A Database for a Car Rental Software System

Project for the class Databases 2
in the Summer Term 2010

Software Technology (Master)

The following persons have contributed to this project:

Radhika Mohan

Priya Swaminathan

Matthias Ruszala

Alexander Weickmann

1 Data Model for the Car Rental System Database

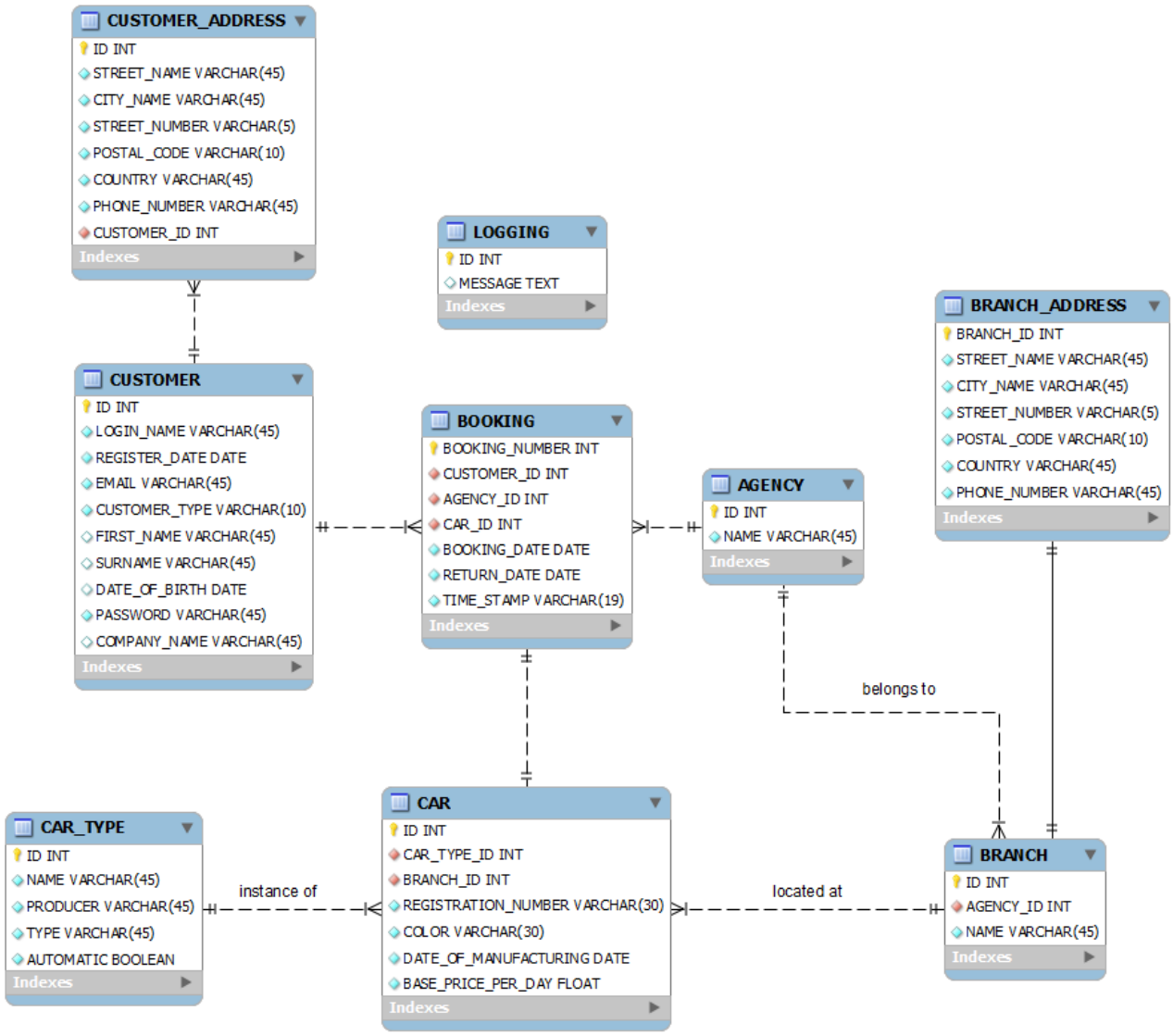
1.1 Explanation of the Data and the Application

Our software can manage different agencies that are able to provide cars to customers. Customers can rent cars for a given time period.

