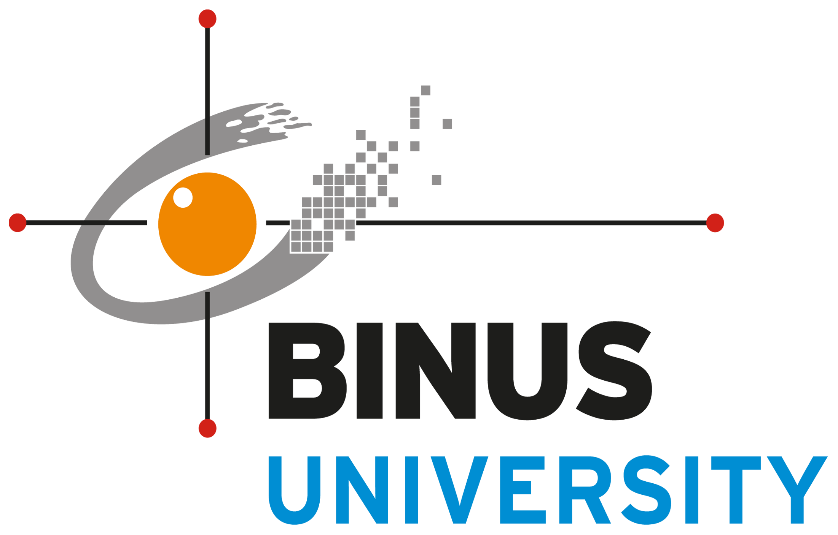
**FINAL PROJECT REPORT**

**CAR RENTAL MANAGEMENT SYSTEM**



**LECTURER**

**Jude Joseph Lamug Martinez MCS**

**ARRANGE BY**

**Haniif Satria Wardana**

**2702358065**

**OBJECT ORIENTED PROGRAMMING**

**COMPUTER SCIENCE**

**Faculty Binus International University**

# Table Of Contents

[Table Of Contents ii](#_Toc168609606)

[Abstract 1](#_Toc168609607)

[Project Background 1](#_Toc168609608)

[Objective 1](#_Toc168609609)

[Chapter 1 - Project Specification 2](#_Toc168609610)

[Chapter 2 - Solution Design 2](#_Toc168609611)

[2.1 Class Diagram 2](#_Toc168609612)

[Chapter 3 -Implementation Overview of Car Rental System Project 3](#_Toc168609613)

[3.1 Classes and Their Roles 3](#_Toc168609614)

[3.2 Algorithms and Solution Scheme 4](#_Toc168609615)

[3.3 Data Structures Used 4](#_Toc168609621)

[Chapter 4 - Explanation of How It Works 4](#_Toc168609622)

[4.1 Initialization 4](#_Toc168609623)

[4.2 Adding Vehicles 5](#_Toc168609624)

[4.3 Renting Vehicles 5](#_Toc168609625)

[4.4 Returning Vehicles 5](#_Toc168609626)

[4.5 Managing Customer Data 5](#_Toc168609627)

[4.6 Maintaining Rental History 6](#_Toc168609628)

[4.7 User Interface and Interaction 6](#_Toc168609629)

[Chaper 5 - Evidence of Working Program 7](#_Toc168609630)

[5.1 Screenshots of Working Program: 7](#_Toc168609631)

[Chapter 6 – Source Code & Poster 11](#_Toc168609632)

[6.1 Github Link 11](#_Toc168609633)

[6.2 Poster 11](#_Toc168609634)

[Conclusion 12](#_Toc168609635)

[References 13](#_Toc168609636)

# Abstract

This report details the design, implementation, and functionality of a Car Rental System. The system manages vehicles, customers, and rentals, providing a comprehensive solution for car rental services. The system allows adding, renting, and returning vehicles, as well as managing customer information and applying loyalty discounts.

# Project Background

The Car Rental System project aims to simplify and automate the process of renting vehicles. It provides a user-friendly interface for managing vehicle availability, customer data, and rental transactions. This system is designed to handle different types of vehicles, including regular cars, luxury cars, electric cars, and SUVs.

# Objective

The primary objective of this project is to develop a robust and efficient car rental system that meets the needs of both rental companies and customers. The system should facilitate vehicle management, rental processes, and customer management while ensuring ease of use and reliability.

