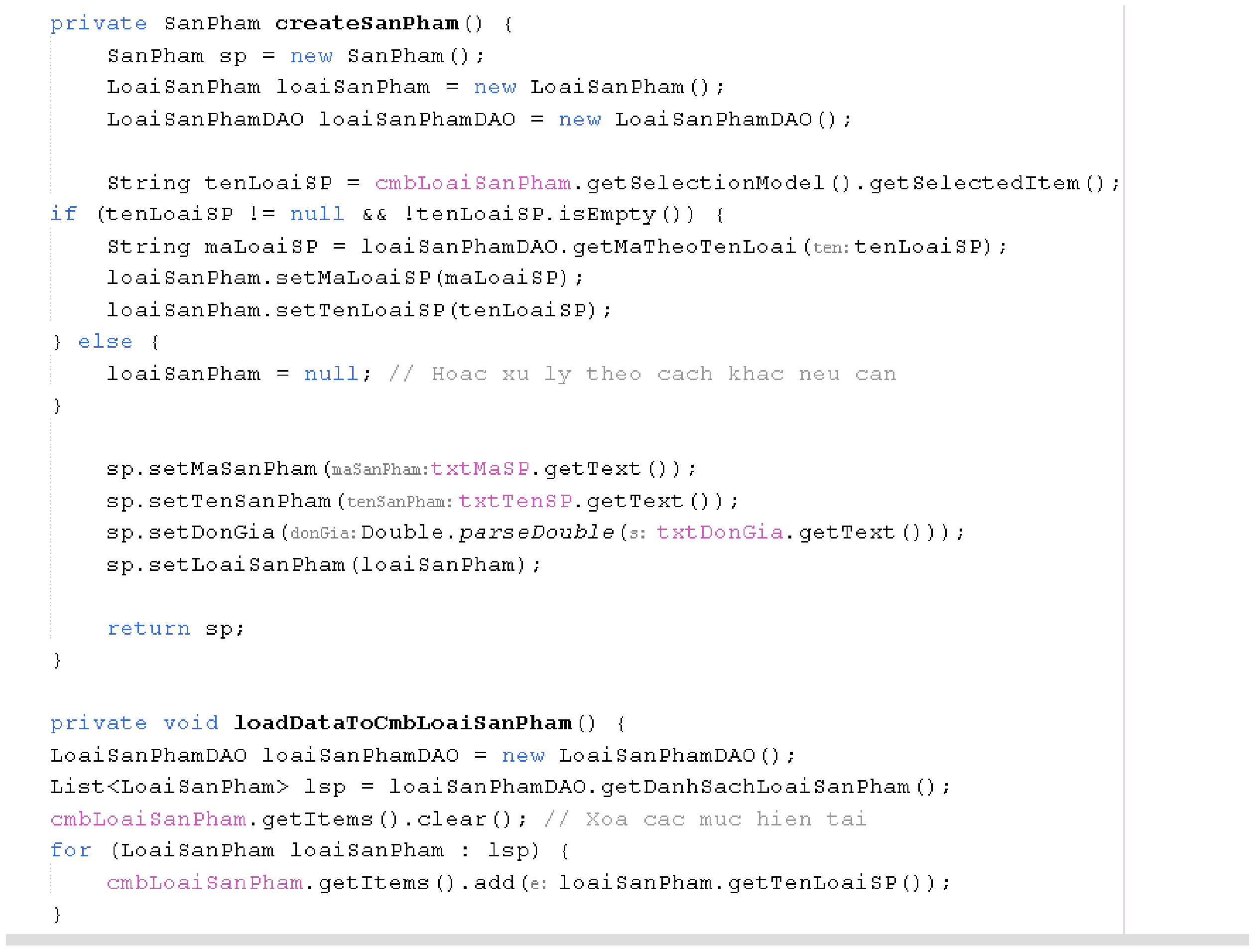
```java
 @Override
 public void initialize(URL url, ResourceBundle rb) {
     // Clear the ComboBox items
     cmbTim.getItems().clear();
     // Add items to the ComboBox
     cmbTim.getItems().addAll(elements: "Tim theo ten san pham", elements: "Tim theo ma san pham");

     // Initialize tables
     initTableLoaiSP();
     initTableSanPham();

     // Load data to TableViews
     loadDataToTblLoaiSanPham();
     loadDataToTblSanPham();

     loadDataToCmbLoaiSanPham();

     txtTim.textProperty().addListener(new ChangeListener<String>() {
         @Override
         public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
             search(str:newValue);
         }
     });

     viewSanPham.setItems(value: filteredSanPhamList);
 }

 private ObservableList<LoaiSanPham> loaiSanPhamList = FXCollections.observableArrayList();
 private ObservableList<SanPham> sanPhamList = FXCollections.observableArrayList();
 private FilteredList<SanPham> filteredSanPhamList = new FilteredList<>(c1: sanPhamList, p -> true);
```



```java
private void loadDataToTblSanPham() {
    SanPhamDAO ds = new SanPhamDAO();
    List<SanPham> list = ds.getDanhSachSanPham();
    sanPhamList.clear();
    sanPhamList.addAll(c: list);
    viewSanPham.setItems(value: sanPhamList);
}

private void initTableLoaiSP() {
    colMaLoaiSP.setCellValueFactory(new PropertyValueFactory<>(string: "maLoaiSP"));
    colTenLoaiSP.setCellValueFactory(new PropertyValueFactory<>(string: "tenLoaiSP"));
}

private void initTableSanPham() {
     colMaSP.setCellValueFactory(new PropertyValueFactory<>(string: "maSanPham"));
colTenSP.setCellValueFactory(new PropertyValueFactory<>(string: "tenSanPham"));
colLoaiSP.setCellValueFactory(cellData -> {
    LoaiSanPham loaiSanPham = cellData.getValue().getLoaiSanPham();
    if (loaiSanPham != null) {
        return new SimpleStringProperty(string:loaiSanPham.getTenLoaiSP());
    } else {
        return new SimpleStringProperty(string: ""); // hoac gia tri mac dinh neu can
    }
});
colDonGia.setCellValueFactory(new PropertyValueFactory<>(string: "donGia"));
}

private void xoaRongTextfieldLoaiSP() {
    txtMaLoaiSP.setText(value: "");
    txtTenLoaiSP.setText(value: "");
    txtMaLoaiSP.requestFocus();
}
```

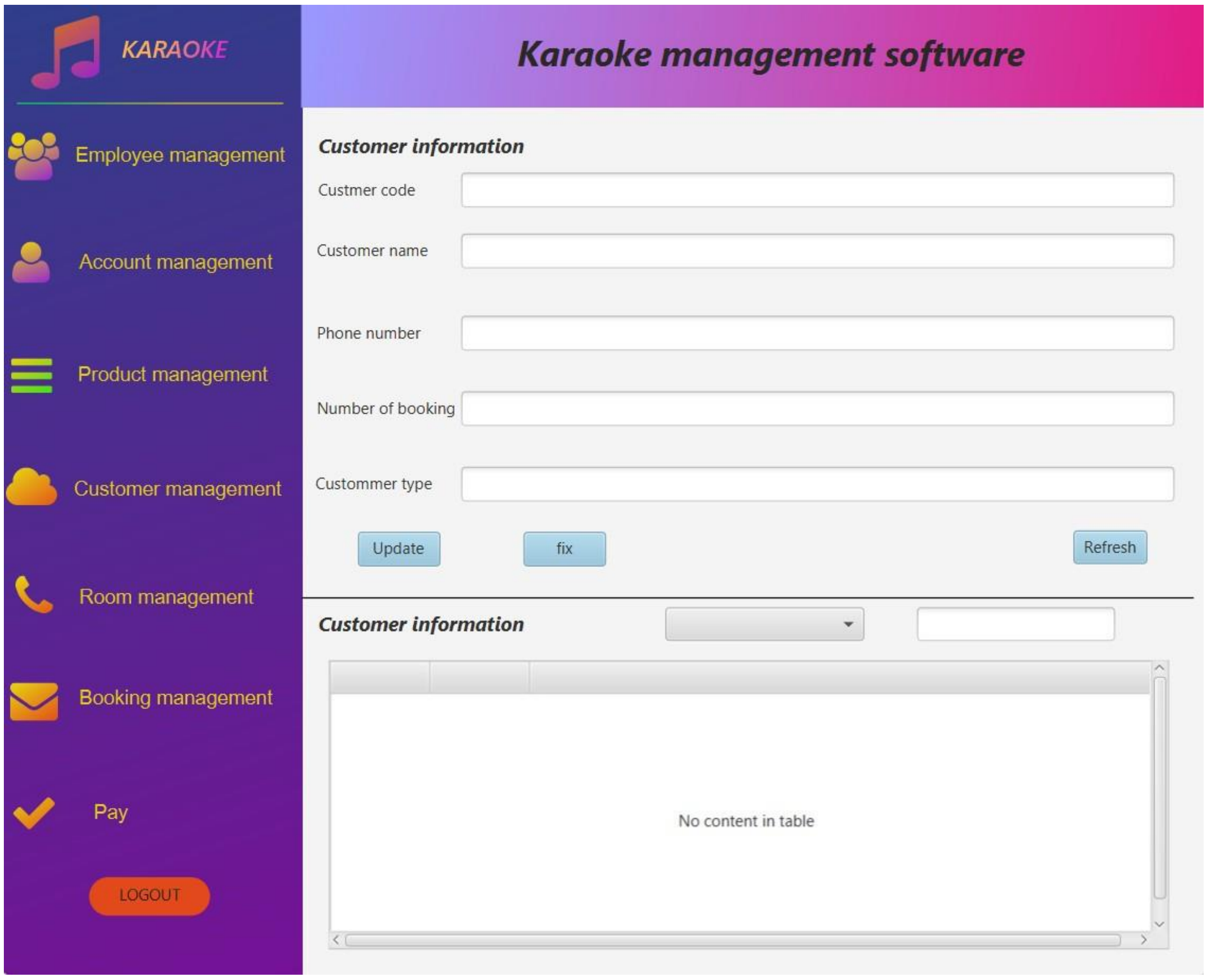


```java
private LoaiSanPham createLoaiSanPham() {
    LoaiSanPham loaiSanPham = new LoaiSanPham();
    loaiSanPham.setTenLoaiSP(tenLoaiSP:txtTenLoaiSP.getText());
    return loaiSanPham;
}

private void dataValidateLoaiSP(StringBuilder sb) {
    DataValidator.validateEmpty(field: txtTenLoaiSP, sb, errorMessage: "Ten loai san pham khong duoc de trong");
    DataValidator.validateVietnameseCharacters(field: txtTenLoaiSP, sb, errorMessage: "Ten loai san pham sai. Khong duoc nhap so
}

private void search(String str) {
    filteredSanPhamList.setPredicate(new Predicate<SanPham>() {
        @Override
        public boolean test(SanPham sanPham) {
            if (str == null || str.isEmpty()) {
                return true;
            }
            String lowerCaseFilter = str.toLowerCase();
            if (cmbTim.getSelectionModel().getSelectedItem().equals(anObject: "Tim theo ten san pham")) {
                return sanPham.getTenSanPham().toLowerCase().contains(s: lowerCaseFilter);
            } else if (cmbTim.getSelectionModel().getSelectedItem().equals(anObject: "Tim theo ma san pham")) {
                return sanPham.getMaSanPham().toLowerCase().contains(s: lowerCaseFilter);
            }
            return false;
        }
    });
}

private void dataValidateSP(StringBuilder sb) {
    DataValidator.validateEmpty(field: txtDonGia, sb, errorMessage: "Don gia khong duoc de trong");
    DataValidator.validateEmpty(field: txtTenSP, sb, errorMessage: "Ten san pham khong duoc de trong");
    DataValidator.validateDonGia(field: txtDonGia, sb, errorMessage: "Don gia chi duoc nhap so");
    DataValidator.validateVietnameseCharactersAndNumber(field: txtTenSP, sb, errorMessage: "Ten san pham sai. Khong co ky tu dac
}
```



```
private SanPham createSanPham() {
    SanPham sp = new SanPham();
    LoaiSanPham loaiSanPham = new LoaiSanPham();
    LoaiSanPhamDAO loaiSanPhamDAO = new LoaiSanPhamDAO();

    String tenLoaiSP = cmbLoaiSanPham.getSelectionModel().getSelectedItem();
if (tenLoaiSP != null && !tenLoaiSP.isEmpty()) {
    String maLoaiSP = loaiSanPhamDAO.getMaTheoTenLoai(ten:tenLoaiSP);
    loaiSanPham.setMaLoaiSP(maLoaiSP);
    loaiSanPham.setTenLoaiSP(tenLoaiSP);
} else {
    loaiSanPham = null; // Hoac xu ly theo cach khac neu can
}

    sp.setMaSanPham(maSanPham:txtMaSP.getText());
    sp.setTenSanPham(tenSanPham:txtTenSP.getText());
    sp.setDonGia(donGia:Double.parseDouble(s: txtDonGia.getText()));
    sp.setLoaiSanPham(loaiSanPham);

    return sp;
}

private void loadDataToCmbLoaiSanPham() {
LoaiSanPhamDAO loaiSanPhamDAO = new LoaiSanPhamDAO();
List<LoaiSanPham> lsp = loaiSanPhamDAO.getDanhSachLoaiSanPham();
cmbLoaiSanPham.getItems().clear(); // Xoa cac muc hien tai
for (LoaiSanPham loaiSanPham : lsp) {
    cmbLoaiSanPham.getItems().add(e: loaiSanPham.getTenLoaiSP());
}
```

4.Customer Relationship Management



Customer information

Custmer code	
Customer name	
Phone number	
Number of booking	
Custommer type	

Customer information

No content in table

1: Click the "Customer Management" button to display the customer management interface

2: Fields used to enter customer information

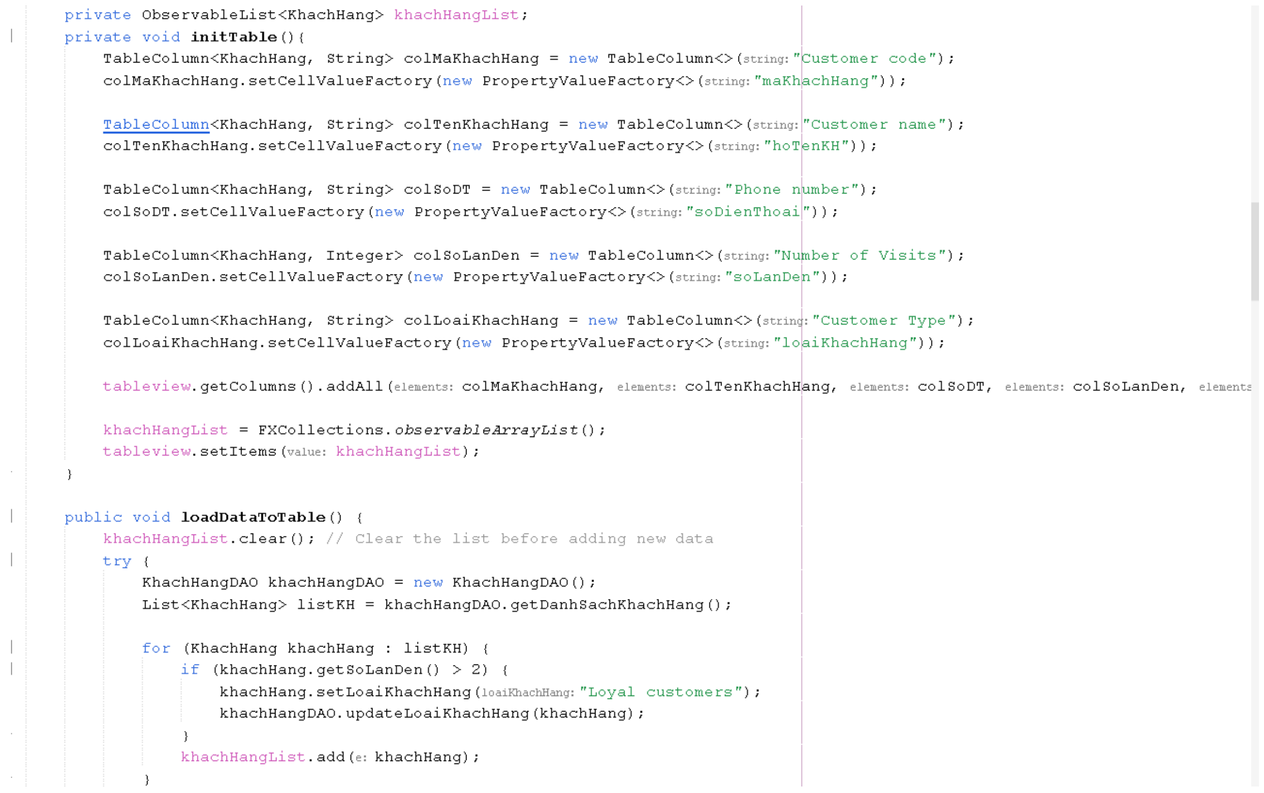
3: Table displaying customer list

4: Enter product information , press the "Add" button to add customers

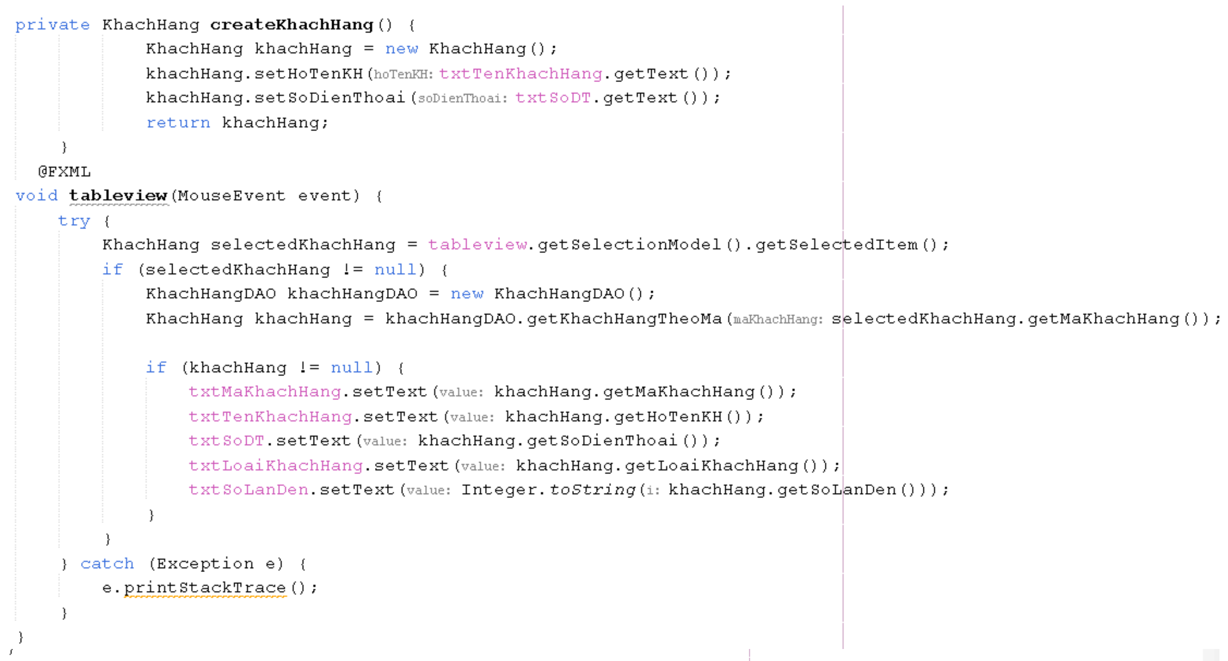
5: Select a customer that needs to be edited in the customer list table , enter the customer information that needs to be edited and press the "Edit" button to edit the customer.

6: Click the "Refresh" to refresh the page again

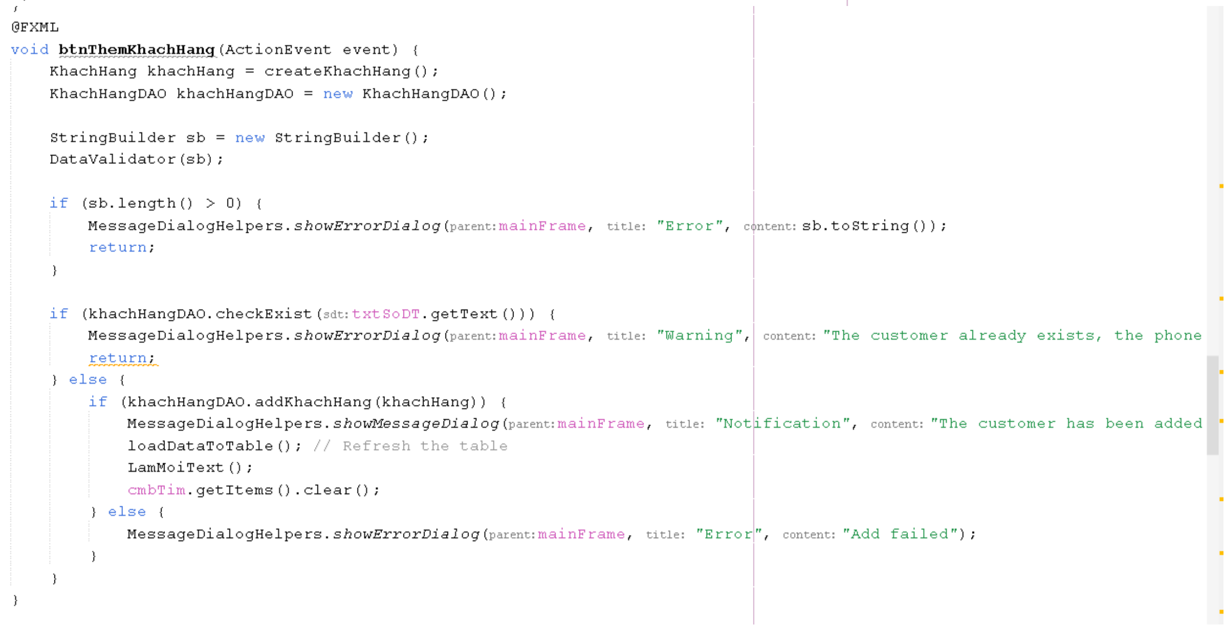
- **Validate:**
  - Customer Name : Customer name cannot be blank,Customer name is wrong. Do not enter numbers and special characters
  - Phone Number:Phone number can not be left blank, The phone number is in the wrong format, must have 9-10 digits, no characters. For example: 0788775877, Phone number already exists



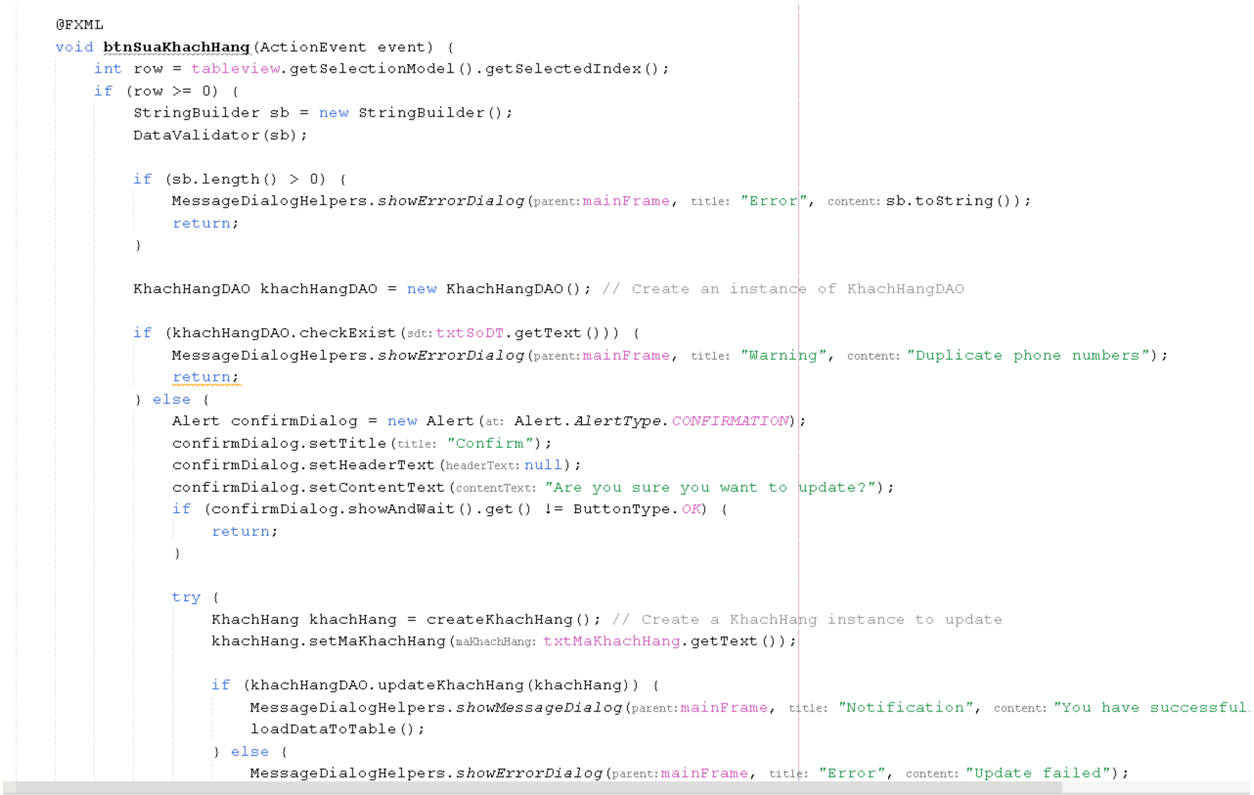
```java
private ObservableList<KhachHang> khachHangList;
private void initTable(){
    TableColumn<KhachHang, String> colMaKhachHang = new TableColumn<>(string: "Customer code");
    colMaKhachHang.setCellValueFactory(new PropertyValueFactory<>(string: "maKhachHang"));

    TableColumn<KhachHang, String> colTenKhachHang = new TableColumn<>(string: "Customer name");
    colTenKhachHang.setCellValueFactory(new PropertyValueFactory<>(string: "hoTenKH"));

    TableColumn<KhachHang, String> colSoDT = new TableColumn<>(string: "Phone number");
    colSoDT.setCellValueFactory(new PropertyValueFactory<>(string: "soDienThoai"));

    TableColumn<KhachHang, Integer> colSoLanDen = new TableColumn<>(string: "Number of Visits");
    colSoLanDen.setCellValueFactory(new PropertyValueFactory<>(string: "soLanDen"));

    TableColumn<KhachHang, String> colLoaiKhachHang = new TableColumn<>(string: "Customer Type");
    colLoaiKhachHang.setCellValueFactory(new PropertyValueFactory<>(string: "loaiKhachHang"));

    tableview.getColumns().addAll(elements: colMaKhachHang, elements: colTenKhachHang, elements: colSoDT, elements: colSoLanDen, elements

    khachHangList = FXCollections.observableArrayList();
    tableview.setItems(value: khachHangList);
}

public void loadDataToTable() {
    khachHangList.clear(); // Clear the list before adding new data
    try {
        KhachHangDAO khachHangDAO = new KhachHangDAO();
        List<KhachHang> listKH = khachHangDAO.getDanhSachKhachHang();

        for (KhachHang khachHang : listKH) {
            if (khachHang.getSoLanDen() > 2) {
                khachHang.setLoaiKhachHang(loaiKhachHang: "Loyal customers");
                khachHangDAO.updateLoaiKhachHang(khachHang);
            }
            khachHangList.add(e: khachHang);
        }
```