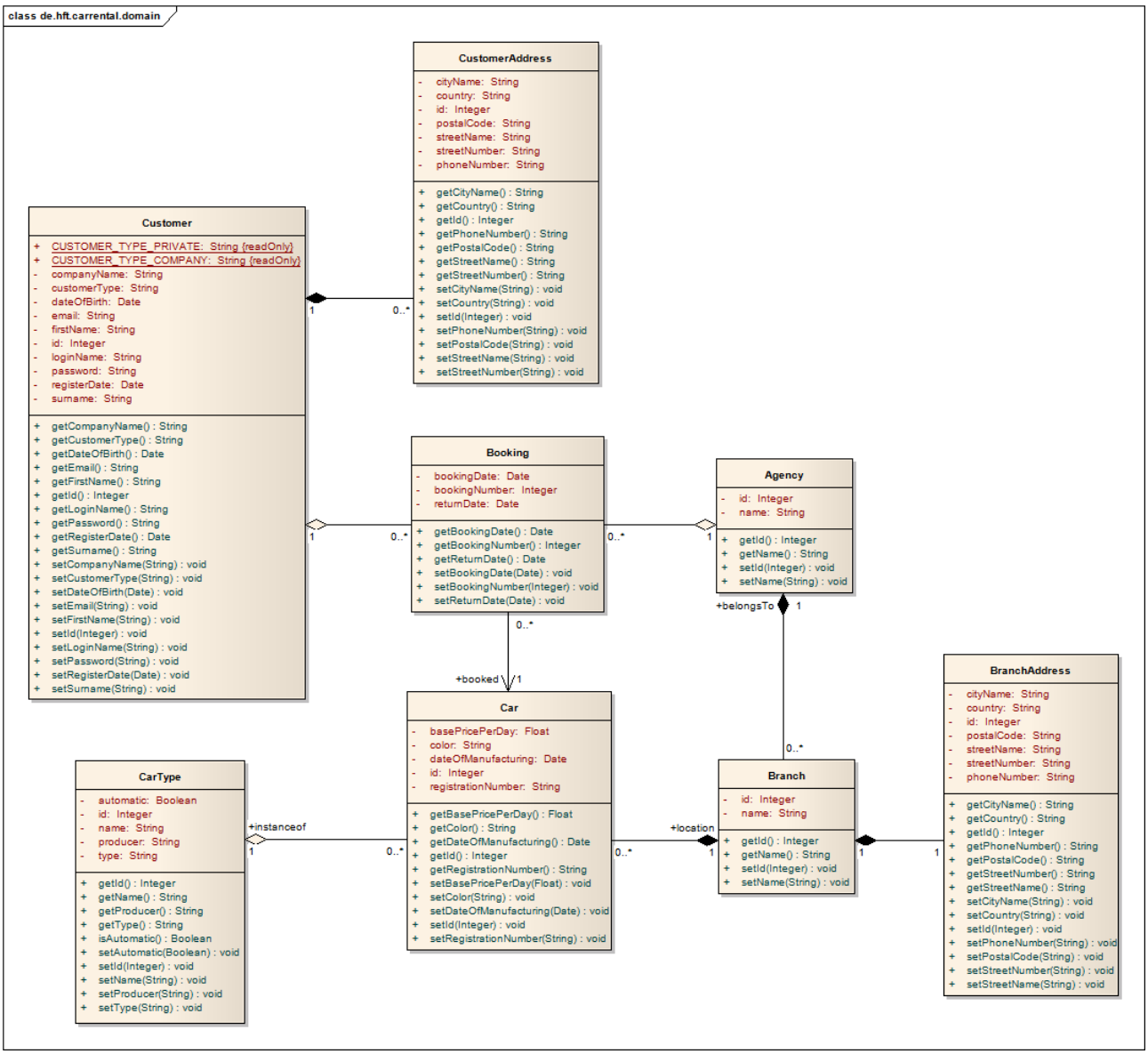
The relevant data contains information about the customers, the bookings, addresses, agencies, cars, branches of agencies and logging.

1.2 The Data Model

1.2.1 ER Model



1.2.2 UML Model



2 Relational Design

2.1 Table Schemas

See 1.2.1

2.2 Normalization

Table: Agency
This table is in **Third Normal Form** (3NF) where the *ID* is the primary key and the other attribute is the *Name*.

Here the data is dependent only on the primary key. Hence, 1NF is satisfied. There are no concatenated/composite primary keys and the other is the non-key attribute of the table which depends wholly on the primary key. Hence 2 NF is also satisfied. Moreover, there are no transitive dependencies on other tables.

Table: Customer
This table is in **Third Normal Form** (3NF) where the *Customer_ID* is the primary key and it avoids transitive dependencies. This eliminates duplication across multiple rows. It also satisfies 2NF.

Table: Customer_Address
Since it has two primary keys cannot determine the normal form??

Table: Booking
This table is in **Third Normal Form** (3NF) where the *Booking_Number_ID* is the primary key and *booking _date* and the *return _date* cannot exist independently. There are also foreign key attributes that come from another table to avoid duplication of data.

Table: Car
This table is in **Second Normal Form** (2NF) where data stored in a table must be dependent only on the primary key and not on any other field in the table. Here we have declared foreign keys which help establish another table ´CarType´.

Table: CarType
This table is in **Third Normal Form** (3NF) where data stored in a table is dependent only on the primary key and not on any other field in the table. Here all the non-key attributes in the table are fully determined by the primary key.

Table: Branch
This table is in **Third Normal Form** (3NF) where the Branch_Id is the primary key and data of the address is stored in a different table to avoid redundancy. Reason same as above for 3NF:

Table: Address:
This table is in **Third Normal Form** (3NF) where the *ID* is the primary key. This eliminates duplication across multiple rows. It also satisfies 2NF.

2.3 Integrity Constraints

Table	Constraint On Delete	Rationale
CUSTOMER_ADDRESS	Cascade	Each address entry is bound to a specific customer by the phone number. It's lifetime is therefore bound to the lifetime the customer it belongs to.
BOOKING	Restrict	This applies to all 3 foreign keys. Since an order is a central piece of information for a car rental system it should not be able to be deleted accidentally (cascade) by deleting either of the associated customer, car or agency. Before any of these can be done outstanding orders have to be resolved first.
BRANCH_ADDRESS	Cascade	Same rationale as for table CUSTOMER_ADDRESS
CAR	Restrict	This applies to both foreign keys. A specific car should obviously not be deleted when its details are deleted. Also cars should not be deleted automatically when the associated branch is deleted as cars could be reassigned to other branches beforehand.

