**Car Rental and User Login Assignment Documentation**

**BSD 214 Object-Oriented Programming 2 - Assignment 1**

**Joseph Muroki Njoroge**

**BSE-01-0237/2025**

**GITHUB REPOSITORY:** [**https://github.com/JosephRocky/assignment**](https://github.com/JosephRocky/assignment)

Table of Contents

[Summary 3](#_Toc210345398)

[Outcomes: 3](#_Toc210345399)

[1.User Login system 3](#_Toc210345400)

[Login system 4](#_Toc210345401)

[Objectives 4](#_Toc210345402)

[Output 5](#_Toc210345403)

[Success in Login 5](#_Toc210345404)

[Fail in Login 5](#_Toc210345405)

[2: Car Rental Management System 6](#_Toc210345406)

[SYSTEM CODE: 6](#_Toc210345407)

[2. System Features 17](#_Toc210345408)

[3. Sample Data (Initial Setup) 18](#_Toc210345409)

[4. Example Workflow 18](#_Toc210345410)

[5. Technology Used 18](#_Toc210345411)

[6. Error Handling 18](#_Toc210345412)

[SYSTEM INTERFACE: 19](#_Toc210345413)

[Car Management Outputs 19](#_Toc210345414)

[System Descriptions 24](#_Toc210345415)

[1. User Login System 24](#_Toc210345416)

[2. Car Rental Management System 25](#_Toc210345417)

Summary

This assignment implements Java applications demonstrating O.O.P principles:

1: Secure user Login System with password masking and limited login attempts.

2: Car Rental Management System supporting car and customer management, rental transactions, and viewing rented cars.

Both programs showcase OOP features such as encapsulation, abstraction, and modular design while addressing real-world problems in security and resource management.

### Outcomes:

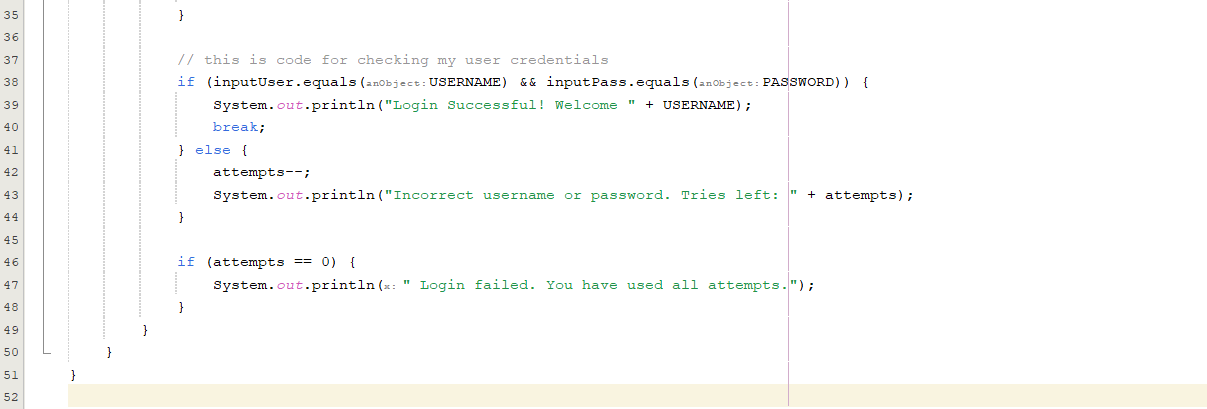
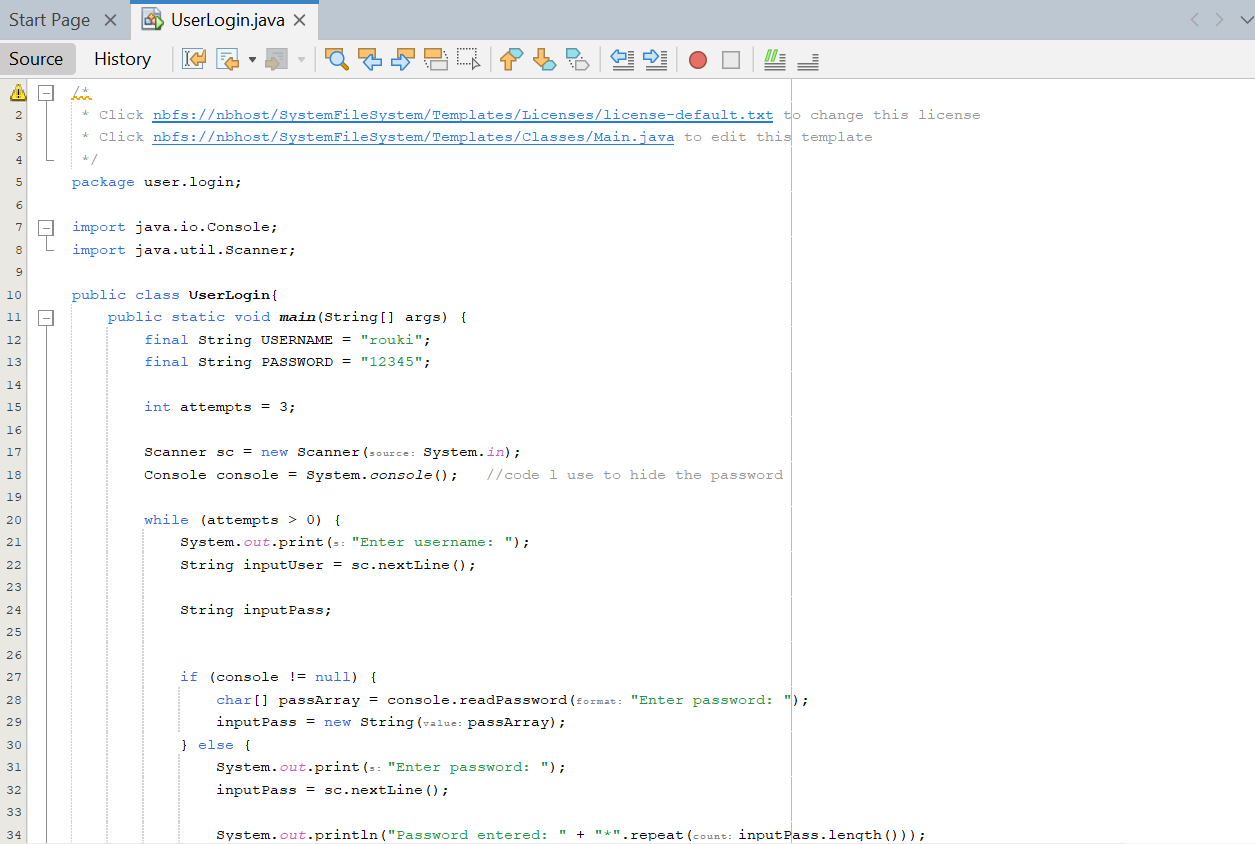
In this assignment l was able to:

* Apply encapsulation and abstraction to design modular, maintainable Java applications.
* Implement user authentication mechanisms with password masking and access control.
* Manage real-world entities (cars, customers, rentals) using OOP classes and relationships.