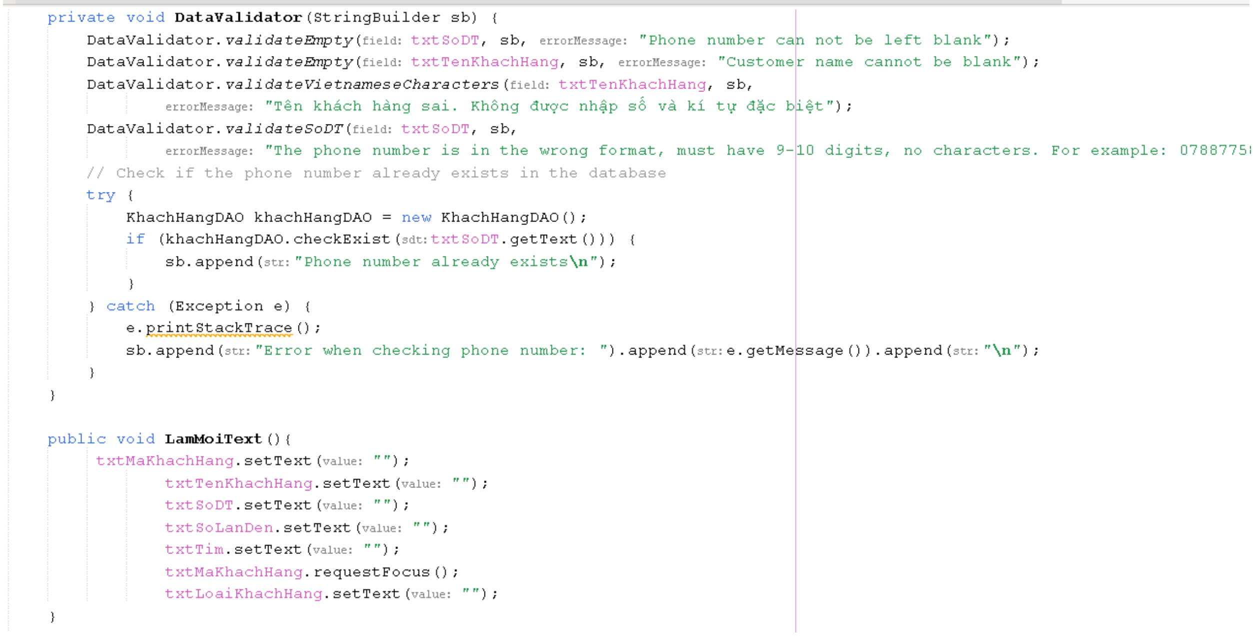
```
private KhachHang createKhachHang() {
            KhachHang khachHang = new KhachHang();
            khachHang.setHoTenKH(hoTenKH: txtTenKhachHang.getText());
            khachHang.setSoDienThoai(soDienThoai: txtSoDT.getText());
            return khachHang;
    }
  @FXML
void tableview(MouseEvent event) {
    try {
        KhachHang selectedKhachHang = tableview.getSelectionModel().getSelectedItem();
        if (selectedKhachHang != null) {
            KhachHangDAO khachHangDAO = new KhachHangDAO();
            KhachHang khachHang = khachHangDAO.getKhachHangTheoMa(maKhachHang: selectedKhachHang.getMaKhachHang());

            if (khachHang != null) {
                txtMaKhachHang.setText(value: khachHang.getMaKhachHang());
                txtTenKhachHang.setText(value: khachHang.getHoTenKH());
                txtSoDT.setText(value: khachHang.getSoDienThoai());
                txtLoaiKhachHang.setText(value: khachHang.getLoaiKhachHang());
                txtSoLanDen.setText(value: Integer.toString(i: khachHang.getSoLanDen()));
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



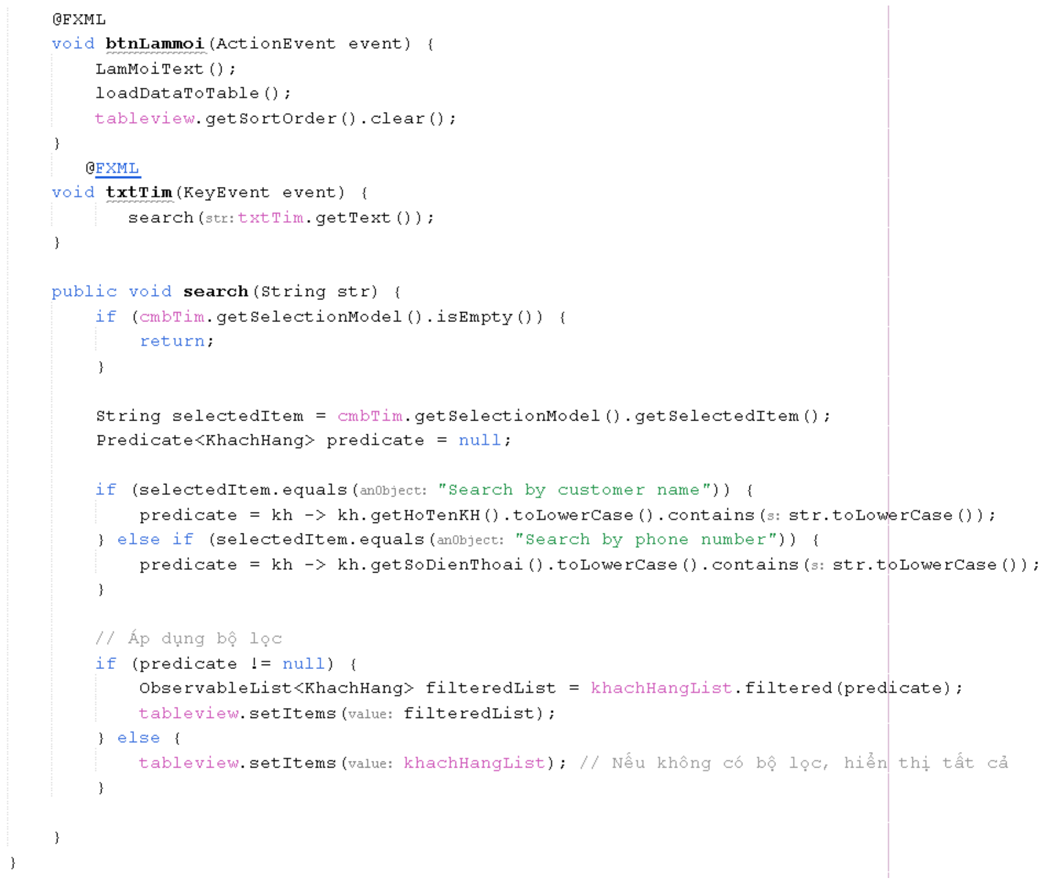
```
@FXML
void btnThemKhachHang(ActionEvent event) {
    KhachHang khachHang = createKhachHang();
    KhachHangDAO khachHangDAO = new KhachHangDAO();

    StringBuilder sb = new StringBuilder();
    DataValidator(sb);

    if (sb.length() > 0) {
        MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Error", content: sb.toString());
        return;
    }

    if (khachHangDAO.checkExist(sdt:txtSoDT.getText())) {
        MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Warning", content: "The customer already exists, the phone
        return;
    } else {
        if (khachHangDAO.addKhachHang(khachHang)) {
            MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Notification", content: "The customer has been added
            loadDataToTable(); // Refresh the table
            LamMoiText();
            cmbTim.getItems().clear();
        } else {
            MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Error", content: "Add failed");
        }
    }
}
```



```
@FXML
void btnSuaKhachHang(ActionEvent event) {
    int row = tableview.getSelectionModel().getSelectedIndex();
    if (row >= 0) {
        StringBuilder sb = new StringBuilder();
        DataValidator(sb);

        if (sb.length() > 0) {
            MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Error", content: sb.toString());
            return;
        }

        KhachHangDAO khachHangDAO = new KhachHangDAO(); // Create an instance of KhachHangDAO

        if (khachHangDAO.checkExist(sdt:txtSoDT.getText())) {
            MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Warning", content: "Duplicate phone numbers");
            return;
        } else {
            Alert confirmDialog = new Alert(at: Alert.AlertType.CONFIRMATION);
            confirmDialog.setTitle(title: "Confirm");
            confirmDialog.setHeaderText(headerText: null);
            confirmDialog.setContentText(contentText: "Are you sure you want to update?");
            if (confirmDialog.showAndWait().get() != ButtonType.OK) {
                return;
            }

            try {
                KhachHang khachHang = createKhachHang(); // Create a KhachHang instance to update
                khachHang.setMaKhachHang(maKhachHang: txtMaKhachHang.getText());

                if (khachHangDAO.updateKhachHang(khachHang)) {
                    MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Notification", content: "You have successful
                    loadDataToTable();
                } else {
                    MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Error", content: "Update failed");
```

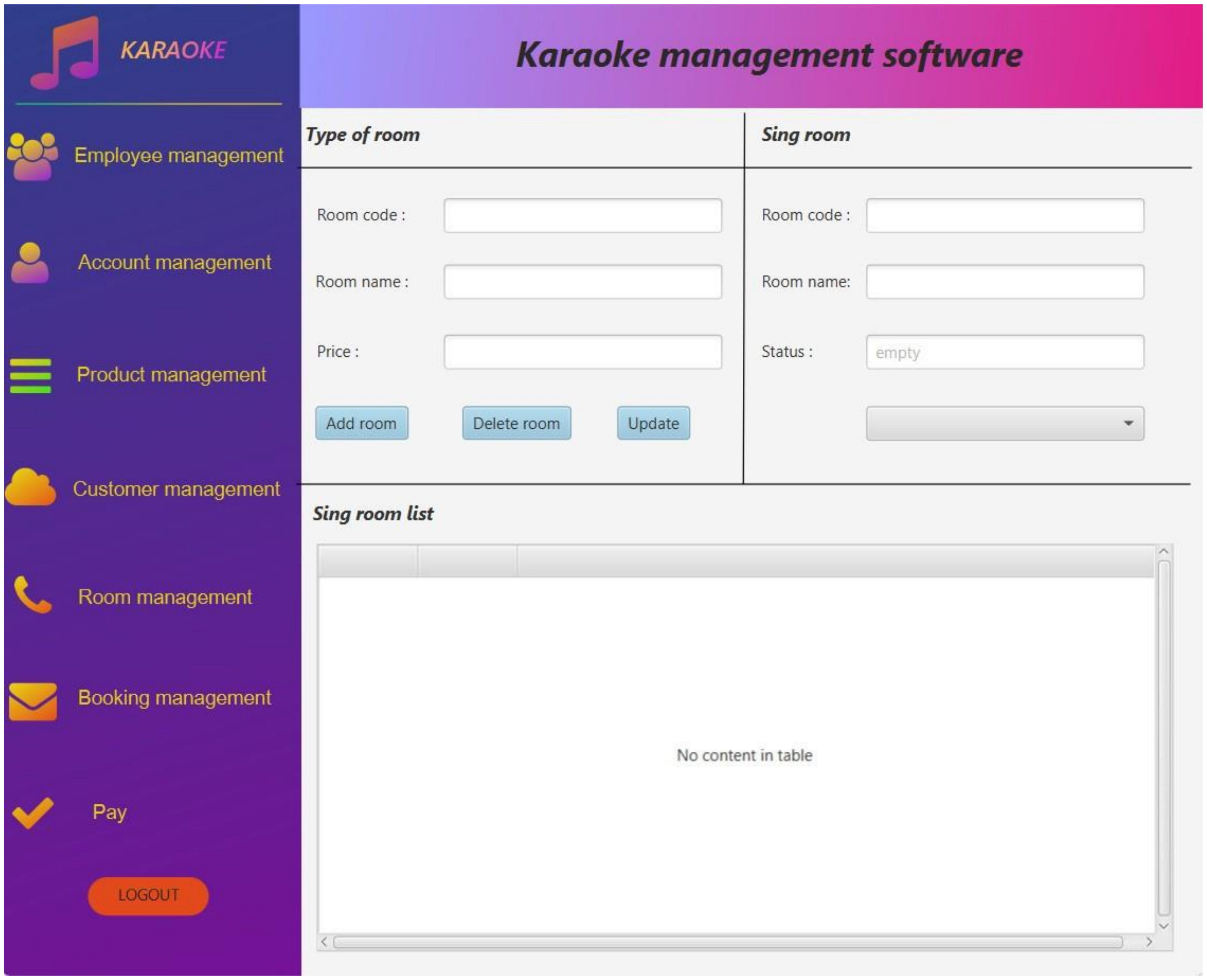


```
private void DataValidator(StringBuilder sb) {
    DataValidator.validateEmpty(field: txtSoDT, sb, errorMessage: "Phone number can not be left blank");
    DataValidator.validateEmpty(field: txtTenKhachHang, sb, errorMessage: "Customer name cannot be blank");
    DataValidator.validateVietnameseCharacters(field: txtTenKhachHang, sb,
            errorMessage: "Tên khách hàng sai. Không được nhập số và kí tự đặc biệt");
    DataValidator.validateSoDT(field: txtSoDT, sb,
            errorMessage: "The phone number is in the wrong format, must have 9-10 digits, no characters. For example: 0788775
    // Check if the phone number already exists in the database
    try {
        KhachHangDAO khachHangDAO = new KhachHangDAO();
        if (khachHangDAO.checkExist(sdt:txtSoDT.getText())) {
            sb.append(str:"Phone number already exists\n");
        }
    } catch (Exception e) {
        e.printStackTrace();
        sb.append(str:"Error when checking phone number: ").append(str:e.getMessage()).append(str:"\n");
    }
}

public void LamMoiText(){
     txtMaKhachHang.setText(value: "");
            txtTenKhachHang.setText(value: "");
            txtSoDT.setText(value: "");
            txtSoLanDen.setText(value: "");
            txtTim.setText(value: "");
            txtMaKhachHang.requestFocus();
            txtLoaiKhachHang.setText(value: "");
}
```



```
    @FXML
    void btnLammoi(ActionEvent event) {
        LamMoiText();
        loadDataToTable();
        tableview.getSortOrder().clear();
    }
       @FXML
    void txtTim(KeyEvent event) {
           search(str: txtTim.getText());
    }

    public void search(String str) {
        if (cmbTim.getSelectionModel().isEmpty()) {
            return;
        }

        String selectedItem = cmbTim.getSelectionModel().getSelectedItem();
        Predicate<KhachHang> predicate = null;

        if (selectedItem.equals(anObject: "Search by customer name")) {
            predicate = kh -> kh.getHoTenKH().toLowerCase().contains(s: str.toLowerCase());
        } else if (selectedItem.equals(anObject: "Search by phone number")) {
            predicate = kh -> kh.getSoDienThoai().toLowerCase().contains(s: str.toLowerCase());
        }

        // Áp dụng bộ lọc
        if (predicate != null) {
            ObservableList<KhachHang> filteredList = khachHangList.filtered(predicate);
            tableview.setItems(value: filteredList);
        } else {
            tableview.setItems(value: khachHangList); // Nếu không có bộ lọc, hiển thị tất cả
        }

    }
}
```

5.room manager



1: Click the "Room management" button to display the room management interface

2: Fields used to enter room information

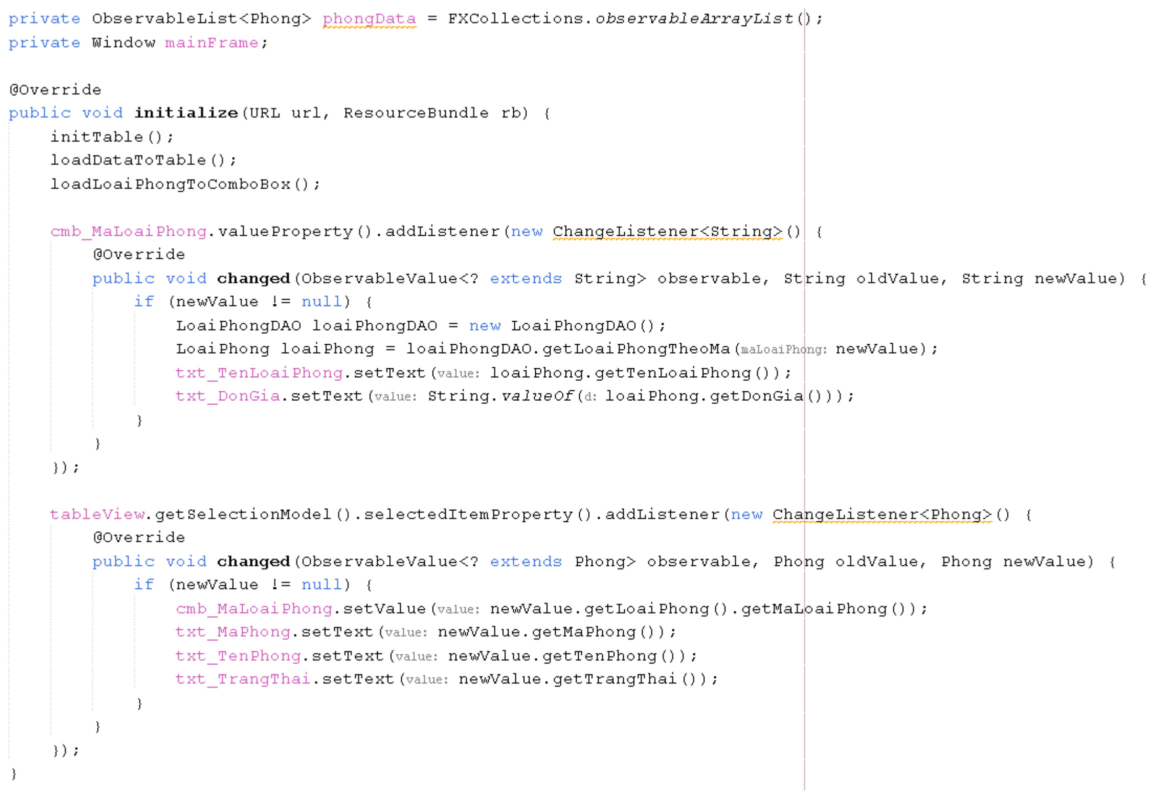
3: Table displaying room list

4: Enter room information , press the "Add Room" button to add a room

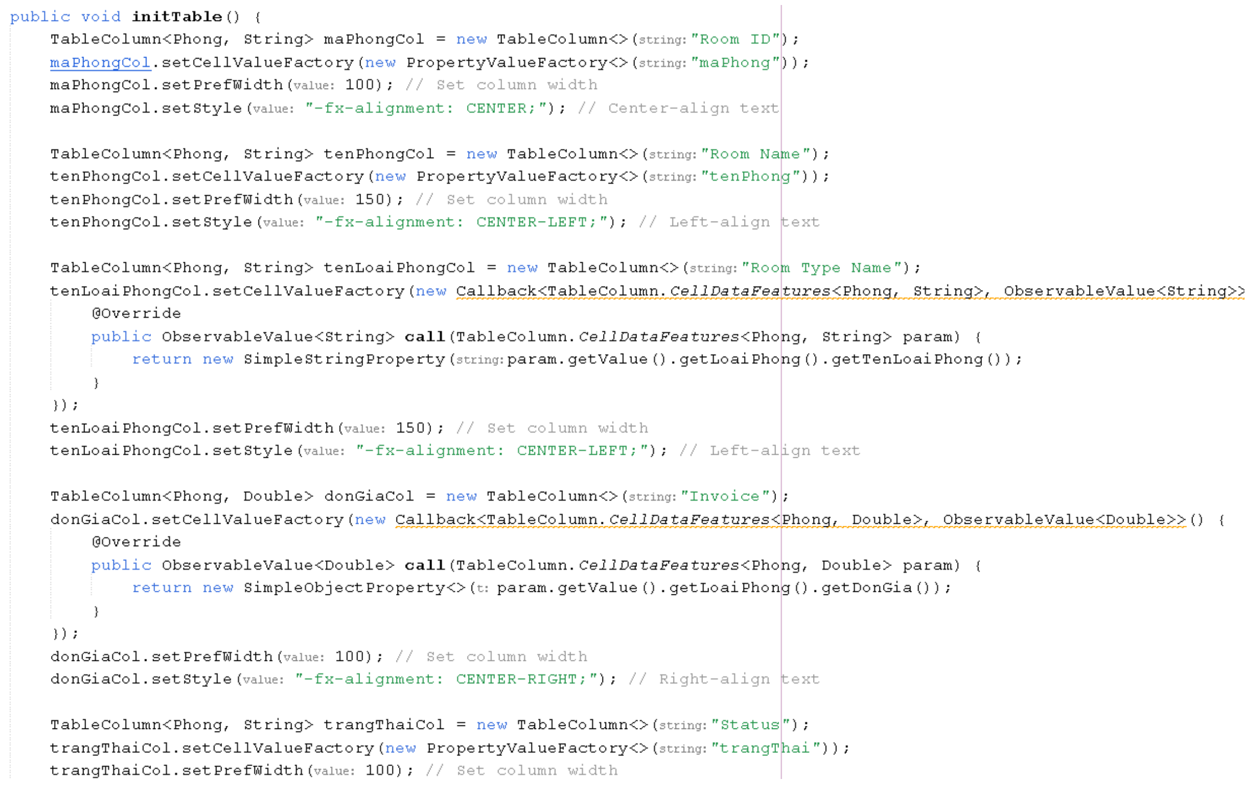
5: Select the room to delete in the room list table , press the "Delete Room" button to delete the room

6: Select a room to edit in the room list table , enter the information to edit and press the "Update" button to update

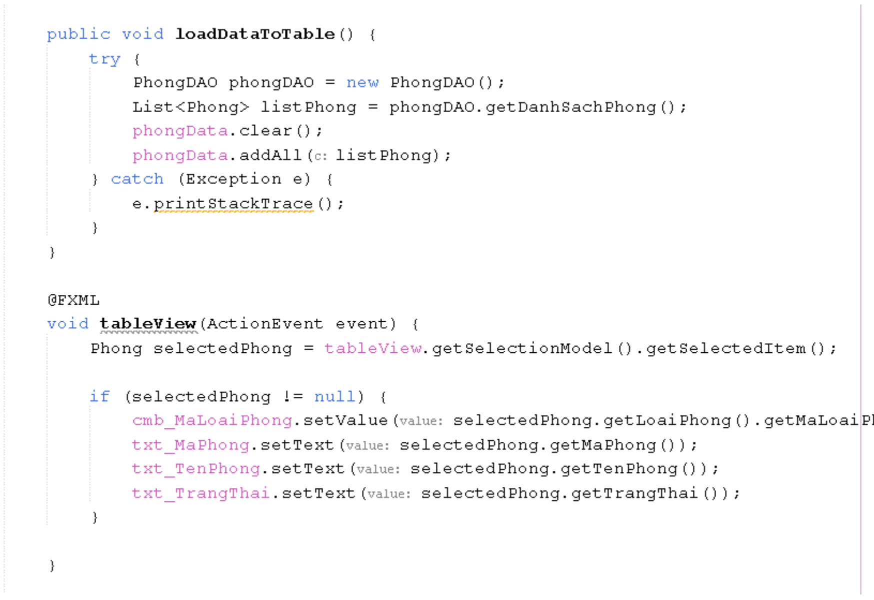
7: Select the room status in the combo box (“Empty”, “In use”), the room information will be displayed in the room list



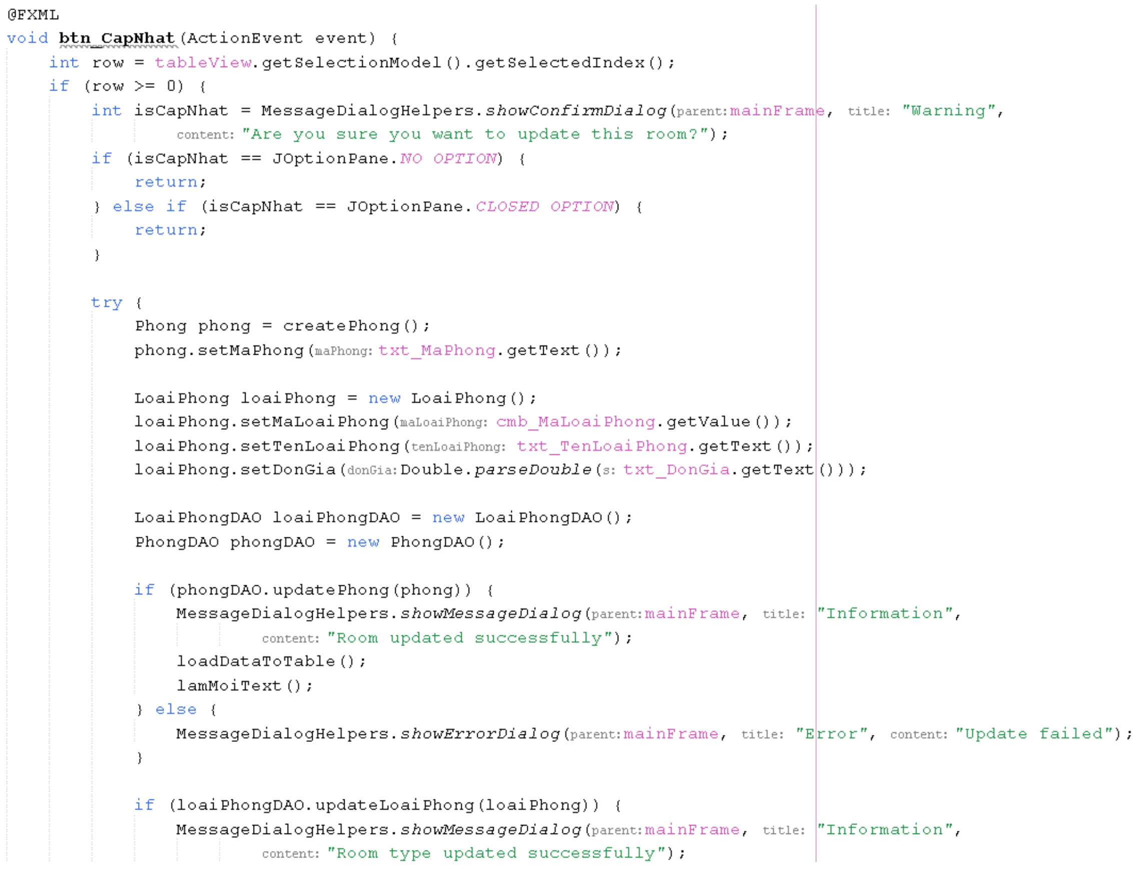
```
private ObservableList<Phong> phongData = FXCollections.observableArrayList();
private Window mainFrame;

