**PRO192 ASSIGNMENT SUBMISSION**

Student code: Tran Phuc Tien  
Student name: DE180866

**I. Project Introduction**  
**1. Project Title:** Motorbike Rental  
**2. Project description:** The Motorbike Rental project aims to create a software system for managing a motorbike rental service. This system will facilitate the rental process, allowing customers to browse available motorbikes, rent them, and return them after use. Additionally, it will keep track of customer and rental information for administrative purposes.

**3. Use-case specification:**

***1. Add New Customer***

+ Description:

* This use-case allows the system to add a new customer to the database.

+ Input:

* ID number (String)
* Name (String)

+ Process:

* Check if the entered ID already exists in the system.
* If ID does not exist, prompt for the customer's name.
* Create a new Customer object with the provided information and add it to the customer list.

+ Output:

* Return the newly created Customer object if successful, otherwise return null.

***2. Add New Motorbike***

+ Description:

* This use-case allows the system to add a new motorbike to the inventory.

+ Input:

* License Plate (String)
* Brand Name (String)
* Rent Price (Double)
* Motorbike Price (Double)

+ Process:

* Generate a unique MotorbikeID.
* Create a new Motorbike object with the provided information and add it to the motorbike list.

+ Output: None

***3. Rent Motorbike***

+ Description:

* This use-case allows a customer to rent a motorbike.

+ Input:

* Choice (Integer)

+ Process:

* Display available motorbikes.
* Prompt the user to choose a motorbike.
* If a valid choice is made, prompt for customer information.
* Create a new RentLog with the customer, chosen motorbike, and descriptor.
* Update motorbike status and add rent log to the list.

+ Output:

* Display rent bill if successful, otherwise display an error message.

***4. Return Motorbike***

+ Description:

* This use-case allows a customer to return a rented motorbike.

+ Input:

* Choice (Integer)

+ Process:

* Display unavailable motorbikes.
* Prompt the user to choose a rented motorbike to return.
* If a valid choice is made, update the motorbike status, set the return time, and display the bill.

+ Output:

* Display return bill if successful, otherwise display an error message.

***5. Display Rent Logs***

+ Description:

* This use-case allows the system to display all rent logs.

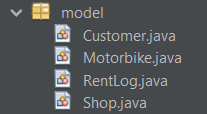
+ Input: None

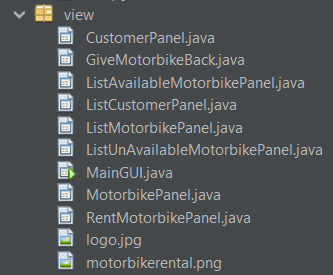
+ Process:

* Iterate through the list of rent logs and display the information for each.

+ Output: Display all rent logs.

**II. Classes design**   
**Model:** Customer, Motorbike, RentLog, Shop.

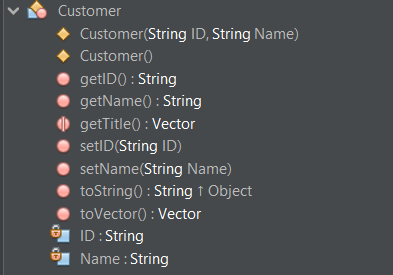
  
**View:** CustomerPanel, RentMotorbikePanel, GiveMotorbikeBack, ListMotorbikePanel, ListAvailableMotorbikePanel, ListCustomerPanel, ListUnAvailableMotorbikePanel, MotorbikePanel, MainGUI.