## 1.User Login system

Provides secure access . It ensures that only authorized users can log in using a **username and password**. For privacy, the password is masked with \* characters while typing.

### Login system

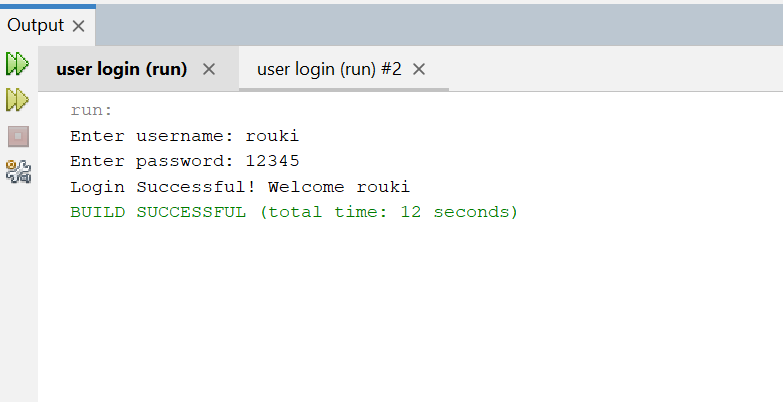


### Objectives

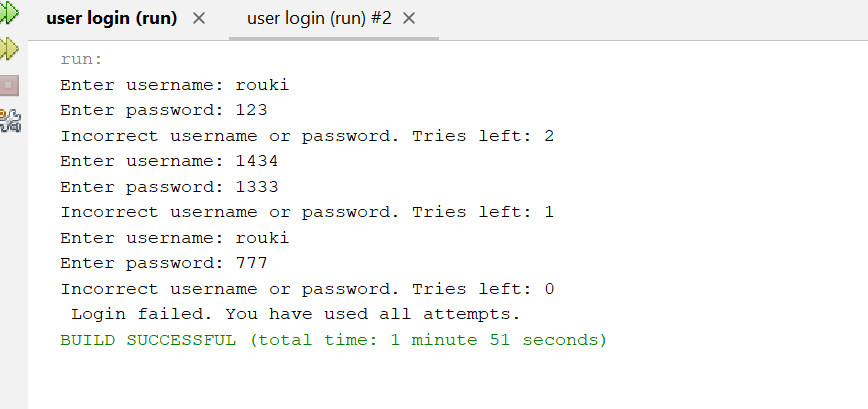
* Accepts a username and password.
* Displays each password character as \*.
* Gives the user **3 attempts**.

