“Share R Us”

MSc in Business Administration and E-business

Introduction to programming and distributed systems (T Course)

Authors:	Dustin Jaacks, 290993-4403 Amir Bohnenkamp, 200392-0057
E-mail addresses:	duja17ab@student.cbs.dk ambo17ab@student.cbs.dk
Submission:	08.12.2017
Number of Pages:	20+

Test Data

Login

1. As a customer
    a. username: mdee
    b. password: dee1423
2. As a car owner
    a. username: jche
    b. password: chev1231
3. administrator
    a. username: jdoe
    b. password: root

Register a user

- First name: Henry
- Last name: Ford
- Street name: Bilenvej
- House number: 42
- Postcode: 2450
- Date of birth: 12.07.1991
- Telephone: 15123421
- CPR: 123456-1234

Credit Card Information

- Number: 4916722136319146
- Expiration: 01/2020
- Security code: 153

Mobile Pay Number

- Phone number: 12345678

Make a Booking

- Login as “customer” and select “Make a booking”
- Type: 4 for Sports Car
- ID: 30
- Confirm (1)
- Start date: 01.01.2018
- Start time: 18:00
- Return time: 20:30

Register a Car

- Login as “car owner”
- Name of the car: Batmobile
- Type: 3 for Sports Car
- Brand: Wayne
- Transmission: 1 for Manual
- Amount of seats: 1
- Hourly rate: 1000
- Description: Top Secret

```java
package controller;

import data.Database;
import view.UserInterface;

/**
 *
 * @author amir.bohnenkamp@googlemail.com, dustin.jaacks@gmail.com
 *
 * This class is the entry point of the ShareRUs Application and as such
 * contains the main method. This class class constructors and methods from
 * other classes when needed.
 *
 * Notes:
 * <ul>
 * <li>We did not implement the CarDatabase. Instead we decided to make a
    global
 * database class that reads and writes to all external files.</li>
 * <li>For the users we decided to not have child classes because the only
    thing
 * that changes is the role. Making separate classes for every role would
    not
 * be good practice, we figured. Check the Payment child classes for a
    demonstration
 * of the concept of inheritance.</li>
 * <li>The sudo password is "supersecret".</li>
 * <li>We decided for a static Database design. This is because we not only
    read but
 * also manipulate/edit/write to the database. Changes made in one instance
    of a
 * database would ultimately lead to different versions of databases which
    would
 * compromise the functionality of this application.</li>
 * <li>We developed our own approach to initialize instances read from the
    database.
 * Please check Booking first to read an explanation.</li>
 * </ul>
 *
 */
public class Application {
    public static void main(String[] args) {
        // Initialize the database
        Database.read("users");
        Database.read("cars");
        Database.read("bookings");
        Database.read("payments");

        /*
         * Debug: Auto-login
         *
         * 1: administrator, 2: customer, 3: car owner
         */
        //Auth.autoLogin(3);

        // Display main navigation
        UserInterface.hello();
    }
}
```

```java
package view;

import data.Database;
import model.Auth;
import model.Booking;
import model.Car;
import model.Cash;
import model.CreditCard;
import model.MobilePay;
import model.Payment;
import model.User;

import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * View Class
 *
 */
public class UserInterface {
    static Scanner input = new Scanner(System.in);

    public static void hello() {
        /*
         * DEBUG
         *//*/
        System.out.print(User.toString(Database.getUsers()));
        System.out.print(Car.toString(Database.getCars()));
        System.out.print(Booking.toString(Database.getBookings()));
        System.out.print(Payment.toString(Database.getPayments()));
        /**/

        // Print welcome message
        System.out.println("Welcome to Share R Us");
        UserInterface.navigation();
    }

    /**
     * The static login class.
     * Sets the activeUser.
     */
    public static void login() {
        input = new Scanner(System.in);
        // Request credentials
        System.out.printf("\n\n\tLOGIN");
        System.out.printf("\n\tUsername: ");
        String usernameInput = input.nextLine();
        System.out.printf("\tPassword: ");
        String passwordInput = input.nextLine();

        for ( User user : Database.getUsers() ) {
            Boolean creentialsChecks = user.checkCredentials(usernameInput,
                passwordInput);
            if ( creentialsChecks ) {
                // Successful login attempt
                System.out.printf("\nYou are now logged in.");
                Auth.setActiveUser(user);
                break;
```

```
            }
        }

        // Unsuccessful login attempt
        if ( Auth.getActiveUser() == null ) {
            System.out.printf("\nYou typed in a wrong password and/or
                username.\n");
        }
    }

    /**
     * Sets the activeUser to null and thus logs the user out.
     */
    public static void logout() {
        Auth.setActiveUser(null);
        System.out.println("You have been logged out.");
        System.out.println("Goodbye!");
    }

    /**
     * Print the navigation
     *
     * Show different navigation options based on the user's role.
     */
    public static void navigation() {
        User activeUser = Auth.getActiveUser();
        System.out.println("\n\n\tHOME");

        int options = 0;
        if ( activeUser == null ) {
            System.out.printf("\t  | %-3s | %-10s |\n", 1, "Login");
            System.out.printf("\t  | %-3s | %-10s |\n\n", 2, "Register");
            options = 2;
        } else if ( activeUser.getRole().equals("customer") ) {
            System.out.printf("\t  | %-3s | %-15s |\n", 1, "Logout");
            System.out.printf("\t  | %-3s | %-15s |\n", 2, "My Bookings");
            System.out.printf("\t  | %-3s | %-15s |\n", 3, "Make a booking")
                ;
            System.out.printf("\t  | %-3s | %-15s |\n", 4, "Search a car");
            System.out.printf("\t  | %-3s | %-15s |\n", 5, "My Account");
            options = 5;
        } else if ( activeUser.getRole().equals("car owner") ) {
            System.out.printf("\t  | %-3s | %-15s |\n", 1, "Logout");
            System.out.printf("\t  | %-3s | %-15s |\n", 2, "My Cars");
            System.out.printf("\t  | %-3s | %-15s |\n", 3, "Register a car")
                ;
            System.out.printf("\t  | %-3s | %-15s |\n", 4, "My Account");
            options = 4;
        } else if ( activeUser.getRole().equals("administrator") ) {
            System.out.printf("\t  | %-3s | %-15s |\n", 1, "Logout");
            System.out.printf("\t  | %-3s | %-15s |\n", 2, "Search a car");
            System.out.printf("\t  | %-3s | %-15s |\n", 3, "Manage database"
                );
            System.out.printf("\t  | %-3s | %-15s |\n", 4, "My Account");
            options = 4;
        }

        int choice;
        do {
            System.out.printf("\nPlease type in your destination: ");
```

```
            choice = input.nextInt();
        } while ( choice <= 0 && choice > options );

        // If user is not logged-in show the following
        if ( activeUser == null ) {
            switch ( choice ) {
            // Login
            case 1:
                UserInterface.login();
                break;

            // Register
            case 2:
                System.out.printf("\n\nNEW USER\n");
                String role             = UserInterface.inputUserRole();
                String firstname    = UserInterface.inputFirstName();
                String lastname         = UserInterface.inputLastName();
                String streetname   = UserInterface.inputStreetname
                    (firstname);
                String houseNumber  = UserInterface.inputHouseNumber
                    (firstname);
                int postcode        = UserInterface.inputPostcode(firstname)
                    ;
                String dob          = UserInterface.inputDOB();
                String telephone    = UserInterface.inputTelephone();
                String cpr          = UserInterface.inputCPR();

                User newUser = new User(role, firstname, lastname,
                    streetname,
                        houseNumber, postcode, dob, telephone, cpr);
                newUser.toDB();
                Database.addUser(newUser);
                UserInterface.login();
                break;
            }
        } else if ( activeUser.getRole().equals("customer") ) {
            switch ( choice ) {
            // Logout
            case 1:
                UserInterface.logout();
                break;

            // My Bookings
            case 2:
                UserInterface.myBookings();
                break;

            // Make a Booking
            case     3:
                Booking booking = UserInterface.makeABooking(0);
                UserInterface.makeAPayment(booking);
                Database.addBooking(booking);
                booking.toDB();
                System.out.println("Thank you for your booking. Enjoy your
                    ride!");

                UserInterface.checkDiscount();
                break;

            // Search a Car
```

```
            case 4:
                UserInterface.searchCar();
                break;

            // My Account
            case 5:
                UserInterface.myAccount();
                break;
            }
        } else if ( activeUser.getRole().equals("car owner") ) {
            switch ( choice ) {
            // Logout
            case 1:
                UserInterface.logout();
                break;

            // My Cars
            case 2:
                UserInterface.myCars();
                break;

            // Register a Car
            case 3:
                UserInterface.registerCar();
                break;

            // My Account
            case 4:
                UserInterface.myAccount();
                break;

            }
        } else if ( activeUser.getRole().equals("administrator") ) {
            switch ( choice ) {
            // Logout
            case 1:
                UserInterface.logout();
                break;

            // Search a Car
            case 2:
                UserInterface.searchCar();
                break;

            // Manage Database
            case 3:
                UserInterface.manageDatebase();
                break;

            // My Account
            case 4:
                UserInterface.myAccount();
                break;
            }
        }

        navigation();
    }

    /**
```

```
     * UI method to create (and edit) a Booking
     *
     * @param update ID of a Booking that will be updated
     * @return The created Booking
     */
    public static Booking makeABooking(int update) {
        Booking booking = null;
        boolean check = true;
        int price, userId, startTimestamp, returnTimestamp;
        String date;

        int choice;
        String filterType = null;

        User activeUser     = Auth.getActiveUser();
        userId = activeUser.getId();

        int confirm;
        Car selectedCar = null;
        do {
            do {
                // Show all available car types
                System.out.printf("\n\n\tAVAILABLE TYPES\n");
                System.out.printf("\t  | %-3s | %-12s |\n", 1, "Show All");
                System.out.printf("\t  | %-3s | %-12s |\n", 2, "Sedan");
                System.out.printf("\t  | %-3s | %-12s |\n", 3, "Convertible"
                    );
                System.out.printf("\t  | %-3s | %-12s |\n", 4, "Sports Car")
                    ;
                System.out.printf("\t  | %-3s | %-12s |\n", 5, "SUV");
                System.out.printf("\nWhat type of car are you looking for
                    (type #)? ");
                choice = input.nextInt();
            } while ( choice <= 0 || choice > 5 );

            switch (choice) {
            case 1:
                filterType = null;
                break;
            case 2:
                filterType = "Sedan";
                break;
            case 3:
                filterType = "Convertible";
                break;
            case 4:
                filterType = "Sports Car";
                break;
            case 5:
                filterType = "SUV";
                break;
            }

            // Filter cars by type and print
            List <Car> filteredCars = Car.filter(Database.getCars(), "type",
                filterType);
            System.out.printf(Car.toString("FILTERED CARS", filteredCars));

            do {
                System.out.printf("\nType in the ID of the car you want to
```

```
            book: ");
            int carChoice = input.nextInt();
            selectedCar = Car.getById(carChoice);
        } while ( selectedCar == null );

        System.out.println("You have chosen:");
        System.out.print(Car.toString(selectedCar));

        System.out.println("\nPlease confirm the booking");
        System.out.printf("\t  | %-3s | %-20s |\n", 1, "Confirm and
            proceed");
        System.out.printf("\t  | %-3s | %-20s |\n", 2, "Go back");
        confirm = input.nextInt();
    } while ( confirm != 1 );

    System.out.printf("\nOn which day do you want to rent your selected
        car?");
    input.nextLine();

    boolean err = false;
    do {
        do {
            System.out.printf("\nPlease type in your start date
                (DD.MM.YYYY): ");
            date = input.nextLine();
        } while( !date.matches("\\d{2}.\\d{2}.\\d{4}") );

        int firstDot    = date.indexOf('.');
        int secondDot   = date.indexOf('.', 5);

        int day         = Integer.parseInt(date.substring(0, firstDot));
        int month   = Integer.parseInt(date.substring(++firstDot,
            secondDot));
        int year    = Integer.parseInt(date.substring(++secondDot));

        // Validate date
        // To Do: Use Calendar class to validate date
        // https://stackoverflow.com/questions/226910/how-to-sanity-
            check-a-date-in-java
        if ( day < 1 || month < 1 || day > 31 || month > 12 || year >
            2050 || year < 2017) {
            System.out.println("The date you entered is invalid, please
                try again:");
            err = true;
        } else {
            err = false;
        }
    } while ( err );

    System.out.println("See here all available time slots:");
    System.out.printf("\n\n\t"+date+"\n");

    // Get time slots
    boolean[] timeSlots = Booking.getTimeSlots(selectedCar.getId(), date
        , update);
    UserInterface.formatTimeSlots(timeSlots);

    int duration = 0, hours, minutes, startHour, startMinutes,
        returnHour, returnMinutes;
    do {
```

```
String startTime;
do {
    do {
        System.out.printf("\n\nYou can rent a car temporarily
            between 30 minutes and 4 hours.\n");
        System.out.print("Please type in your start time
            (hh:mm): ");
        startTime = input.nextLine();
    } while( !startTime.matches("\\d{2}:\\d{2}") );
    startHour       = Integer.parseInt(startTime.substring(0,2))
        ;
    startMinutes    = Integer.parseInt(startTime.substring(3,5))
        ;
} while (startHour < 0 || startHour > 23 || startMinutes > 59);
startTimestamp = startHour*60 + startMinutes;

String returnTime;
do {
    do {
        System.out.print("Please type in your return time
            (hh:mm): ");
        returnTime = input.nextLine();
    } while( !returnTime.matches("\\d{2}:\\d{2}") );
    returnHour      = Integer.parseInt(returnTime.substring(0,2)
        );
    returnMinutes   = Integer.parseInt(returnTime.substring(3,5)
        );
} while ( returnHour < 0 || returnHour > 23 || returnMinutes >
    59 );
returnTimestamp = returnHour*60 + returnMinutes;

duration    = returnTimestamp-startTimestamp;
hours       = duration/60;
minutes         = duration-hours*60;

// Check for possible discount
// Get number of bookings
int counter = update > 0 ? -1 : 0;
for ( Booking b : Database.getBookings() ) {
    if ( b.getCustomerId() == userId ) {
        counter++;
    }
}

// Check for discount
double discount = 0;
if ( counter > 20 ) {
    System.out.println("\nCongratulations! As a Platium member,
        you get 20% off this booking!");
    discount = 0.2;
} else if ( counter > 15 ) {
    System.out.println("\nCongratulations! As a Gold member, you
        get 15% off this booking!");
    discount = 0.15;
} else if ( counter > 10 ) {
    System.out.println("\nCongratulations! As a Silver member,
        you get 10% off this booking!");
    discount = 0.1;
} else if ( counter > 5 ) {
    System.out.println("\nCongratulations! As a Bronze member,
```

```
                you get 5% off this booking!");
            discount = 0.05;
        }

        // Calculate price and apply possible discount
        price = (int) (selectedCar.getRate()/60*duration*(1-discount)*
            100);

        // Check if time slot is available
        check = true; // reset
        for ( int i = 0; i<timeSlots.length; i++ ) {

            if ( timeSlots[i] == false && i*30 >= startTimestamp && i*30
                <= returnTimestamp ) {
                check = false;
            }
        }
        if ( !check ) {
            System.out.println("\nSorry, your requested time slots are
                already taken.");
        } else {
            System.out.printf("\nSelected duration: %2s:%2sh", hours,
                minutes);
        }
    } while ( hours > 23 || minutes >= 60 || duration < 30 || duration >
        4*60 || !check );
    System.out.printf("\nCalculated price "+price/100.0);

    if ( update == 0 ) {
        // Regular case, create new Booking
        booking = new Booking(selectedCar.getId(), price, userId, date,
            startTimestamp, returnTimestamp);
    } else {
        // If this method is called with an update ID, the Booking
            instance gets updated
        booking = Booking.getById(update);
        booking.setPrice(price);
        booking.setStartTimestamp(startTimestamp);
        booking.setReturnTimestamp(returnTimestamp);
    }

    return booking;
}

/**
 * Helper method to print time slots
 *
 * @param timeSlots List of time slots as booleans
 */
private static void formatTimeSlots(boolean[] timeSlots) {
    for ( int i = 0; i<timeSlots.length; i++ ) {
        int hours = i*30/60;
        int minutes = i*30-hours*60;
        String time = String.format("%2d:%2d", hours, minutes);
        String availability = timeSlots[i] ? "Available" : "--";
        System.out.printf("\t  | %-5s | %-10s |\n", time, availability);
    }
}

/**
```

```
     * UI method to make a payment
     *
     * @param booking Booking that will be paid
     * @return The created Payment instance
     */
    public static Payment makeAPayment(Booking booking) {
        System.out.printf("\n\n\tHow would you like to pay?\n");
        System.out.printf("\t  | %-3s | %-15s |\n", 1, "Credit Card");
        System.out.printf("\t  | %-3s | %-15s |\n", 2, "Mobile Pay");
        System.out.printf("\t  | %-3s | %-15s |\n", 3, "Cash");
        int selectPayment = input.nextInt();

        Payment payment = null;
        switch ( selectPayment ) {
        case 1:
            System.out.println("You chose credit card.");
            payment = new CreditCard(booking.getBookingRef(), booking.
                getPrice());
            break;
        case 2:
            System.out.println("You chose mobile pay.");
            payment = new MobilePay(booking.getBookingRef(), booking.
                getPrice());
            break;
        case 3:
            System.out.println("You chose cash.");
            payment = new Cash(booking.getBookingRef(), booking.getPrice());

            break;
        }
        System.out.print(payment.toString());
        Database.addPayment(payment);
        payment.toDB();

        return payment;
    }


    /**
     * UI method to list a user's bookings.
     */
    public static void myBookings() {
        User activeUser = Auth.getActiveUser();
        List<Booking> myBookings = new ArrayList<Booking>();

        for ( Booking booking: Database.getBookings() ) {
            if ( booking.getCustomerId() == activeUser.getId() ) {
                myBookings.add(booking);
            }
        }

        if ( myBookings.size() == 0 ) {
            System.out.println("You have no bookings yet.");
        } else {
            System.out.print(Booking.toString(myBookings));
        }

        System.out.printf("\n\n\tNEXT\n");
        System.out.printf("\t  | %-3s | %-15s |\n", 1, "Edit Booking");
        System.out.printf("\t  | %-3s | %-15s |\n", 2, "Delete Booking");
```

```
        System.out.printf("\t  | %-3s | %-15s |\n", 3, "Exit");
        int choice = input.nextInt();

        int id, exclude;
        switch ( choice ) {
        case 1:
            System.out.printf("\nType in the ID of the booking you want to
                edit: ");
            exclude = input.nextInt();
            // Include the exclude parameter as the ID of the Booking which
                will be excluded
            // in order to edit the Booking made.
            Booking edited = UserInterface.makeABooking(exclude);
            Boolean editSuccess = edited.update();
            if ( editSuccess ) {
                System.out.printf("\nThe item has been successfully edited."
                    );
            }
            break;
        case 2:
            System.out.printf("\nType in the ID of the booking you want to
                delete: ");
            id = input.nextInt();
            System.out.printf("\nAre you sure you want to delete this
                Booking? This action can not be reserved.\n");
            System.out.printf("\t  | %-3s | %-5s |\n", 1, "Yes");
            System.out.printf("\t  | %-3s | %-5s |\n", 2, "Abort");
            int confirm = input.nextInt();
            if ( confirm == 1 ) {
                Booking booking = Booking.getById(id);
                Boolean deleteSuccess = booking.delete();
                if ( deleteSuccess ) {
                    System.out.printf("\nThe item has been successfully
                        deleted.");
                } else {
                    System.out.printf("\nThere was an error deleting the
                        item.");
                }
            }
            break;
        }
    }

    /**
     * UI method to print a user's account information
     */
    public static void myAccount() {
        System.out.print(Auth.getActiveUser().toString());
    }

    /**
     * UI method to register a car
     */
    public static void registerCar() {
        input = new Scanner(System.in);
        System.out.printf("\n\nREGISTER CAR\n");
        System.out.print("Name of the car: ");
        String name = input.nextLine();

        System.out.printf("\n\n\tTYPE\n");
```

```
System.out.printf("\t  | %-3s | %-15s |\n", 1, "Sedan");
System.out.printf("\t  | %-3s | %-15s |\n", 2, "Convertible");
System.out.printf("\t  | %-3s | %-15s |\n", 3, "Sports Car");
System.out.printf("\t  | %-3s | %-15s |\n", 4, "SUV");
System.out.printf("\t  | %-3s | %-15s |\n", 5, "Other");
int typePrompt = input.nextInt();

String type;
switch ( typePrompt ) {
case 1:
    type = "Sedan";
    break;
case 2:
    type = "Convertible";
    break;
case 3:
    type = "Sports Car";
    break;
case 4:
    type = "SUV";
    break;
default:
    type = "Other";
}


System.out.print ("Brand: ");
String brand = input.next();

System.out.printf("\n\n\tTRANSMISSION\n");
System.out.printf("\t  | %-3s | %-15s |\n", 1, "Manual");
System.out.printf("\t  | %-3s | %-15s |\n", 2, "Automatic");
int transmissionPrompt = input.nextInt();

String transmission = null;
switch ( transmissionPrompt ) {
    case 1: transmission = "Manual";
            break;
    case 2: transmission = "Automatic";
            break;
}

System.out.print("Amount of seats: ");
int seats = input.nextInt();

System.out.print("Hourly rate: ");
double rate = input.nextDouble();

System.out.print("Description: ");
String description = input.next();

System.out.print("Location: ");
String location = input.next();

User activeUser = Auth.getActiveUser();
int owner = activeUser.getId();
Car newCar = new Car(owner, name, type, brand, transmission, seats,
    rate, description, location);
newCar.toDB();
Database.addCar(newCar);
```

```
        System.out.print(newCar.toString());
    }

    /**
     * UI method to search a car with a recursive filter algorithm.
     */
    public static void searchCar() {
        List<Car> cars = Database.getCars();
        boolean cont = true;

        boolean showType = true, showBrand = true, showTransmission = true,
            showSeats = true, showRate = true;

        System.out.println("These are all cars available:");
        System.out.print(Car.toString("ALL CARS", cars));

        do {
            System.out.printf("\n\n\tREFINE SEARCH\n");
            System.out.printf("\t  | %-3s | %-20s |\n", 0, "Exit");

            if ( showType ) {
                System.out.printf("\t  | %-3s | %-20s |\n", 1, "By type");
            }
            if ( showBrand ) {
                System.out.printf("\t  | %-3s | %-20s |\n", 2, "By brand");
            }
            if ( showTransmission ) {
                System.out.printf("\t  | %-3s | %-20s |\n", 3, "By
                    transmission");
            }
            if ( showSeats ) {
                System.out.printf("\t  | %-3s | %-20s |\n", 4, "By seats");
            }
            if ( showRate ) {
                System.out.printf("\t  | %-3s | %-20s |\n", 5, "By rate per
                    hour");
            }
            int typePrompt = input.nextInt();
            input.nextLine();

            if ( typePrompt != 0 && typePrompt < 6 ) {
                String key = null;
                switch ( typePrompt ) {
                    case 1: key = "type";
                            showType = false;
                            break;
                    case 2: key = "brand";
                            showBrand = false;
                            break;
                    case 3: key = "transmission";
                            showTransmission = false;
                            break;
                    case 4: key = "seats";
                            showSeats = false;
                            break;
                    case 5: key = "rate";
                            showRate = false;
                            break;
                }
```

```
            String value = null;
            if ( key == "seats" ) {
                System.out.println("Please type in the number of seats:
                    ");
                value = input.nextLine();
            } else if ( key == "rate" ) {
                System.out.println("Please type in the maximal rate per
                    hours: ");
                value = input.nextLine();
            } else if ( key.equals("type") ) {
                int choice;
                do {
                    System.out.printf("\n\n\tAVAILABLE TYPES\n");
                    System.out.printf("\t  | %-3s | %-12s |\n", 1, "Show
                        All");
                    System.out.printf("\t  | %-3s | %-12s |\n", 2,
                        "Sedan");
                    System.out.printf("\t  | %-3s | %-12s |\n", 3,
                        "Convertible");
                    System.out.printf("\t  | %-3s | %-12s |\n", 4,
                        "Sports Car");
                    System.out.printf("\t  | %-3s | %-12s |\n", 5, "SUV"
                        );
                    choice = input.nextInt();
                } while ( choice <= 0 || choice > 5 );

                switch (choice) {
                case 1:
                    value = null;
                    break;
                case 2:
                    value = "Sedan";
                    break;
                case 3:
                    value = "Convertible";
                    break;
                case 4:
                    value = "Sports Car";
                    break;
                case 5:
                    value = "SUV";
                    break;
                }
            } else {
                System.out.println("Please type in the "+key+": ");
                value = input.nextLine();
            }

            cars = Car.filter(cars, key, value);
            System.out.print(Car.toString("FILTERED CARS", cars));
        } else {
            cont = false;
        }
    } while ( cont );
}

/**
 * UI method to list a all cars registered by a user
 */
```

```
    public static void myCars() {
        int id = Auth.getActiveUser().getId();
        String value        = Integer.toString(id);
        List<Car> myCars    = Car.filter(Database.getCars(), "owner", value)
            ;

        System.out.print(Car.toString("MY CARS", myCars));

        System.out.printf("\n\n\tCONTINUE\n");
        System.out.printf("\t  | %-3s | %-20s |\n", 0, "Exit");
        System.out.printf("\t  | %-3s | %-20s |\n", 1, "See availability");
        int contPrompt = input.nextInt();

        if ( contPrompt == 1 ) {
            Car selectedCar;
            do {
                System.out.printf("\nType in the ID of the car you want to
                    check: ");
                int carChoice = input.nextInt();
                selectedCar = Car.getById(carChoice);
            } while ( selectedCar == null );
            input.nextLine();

            System.out.println("You have chosen:");
            System.out.print(Car.toString(selectedCar));

            String date;