@Override
public void initialize(URL url, ResourceBundle rb) {
    initTable();
    loadDataToTable();
    loadLoaiPhongToComboBox();

    cmb_MaLoaiPhong.valueProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            if (newValue != null) {
                LoaiPhongDAO loaiPhongDAO = new LoaiPhongDAO();
                LoaiPhong loaiPhong = loaiPhongDAO.getLoaiPhongTheoMa(maLoaiPhong: newValue);
                txt_TenLoaiPhong.setText(value: loaiPhong.getTenLoaiPhong());
                txt_DonGia.setText(value: String.valueOf(d: loaiPhong.getDonGia()));
            }
        }
    });

    tableView.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Phong>() {
        @Override
        public void changed(ObservableValue<? extends Phong> observable, Phong oldValue, Phong newValue) {
            if (newValue != null) {
                cmb_MaLoaiPhong.setValue(value: newValue.getLoaiPhong().getMaLoaiPhong());
                txt_MaPhong.setText(value: newValue.getMaPhong());
                txt_TenPhong.setText(value: newValue.getTenPhong());
                txt_TrangThai.setText(value: newValue.getTrangThai());
            }
        }
    });
}
```



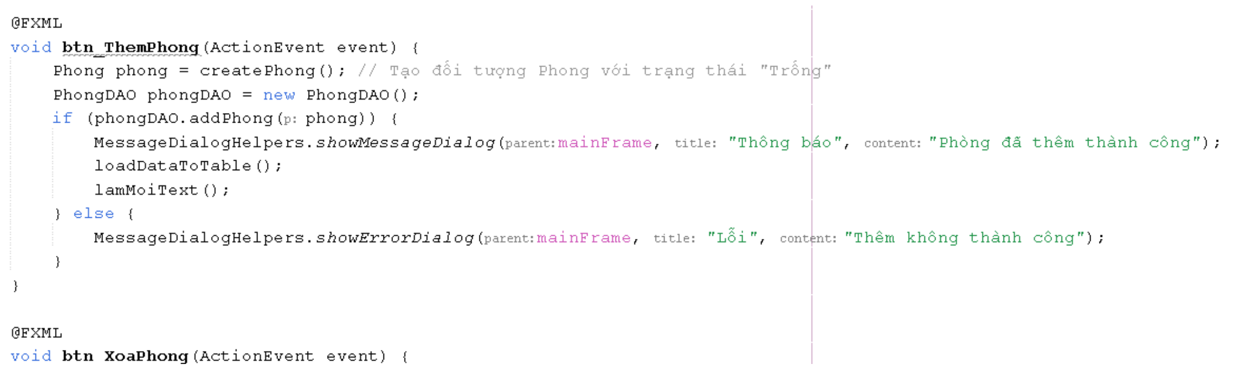
```
public void initTable() {
    TableColumn<Phong, String> maPhongCol = new TableColumn<>(string: "Room ID");
    maPhongCol.setCellValueFactory(new PropertyValueFactory<>(string: "maPhong"));
    maPhongCol.setPrefWidth(value: 100); // Set column width
    maPhongCol.setStyle(value: "-fx-alignment: CENTER;"); // Center-align text

    TableColumn<Phong, String> tenPhongCol = new TableColumn<>(string: "Room Name");
    tenPhongCol.setCellValueFactory(new PropertyValueFactory<>(string: "tenPhong"));
    tenPhongCol.setPrefWidth(value: 150); // Set column width
    tenPhongCol.setStyle(value: "-fx-alignment: CENTER-LEFT;"); // Left-align text

    TableColumn<Phong, String> tenLoaiPhongCol = new TableColumn<>(string: "Room Type Name");
    tenLoaiPhongCol.setCellValueFactory(new Callback<TableColumn.CellDataFeatures<Phong, String>, ObservableValue<String>>
        @Override
        public ObservableValue<String> call(TableColumn.CellDataFeatures<Phong, String> param) {
            return new SimpleStringProperty(string: param.getValue().getLoaiPhong().getTenLoaiPhong());
        }
    });
    tenLoaiPhongCol.setPrefWidth(value: 150); // Set column width
    tenLoaiPhongCol.setStyle(value: "-fx-alignment: CENTER-LEFT;"); // Left-align text

    TableColumn<Phong, Double> donGiaCol = new TableColumn<>(string: "Invoice");
    donGiaCol.setCellValueFactory(new Callback<TableColumn.CellDataFeatures<Phong, Double>, ObservableValue<Double>>() {
        @Override
        public ObservableValue<Double> call(TableColumn.CellDataFeatures<Phong, Double> param) {
            return new SimpleObjectProperty<>(t: param.getValue().getLoaiPhong().getDonGia());
        }
    });
    donGiaCol.setPrefWidth(value: 100); // Set column width
    donGiaCol.setStyle(value: "-fx-alignment: CENTER-RIGHT;"); // Right-align text

    TableColumn<Phong, String> trangThaiCol = new TableColumn<>(string: "Status");
    trangThaiCol.setCellValueFactory(new PropertyValueFactory<>(string: "trangThai"));
    trangThaiCol.setPrefWidth(value: 100); // Set column width
```



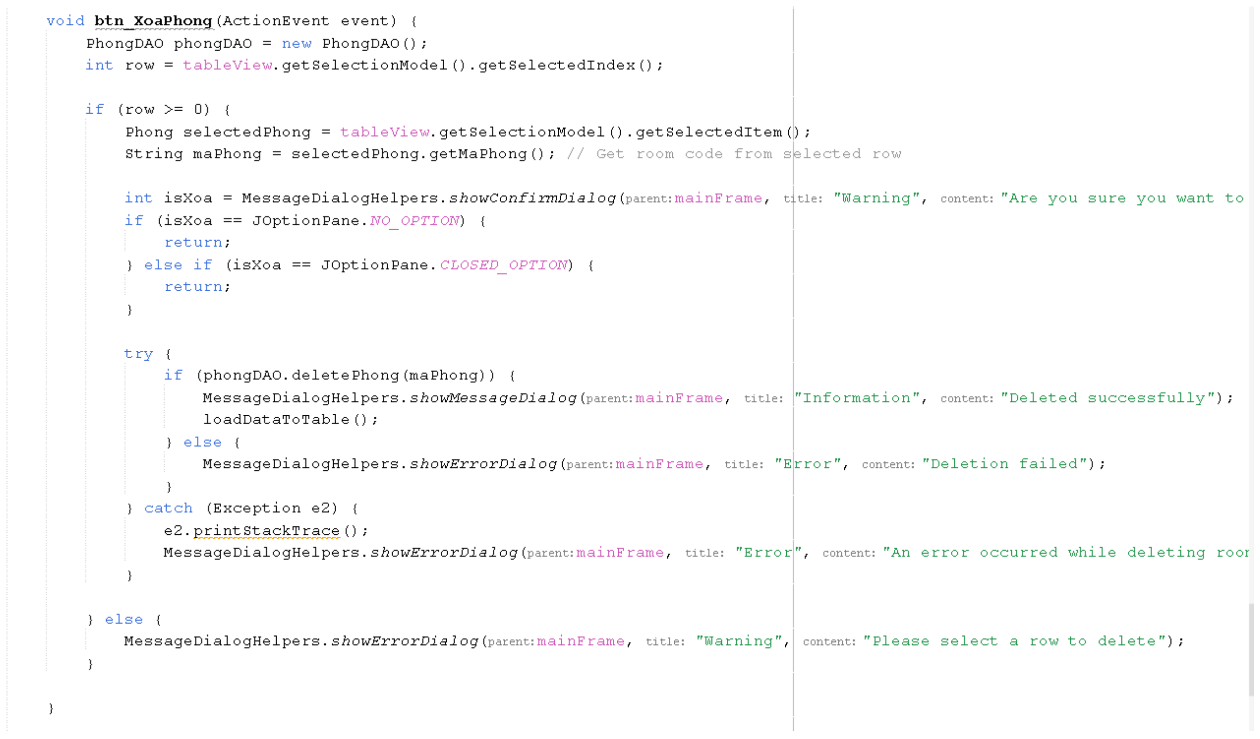
```
public void loadDataToTable() {
    try {
        PhongDAO phongDAO = new PhongDAO();
        List<Phong> listPhong = phongDAO.getDanhSachPhong();
        phongData.clear();
        phongData.addAll(c: listPhong);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@FXML
void tableView(ActionEvent event) {
    Phong selectedPhong = tableView.getSelectionModel().getSelectedItem();

    if (selectedPhong != null) {
        cmb_MaLoaiPhong.setValue(value: selectedPhong.getLoaiPhong().getMaLoaiPhong());
        txt_MaPhong.setText(value: selectedPhong.getMaPhong());
        txt_TenPhong.setText(value: selectedPhong.getTenPhong());
        txt_TrangThai.setText(value: selectedPhong.getTrangThai());
    }

}
```



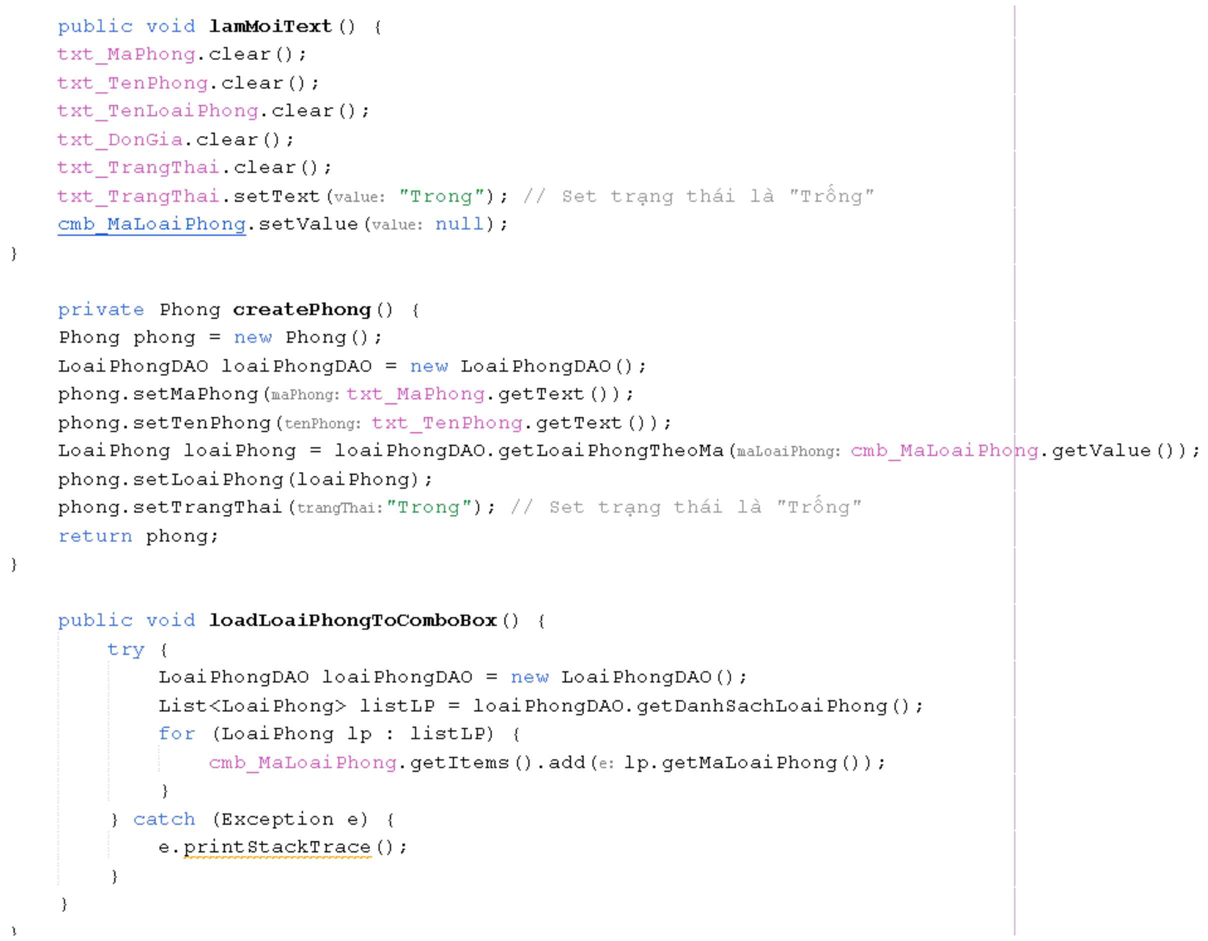
```
@FXML
void btn_CapNhat(ActionEvent event) {
    int row = tableView.getSelectionModel().getSelectedIndex();
    if (row >= 0) {
        int isCapNhat = MessageDialogHelpers.showConfirmDialog(parent:mainFrame, title: "Warning",
                content: "Are you sure you want to update this room?");
        if (isCapNhat == JOptionPane.NO_OPTION) {
            return;
        } else if (isCapNhat == JOptionPane.CLOSED_OPTION) {
            return;
        }

        try {
            Phong phong = createPhong();
            phong.setMaPhong(maPhong: txt_MaPhong.getText());

            LoaiPhong loaiPhong = new LoaiPhong();
            loaiPhong.setMaLoaiPhong(maLoaiPhong: cmb_MaLoaiPhong.getValue());
            loaiPhong.setTenLoaiPhong(tenLoaiPhong: txt_TenLoaiPhong.getText());
            loaiPhong.setDonGia(donGia:Double.parseDouble(s: txt_DonGia.getText()));

            LoaiPhongDAO loaiPhongDAO = new LoaiPhongDAO();
            PhongDAO phongDAO = new PhongDAO();

            if (phongDAO.updatePhong(phong)) {
                MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Information",
                        content: "Room updated successfully");
                loadDataToTable();
                lamMoiText();
            } else {
                MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Error", content: "Update failed");
            }

            if (loaiPhongDAO.updateLoaiPhong(loaiPhong)) {
                MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Information",
                        content: "Room type updated successfully");
```



```
@FXML
void btn_ThemPhong(ActionEvent event) {
    Phong phong = createPhong(); // Tạo đối tượng Phong với trạng thái "Trống"
    PhongDAO phongDAO = new PhongDAO();
    if (phongDAO.addPhong(p: phong)) {
        MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Thông báo", content: "Phòng đã thêm thành công");
        loadDataToTable();
        lamMoiText();
    } else {
        MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Lỗi", content: "Thêm không thành công");
    }
}

@FXML
void btn XoaPhong(ActionEvent event) {
```

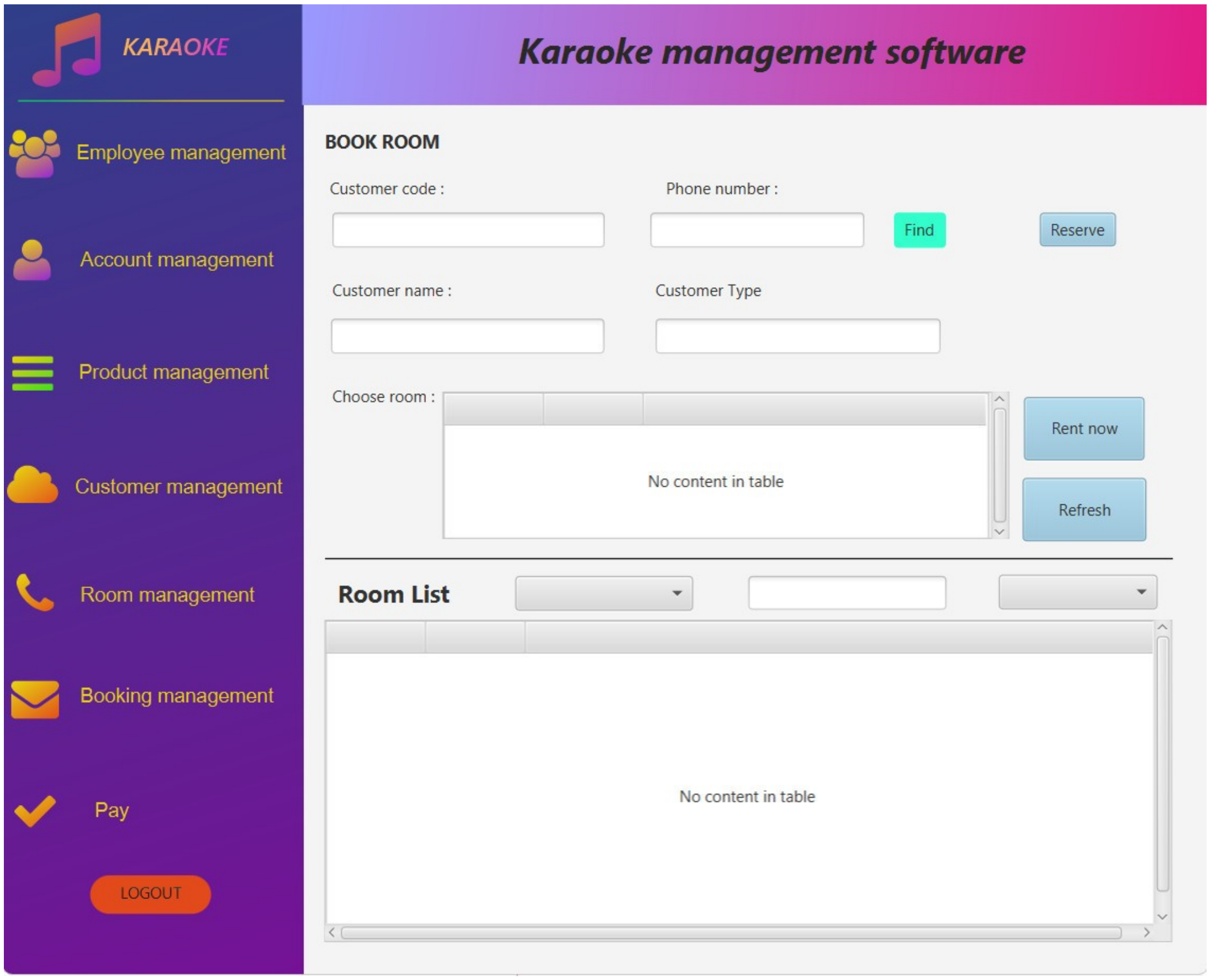


```
void btn_XoaPhong(ActionEvent event) {
    PhongDAO phongDAO = new PhongDAO();
    int row = tableView.getSelectionModel().getSelectedIndex();

    if (row >= 0) {
        Phong selectedPhong = tableView.getSelectionModel().getSelectedItem();
        String maPhong = selectedPhong.getMaPhong(); // Get room code from selected row

        int isXoa = MessageDialogHelpers.showConfirmDialog(parent:mainFrame, title: "Warning", content: "Are you sure you want to
        if (isXoa == JOptionPane.NO_OPTION) {
            return;
        } else if (isXoa == JOptionPane.CLOSED_OPTION) {
            return;
        }

        try {
            if (phongDAO.deletePhong(maPhong)) {
                MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Information", content: "Deleted successfully");
                loadDataToTable();
            } else {
                MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Error", content: "Deletion failed");
            }
        } catch (Exception e2) {
            e2.printStackTrace();
            MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Error", content: "An error occurred while deleting roo
        }

    } else {
        MessageDialogHelpers.showErrorDialog(parent:mainFrame, title: "Warning", content: "Please select a row to delete");
    }

}
```



```
    public void lamMoiText() {
    txt_MaPhong.clear();
    txt_TenPhong.clear();
    txt_TenLoaiPhong.clear();
    txt_DonGia.clear();
    txt_TrangThai.clear();
    txt_TrangThai.setText(value: "Trong"); // Set trạng thái là "Trống"
    cmb_MaLoaiPhong.setValue(value: null);
}

    private Phong createPhong() {
    Phong phong = new Phong();
    LoaiPhongDAO loaiPhongDAO = new LoaiPhongDAO();
    phong.setMaPhong(maPhong: txt_MaPhong.getText());
    phong.setTenPhong(tenPhong: txt_TenPhong.getText());
    LoaiPhong loaiPhong = loaiPhongDAO.getLoaiPhongTheoMa(maLoaiPhong: cmb_MaLoaiPhong.getValue());
    phong.setLoaiPhong(loaiPhong);
    phong.setTrangThai(trangThai:"Trong"); // Set trạng thái là "Trống"
    return phong;
}

    public void loadLoaiPhongToComboBox() {
        try {
            LoaiPhongDAO loaiPhongDAO = new LoaiPhongDAO();
            List<LoaiPhong> listLP = loaiPhongDAO.getDanhSachLoaiPhong();
            for (LoaiPhong lp : listLP) {
                cmb_MaLoaiPhong.getItems().add(e: lp.getMaLoaiPhong());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

6. Book Room



1: Click the "Book a room" button to display the booking interface

2: Enter the phone number of the customer who wants to book a room and press the search button to display the information of the customer coming to book the room

3: List of available rooms

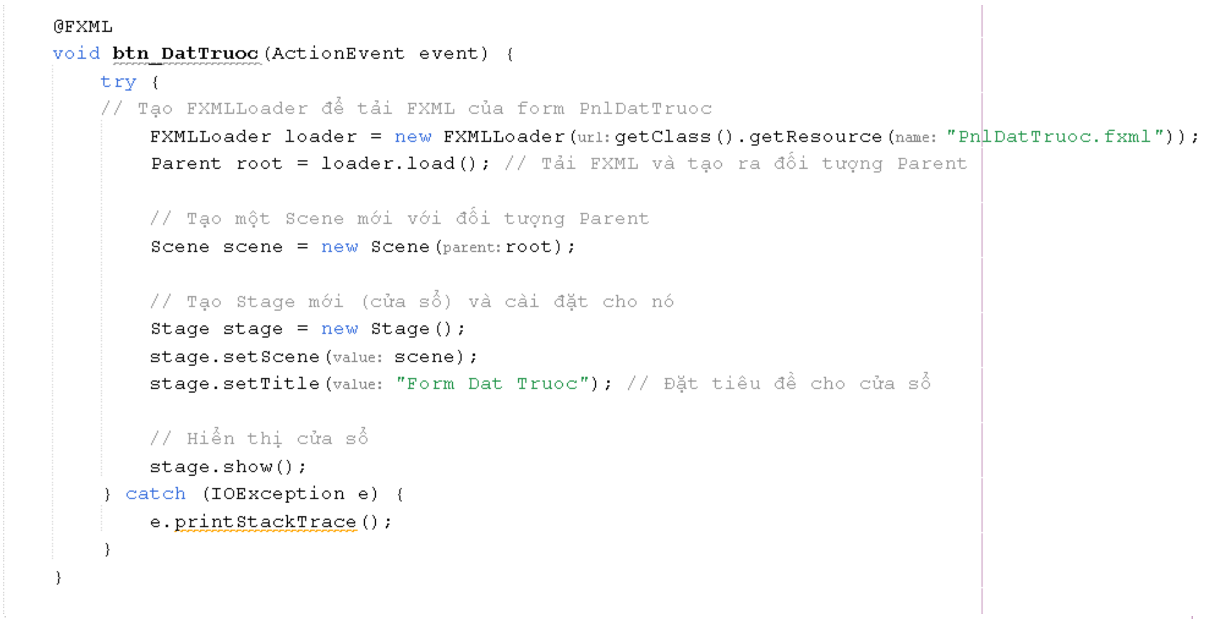
4: List of bookings

5: After successfully searching for the customer and selecting an available room according to the customer's request, press the "Rent now" button to book a room for the customer to check in directly, and the reservation information will appear in the order list. booking