3 System Requirements

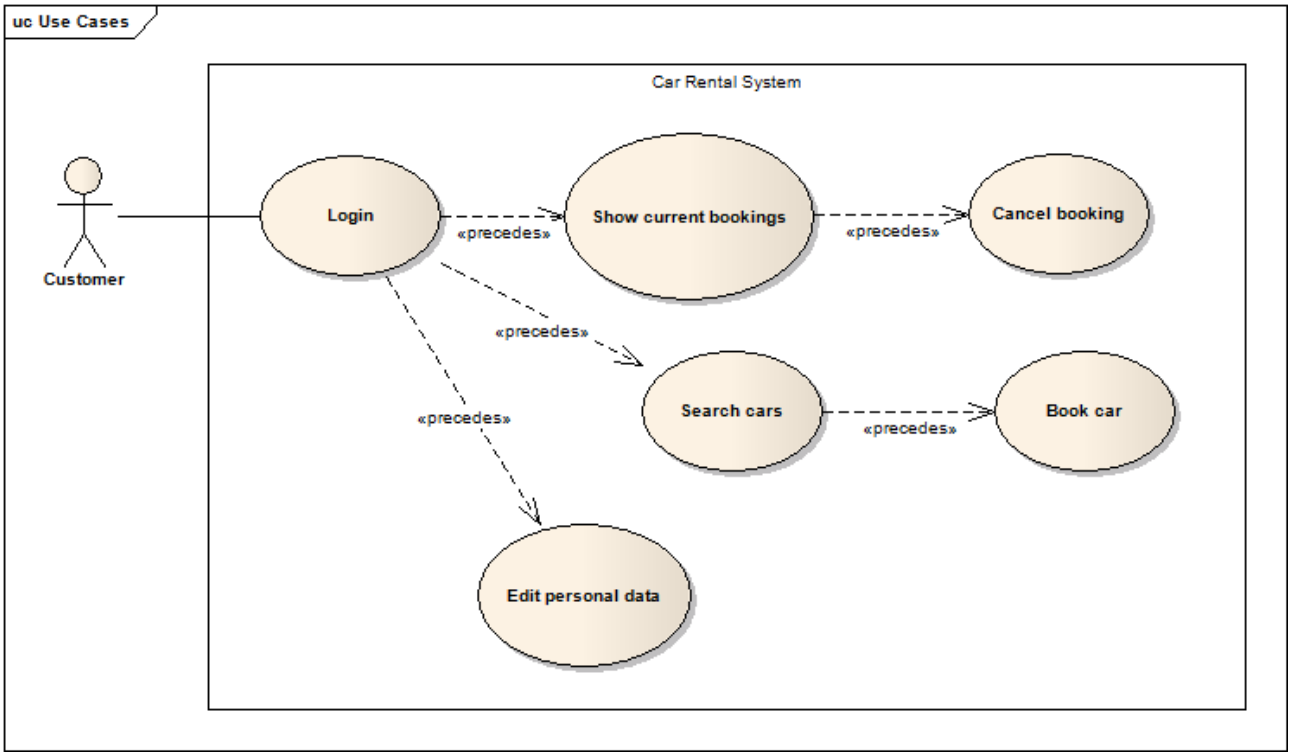
For this project a MySQL 5.1 database was used.

Since the HFT did not manage to set up a database for us and we do not have a server available, each developer had to set up a local database on his computer. That lead to a lot of different operating systems in use, e.g. MAC OS, Windows XP and Windows 7 also differing in 32bit to 64bit architecture.

Additionally, the Car Rental System requires an installed Java Runtime Environment 6.

4 Use Cases / Transactions / Triggers

4.1 Use Cases



Login

A customer may log in to the system. For this he needs to provide his user name.

Show current bookings

A customer may review his current bookings. The current bookings are listed in a table showing the most important information.

Cancel booking

A customer may cancel each individual booking at any time. For this he provides the booking number of the booking to cancel.

Search cars

A customer may search for cars that he's interested in renting. A table shows all the search results.

Book car

A customer may book a car. When doing so, the car is reserved for a given time period for that customer.

Edit personal data

A customer may edit his personal data. This includes changing his email address and adding and editing personal addresses.

4.2 Transactions

Explanation of the transactions not necessary as stated in the Database 2 lecture of 10.06.2010.

Our queries can be found in section 7.2 Query Examples and Transactions however.

4.3 Triggers

4.3.1 Logging

When doing any database action like inserting a new booking or canceling a booking this action is logged as a log entry in a special table shortly describing the transaction.

Fires upon: INSERT / DELETE

Table: Bookings

4.3.2 Server Timestamps

When adding a new booking a booking, the current time stamp should be inserted into the database entry. This time stamp shall correspond to the server clock, not to the client clock.

Fires upon: INSERT

Table: Bookings

5 Physical Design

The Car Rental database consists of the following tables:

1. Customer Table - Contains information about the customers
2. Customer_Address Table - Contains the foreign keys customer_id and address_id to link the customer and address tables

Address Table - Contains information about the address details of the customer.

Booking Table - Contains the customer booking details for cars.

Agency Table - Contains the information about car rental agencies

Branch Table - Contains the information about the different agencies at various branch locations.

Car Table - Contains the information about the cars and agencies associated with cars.

Car_Type Table - Contains the information about the types of car available.

This table below lists some columns in the car rental database:

Table Name	Columns	Data type	Details
Customer	ID	INT	Primary Key,Not null, Auto Incremented
	Login_Name,Email,Name	Varchar(length)	
	DOB	Date	
	Password	?	

Every table has columns and has a data type associated with it. For example, in the Customer table we choose the data type INT for Customer_id (primary key); by choosing INT - the column can have numeric values which are unique for every customer.

Columns like Name, Email can have any length of values; to effectively use the disk space we use the datatype Varchar

Missing values: Certain columns cannot hold missing values and hence those are added a Not Null constraint.

Quantity of Data: We assume each table will hold 100 records initially.

Types of access: The database supports the following types of access: Insert,Update, Modify and Delete for all tables.

Examples:

1. Creating a new customer will insert new customer records into the customer table.

2. Updating the address details of a existing customer will access the customer and address tables.

File Organizations:

Query1: To list the current bookings for a given customer.

```
SELECT
      BOOKING.BOOKING_NUMBER, BOOKING.BOOKING_DATE,
BOOKING.RETURN_DATE, AGENCY.NAME, CAR.REGISTRATION_NUMBER,
CAR_TYPE.NAME
FROM
        BOOKING
JOIN AGENCY ON
        BOOKING.AGENCY_ID = AGENCY.ID
JOIN CAR ON
        CAR.ID = BOOKING.CAR_ID
JOIN CAR_TYPE ON
        CAR_TYPE.ID = CAR.CAR_TYPE_ID
WHERE
        BOOKING.CUSTOMER_ID = ?
```

The above query involves data access from many tables and hence a JOIN between the tables and setting indexes on the primary key columns will help in fast data retrieval.

6 List of References

No additional references were used.