            do {
                System.out.printf("\nWhich day do you want to see for the
                    selected car? (DD.MM.YYYY): ");
                date = input.nextLine();
            } while( !date.matches("\\d{2}.\\d{2}.\\d{4}") );

            System.out.println("See here all available time slots:");
            System.out.printf("\n\n\t"+date+"\n");

            // Print all time slots
            boolean[] timeSlots = Booking.getTimeSlots(selectedCar.getId(),
                date, 0);
            UserInterface.formatTimeSlots(timeSlots);
        }
    }

    /**
     * UI method to manage the database.
     */
    public static void manageDatebase() {
        System.out.printf("\n\n\tCONTINUE\n");
        System.out.printf("\t  | %-3s | %-20s |\n", 0, "Exit");
        System.out.printf("\t  | %-3s | %-20s |\n", 1, "Cars report");
        System.out.printf("\t  | %-3s | %-20s |\n", 2, "Bookings report");
        System.out.printf("\t  | %-3s | %-20s |\n", 3, "Payments report");
        System.out.printf("\t  | %-3s | %-20s |\n", 4, "Users report");
        int contPrompt = input.nextInt();

        switch ( contPrompt ) {
        case 1:
            System.out.print(Car.report());

            System.out.printf("\n\n\tCONTINUE\n");
```

```
            System.out.printf("\t  | %-3s | %-20s |\n", 0, "Exit");
            System.out.printf("\t  | %-3s | %-20s |\n", 1, "Delete Car");
            int contPrompt2 = input.nextInt();

            if ( contPrompt2 == 1 ) {
                Car selectedCar = null;
                do {
                    System.out.printf("\nType in the ID of the car you want
                        to delete: ");
                    int carChoice = input.nextInt();

                    System.out.printf("\nAre you sure you want to delete
                        this car? This action can not be reserved.\n");
                    System.out.printf("\t  | %-3s | %-5s |\n", 1, "Yes");
                    System.out.printf("\t  | %-3s | %-5s |\n", 2, "Abort");
                    int sure = input.nextInt();
                    if ( sure == 1 ) {
                        selectedCar = Car.getById(carChoice);
                        Boolean success = selectedCar.delete();
                        if ( success ) {
                            System.out.printf("\nThe item has been
                                successfully deleted.");
                        } else {
                            System.out.printf("\nThere was an error deleting
                                the item.");
                        }
                    } else if ( sure == 2 ) {
                        // Abort
                    }
                } while ( selectedCar == null );
            } break;

        case 2:
            System.out.print(Booking.report());
            break;

        case 3:
            System.out.print(Payment.report());
            break;

        case 4:
            System.out.print(User.report());
            break;
        }
    }

    /**
     * Helper UI method to check for a possible discount
     */
    public static void checkDiscount() {
        User activeUser = Auth.getActiveUser();
        int userId = activeUser.getId();

        // Get number of bookings
        int counter = 0;
        for ( Booking b : Database.getBookings() ) {
            if ( b.getCustomerId() == userId ) {
                counter++;
            }
        }
```

```
        if ( counter == 20 ) {
            System.out.println("\nCongratulations! As a Platium member, you
                now get 20% off your next bookings!");
        } else if ( counter == 15 ) {
            System.out.println("\nCongratulations! As a Gold member, you now
                get 15% off your next bookings!");
        } else if ( counter == 10 ) {
            System.out.println("\nCongratulations! As a Silver member, you
                now get 10% off your next bookings!");
        } else if ( counter == 5 ) {
            System.out.println("\nCongratulations! As a Bronze member, you
                now get 5% off your next bookings!");
        }
    }

    /**
     * UI method to enter a user's role
     * @return User role
     */
    private static String inputUserRole() {
        boolean error = false;
        String role = null;
        int rolePrompt;

        do {
            System.out.println("As what role do you want to be reigstered");
            System.out.printf("\t  | %-3s | %-15s |\n", 1, "Customer");
            System.out.printf("\t  | %-3s | %-15s |\n", 2, "Car Owner");
            System.out.printf("\t  | %-3s | %-15s |\n", 3, "Administrator");
            rolePrompt = input.nextInt();
            input.nextLine();

            switch ( rolePrompt ) {
                case 1: role = "customer";
                        break;
                case 2: role = "car owner";
                        break;
                case 3: System.out.println("Please type in the sudo password
                    to continue: ");
                        String sudoPassword = input.nextLine();
                        if ( sudoPassword.equals("supersecret") ) {
                            role = "administrator";
                        } else {
                            System.out.printf("\n\nSorry, the password is
                                incorrect.\n");
                            error = true;
                        }
                        break;
                default: role = "customer";
            }
        } while ( rolePrompt < 0 || rolePrompt > 3 || error == true );

        return role;
    }

    /**
     * UI method to input user's first name
     * @return First name
     */
```

```
    private static String inputFirstName() {
        System.out.println("What is your first name?");

        String firstname = input.nextLine();
        return firstname;
    }

    /**
     * UI method to input user's last name
     * @return Last name
     */
    private static String inputLastName() {
        System.out.println("What is your last name?");

        String lastname = input.nextLine();
        return lastname;
    }

    /**
     * UI method to input user's street name
     *
     * @param firstname First name of the user to greet her/him
     * @return Street name
     */
    private static String inputStreetname(String firstname) {
        // Get address
        System.out.printf("Thank you %s. Please now enter the name of your
            street\n", firstname);
        String streetname = input.nextLine();
        return streetname;
    }

    /**
     * UI method to input user's house number
     *
     * @param firstname First name of the user to greet her/him
     * @return House number
     */
    private static String inputHouseNumber(String firstname) {
        System.out.printf("Thank you %s. Please now enter your house
            number\n", firstname);
        String houseNumber = input.nextLine();
        return houseNumber;
    }

    /**
     * UI method to input user's post code
     *
     * @param firstname First name of the user to greet her/him
     * @return Post code
     */
    private static int inputPostcode(String firstname) {
        System.out.printf("Thanks %s. Please enter your postcode\n",
            firstname);
        int postcode = input.nextInt();
        // https://stackoverflow.com/questions/13102045/scanner-is-skipping-
            nextline-after-using-next-nextint-or-other-nextfoo
        input.nextLine();

        // Check if post code is valid for the København region
```

```
        while (postcode > 9999 || postcode < 1000 ) {
            System.out.println("Whoops! Please make sure you entered a valid
                4-digit postcode. Try again:");
            postcode = input.nextInt();
        }
        while (postcode >= 2500 || postcode <= 1000) {
            System.out.println("Sorry! We only lend  out bikes within the
                main Copenhagen area!");
        }

        return postcode;
    }

    /**
     * UI method to input user's date of birth
     *
     * @return Date of birth
     */
    private static String inputDOB() {
        System.out.println("Ok, next, please enter your birthday in the
            format DD.MM.YYYY");
        boolean err = true;
        String dob = null;
        while (err) {
            dob = input.nextLine();

            int firstDot    = dob.indexOf('.');
            int secondDot   = dob.indexOf('.', 5);

            while (firstDot < 0 || secondDot < 0) {
                System.out.println("Whoops! Please make sure you entered a
                    valid birthday in the format DD.MM.YYYY. Try again:");
                // I repeat myself :(
                dob = input.nextLine();
                firstDot = dob.indexOf('.');
                secondDot = dob.indexOf('.', 5);
            }

            int day         = Integer.parseInt(dob.substring(0, firstDot));
            int month   = Integer.parseInt(dob.substring(++firstDot,
                secondDot));
            int year    = Integer.parseInt(dob.substring(++secondDot));

            // Validate DOB
            if (day > 31 || month > 12 || year > 2017 || year < 1900) {
                System.out.println("The date you entered is invalid, please
                    try again:");
                err = true;
            } else {
                err = false;
            }
        }

        return dob;
    }


    /**
     * UI method to input user's telephone number.
     * @return Telephone number
```

```
     */
    private static String inputTelephone() {
        // Get telephone
        boolean err = true;
        String telephone = null;
        while (err) {
            System.out.println("Please now enter your 8-digit telephone
                number:");
            telephone = input.nextLine();

            // Check format
            if (telephone.length() != 8) {
                System.out.println("The number you entered is invalid.");
                err = true;
            } else {
                err = false;
            }
        }

        return telephone;
    }

    /**
     * UI method to input user's CPR number
     *
     * @return CPR
     */
    private static String inputCPR() {
        boolean err     = true;
        String cpr = null;
        while (err) {
            System.out.println("Please enter your 10-digit CPR number in the
                following format xxxxxx-xxxx:");
            cpr = input.nextLine();

            // Checking CPR if 10-digit
            char dash = cpr.charAt(6);
            if (cpr.length() != 11 || dash != '-' ) {
                System.out.println("There was an error!");
                err = true;
            } else {
                err = false;
            }
        }

        return cpr;
    }

    /**
     * Input credit card number method
     *
     * @return validated credit card number
     */
    public static String inputCreditCardNumber() {
        input = new Scanner(System.in);
        String cardNumber;

        do {
            System.out.print("Enter your card number: ");
            cardNumber = input.nextLine();
```

```
            // DEBUG
            // cardNumber = "4916722136319146";

            if ( !CreditCard.luhnCheck(cardNumber) ) {
                System.out.println("The entered number is incorrect. Please
                    try again.");
            }
        } while( !CreditCard.luhnCheck(cardNumber) );

        return cardNumber;
    }

    /**
     * Input expiry date method
     *
     * @return validated expiry date
     */
    public static String inputCreditCardExpiryDate() {
        input = new Scanner(System.in);
        String expDate;

        int month = 0, year = 0;
        LocalDate now = LocalDate.now();
        int thisMonth = now.getMonthValue();
        int thisYear = now.getYear();

        do {
            do {
                System.out.print("Enter your expiry date (MM/YYYY): ");
                expDate = input.nextLine();
            } while(!expDate.matches("\\d{2}/\\d{4}"));

            month   = Integer.parseInt(expDate.substring(0,2));
            year    = Integer.parseInt(expDate.substring(3,7));

            // Check if expiry date is in the future
        } while ( year < thisYear || (year == thisYear && month < thisMonth)
            );
        return expDate;
    }

    /**
     * Input security code method
     *
     * @return validated security code
     */
    public static String inputCreditCardSecurityCode() {
        input = new Scanner(System.in);
        String cvv;

        do {
            System.out.print("Enter your CVV (3 digits): ");
            cvv = input.nextLine();
        } while(cvv.length() != 3);

        return cvv;
    }

    // Input mobile number
    public static String inputMobilePayNumber() {
```

```
            input = new Scanner(System.in);
            String telNumber;

            do {
                System.out.print("Enter your telephone number:");
                telNumber = input.nextLine();
            } while(telNumber.length() == 0);

            return telNumber;
        }
    }
```

```
package data;

import model.User;
import model.Car;
import model.Cash;
import model.CreditCard;
import model.MobilePay;
import model.Booking;
import model.Payment;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * This class handles read and write activities related to the external CSV
    files.
 *
 * Setting a custom delimiter (;), CSV files are easier to manipulate via
    external tools
 * (e.g. MS Excel) and allow for white space in the values.
 *
 */
public class Database {
    private static List<User> users             = new ArrayList<User>();
    private static List<Car> cars           = new ArrayList<Car>();
    private static List<Booking> bookings   = new ArrayList<Booking>();
    private static List<Payment> payments   = new ArrayList<Payment>();

    public static List<User> getUsers() {
        return users;
    }

    public static List<Car> getCars() {
        return cars;
    }

    public static List<Booking> getBookings() {
        return bookings;
    }

    public static List<Payment> getPayments() {
        return payments;
    }

    public static void addUser(User newUser) {
        users.add(newUser);
    }

    public static void addCar(Car newCar) {
        cars.add(newCar);
    }
```

```
public static void addBooking(Booking newBooking) {
    bookings.add(newBooking);
}

public static void addPayment(Payment newPayment) {
    payments.add(newPayment);
}


/**
 * Write a line to a file
 *
 * @param table The name of the table / file to write to
 * @param line The content of the new line
 */
public static Boolean write(String table, String line) {
    try {
        BufferedWriter writer = new BufferedWriter(new
            OutputStreamWriter(new FileOutputStream("src/
            data/"+table+".csv", true), "utf-8"));
        PrintWriter out = new PrintWriter(writer);
        out.println(line);
        out.close();
        return true;
    } catch (IOException ex) {
        return false;
    }
}

/**
 * Truncate (= clear) table method
 *
 * @param table Name of the table to be truncated
 */
public static void truncate(String table) {
    if ( table == "bookings" ) {
        bookings = new ArrayList<Booking>();
    } else if ( table == "cars" ) {
        cars = new ArrayList<Car>();
    } else if ( table == "users" ) {
        users = new ArrayList<User>();
    } else if ( table == "payment" ) {
        payments = new ArrayList<Payment>();
    }
}

/**
 * Delete a object from a table by the object's ID
 *
 * @param table The table which will be deleted from
 * @param searchId The ID of the object that will be deleted
 */
// return boolean, print in UI
public static Boolean deleteById(String table, int searchId) {
    return Database.manipulate(table, searchId, null);
}

/**
 * Edit a object in a table by the object's ID
 *
```

```
     * @param table The table which will be edited
     * @param searchId The ID of the object that will be edited
     * @param newLine
     */
    // see above
    public static Boolean editById(String table, int searchId, String
        newLine) {
        return Database.manipulate(table, searchId, newLine);
    }

    /**
     * Method to manipulate (delete and edit) a table.
     * Copies parts of the existing table to a new temporary file, then
         deletes the
     * old file and renames the new one.
     *
     * Critical problem with this approach: When multiple manipulations are
         executed
     * within a short period of time, this might lead to data loss because
         the reading
     * and writing may take longer than the copying and renaming.
     *
     * NOT PRODUCTION READY!
     *
     * @param table The name of the table that will be manipulated
     * @param searchId The ID of the object that will be manipulated
     * @param newLine The content of the new line
     * @return A boolean variable indication success (true) or failure
         (false)
     */
    public static Boolean manipulate(String table, int searchId, String
        newLine) {
        boolean success = false;

        try {
            String fileName     = "src/data/"+table+".csv";
            File inputFile  = new File(fileName);
            File tempFile   = new File(table+"Temp.csv");

            BufferedWriter writer = new BufferedWriter(new FileWriter
                (tempFile));

            // Read File, set delimiter to read CSV correctly
            @SuppressWarnings("resource")
            Scanner readFile    = new Scanner(inputFile).useDelimiter("\\s*;
                \\s*");
            // Get header of that table
            String header       = readFile.nextLine();
            // Write that header to the new file
            writer.write(header + System.getProperty("line.separator"));

            while ( readFile.hasNext() ) {
                int id      = Integer.parseInt(readFile.next());
                String line     = id + readFile.nextLine();

                if ( searchId != id ) {
                    writer.write(line + System.getProperty("line.separator")
                        );
                } else if ( newLine != null && !newLine.isEmpty() ) {
                    writer.write(newLine +
```

```
                    System.getProperty("line.separator"));
            }
        }

        writer.close();
        inputFile.delete();
        success = tempFile.renameTo(new File(fileName));
        Database.truncate(table);
        Database.read(table);
    } catch (IOException ex) {
        success = false;
    }

    return success;
}

/**
 * Read a table and create respective instances.
 *
 * This approach is a little "hacky". The last item of a line has to be
     cropped to be
 * read correctly.
 *
 * @param table Name of the table which will be read
 */
public static Boolean read(String table) {
    Boolean success = false;

    try {
        File file = new File("src/data/"+table+".csv");
        @SuppressWarnings("resource")
        Scanner readFile = new Scanner(file).useDelimiter("\\s*;\\s*");
        readFile.nextLine();

        while ( readFile.hasNext() ) {
            if ( table.equals("users") ) {
                int id              = Integer.parseInt(readFile.next());
                String role            = readFile.next();
                String name            = readFile.next();
                String streetname   = readFile.next();
                String housenumber  = readFile.next();
                int postcode        = Integer.parseInt(readFile.next());
                String dob          = readFile.next();
                String telephone    = readFile.next();
                String cpr          = readFile.next();
                String username         = readFile.next();
                String password         = readFile.nextLine();
                password            = password.substring(1, password.
                    length()-1);


                /*
                 * Debug
                 */
                //System.out.println(id+" "+name+" "+streetname+"
                    "+housenumber+" "+postcode+" "+dob+" "+telephone+"
                    "+cpr+" "+username+" "+password+".");

                User newUser = new User(id, role, name, streetname,
                    housenumber, postcode, dob, telephone, cpr, username
```

```
            , password);
        users.add(newUser);
    }
    else if ( table.equals("cars") ) {
        int id              = Integer.parseInt(readFile.next());
        int owner           = Integer.parseInt(readFile.next());
        String name             = readFile.next();
        String type             = readFile.next();
        String brand        = readFile.next();
        String transmission     = readFile.next();
        int seats           = Integer.parseInt(readFile.next());
        double rate             = Double.parseDouble(readFile.
            next());
        String description  = readFile.next();
        String location         = readFile.nextLine();
        location            = location.substring(1, location.
            length()-1);

        /*
         * Debug
         */
        //System.out.println(id+" "+name+" "+type+" "+brand+"
            "+transmission+" "+seats+" "+rate+" "+description);

        // Make users
        Car newCar = new Car(id, owner, name, type, brand,
            transmission, seats, rate, description, location);
        cars.add(newCar);
    }
    else if ( table.equals("bookings") ) {
        int id                      = Integer.parseInt(readFile.
            next());
        String date                     = readFile.next();
        int carId                   = Integer.parseInt(readFile.
            next());
        int price                   = Integer.parseInt(readFile.
            next());
        int userId                  = Integer.parseInt(readFile.
            next());
        String rentalDate           = readFile.next();
        int rentalStartTimestamp    = Integer.parseInt(readFile.
            next());
        int rentalEndTimestamp      = Integer.parseInt(readFile.
            next());
        String bookingRef           = readFile.nextLine();
        bookingRef                  = bookingRef.substring(1,
            bookingRef.length()-1);

        /*
         * Debug
         */
        //System.out.println(id+" "+name+" "+type+" "+brand+"
            "+transmission+" "+seats+" "+rate+" "+description);

        // Make users
        Booking newBooking = new Booking(id, date, carId, price,
            userId, rentalDate, rentalStartTimestamp,
            rentalEndTimestamp, bookingRef);
        bookings.add(newBooking);
    }
```

```
                else if ( table.equals("payments") ) {
                    int id                      = Integer.parseInt(readFile.
                        next());
                    String bookingRef           = readFile.next();
                    int amount                  = Integer.parseInt(readFile.
                        next());
                    String method               = readFile.next();

                    // Check for payment method
                    // Item must always contain 4 + 3 = 7 items
                    if ( method.equals("credit card") ) {
                        String number           = readFile.next();
                        String expiryDate       = readFile.next();
                        String securityCode     = readFile.nextLine();
                        securityCode = securityCode.substring(0,
                            securityCode.length()-1);

                        Payment newCreditCard = new CreditCard(id,
                            bookingRef, amount, number, expiryDate,
                            securityCode);
                        payments.add(newCreditCard);
                    } else if ( method.equals("cash") ) {
                        // Cannibalize remaining items
                        readFile.next();
                        readFile.next();
                        readFile.nextLine();

                        Payment newCash = new Cash(id, bookingRef, amount);
                        payments.add(newCash);
                    } else if ( method.equals("mobile pay") ) {
                        String mobilePhone = readFile.next();
                        // Cannibalize remaining items
                        readFile.next();
                        readFile.nextLine();

                        Payment newMobilePay = new MobilePay(id, bookingRef,
                            amount, mobilePhone);
                        payments.add(newMobilePay);
                    }
                }
            }

            success = true;
        } catch (IOException ex) {
            return success = false;
        }

        return success;
    }
}
```

```
package model;

import data.Database;

import java.util.List;
import java.util.Scanner;

/**
 * User class
 */
public class User {
    private String name, streetname, dob, telephone, cpr, houseNumber, role;
    private String username, password;
    private int id, postcode;
    boolean err = true;
    char dash;
    private static int counter;

    Scanner input = new Scanner(System.in);

    /**
     * Database Constructor
     *
     * @param newId ID of the User
     * @param newRole Role of the User (administrator, customer, car owner)
     * @param newName Full name of the user
     * @param newStreetname Street name of the user's home address
     * @param newHouseNumber House number of the user's home address
     * @param newPostcode Post code of the user's home address
     * @param newDob Date of Birth of the user
     * @param newTelephone Telephone number of the user
     * @param newCpr CPR of the user
     * @param newUsername user name of the user
     * @param newPassword password of the user
     */
    public User(int newId, String newRole, String newName, String
        newStreetname, String newHouseNumber, int newPostcode,
            String newDob, String newTelephone, String newCpr, String
                newUsername, String newPassword) {
        id          = newId;
        role        = newRole;
        name        = newName;
        streetname  = newStreetname;
        houseNumber     = newHouseNumber;
        postcode    = newPostcode;
        dob             = newDob;
        telephone   = newTelephone;
        cpr             = newCpr;
        username    = newUsername;
        password    = newPassword;
        counter++;
    }

    /**
     * In-App Constructor
     *
     * @param newRole Role of the User (administrator, customer, car owner)
     * @param newName Full name of the user
     * @param newStreetname Street name of the user's home address
     * @param newHouseNumber House number of the user's home address
```

```
     * @param newPostcode Post code of the user's home address
     * @param newDob Date of Birth of the user
     * @param newTelephone Telephone number of the user
     * @param newCpr CPR of the user
     */
    public User (String newRole, String newFirstname, String newLastname,
        String newStreetname, String newHouseNumber, int newPostcode,
            String newDob, String newTelephone, String newCpr) {
        id          = ++counter;
        role        = newRole;
        name        = newFirstname+" "+newLastname;
        streetname  = newStreetname;
        houseNumber     = newHouseNumber;
        postcode    = newPostcode;
        dob             = newDob;
        telephone   = newTelephone;
        cpr             = newCpr;
        username    = makeUsername(newFirstname, newLastname);
        password    = makePassword(newFirstname, newLastname);
    }

    /**
     * Writes a user object (line) to the databse
     */
    public void toDB() {
        Database.write("users", this.id+";"+this.role+";"+this.name+";"+this
            .streetname+";"+this.houseNumber+";"+this.postcode+";"+

                    this.dob+";"+this.telephone+";"+this.cpr+";"+this.
                    username+";"+this.password+" ");
    }

    // Start getter
    public int getId() {
        return id;
    }

    public String getRole() {
        return role;
    }

    public String getName() {
        return name;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public int getID() {
        return id;
    }

    public String getAddress() {
        return streetname + " " + houseNumber + " " + postcode;
    }
```

```
public String getDOB() {
    return dob;
}

public String getTelephone() {
    return username;
}

public String getCPR() {
    return cpr;
}
// End getter

// Start setters
public void setStreetname(String newStreetname) {
    streetname = newStreetname;
}

public void setHouseNumber(String newHouseNumber) {
    houseNumber = newHouseNumber;
}

public void setPostcode(int newPostcode) {
    postcode = newPostcode;
}

public void setTelephone(String newTelephone) {
    telephone = newTelephone;
}

public void newPassword(String newPassword) {
    password = newPassword;
}
// End setters

/**
 * Validate credentials method
 *
 * @param testUsername User name that will be tested against this
     instance's password
 * @param testPassword Password that will be tested against this
     instance's password
 * @return
 */
public Boolean checkCredentials(String testUsername, String testPassword
    ) {
    if ( username.equals(testUsername) && password.equals(testPassword)
        ) {
        return true;
    } else {
        return false;
    }
}

/**
 * Create user name from first letter of the first name and the first
     three letters from
 * the last name.
 *
```