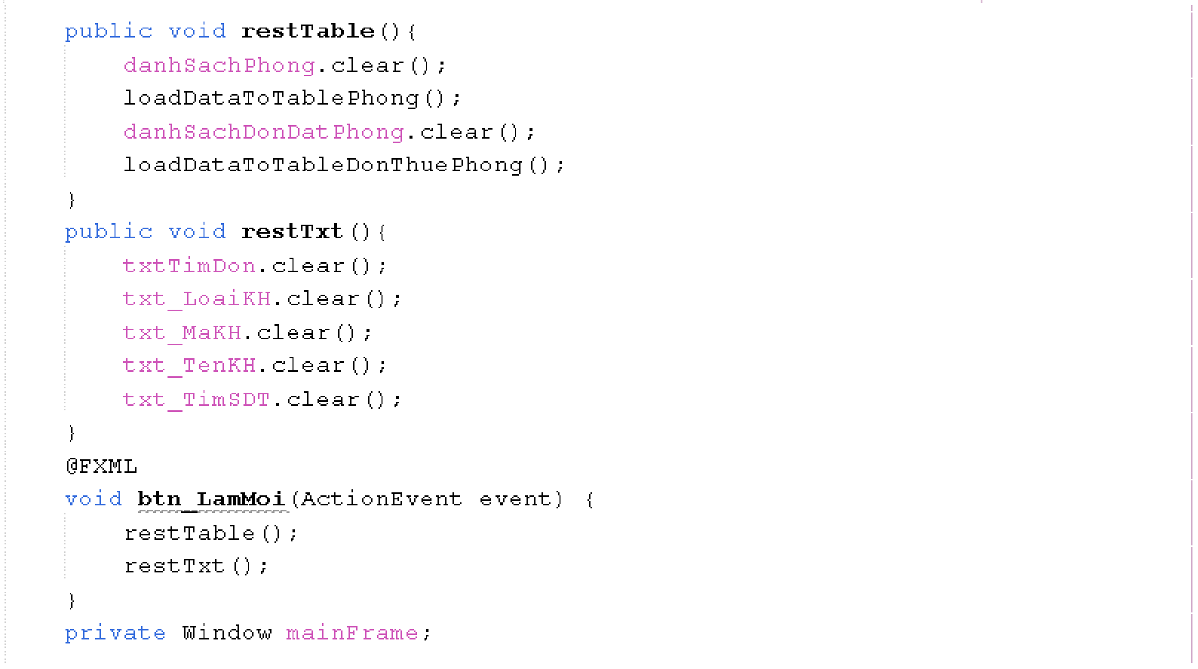
6: Refresh the reservation list and customer information to the original state

7: Select search criteria in the combo box, enter information into the text field, search results will display in the list of bookings

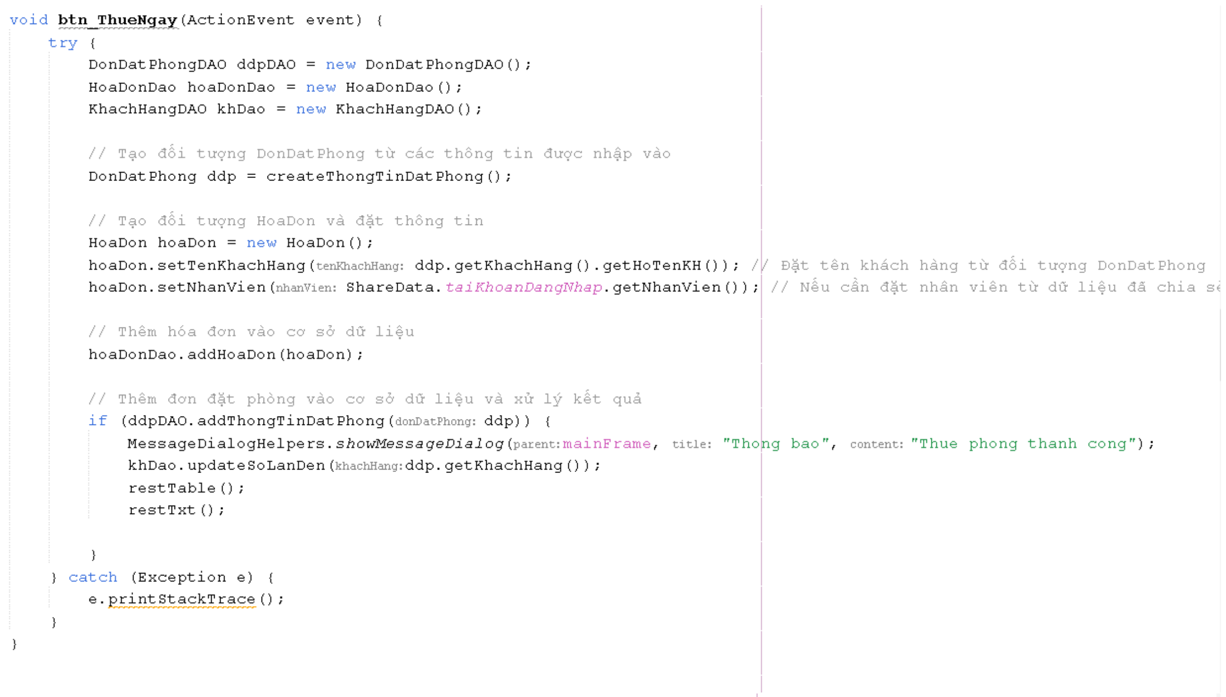
8: Click “Reserve” if the customer wants to reserve a room in advance. After pressing the button, a Reservation interface will appear.



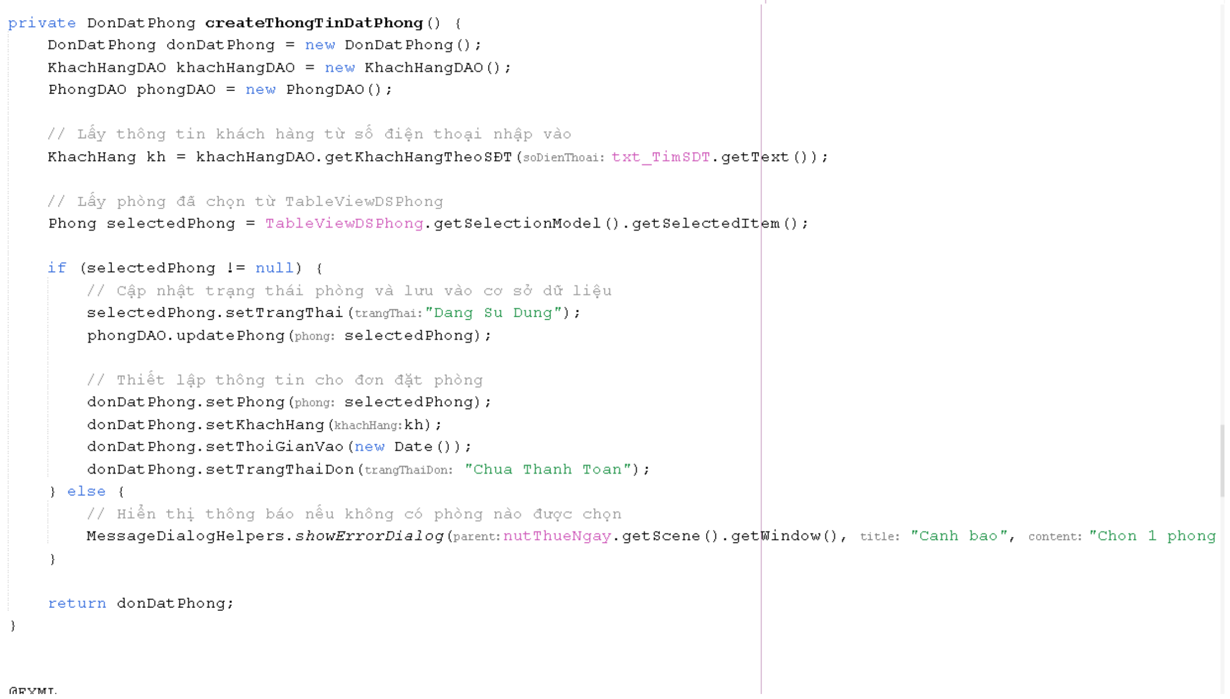
```
@FXML
void btn_DatTruoc(ActionEvent event) {
    try {
    // Tạo FXMLLoader để tải FXML của form PnlDatTruoc
        FXMLLoader loader = new FXMLLoader(url:getClass().getResource(name: "PnlDatTruoc.fxml"));
        Parent root = loader.load(); // Tải FXML và tạo ra đối tượng Parent

        // Tạo một Scene mới với đối tượng Parent
        Scene scene = new Scene(parent:root);

        // Tạo Stage mới (cửa sổ) và cài đặt cho nó
        Stage stage = new Stage();
        stage.setScene(value: scene);
        stage.setTitle(value: "Form Dat Truoc"); // Đặt tiêu đề cho cửa sổ

        // Hiển thị cửa sổ
        stage.show();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```



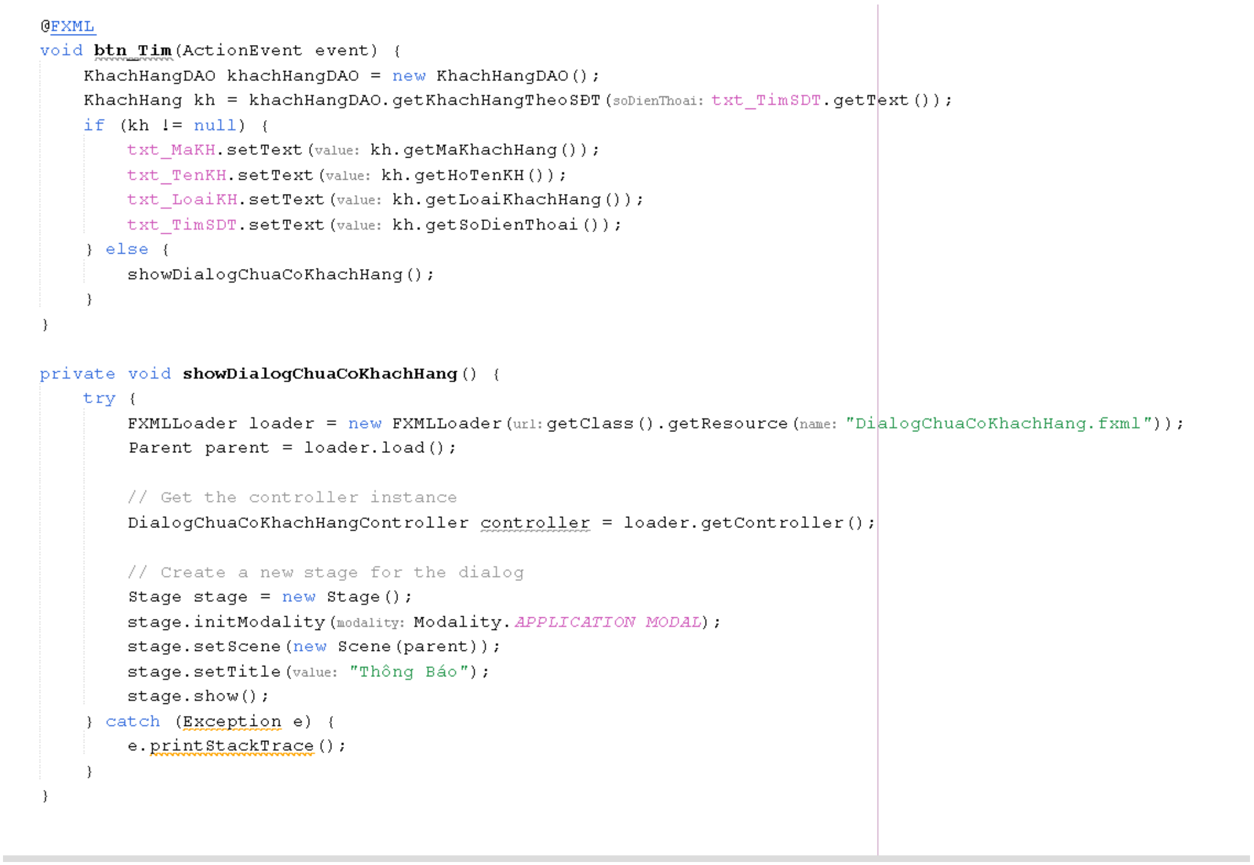
```
public void restTable(){
    danhSachPhong.clear();
    loadDataToTablePhong();
    danhSachDonDatPhong.clear();
    loadDataToTableDonThuePhong();
}
public void restTxt(){
    txtTimDon.clear();
    txt_LoaiKH.clear();
    txt_MaKH.clear();
    txt_TenKH.clear();
    txt_TimSDT.clear();
}
@FXML
void btn_LamMoi(ActionEvent event) {
    restTable();
    restTxt();
}
private Window mainFrame;
```



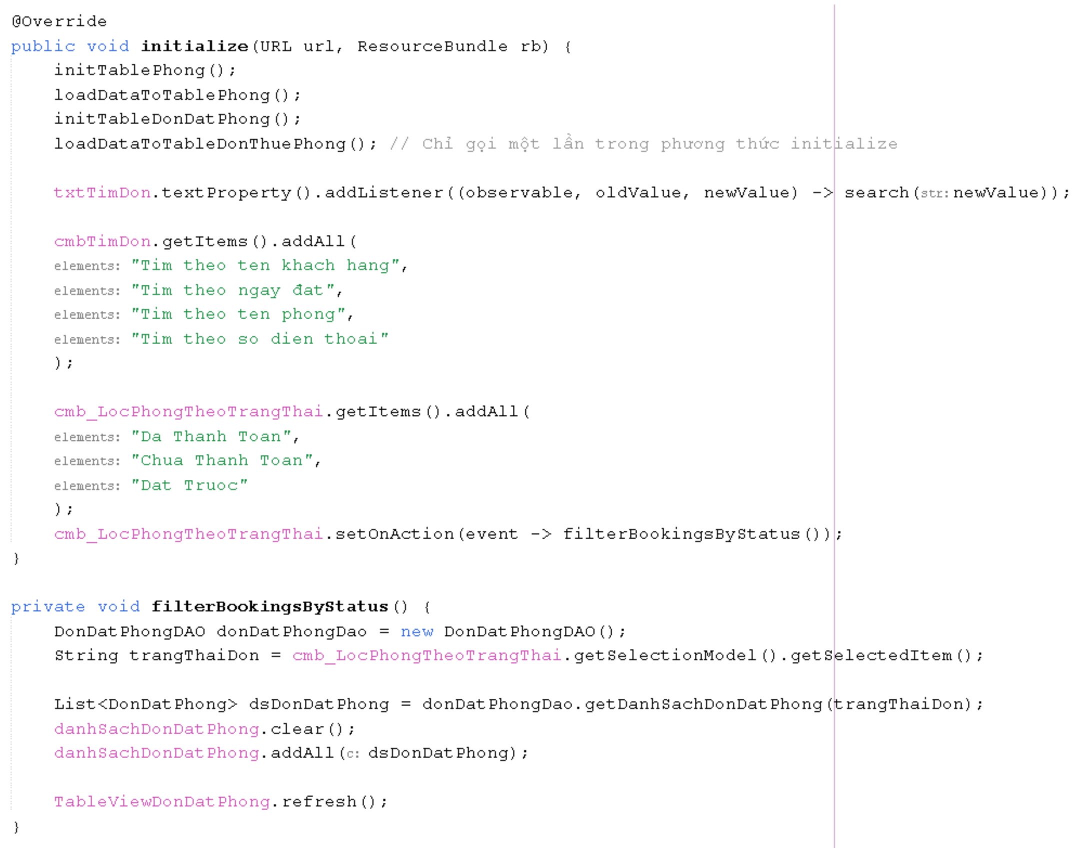
```
void btn_ThueNgay(ActionEvent event) {
    try {
        DonDatPhongDAO ddpDAO = new DonDatPhongDAO();
        HoaDonDao hoaDonDao = new HoaDonDao();
        KhachHangDAO khDao = new KhachHangDAO();

        // Tạo đối tượng DonDatPhong từ các thông tin được nhập vào
        DonDatPhong ddp = createThongTinDatPhong();

        // Tạo đối tượng HoaDon và đặt thông tin
        HoaDon hoaDon = new HoaDon();
        hoaDon.setTenKhachHang(tenKhachHang: ddp.getKhachHang().getHoTenKH()); // Đặt tên khách hàng từ đối tượng DonDatPhong
        hoaDon.setNhanVien(nhanVien: ShareData.taiKhoanDangNhap.getNhanVien()); // Nếu cần đặt nhân viên từ dữ liệu đã chia sẻ

        // Thêm hóa đơn vào cơ sở dữ liệu
        hoaDonDao.addHoaDon(hoaDon);

        // Thêm đơn đặt phòng vào cơ sở dữ liệu và xử lý kết quả
        if (ddpDAO.addThongTinDatPhong(donDatPhong: ddp)) {
            MessageDialogHelpers.showMessageDialog(parent:mainFrame, title: "Thong bao", content: "Thue phong thanh cong");
            khDao.updateSoLanDen(khachHang:ddp.getKhachHang());
            restTable();
            restTxt();

        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



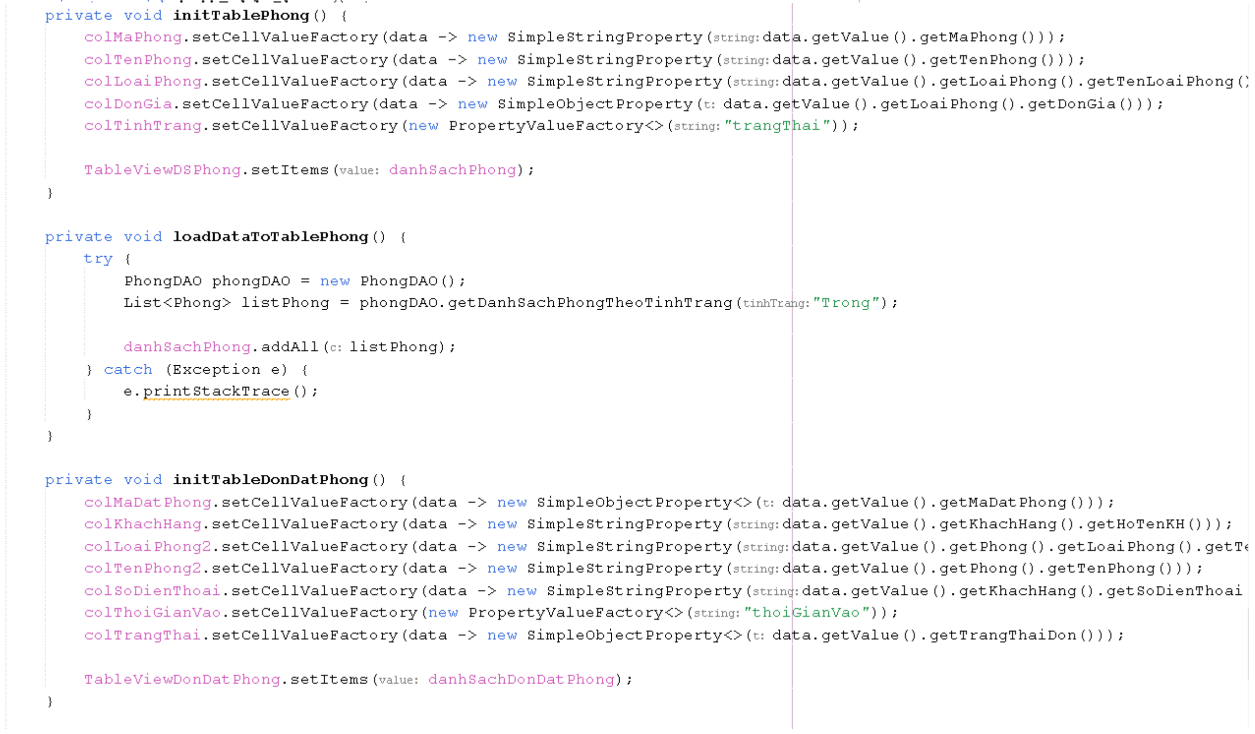
```
private DonDatPhong createThongTinDatPhong() {
    DonDatPhong donDatPhong = new DonDatPhong();
    KhachHangDAO khachHangDAO = new KhachHangDAO();
    PhongDAO phongDAO = new PhongDAO();

    // Lấy thông tin khách hàng từ số điện thoại nhập vào
    KhachHang kh = khachHangDAO.getKhachHangTheoSĐT(soDienThoai: txt_TimSDT.getText());

    // Lấy phòng đã chọn từ TableViewDSPhong
    Phong selectedPhong = TableViewDSPhong.getSelectionModel().getSelectedItem();

    if (selectedPhong != null) {
        // Cập nhật trạng thái phòng và lưu vào cơ sở dữ liệu
        selectedPhong.setTrangThai(trangThai:"Dang Su Dung");
        phongDAO.updatePhong(phong: selectedPhong);

        // Thiết lập thông tin cho đơn đặt phòng
        donDatPhong.setPhong(phong: selectedPhong);
        donDatPhong.setKhachHang(khachHang:kh);
        donDatPhong.setThoiGianVao(new Date());
        donDatPhong.setTrangThaiDon(trangThaiDon: "Chua Thanh Toan");
    } else {
        // Hiển thị thông báo nếu không có phòng nào được chọn
        MessageDialogHelpers.showErrorDialog(parent:nutThueNgay.getScene().getWindow(), title: "Canh bao", content: "Chon 1 phong
    }

    return donDatPhong;
}


@FXML
```



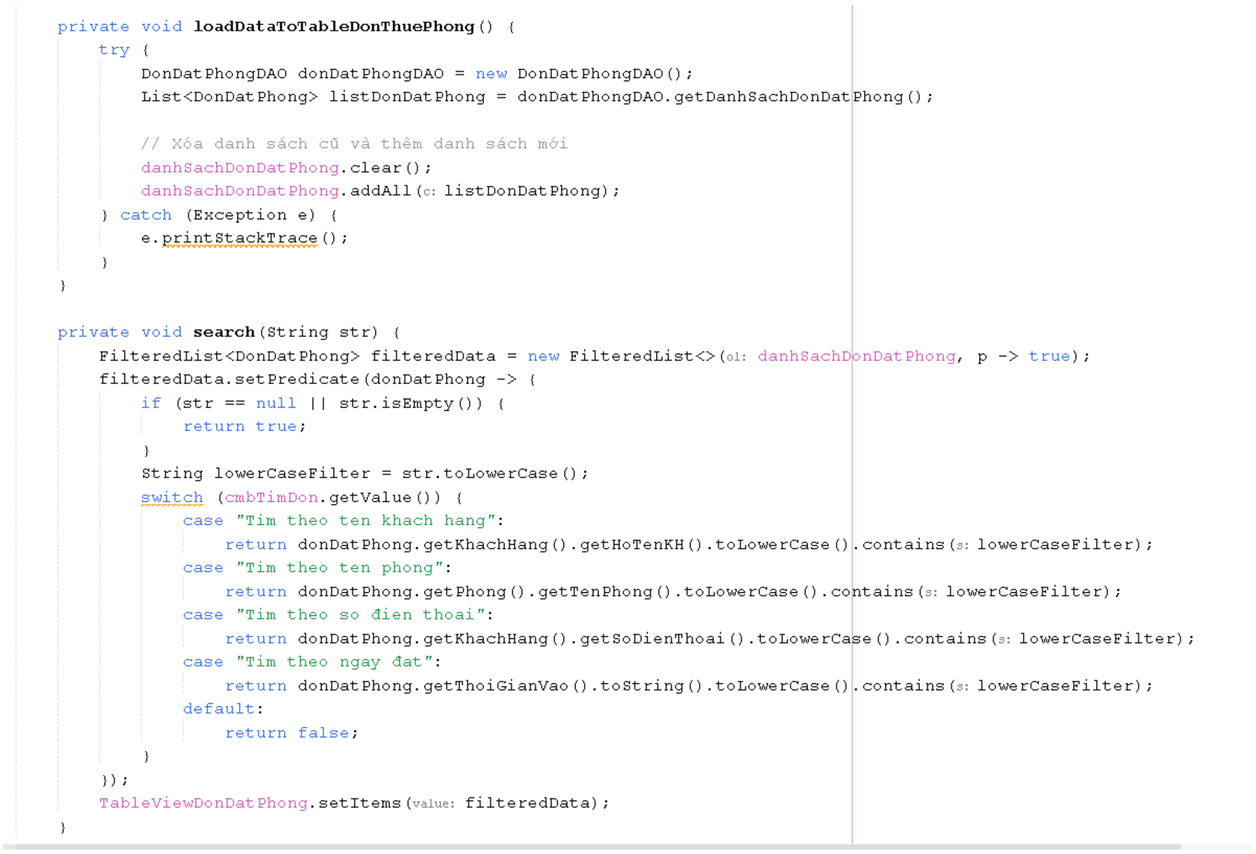
```
@FXML
void btn_Tim(ActionEvent event) {
    KhachHangDAO khachHangDAO = new KhachHangDAO();
    KhachHang kh = khachHangDAO.getKhachHangTheoSĐT(soDienThoai: txt_TimSDT.getText());
    if (kh != null) {
        txt_MaKH.setText(value: kh.getMaKhachHang());
        txt_TenKH.setText(value: kh.getHoTenKH());
        txt_LoaiKH.setText(value: kh.getLoaiKhachHang());
        txt_TimSDT.setText(value: kh.getSoDienThoai());
    } else {
        showDialogChuaCoKhachHang();
    }
}

private void showDialogChuaCoKhachHang() {
    try {
        FXMLLoader loader = new FXMLLoader(url:getClass().getResource(name: "DialogChuaCoKhachHang.fxml"));
        Parent parent = loader.load();

        // Get the controller instance
        DialogChuaCoKhachHangController controller = loader.getController();

        // Create a new stage for the dialog
        Stage stage = new Stage();
        stage.initModality(modality: Modality.APPLICATION_MODAL);
        stage.setScene(new Scene(parent));
        stage.setTitle(value: "Thông Báo");
        stage.show();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```