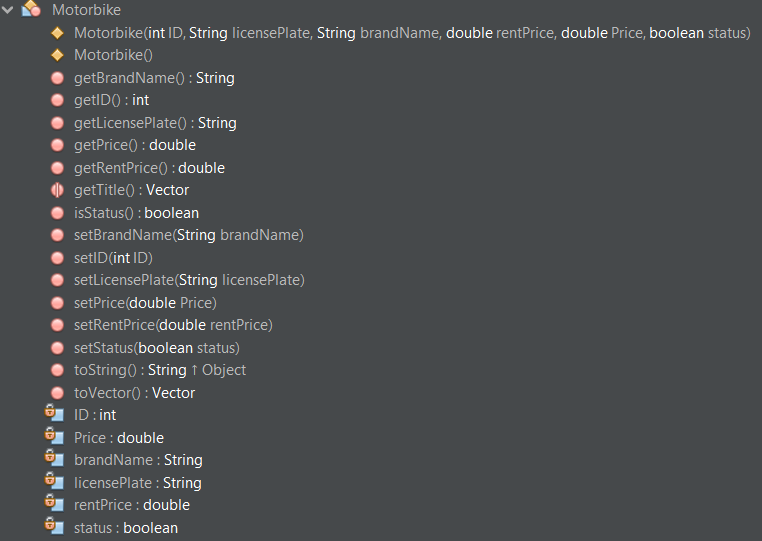
  
**Controller:** ShopApp, Menu.

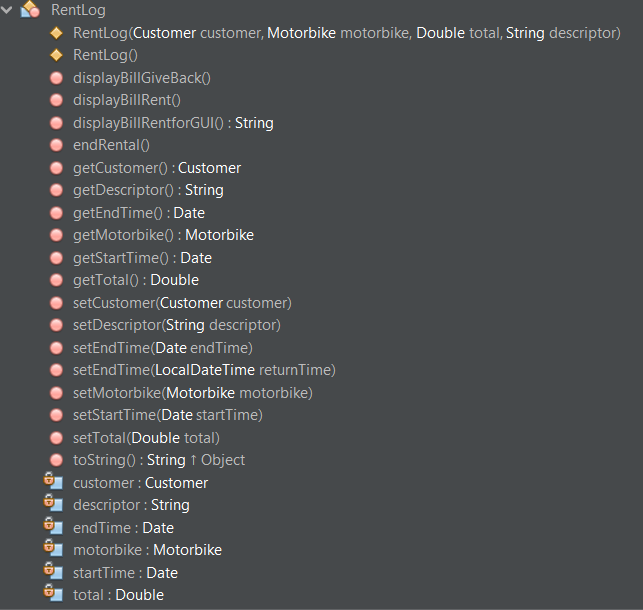


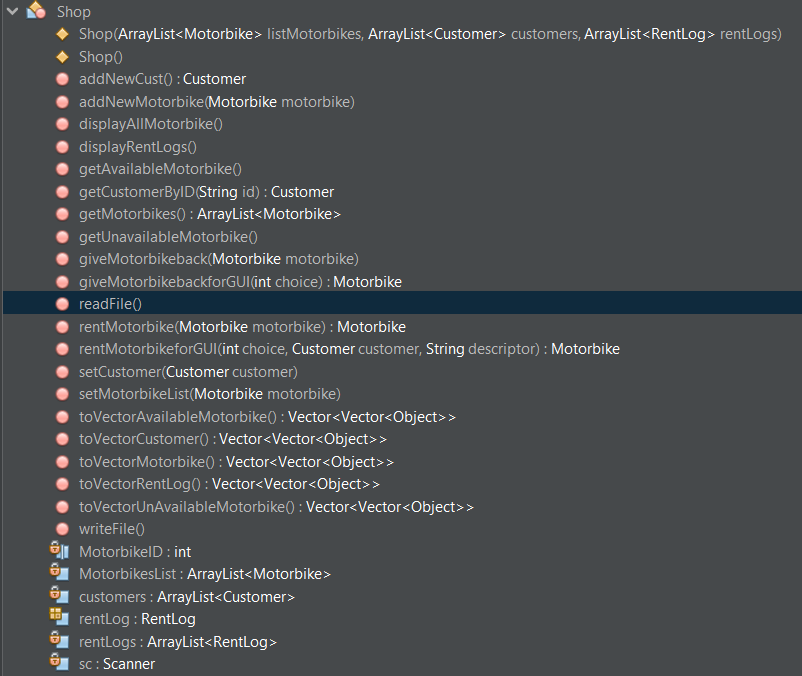
**Class diagram:**

*Model:*

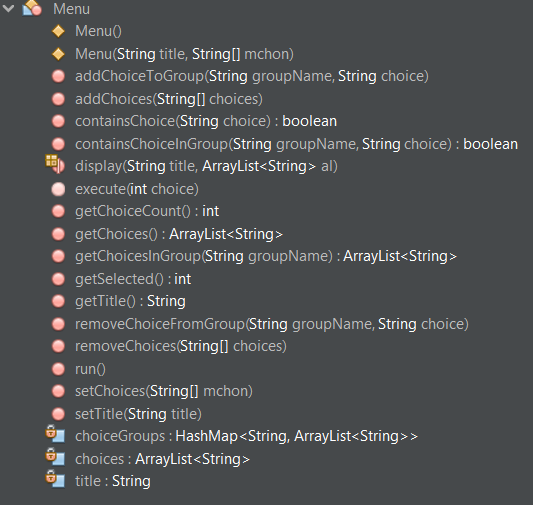


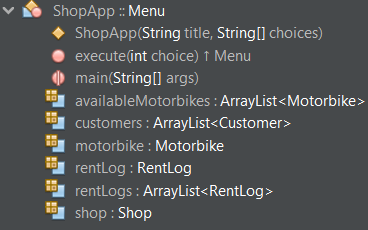




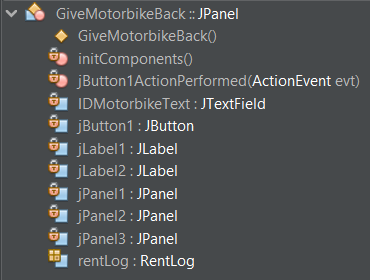