```
     * Critical Problem: Users with similar names might get the same user
         name.
     *
     * To Do: Make recursive unique-validator
     *
     * @param firstname User's first name
     * @param lastname User's last name
     * @return User name
     */
    private String makeUsername(String firstname, String lastname) {
        String username = firstname.substring(0, 1).toLowerCase() + lastname
            .substring(0, 3).toLowerCase();
        System.out.printf("Your username is %s\n", username);
        return username;
    }

    /**
     * Create password from the first three letters of the last name and the
         last four numbers
     * of the CPR.
     *
     * Critical Problem: Knowing the last name and the CPR of a user enables
         hackers to get
     * access to foreign accounts.
     *
     * To Do: Make password less predictable (safer)
     *
     * @param firstname User's first name
     * @param lastname User's last name
     * @return String password
     */
    private String makePassword(String firstname, String lastname) {
        String password = lastname.substring(0, 3).toLowerCase() + cpr.
            substring(cpr.length() - 4).toLowerCase();
        System.out.printf("Your password is %s.\n", password);
        return password;
    }

    // Start print methods
    public static String printTableHeader() {
        return String.format("\t  | %-2s | %-15s | %-20s | %-11s | %-8s |\n"
            ,
                "ID", "Role", "Name", "CPR", "Username");
    }

    public static String printLine(User user) {
        return String.format("\t  | %-2s | %-15s | %-20s | %-11s | %-8s |\n"
            ,
                user.getId(), user.getRole(), user.getName(), user.getCPR(),
                    user.getUsername());
    }

    public static String toString(List<User> users) {
        String out = String.format("\n\n\tUSERS:\n");
        out += printTableHeader();
        for ( User user : users ) {
            out += printLine(user);
        }
        return out;
    }
```

```
    public static String toString(User user) {
        String out = String.format("\n\n\tACCOUNT:\n");
        out += printTableHeader();
        out += printLine(user);
        return out;
    }

    public String toString() {
        String out = String.format("\n\n\tUSER:\n");
        out += printTableHeader();
        out += printLine(this);
        return out;
    }

    public static String report() {
        return toString(Database.getUsers());
    }
    // End print methods
}
```

```java
package model;

import data.Database;

import java.util.List;

/**
 * This class is the parent class for the three payment methods, namely
     CreditCard,
 * Cash and MobilePay.
 *
 */
public class Payment {
    private String bookingRef, method;
    private int amount, id;
    private static int counter;

    /**
     * Database Constructor
     *
     * @param newId ID of the payment
     * @param newBookingRef Reference to a (unique) Booking
     * @param newAmount Amount of the payment
     * @param newMethod Method of the payment
     */
    public Payment(int newId, String newBookingRef, int newAmount, String
        newMethod) {
        id              = newId;
        bookingRef      = newBookingRef;
        amount          = newAmount;
        method          = newMethod;
        counter++;
    }

    /**
     * In-App Constructor
     *
     * @param newBookingRef Reference to a (unique) Booking
     * @param newAmount Amount of the payment
     */
    public Payment(String newBookingRef, int newAmount) {
        id              = ++counter;
        bookingRef      = newBookingRef;
        amount          = newAmount;
        method          = null;
    }

    /**
     * Write a line to the database
     */
    public void toDB() {
        Database.write("payments", this.id+";"+this.bookingRef+";"+this.
            amount+";"+this.method+";;; ");
    }

    // Start getter
    public int getId() {
        return this.id;
    }
```

```
    public String getBookingRef() {
        return this.bookingRef;
    }

    public int getAmount() {
        return this.amount;
    }

    public String getMethod() {
        return this.method;
    }
    // End getter

    /**
     * Set payment method method
     * @param paymentMethod Payment Method as String (credit card, cash, or
         mobile pay)
     */
    public void setMethod(String paymentMethod) {
        this.method = paymentMethod;
    }

    // Start print methods
    public static String printTableHeader() {
        return String.format("\t  | %-18s | %-8s | %-15s | %-9s | %-10s |\n"
            ,
                "Booking Reference", "Amount", "Payment Method", "Last Four"
                    , "Mobile Phone");
    }

    public static String printLine(Payment payment) {
        return String.format("\t  | %-18s | %-8s |\n",
            payment.getBookingRef(), payment.getAmount()/100.0);
    }

    public String printLine() {
        return String.format("\t  | %-18s | %-8s |\n",
            this.getBookingRef(), this.getAmount()/100.0);
    }

    public static String toString(List<Payment> payments) {
        String out = String.format("\n\n\tPAYMENTS:\n");
        out += printTableHeader();
        for ( Payment payment : payments ) {
            out += printLine(payment);
        }
        return out;
    }

    public String toString() {
        String out = String.format("\n\n\tPAYMENT:\n");
        out += printTableHeader();
        out += printLine(this);
        return out;
    }
    // End print methods

    /**
     * Report methods printing all payments in the database.
     */
```

```
    public static String report() {
        String out = String.format("\n\n\tPAYMENTS:\n");
        out += printTableHeader();
        for ( Payment payment : Database.getPayments() ) {
            out += payment.printLine();
        }
        return out;
    }
}
```

```java
package model;

import data.Database;
import view.UserInterface;

/**
 * This class describes a payment method and thus extends the Payment Class.
 * Added variables is (String) mobile phone
 */
public class MobilePay extends Payment {
    String mobilePhone;

    /**
     * Database Constructor
     *
     * @param newId Id of the payment
     * @param newBookingRef Reference to a (unique) booking of the payment
     * @param newAmount Amount of the payment
     * @param newMobilePhone Mobile phone number of the payment / MP account
     */
    public MobilePay(int newId, String newBookingRef, int newAmount, String
        newMobilePhone) {
        super(newId, newBookingRef, newAmount, "mobile pay");

        mobilePhone = newMobilePhone;
        super.setMethod("mobile pay");
    }

    /**
     * In-App Constructor
     *
     * @param newBookingRef Reference to a (unique) booking of the payment
     * @param newAmount Amount of the payment
     */
    public MobilePay(String newBookingRef, int newAmount) {
        super(newBookingRef, newAmount);

        mobilePhone = UserInterface.inputMobilePayNumber();
        super.setMethod("mobile pay");
    }

    /**
     * Write a line to the database
     */
    public void toDB() {
        Database.write("payments", super.getId()+";"+super.getBookingRef()
            +";"+super.getAmount()+";"+super.getMethod()+";"+this.
            mobilePhone+";; ");
    }

    // Get mobile phone
    public String getMobilePhone() {
        return this.mobilePhone;
    }

    // Start print methods
    public static String printTableHeader() {
        return String.format("\t  | %-18s | %-8s | %-15s | %-12s |\n",
                "Booking Reference", "Amount", "Payment Method",
                    "MobilePhone");
```

```
    }

    public static String printLine(MobilePay mobilepay) {
        return String.format("\t  | %-18s | %-8s | %-15s | %-12s |\n",
                mobilepay.getBookingRef(), mobilepay.getAmount()/100.0,
                    "Mobile Pay", mobilepay.getMobilePhone());
    }

    public String printLine() {
        return String.format("\t  | %-18s | %-8s | %-15s | %-12s | %-12s |
            \n",
                super.getBookingRef(), super.getAmount()/100.0, "Mobile Pay"
                    , "--", this.getMobilePhone());
    }

    public String toString() {
        String out = ("\n\n\tCREDIT CARD:\n");
        out += printTableHeader();
        out += printLine(this);
        return out;
    }
    // End print methods
}
```

```
package model;

import view.UserInterface;
import data.Database;

/**
 * This class describes a payment method and thus extends the Payment Class.
 * Added variables are (String) number, expiry date and security code.
 */
public class CreditCard extends Payment {
    private String number, expiryDate, securityCode;

    /**
     * Database Constructor
     *
     * @param newId ID of the payment instance
     * @param newBookingRef Reference to a (unique) Booking
     * @param newAmount Amount of the payment
     * @param newNumber Credit card number
     * @param newExpiryDate Credit card expiry date
     * @param newSecurityCode Credit card security code
     */
    public CreditCard(int newId, String newBookingRef, int newAmount, String
        newNumber, String newExpiryDate, String newSecurityCode) {
        super(newId, newBookingRef, newAmount, "credit card");

        number          = newNumber;
        expiryDate      = newExpiryDate;
        securityCode        = newSecurityCode;
    }

    /**
     * In-App Constructor
     *
     * @param newBookingRef Reference to a (unique) Booking
     * @param newAmount Amount of the payment
     */
    public CreditCard(String newBookingRef, int newAmount) {
        super(newBookingRef, newAmount);

        number          = UserInterface.inputCreditCardNumber();
        expiryDate      = UserInterface.inputCreditCardExpiryDate();
        securityCode        = UserInterface.inputCreditCardSecurityCode();
        super.setMethod("credit card");
    }

    /**
     * Write a line to the database
     */
    public void toDB() {
        Database.write("payments", super.getId()+";"+super.getBookingRef()
            +";"+super.getAmount()+";"+super.getMethod()+";"+this.number+";"
            +this.expiryDate+";"+this.securityCode+" ");
    }

    // Start getter
    public String getNumber() {
        return this.number;
    }
```

```
public String getLastFour() {
    return this.number.substring(number.length()-4);
}

public String getExpiryDate() {
    return this.expiryDate;
}

public String getCvvCode() {
    return this.securityCode;
}
// End getter


/**
 * Implementation of the Luhn algorithm to validate credit card numbers.
 *
 * Adapted from: https://github.com/eix128/gnuc-credit-card-checker/blob
     /master/CCCheckerPro/src/com/gnuc/java/ccc/Luhn.java
 *
 * @param ccNumber Credit card number
 * @return boolean check variable.
 */
public static boolean luhnCheck(String ccNumber) {
        int sum = 0;
        boolean alternate = false;
        for (int i = ccNumber.length() - 1; i >= 0; i--) {
                int n = Integer.parseInt(ccNumber.substring(i, i + 1));
                if (alternate) {
                        n *= 2;
                        if (n > 9) {
                                n = (n % 10) + 1;
                        }
                }
                sum += n;
                alternate = !alternate;
        }
        return (sum % 10 == 0);
}

// Start print methods
public static String printTableHeader() {
    return String.format("\t  | %-18s | %-8s | %-15s | %-9s |\n",
            "Booking Reference", "Amount", "Payment Method", "Last Four"
                );
}

public static String printLine(CreditCard creditcard) {
    return String.format("\t  | %-18s | %-8s | %-15s | %-9s |\n",
            creditcard.getBookingRef(), creditcard.getAmount()/100.0,
                "Credit Card", "****"+creditcard.getLastFour());
}

public String printLine() {
    return String.format("\t  | %-18s | %-8s | %-15s | %-9s | %-12s |\n"
        ,
            super.getBookingRef(), super.getAmount()/100.0, "Credit
                Card", "****"+this.getLastFour(), "--");
}
```

```
    public String toString() {
        String out = String.format("\n\n\tCREDIT CARD:\n");
        out += printTableHeader();
        out += printLine(this);
        return out;
    }
    // End print methods
}
```

```java
package model;

import data.Database;

/**
 * This class describes a payment method and thus extends the Payment Class.
 * Added variables is (String) mobilePhone.
 */
public class Cash extends Payment {
    String mobilePhone;

    /**
     * Database constructor
     *
     * @param newId ID of the payment instance
     * @param newBookingRef Reference to a (unique) Booking
     * @param newAmount Amount of the payment
     */
    public Cash(int newId, String newBookingRef, int newAmount) {
        super(newId, newBookingRef, newAmount, "cash");
    }

    /**
     * In-App Constructor.
     *
     * @param newBookingRef Reference to a (unique) Booking
     * @param newAmount Amount of the payment
     */
    public Cash(String newBookingRef, int newAmount) {
        super(newBookingRef, newAmount);
        super.setMethod("cash");
    }

    /**
     * Write a line to the database
     */
    public void toDB() {
        Database.write("payments", super.getId()+";"+super.getBookingRef()
            +";"+super.getAmount()+";"+super.getMethod()+";;; ");
    }

    // Start print methods
    public static String printTableHeader() {
        return String.format("\t  | %-18s | %-8s | %-15s |\n",
                "Booking Reference", "Amount", "Payment Method");
    }

    public static String printLine(Cash cash) {
        return String.format("\t  | %-18s | %-8s | %-15s |\n",
                cash.getBookingRef(), cash.getAmount()/100.0, "Cash");
    }

    public String printLine() {
        return String.format("\t  | %-18s | %-8s | %-15s | %-9s | %-12s |\n"
            ,
                super.getBookingRef(), super.getAmount()/100.0, "Cash", "--"
                    , "--");
    }

    public String toString() {
```

```
            String out = String.format("\n\n\tCREDIT CARD:\n");
            out += printTableHeader();
            out += printLine(this);
            return out;
        }
        // End print methods
    }
```

```
package model;

import data.Database;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * This class functions as the Car model of the application. Car can be
    constructed two-fold:
 * <br>
 * <br>
 * <ul>
 * <li><strong>1: Construct the Car with all variables as parameters.</
    strong>
 * <br> This is needed for the initialization of Cars read from the database
    .</li>
 * <li><strong>2: Construct a Car with limited parameters.<strong>
 * <br> This is needed for the construction of new Cars within the
    application.</li>
 * </ul>
 */
public class Car {
    private int id, seats, owner;
    private String name, transmission, description, brand, type, location;
    private double rate;
    private static int counter;

    /**
     * Constructor with all variables available as parameters.
     *
     * @param newId ID of the car
     * @param newOwner User ID of the car's owner
     * @param newName Name of the car
     * @param newType Type of the car
     * @param newBrand Brand of the car
     * @param newTransmission Transmission of the car
     * @param newSeats Number of seats of the car
     * @param newRate Rate of the car
     * @param newDescription Description of the car
     * @param newLocation Location (street name) of the car
     */
    public Car(int newId, int newOwner, String newName, String newType,
        String newBrand, String newTransmission, int newSeats, double
        newRate, String newDescription, String newLocation) {
        id              = newId;
        owner           = newOwner;
        name            = newName;
        type            = newType;
        brand           = newBrand;
        transmission    = newTransmission;
        seats           = newSeats;
        rate            = newRate;
        description     = newDescription;
        location        = newLocation;
        counter++;
    }
```

```
    /**
     * Constructor with limited variables available as parameters.
     *
     * @param owner2 User ID of the car's owner
     * @param name2 Name of the car
     * @param type2 Type of the car
     * @param brand2 Brand of the car
     * @param transmission2 Transmission of the car
     * @param seats2 Number of seats of the car
     * @param rate2 Rate of the car
     * @param description2 Description of the car
     * @param location2 Location (street name) of the car
     */
    public Car(int owner2, String name2, String type2, String brand2, String
        transmission2, int seats2, double rate2, String description2, String
        location2) {
        id              = ++counter;
        owner           = owner2;
        name            = name2;
        type            = type2;
        brand           = brand2;
        transmission    = transmission2;
        seats           = seats2;
        rate            = rate2;
        description     = description2;
        location        = location2;
    }

    /**
     * Writes a query to add a Car to the database
     */
    public void toDB() {
        Database.write("cars",
            this.id+";"+this.owner+";"+this.name+";"+this.type+";"+this.
            brand+";"+this.transmission+";"+
                this.seats+";"+this.rate+";"+this.description+";"+this.
                    location+" ");
    }

    // Start getter methods
    public int getId() {
        return this.id;
    }

    public int getOwner() {
        return this.owner;
    }

    public String getName() {
        return this.name;
    }

    public String getType() {
        return this.type;
    }

    public int getSeats() {
        return this.seats;
    }
```

```
    public String getDescription() {
        return this.type;
    }

    public String getLocation() {
        return this.location;
    }

    public double getRate() {
        return this.rate;
    }

    public String getBrand() {
        return this.brand;
    }

    public String getTransmission() {
        return this.transmission;
    }
    // End getter methods

    /**
     * Delete this car from the database
     */
    public boolean delete() {
        return Database.deleteById("cars", this.id);
    }

    // Start print methods
    public static String printTableHeader() {
        return String.format("\t  | %-2s | %-10s | %-11s | %-20s | %-12s |
            %-15s | %-13s | %-12s | %-30s |\n",
                "ID", "Name", "Type", "Brand", "Transmission", "Seats",
                    "Rate per hour", "Description", "Location");
    }

    public static String printLine(Car car) {
        return String.format("\t  | %-2s | %-10s | %-11s | %-20s | %-12s |
            %-15s | %-13s | %-12s | %-30s |\n",
                car.getId(), car.getName(), car.getType(), car.getBrand(),
                    car.getTransmission(), car.getSeats(), car.getRate(),
                    car.getDescription(), car.getLocation());
    }

    /**
     * Static method to print multiple cars
     *
     * @param headline Headline the table has
     * @param cars List of cars that will be printed
     */
    public static String toString(String headline, List<Car> cars) {
        String out = String.format("\n\n\t"+headline+":\n");
        out += printTableHeader();
        for ( Car car : cars ) {
            out += printLine(car);
        }
        return out;
    }

    /**
```

```
     * Static method to print a single car
     * @param car
     */
    public static String toString(Car car) {
        String out = String.format("\n\n\tCAR:\n");
        out += printTableHeader();
        out += printLine(car);
        return out;
    }

    /**
     * Print bookings when parameter is a list of cars
     * Note: static
     *
     * @param cars List of cars which will be printed
     */
    public static String toString(List<Car> cars) {
        String out = String.format("\n\n\tCARS:\n");
        out += printTableHeader();
        for ( Car car : cars ) {
            out += printLine(car);
        }
        return out;
    }

    public String toString() {
        String out = String.format("\n\n\tCAR:\n");
        out += printTableHeader();
        out += printLine(this);
        return out;
    }
    // End print methods

    /**
     * Static method to filter cars.
     *
     * @param cars The list of cars which will be filtered
     * @param key Key (type, brand, transmission, seats, rate, or owner) by
         which to filter
     * @param value Value of that key by which to filter
     * @return The filtered List
     */
    public static List<Car> filter(List<Car> cars, String key, String value)
        {
        List<Car> filtered = new ArrayList<Car>();

        for ( Car car: cars ) {
            boolean check = false;
            if ( key == "type" ) {
                check = car.getType().equals(value);
            } else if ( key == "brand" ) {
                check = car.getBrand().equals(value);
            } else if ( key == "transmission" ) {
                check = car.getTransmission().equals(value);
            } else if ( key == "seats" ) {
                check = car.getSeats() == Integer.parseInt(value);
            } else if ( key == "rate" ) {
                check = car.getRate() <= Double.parseDouble(value);
            } else if ( key == "owner" ) {
                check = car.getOwner() == Integer.parseInt(value);
```

```
        }

        if ( value == null ) {
            // Add all cars if search value equals null
            filtered.add(car);
        } else if ( check ) {
            // Add car to filtered list if above applies (check == true)
            filtered.add(car);
        }
    }

    return filtered;
}

/**
 * Get a car by its ID.
 *
 * @param searchId ID of the car which is to be found
 * @return The found car (or null if 404)
 */
// to db
public static Car getById(int searchId) {
    Car found = null;
    for ( Car car : Database.getCars() ) {
        if ( car.getId() == searchId ) {
            found = car;
        }
    }
    return found;
}

/**
 * Make a report of all cars. Make two lists, one of booked cars the
     other
 * one of not-booked cars. Print both lists separately.
 */
public static String report() {
    List<Car> bookedCars    = new ArrayList<Car>();
    List<Car> notBookedCars     = new ArrayList<Car>();

    for ( Car car : Database.getCars() ) {
        for ( Booking booking : Database.getBookings() ) {
            if ( car.getId() == booking.getCarId() && !bookedCars.
                contains(car) ) {
                bookedCars.add(car);
            }
        }
    }

    for ( Car car : Database.getCars() ) {
        if ( !bookedCars.contains(car) && !notBookedCars.contains(car) )
            {
            notBookedCars.add(car);
        }
    }

    String out = Car.toString("BOOKED CARS", bookedCars);
    out += Car.toString("NOT-BOOKED CARS", notBookedCars);

    return out;
```

```
        }
    }
```

```
package model;

import data.Database;

import java.time.LocalDate;
import java.util.List;


/**
 *
 * This class functions as the Booking model of the application. Bookings
    can be
 * constructed two-fold:
 * <br>
 * <br>
 * <ul>
 * <li><strong>1: Construct the Booking with all variables as parameters.</
    strong>
 * <br> This is needed for the initialization of Bookings read from the
    database.</li>
 * <li><strong>2: Construct a Booking with limited parameters.<strong>
 * <br> This is needed for the construction of new Bookings within the
    application.</li>
 * </ul>
 */
public class Booking {
    String date;
    String bookingRef;
    int id, price, carId, userId, rentalStartTimestamp, rentalEndTimestamp;
    String rentalDate;
    static int counter;

    /**
     *
     * Constructor with all variables available as parameters.
     *
     * @param newId Unique ID of the Booking.
     * @param newDate Date of the creation of the Booking.
     * @param newCarId Car ID which the Booking is associated with.
     * @param newPrice Price of the Booking.
     * @param newUserId ID of the user that created that Booking.
     * @param newRentalDate Date of the Rental.
     * @param newRentalStartTimestamp Rental Start Timestamp as minutes from
        Start Of Day.
     * @param newRentalEndTimestamp Rental Timestamp as minutes from Start
        Of Day.
     * @param newBookingRef Booking Reference for relational and identifying
        purposes.
     */
    public Booking(int newId, String newDate, int newCarId, int newPrice,
        int newUserId, String newRentalDate, int newRentalStartTimestamp,
        int newRentalEndTimestamp, String newBookingRef) {
        id                      = newId;
        date                    = newDate;
        carId                   = newCarId;
        price                   = newPrice;
        userId                  = newUserId;
        rentalDate              = newRentalDate;
        rentalStartTimestamp    = newRentalStartTimestamp;
        rentalEndTimestamp      = newRentalEndTimestamp;
```

```
        bookingRef                  = newBookingRef;
        counter++;
    }

    /**
     *
     * Constructor with limited variables available as Parameters.
     *
     * The ID generated is a iteration of a counter which gets added by one
     * for every new user.
     *
     * The Date is a LocalDate which is then stringified to make it
         consistent to other
     * usecases of date (following).
     *
     * @param newCarId Car ID which the Booking is associated with.
     * @param newPrice Price of the Booking.
     * @param newUserId ID of the user that created that Booking.
     * @param newRentalDate Date of the Rental.
     * @param newRentalStartTimestamp Rental Start Timestamp as minutes from
         Start Of Day.
     * @param newRentalEndTimestamp Booking Reference for relational and
         identifying purposes.
     */
    public Booking(int newCarId, int newPrice, int newUserId, String
        newRentalDate, int newRentalStartTimestamp, int
        newRentalEndTimestamp) {
        id                      = ++counter;
        date                    = LocalDate.now().toString();
        carId                   = newCarId;
        price                   = newPrice;
        userId                  = newUserId;
        rentalDate              = newRentalDate;
        rentalStartTimestamp    = newRentalStartTimestamp;
        rentalEndTimestamp      = newRentalEndTimestamp;
        bookingRef                  = makeBookingRef(counter);
    }

    /**
     * Writes a query (see toQuery()) to the database
     */
    public void toDB() {
        Database.write("bookings", this.toQuery());
    }

    /**
     * Produces a string which can then be used in the writing to database
         case
     *
     * @return String query
     */
    public String toQuery() {
        String string = new String(this.id+";"+this.date+";"+this.carId+";"+
            this.price+";"+this.userId+";"+

                    this.rentalDate+";"+this.rentalStartTimestamp+";"+this.
                    rentalEndTimestamp+";"+this.bookingRef+" ");
        return string;
    }
```

```
    /**
     * Date getter
     *
     * @return String date
     */
    public String getDate() {
        return this.date;
    }

    /**
     * ID getter
     *
     * @return int id
     */
    public int getId() {
        return this.id;
    }

    /**
     * Car ID getter
     *
     * @return int car ID
     */
    public int getCarId() {
        return carId;
    }

    /**
     * Price getter
     * @return int price
     */
    public int getPrice() {
        return price;
    }

    /**
     * Customer ID getter
     *
     * @return int user (here: customer) ID
     */
    public int getCustomerId() {
        return userId;
    }

    /**
     * Rental date getter
     *
     * @return String rental date
     */
    public String getRentalDate() {
        return rentalDate;
    }

    /**
     * Rental start timestamp getter
     *
     * @return int rental start timestamp
     */
    public int getRentalStartTimestamp() {
        return rentalStartTimestamp;
```

```
    }

    /**
     * Rental End Timestamp getter
     *
     * @return int rental end timestamp
     */
    public int getRentalEndTimestamp() {
        return rentalEndTimestamp;
    }

    /**
     * Booking Reference getter
     *
     * @return String booking reference
     */
    public String getBookingRef() {
        return bookingRef;
    }

    /**
     * Price setter
     *
     * @param newPrice Set this instance's price to newPrice
     */
    public void setPrice(int newPrice) {
        this.price = newPrice;
    }

    /**
     * Rental start timestamp setter
     *
     * @param startTimestamp Set this instance's rentalStartTimestamp to
         startTimestamp
     */
    public void setStartTimestamp(int startTimestamp) {
        this.rentalStartTimestamp = startTimestamp;
    }


    /**
     * Rental return timestamp setter
     *
     * @param returnTimestamp Set this instance's rentalEndTimestamp to
         returnTimestamp
     */
    public void setReturnTimestamp(int returnTimestamp) {
        this.rentalEndTimestamp = returnTimestamp;
    }

    /**
     * Creates a booking reference
     *
     * @return String booking reference
     */
    public String makeBookingRef(int counter) {
        return counter + "_" + date + "/" + carId + "/" + userId;
    }

    /**
```