```
@Override
public void initialize(URL url, ResourceBundle rb) {
    initTablePhong();
    loadDataToTablePhong();
    initTableDonDatPhong();
    loadDataToTableDonThuePhong(); // Chỉ gọi một lần trong phương thức initialize

    txtTimDon.textProperty().addListener((observable, oldValue, newValue) -> search(str: newValue));

    cmbTimDon.getItems().addAll(
    elements: "Tim theo ten khach hang",
    elements: "Tim theo ngay đat",
    elements: "Tim theo ten phong",
    elements: "Tim theo so dien thoai"
    );

    cmb_LocPhongTheoTrangThai.getItems().addAll(
    elements: "Da Thanh Toan",
    elements: "Chua Thanh Toan",
    elements: "Dat Truoc"
    );
    cmb_LocPhongTheoTrangThai.setOnAction(event -> filterBookingsByStatus());
}

private void filterBookingsByStatus() {
    DonDatPhongDAO donDatPhongDao = new DonDatPhongDAO();
    String trangThaiDon = cmb_LocPhongTheoTrangThai.getSelectionModel().getSelectedItem();

    List<DonDatPhong> dsDonDatPhong = donDatPhongDao.getDanhSachDonDatPhong(trangThaiDon);
    danhSachDonDatPhong.clear();
    danhSachDonDatPhong.addAll(c: dsDonDatPhong);

    TableViewDonDatPhong.refresh();
}
```

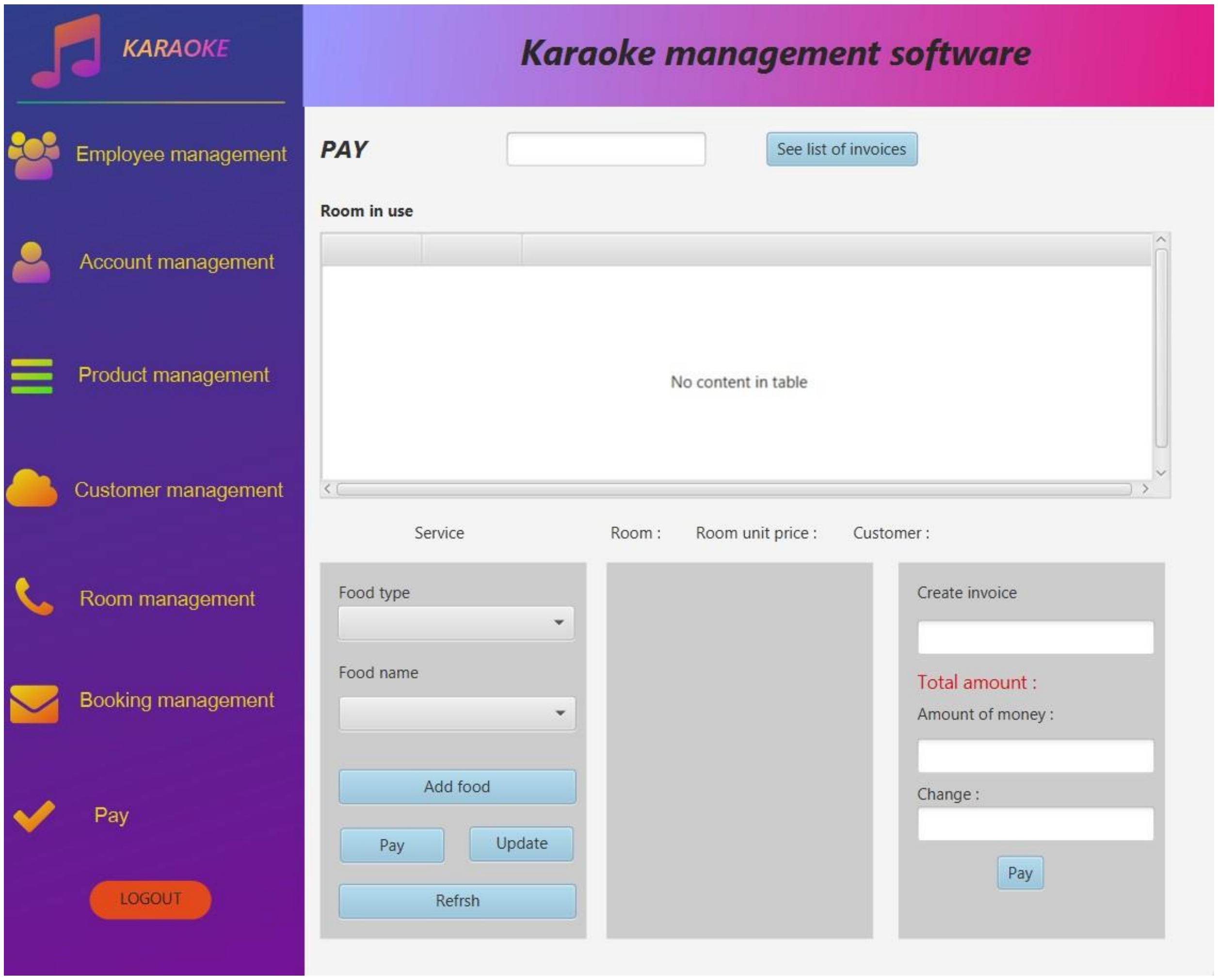


```
private void initTablePhong() {
    colMaPhong.setCellValueFactory(data -> new SimpleStringProperty(string: data.getValue().getMaPhong()));
    colTenPhong.setCellValueFactory(data -> new SimpleStringProperty(string: data.getValue().getTenPhong()));
    colLoaiPhong.setCellValueFactory(data -> new SimpleStringProperty(string: data.getValue().getLoaiPhong().getTenLoaiPhong()
    colDonGia.setCellValueFactory(data -> new SimpleObjectProperty(t: data.getValue().getLoaiPhong().getDonGia()));
    colTinhTrang.setCellValueFactory(new PropertyValueFactory<>(string: "trangThai"));

    TableViewDSPhong.setItems(value: danhSachPhong);
}

private void loadDataToTablePhong() {
    try {
        PhongDAO phongDAO = new PhongDAO();
        List<Phong> listPhong = phongDAO.getDanhSachPhongTheoTinhTrang(tinhTrang: "Trong");

        danhSachPhong.addAll(c: listPhong);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void initTableDonDatPhong() {
    colMaDatPhong.setCellValueFactory(data -> new SimpleObjectProperty<>(t: data.getValue().getMaDatPhong()));
    colKhachHang.setCellValueFactory(data -> new SimpleStringProperty(string: data.getValue().getKhachHang().getHoTenKH()));
    colLoaiPhong2.setCellValueFactory(data -> new SimpleStringProperty(string: data.getValue().getPhong().getLoaiPhong().getTe
    colTenPhong2.setCellValueFactory(data -> new SimpleStringProperty(string: data.getValue().getPhong().getTenPhong()));
    colSoDienThoai.setCellValueFactory(data -> new SimpleStringProperty(string: data.getValue().getKhachHang().getSoDienThoai
    colThoiGianVao.setCellValueFactory(new PropertyValueFactory<>(string: "thoiGianVao"));
    colTrangThai.setCellValueFactory(data -> new SimpleObjectProperty<>(t: data.getValue().getTrangThaiDon()));

    TableViewDonDatPhong.setItems(value: danhSachDonDatPhong);
}
```



```
private void loadDataToTableDonThuePhong() {
    try {
        DonDatPhongDAO donDatPhongDAO = new DonDatPhongDAO();
        List<DonDatPhong> listDonDatPhong = donDatPhongDAO.getDanhSachDonDatPhong();

        // Xóa danh sách cũ và thêm danh sách mới
        danhSachDonDatPhong.clear();
        danhSachDonDatPhong.addAll(c: listDonDatPhong);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void search(String str) {
    FilteredList<DonDatPhong> filteredData = new FilteredList<>(ol: danhSachDonDatPhong, p -> true);
    filteredData.setPredicate(donDatPhong -> {
        if (str == null || str.isEmpty()) {
            return true;
        }
        String lowerCaseFilter = str.toLowerCase();
        switch (cmbTimDon.getValue()) {
            case "Tim theo ten khach hang":
                return donDatPhong.getKhachHang().getHoTenKH().toLowerCase().contains(s: lowerCaseFilter);
            case "Tim theo ten phong":
                return donDatPhong.getPhong().getTenPhong().toLowerCase().contains(s: lowerCaseFilter);
            case "Tim theo so đien thoai":
                return donDatPhong.getKhachHang().getSoDienThoai().toLowerCase().contains(s: lowerCaseFilter);
            case "Tim theo ngay đat":
                return donDatPhong.getThoiGianVao().toString().toLowerCase().contains(s: lowerCaseFilter);
            default:
                return false;
        }
    });
    TableViewDonDatPhong.setItems(value: filteredData);
}
```

7.Payment Management



1: Click the "Pay" button to display the payment interface

2: Select the booking you want to order the service

3: Fields display customer information, room and room price

4: Select product type to filter products by type

5: Select products to add food to customers

6: Enter the quantity of food to order

7: Click the “Add food” button to add food

8: Select the food to delete in the invoice details list , press the "Delete" button to delete the food

9: Select the food to be changed in the invoice detail list , enter the quantity to be changed then press the "Update" button to update the food quantity

10: Click the “Refresh” button to refresh the data

11: List of invoice details for 1 invoice

12: Total customer service fee

13: Display employee information currently logged in

14: Total amount of the bill

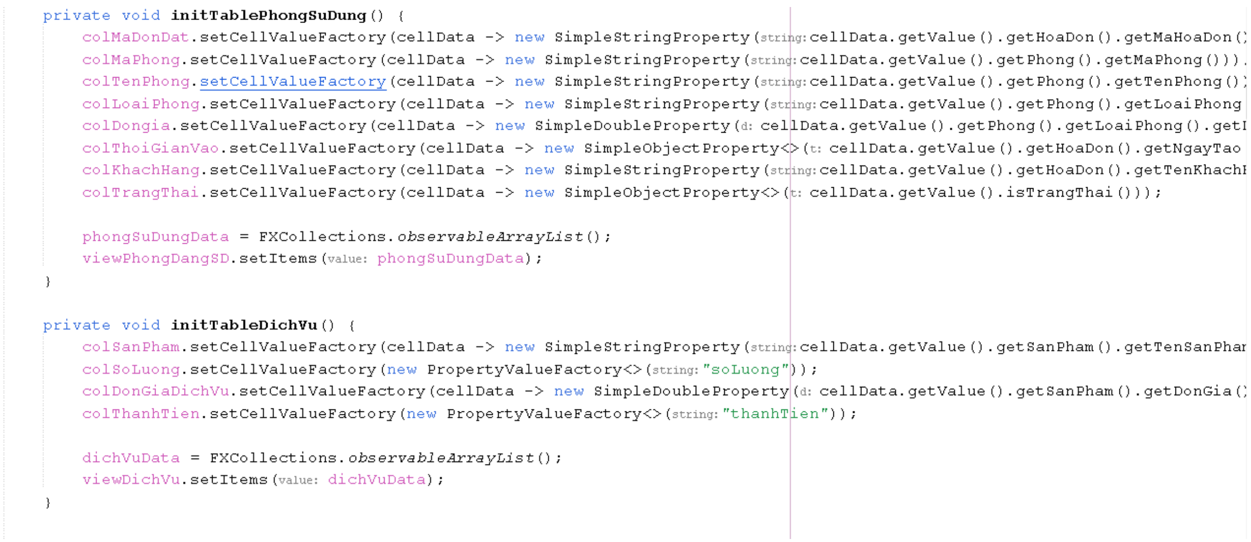
15: Enter the amount given by the customer

16: Excess money needs to be returned to customers

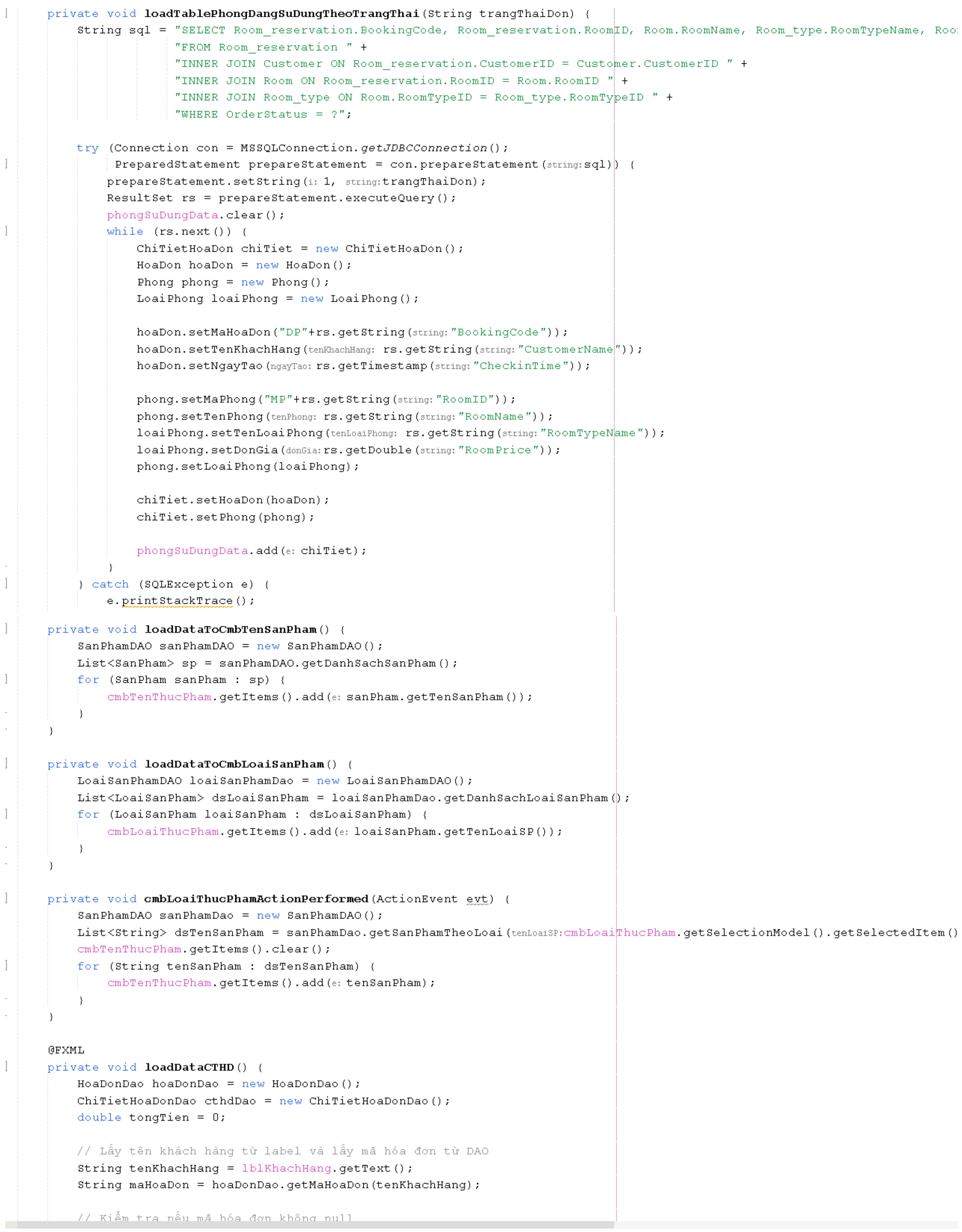
17: Click the “Pay” button to pay the bill

18: Click the "View invoice list" button to see the list of invoices by employee

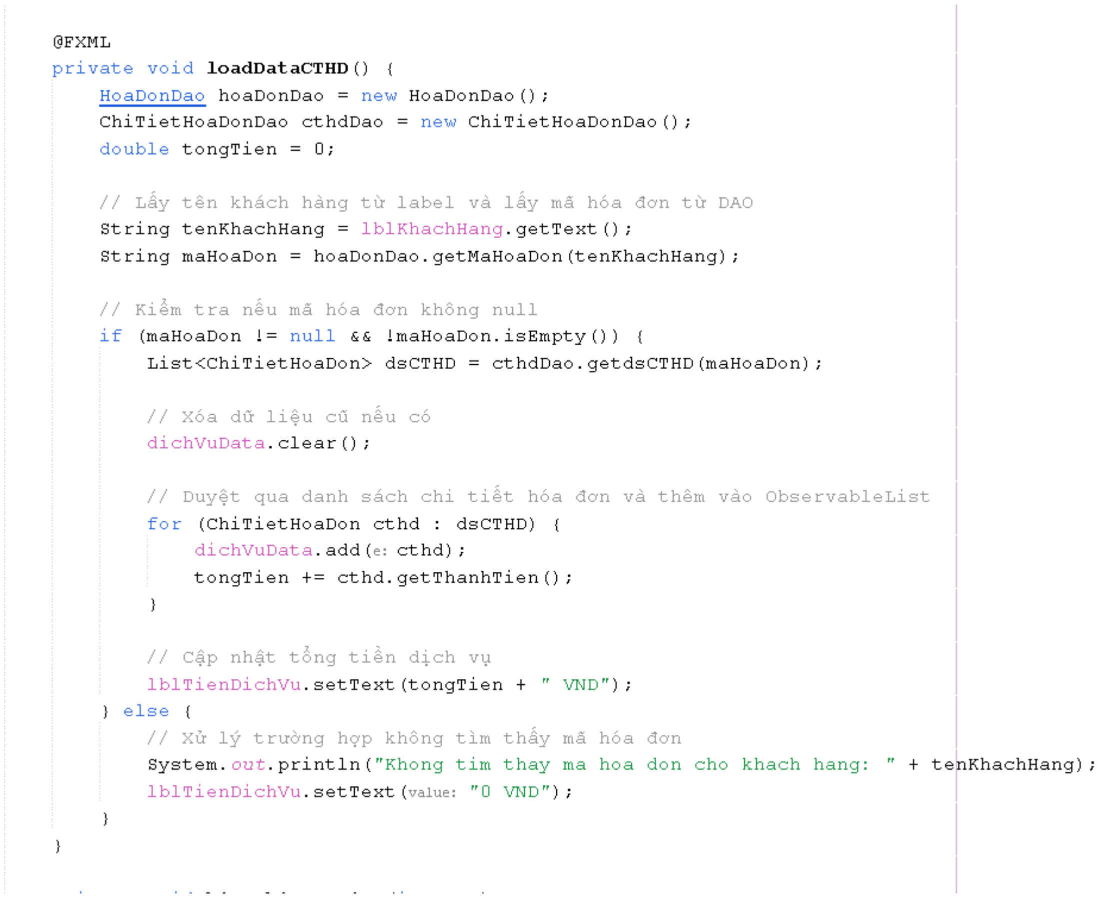
19: Select search criteria in the combo box, enter information in the text field, search results will be displayed in the list of rooms in use



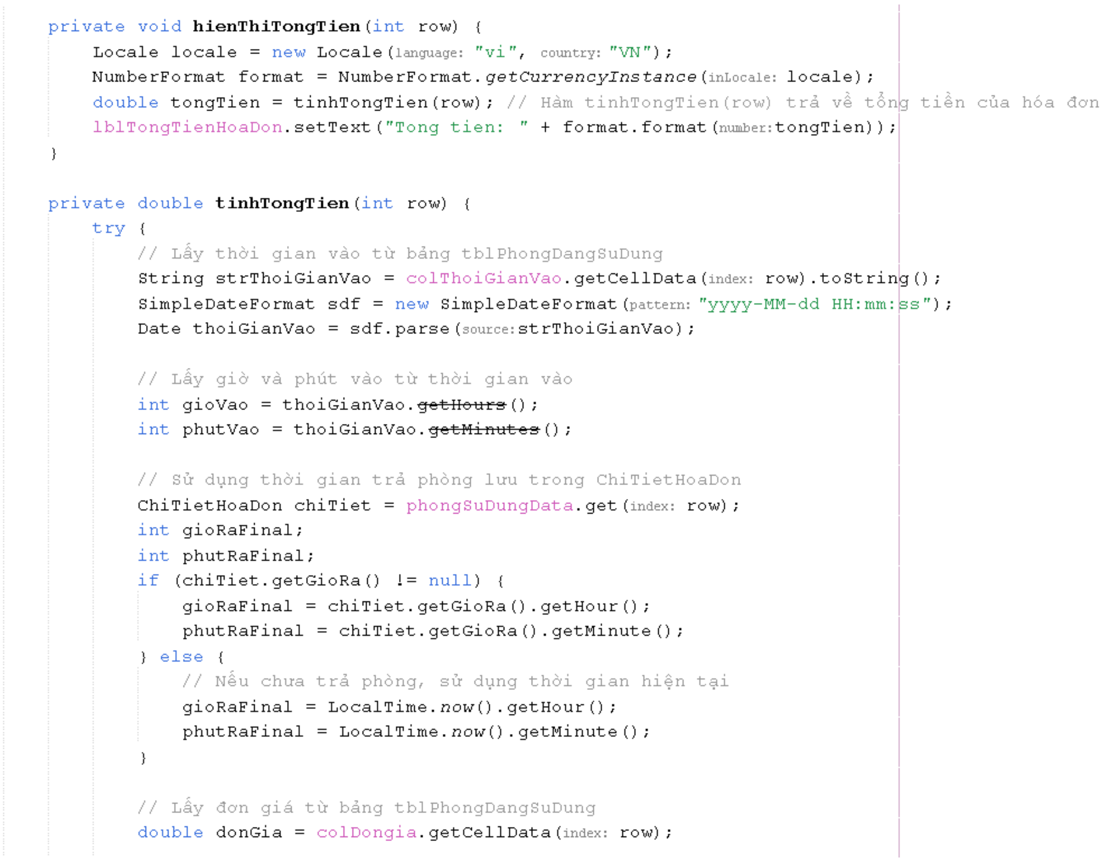
```
private void initTablePhongSuDung() {
    colMaDonDat.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getHoaDon().getMaHoaDon()
    colMaPhong.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getPhong().getMaPhong())).
    colTenPhong.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getPhong().getTenPhong()
    colLoaiPhong.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getPhong().getLoaiPhong
    colDongia.setCellValueFactory(cellData -> new SimpleDoubleProperty(d: cellData.getValue().getPhong().getLoaiPhong().getI
    colThoiGianVao.setCellValueFactory(cellData -> new SimpleObjectProperty<>(t: cellData.getValue().getHoaDon().getNgayTao
    colKhachHang.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getHoaDon().getTenKhachI
    colTrangThai.setCellValueFactory(cellData -> new SimpleObjectProperty<>(t: cellData.getValue().isTrangThai()));

    phongSuDungData = FXCollections.observableArrayList();
    viewPhongDangSD.setItems(value: phongSuDungData);
}

private void initTableDichVu() {
    colSanPham.setCellValueFactory(cellData -> new SimpleStringProperty(string:cellData.getValue().getSanPham().getTenSanPhar
    colSoLuong.setCellValueFactory(new PropertyValueFactory<>(string: "soLuong"));
    colDonGiaDichVu.setCellValueFactory(cellData -> new SimpleDoubleProperty(d: cellData.getValue().getSanPham().getDonGia()
    colThanhTien.setCellValueFactory(new PropertyValueFactory<>(string: "thanhTien"));

    dichVuData = FXCollections.observableArrayList();
    viewDichVu.setItems(value: dichVuData);
}
```



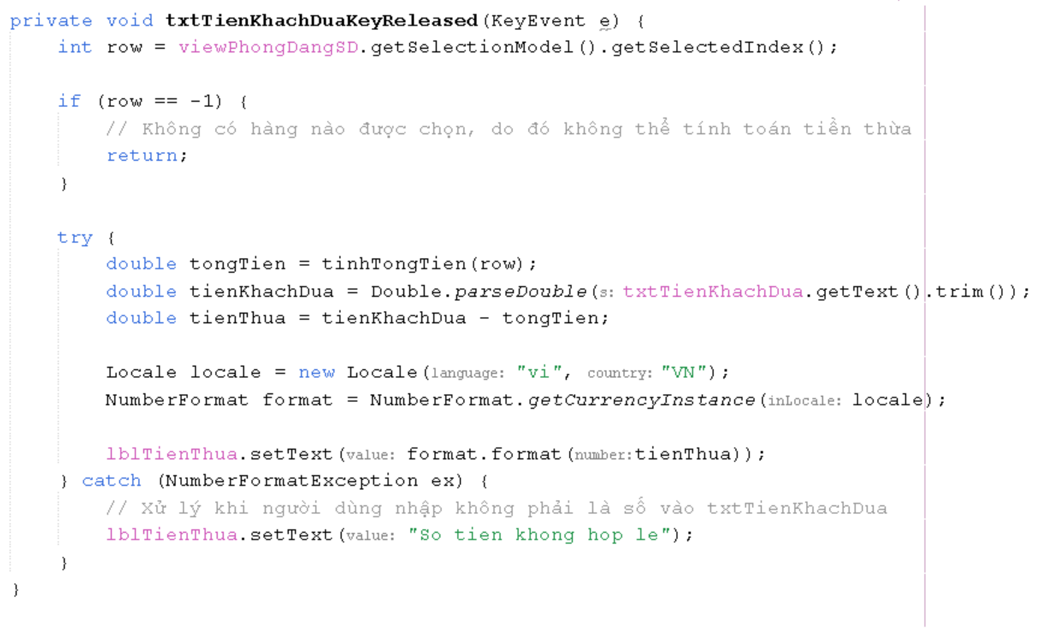
```java
private void loadTablePhongDangSuDungTheoTrangThai(String trangThaiDon) {
    String sql = "SELECT Room_reservation.BookingCode, Room_reservation.RoomID, Room.RoomName, Room_type.RoomTypeName, Roo
                 "FROM Room_reservation " +
                 "INNER JOIN Customer ON Room_reservation.CustomerID = Customer.CustomerID " +
                 "INNER JOIN Room ON Room_reservation.RoomID = Room.RoomID " +
                 "INNER JOIN Room_type ON Room.RoomTypeID = Room_type.RoomTypeID " +
                 "WHERE OrderStatus = ?";

    try (Connection con = MSSQLConnection.getJDBCConnection();
         PreparedStatement prepareStatement = con.prepareStatement(string: sql)) {
        prepareStatement.setString(i: 1, string: trangThaiDon);
        ResultSet rs = prepareStatement.executeQuery();
        phongSuDungData.clear();
        while (rs.next()) {
            ChiTietHoaDon chiTiet = new ChiTietHoaDon();
            HoaDon hoaDon = new HoaDon();
            Phong phong = new Phong();
            LoaiPhong loaiPhong = new LoaiPhong();

            hoaDon.setMaHoaDon("DP"+rs.getString(string: "BookingCode"));
            hoaDon.setTenKhachHang(tenKhachHang: rs.getString(string: "CustomerName"));
            hoaDon.setNgayTao(ngayTao: rs.getTimestamp(string: "CheckinTime"));

            phong.setMaPhong("MP"+rs.getString(string: "RoomID"));
            phong.setTenPhong(tenPhong: rs.getString(string: "RoomName"));
            loaiPhong.setTenLoaiPhong(tenLoaiPhong: rs.getString(string: "RoomTypeName"));
            loaiPhong.setDonGia(donGia: rs.getDouble(string: "RoomPrice"));
            phong.setLoaiPhong(loaiPhong);

            chiTiet.setHoaDon(hoaDon);
            chiTiet.setPhong(phong);

            phongSuDungData.add(e: chiTiet);
        }
    } catch (SQLException e) {
        e.printStackTrace();
private void loadDataToCmbTenSanPham() {
    SanPhamDAO sanPhamDAO = new SanPhamDAO();
    List<SanPham> sp = sanPhamDAO.getDanhSachSanPham();
    for (SanPham sanPham : sp) {
        cmbTenThucPham.getItems().add(e: sanPham.getTenSanPham());
    }
}

private void loadDataToCmbLoaiSanPham() {
    LoaiSanPhamDAO loaiSanPhamDao = new LoaiSanPhamDAO();
    List<LoaiSanPham> dsLoaiSanPham = loaiSanPhamDao.getDanhSachLoaiSanPham();
    for (LoaiSanPham loaiSanPham : dsLoaiSanPham) {
        cmbLoaiThucPham.getItems().add(e: loaiSanPham.getTenLoaiSP());
    }
}

private void cmbLoaiThucPhamActionPerformed(ActionEvent evt) {
    SanPhamDAO sanPhamDao = new SanPhamDAO();
    List<String> dsTenSanPham = sanPhamDao.getSanPhamTheoLoai(tenLoaiSP: cmbLoaiThucPham.getSelectionModel().getSelectedItem()
    cmbTenThucPham.getItems().clear();
    for (String tenSanPham : dsTenSanPham) {
        cmbTenThucPham.getItems().add(e: tenSanPham);
    }
}