### Output

### Success in Login

****

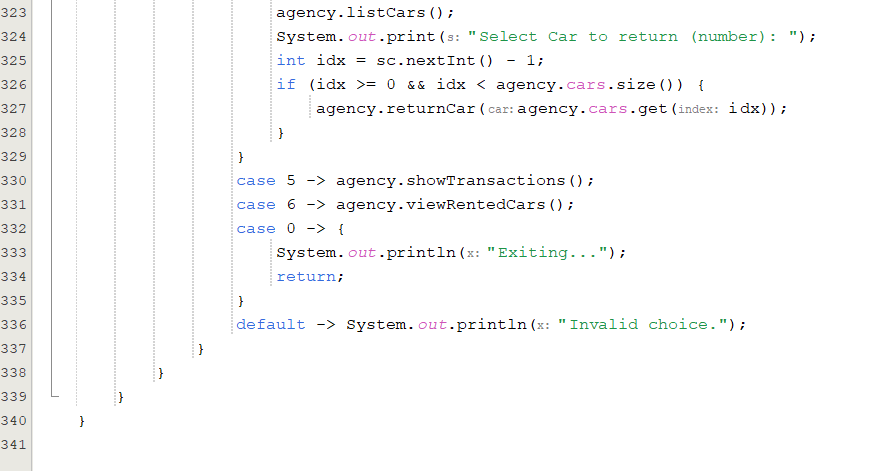
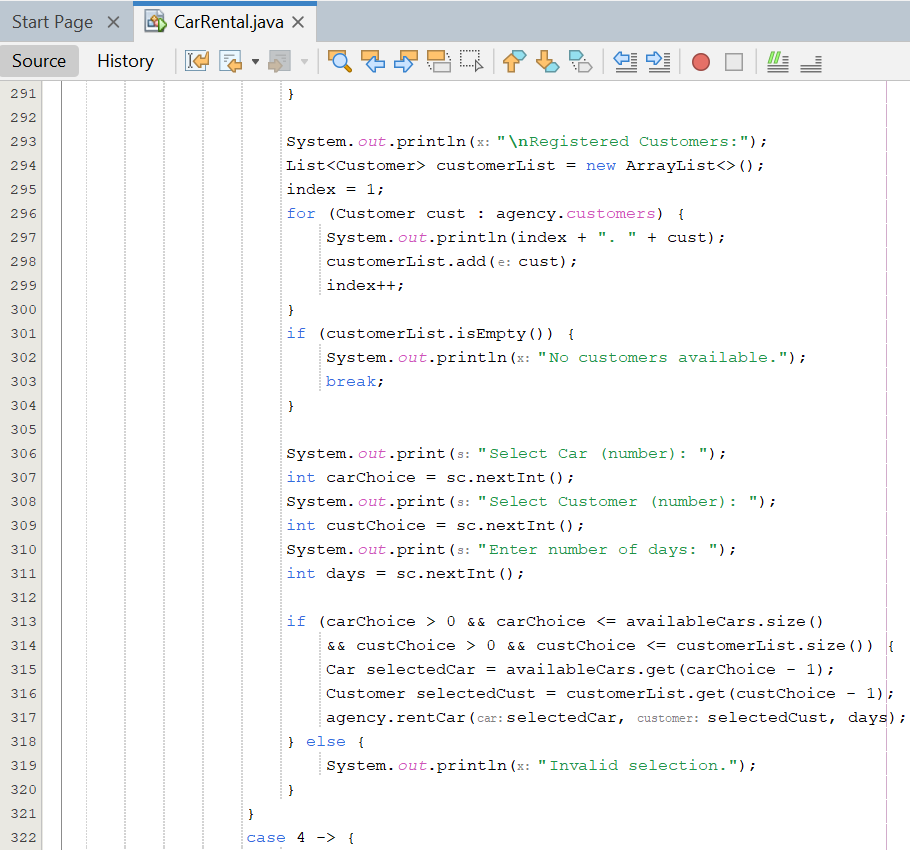
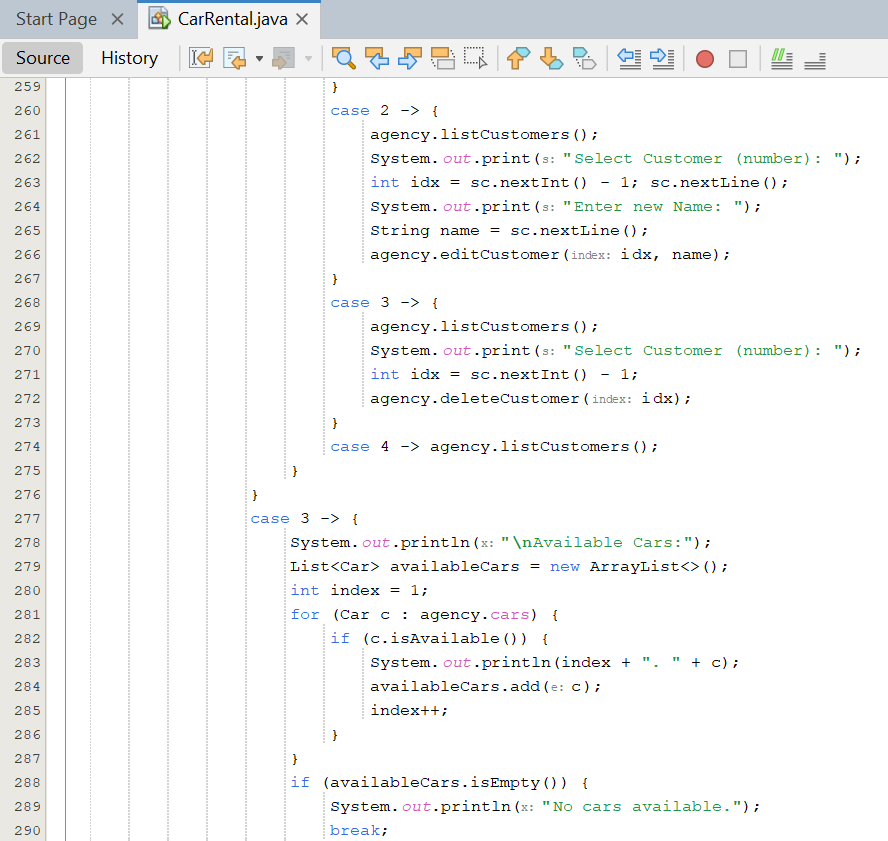