# Chapter 1 - Project Specification

* The Car Rental System should include the following features:
* Add new vehicles to the system.
* Rent vehicles to customers.
* Return vehicles and update their availability.
* Manage customer information.
* Calculate rental prices, including loyalty discounts.
* Handle different types of vehicles, including luxury, electric, and SUVs.

# Chapter 2 - Solution Design

## **2.1 Class Diagram**

**Sebuah gambar berisi teks, diagram, cuplikan layar, garis

Deskripsi dibuat secara otomatis**

# Chapter 3 -Implementation Overview of Car Rental System Project

The implementation of the Car Rental System project involves several classes, each with specific roles and responsibilities. The design focuses on modularity and clear separation of concerns, ensuring that each class has a distinct purpose and interacts with others in a cohesive manner. Below is an overview of the key classes used in the project, their roles, and how they contribute to the overall functionality of the system.

## 3.1 **Classes and Their Roles**

|  |  |
| --- | --- |
| **Class** | **Role** |
| Car | Represents a regular car available for rental. Implements the Vehicle interface. |
| LuxuryCar | Extends the Car class and adds luxury tax features. |
| ElectricCar | Extends the Car class, representing electric vehicles. |
| SUV | Extends the Car class, representing SUVs. |
| Vehicle | An interface defining the common methods all vehicle types must implement. |
| Rental | Represents a rental transaction, including details about the vehicle, customer, and rental period. |
| Customer | Represents a customer, including personal details and rental history. |
| CustomerManager | Manages customer information, including adding, updating, and retrieving customer data. |
| CarRentalSystem | The main class managing the overall car rental operations, including vehicle and rental management. |
| Main | Contains the main method to run the application, initializing the system and starting the menu. |

## 3.2 **Algorithms and Solution Scheme**

Adding Vehicles: Algorithm to add new vehicles to the system, ensuring no duplicates.

Renting Vehicles: Algorithm to rent vehicles, checking for availability and updating the rental status.

Returning Vehicles: Algorithm to return rented vehicles and update their availability.

Customer Management: Algorithm to add, update, and retrieve customer information, ensuring data integrity.

Rental History: Algorithm to maintain rental history for each customer, applying loyalty discounts where applicable.

## 3.3 **Data Structures Used**

* **Maps:** Used for storing customer and vehicle information for quick lookups.
* **Lists:** Used for maintaining a list of all vehicles and rentals.
* **ArrayLists:** A specific implementation of lists used for dynamic arrays that grow as needed.
* **HashMaps:** Used for associating keys with values, such as customer IDs with customer objects or vehicle IDs with vehicle objects.

# Chapter 4 - Explanation of How It Works

The Car Rental System is designed to manage vehicle rentals efficiently through a user-friendly command-line interface. Below is a detailed explanation of how the system operates, focusing on the primary functionalities such as adding vehicles, renting and returning vehicles, and managing customer data.

## 4.1 Initialization

When the application starts, it initializes the main components of the system:

* Vehicle Inventory: A collection of all available vehicles, stored in an ArrayList.
* Customer Database: A collection of all registered customers, stored in a HashMap for quick lookup by customer ID.
* Rental Records: A list of all rental transactions, stored in an ArrayList.

The CarRentalSystem class orchestrates these components, ensuring they interact seamlessly.

## 4.2 Adding Vehicles

Users can add new vehicles to the inventory through a menu option. The process is as follows:

* The system prompts the user for vehicle details such as type, make, model, and registration number.
* A new vehicle object (e.g., Car, LuxuryCar, ElectricCar, or SUV) is created based on the input.
* The vehicle is added to the inventory list.
* The system checks for duplicates by comparing registration numbers to ensure each vehicle is unique.

## 4.3 Renting Vehicles

Renting a vehicle involves several steps to ensure availability and correct data handling:

* The user selects the rental option from the menu.
* The system displays a list of available vehicles.
* The user chooses a vehicle to rent and provides their customer ID.
* The system checks if the vehicle is available and if the customer exists in the database.
* If both conditions are met, a new Rental object is created, recording the customer ID, vehicle details, and rental period.
* The vehicle's status is updated to indicate it is rented.
* The rental record is added to the list of rentals.