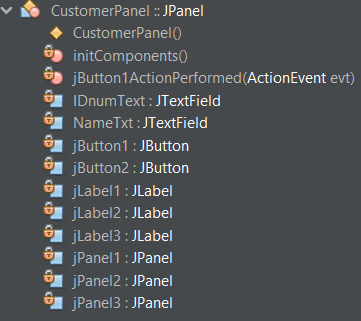


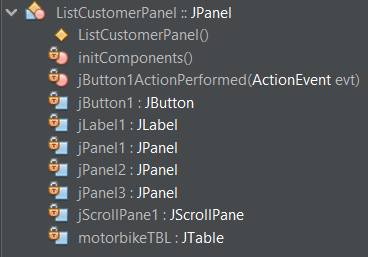
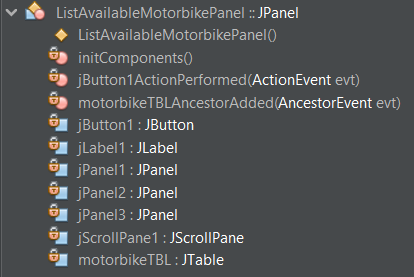
*Controller:*

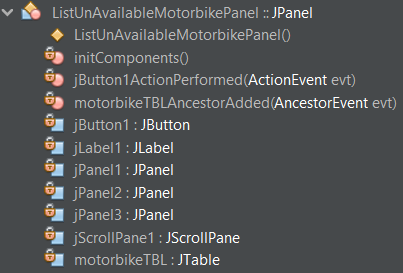
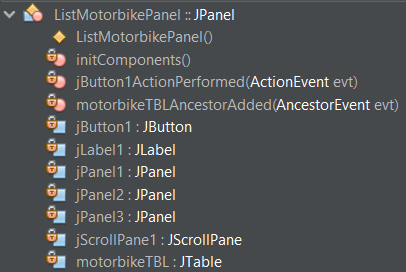
**

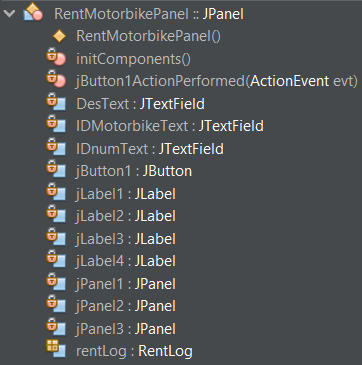
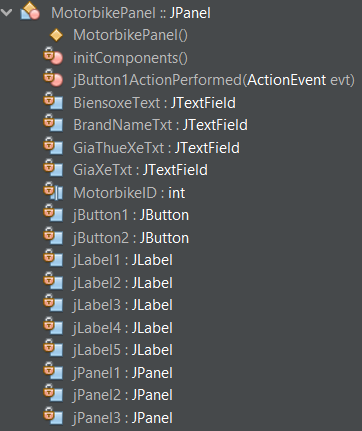
**

