JAVA I
DOCUMENTATION

Team: MMR

Project Name: CarHub360

TEAM MEMBERS

Mohammad Taiba	3. Ingenieurinformatik
Marc Praus	3. Ingenieurinformatik
Rudolf Minz	3. Medieninformatik

System Overview

The system overview for the CarHub360 application provides a comprehensive insight into the functioning and objectives of the application. CarHub360 is designed as an all-encompassing solution for the automotive sector, covering both vehicle sales and rentals, offering users an intuitive and efficient management experience.

Purpose of the System

The primary goal of CarHub360 is to deliver a robust platform that consolidates all aspects of vehicle management into a single, user-friendly application. It enables users to manage new and used car sales as well as short- and long-term rentals, including the extension of rental agreements. The application aims to provide users with a seamless and error-free experience by simplifying the purchasing and rental process through automation and well-structured workflows.

Moreover, the application is designed to be easily scalable to keep pace with the company's growth. New features and modules can be integrated to respond to market changes or specific user requirements. In addition, comprehensive documentation and support are provided to facilitate users' onboarding and ongoing usage of the application.
Thus, CarHub360 is not just a tool for day-to-day operations but a strategic partner that helps ensure the long-term competitiveness and success of our clients.

Target Audience

CarHub360 is designed to cater to a wide range of users in the automotive industry, from small car dealerships to large dealership chains. The Primary end-users include:

- Sales Personnel: Employees who are directly involved in the sale of vehicles, responsible for registering new vehicles in the system, creating offers, and finalizing sales contracts.
- Customers: Individuals or companies that use the dealership's services, whether it's purchasing a vehicle or renting for short or long periods.
- Customer Service Representatives: Staff responsible for customer care and advice, including managing customer inquiries and tracking customer satisfaction.
- Accountants: Professionals tasked with payment processing, bookkeeping, and financial reporting.
- Maintenance Technicians: Technical staff responsible for maintaining and servicing the vehicle fleet to ensure operational safety and service for customers.
- Executives: Decision-makers who engage in strategic planning and direct the business course by analyzing usage data and customer feedback.
- IT Administrators: IT professionals who manage system settings, control user access, and ensure compliance with data protection and security.

These diverse groups require a system that is both flexible and robust, supporting their respective tasks efficiently and user-friendly. CarHub360 is designed to meet the varied needs of all users

through an intuitive intelligent backend logic. By providing user-specific features and a customizable platform, the application ensures it adds value to every role within the automotive industry. Whether it's managing sales processes, optimizing customer experiences, or ensuring operational workflows, CarHub360 offers a solution specifically tailored to enhance efficiency and support business growth.

Data Model and Logic

Contract Class

Overview

The Contract class handles the creation and management of both purchase and rental contracts for vehicles. It uses a HashMap to store contract details, ensuring efficient access and manipulation of contract data.

Data Model

Attributes:

- **contracts**: A HashMap mapping integer contract IDs to **ContractDetails** objects, storing the details of each contract.

Methods

- **createPurchaseContract**: Creates a purchase contract for a vehicle. Checks for valid inputs and non-existing contract ID before creation.
- **createRentalContract**: Similar to purchase contracts, but for rentals. Includes start and end dates for the rental period.
- **terminateRentalContract**: Terminates an existing rental contract, marking the vehicle as available again.
- **renewRentalContract**: Renews a rental contract with a new end date, keeping the vehicle unavailable.
- **getTotalPrice**: Calculates the total price for a rental contract based on the number of days rented.
- **getRentalContractDetails**: Retrieves a formatted string of details for a specific rental contract.
- **getPurchaseContractDetails**: Retrieves details for a specific purchase contract in a formatted string.
- **validateRentalPeriod**: A static method to validate the rental period dates.

Inner Class: ContractDetails

- Stores detailed information about a contract, including customer and vehicle details, and rental-specific data.

Payment Class

Overview

The Payment class is responsible for processing payments. It maintains a record of all payments in a HashMap.

Data Model

Attributes:

- **payments**: A HashMap mapping integer payment IDs to **PaymentDetails** objects.