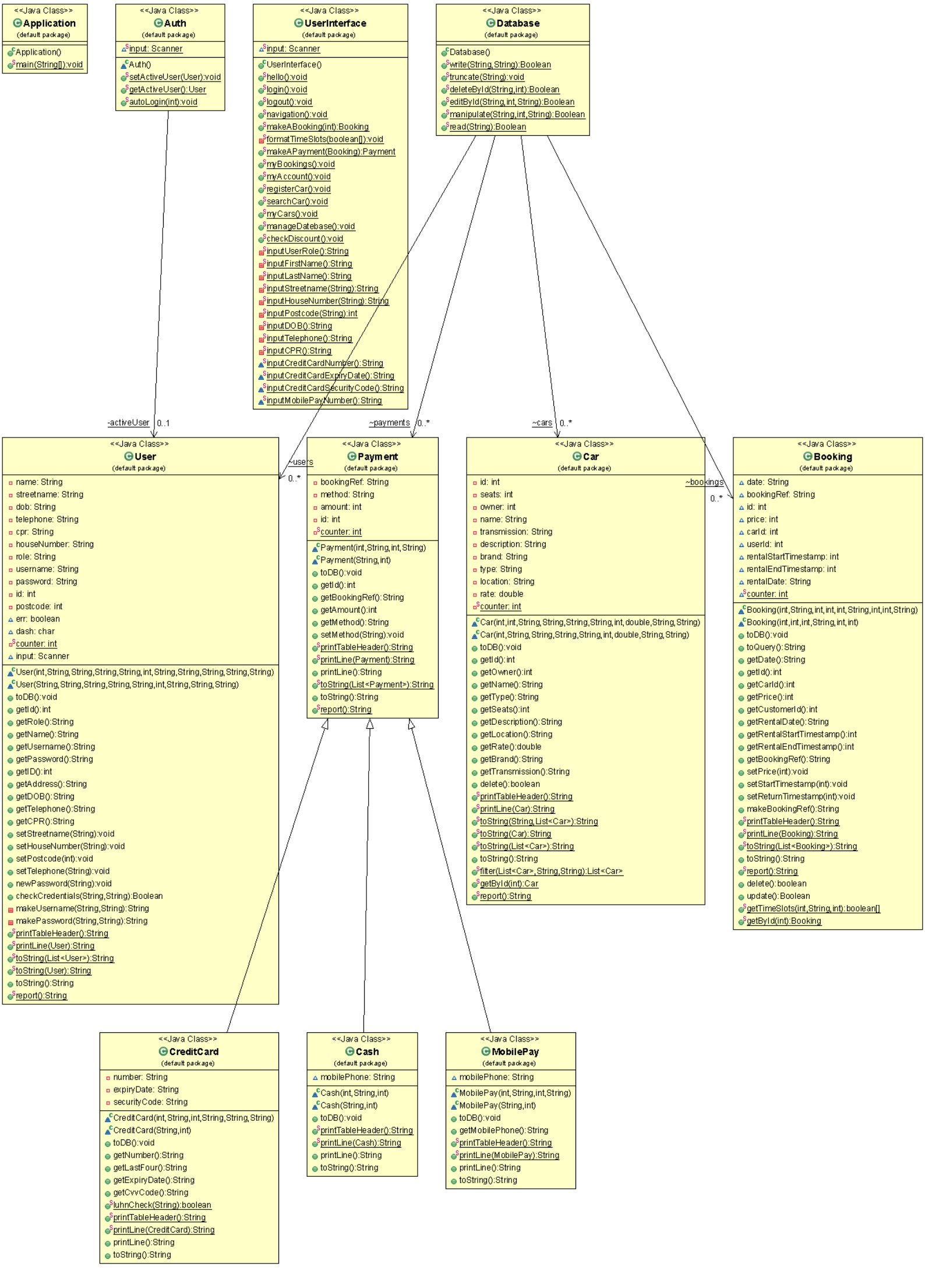
```
     * Print formated table header
     */
    public static String printTableHeader() {
        return String.format("\t  | %-3s | %-20s | %-7s | %-8s | %-11s |
            %-10s |\n",
                "ID", "Booking Ref", "Car ID", "Price", "Customer ID",
                    "Date");
    }

    /**
     * Print formated line of a booking
     */
    public static String printLine(Booking booking) {
        return String.format("\t  | %-3s | %-20s | %-7s | %-8s | %-11s |
            %-10s |\n",
                booking.getId(), booking.getBookingRef(), booking.getCarId()
                    , booking.getPrice()/100.0, booking.getCustomerId(),
                    booking.getDate());
    }

    /**
     * Print bookings when parameter is a list of bookings
     * Note: static
     *
     * @param bookings List of bookings which are to be printed
     */
    public static String toString(List<Booking> bookings) {
        String out = String.format("\n\n\tBOOKINGS:\n");
        out += printTableHeader();
        for ( Booking booking : bookings ) {
            out += printLine(booking);
        }
        return out;
    }

    /**
     * Print an instance's details
     * @return
     */
    public String toString() {
        String out = String.format("\n\n\tYOUR BOOKING:\n");
        out += printTableHeader();
        out += printLine(this);
        return out;
    }

    /**
     * Print all bookings in the database
     */
    public static String report() {
        return toString(Database.getBookings());
    }

    /**
     * Delete this Booking instance from the database
     */
    public boolean delete() {
        return Database.deleteById("bookings", this.id);
    }
```

```
    /**
     * Update this Booking instance in the database
     */
    public Boolean update() {
        String query = this.toQuery();
        return Database.editById("bookings", this.id, query);
    }

    /**
     * Get all a list of all time slots for a specific car and date in 30
         minute slots
     * (48 slots per day).
     *
     * @param carId The ID of the car a user wants to see
     * @param date The date of the time slots
     * @param exclude In the case of an update (edit booking) setting this
         parameter allows
     * to exclude the booking which is to be changed from the time slot
         overview.
     * @return boolean[] array of time slots.
     */
    public static boolean[] getTimeSlots(int carId, String date, int exclude
        ) {
        boolean[] timeSlots = new boolean[48];

        // Initialize the array
        for ( int i = 0; i<timeSlots.length; i++ ) {
            timeSlots[i] = true;
        }

        // Loop through all bookings and time slots
        for ( Booking booking: Database.getBookings() ) {
            for ( int i = 0; i<timeSlots.length; i++ ) {
                if ( booking.getCarId() == carId && booking.getRentalDate().
                    equals(date) && booking.getId() != exclude ) {
                    int start       = booking.getRentalStartTimestamp();
                    int end             = booking.getRentalEndTimestamp();

                    // If this time booking falls within a time slot, mark
                        it false (= unavailable)
                    if ( i*30 >= start && i*30 < end) {
                        timeSlots[i] = false;
                    }
                }
            }
        }

        return timeSlots;
    }

    /**
     * Static method to get a Booking by its ID.
     *
     * @param searchId The ID you want to search for.
     * @return The found Booking (or null if 404)
     */
    public static Booking getById(int searchId) {
        Booking found = null;
        for ( Booking booking : Database.getBookings() ) {
            if ( booking.getId() == searchId ) {
```

```
                    found = booking;
                }
            }
            return found;
        }

    }
```

```java
package model;

import data.Database;

import java.util.Scanner;


/**
 *
 * This class' purpose is to allow authentication for users.
 * The constructor's purpose is to set the active user to null which
 * then will be modified by the login method (see UserInterace.login).
 * Since only one user can be logged-in at once, the static User variable
 * is set globally and accessible via the public getter getActiveUser();
     below.
 *
 */
public class Auth {
    static Scanner input = new Scanner(System.in);
    private static User activeUser;


    // To Do: needed?
    public Auth() {
        activeUser = null;
    }

    /**
     * Set the activeUser.
     */
    public static void setActiveUser(User newActiveUser) {
        activeUser = newActiveUser;
    }

    /**
     * Retrieve the activeUser.
     *
     * @return activeUser
     */
    public static User getActiveUser() {
        return activeUser;
    }

    /**
     * DEBUG FUNCTION
     *
     * Logs in a user with a specific id automatically to test the
         application
     * more rapidly.
     * Sets the activeUser variable.
     *
     * @param id The id of the user you want to log in as.
     */
    public static void autoLogin(int id) {
        activeUser = null;
        for ( User user : Database.getUsers() ) {
            if ( user.getId() == id ) {
                activeUser = user;
            }
        }
```

```
        }
    }
```