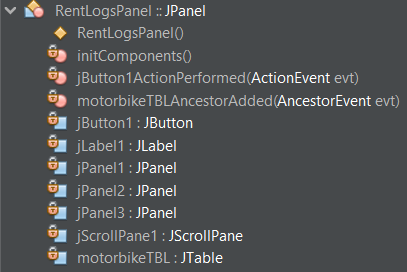
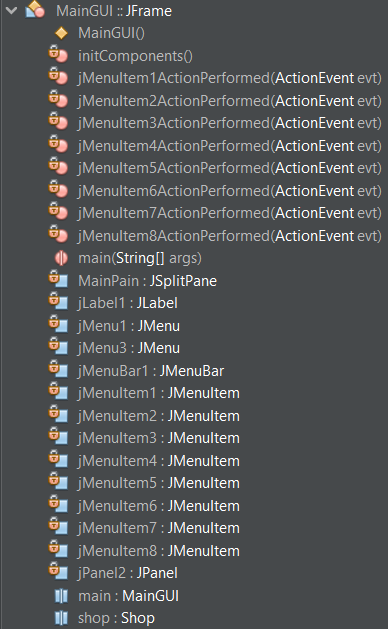
*View:*





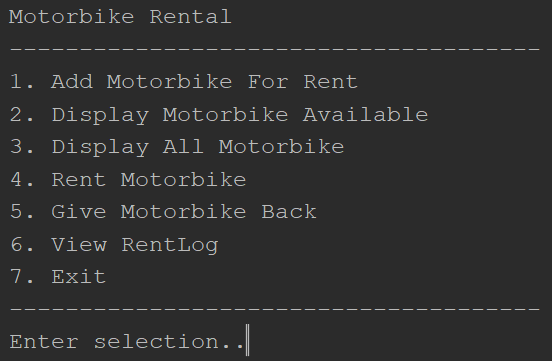




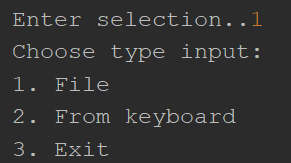


**III. Implementation**

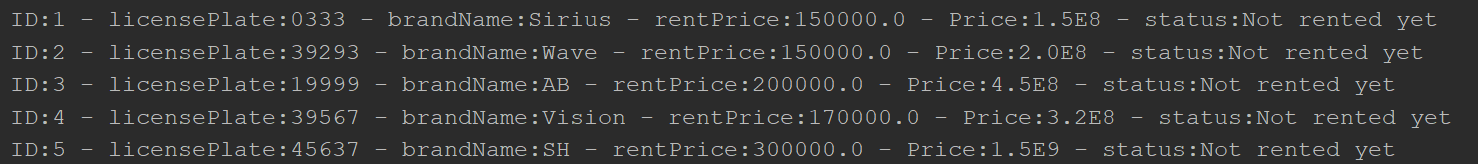
1. **For console log:**

****

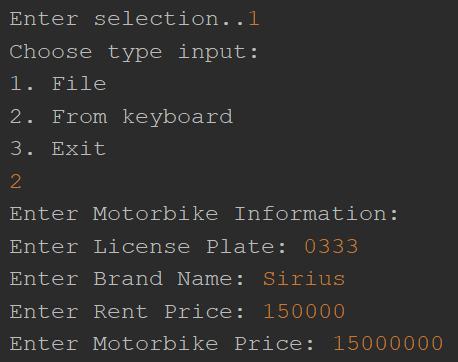
***1. Add New Motorbike***

******

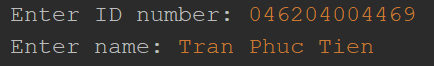
1.