@FXML
private void loadDataCTHD() {
    HoaDonDao hoaDonDao = new HoaDonDao();
    ChiTietHoaDonDao cthdDao = new ChiTietHoaDonDao();
    double tongTien = 0;

    // Lấy tên khách hàng từ label và lấy mã hóa đơn từ DAO
    String tenKhachHang = lblKhachHang.getText();
    String maHoaDon = hoaDonDao.getMaHoaDon(tenKhachHang);

    // Kiểm tra nếu mã hóa đơn không null
```



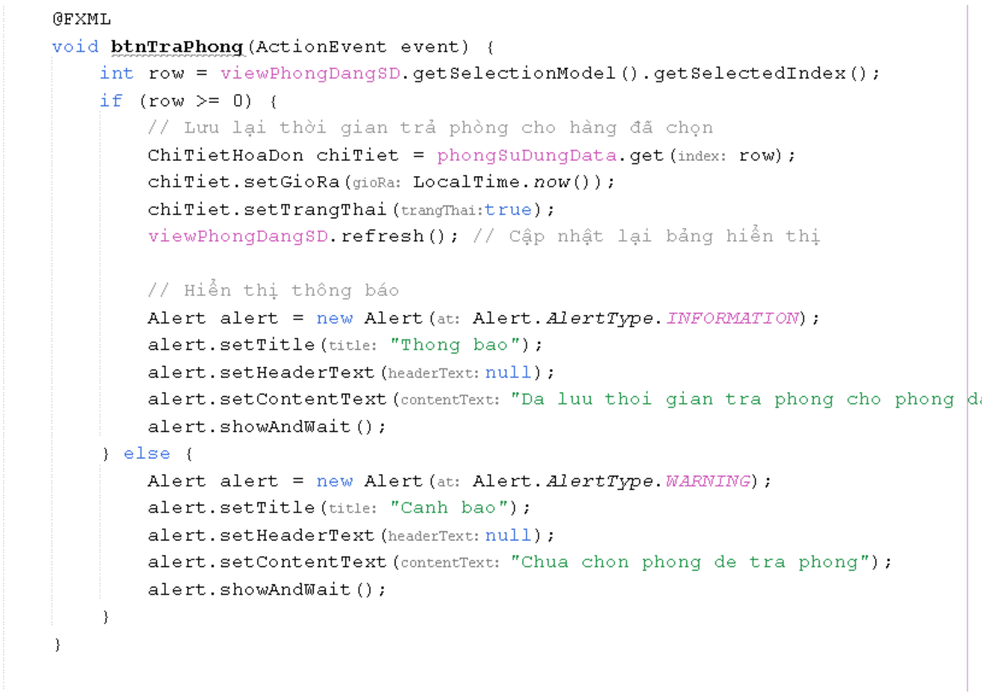
```
@FXML
private void loadDataCTHD() {
    HoaDonDao hoaDonDao = new HoaDonDao();
    ChiTietHoaDonDao cthdDao = new ChiTietHoaDonDao();
    double tongTien = 0;

    // Lấy tên khách hàng từ label và lấy mã hóa đơn từ DAO
    String tenKhachHang = lblKhachHang.getText();
    String maHoaDon = hoaDonDao.getMaHoaDon(tenKhachHang);

    // Kiểm tra nếu mã hóa đơn không null
    if (maHoaDon != null && !maHoaDon.isEmpty()) {
        List<ChiTietHoaDon> dsCTHD = cthdDao.getdsCTHD(maHoaDon);

        // Xóa dữ liệu cũ nếu có
        dichVuData.clear();

        // Duyệt qua danh sách chi tiết hóa đơn và thêm vào ObservableList
        for (ChiTietHoaDon cthd : dsCTHD) {
            dichVuData.add(e: cthd);
            tongTien += cthd.getThanhTien();
        }

        // Cập nhật tổng tiền dịch vụ
        lblTienDichVu.setText(tongTien + " VND");
    } else {
        // Xử lý trường hợp không tìm thấy mã hóa đơn
        System.out.println("Khong tim thay ma hoa don cho khach hang: " + tenKhachHang);
        lblTienDichVu.setText(value: "0 VND");
    }
}
```



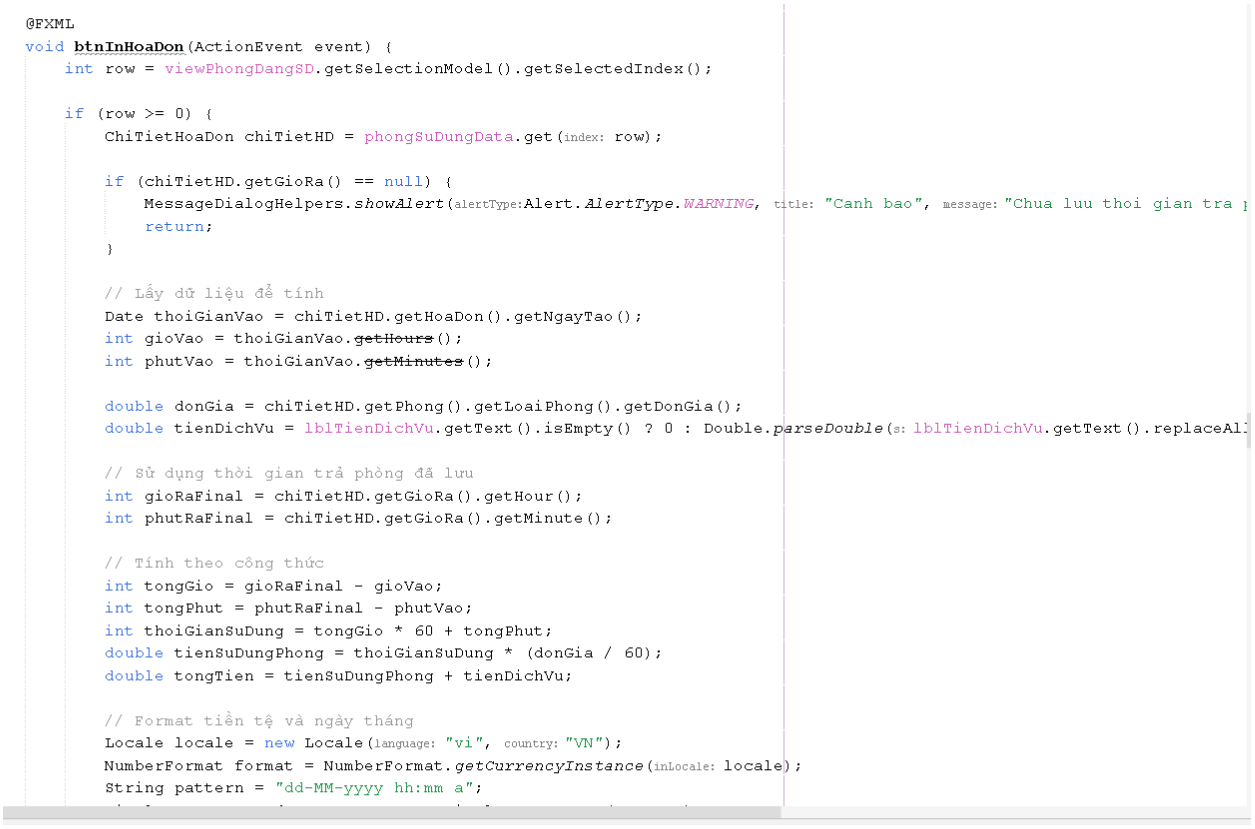
```java
    private void hienThiTongTien(int row) {
        Locale locale = new Locale(language: "vi", country: "VN");
        NumberFormat format = NumberFormat.getCurrencyInstance(inLocale: locale);
        double tongTien = tinhTongTien(row); // Hàm tinhTongTien(row) trả về tổng tiền của hóa đơn
        lblTongTienHoaDon.setText("Tong tien: " + format.format(number:tongTien));
    }

    private double tinhTongTien(int row) {
        try {
            // Lấy thời gian vào từ bảng tblPhongDangSuDung
            String strThoiGianVao = colThoiGianVao.getCellData(index: row).toString();
            SimpleDateFormat sdf = new SimpleDateFormat(pattern: "yyyy-MM-dd HH:mm:ss");
            Date thoiGianVao = sdf.parse(source:strThoiGianVao);

            // Lấy giờ và phút vào từ thời gian vào
            int gioVao = thoiGianVao.~~getHours~~();
            int phutVao = thoiGianVao.~~getMinutes~~();

            // Sử dụng thời gian trả phòng lưu trong ChiTietHoaDon
            ChiTietHoaDon chiTiet = phongSuDungData.get(index: row);
            int gioRaFinal;
            int phutRaFinal;
            if (chiTiet.getGioRa() != null) {
                gioRaFinal = chiTiet.getGioRa().getHour();
                phutRaFinal = chiTiet.getGioRa().getMinute();
            } else {
                // Nếu chưa trả phòng, sử dụng thời gian hiện tại
                gioRaFinal = LocalTime.now().getHour();
                phutRaFinal = LocalTime.now().getMinute();
            }

            // Lấy đơn giá từ bảng tblPhongDangSuDung
            double donGia = colDongia.getCellData(index: row);
```



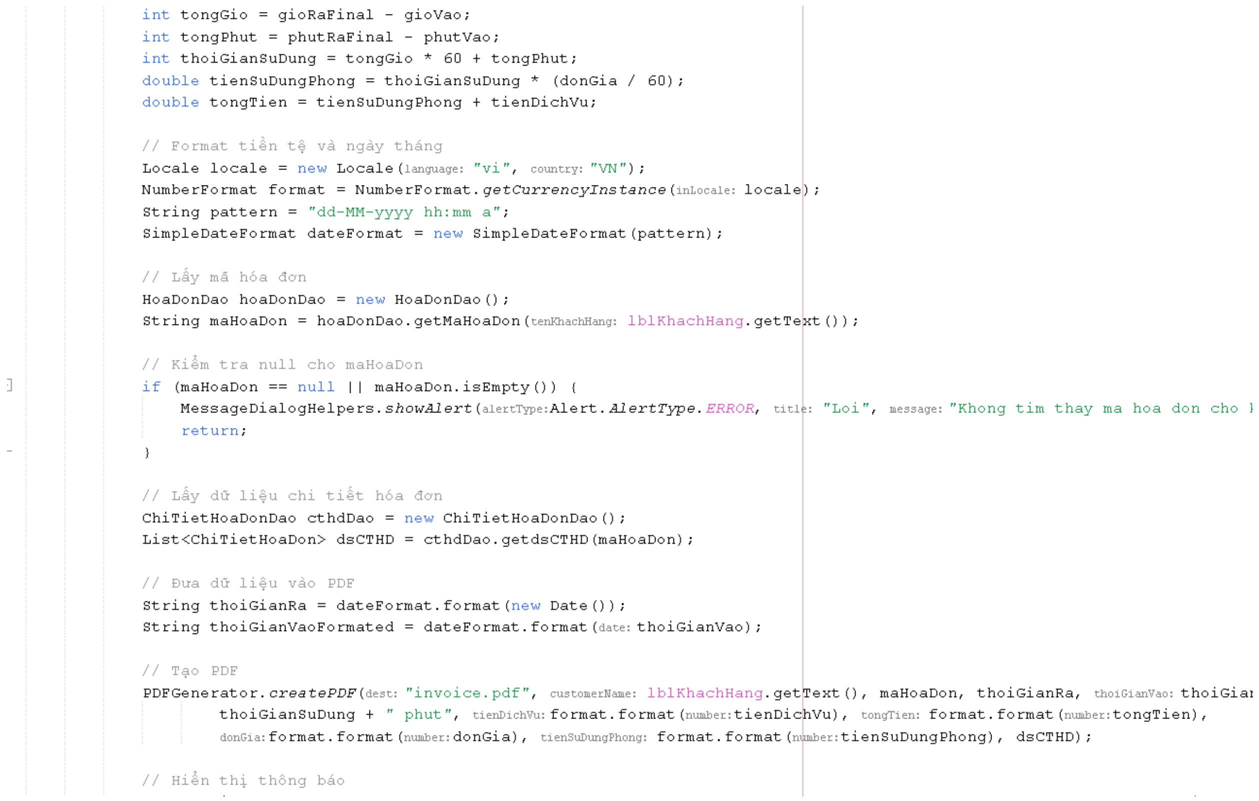
```java
private void txtTienKhachDuaKeyReleased(KeyEvent e) {
    int row = viewPhongDangSD.getSelectionModel().getSelectedIndex();

    if (row == -1) {
        // Không có hàng nào được chọn, do đó không thể tính toán tiền thừa
        return;
    }

    try {
        double tongTien = tinhTongTien(row);
        double tienKhachDua = Double.parseDouble(s: txtTienKhachDua.getText().trim());
        double tienThua = tienKhachDua - tongTien;

        Locale locale = new Locale(language: "vi", country: "VN");
        NumberFormat format = NumberFormat.getCurrencyInstance(inLocale: locale);

        lblTienThua.setText(value: format.format(number:tienThua));
    } catch (NumberFormatException ex) {
        // Xử lý khi người dùng nhập không phải là số vào txtTienKhachDua
        lblTienThua.setText(value: "So tien khong hop le");
    }
}
```



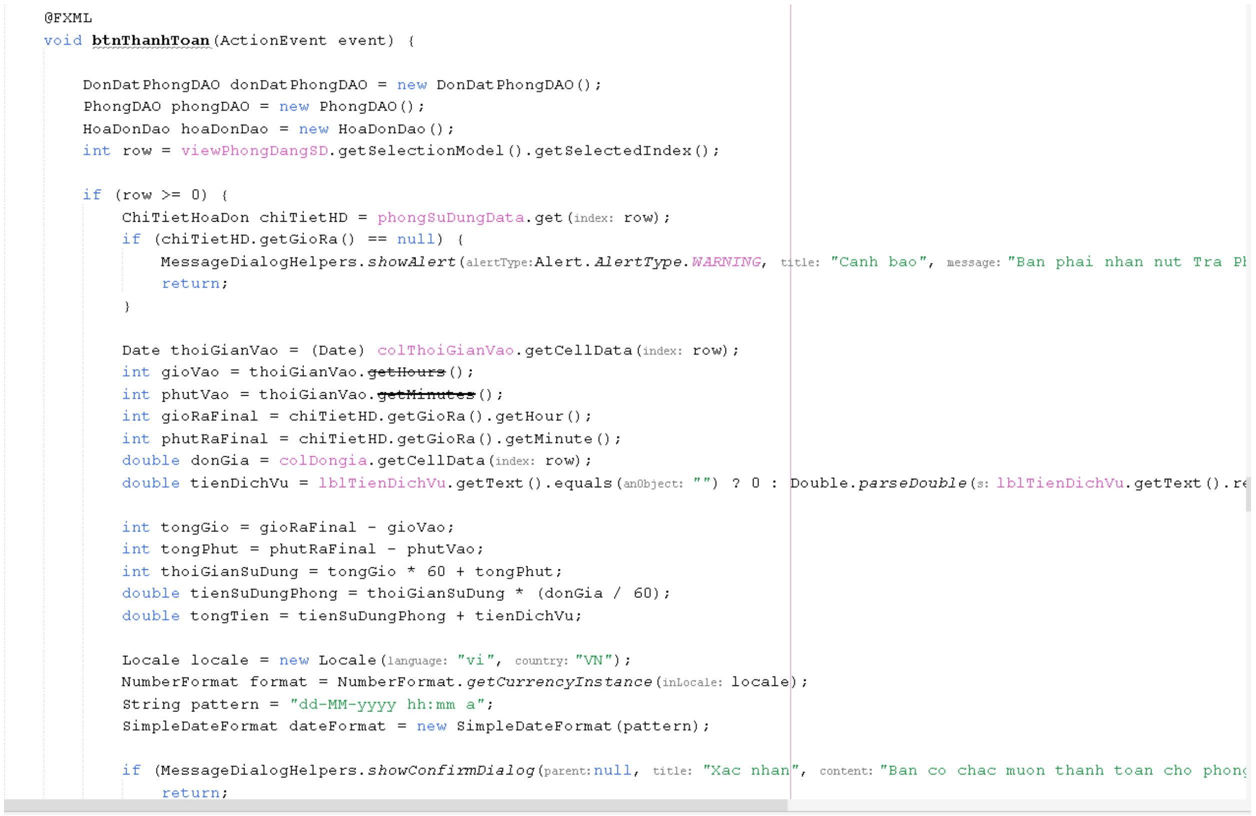
```java
@FXML
void btnTraPhong(ActionEvent event) {
    int row = viewPhongDangSD.getSelectionModel().getSelectedIndex();
    if (row >= 0) {
        // Lưu lại thời gian trả phòng cho hàng đã chọn
        ChiTietHoaDon chiTiet = phongSuDungData.get(index: row);
        chiTiet.setGioRa(gioRa: LocalTime.now());
        chiTiet.setTrangThai(trangThai:true);
        viewPhongDangSD.refresh(); // Cập nhật lại bảng hiển thị

        // Hiển thị thông báo
        Alert alert = new Alert(at: Alert.AlertType.INFORMATION);
        alert.setTitle(title: "Thong bao");
        alert.setHeaderText(headerText: null);
        alert.setContentText(contentText: "Da luu thoi gian tra phong cho phong da chon");
        alert.showAndWait();
    } else {
        Alert alert = new Alert(at: Alert.AlertType.WARNING);
        alert.setTitle(title: "Canh bao");
        alert.setHeaderText(headerText: null);
        alert.setContentText(contentText: "Chua chon phong de tra phong");
        alert.showAndWait();
    }
}
```



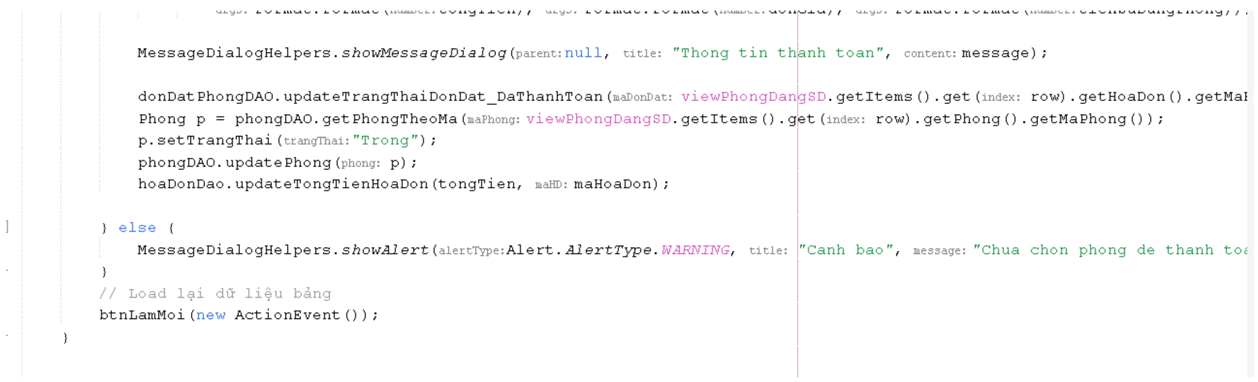
```java
@FXML
void btnInHoaDon(ActionEvent event) {
    int row = viewPhongDangSD.getSelectionModel().getSelectedIndex();

    if (row >= 0) {
        ChiTietHoaDon chiTietHD = phongSuDungData.get(index: row);

        if (chiTietHD.getGioRa() == null) {
            MessageDialogHelpers.showAlert(alertType:Alert.AlertType.WARNING, title: "Canh bao", message: "Chua luu thoi gian tra
            return;
        }

        // Lấy dữ liệu để tính
        Date thoiGianVao = chiTietHD.getHoaDon().getNgayTao();
        int gioVao = thoiGianVao.~~getHours~~();
        int phutVao = thoiGianVao.~~getMinutes~~();

        double donGia = chiTietHD.getPhong().getLoaiPhong().getDonGia();
        double tienDichVu = lblTienDichVu.getText().isEmpty() ? 0 : Double.parseDouble(s: lblTienDichVu.getText().replaceAl

        // Sử dụng thời gian trả phòng đã lưu
        int gioRaFinal = chiTietHD.getGioRa().getHour();
        int phutRaFinal = chiTietHD.getGioRa().getMinute();

        // Tính theo công thức
        int tongGio = gioRaFinal - gioVao;
        int tongPhut = phutRaFinal - phutVao;
        int thoiGianSuDung = tongGio * 60 + tongPhut;
        double tienSuDungPhong = thoiGianSuDung * (donGia / 60);
        double tongTien = tienSuDungPhong + tienDichVu;

        // Format tiền tệ và ngày tháng
        Locale locale = new Locale(language: "vi", country: "VN");
        NumberFormat format = NumberFormat.getCurrencyInstance(inLocale: locale);
        String pattern = "dd-MM-yyyy hh:mm a";
```



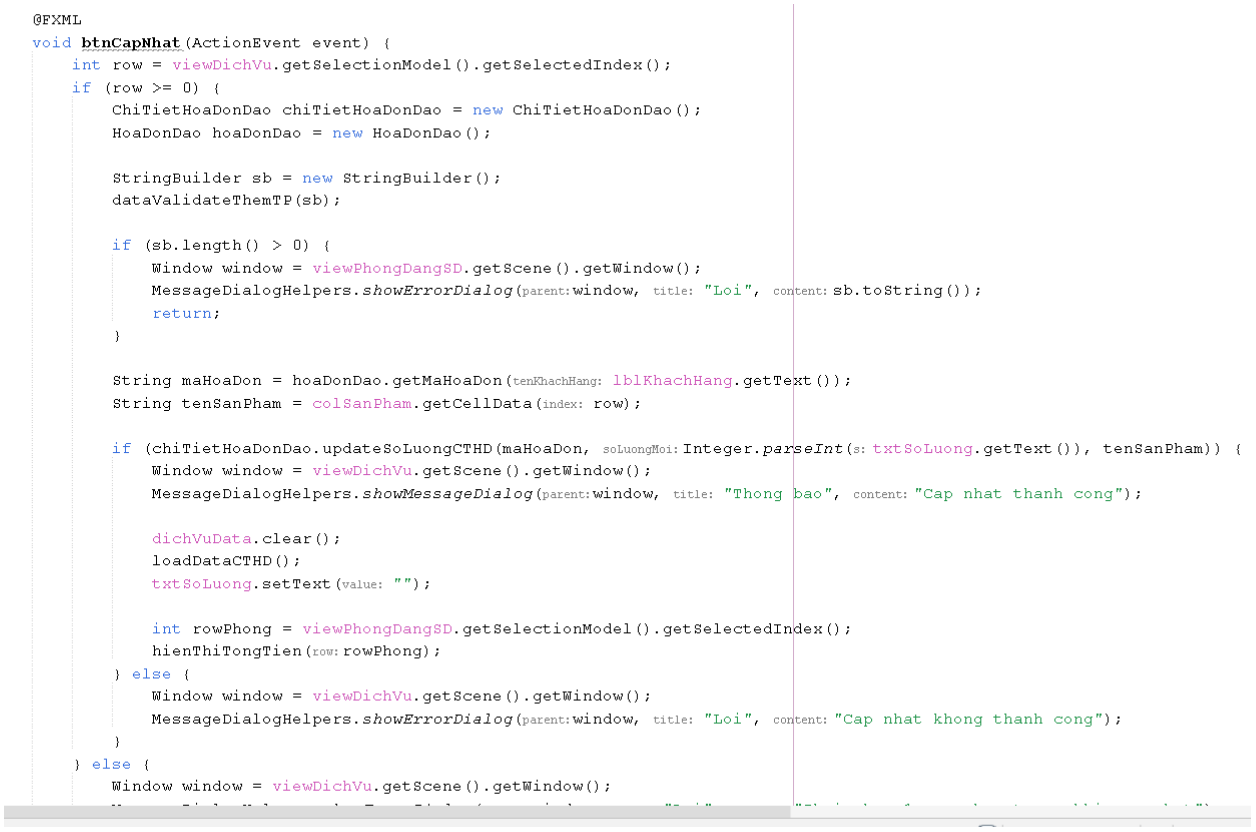
```
        int tongGio = gioRaFinal - gioVao;
        int tongPhut = phutRaFinal - phutVao;
        int thoiGianSuDung = tongGio * 60 + tongPhut;
        double tienSuDungPhong = thoiGianSuDung * (donGia / 60);
        double tongTien = tienSuDungPhong + tienDichVu;