«Java Class»
Application
(default package)
Application()
main(String[]):void
«Java Class»
Auth
(default package)
input: Scanner
Auth()
setActiveUser(User):void
getActiveUser():User
autoLogin(int):void
«Java Class»
UserInterface
(default package)
input: Scanner
UserInterface()
hello():void
login():void
logout():void
navigation():void
makeABooking(int):Booking
formatTimeSlots(boolean[]):void
makeAPayment(Booking):Payment
myBookings():void
myAccount():void
registerCar():void
searchCar():void
myCars():void
manageDatebase():void
checkDiscount():void
inputUserRole():String
inputFirstName():String
inputLastName():String
inputStreetname(String):String
inputHouseNumber(String):String
inputPostcode(String):int
inputDOB():String
inputTelephone():String
inputCPR():String
inputCreditCardNumber():String
inputCreditCardExpiryDate():String
inputCreditCardSecurityCode():String
inputMobilePayNumber():String
«Java Class»
Database
(default package)
Database()
write(String,String):Boolean
truncate(String):void
deleteById(String,int):Boolean
editById(String,int,String):Boolean
manipulate(String,int,String):Boolean
read(String):Boolean
-activeUser
0..1
~users
0..*
~payments
0..*
~cars
0..*
~bookings
0..*
«Java Class»
User
(default package)
name: String
streetname: String
dob: String
telephone: String
cpr: String
houseNumber: String
role: String
username: String
password: String
id: int
postcode: int
err: boolean
dash: char
counter: int
input: Scanner
User(int,String,String,String,String,int,String,String,String,String,String)
User(String,String,String,String,String,int,String,String,String)
toDB():void
getId():int
getRole():String
getName():String
getUsername():String
getPassword():String
getID():int
getAddress():String
getDOB():String
getTelephone():String
getCPR():String
setStreetname(String):void
setHouseNumber(String):void
setPostcode(int):void
setTelephone(String):void
newPassword(String):void
checkCredentials(String,String):Boolean
makeUsername(String,String):String
makePassword(String,String):String
printTableHeader():String
printLine(User):String
toString(List<User>):String
toString(User):String
toString():String
report():String
«Java Class»
Payment
(default package)
bookingRef: String
method: String
amount: int
id: int
counter: int
Payment(int,String,int,String)
Payment(String,int)
toDB():void
getId():int
getBookingRef():String
getAmount():int
getMethod():String
setMethod(String):void
printTableHeader():String
printLine(Payment):String
printLine():String
toString(List<Payment>):String
toString():String
report():String
«Java Class»
Car
(default package)
id: int
seats: int
owner: int
name: String
transmission: String
description: String
brand: String
type: String
location: String
rate: double
counter: int
Car(int,int,String,String,String,String,int,double,String,String)
Car(int,String,String,String,String,int,double,String,String)
toDB():void
getId():int
getOwner():int
getName():String
getType():String
getSeats():int
getDescription():String
getLocation():String
getRate():double
getBrand():String
getTransmission():String
delete():boolean
printTableHeader():String
printLine(Car):String
toString(String,List<Car>):String
toString(Car):String
toString(List<Car>):String
toString():String
filter(List<Car>,String,String):List<Car>
getById(int):Car
report():String
«Java Class»
Booking
(default package)
date: String
bookingRef: String
id: int
price: int
carId: int
userId: int
rentalStartTimestamp: int
rentalEndTimestamp: int
rentalDate: String
counter: int
Booking(int,String,int,int,int,String,int,int,String)
Booking(int,int,int,String,int,int)
toDB():void
toQuery():String
getDate():String
getId():int
getCarId():int
getPrice():int
getCustomerId():int
getRentalDate():String
getRentalStartTimestamp():int
getRentalEndTimestamp():int
getBookingRef():String
setPrice(int):void
setStartTimestamp(int):void
setReturnTimestamp(int):void
makeBookingRef():String
printTableHeader():String
printLine(Booking):String
toString(List<Booking>):String
toString():String
report():String
delete():boolean
update():Boolean
getTimeSlots(int,String,int):boolean[]
getById(int):Booking
«Java Class»
CreditCard
(default package)
number: String
expiryDate: String
securityCode: String
CreditCard(int,String,int,String,String,String)
CreditCard(String,int)
toDB():void
getNumber():String
getLastFour():String
getExpiryDate():String
getCvvCode():String
luhnCheck(String):boolean
printTableHeader():String
printLine(CreditCard):String
printLine():String
toString():String
«Java Class»
Cash
(default package)
mobilePhone: String
Cash(int,String,int)
Cash(String,int)
toDB():void
printTableHeader():String
printLine(Cash):String
printLine():String
toString():String
«Java Class»
MobilePay
(default package)
mobilePhone: String
MobilePay(int,String,int,String)
MobilePay(String,int)
toDB():void
getMobilePhone():String
printTableHeader():String
printLine(MobilePay):String
printLine():String
toString():String