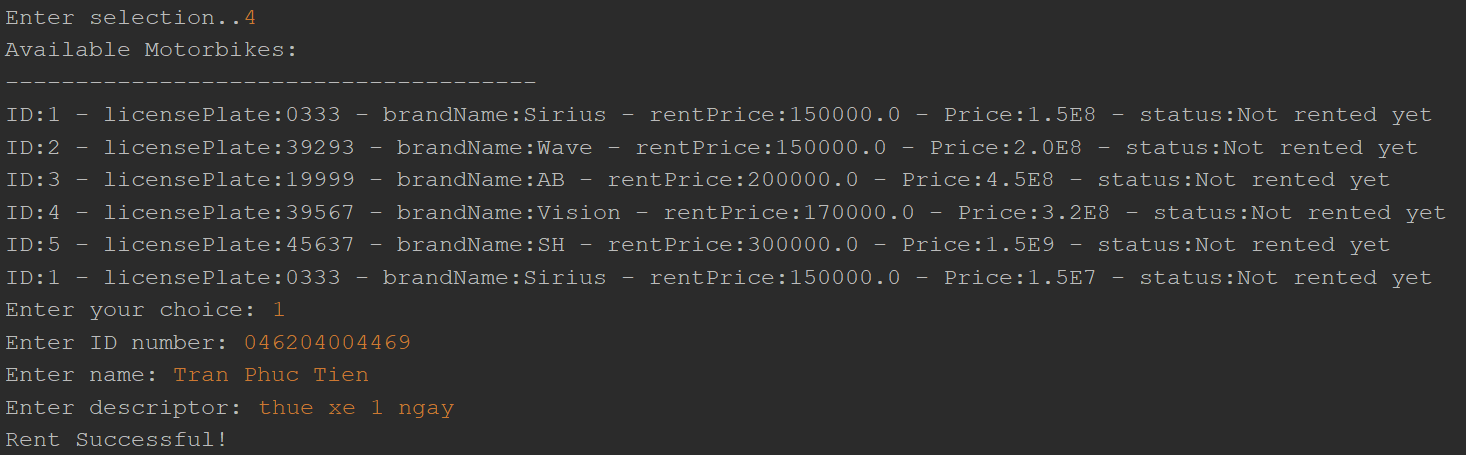
2.

