```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;
        }
    }
```