# CarHub360

Rudolf Minz, Marc Praus und Mohammad Tahiba

# Gliederung:

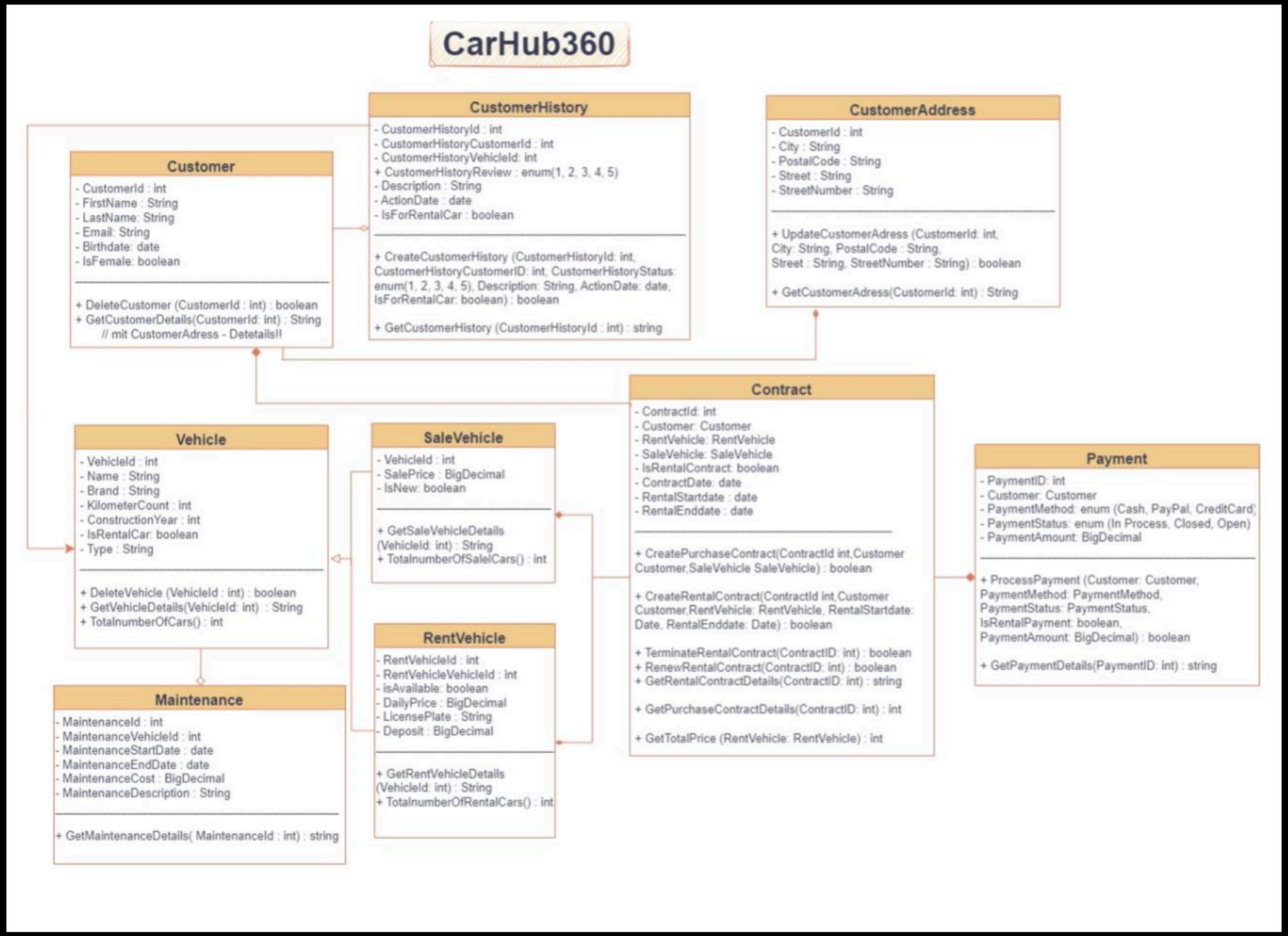
- Unsere Idee
- UML Diagramm
- Was wurde erreicht
- Schwierigkeiten
- Lesson Learned

# Was war unsere Idee und was sollte damit erreicht werden?

CarHub360

- Ein Projekt für einen Autohandel, der sowohl Autos verkauft, als auch verleiht.
- Es soll die Verwaltung von Verträgen, Kunden und Fahrzeugen vereinfachen.
- Soll die Verwaltung von Rechnungen, Zahlungen, und Buchführung automatisieren.