Quick Access

Console ×

Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:34:36)

```
Welcome to Share R Us


        HOME
          | 1   | Login      |
          | 2   | Register   |


Please type in your destination: 2


NEW USER
As what role do you want to be reigstered
          | 1   | Customer       |
          | 2   | Car Owner      |
          | 3   | Administrator  |
3
Please type in the sudo password to continue:
supersecret
What is your first name?
John
What is your last name?
Deer
Thank you John. Please now enter the name of your street
Javagade
Thank you John. Please now enter your house number
42
Thanks John. Please enter your postcode
2450
Ok, next, please enter your birthday in the format DD.MM.YYYY
12.01.1987
Please now enter your 8-digit telephone number:
15234231
Please enter your 10-digit CPR number in the following format xxxxxx-xxxx:
123423-1234
Your username is jdee
Your password is dee1234.


        LOGIN
        Username: jdee
        Password: dee1234

You are now logged in.

        HOME
          | 1   | Logout         |
          | 2   | Search a car   |
          | 3   | Manage database |
          | 4   | My Account     |

Please type in your destination: 4


        USER:
          | ID | Role           | Name              | CPR         | Username |
          | 4  | administrator  | John Deer         | 123423-1234 | jdee     |


        HOME
          | 1   | Logout         |
```

Quick Access

Console

Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:34:36)

```
        HOME
          | 1   | Logout          |
          | 2   | Search a car    |
          | 3   | Manage database |
          | 4   | My Account      |

Please type in your destination: 1
You have been logged out.
Goodbye!


        HOME
          | 1   | Login     |
          | 2   | Register  |


Please type in your destination: 2


NEW USER
As what role do you want to be reigstered
          | 1   | Customer        |
          | 2   | Car Owner       |
          | 3   | Administrator   |
1
What is your first name?
Michael
What is your last name?
Nox
Thank you Michael. Please now enter the name of your street
Javakaj
Thank you Michael. Please now enter your house number
12
Thanks Michael. Please enter your postcode
1450
Ok, next, please enter your birthday in the format DD.MM.YYYY
01.10.1990
Please now enter your 8-digit telephone number:
152435984
The number you entered is invalid.
Please now enter your 8-digit telephone number:
98715342
Please enter your 10-digit CPR number in the following format xxxxxx-xxxx:
162435-1535
Your username is mnox
Your password is nox1535.


        LOGIN
        Username: mnox
        Password: nox1535

You are now logged in.

        HOME
          | 1   | Logout          |
          | 2   | My Bookings     |
          | 3   | Make a booking  |
          | 4   | Search a car    |
          | 5   | My Account      |
```

Quick Access

Console

Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:34:36)

```
        HOME
          | 1   | Login      |
          | 2   | Register   |


Please type in your destination: 2


NEW USER
As what role do you want to be reigstered
          | 1   | Customer       |
          | 2   | Car Owner      |
          | 3   | Administrator  |
1
What is your first name?
Michael
What is your last name?
Nox
Thank you Michael. Please now enter the name of your street
Javakaj
Thank you Michael. Please now enter your house number
12
Thanks Michael. Please enter your postcode
1450
Ok, next, please enter your birthday in the format DD.MM.YYYY
01.10.1990
Please now enter your 8-digit telephone number:
152435984
The number you entered is invalid.
Please now enter your 8-digit telephone number:
98715342
Please enter your 10-digit CPR number in the following format xxxxxx-xxxx:
162435-1535
Your username is mnox
Your password is nox1535.


        LOGIN
        Username: mnox
        Password: nox1535

You are now logged in.

        HOME
          | 1   | Logout         |
          | 2   | My Bookings    |
          | 3   | Make a booking |
          | 4   | Search a car   |
          | 5   | My Account     |

Please type in your destination: 1
You have been logged out.
Goodbye!


        HOME
          | 1   | Login      |
          | 2   | Register   |


Please type in your destination:
```

Quick Access

Console ×

Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:38:04)

```
Welcome to Share R Us


        HOME
          | 1   | Login      |
          | 2   | Register   |


Please type in your destination: 1


        LOGIN
        Username: jche
        Password: chev1231

You are now logged in.

        HOME
          | 1   | Logout          |
          | 2   | My Cars         |
          | 3   | Register a car  |
          | 4   | My Account      |

Please type in your destination: 3


REGISTER CAR
Name of the car: Batmobile


        TYPE
          | 1   | Sedan           |
          | 2   | Convertible     |
          | 3   | Sports Car      |
          | 4   | SUV             |
          | 5   | Other           |
3
Brand: WayneCorp


        TRANSMISSION
          | 1   | Manual          |
          | 2   | Automatic       |
2
Amount of seats: 1
Hourly rate: 2000
Description: Top Secret
Location:

        CAR:
          | ID | Name       | Type         | Brand                | Transmission | Seats          | Rate per hour
          | 45 | Batmobile  | Sports Car   | WayneCorp            | Automatic    | 1              | 2000.0


        HOME
          | 1   | Logout          |
          | 2   | My Cars         |
          | 3   | Register a car  |
          | 4   | My Account      |

Please type in your destination: 2
```