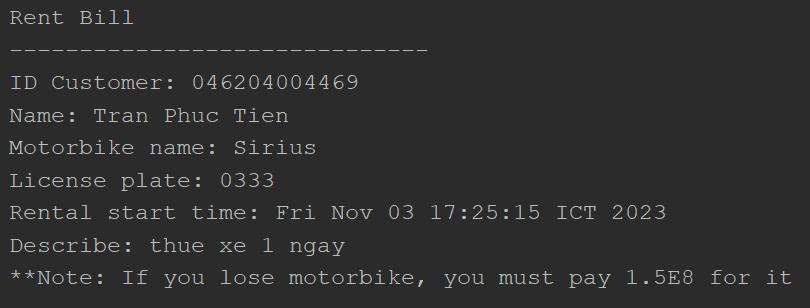


***2. Add New Customer***

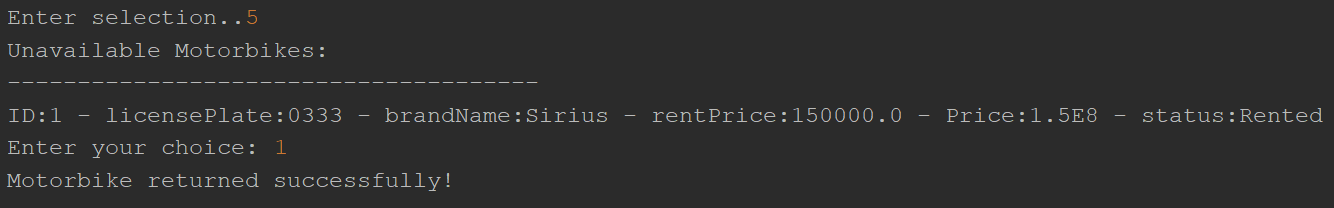
******

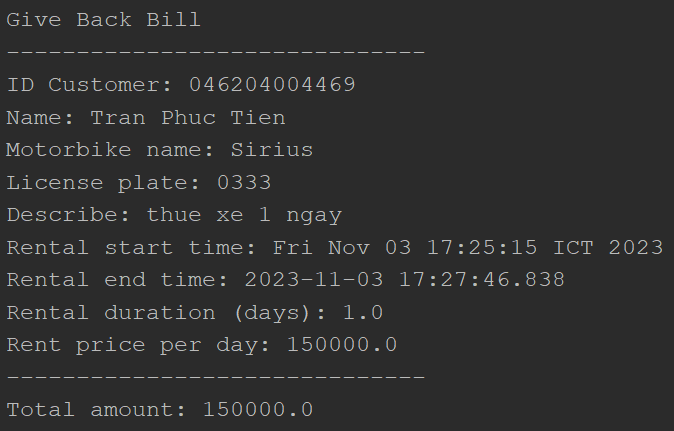
***3. Rent Motorbike***

******

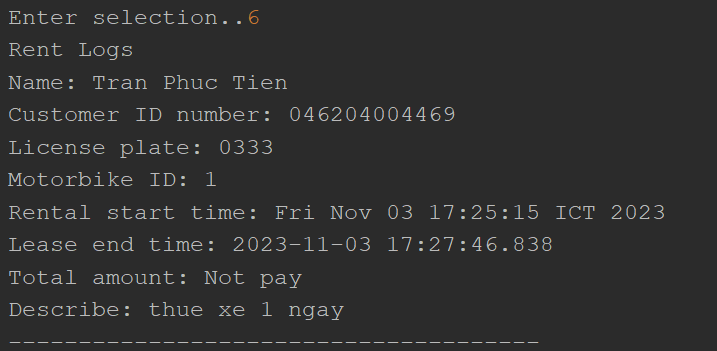
**

***4. Return Motorbike***

******

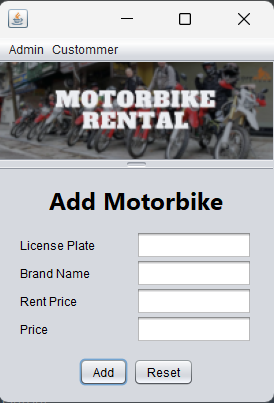
****

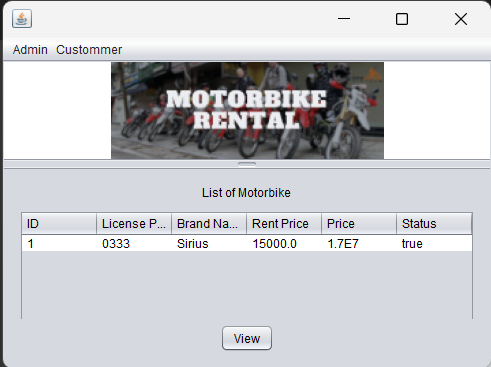
***5. Display Rent Logs***

******

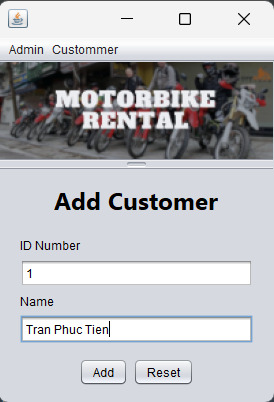
1. **For GUI:**