# UML


CarHub360
CustomerHistory
- CustomerHistoryId : int
- CustomerHistoryCustomerId : int
- CustomerHistoryVehicleId: int
+ CustomerHistoryReview : enum(1, 2, 3, 4, 5)
- Description : String
- ActionDate : date
- IsForRentalCar : boolean
+ CreateCustomerHistory (CustomerHistoryId: int, CustomerHistoryCustomerID: int, CustomerHistoryStatus: enum(1, 2, 3, 4, 5), Description: String, ActionDate: date, IsForRentalCar: boolean) : boolean
+ GetCustomerHistory (CustomerHistoryId : int) : string
CustomerAddress
- CustomerId : int
- City : String
- PostalCode : String
- Street : String
- StreetNumber : String
+ UpdateCustomerAdress (CustomerId: int, City: String, PostalCode : String, Street : String, StreetNumber : String) : boolean
+ GetCustomerAdress(CustomerId: int) : String
Customer
- CustomerId : int
- FirstName : String
- LastName: String
- Email: String
- Birthdate: date
- IsFemale: boolean
+ DeleteCustomer (CustomerId : int) : boolean
+ GetCustomerDetails(CustomerId: int) : String
// mit CustomerAdress - Detetails!!
Contract
- ContractId: int
- Customer: Customer
- RentVehicle: RentVehicle
- SaleVehicle: SaleVehicle
- IsRentalContract: boolean
- ContractDate: date
- RentalStartdate : date
- RentalEnddate : date
+ CreatePurchaseContract(ContractId int,Customer Customer,SaleVehicle SaleVehicle) : boolean
+ CreateRentalContract(ContractId int,Customer Customer,RentVehicle: RentVehicle, RentalStartdate: Date, RentalEnddate: Date) : boolean
+ TerminateRentalContract(ContractID: int) : boolean
+ RenewRentalContract(ContractID: int) : boolean
+ GetRentalContractDetails(ContractID: int) : string
+ GetPurchaseContractDetails(ContractID: int) : int
+ GetTotalPrice (RentVehicle: RentVehicle) : int
Vehicle
- VehicleId : int
- Name : String
- Brand : String
- KilometerCount : int
- ConstructionYear : int
- IsRentalCar: boolean
- Type : String
+ DeleteVehicle (VehicleId : int) : boolean
+ GetVehicleDetails(VehicleId: int) : String
+ TotalnumberOfCars() : int
SaleVehicle
- VehicleId : int
- SalePrice : BigDecimal
- IsNew: boolean
+ GetSaleVehicleDetails (VehicleId: int) : String
+ TotalnumberOfSaleiCars() : int
Payment
- PaymentID: int
- Customer: Customer
- PaymentMethod: enum (Cash, PayPal, CreditCard)
- PaymentStatus: enum (In Process, Closed, Open)
- PaymentAmount: BigDecimal
+ ProcessPayment (Customer: Customer, PaymentMethod: PaymentMethod, PaymentStatus: PaymentStatus, IsRentalPayment: boolean, PaymentAmount: BigDecimal) : boolean
+ GetPaymentDetails(PaymentID: int) : string
RentVehicle
- RentVehicleId : int
- RentVehicleVehicleId : int
- isAvailable: boolean
- DailyPrice : BigDecimal
- LicensePlate : String
- Deposit : BigDecimal
+ GetRentVehicleDetails (VehicleId: int) : String
+ TotalnumberOfRentalCars() : int
Maintenance
- MaintenanceId : int
- MaintenanceVehicleId : int
- MaintenanceStartDate : date
- MaintenanceEndDate : date
- MaintenanceCost : BigDecimal
- MaintenanceDescription : String
+ GetMaintenanceDetails( MaintenanceId : int) : string

# Was wurde Erreicht?

```java
import ...
public class Customer {

    private class CustomerDetails {
        private int customerId;
        private String firstName;
        private String lastName;
        private String email;
        private Date birthdate;
        private boolean isFemale;
        private boolean isDeleted;
        private CustomerAddress customerAddress; // this is composition

        public CustomerDetails(int customerId, String firstName, String lastName,
                               String email, Date birthdate, boolean isFemale) {
            this.customerId = customerId;
            this.firstName = firstName;
            this.lastName = lastName;
            this.email = email;
            this.birthdate = birthdate;
            this.isFemale = isFemale;
            this.isDeleted = false;
            this.customerAddress = new CustomerAddress();
        }
```

# Was wurde Erreicht?

```
public int getCustomerId() { return customerId; }
public void setCustomerId(int customerId) { this.customerId = customerId; }

public String getFirstName() { return firstName; }
public void setFirstName(String firstName) { this.firstName = firstName; }

public String getLastName() { return lastName; }
public void setLastName(String lastName) { this.lastName = lastName; }

public String getEmail() { return email; }
public void setEmail(String email) { this.email = email; }

public Date getBirthdate() { return birthdate; }
public void setBirthdate(Date birthdate) { this.birthdate = birthdate; }

public boolean isFemale() { return isFemale; }
public void setFemale(boolean female) { this.isFemale = female; }

public boolean isDeleted() { return isDeleted; }
public void setDelete(boolean deleted) { this.isDeleted = deleted; }

public CustomerAddress getCustomerAddress() { return customerAddress; }
```

# Was wurde Erreicht?

```
private Map<Integer, CustomerDetails> customers = new HashMap<>();


public boolean createCustomer(int customerId, String firstName, String lastName,
                              String email, Date birthdate, boolean isFemale) {
    if (!customers.containsKey(customerId)) {
        CustomerDetails newCustomer = new CustomerDetails(customerId, firstName, lastName, email, birthdate, isFemale);
        customers.put(customerId, newCustomer);
        return true;
    } else {
        return false;
    }
}
```

# Was wurde Erreicht?

```java
package de.fherfurt.customer;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.util.Date;

import static org.junit.Assert.*;


public class CustomerTest {

    private Customer customer;

    @Before
    public void setUp() { customer = new Customer(); }

    @After
    public void tearDown() { customer = null; }

    @Test
    public void testCreateCustomer() {
        assertTrue(customer.createCustomer( customerId: 1, firstName: "Helmut", lastName: "Mustermann",
                email: "HelmutDerMann@mail.de", new Date(), isFemale: true));
        assertFalse(customer.createCustomer( customerId: 1, firstName: "Elena", lastName: "Mustermann",
                email: "kokosnuss@gmail.com", new Date(), isFemale: false));
    }
```

# Herausforderungen

- Den Gesamtüberblick zu behalten
- Die Beziehungen zwischen den Klassen zu realisieren
- Fehler im Code finden und korrigieren

# Lesson learned:

- Genereller Umgang mit Java
- Umgang mit Unit-Tests
- Strukturierte Planung
- Klare Verantwortlichkeiten
- Dokumentation
- Fehler Korrektur

# Danke fürs Zuhören