7 Appendix

7.1 Database Tables with Data

7.1.1 AGENCY

ID	NAME
1	First Agency
2	Agency Bond

7.1.2 BOOKING

BOOKING_NUMBER	CUSTOMER_ID	AGENCY_ID	CAR_ID	BOOKING_DATE	RETURN_DATE	TIME_STAMP
1	1	1	1	01.01.02	01.02.02	15.06.10 20:10
2	2	1	2	22.04.09	23.04.09	15.06.10 20:10
3	3	2	1	02.02.03	20.02.03	15.06.10 20:10
4	4	2	2	20.05.45	21.05.45	15.06.10 20:10
8	1	1	4	03.02.00	03.03.02	15.06.10 20:10
9	1	2	3	02.02.02	03.02.02	15.06.10 20:10
10	1	2	4	03.02.02	04.02.02	15.06.10 20:12

7.1.3 BRANCH

ID	AGENCY_ID	NAME
1	1	Spain
2	1	Germany
3	2	Germany
4	2	USA

7.1.4 BRANCH_ADDRESS

BRANCH_ID	STREET_NAME	CITY_NAME	STREET_NUMBER	POSTAL_CODE	COUNTRY	PHONE_NUMBER
1	Spain Street	Madrid	32	443020S	Spain	2020 / 2399
2	New Street	Hamburg	43	343023	Germany	903403
3	Other Street	Frankfurt	33a	9402	Germany	232393
4	Liberty Street	New York	53a	123344	USA	84398

7.1.5 CAR

ID	CAR_TYPE_ID	BRANCH_ID	REGISTRATION_NUMBER	COLOR	DATE_OF_MANUFACTURING	BASE_PRICE_PER_DAY
1	1	1	X23-234	green	01.02.99	20
2	2	2	BC-343	red	21.02.02	15
3	3	3	TW-435	blue	11.03.01	23
4	4	4	232-444	pink	03.04.45	100

7.1.6 CAR_TYPE

ID	NAME	PRODUCER	TYPE	AUTOMATIC
1	Kaefer	VW	PKW	0
2	E90	BMW	PKW	1
3	Fiesta	Ford	PKW	0
4	Tank	Army	Military	0

7.1.7 CUSTOMER

ID	LOGIN_NAME	REGISTER_DATE	EMAIL	CUSTOMER_TYPE	FIRST_NAME	SURNAME	DATE_OF_BIRTH	PASSWORD	COMPANY_NAME
1	Alex	06.06.10	alex@hft.de	private	Alexander	Weickmann	01.01.00	none	
2	Matthias	02.02.08	matze@hft.de	private	Matthias	Ruszala	02.01.00	none	
3	Priya	03.06.99	priya@hft.com	private	Priya	S	05.03.99	none	
4	Radhika	05.04.03	radhika@hft.com	private	Radhika	Mohan	22.04.38	none	
5	HFT	04.02.01	hft@stuttgart.de	company				none	HFT Stuttgart

7.1.8 CUSTOMER_ADDRESS

ID	STREET_NAME	CITY_NAME	STREET_NUMBER	POSTAL_CODE	COUNTRY	PHONE_NUMBER	CUSTOMER_ID
2	Matthias Street	Matthias City	11	3254	Germany	23443	2
3	Priya Street	Priya City	23	347687	India	6436547	3
4	Radhika Street	Radhika City	443	8673	India	5434	4
5	HFT Street	HFT City	32a	6342	Germany	3423 / 3432	5
7	Alex Street	Alex Town	49	4903	Alex Country	90340	1

7.1.9 LOGGING

ID	MESSAGE
1	Booking was deleted.
2	New booking was inserted.
3	New booking was inserted.

7.2 Query Examples and Transactions

7.2.1 List all available cars for a given date and location

```
SELECT CAR_TYPE.NAME, CAR_TYPE.TYPE, CAR.BASE_PRICE_PER_DAY,
CAR.REGISTRATION_NUMBER, AGENCY.NAME AS AGENCY_NAME, BRANCH.NAME AS
BRANCH_NAME

FROM CAR

JOIN CAR_TYPE ON CAR_TYPE.ID = CAR.CAR_TYPE_ID

JOIN BRANCH ON BRANCH.ID = CAR.BRANCH_ID

JOIN BRANCH_ADDRESS ON BRANCH_ADDRESS.BRANCH_ID = BRANCH.ID

JOIN AGENCY ON AGENCY.ID = BRANCH.AGENCY_ID

WHERE CAR.ID NOT IN

(SELECT BOOKING.CAR_ID FROM BOOKING WHERE BOOKING.BOOKING_DATE <= '2002-
02-01' AND BOOKING.RETURN_DATE >= '2002-02-30')

AND BRANCH_ADDRESS.CITY_NAME = 'New York'
```

NAME	TYPE	BASE_PRICE_PER_DAY	REGISTRATION_NUMBER	AGENCY_NAME	BRANCH_NAME
Tank	Military	100	232-444	Agency Bond	USA

7.2.2 List all current bookings for a given customer

SELECT BOOKING.BOOKING_NUMBER, BOOKING.BOOKING_DATE, BOOKING.RETURN_DATE, AGENCY.NAME, CAR.REGISTRATION_NUMBER, CAR_TYPE.NAME

FROM BOOKING

JOIN AGENCY ON BOOKING.AGENCY_ID = AGENCY.ID

JOIN CAR ON CAR.ID = BOOKING.CAR_ID

JOIN CAR_TYPE ON CAR_TYPE.ID = CAR.CAR_TYPE_ID

WHERE BOOKING.CUSTOMER_ID = '1'

BOOKING_NUMBER	BOOKING_DATE	RETURN_DATE	NAME	REGISTRATION_NUMBER	NAME
1	01.01.02	01.02.02	First Agency	X23-234	Kaefer
8	03.02.00	03.03.02	First Agency	232-444	Tank
9	02.02.02	03.02.02	Agency Bond	TW-435	Fiesta
10	03.02.02	04.02.02	Agency Bond	232-444	Tank