Quick Access

Console

Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:38:04)

```
          | 4    | My Account       |

Please type in your destination: 2


        MY CARS:
          | ID | Name        | Type         | Brand          | Transmission | Seats | Rate per hour
          | 23 | Boxster     | Sports Car   | Porsche        | Automatic    | 4     | 900.0
          | 24 | 911         | Sports Car   | Porsche        | Manual       | 4     | 1200.0
          | 25 | Carrera     | Sports Car   | Porsche        | Manual       | 2     | 1200.0
          | 26 | Panamera    | Sports Car   | Porsche        | Manual       | 2     | 1200.0
          | 27 | Cayenne     | Sports Car   | Porsche        | Manual       | 2     | 1500.0
          | 28 | Macan       | Sports Car   | Porsche        | Manual       | 2     | 2000.0
          | 29 | Model X     | Sports Car   | Tesla          | Automatic    | 2     | 300.0
          | 30 | Model S     | Sports Car   | Tesla          | Automatic    | 2     | 500.0
          | 31 | Model 3     | Sports Car   | Tesla          | Automatic    | 4     | 600.0
          | 32 | Model E     | Sports Car   | Tesla          | Automatic    | 6     | 250.0
          | 33 | Model F     | Sports Car   | Tesla          | Automatic    | 2     | 350.0
          | 34 | 108         | Other        | Peugeot        | Automatic    | 4     | 150.0
          | 35 | 208         | Sedan        | Peugeot        | Automatic    | 4     | 150.0
          | 36 | 2008        | SUV          | Peugeot        | Automatic    | 4     | 150.0
          | 37 | NY 308      | Other        | Peugeot        | Automatic    | 5     | 125.0
          | 38 | 3008        | Other        | Peugeot        | Manual       | 5     | 190.0
          | 39 | 508         | Other        | Peugeot        | Manual       | 5     | 250.0
          | 40 | Fusion      | Other        | Ford           | Manual       | 5     | 200.0
          | 41 | Espace      | Other        | Ford           | Manual       | 5     | 190.0
          | 42 | Focus       | Other        | Ford           | Manual       | 5     | 250.0
          | 43 | Fiesta      | Sedan        | Ford           | Automatic    | 2     | 260.0
          | 44 | Mustang     | Sedan        | Ford           | Automatic    | 2     | 280.0
          | 45 | Batmobile   | Sports Car   | WayneCorp      | Automatic    | 1     | 2000.0


        CONTINUE
          | 0    | Exit                 |
          | 1    | See availability     |
1

Type in the ID of the car you want to check: 45
You have chosen:


        CAR:
          | ID | Name        | Type         | Brand          | Transmission | Seats | Rate per hour
          | 45 | Batmobile   | Sports Car   | WayneCorp      | Automatic    | 1     | 2000.0

Which day do you want to see for the selected car? (DD.MM.YYYY): 24.12.2017
See here all available time slots:


        24.12.2017
          |  0: 0 | Available  |
          |  0:30 | Available  |
          |  1: 0 | Available  |
          |  1:30 | Available  |
          |  2: 0 | Available  |
          |  2:30 | Available  |
          |  3: 0 | Available  |
          |  3:30 | Available  |
          |  4: 0 | Available  |
          |  4:30 | Available  |
          |  5: 0 | Available  |
```

Quick Access

Console

Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:38:04)

```
|  3: 0 | Available  |
|  3:30 | Available  |
|  4: 0 | Available  |
|  4:30 | Available  |
|  5: 0 | Available  |
|  5:30 | Available  |
|  6: 0 | Available  |
|  6:30 | Available  |
|  7: 0 | Available  |
|  7:30 | Available  |
|  8: 0 | Available  |
|  8:30 | Available  |
|  9: 0 | Available  |
|  9:30 | Available  |
| 10: 0 | Available  |
| 10:30 | Available  |
| 11: 0 | Available  |
| 11:30 | Available  |
| 12: 0 | Available  |
| 12:30 | Available  |
| 13: 0 | Available  |
| 13:30 | Available  |
| 14: 0 | Available  |
| 14:30 | Available  |
| 15: 0 | Available  |
| 15:30 | Available  |
| 16: 0 | Available  |
| 16:30 | Available  |
| 17: 0 | Available  |
| 17:30 | Available  |
| 18: 0 | Available  |
| 18:30 | Available  |
| 19: 0 | Available  |
| 19:30 | Available  |
| 20: 0 | Available  |
| 20:30 | Available  |
| 21: 0 | Available  |
| 21:30 | Available  |
| 22: 0 | Available  |
| 22:30 | Available  |
| 23: 0 | Available  |
| 23:30 | Available  |


      HOME
        | 1   | Logout          |
        | 2   | My Cars         |
        | 3   | Register a car  |
        | 4   | My Account      |

Please type in your destination: 1
You have been logged out.
Goodbye!


      HOME
        | 1   | Login     |
        | 2   | Register  |


Please type in your destination:
```

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
Welcome to Share R Us


        HOME
          | 1   | Login      |
          | 2   | Register   |


Please type in your destination: 1


        LOGIN
        Username: jdoe
        Password: root

You are now logged in.

        HOME
          | 1   | Logout          |
          | 2   | Search a car    |
          | 3   | Manage database |
          | 4   | My Account      |

Please type in your destination: 2
These are all cars available:


        ALL CARS:
          | ID | Name       | Type        | Brand         | Transmission | Seats | Rate per hour
          | 1  | GLA        | Convertible | Mercedes Benz | Manual       | 2     | 250.0
          | 2  | M          | Sedan       | Mercedes Benz | Manual       | 4     | 250.0
          | 3  | RR         | Sedan       | Mercedes Benz | Manual       | 4     | 250.0
          | 4  | AMG        | Sedan       | Mercedes Benz | Manual       | 4     | 250.0
          | 5  | C          | Sedan       | Mercedes Benz | Manual       | 4     | 400.0
          | 6  | S          | Sedan       | Mercedes Benz | Automatic    | 4     | 1000.0
          | 7  | G          | Sedan       | Mercedes Benz | Automatic    | 4     | 750.0
          | 8  | X5         | Sedan       | BMW           | Manual       | 4     | 800.0
          | 9  | X6         | Sedan       | BMW           | Manual       | 4     | 1000.0
          | 10 | i8         | SUV         | BMW           | Manual       | 4     | 200.0
          | 11 | i5         | SUV         | BMW           | Manual       | 4     | 200.0
          | 12 | i3         | SUV         | BMW           | Manual       | 4     | 200.0
          | 13 | 2          | SUV         | BMW           | Manual       | 2     | 250.0
          | 14 | 3          | SUV         | BMW           | Manual       | 2     | 250.0
          | 15 | 5          | SUV         | BMW           | Manual       | 6     | 250.0
          | 16 | 4          | SUV         | BMW           | Manual       | 8     | 500.0
          | 17 | A1         | SUV         | Audi          | Manual       | 4     | 300.0
          | 18 | A2         | SUV         | Audi          | Manual       | 4     | 300.0
          | 19 | A3         | SUV         | Audi          | Automatic    | 4     | 300.0
          | 20 | Q1         | Convertible | Audi          | Automatic    | 4     | 300.0
          | 21 | Q3         | Convertible | Audi          | Automatic    | 4     | 300.0
          | 22 | Q5         | Convertible | Audi          | Automatic    | 2     | 700.0
          | 23 | Boxster    | Sports Car  | Porsche       | Automatic    | 4     | 900.0
          | 24 | 911        | Sports Car  | Porsche       | Manual       | 4     | 1200.0
          | 25 | Carrera    | Sports Car  | Porsche       | Manual       | 2     | 1200.0
          | 26 | Panamera   | Sports Car  | Porsche       | Manual       | 2     | 1200.0
          | 27 | Cayenne    | Sports Car  | Porsche       | Manual       | 2     | 1500.0
          | 28 | Macan      | Sports Car  | Porsche       | Manual       | 2     | 2000.0
          | 29 | Model X    | Sports Car  | Tesla         | Automatic    | 2     | 300.0
          | 30 | Model S    | Sports Car  | Tesla         | Automatic    | 2     | 500.0
          | 31 | Model 3    | Sports Car  | Tesla         | Automatic    | 4     | 600.0
          | 32 | Model E    | Sports Car  | Tesla         | Automatic    | 6     | 250.0
```

Quick Access

Console ×

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

Please type in your destination: 2

These are all cars available:

ALL CARS:

ID	Name	Type	Brand	Transmission	Seats	Rate per hour
1	GLA	Convertible	Mercedes Benz	Manual	2	250.0
2	M	Sedan	Mercedes Benz	Manual	4	250.0
3	RR	Sedan	Mercedes Benz	Manual	4	250.0
4	AMG	Sedan	Mercedes Benz	Manual	4	250.0
5	C	Sedan	Mercedes Benz	Manual	4	400.0
6	S	Sedan	Mercedes Benz	Automatic	4	1000.0
7	G	Sedan	Mercedes Benz	Automatic	4	750.0
8	X5	Sedan	BMW	Manual	4	800.0
9	X6	Sedan	BMW	Manual	4	1000.0
10	i8	SUV	BMW	Manual	4	200.0
11	i5	SUV	BMW	Manual	4	200.0
12	i3	SUV	BMW	Manual	4	200.0
13	2	SUV	BMW	Manual	2	250.0
14	3	SUV	BMW	Manual	2	250.0
15	5	SUV	BMW	Manual	6	250.0
16	4	SUV	BMW	Manual	8	500.0
17	A1	SUV	Audi	Manual	4	300.0
18	A2	SUV	Audi	Manual	4	300.0
19	A3	SUV	Audi	Automatic	4	300.0
20	Q1	Convertible	Audi	Automatic	4	300.0
21	Q3	Convertible	Audi	Automatic	4	300.0
22	Q5	Convertible	Audi	Automatic	2	700.0
23	Boxster	Sports Car	Porsche	Automatic	4	900.0
24	911	Sports Car	Porsche	Manual	4	1200.0
25	Carrera	Sports Car	Porsche	Manual	2	1200.0
26	Panamera	Sports Car	Porsche	Manual	2	1200.0
27	Cayenne	Sports Car	Porsche	Manual	2	1500.0
28	Macan	Sports Car	Porsche	Manual	2	2000.0
29	Model X	Sports Car	Tesla	Automatic	2	300.0
30	Model S	Sports Car	Tesla	Automatic	2	500.0
31	Model 3	Sports Car	Tesla	Automatic	4	600.0
32	Model E	Sports Car	Tesla	Automatic	6	250.0
33	Model F	Sports Car	Tesla	Automatic	2	350.0
34	108	Other	Peugeot	Automatic	4	150.0
35	208	Sedan	Peugeot	Automatic	4	150.0
36	2008	SUV	Peugeot	Automatic	4	150.0
37	NY 308	Other	Peugeot	Automatic	5	125.0
38	3008	Other	Peugeot	Manual	5	190.0
39	508	Other	Peugeot	Manual	5	250.0
40	Fusion	Other	Ford	Manual	5	200.0
41	Espace	Other	Ford	Manual	5	190.0
42	Focus	Other	Ford	Manual	5	250.0
43	Fiesta	Sedan	Ford	Automatic	2	260.0
44	Mustang	Sedan	Ford	Automatic	2	280.0
45	Taurus	Sedan	Ford	Automatic	2	180.0

REFINE SEARCH

0	Exit
1	By type
2	By brand
3	By transmission
4	By seats
5	By rate per hour

1

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
      AVAILABLE TYPES
        | 1   | Show All     |
        | 2   | Sedan        |
        | 3   | Convertible  |
        | 4   | Sports Car   |
        | 5   | SUV          |
2


      FILTERED CARS:
        | ID | Name       | Type       | Brand          | Transmission | Seats | Rate per hour
        | 2  | M          | Sedan      | Mercedes Benz  | Manual       | 4     | 250.0
        | 3  | RR         | Sedan      | Mercedes Benz  | Manual       | 4     | 250.0
        | 4  | AMG        | Sedan      | Mercedes Benz  | Manual       | 4     | 250.0
        | 5  | C          | Sedan      | Mercedes Benz  | Manual       | 4     | 400.0
        | 6  | S          | Sedan      | Mercedes Benz  | Automatic    | 4     | 1000.0
        | 7  | G          | Sedan      | Mercedes Benz  | Automatic    | 4     | 750.0
        | 8  | X5         | Sedan      | BMW            | Manual       | 4     | 800.0
        | 9  | X6         | Sedan      | BMW            | Manual       | 4     | 1000.0
        | 35 | 208        | Sedan      | Peugeot        | Automatic    | 4     | 150.0
        | 43 | Fiesta     | Sedan      | Ford           | Automatic    | 2     | 260.0
        | 44 | Mustang    | Sedan      | Ford           | Automatic    | 2     | 280.0
        | 45 | Taurus     | Sedan      | Ford           | Automatic    | 2     | 180.0


      REFINE SEARCH
        | 0   | Exit              |
        | 2   | By brand          |
        | 3   | By transmission   |
        | 4   | By seats          |
        | 5   | By rate per hour  |
2
Please type in the brand:
BMW


      FILTERED CARS:
        | ID | Name       | Type       | Brand          | Transmission | Seats | Rate per hour
        | 8  | X5         | Sedan      | BMW            | Manual       | 4     | 800.0
        | 9  | X6         | Sedan      | BMW            | Manual       | 4     | 1000.0


      REFINE SEARCH
        | 0   | Exit              |
        | 3   | By transmission   |
        | 4   | By seats          |
        | 5   | By rate per hour  |
0


      HOME
        | 1   | Logout          |
        | 2   | Search a car    |
        | 3   | Manage database |
        | 4   | My Account      |

Please type in your destination: 4


      USER:
```

Quick Access

Console ×

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
Please type in your destination: 4


        USER:
          | ID | Role          | Name          | CPR         | Username |
          | 1  | administrator | John Doe      | 120190-2201 | jdoe     |


        HOME
          | 1   | Logout          |
          | 2   | Search a car    |
          | 3   | Manage database |
          | 4   | My Account      |

Please type in your destination: 3


        CONTINUE
          | 0   | Exit            |
          | 1   | Cars report     |
          | 2   | Bookings report |
          | 3   | Payments report |
          | 4   | Users report    |
2