Methods

- **processPayment**: Processes a new payment, ensuring the ID is unique and all parameters are valid before adding it to the collection.
- **getPaymentDetails**: Retrieves detailed information about a payment in a formatted string.

Inner Class: PaymentDetails

- Stores detailed information about a payment, including customer data, payment method, status, and amount.

Customer Class

Overview

The Customer class represents customers and manages their information. It provides functionality to create, delete and retrieve customer details.

Data Model

Attributes:

- customers: A list containing CustomerDetails objects representing individual customers.

Methods:

- createCustomer: Creates a new customer with the provided information and adds it to the customer list if the ID and email are unique.
- deleteCustomer: Deletes a customer from the customer list based on the provided ID.
- getCustomerDetails: Retrieves detailed information about a customer based on the provided ID.

Inner Class: CustomerDetails

- Stores detailed information about a customer, including ID, name, email, birthdate, gender, deletion status, and address.

Customer-Adress Class

Overview

The CustomerAddress class manages customer address information and provides methods for updating and retrieving address details.

Data Model

Attributes:

- customerAddresses: A list storing instances of CustomerAddressDetails representing customer addresses.

Methods:

- updateCustomerAddress: Updates the address details for a customer with the provided information. Returns true if the update is successful, false otherwise.

- getCustomerAddressDetails: Retrieves the details of a customer's address based on the customer ID. Returns a formatted string containing the address details or a message indicating that the customer does not have an address.

Inner Class: CustomerAddressDetails

- The CustomerAddressDetails inner class encapsulates the details of a customer's address.

CustomerHistory Class

Overview

The CustomerHistory class manages customer history records, which include details about customer actions such as vehicle rentals, reviews, and other interactions. It provides methods for creating and retrieving customer history records.

Data Model

Attributes:

- The class contains an inner class CustomerHistoryDetails, which encapsulates the details of each customer history record, including the customer ID, associated customer, vehicle involved, review, description of the action, action date, and indication of whether the action pertains to a rental car.

Methods:

- createCustomerHistory: Creates a new customer history record with the provided information. Returns true if the creation is successful, false otherwise.

- getCustomerHistory: Retrieves the customer history record associated with the specified ID.

- getCustomerFinalReview: Retrieves the final review associated with the specified customer history ID.

Inner Class: CustomerHistoryDetails

- This inner class encapsulates the details of each customer history record.

Maintenance Class

Overview

The Maintenance class handles maintenance records for vehicles.

Data Model

Attributes:

- maintenances: A list storing instances of MaintenanceInfo representing maintenance records.

Methods:

- addMaintenance: Adds a maintenance record to the list. Returns true if the addition is successful, false otherwise.

- getMaintenanceDetails: Retrieves the details of a maintenance record based on the maintenance ID. Returns a formatted string containing the maintenance details or a message indicating that the maintenance ID was not found.

Inner Class: MaintenanceInfo

- The MaintenanceInfo inner class encapsulates the details of a maintenance record.

Vehicle Class

Overview

The Vehicle class handles the creation, updates and deletes of all vehicles. It uses an ArrayList to store vehicle details and their IDs, ensuring an efficient manipulation of vehicles.

Data Model

Attributes:

- vehicles: A list containing CustomerDetails objects representing individual customers.

Methods:

- CreateVehicle: Creates a new vehicle with the provided information and adds it to the vehicle list if the ID is unique.
- UpdateVehicle: Updates the Vehicle details with the provided information. Returns true if the update is successful, false otherwise.
- DeleteVehicle: Deletes an existing vehicle from the array list. Returns true if the update is successful, false otherwise.
- CheckNewKilometerCount: It checks if the new kilometer count for a vehicle is greater than or equal to the current kilometer count. If the vehicle with the specified ID is found, and the new count is acceptable, it updates the kilometer count of the vehicle and returns true, false otherwise.
- getVehicleDetails: Retrieves detailed information about a Vehicle in a formatted string.

Rent-Vehicle Class

Overview

The RentVehicle class represents vehicles available for rent. It extends the Vehicle class to inherit basic vehicle properties.