7.2.3 Retrieve details for a given customer

SELECT * FROM CUSTOMER WHERE CUSTOMER.ID = 1

ID	LOGIN_NAME	REGISTER_DATE	EMAIL	CUSTOMER_TYPE	FIRST_NAME	SURNAME	DATE_OF_BIRTH	PASSWORD	COMPANY_NAME
1	Alex	06.06.10	alex@hft.de	private	Alexander	Weickmann	01.01.00	none	

7.2.4 List all addresses for a given customer

SELECT CUSTOMER.FIRST_NAME, CUSTOMER.SURNAME, CUSTOMER_ADDRESS.STREET_NAME, CUSTOMER_ADDRESS.STREET_NUMBER

FROM CUSTOMER

JOIN CUSTOMER_ADDRESS ON CUSTOMER_ADDRESS.CUSTOMER_ID = CUSTOMER.ID

WHERE CUSTOMER.ID = 1

FIRST_NAME	SURNAME	STREET_NAME	STREET_NUMBER
Alexander	Weickmann	Alex Street	49

7.2.5 List all car types

SELECT CAR_TYPE.NAME, CAR_TYPE.PRODUCER

FROM CAR_TYPE

NAME	PRODUCER
Kaefer	VW
E90	BMW
Fiesta	Ford
Tank	Army

7.2.6 List all agencies

SELECT AGENCY.NAME

FROM AGENCY

NAME
First Agency
Agency Bond

7.2.7 List all bookings for a given date and branch

SELECT CAR.REGISTRATION_NUMBER, CUSTOMER.FIRST_NAME, CUSTOMER.SURNAME, CUSTOMER.EMAIL, BOOKING.BOOKING_DATE, BOOKING.RETURN_DATE

FROM BOOKING

JOIN CUSTOMER ON BOOKING.CUSTOMER_ID = CUSTOMER.ID

JOIN CAR ON BOOKING.CAR_ID = CAR.ID

JOIN BRANCH ON CAR.BRANCH_ID = BRANCH.ID

WHERE BRANCH.ID = 1 AND BOOKING.BOOKING_DATE = '2002-01-01'

REGISTRATION_NUMBER	FIRST_NAME	SURNAME	EMAIL	BOOKING_DATE	RETURN_DATE
X23-234	Alexander	Weickmann	alex@hft.de	01.01.02	01.02.02

7.3 Description of the Application and User Interface

7.3.1 SessionManager.java

```
package de.hft.carrental.database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

/**
 * The session manager is a singleton that enables clients to retrieve sessions
 * from everywhere in the code.
 *
 * @author Alexander Weickmann
 */
public final class SessionManager {

	private static final SessionManager instance = new SessionManager();

	private final AnnotationConfiguration configuration;

	private final SessionFactory sessionFactory;

	private Session session;

	private SessionManager() {
		configuration = new AnnotationConfiguration()
				.configure("hibernate.cfg.xml");
		sessionFactory = configuration.buildSessionFactory();
```

```
            openSession();
      }

      public static SessionManager getInstance() {
            return instance;
      }

      public Session openSession() {
            if ((session == null) || !session.isOpen()) {
                  if (!(isDatabaseConnectionAvailable())) {
                        return null;
                  }
                  session = sessionFactory.openSession();
            }

            return session;
      }

      public void closeSession() {
            if (session == null) {
                  return;
            }

            session.flush();
            session.close();
            session = null;
      }

      public void dispose() {
            sessionFactory.close();
      }

      private boolean isDatabaseConnectionAvailable() {
            String url = configuration.getProperty("hibernate.connection.url");
            String dbuser = configuration
                        .getProperty("hibernate.connection.username");
            String dbpassword = configuration
                        .getProperty("hibernate.connection.password");
            try {
                  Connection connection = DriverManager.getConnection(url,
dbuser,
                              dbpassword);
                  return (connection == null) ? false : true;
            } catch (SQLException e) {
                  return false;
            }
      }

}
```

7.3.2 Agency.java

```
package de.hft.carrental.domain;

import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
```

```
import javax.persistence.Table;

@Entity
@Table(name = "AGENCY")
public final class Agency {

	private Integer id;

	private String name;

	private Set<Branch> branches;

	private Set<Booking> bookings;

	@Id
	@GeneratedValue
	@Column(name = "ID", updatable = false, nullable = false, length = 45)
	public Integer getId() {
		return id;
	}

	@Column(name = "NAME", updatable = true, nullable = false)
	public String getName() {
		return name;
	}

	@OneToMany(cascade = CascadeType.ALL, mappedBy = "agency", orphanRemoval =
true, targetEntity = Branch.class)
	public Set<Branch> getBranches() {
		return branches;
	}

	@OneToMany(cascade = CascadeType.ALL, mappedBy = "agency", orphanRemoval =
true, targetEntity = Booking.class)
	public Set<Booking> getBookings() {
		return bookings;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public void setName(String name) {
		this.name = name;
	}

	public void setBranches(Set<Branch> branches) {
		this.branches = branches;
	}

	public void setBookings(Set<Booking> bookings) {
		this.bookings = bookings;
	}
}
```

7.3.3 Booking.java

```
package de.hft.carrental.domain;

import java.util.Date;

import javax.persistence.Column;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "BOOKING")
public final class Booking {

	private Date bookingDate;

	private Integer bookingNumber;

	private Date returnDate;

	private Customer customer;

	private Agency agency;

	private Car car;

	@Column(name = "BOOKING_DATE", updatable = true, nullable = false)
	public Date getBookingDate() {
		return bookingDate;
	}

	@Id
	@GeneratedValue
	@Column(name = "BOOKING_NUMBER", updatable = false, nullable = false)
	public Integer getBookingNumber() {
		return bookingNumber;
	}

	@Column(name = "RETURN_DATE", updatable = true, nullable = false)
	public Date getReturnDate() {
		return returnDate;
	}

	@ManyToOne(optional = false, targetEntity = Customer.class)
	@JoinColumn(name = "CUSTOMER_ID", updatable = false, nullable = false,
referencedColumnName = "ID")
	public Customer getCustomer() {
		return customer;
	}

	@ManyToOne(optional = false, targetEntity = Agency.class)
	@JoinColumn(name = "AGENCY_ID", updatable = false, nullable = false,
referencedColumnName = "ID")
	public Agency getAgency() {
		return agency;
	}

	@ManyToOne(optional = false, targetEntity = Car.class)
	@JoinColumn(name = "CAR_ID", updatable = false, nullable = false,
referencedColumnName = "ID")
	public Car getCar() {
		return car;
	}

	public void setBookingDate(Date bookingDate) {
		this.bookingDate = bookingDate;
	}
```

```
	public void setBookingNumber(Integer bookingNumber) {
		this.bookingNumber = bookingNumber;
	}

	public void setReturnDate(Date returnDate) {
		this.returnDate = returnDate;
	}

	public void setCustomer(Customer customer) {
		this.customer = customer;
	}

	public void setAgency(Agency agency) {
		this.agency = agency;
	}

	public void setCar(Car car) {
		this.car = car;
	}

}
```