        // Format tiền tệ và ngày tháng
        Locale locale = new Locale(language: "vi", country: "VN");
        NumberFormat format = NumberFormat.getCurrencyInstance(inLocale: locale);
        String pattern = "dd-MM-yyyy hh:mm a";
        SimpleDateFormat dateFormat = new SimpleDateFormat(pattern);

        // Lấy mã hóa đơn
        HoaDonDao hoaDonDao = new HoaDonDao();
        String maHoaDon = hoaDonDao.getMaHoaDon(tenKhachHang: lblKhachHang.getText());

        // Kiểm tra null cho maHoaDon
        if (maHoaDon == null || maHoaDon.isEmpty()) {
            MessageDialogHelpers.showAlert(alertType:Alert.AlertType.ERROR, title: "Loi", message: "Khong tim thay ma hoa don cho k
            return;
        }

        // Lấy dữ liệu chi tiết hóa đơn
        ChiTietHoaDonDao cthdDao = new ChiTietHoaDonDao();
        List<ChiTietHoaDon> dsCTHD = cthdDao.getdsCTHD(maHoaDon);

        // Đưa dữ liệu vào PDF
        String thoiGianRa = dateFormat.format(new Date());
        String thoiGianVaoFormated = dateFormat.format(date: thoiGianVao);

        // Tạo PDF
        PDFGenerator.createPDF(dest: "invoice.pdf", customerName: lblKhachHang.getText(), maHoaDon, thoiGianRa, thoiGianVao: thoiGian
                thoiGianSuDung + " phut", tienDichVu: format.format(number:tienDichVu), tongTien: format.format(number:tongTien),
                donGia: format.format(number:donGia), tienSuDungPhong: format.format(number:tienSuDungPhong), dsCTHD);

        // Hiển thị thông báo
```



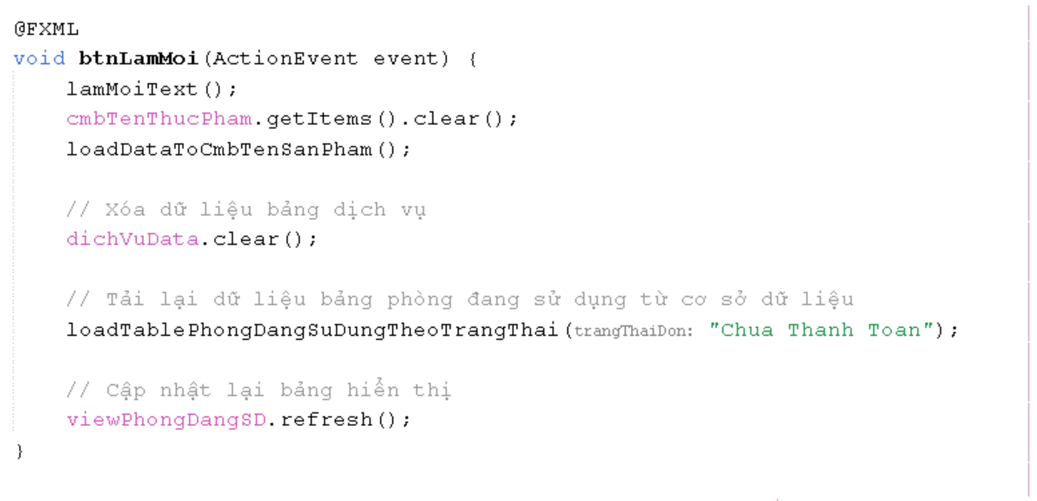
```
@FXML
void btnThanhToan(ActionEvent event) {

    DonDatPhongDAO donDatPhongDAO = new DonDatPhongDAO();
    PhongDAO phongDAO = new PhongDAO();
    HoaDonDao hoaDonDao = new HoaDonDao();
    int row = viewPhongDangSD.getSelectionModel().getSelectedIndex();

    if (row >= 0) {
        ChiTietHoaDon chiTietHD = phongSuDungData.get(index: row);
        if (chiTietHD.getGioRa() == null) {
            MessageDialogHelpers.showAlert(alertType:Alert.AlertType.WARNING, title: "Canh bao", message: "Ban phai nhan nut Tra Ph
            return;
        }

        Date thoiGianVao = (Date) colThoiGianVao.getCellData(index: row);
        int gioVao = thoiGianVao.~~getHours~~();
        int phutVao = thoiGianVao.~~getMinutes~~();
        int gioRaFinal = chiTietHD.getGioRa().getHour();
        int phutRaFinal = chiTietHD.getGioRa().getMinute();
        double donGia = colDongia.getCellData(index: row);
        double tienDichVu = lblTienDichVu.getText().equals(anObject: "") ? 0 : Double.parseDouble(s: lblTienDichVu.getText().re

        int tongGio = gioRaFinal - gioVao;
        int tongPhut = phutRaFinal - phutVao;
        int thoiGianSuDung = tongGio * 60 + tongPhut;
        double tienSuDungPhong = thoiGianSuDung * (donGia / 60);
        double tongTien = tienSuDungPhong + tienDichVu;

        Locale locale = new Locale(language: "vi", country: "VN");
        NumberFormat format = NumberFormat.getCurrencyInstance(inLocale: locale);
        String pattern = "dd-MM-yyyy hh:mm a";
        SimpleDateFormat dateFormat = new SimpleDateFormat(pattern);

        if (MessageDialogHelpers.showConfirmDialog(parent:null, title: "Xac nhan", content: "Ban co chac muon thanh toan cho phong
            return;
```



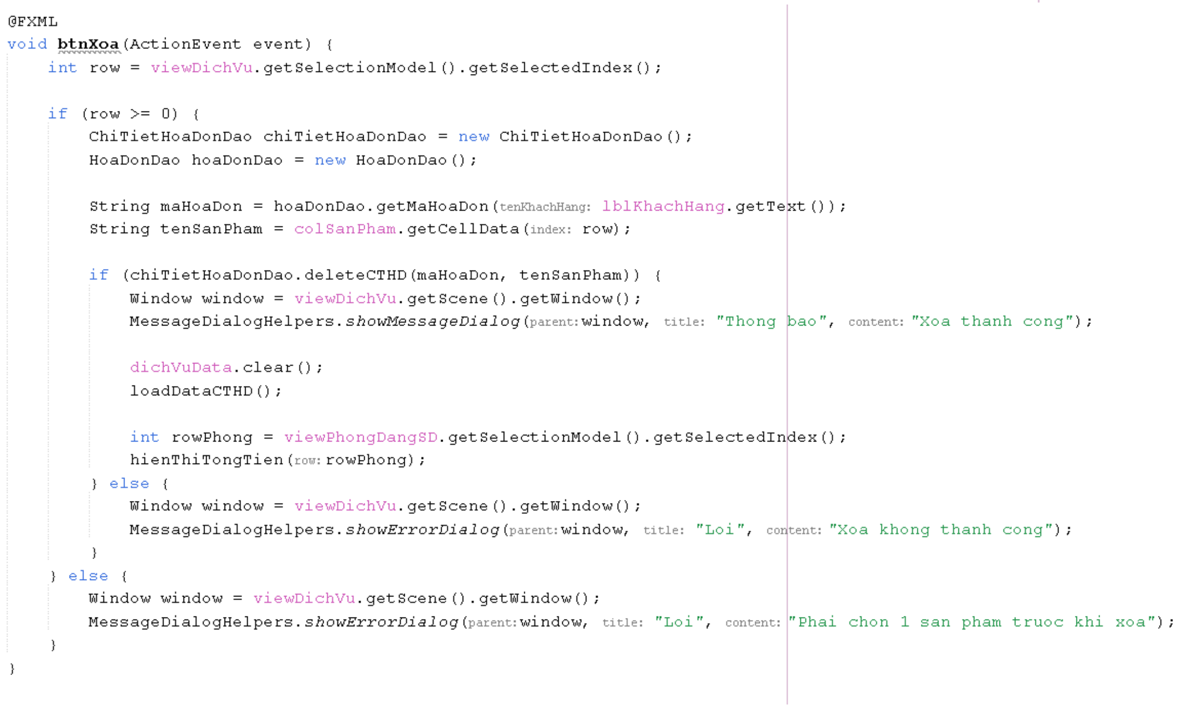
```
                    args: format.format(number:tongTien), args: format.format(number:donGia), args: format.format(number:tienDungPhong));

                MessageDialogHelpers.showMessageDialog(parent:null, title: "Thong tin thanh toan", content: message);

                donDatPhongDAO.updateTrangThaiDonDat_DaThanhToan(maDonDat: viewPhongDangSD.getItems().get(index: row).getHoaDon().getMa
                Phong p = phongDAO.getPhongTheoMa(maPhong: viewPhongDangSD.getItems().get(index: row).getPhong().getMaPhong());
                p.setTrangThai(trangThai:"Trong");
                phongDAO.updatePhong(phong: p);
                hoaDonDao.updateTongTienHoaDon(tongTien, maHD: maHoaDon);

            } else {
                MessageDialogHelpers.showAlert(alertType:Alert.AlertType.WARNING, title: "Canh bao", message: "Chua chon phong de thanh toa
            }
            // Load lại dữ liệu bảng
            btnLamMoi(new ActionEvent());
        }
```



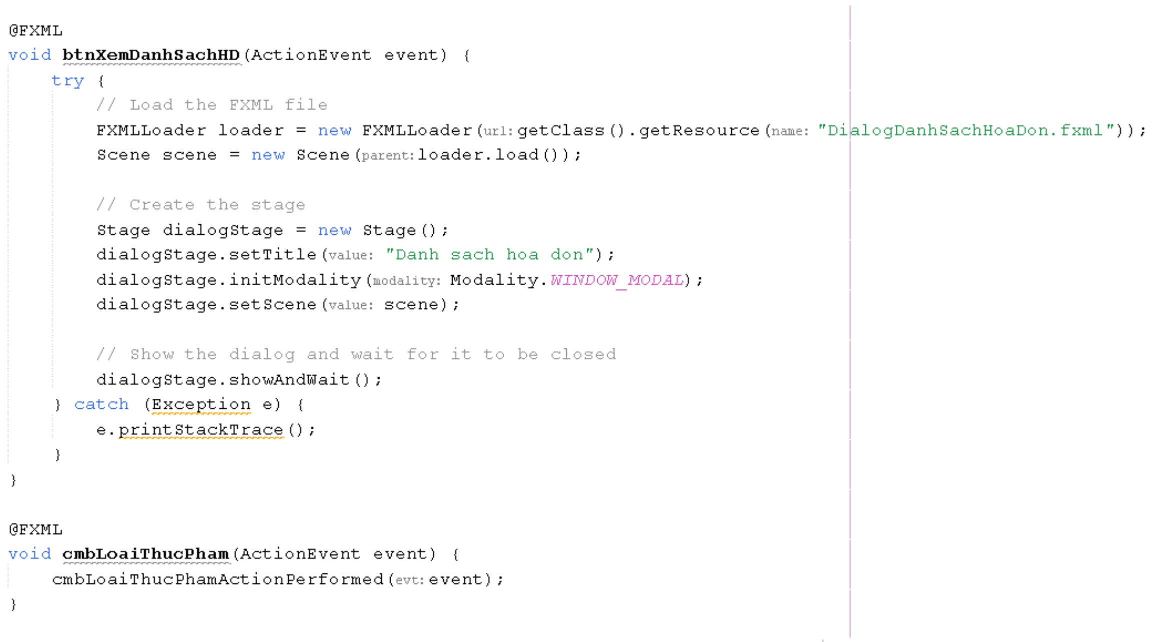
```
@FXML
void btnCapNhat(ActionEvent event) {
    int row = viewDichVu.getSelectionModel().getSelectedIndex();
    if (row >= 0) {
        ChiTietHoaDonDao chiTietHoaDonDao = new ChiTietHoaDonDao();
        HoaDonDao hoaDonDao = new HoaDonDao();

        StringBuilder sb = new StringBuilder();
        dataValidateThemTP(sb);

        if (sb.length() > 0) {
            Window window = viewPhongDangSD.getScene().getWindow();
            MessageDialogHelpers.showErrorDialog(parent:window, title: "Loi", content: sb.toString());
            return;
        }

        String maHoaDon = hoaDonDao.getMaHoaDon(tenKhachHang: lblKhachHang.getText());
        String tenSanPham = colSanPham.getCellData(index: row);

        if (chiTietHoaDonDao.updateSoLuongCTHD(maHoaDon, soLuongMoi: Integer.parseInt(s: txtSoLuong.getText()), tenSanPham)) {
            Window window = viewDichVu.getScene().getWindow();
            MessageDialogHelpers.showMessageDialog(parent:window, title: "Thong bao", content: "Cap nhat thanh cong");

            dichVuData.clear();
            loadDataCTHD();
            txtSoLuong.setText(value: "");

            int rowPhong = viewPhongDangSD.getSelectionModel().getSelectedIndex();
            hienThiTongTien(row: rowPhong);
        } else {
            Window window = viewDichVu.getScene().getWindow();
            MessageDialogHelpers.showErrorDialog(parent:window, title: "Loi", content: "Cap nhat khong thanh cong");
        }
    } else {
        Window window = viewDichVu.getScene().getWindow();
```



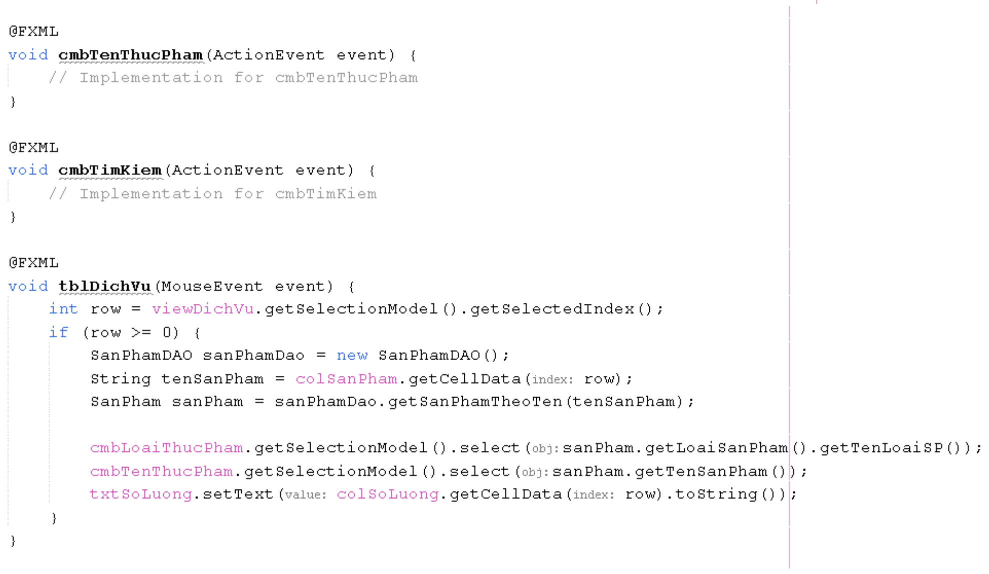
```java
@FXML
void btnLamMoi(ActionEvent event) {
    lamMoiText();
    cmbTenThucPham.getItems().clear();
    loadDataToCmbTenSanPham();

    // Xóa dữ liệu bảng dịch vụ
    dichVuData.clear();

    // Tải lại dữ liệu bảng phòng đang sử dụng từ cơ sở dữ liệu
    loadTablePhongDangSuDungTheoTrangThai(trangThaiDon: "Chua Thanh Toan");

    // Cập nhật lại bảng hiển thị
    viewPhongDangSD.refresh();
}
```



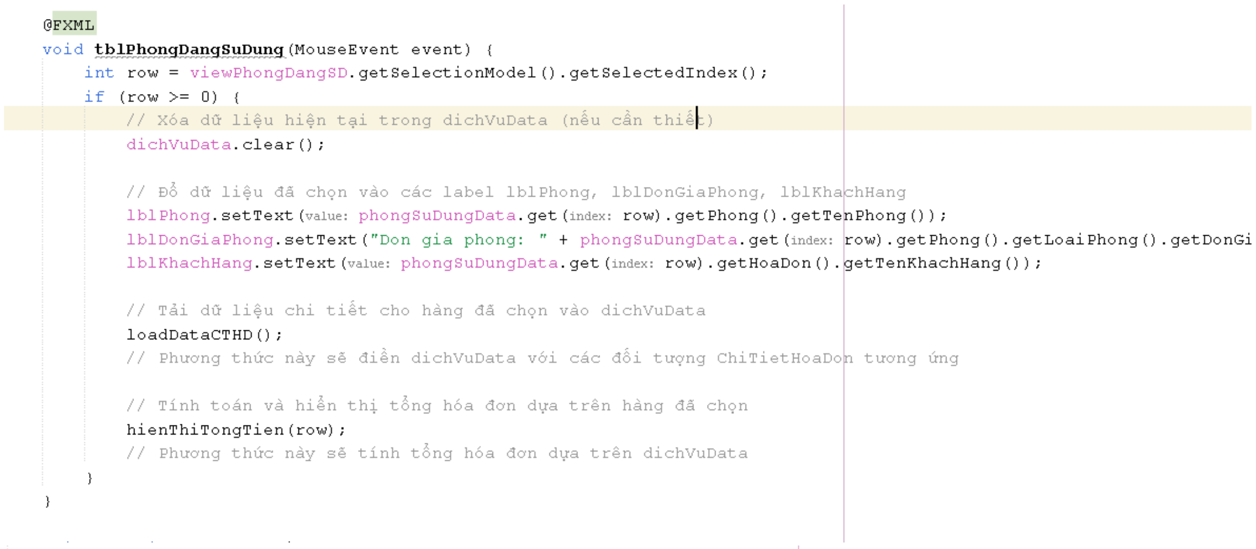
```java
@FXML
void btnXoa(ActionEvent event) {
    int row = viewDichVu.getSelectionModel().getSelectedIndex();

    if (row >= 0) {
        ChiTietHoaDonDao chiTietHoaDonDao = new ChiTietHoaDonDao();
        HoaDonDao hoaDonDao = new HoaDonDao();

        String maHoaDon = hoaDonDao.getMaHoaDon(tenKhachHang: lblKhachHang.getText());
        String tenSanPham = colSanPham.getCellData(index: row);

        if (chiTietHoaDonDao.deleteCTHD(maHoaDon, tenSanPham)) {
            Window window = viewDichVu.getScene().getWindow();
            MessageDialogHelpers.showMessageDialog(parent: window, title: "Thong bao", content: "Xoa thanh cong");

            dichVuData.clear();
            loadDataCTHD();

            int rowPhong = viewPhongDangSD.getSelectionModel().getSelectedIndex();
            hienThiTongTien(row: rowPhong);
        } else {
            Window window = viewDichVu.getScene().getWindow();
            MessageDialogHelpers.showErrorDialog(parent: window, title: "Loi", content: "Xoa khong thanh cong");
        }
    } else {
        Window window = viewDichVu.getScene().getWindow();
        MessageDialogHelpers.showErrorDialog(parent: window, title: "Loi", content: "Phai chon 1 san pham truoc khi xoa");
    }
}
```



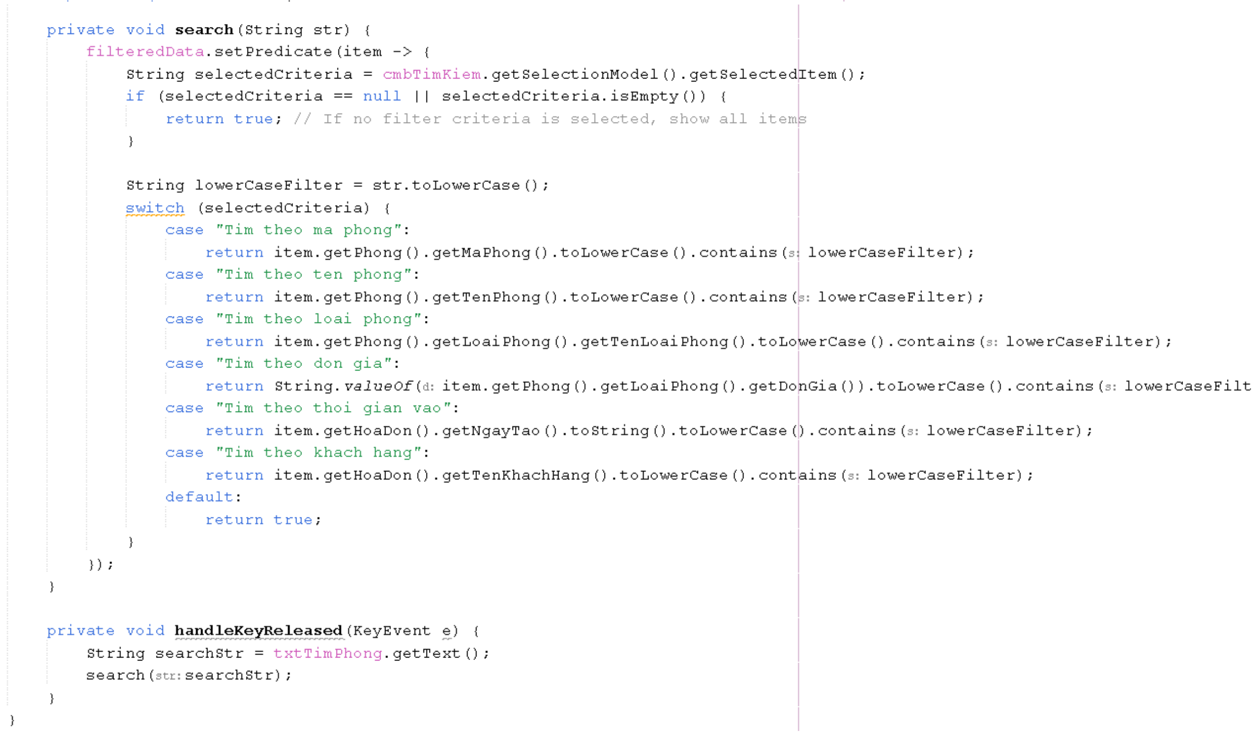
```
@FXML
void btnXemDanhSachHD(ActionEvent event) {
    try {
        // Load the FXML file
        FXMLLoader loader = new FXMLLoader(url: getClass().getResource(name: "DialogDanhSachHoaDon.fxml"));
        Scene scene = new Scene(parent: loader.load());

        // Create the stage
        Stage dialogStage = new Stage();
        dialogStage.setTitle(value: "Danh sach hoa don");
        dialogStage.initModality(modality: Modality.WINDOW_MODAL);
        dialogStage.setScene(value: scene);

        // Show the dialog and wait for it to be closed
        dialogStage.showAndWait();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@FXML
void cmbLoaiThucPham(ActionEvent event) {
    cmbLoaiThucPhamActionPerformed(evt: event);
}
```



```
@FXML
void cmbTenThucPham(ActionEvent event) {
    // Implementation for cmbTenThucPham
}

@FXML
void cmbTimKiem(ActionEvent event) {
    // Implementation for cmbTimKiem
}