## 4.4 Returning Vehicles

When a vehicle is returned:

* The user selects the return option from the menu.
* The system prompts for the rental ID or vehicle registration number.
* The system finds the corresponding rental record and updates the vehicle's status to available.
* The rental record is updated to reflect the return date and time.
* If the customer qualifies for a loyalty discount based on their rental history, it is applied to their account for future rentals.

## 4.5 Managing Customer Data

Customer data management includes adding, updating, and retrieving customer information:

* Adding Customers: The system prompts for customer details such as name, address, and ID. A new Customer object is created and added to the HashMap using the customer ID as the key.
* Updating Customers: The system retrieves the customer record by ID, updates the necessary fields, and saves the changes.
* Retrieving Customers: The system can quickly look up customer details using the ID from the HashMap.

## 4.6 Maintaining Rental History

The system maintains a detailed rental history for each customer:

* Each Customer object has an associated list of rental transactions.
* When a rental is created, it is added to the customer's rental history.
* The system calculates the number of rentals and applies loyalty discounts where applicable.

## 4.7 User Interface and Interaction

The command-line interface guides the user through the various functionalities:

* A main menu displays options for adding vehicles, renting vehicles, returning vehicles, and managing customers.
* Users interact with the system by selecting menu options and providing necessary inputs.
* The system provides feedback and confirmations for each operation, ensuring a smooth user experience

# Chaper 5 - Evidence of Working Program

## 5.1 **Screenshots of Working Program**:

1. Main menu

Sebuah gambar berisi teks, Font, cuplikan layar

Deskripsi dibuat secara otomatis

1. Vehicle rental process

Sebuah gambar berisi teks, cuplikan layar

Deskripsi dibuat secara otomatis

1. Vehicle return process

Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis

1. Adding a new vehicle

Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis

1. Managing customers

Sebuah gambar berisi teks, cuplikan layar, deasin

Deskripsi dibuat secara otomatis

1. Tax Features

Sebuah gambar berisi teks, cuplikan layar, deasin

Deskripsi dibuat secara otomatis

Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis

1. Loyalty customers

Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis

This loyalty is obtained when a customer has rented a car 3 times, if he has rented a car 3 times he will get a 20% discount and if he has rented a car 3 times, the customer will get 30%.

# Chapter 6 – Source Code & Poster

6.1 Github Link **:** [**https://github.com/Mbulss/Final\_Project\_OOP.git**](https://github.com/Mbulss/Final_Project_OOP.git)

6.2 Poster **: Sebuah gambar berisi teks, elektronik, komputer, cuplikan layar

Deskripsi dibuat secara otomatis**

# Conclusion

The Car Rental System represents a comprehensive solution for managing vehicle rentals, designed with a focus on efficiency, scalability, and user experience. The system integrates multiple components seamlessly, enabling users to perform key operations such as adding vehicles, managing rentals, and handling customer data with ease.

Key aspects of the system include:

* Modular Design: By leveraging Java's object-oriented programming capabilities, the system is broken down into distinct classes and methods, each responsible for specific functionalities. This modular approach ensures ease of maintenance and extensibility.
* Efficient Data Management: Using data structures like ArrayList and HashMap, the system ensures quick access and manipulation of data. This is crucial for maintaining real-time updates and providing a responsive user interface.
* User-Friendly Interface: The command-line interface is designed to guide users through various operations intuitively, making the system accessible even to those with limited technical knowledge.
* Robust Rental Process: The system ensures that the rental process is seamless and error-free by checking vehicle availability, validating customer information, and maintaining detailed rental records.
* Customer Management: By storing customer data in a HashMap, the system allows for quick lookups and updates, enhancing the overall efficiency and user experience.
* Loyalty Features: The inclusion of a loyalty discount system rewards frequent customers, promoting customer retention and satisfaction.

Overall, the Car Rental System demonstrates how careful planning and the use of appropriate data structures and algorithms can result in a robust application that meets the needs of both the users and the administrators. It serves as a practical example of applying object-oriented programming principles to solve real-world problems effectively. As a future improvement, the system could be enhanced with a graphical user interface (GUI) and integrated with a database for even better performance and scalability.

# References

Indian Programmer. (2023, August 28). *Java OOPs Project - Java Car 🚗 Rental System Project 🔥🔥 | OOP Concepts & Implementation* [Video]. YouTube. <https://www.youtube.com/watch?v=MHJnHv88WSI>

**NOTE :** This reference I only took the idea from him