7.3.4 Branch.java

```
package de.hft.carrental.domain;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "BRANCH")
public final class Branch {

	private Integer id;

	private String name;

	private Agency agency;

	private BranchAddress branchAddress;

	@Id
	@GeneratedValue
	@Column(name = "ID", updatable = false, nullable = false)
	public Integer getId() {
		return id;
	}

	@Column(name = "NAME", updatable = true, nullable = false, length = 45)
	public String getName() {
		return name;
	}

	@ManyToOne(optional = false, targetEntity = Agency.class)
```

```
	@JoinColumn(name = "AGENCY_ID", updatable = false, nullable = false,
referencedColumnName = "ID")
	public Agency getAgency() {
		return agency;
	}

	@OneToOne(cascade = CascadeType.ALL, mappedBy = "branch", optional =
false, orphanRemoval = true, targetEntity = BranchAddress.class)
	public BranchAddress getBranchAddress() {
		return branchAddress;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public void setName(String name) {
		this.name = name;
	}

	public void setAgency(Agency agency) {
		this.agency = agency;
	}

	public void setBranchAddress(BranchAddress address) {
		branchAddress = address;
	}
}
```

7.3.5 BranchAddress.java

```
package de.hft.carrental.domain;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table(name = "BRANCH_ADDRESS")
public final class BranchAddress {

	private String cityName;

	private String country;

	private Integer id;

	private String phoneNumber;

	private String postalCode;

	private String streetName;

	private String streetNumber;

	private Branch branch;
```

```
	@Column(name = "CITY_NAME", updatable = false, nullable = false, length =
45)
	public String getCityName() {
		return cityName;
	}

	@Column(name = "COUNTRY", updatable = false, nullable = false, length =
45)
	public String getCountry() {
		return country;
	}

	@Id
	@GeneratedValue
	@Column(name = "BRANCH_ID", updatable = false, nullable = false)
	public Integer getId() {
		return id;
	}

	@Column(name = "PHONE_NUMBER", updatable = true, nullable = false, length
= 45)
	public String getPhoneNumber() {
		return phoneNumber;
	}

	@Column(name = "POSTAL_CODE", updatable = true, nullable = false, length =
10)
	public String getPostalCode() {
		return postalCode;
	}

	@Column(name = "STREET_NAME", updatable = true, nullable = false, length =
45)
	public String getStreetName() {
		return streetName;
	}

	@Column(name = "STREET_NUMBER", updatable = true, nullable = false, length
= 5)
	public String getStreetNumber() {
		return streetNumber;
	}

	@OneToOne(cascade = CascadeType.ALL, optional = false, targetEntity =
Branch.class)
	@JoinColumn(name = "BRANCH_ID", unique = true, updatable = false, nullable
= false)
	public Branch getBranch() {
		return branch;
	}

	public void setCityName(String cityName) {
		this.cityName = cityName;
	}

	public void setCountry(String country) {
		this.country = country;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public void setPhoneNumber(String phoneNumber) {
```

```
		this.phoneNumber = phoneNumber;
	}

	public void setPostalCode(String postalCode) {
		this.postalCode = postalCode;
	}

	public void setStreetName(String streetName) {
		this.streetName = streetName;
	}

	public void setStreetNumber(String streetNumber) {
		this.streetNumber = streetNumber;
	}

	public void setBranch(Branch branch) {
		this.branch = branch;
	}

}
```

7.3.6 Car.java

```
package de.hft.carrental.domain;

import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "CAR")
public final class Car {

	private Float basePricePerDay;

	private String color;

	private Date dateOfManufacturing;

	private Integer id;

	private String registrationNumber;

	private CarType carType;

	private Branch branch;

	private Set<Booking> bookings;

	@Column(name = "BASE_PRICE_PER_DAY", updatable = true, nullable = false)
	public Float getBasePricePerDay() {
		return basePricePerDay;
	}
```

```
	@Column(name = "COLOR", updatable = true, nullable = false, length = 30)
	public String getColor() {
		return color;
	}

	@Column(name = "DATE_OF_MANUFACTURING", updatable = false, nullable =
false)
	public Date getDateOfManufacturing() {
		return dateOfManufacturing;
	}

	@Id
	@GeneratedValue
	@Column(name = "ID", updatable = false, nullable = false)
	public Integer getId() {
		return id;
	}

	@Column(name = "REGISTRATION_NUMBER", updatable = false, nullable = false)
	public String getRegistrationNumber() {
		return registrationNumber;
	}

	@ManyToOne(optional = false, targetEntity = CarType.class)
	@JoinColumn(name = "CAR_TYPE_ID", updatable = false, nullable = false,
referencedColumnName = "ID")
	public CarType getCarType() {
		return carType;
	}

	@ManyToOne(optional = false, targetEntity = Branch.class)
	@JoinColumn(name = "BRANCH_ID", updatable = true, nullable = false,
referencedColumnName = "ID")
	public Branch getBranch() {
		return branch;
	}

	@OneToMany(cascade = CascadeType.ALL, mappedBy = "car", targetEntity =
Booking.class)
	public Set<Booking> getBookings() {
		return bookings;
	}

	public void setBasePricePerDay(Float basePricePerDay) {
		this.basePricePerDay = basePricePerDay;
	}

	public void setColor(String color) {
		this.color = color;
	}

	public void setDateOfManufacturing(Date dateOfManufacturing) {
		this.dateOfManufacturing = dateOfManufacturing;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public void setRegistrationNumber(String registrationNumber) {
		this.registrationNumber = registrationNumber;
	}
```

```
	public void setCarType(CarType carType) {
		this.carType = carType;
	}

	public void setBranch(Branch branch) {
		this.branch = branch;
	}

	public void setBookings(Set<Booking> bookings) {
		this.bookings = bookings;
	}
}
```