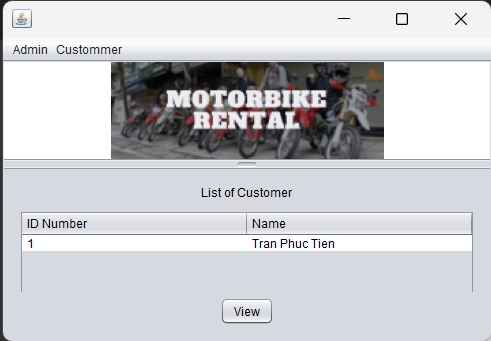
***1. Add New Motorbike***

******

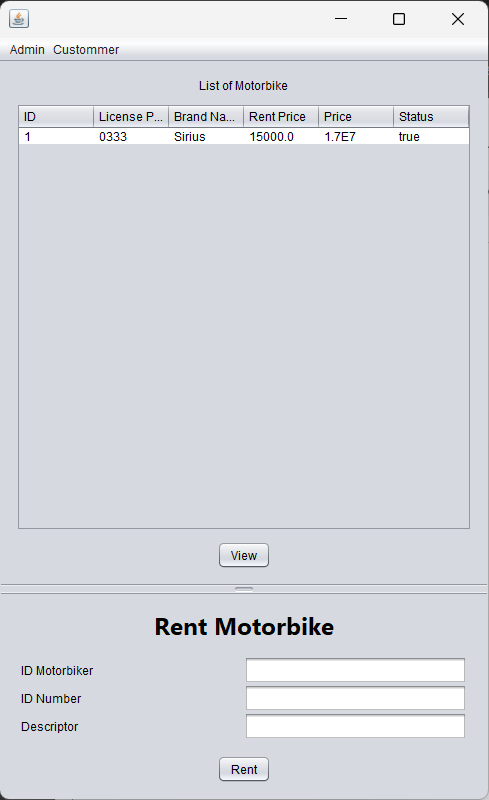
******

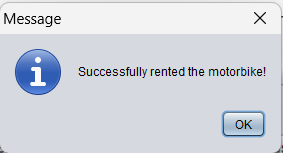
***2. Add New Customer***

******

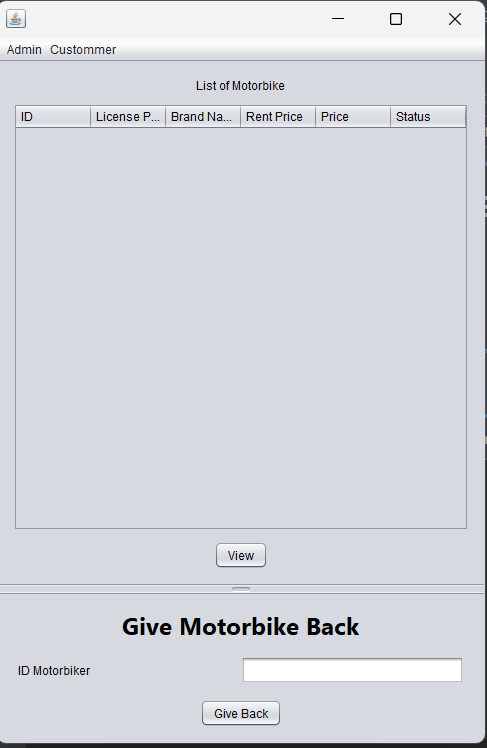
******

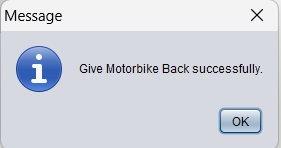
***3. Rent Motorbike***

******

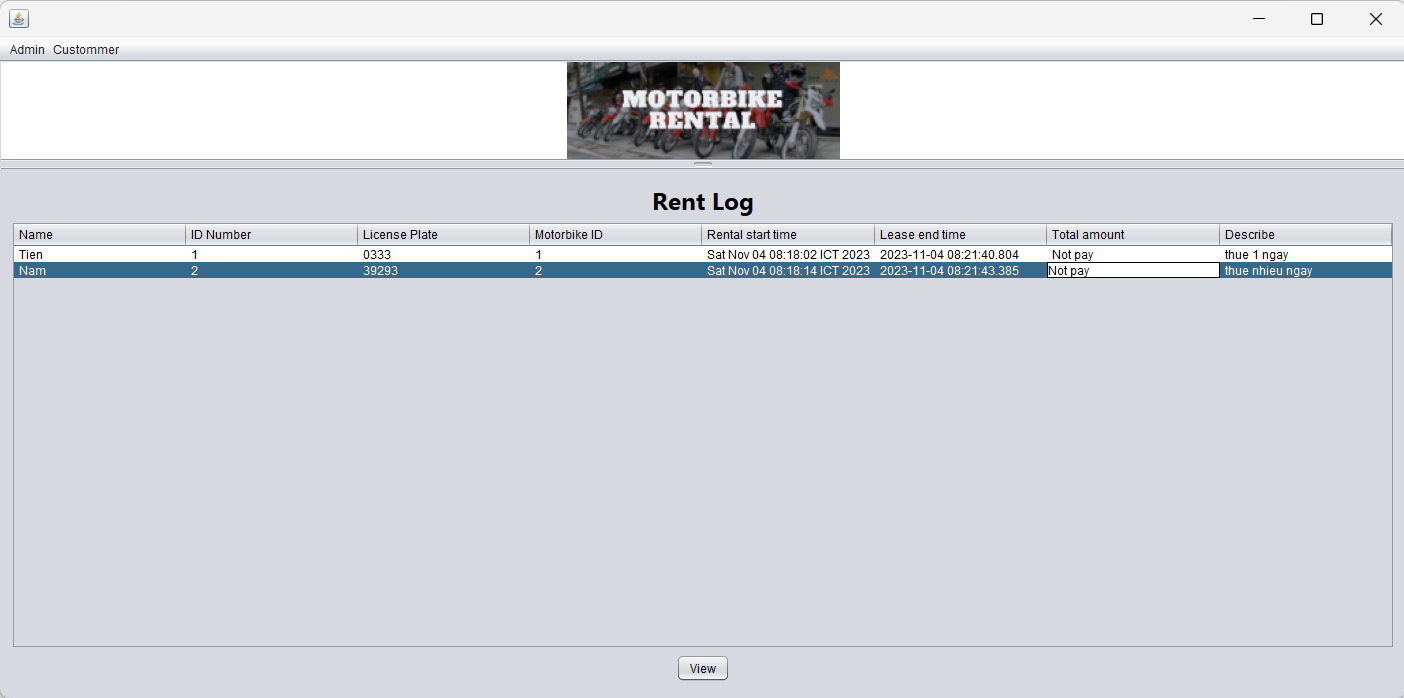
******

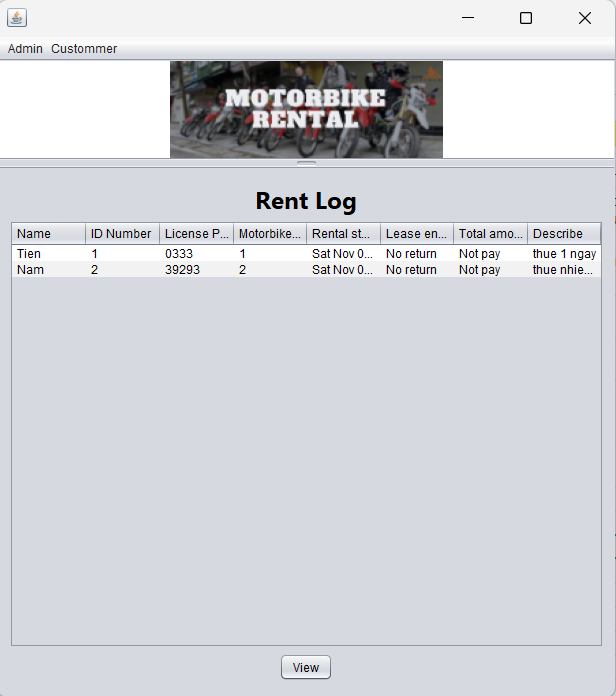
***4. Return Motorbike***

******

******

***5. Display RentLog:***

******

******

**IV. Conclusion:**  
1. Develop an application with a console and/or GUI interface: Completed.

2. Apply OOP features in class design, especially using interfaces, abstract classes, and subclasses: Completed.

3. Load/Save data from files on disk: Completed.

4. Follow the MVC (Model-View-Controller) architecture: Completed.

5. Documentation should reflect the process: business requirement specification, analysis, class design: Completed.