@FXML
void tblDichVu(MouseEvent event) {
    int row = viewDichVu.getSelectionModel().getSelectedIndex();
    if (row >= 0) {
        SanPhamDAO sanPhamDao = new SanPhamDAO();
        String tenSanPham = colSanPham.getCellData(index: row);
        SanPham sanPham = sanPhamDao.getSanPhamTheoTen(tenSanPham);

        cmbLoaiThucPham.getSelectionModel().select(obj: sanPham.getLoaiSanPham().getTenLoaiSP());
        cmbTenThucPham.getSelectionModel().select(obj: sanPham.getTenSanPham());
        txtSoLuong.setText(value: colSoLuong.getCellData(index: row).toString());
    }
}
```

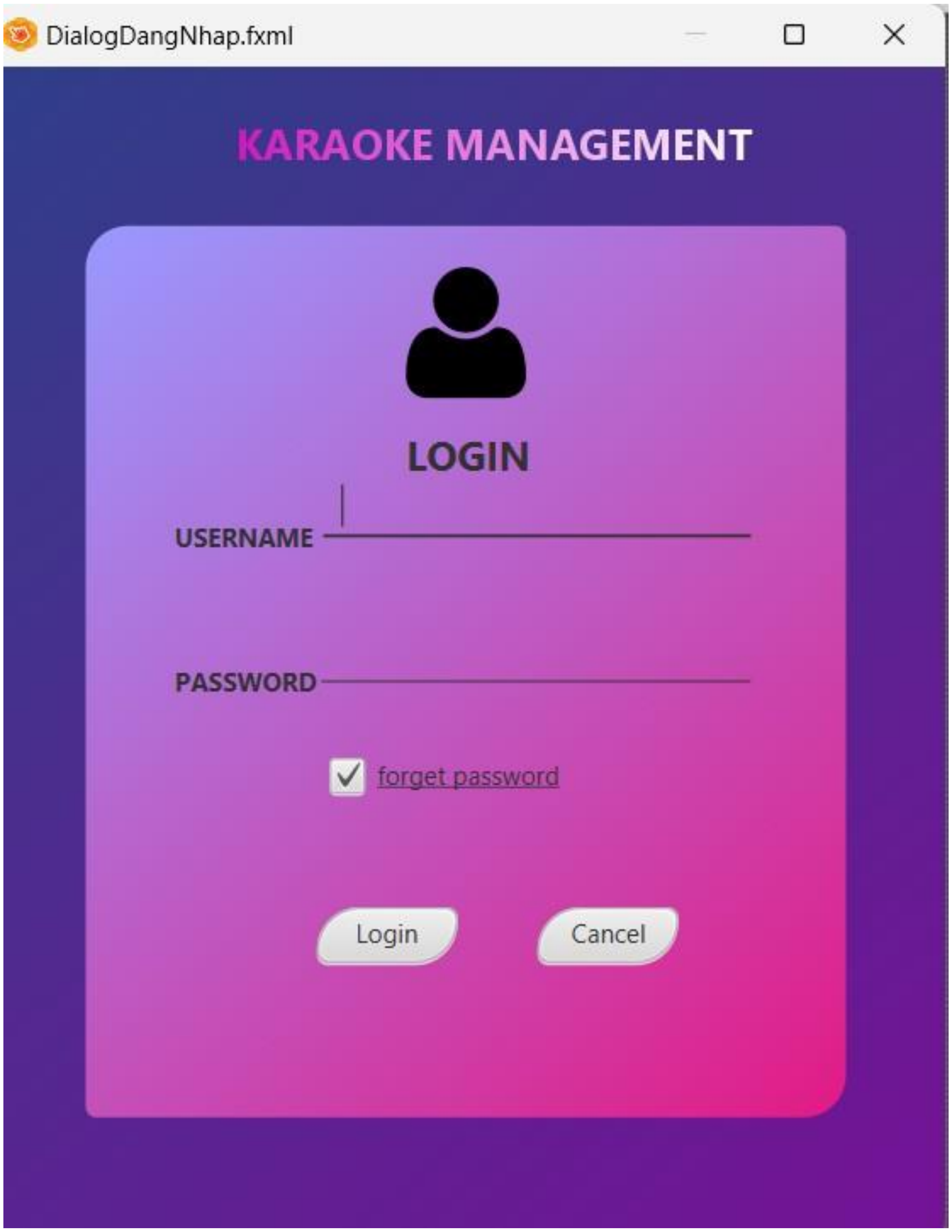


```
@FXML
void tblPhongDangSuDung(MouseEvent event) {
    int row = viewPhongDangSD.getSelectionModel().getSelectedIndex();
    if (row >= 0) {
        // Xóa dữ liệu hiện tại trong dichVuData (nếu cần thiết)
        dichVuData.clear();

        // Đổ dữ liệu đã chọn vào các label lblPhong, lblDonGiaPhong, lblKhachHang
        lblPhong.setText(value: phongSuDungData.get(index: row).getPhong().getTenPhong());
        lblDonGiaPhong.setText("Don gia phong: " + phongSuDungData.get(index: row).getPhong().getLoaiPhong().getDonGi
        lblKhachHang.setText(value: phongSuDungData.get(index: row).getHoaDon().getTenKhachHang());

        // Tải dữ liệu chi tiết cho hàng đã chọn vào dichVuData
        loadDataCTHD();
        // Phương thức này sẽ điền dichVuData với các đối tượng ChiTietHoaDon tương ứng

        // Tính toán và hiển thị tổng hóa đơn dựa trên hàng đã chọn
        hienThiTongTien(row);
        // Phương thức này sẽ tính tổng hóa đơn dựa trên dichVuData
    }
}
```



```
    private void search(String str) {
        filteredData.setPredicate(item -> {
            String selectedCriteria = cmbTimKiem.getSelectionModel().getSelectedItem();
            if (selectedCriteria == null || selectedCriteria.isEmpty()) {
                return true; // If no filter criteria is selected, show all items
            }

            String lowerCaseFilter = str.toLowerCase();
            switch (selectedCriteria) {
                case "Tim theo ma phong":
                    return item.getPhong().getMaPhong().toLowerCase().contains(s: lowerCaseFilter);
                case "Tim theo ten phong":
                    return item.getPhong().getTenPhong().toLowerCase().contains(s: lowerCaseFilter);
                case "Tim theo loai phong":
                    return item.getPhong().getLoaiPhong().getTenLoaiPhong().toLowerCase().contains(s: lowerCaseFilter);
                case "Tim theo don gia":
                    return String.valueOf(d: item.getPhong().getLoaiPhong().getDonGia()).toLowerCase().contains(s: lowerCaseFilt
                case "Tim theo thoi gian vao":
                    return item.getHoaDon().getNgayTao().toString().toLowerCase().contains(s: lowerCaseFilter);
                case "Tim theo khach hang":
                    return item.getHoaDon().getTenKhachHang().toLowerCase().contains(s: lowerCaseFilter);
                default:
                    return true;
            }
        });
    }

    private void handleKeyReleased(KeyEvent e) {
        String searchStr = txtTimPhong.getText();
        search(str: searchStr);
    }
}
```

8.Login

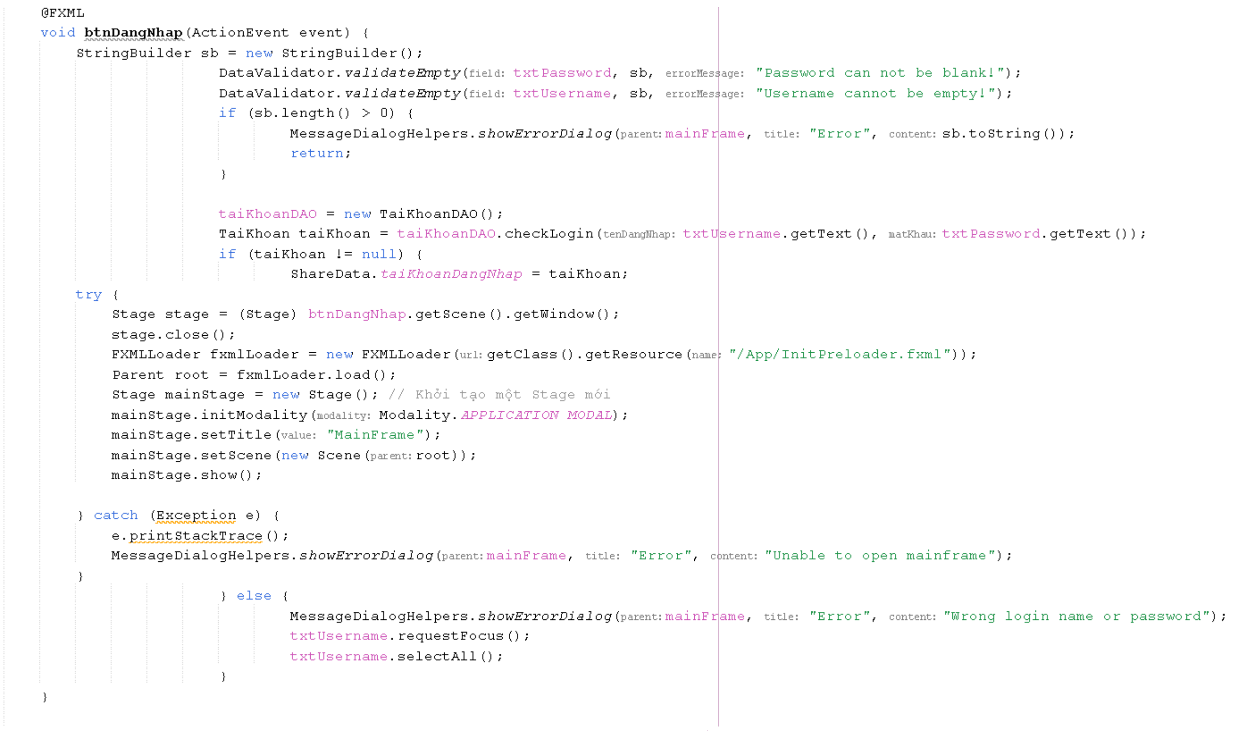


1: Fields used to enter employee account information

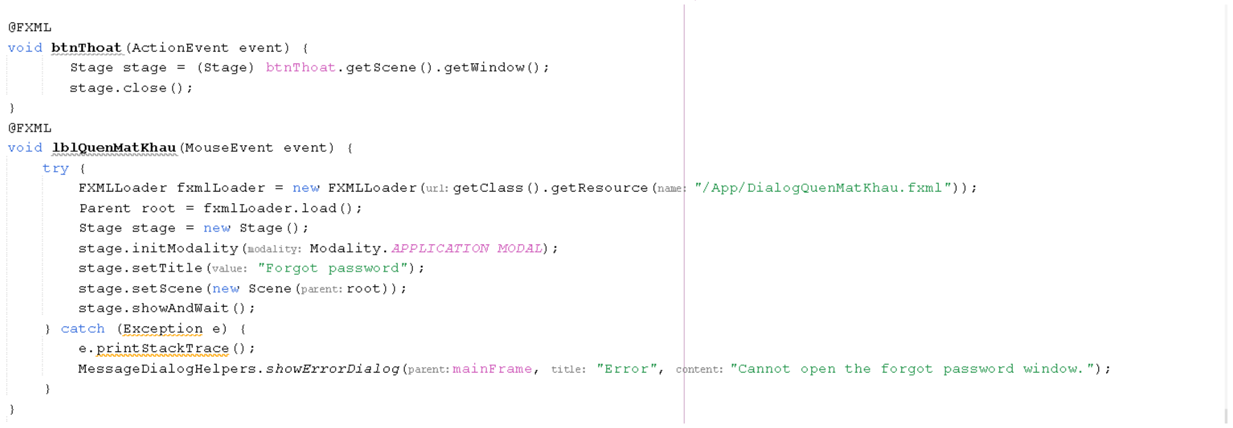
2: Click "Login" to be able to use the application

3: Tap “Cancel” if you want to exit the application

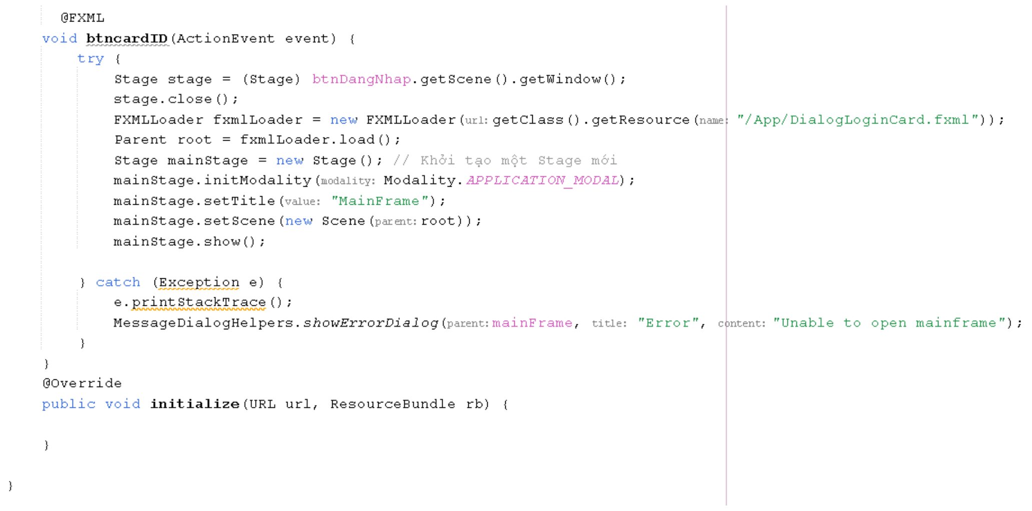
4: If you forget your password, tap “Forget password?”



```
@FXML
void btnDangNhap(ActionEvent event) {
    StringBuilder sb = new StringBuilder();
                DataValidator.validateEmpty( field: txtPassword, sb,  errorMessage: "Password can not be blank!");
                DataValidator.validateEmpty( field: txtUsername, sb,  errorMessage: "Username cannot be empty!");
                if (sb.length() > 0) {
                        MessageDialogHelpers.showErrorDialog( parent: mainFrame,  title: "Error",  content: sb.toString());
                        return;
                }

                taiKhoanDAO = new TaiKhoanDAO();
                TaiKhoan taiKhoan = taiKhoanDAO.checkLogin( tenDangNhap: txtUsername.getText(),  matKhau: txtPassword.getText());
                if (taiKhoan != null) {
                        ShareData.taiKhoanDangNhap = taiKhoan;
    try {
        Stage stage = (Stage) btnDangNhap.getScene().getWindow();
        stage.close();
        FXMLLoader fxmlLoader = new FXMLLoader( url: getClass().getResource( name: "/App/InitPreloader.fxml"));
        Parent root = fxmlLoader.load();
        Stage mainStage = new Stage(); // Khởi tạo một Stage mới
        mainStage.initModality( modality: Modality.APPLICATION_MODAL);
        mainStage.setTitle( value: "MainFrame");
        mainStage.setScene(new Scene( parent: root));
        mainStage.show();

    } catch (Exception e) {
        e.printStackTrace();
        MessageDialogHelpers.showErrorDialog( parent: mainFrame,  title: "Error",  content: "Unable to open mainframe");
    }
                } else {
                        MessageDialogHelpers.showErrorDialog( parent: mainFrame,  title: "Error",  content: "Wrong login name or password");
                        txtUsername.requestFocus();
                        txtUsername.selectAll();
                }
}
```

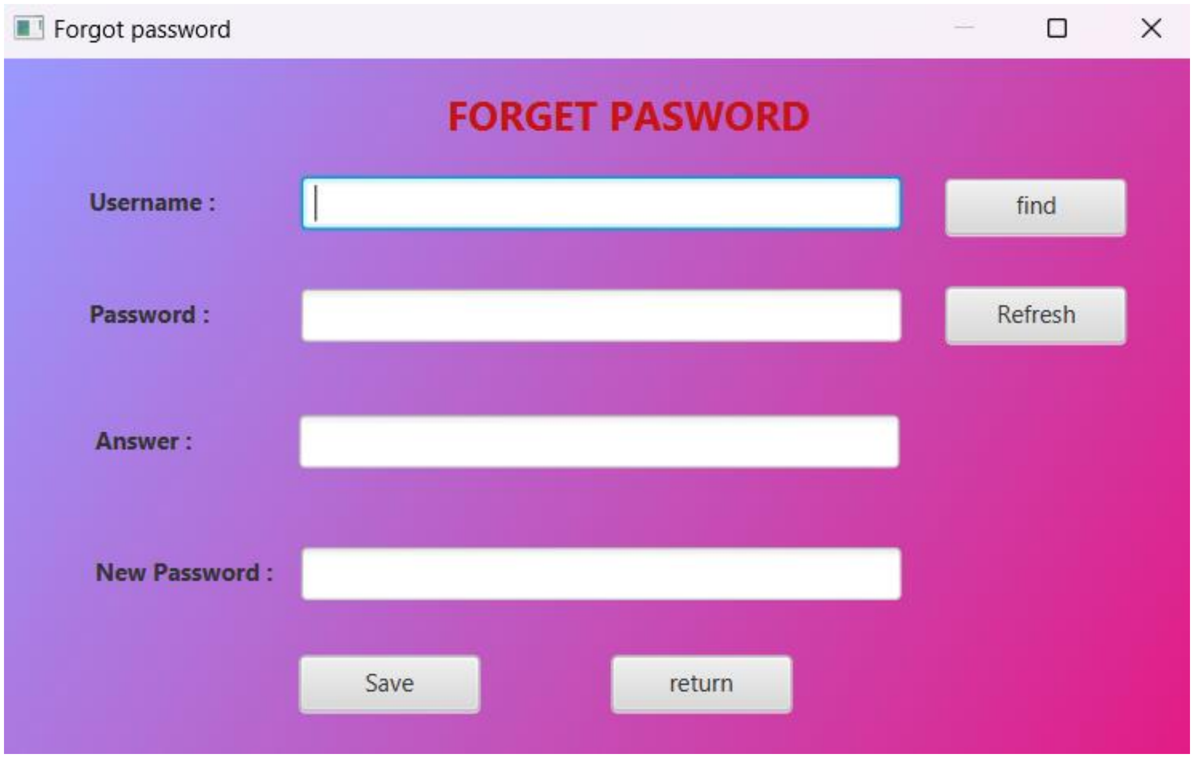


```
@FXML
void btnThoat(ActionEvent event) {
        Stage stage = (Stage) btnThoat.getScene().getWindow();
        stage.close();
}
@FXML
void lblQuenMatKhau(MouseEvent event) {
    try {
        FXMLLoader fxmlLoader = new FXMLLoader( url: getClass().getResource( name: "/App/DialogQuenMatKhau.fxml"));
        Parent root = fxmlLoader.load();
        Stage stage = new Stage();
        stage.initModality( modality: Modality.APPLICATION_MODAL);
        stage.setTitle( value: "Forgot password");
        stage.setScene(new Scene( parent: root));
        stage.showAndWait();
    } catch (Exception e) {
        e.printStackTrace();
        MessageDialogHelpers.showErrorDialog( parent: mainFrame,  title: "Error",  content: "Cannot open the forgot password window.");
    }
}
```



```
    @FXML
    void btncardID(ActionEvent event) {
        try {
            Stage stage = (Stage) btnDangNhap.getScene().getWindow();
            stage.close();
            FXMLLoader fxmlLoader = new FXMLLoader(url: getClass().getResource(name: "/App/DialogLoginCard.fxml"));
            Parent root = fxmlLoader.load();
            Stage mainStage = new Stage(); // Khởi tạo một Stage mới
            mainStage.initModality(modality: Modality.APPLICATION_MODAL);
            mainStage.setTitle(value: "MainFrame");
            mainStage.setScene(new Scene(parent: root));
            mainStage.show();

        } catch (Exception e) {
            e.printStackTrace();
            MessageDialogHelpers.showErrorDialog(parent: mainFrame, title: "Error", content: "Unable to open mainframe");
        }
    }
    @Override
    public void initialize(URL url, ResourceBundle rb) {

    }

}
```

9. FORGET PASSWORD



1: Enter your login name into the application

2: Click the “find” button to check your username

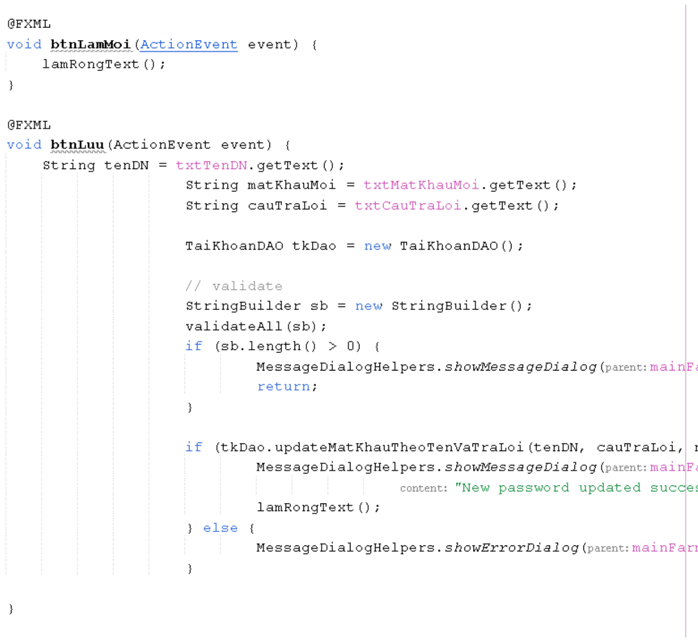
3: If the account is correct, the security question information that the manager has created for you will appear

4: Enter the answer to confirm the account

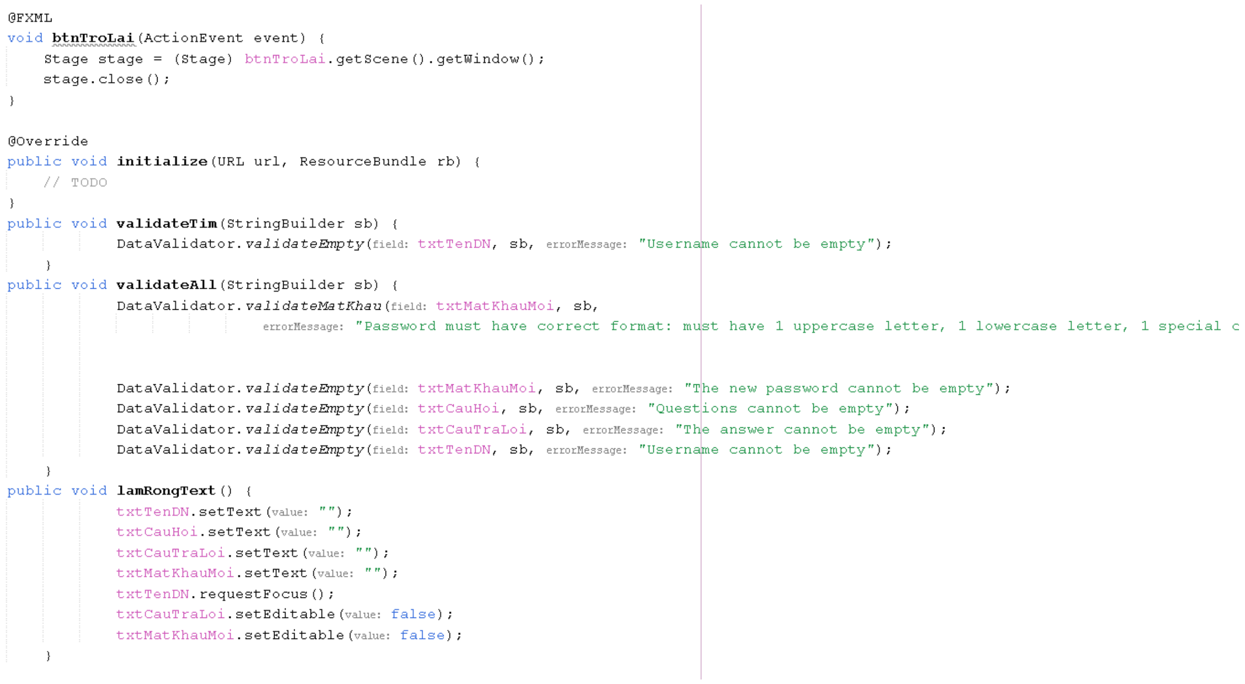
5: Enter a new password if your account is correct

6: press the “Save” button to save the new password

7: Press the “Refresh” button to refresh all data



```
@FXML
void btnLamMoi(ActionEvent event) {
    lamRongText();
}

@FXML
void btnLuu(ActionEvent event) {
    String tenDN = txtTenDN.getText();
                String matKhauMoi = txtMatKhauMoi.getText();
                String cauTraLoi = txtCauTraLoi.getText();

                TaiKhoanDAO tkDao = new TaiKhoanDAO();

                // validate
                StringBuilder sb = new StringBuilder();
                validateAll(sb);
                if (sb.length() > 0) {
                        MessageDialogHelpers.showMessageDialog(parent: mainFarme, title: "Remind", content: sb.toString());
                        return;
                }

                if (tkDao.updateMatKhauTheoTenVaTraLoi(tenDN, cauTraLoi, matKhauMoi)) {
                        MessageDialogHelpers.showMessageDialog(parent: mainFarme, title: "Notification",
                                        content: "New password updated successfully");
                        lamRongText();
                } else {
                        MessageDialogHelpers.showErrorDialog(parent: mainFarme, title: " Error ", content: "Update failed");
                }

}
```



```
@FXML
void btnTroLai(ActionEvent event) {
    Stage stage = (Stage) btnTroLai.getScene().getWindow();
    stage.close();
}

@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}
public void validateTim(StringBuilder sb) {
            DataValidator.validateEmpty(field: txtTenDN, sb, errorMessage: "Username cannot be empty");
    }
public void validateAll(StringBuilder sb) {
            DataValidator.validateMatKhau(field: txtMatKhauMoi, sb,
                                errorMessage: "Password must have correct format: must have 1 uppercase letter, 1 lowercase letter, 1 special c


            DataValidator.validateEmpty(field: txtMatKhauMoi, sb, errorMessage: "The new password cannot be empty");
            DataValidator.validateEmpty(field: txtCauHoi, sb, errorMessage: "Questions cannot be empty");
            DataValidator.validateEmpty(field: txtCauTraLoi, sb, errorMessage: "The answer cannot be empty");
            DataValidator.validateEmpty(field: txtTenDN, sb, errorMessage: "Username cannot be empty");
    }
public void lamRongText() {
            txtTenDN.setText(value: "");
            txtCauHoi.setText(value: "");
            txtCauTraLoi.setText(value: "");
            txtMatKhauMoi.setText(value: "");
            txtTenDN.requestFocus();
            txtCauTraLoi.setEditable(value: false);
            txtMatKhauMoi.setEditable(value: false);
    }
```

TASK SHEET REVIEW 3

Project: Karaoke management			**Date of preparation of activity plan:**			
#	**Task**	**Prepared by**	**Start date**	**Actual Days**	**Team member name**	**status**
1	GUI DESIGN	DANG TO NHAN NGUYEN HOANG ANH PHAN TRAN DANG CHI NGUYEN HOANG MINH NGOC PHAN VAN DUY	07/06/2024	17/06/2024	DANG TO NHAN NGUYEN HOANG ANH PHAN TRAN DANG CHI NGUYEN HOANG MINH NGOC PHAN VAN DUY	completed
Review			Signature of instructor			
			Mr.Pham Cong Danh			