7.3.7 CarType.java

```
package de.hft.carrental.domain;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "CAR_TYPE")
public final class CarType {

	private Boolean automatic;

	private Integer id;

	private String name;

	private String producer;

	private String type;

	@Column(name = "AUTOMATIC", updatable = true, nullable = false)
	public Boolean getAutomatic() {
		return automatic;
	}

	@Id
	@GeneratedValue
	@Column(name = "ID", updatable = false, nullable = false)
	public Integer getId() {
		return id;
	}

	@Column(name = "NAME", updatable = true, nullable = false)
	public String getName() {
		return name;
	}

	@Column(name = "PRODUCER", updatable = true, nullable = false)
	public String getProducer() {
		return producer;
	}

	@Column(name = "TYPE", updatable = true, nullable = false)
	public String getType() {
		return type;
```

```
	}

	public void setAutomatic(Boolean automatic) {
		this.automatic = automatic;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public void setName(String name) {
		this.name = name;
	}

	public void setProducer(String producer) {
		this.producer = producer;
	}

	public void setType(String type) {
		this.type = type;
	}

}
```

7.3.8 Customer.java

```
package de.hft.carrental.domain;

import java.util.Date;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "CUSTOMER")
public final class Customer {

	public static final String CUSTOMER_TYPE_PRIVATE = "private";

	public static final String CUSTOMER_TYPE_COMPANY = "company";

	private String companyName;

	private String customerType;

	private Date dateOfBirth;

	private String email;

	private String firstName;

	private Integer id;

	private String loginName;

	private String password;
```

```
	private Date registerDate;

	private String surname;

	private Set<CustomerAddress> customerAddresses;

	private Set<Booking> bookings;

	@Column(name = "COMPANY_NAME", updatable = true, nullable = true)
	public String getCompanyName() {
		return companyName;
	}

	@Column(name = "CUSTOMER_TYPE", updatable = true, nullable = false)
	public String getCustomerType() {
		return customerType;
	}

	@Column(name = "DATE_OF_BIRTH", updatable = true, nullable = true)
	public Date getDateOfBirth() {
		return dateOfBirth;
	}

	@Column(name = "EMAIL", updatable = true, nullable = false, length = 45)
	public String getEmail() {
		return email;
	}

	@Column(name = "FIRST_NAME", updatable = true, nullable = true, length =
45)
	public String getFirstName() {
		return firstName;
	}

	@Id
	@GeneratedValue
	@Column(name = "ID", updatable = false, nullable = false)
	public Integer getId() {
		return id;
	}

	@Column(name = "LOGIN_NAME", updatable = false, nullable = false, length =
45)
	public String getLoginName() {
		return loginName;
	}

	@Column(name = "PASSWORD", updatable = true, nullable = false, length =
45)
	public String getPassword() {
		return password;
	}

	@Column(name = "REGISTER_DATE", updatable = false, nullable = false)
	public Date getRegisterDate() {
		return registerDate;
	}

	@Column(name = "SURNAME", updatable = true, nullable = true, length = 45)
	public String getSurname() {
		return surname;
	}
```

```
	@OneToMany(cascade = CascadeType.ALL, mappedBy = "customer", orphanRemoval
= true, targetEntity = CustomerAddress.class)
	public Set<CustomerAddress> getCustomerAddresses() {
		return customerAddresses;
	}

	@OneToMany(cascade = CascadeType.ALL, mappedBy = "customer", orphanRemoval
= true, targetEntity = Booking.class)
	public Set<Booking> getBookings() {
		return bookings;
	}

	public void setCompanyName(String companyName) {
		this.companyName = companyName;
	}

	public void setCustomerType(String customerType) {
		this.customerType = customerType;
	}

	public void setDateOfBirth(Date dateOfBirth) {
		this.dateOfBirth = dateOfBirth;
	}

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

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

	public void setId(Integer id) {
		this.id = id;
	}

	public void setLoginName(String loginName) {
		this.loginName = loginName;
	}

	public void setPassword(String password) {
		this.password = password;
	}

	public void setRegisterDate(Date registerDate) {
		this.registerDate = registerDate;
	}

	public void setSurname(String surname) {
		this.surname = surname;
	}

	public void setBookings(Set<Booking> bookings) {
		if (this.bookings == null) {
			this.bookings = bookings;
		} else {
			this.bookings.clear();
			this.bookings.addAll(bookings);
		}
	}

	public void setCustomerAddresses(Set<CustomerAddress> customerAddresses) {
		if (this.customerAddresses == null) {
			this.customerAddresses = customerAddresses;
```

```
			} else {
				this.customerAddresses.clear();
				this.customerAddresses.addAll(customerAddresses);
			}
		}
}
```