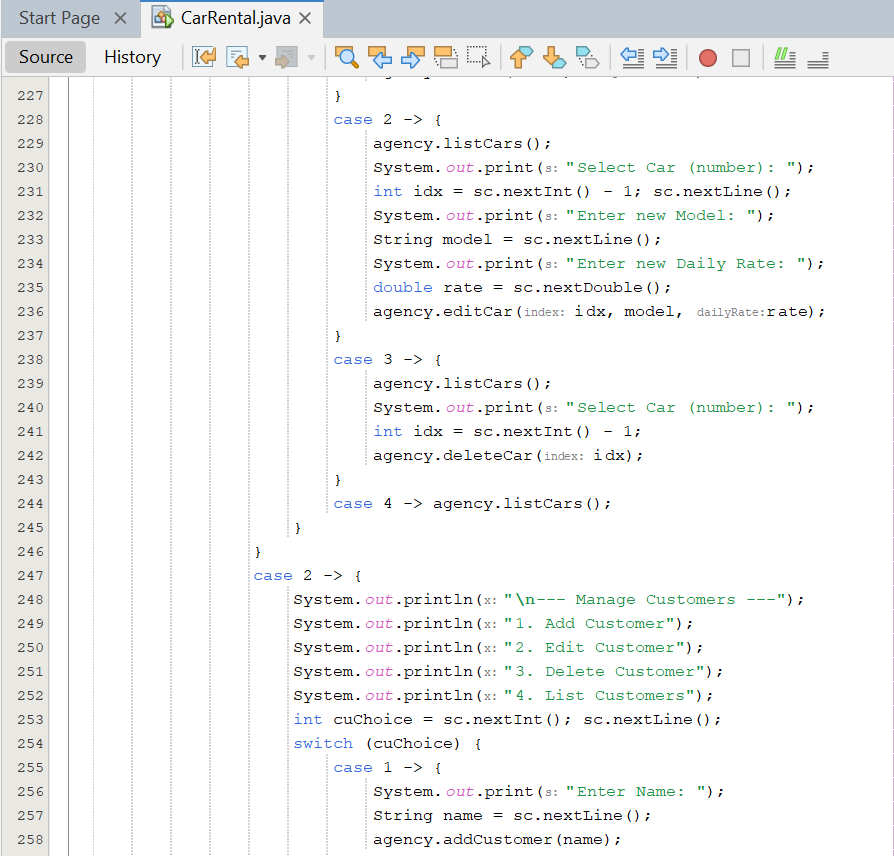
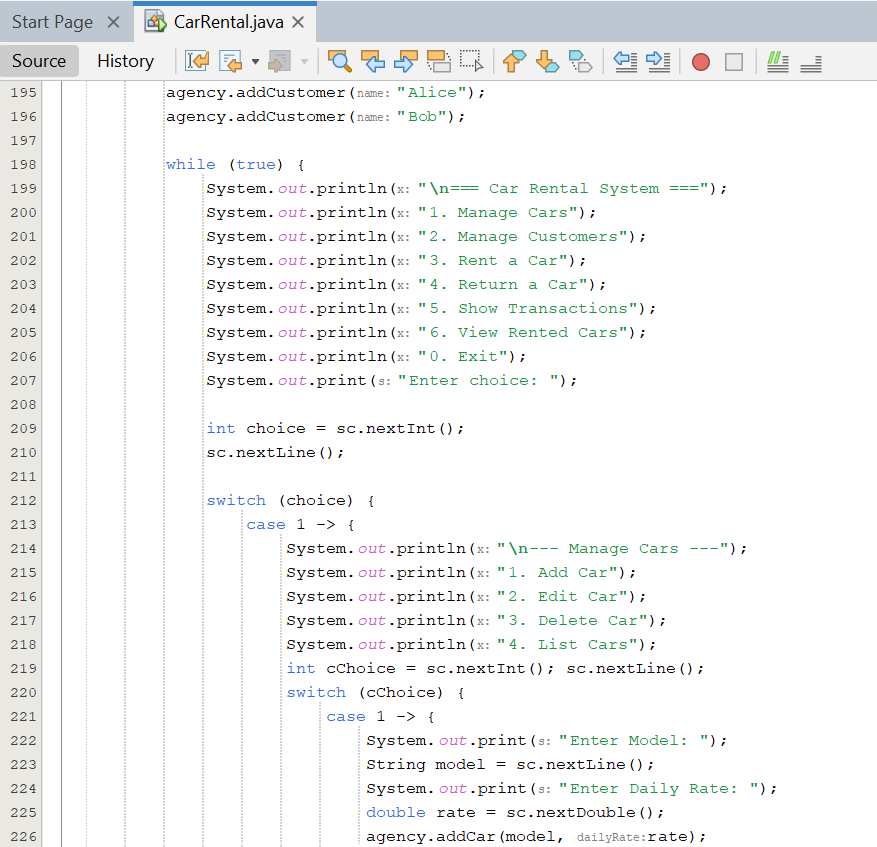
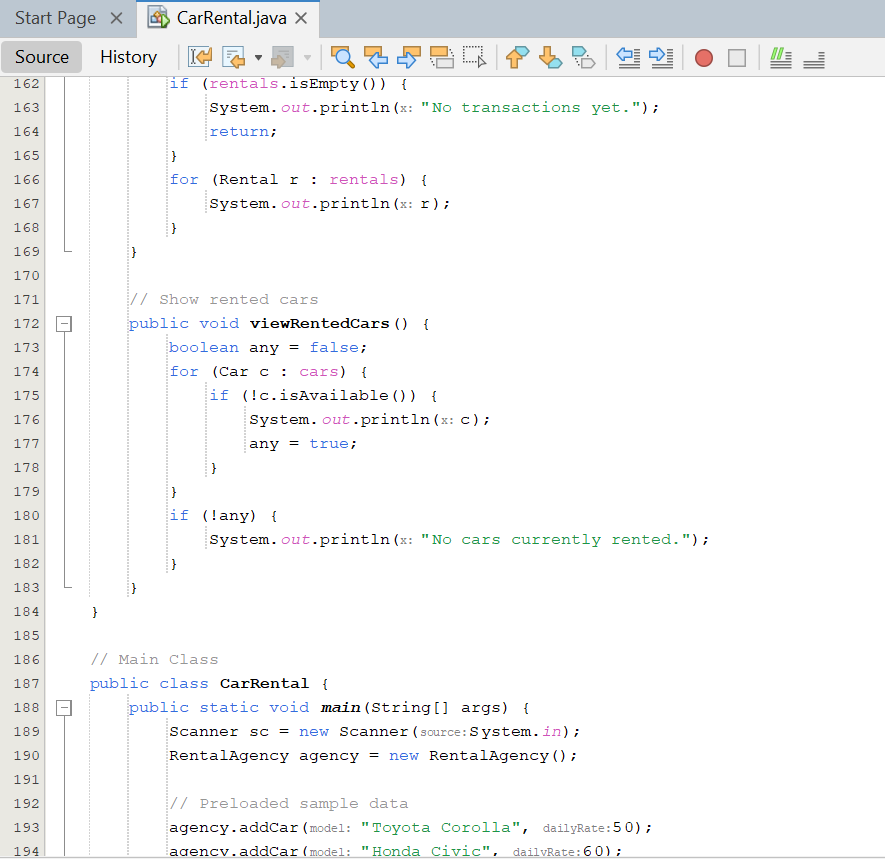
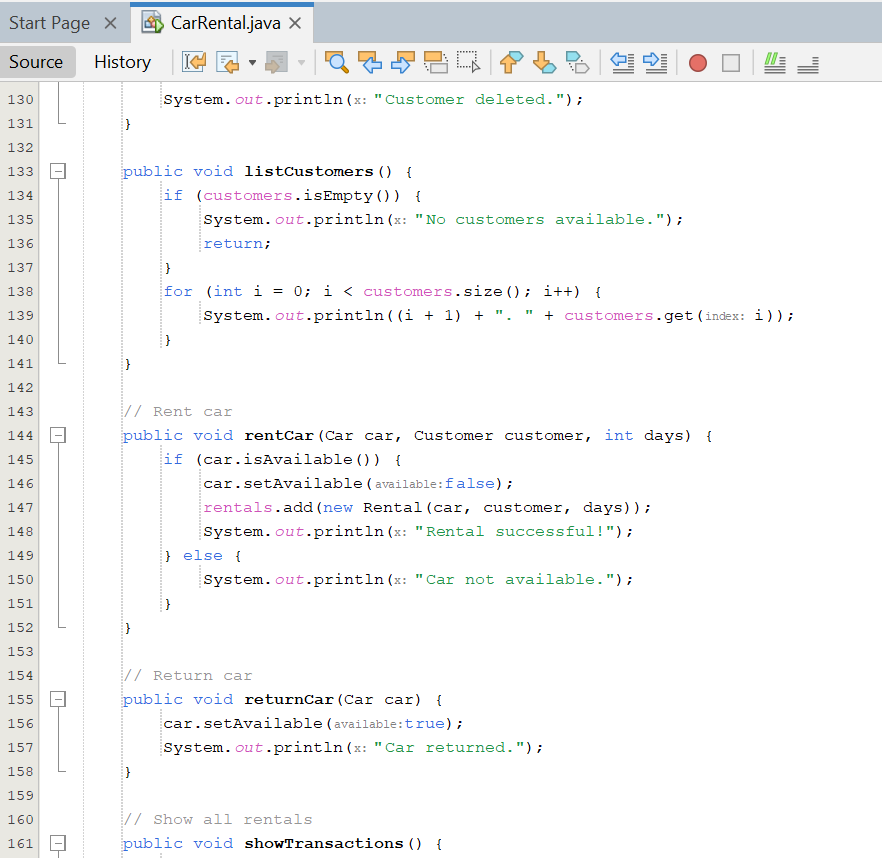