        BOOKINGS:
          | ID  | Booking Ref           | Car ID | Price   | Customer ID | Date       |
          | 1   | 2017-11-03/2/1        | 2      | 150.0   | 2           | 2017-11-03 |
          | 2   | 2017-11-03/2/1        | 2      | 250.0   | 2           | 2017-11-03 |
          | 3   | 2017-11-03/2/1        | 2      | 450.0   | 2           | 2017-11-03 |
          | 4   | 2017-11-03/2/1        | 2      | 250.0   | 2           | 2017-11-03 |
          | 5   | 2017-11-03/2/1        | 2      | 213.75  | 2           | 2017-11-03 |
          | 6   | 2017-11-03/2/1        | 2      | 380.0   | 2           | 2017-11-03 |
          | 7   | 2017-11-03/2/1        | 2      | 300.0   | 2           | 2017-11-03 |
          | 8   | 2017-11-03/1/1        | 1      | 200.0   | 2           | 2017-11-03 |
          | 9   | 2017-11-03/1/1        | 1      | 200.0   | 2           | 2017-11-03 |
          | 10  | 2017-11-03/1/1        | 1      | 200.0   | 1           | 2017-11-03 |
          | 11  | 2017-11-03/1/1        | 1      | 200.0   | 1           | 2017-11-03 |
          | 12  | 2017-11-03/1/1        | 1      | 200.0   | 1           | 2017-11-03 |
          | 13  | 2017-11-03/1/1        | 1      | 200.0   | 1           | 2017-11-03 |
          | 14  | 2017-11-03/1/1        | 1      | 200.0   | 1           | 2017-11-03 |
          | 15  | 2017-11-03/2/1        | 2      | 100.0   | 1           | 2017-11-03 |
          | 17  | 2017-11-03/2/1        | 2      | 100.0   | 1           | 2017-11-03 |
          | 18  | 2017-11-03/2/1        | 2      | 100.0   | 1           | 2017-11-03 |
          | 19  | 2017-11-03/2/1        | 2      | 100.0   | 1           | 2017-11-03 |
          | 20  | 2017-11-05/2/1        | 2      | 100.0   | 1           | 2017-11-05 |
          | 20  | 2017-11-06/4/1        | 4      | 150.0   | 1           | 2017-11-06 |
          | 21  | 2017-11-06/4/1        | 4      | 100.0   | 1           | 2017-11-06 |
          | 22  | 2017-11-07/2/1        | 2      | 83.33   | 1           | 2017-11-07 |
          | 23  | 2017-11-07/4/2        | 4      | 190.0   | 2           | 2017-11-07 |
          | 24  | 24_2017-12-07/30/4    | 30     | 500.0   | 4           | 2017-12-07 |
          | 25  | 25_2017-12-07/5/4     | 5      | 400.0   | 4           | 2017-12-07 |
          | 27  | 27_2017-12-07/24/4    | 24     | 600.0   | 4           | 2017-12-07 |
          | 28  | 28_2017-12-07/22/4    | 22     | 2100.0  | 4           | 2017-12-07 |
          | 87  | 87_2017-12-07/1/4     | 1      | 950.0   | 4           | 2017-12-07 |


HOME
          | 1   | Logout          |
```

Quick Access

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
          | 28  | 28_2017-12-07/22/4   | 22       | 2100.0   | 4             | 2017-12-07 |
          | 87  | 87_2017-12-07/1/4    | 1        | 950.0    | 4             | 2017-12-07 |


        HOME
          | 1   | Logout           |
          | 2   | Search a car     |
          | 3   | Manage database  |
          | 4   | My Account       |

Please type in your destination: 3


        CONTINUE
          | 0   | Exit                |
          | 1   | Cars report         |
          | 2   | Bookings report     |
          | 3   | Payments report     |
          | 4   | Users report        |
4


        USERS:
          | ID | Role           | Name              | CPR          | Username |
          | 1  | administrator  | John Doe          | 120190-2201  | jdoe     |
          | 2  | customer       | Michelle Deer     | 290791-1234  | mdee     |
          | 3  | car owner      | Jason Chevalier   | 091277-4123  | jche     |


        HOME
          | 1   | Logout           |
          | 2   | Search a car     |
          | 3   | Manage database  |
          | 4   | My Account       |

Please type in your destination: 3


        CONTINUE
          | 0   | Exit                |
          | 1   | Cars report         |
          | 2   | Bookings report     |
          | 3   | Payments report     |
          | 4   | Users report        |
1


        BOOKED CARS:
          | ID | Name     | Type        | Brand          | Transmission | Seats | Rate per hour
          | 1  | GLA      | Convertible | Mercedes Benz  | Manual       | 2     | 250.0
          | 2  | M        | Sedan       | Mercedes Benz  | Manual       | 4     | 250.0
          | 4  | AMG      | Sedan       | Mercedes Benz  | Manual       | 4     | 250.0
          | 5  | C        | Sedan       | Mercedes Benz  | Manual       | 4     | 400.0
          | 22 | Q5       | Convertible | Audi           | Automatic    | 2     | 700.0
          | 24 | 911      | Sports Car  | Porsche        | Manual       | 4     | 1200.0
          | 30 | Model S  | Sports Car  | Tesla          | Automatic    | 2     | 500.0


        NOT-BOOKED CARS:
          | ID | Name     | Type        | Brand          | Transmission | Seats | Rate per hour
          | 3  | RR       | Sedan       | Mercedes Benz  | Manual       | 4     | 250.0
```

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

NOT-BOOKED CARS:

ID	Name	Type	Brand	Transmission	Seats	Rate per hour
3	RR	Sedan	Mercedes Benz	Manual	4	250.0
6	S	Sedan	Mercedes Benz	Automatic	4	1000.0
7	G	Sedan	Mercedes Benz	Automatic	4	750.0
8	X5	Sedan	BMW	Manual	4	800.0
9	X6	Sedan	BMW	Manual	4	1000.0
10	i8	SUV	BMW	Manual	4	200.0
11	i5	SUV	BMW	Manual	4	200.0
12	i3	SUV	BMW	Manual	4	200.0
13	2	SUV	BMW	Manual	2	250.0
14	3	SUV	BMW	Manual	2	250.0
15	5	SUV	BMW	Manual	6	250.0
16	4	SUV	BMW	Manual	8	500.0
17	A1	SUV	Audi	Manual	4	300.0
18	A2	SUV	Audi	Manual	4	300.0
19	A3	SUV	Audi	Automatic	4	300.0
20	Q1	Convertible	Audi	Automatic	4	300.0
21	Q3	Convertible	Audi	Automatic	4	300.0
23	Boxster	Sports Car	Porsche	Automatic	4	900.0
25	Carrera	Sports Car	Porsche	Manual	2	1200.0
26	Panamera	Sports Car	Porsche	Manual	2	1200.0
27	Cayenne	Sports Car	Porsche	Manual	2	1500.0
28	Macan	Sports Car	Porsche	Manual	2	2000.0
29	Model X	Sports Car	Tesla	Automatic	2	300.0
31	Model 3	Sports Car	Tesla	Automatic	4	600.0
32	Model E	Sports Car	Tesla	Automatic	6	250.0
33	Model F	Sports Car	Tesla	Automatic	2	350.0
34	108	Other	Peugeot	Automatic	4	150.0
35	208	Sedan	Peugeot	Automatic	4	150.0
36	2008	SUV	Peugeot	Automatic	4	150.0
37	NY 308	Other	Peugeot	Automatic	5	125.0
38	3008	Other	Peugeot	Manual	5	190.0
39	508	Other	Peugeot	Manual	5	250.0
40	Fusion	Other	Ford	Manual	5	200.0
41	Espace	Other	Ford	Manual	5	190.0
42	Focus	Other	Ford	Manual	5	250.0
43	Fiesta	Sedan	Ford	Automatic	2	260.0
44	Mustang	Sedan	Ford	Automatic	2	280.0
45	Taurus	Sedan	Ford	Automatic	2	180.0

CONTINUE
| 0 | Exit |
| 1 | Delete Car |

1

Type in the ID of the car you want to delete: 45

Are you sure you want to delete this car? This action can not be reserved.
| 1 | Yes |
| 2 | Abort |

1

The item has been successfully deleted.

HOME
1	Logout
2	Search a car
3	Manage database
4	My Account

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
        HOME
          | 1   | Logout          |
          | 2   | Search a car    |
          | 3   | Manage database |
          | 4   | My Account      |

Please type in your destination: 1
You have been logged out.
Goodbye!


        HOME
          | 1   | Login     |
          | 2   | Register  |


Please type in your destination: 1


        LOGIN
        Username: mdee
        Password: dee1423

You are now logged in.

        HOME
          | 1   | Logout          |
          | 2   | My Bookings     |
          | 3   | Make a booking  |
          | 4   | Search a car    |
          | 5   | My Account      |

Please type in your destination: 2


        BOOKINGS:
          | ID  | Booking Ref         | Car ID  | Price    | Customer ID | Date       |
          | 1   | 2017-11-03/2/1      | 2       | 150.0    | 2           | 2017-11-03 |
          | 2   | 2017-11-03/2/1      | 2       | 250.0    | 2           | 2017-11-03 |
          | 3   | 2017-11-03/2/1      | 2       | 450.0    | 2           | 2017-11-03 |
          | 4   | 2017-11-03/2/1      | 2       | 250.0    | 2           | 2017-11-03 |
          | 5   | 2017-11-03/2/1      | 2       | 213.75   | 2           | 2017-11-03 |
          | 6   | 2017-11-03/2/1      | 2       | 380.0    | 2           | 2017-11-03 |
          | 7   | 2017-11-03/2/1      | 2       | 300.0    | 2           | 2017-11-03 |
          | 8   | 2017-11-03/1/1      | 1       | 200.0    | 2           | 2017-11-03 |
          | 9   | 2017-11-03/1/1      | 1       | 200.0    | 2           | 2017-11-03 |
          | 23  | 2017-11-07/4/2      | 4       | 190.0    | 2           | 2017-11-07 |


        NEXT
          | 1   | Edit Booking    |
          | 2   | Delete Booking  |
          | 3   | Exit            |
2

Type in the ID of the booking you want to delete: 23

Are you sure you want to delete this Booking? This action can not be reserved.
          | 1   | Yes   |
          | 2   | Abort |
1
```

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
          | 2   | Abort |
1

The item has been successfully deleted.

        HOME
          | 1   | Logout          |
          | 2   | My Bookings     |
          | 3   | Make a booking  |
          | 4   | Search a car    |
          | 5   | My Account      |

Please type in your destination: 2


        BOOKINGS:
          | ID  | Booking Ref          | Car ID  | Price    | Customer ID | Date       |
          | 1   | 2017-11-03/2/1       | 2       | 150.0    | 2           | 2017-11-03 |
          | 2   | 2017-11-03/2/1       | 2       | 250.0    | 2           | 2017-11-03 |
          | 3   | 2017-11-03/2/1       | 2       | 450.0    | 2           | 2017-11-03 |
          | 4   | 2017-11-03/2/1       | 2       | 250.0    | 2           | 2017-11-03 |
          | 5   | 2017-11-03/2/1       | 2       | 213.75   | 2           | 2017-11-03 |
          | 6   | 2017-11-03/2/1       | 2       | 380.0    | 2           | 2017-11-03 |
          | 7   | 2017-11-03/2/1       | 2       | 300.0    | 2           | 2017-11-03 |
          | 8   | 2017-11-03/1/1       | 1       | 200.0    | 2           | 2017-11-03 |
          | 9   | 2017-11-03/1/1       | 1       | 200.0    | 2           | 2017-11-03 |


        NEXT
          | 1   | Edit Booking    |
          | 2   | Delete Booking  |
          | 3   | Exit            |
3


        HOME
          | 1   | Logout          |
          | 2   | My Bookings     |
          | 3   | Make a booking  |
          | 4   | Search a car    |
          | 5   | My Account      |

Please type in your destination: 5


        USER:
          | ID | Role            | Name                | CPR         | Username |
          | 2  | customer        | Michelle Deer       | 290791-1234 | mdee     |


        HOME
          | 1   | Logout          |
          | 2   | My Bookings     |
          | 3   | Make a booking  |
          | 4   | Search a car    |
          | 5   | My Account      |

Please type in your destination: 3


        AVAILABLE TYPES
```

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
Please type in your destination: 3


        AVAILABLE TYPES
          | 1   | Show All     |
          | 2   | Sedan        |
          | 3   | Convertible  |
          | 4   | Sports Car   |
          | 5   | SUV          |

What type of car are you looking for (type #)? 4


        FILTERED CARS:
          | ID | Name       | Type        | Brand     | Transmission | Seats | Rate per hour
          | 23 | Boxster    | Sports Car  | Porsche   | Automatic    | 4     | 900.0
          | 24 | 911        | Sports Car  | Porsche   | Manual       | 4     | 1200.0
          | 25 | Carrera    | Sports Car  | Porsche   | Manual       | 2     | 1200.0
          | 26 | Panamera   | Sports Car  | Porsche   | Manual       | 2     | 1200.0
          | 27 | Cayenne    | Sports Car  | Porsche   | Manual       | 2     | 1500.0
          | 28 | Macan      | Sports Car  | Porsche   | Manual       | 2     | 2000.0
          | 29 | Model X    | Sports Car  | Tesla     | Automatic    | 2     | 300.0
          | 30 | Model S    | Sports Car  | Tesla     | Automatic    | 2     | 500.0
          | 31 | Model 3    | Sports Car  | Tesla     | Automatic    | 4     | 600.0
          | 32 | Model E    | Sports Car  | Tesla     | Automatic    | 6     | 250.0
          | 33 | Model F    | Sports Car  | Tesla     | Automatic    | 2     | 350.0

Type in the ID of the car you want to book: 30
You have chosen:


        CAR:
          | ID | Name       | Type        | Brand     | Transmission | Seats | Rate per hour
          | 30 | Model S    | Sports Car  | Tesla     | Automatic    | 2     | 500.0

Please confirm the booking
          | 1   | Confirm and proceed  |
          | 2   | Go back              |
1

On which day do you want to rent your selected car?
Please type in your start date (DD.MM.YYYY): 01.01.2020
See here all available time slots:


        01.01.2020
          |  0: 0 | Available  |
          |  0:30 | Available  |
          |  1: 0 | Available  |
          |  1:30 | Available  |
          |  2: 0 | Available  |
          |  2:30 | Available  |
          |  3: 0 | Available  |
          |  3:30 | Available  |
          |  4: 0 | Available  |
          |  4:30 | Available  |
          |  5: 0 | Available  |
          |  5:30 | Available  |
          |  6: 0 | Available  |
          |  6:30 | Available  |
          |  7: 0 | Available  |
```

Quick Access

Console

<terminated> Application (2) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (07.12.2017, 20:

```
|  8:30 | Available  |
|  9: 0 | Available  |
|  9:30 | Available  |
| 10: 0 | Available  |
| 10:30 | Available  |
| 11: 0 | Available  |
| 11:30 | Available  |
| 12: 0 | Available  |
| 12:30 | Available  |
| 13: 0 | Available  |
| 13:30 | Available  |
| 14: 0 | Available  |
| 14:30 | Available  |
| 15: 0 | Available  |
| 15:30 | Available  |
| 16: 0 | Available  |
| 16:30 | Available  |
| 17: 0 | Available  |
| 17:30 | Available  |
| 18: 0 | Available  |
| 18:30 | Available  |
| 19: 0 | Available  |
| 19:30 | Available  |
| 20: 0 | Available  |
| 20:30 | Available  |
| 21: 0 | Available  |
| 21:30 | Available  |
| 22: 0 | Available  |
| 22:30 | Available  |
| 23: 0 | Available  |
| 23:30 | Available  |


You can rent a car temporarily between 30 minutes and 4 hours.
Please type in your start time (hh:mm): 12:00
Please type in your return time (hh:mm): 15
Please type in your return time (hh:mm): 15:00

Congratulations! As a Bronze member, you get 5% off this booking!

Selected duration:  3: 0h
Calculated price 1425.0

        How would you like to pay?
          | 1   | Credit Card     |
          | 2   | Mobile Pay      |
          | 3   | Cash            |
1
You chose credit card.
Enter your card number: 4916722136319146
Enter your expiry date (MM/YYYY): 01/2020
Enter your CVV (3 digits): 312


        CREDIT CARD:
          | Booking Reference  | Amount   | Payment Method  | Last Four |
          | 56_2017-12-07/30/2 | 1425.0   | Credit Card     | ****9146  |
Thank you for your booking. Enjoy your ride!

Congratulations! As a Silver member, you now get 10% off your next bookings!
```