Data Model

Attributes:

- RentVehicleId: This attribute represents the unique identifier for a rented vehicle.
- RentVehicleVehicleId: It signifies the identifier of the vehicle being rented. This is essentially linking the rental record to the specific vehicle.
- IsAvailable: This boolean attribute indicates whether the vehicle is currently available for rent. If it's true, the vehicle is available; if false, it's currently unavailable.
- DailyPrice: This attribute denotes the daily rental price for the vehicle. It specifies the amount a customer needs to pay per day to rent the vehicle.

- LicensePlate: It represents the license plate number of the vehicle. This serves as a unique identifier for the vehicle, typically displayed on its license plate.
- Deposit: This attribute signifies the amount of money a customer needs to deposit when renting the vehicle. It's a security measure against potential damages or late returns.

Methods:

- getRentVehicleDetails: Retrieves detailed information about the rent vehicle, including basic vehicle details and rent-specific information such as daily price, avilable status and license plate.

Sale-Vehicle Class

Overview

The SaleVehicle class represents vehicles available for sale. It extends the Vehicle class to inherit basic vehicle properties.

Data Model

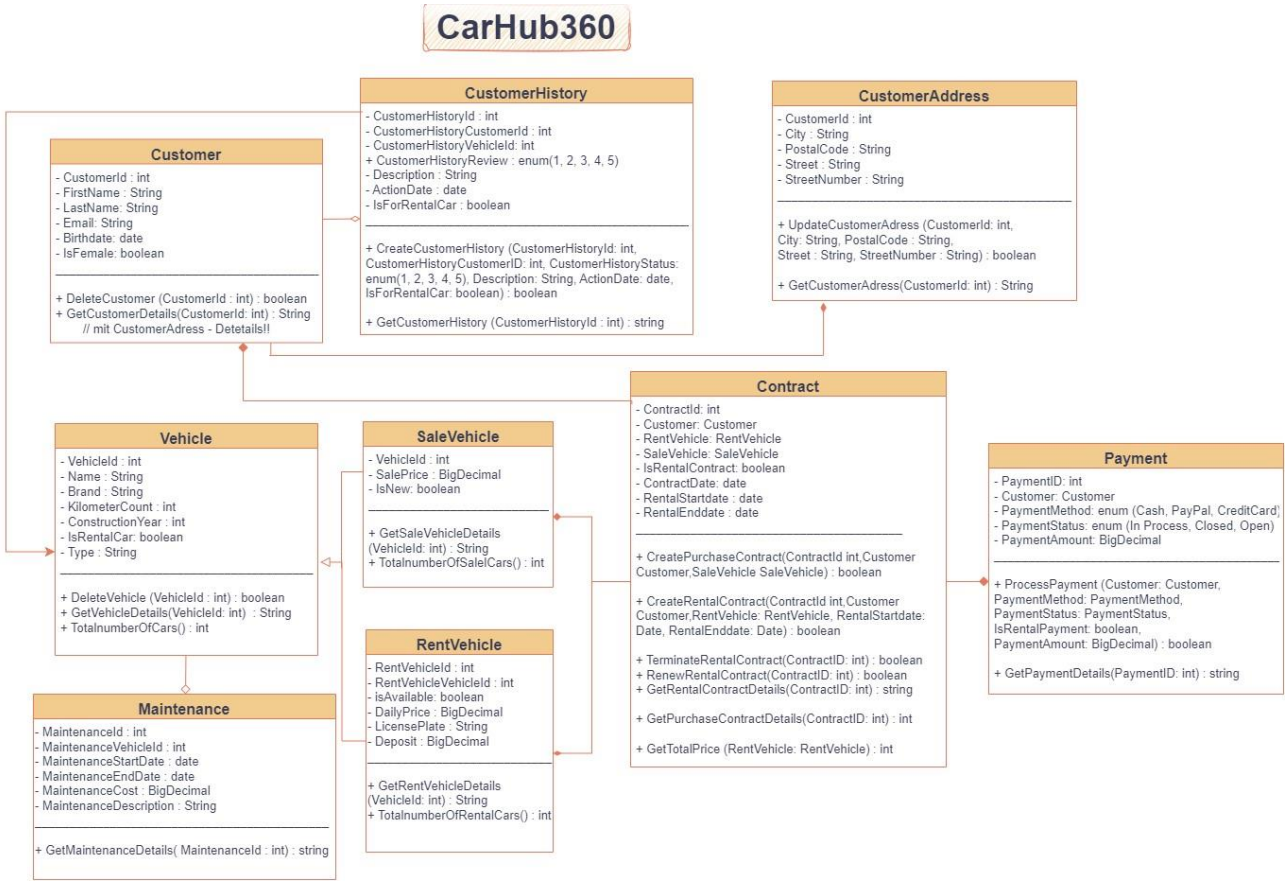
Attributes:

- VehicleId: An integer representing the unique identifier of the vehicle.
- SalePrice: A float representing the sale price of the vehicle.
- IsNew: A boolean indicating whether the vehicle is new.

Methods:

- getSaleVehicleDetails: Retrieves detailed information about the sale vehicle, including basic vehicle details and sale-specific information such as sale price and new status.

System Architecture



User Stories

1. As a sales associate, I want to register new vehicles in the system so that I can keep the sales inventory up to date.
2. As a customer, I want to be able to quickly and easily rent a vehicle so that I can be mobile without lengthy delays.
3. As a customer service representative, I want to access the complete history of customer interactions so that I can provide efficient and personalized service.
4. As an accountant, I want to track and process payments so that I can ensure accurate bookkeeping.
5. As a customer, I want to be able to book vehicles through an online platform so that I can perform the booking process at any time and from anywhere.
6. As a maintenance technician, I want to create and manage maintenance schedules for vehicles so that I can ensure that all vehicles are reliable and safe.
7. As a CEO, I want to be able to view data on vehicle usage and customer feedback so that I can make strategic business decisions.
8. As a marketing manager, I want to set up promotions and discounts in the system so that I can execute effective marketing campaigns.

9. As an IT administrator, I want to manage user roles and access rights in the system so that I can ensure the security of the application and compliance with data protection regulations.
10. As a customer, I want to be able to leave reviews for vehicles so that I can share feedback on quality and service and help other customers with their decision-making.
11. As a warehouse manager, I want to monitor the condition and availability of vehicles so that I can provide accurate information to the sales and rental teams.
12. As a customer, I want to be able to view my bookings and contracts through my user account so that I can conveniently manage my transactions and make adjustments if necessary.

Requirements and dependencies

In order to run CarHub360, you will need to meet the following requirements:

1. **Operating system:** CarHub360is compatible with Windows 10, macOS, and Linux.
2. **Programming language:** CarHub360 is written in Java and requires at least version 8 of the Java Development Kit (JDK) to be installed on your system.
3. **Testing:** CarHub360 includes unit tests that are written using JUnit. In order to run these tests, you will need to have JUnit installed on your system.
4. **Build management:** CarHub360 uses Maven for build management. You will need to have Maven installed on your system in order to build and run CarHub360.
5. **Other dependencies:** CarHub360 does not use any third-party libraries and frameworks.

Please make sure that you meet these requirements before attempting to install and run CarHub360. If you have any questions or encounter any issues, please consult the troubleshooting section of this documentation or contact us for assistance.

Installation instructions

To install and set up CarHub360, follow these steps:

1. **Download** or clone the latest version of CarHub360 from our git repository (https://git.ai.fh-erfurt.de/prgj1-23/mmra)
2. **Extract** the downloaded file to a folder on your computer.
3. **Install** the Java Development Kit (JDK) (https://www.oracle.com/java/technologies/downloads/) if you don't already have it installed. You can download the JDK from the Oracle website.
4. **Install** and **configure** an IDE, such as Eclipse (https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2023-12/R/eclipse-inst-jre-win64.exe&mirror_id=1301) or IntelliJ IDEA (https://www.jetbrains.com/idea).
5. **Install** JUnit and Maven if you don't already have them installed. You can find instructions for installing JUnit and Maven on the JUnit (https://junit.org/junit5) and Maven (https://maven.apache.org/what-is-maven.html) websites, respectively.

After completing these steps, you should be ready to build and test CarHub360. If you have any issues or questions during the installation process, please consult the troubleshooting section of this documentation or contact us for assistance.