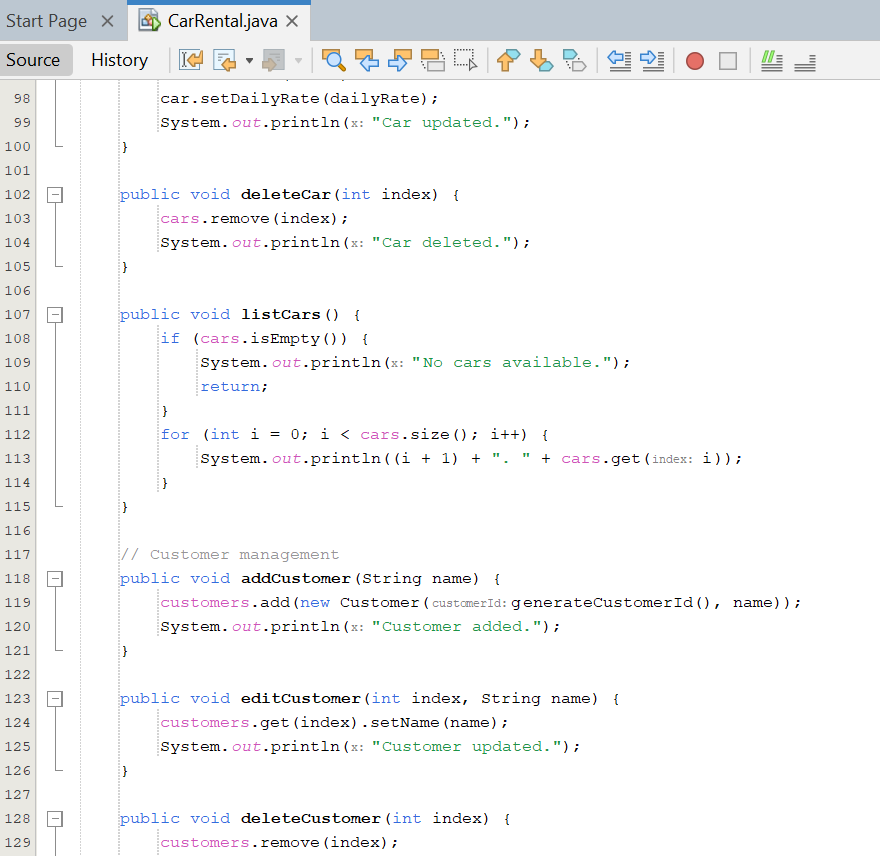
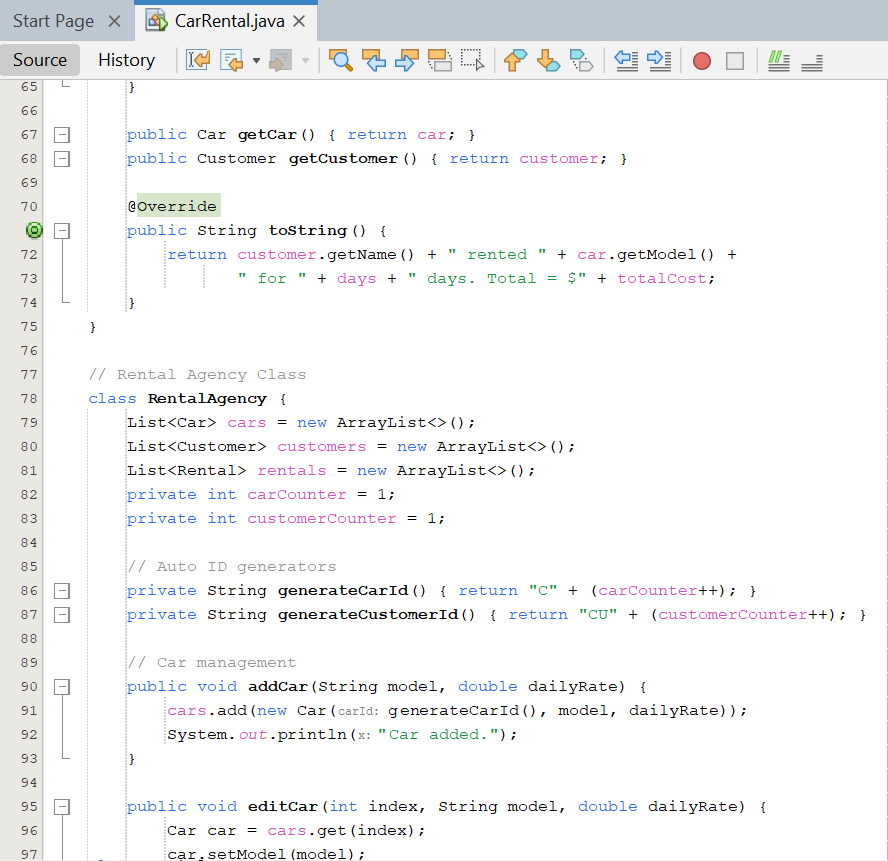
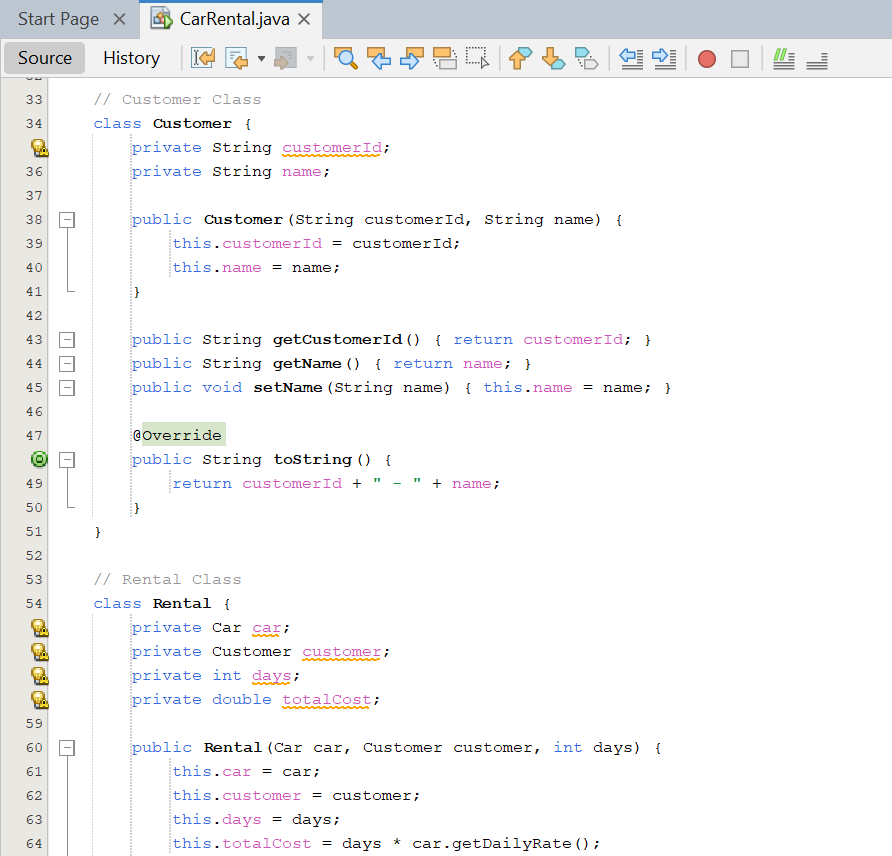
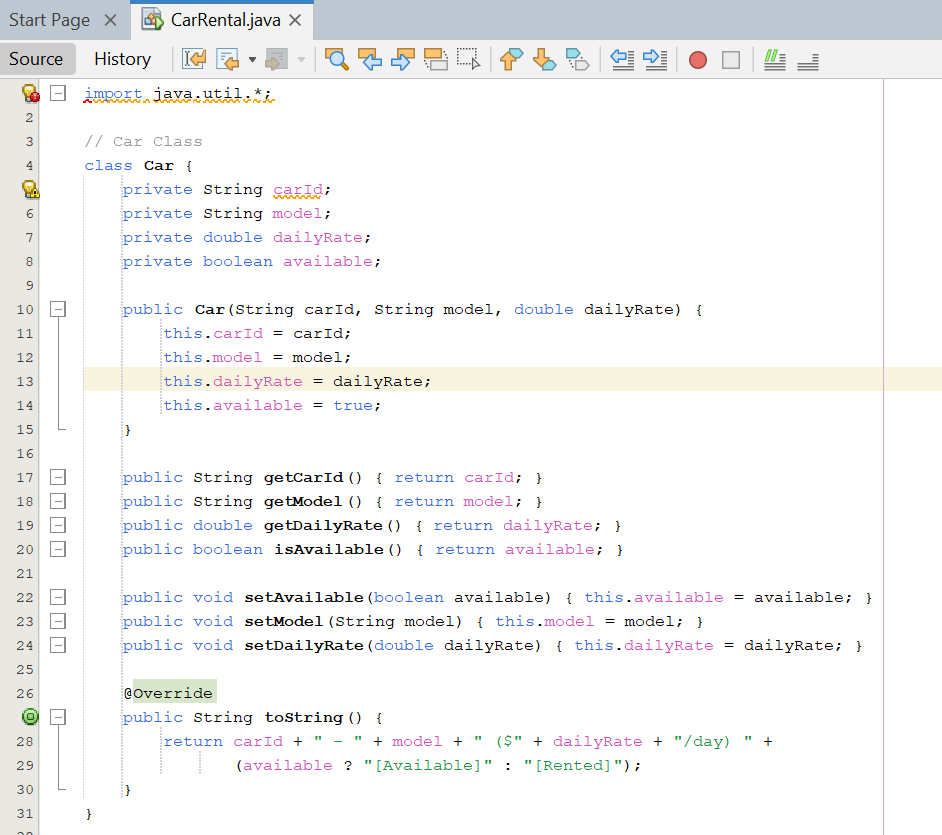
### Fail in Login

