```
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;
        }

    }
```