****

## 2: Car Rental Management System

The Car Rental Management System is a Java-based console application designed to manage cars, customers, and rental transactions. It allows administrators to maintain a database of available cars and registered customers, process car rentals and returns, and track active and completed transactions.

### SYSTEM CODE:

****

### 2. System Features

#### Car Management

* + Add, edit, and delete cars.
  + View list of available cars.
  + Mark cars as rented or available.

#### Customer Management

* + Add, edit, and delete customer records.
  + View registered customers.

#### Rental Management

* + Rent a car to a customer (only if available).
  + Return rented cars.
  + View all rental transactions.
  + Track currently rented cars.

### 3. Sample Data (Initial Setup)

At system startup, the following sample records exist:

#### Available Cars:

1. C1 – Toyota Corolla ($50/day) [Available]
2. C2 – Honda Civic ($60/day) [Available]

#### Registered Customers:

1. CU1 – Alice
2. CU2 – Bob

### 4. Example Workflow

1. **View Cars** – The system lists all cars with status (Available/Rented).
2. **Add Customer** – Administrator registers a new customer.
3. **Rent a Car** – System verifies availability and links the car to a customer.
4. **Return a Car** – System updates availability and closes the rental.
5. **View Transactions** – Shows all completed and active rentals with costs.

### 5. Technology Used

* **Programming Language**: Java (Object-Oriented)
* **Paradigms**: Encapsulation, Abstraction, Inheritance, Polymorphism
* **Data Structures**: ArrayList for storing Cars, Customers, and Rentals
* **User Interface**: Console-based, menu-driven system

### 6. Error Handling

* Prevents renting cars that are already rented.
* Ensures customers and cars exist before processing rentals.
* Validates user inputs to avoid system crashes.

### SYSTEM INTERFACE:

=== Car Rental System ===

1. Manage Cars

2. Manage Customers

3. Rent a Car

4. Return a Car

5. Show Transactions

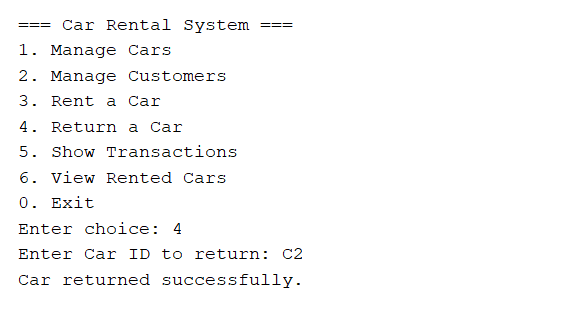
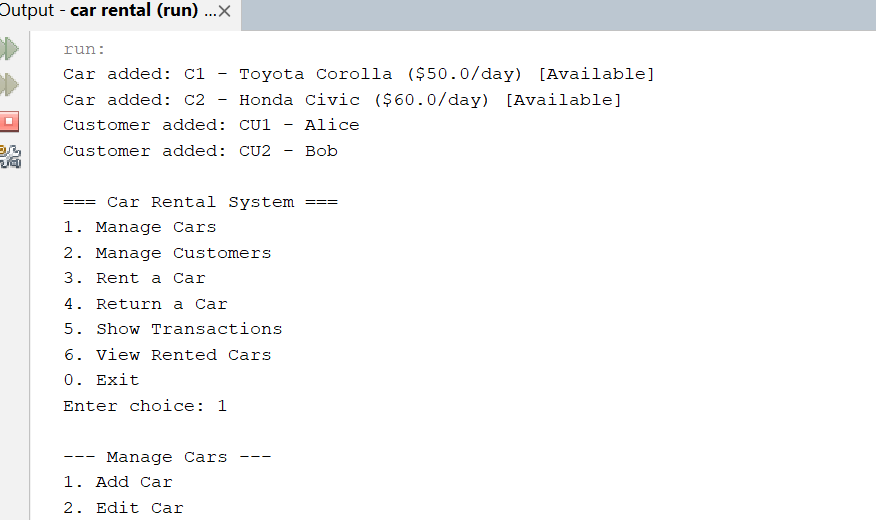
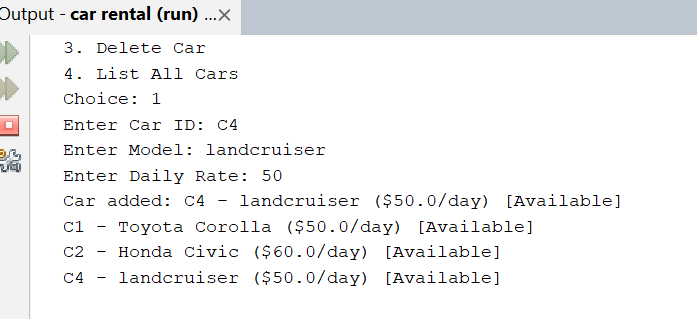
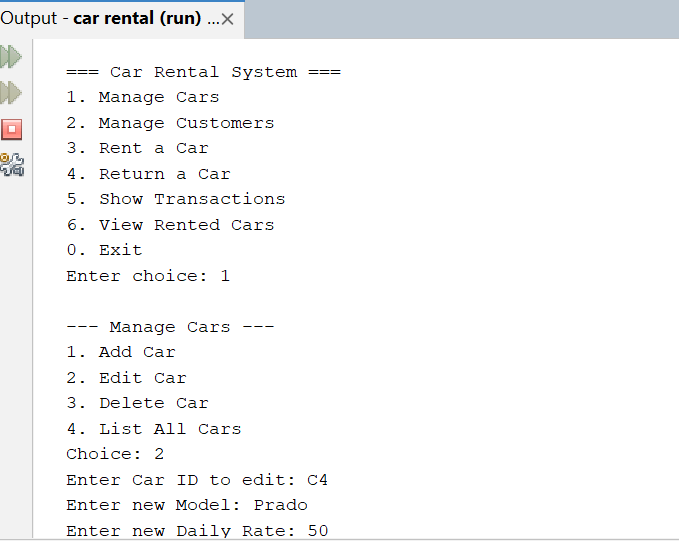
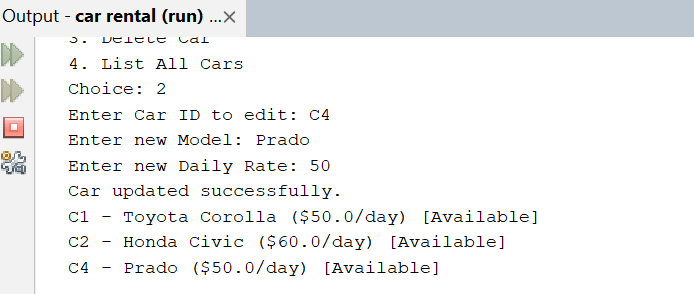
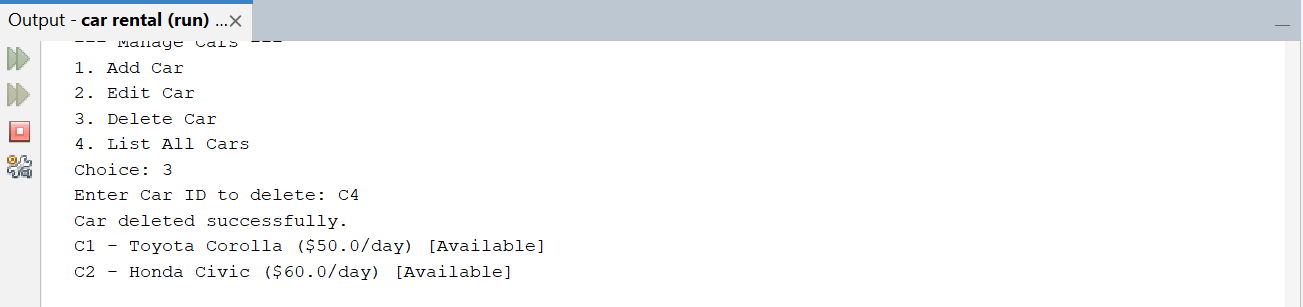
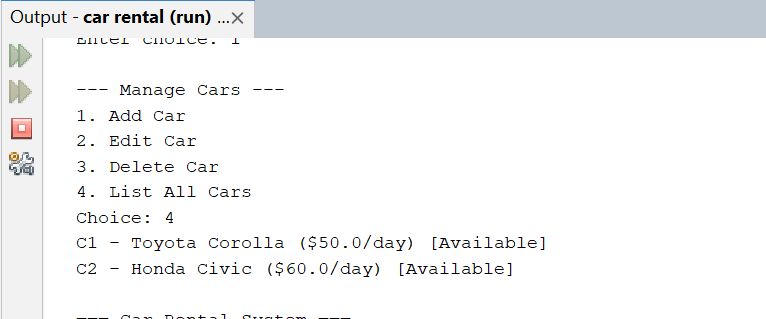
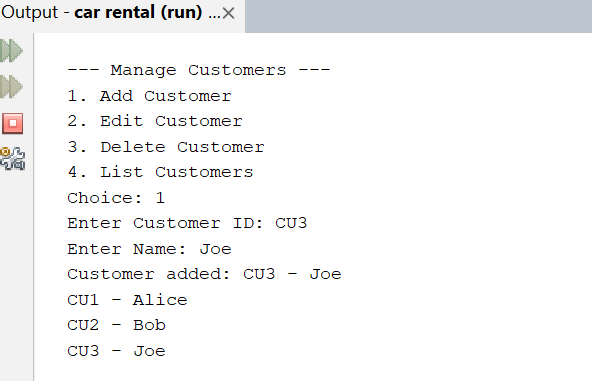
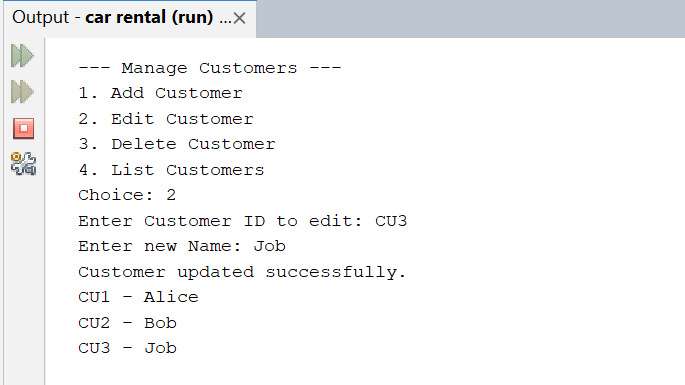
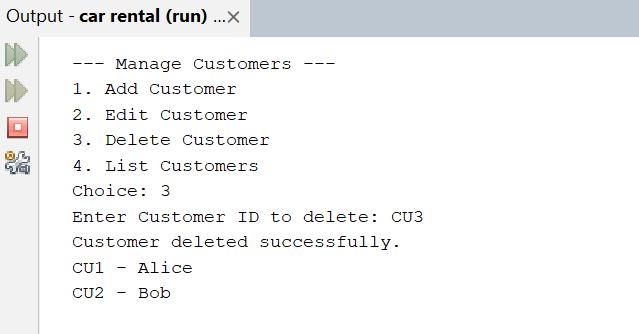
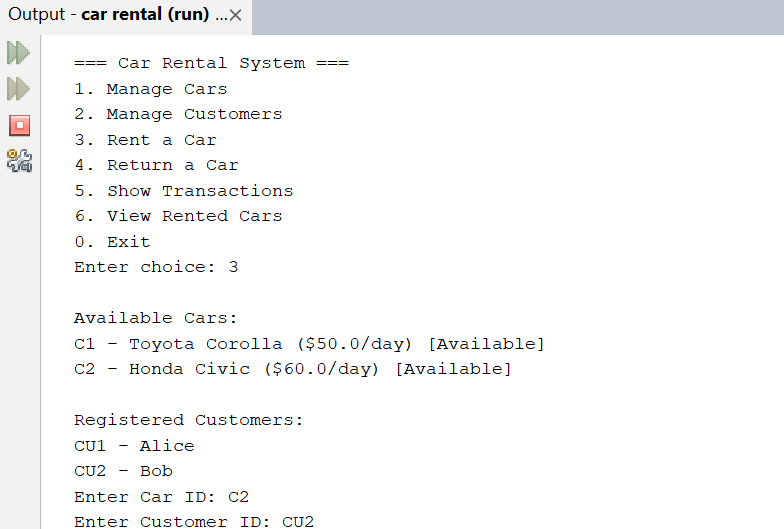
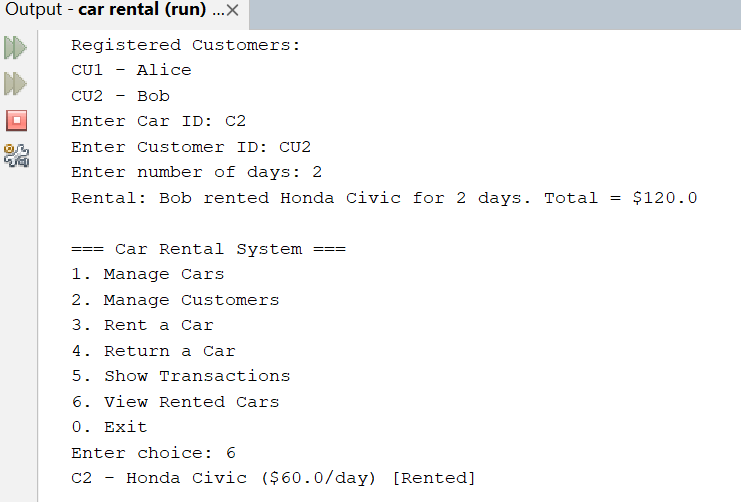
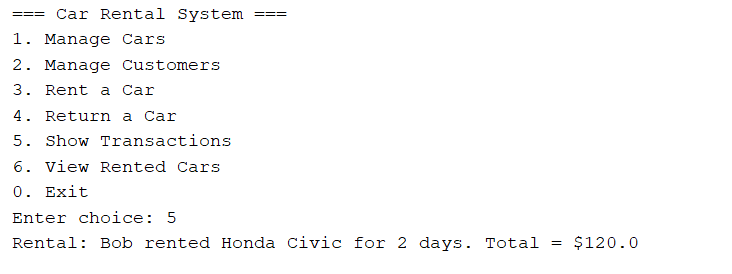
6. View Rented Cars

0. Exit

Enter choice:

### Car Management Outputs

Here is where there is adding editing and deleting of users or cars and renting and returning of the cars viewing of rented cars and showing of transactions is seen



## System Descriptions

### 1. User Login System

The User Login System is a simple yet secure authentication program built in Java. It allows users to log in with a username and password while incorporating features such as password masking (hiding input characters for security) and attempt limitation (restricting login attempts to prevent brute force access).

When the application runs, it prompts the user for credentials. If the input matches the stored username and password, access is granted; otherwise, the user is informed of the failure. After three unsuccessful attempts, the account is locked, ensuring basic security.

This application demonstrates Object-Oriented Programming (OOP) concepts such as encapsulation (hiding password storage details), abstraction (providing a simple login interface), and modularity (separating authentication logic from user interaction).

### 2. Car Rental Management System

The Car Rental Management System is a comprehensive console-based Java application designed to manage cars, customers, and rental transactions. The system allows administrators to:

* Add, edit, delete, and view cars
* Register and manage customers
* Rent out cars to customers with cost calculation
* Process car returns
* Display transaction histories
* View currently rented cars

The program follows a menu-driven approach, making it intuitive for users to navigate different functions. Data such as car details, customer records, and rental history are stored in object lists, ensuring modular and extensible design.

This project highlights OOP principles such as encapsulation (managing car and customer attributes), abstraction (hiding internal logic behind methods), and polymorphism (handling different menu options dynamically). It provides a real-world simulation of a rental agency’s workflow.