7.3.9 CustomerAddress.java

```
package de.hft.carrental.domain;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "CUSTOMER_ADDRESS")
public final class CustomerAddress {

	private String cityName;

	private String country;

	private Integer id;

	private String phoneNumber;

	private String postalCode;

	private String streetNumber;

	private String streetName;

	private Customer customer;

	@Column(name = "CITY_NAME", updatable = true, nullable = false, length =
45)
	public String getCityName() {
		return cityName;
	}

	@Column(name = "COUNTRY", updatable = true, nullable = false, length = 45)
	public String getCountry() {
		return country;
	}

	@Id
	@GeneratedValue
	@Column(name = "ID", updatable = false, nullable = false)
	public Integer getId() {
		return id;
	}

	@Column(name = "PHONE_NUMBER", updatable = true, nullable = false, length
= 45)
	public String getPhoneNumber() {
		return phoneNumber;
	}
```

```
	@Column(name = "POSTAL_CODE", updatable = true, nullable = false, length =
10)
	public String getPostalCode() {
		return postalCode;
	}

	@Column(name = "STREET_NAME", updatable = true, nullable = false, length =
45)
	public String getStreetName() {
		return streetName;
	}

	@Column(name = "STREET_NUMBER", updatable = true, nullable = false, length
= 5)
	public String getStreetNumber() {
		return streetNumber;
	}

	@ManyToOne(optional = false, targetEntity = Customer.class)
	@JoinColumn(name = "CUSTOMER_ID", updatable = false, nullable = false,
referencedColumnName = "ID")
	public Customer getCustomer() {
		return customer;
	}

	public void setCityName(String cityName) {
		this.cityName = cityName;
	}

	public void setCountry(String country) {
		this.country = country;
	}

	public void setId(Integer id) {
		this.id = id;
	}

	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}

	public void setPostalCode(String postalCode) {
		this.postalCode = postalCode;
	}

	public void setStreetNumber(String streetNumber) {
		this.streetNumber = streetNumber;
	}

	public void setStreetName(String streetName) {
		this.streetName = streetName;
	}

	public void setCustomer(Customer customer) {
		this.customer = customer;
	}

}
```

7.3.10 Main.java

```
package de.hft.carrental.main;

import de.hft.carrental.ui.splash.SplashWindow;

/**
 * This class contains the main method allowing the program to be executed.
 *
 * @author Alexander Weickmann
 */
public final class Main {

	/**
	 * This main method opens up the splash window.
	 */
	public static void main(String[] args) {
		new SplashWindow();
	}

}
```

7.3.11 Window.java

```
package de.hft.carrental.ui;

import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.Toolkit;

import javax.swing.JFrame;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;

import de.hft.carrental.ui.util.GridBagUtil;

/**
 * This class represents an application window. In addition to a normal
 * {@link JFrame} it uses the native look and feel of the used OS. The window
 * will also be positioned in the center of the screen by default. The title of
 * the window as well as the default close operation will be already set.
 * <p>
 * Furthermore, a grid layout is assigned to the window. A convenience method is
 * offered to subclasses that enables rapid creation of
 * {@link GridBagConstraints}.
 * <p>
 * Also, the concept of a current {@link WindowPage} is introduced. Using the
 * method {@link #switchPageTo(WindowPage)} subclasses can switch to another
 * window page at any time.
 *
 * @author Alexander Weickmann
 */
public abstract class Window extends JFrame {

	private static final String WINDOW_TITLE = "Car Rental System";

	private static final long serialVersionUID = 50501854038887694 34L;

	/**
	 * The {@link WindowPage} that is currently displayed.
```

```
 */
private WindowPage currentPage;

protected Window() {
	setNativeLookAndFeel();
	setDefaultCloseOperation(EXIT_ON_CLOSE);
	setTitle(WINDOW_TITLE);

	centerOnScreen();

	createLayout();
}

private void centerOnScreen() {
	Toolkit toolKit = Toolkit.getDefaultToolkit();
	Dimension screenSize = toolKit.getScreenSize();
	int screenHeight = screenSize.height;
	int screenWidth = screenSize.width;
	int x = (screenWidth / 2) - (getMinWidth() / 2);
	int y = (screenHeight / 2) - (getMinHeight() / 2);
	setMinimumSize(new Dimension(getMinWidth(), getMinHeight()));
	setLocation(x, y);
}

private void createLayout() {
	GridBagLayout layout = new GridBagLayout();
	setLayout(layout);
}

/**
 * Switches to the provided {@link WindowPage}. Causes the window to be
 * repainted so the contents of the new page are shown immediately.
 *
 * @param page
 *            The {@link WindowPage} to switch to.
 */
protected final void switchPageTo(WindowPage page) {
	if (currentPage != null) {
		remove(currentPage);
	}

	add(page, GridBagUtil.createGridBagConstraints(0, 1, 1, 1,
			GridBagConstraints.BOTH, new Insets(10, 0, 0, 0),
			GridBagConstraints.FIRST_LINE_START, 0, 0));

	currentPage = page;
	currentPage.refresh();
	validate();
	repaint();
}

/**
 * Must return the minimum width this window shall have.
 */
protected abstract int getMinWidth();

/**
 * Must return the minimum height this window shall have.
 */
protected abstract int getMinHeight();

private void setNativeLookAndFeel() {
	try {
```

```
		UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
			} catch (ClassNotFoundException e) {
				e.printStackTrace();
			} catch (InstantiationException e) {
				e.printStackTrace();
			} catch (IllegalAccessException e) {
				e.printStackTrace();
			} catch (UnsupportedLookAndFeelException e) {
				e.printStackTrace();
			}
		}

}
```

7.3.12 WindowPage.java

```
package de.hft.carrental.ui;

import java.awt.Container;
import java.awt.GridLayout;
import java.util.ArrayList;
import java.util.List;

/**
 * Window pages are used by the class {@link Window}. One window page bundles
 * together a number of {@link WindowPageSection}s. For example, there could
 * exist a window page with two window page sections. One section shows a search
 * formula, while the second section shows a table with the search results.
 *
 * @see Window
 *
 * @author Alexander Weickmann
 */
public abstract class WindowPage extends Container {

	private static final long serialVersionUID = -7371192976582192750L;

	/** The {@link Window} this window page belongs to. */
	private final Window window;

	private final List<WindowPageSection> sections;

	/**
	 * @param window
	 *            The {@link Window} this window page will belong to.
	 * @param layoutRows
	 *            Specifies the number of how many rows the page's grid layout
	 *            consists of.
	 * @param layoutColumns
	 *            Specifies the number of how many columns the page's grid
	 *            layout consists of.
	 */
	protected WindowPage(Window window, int layoutRows, int layoutColumns) {
		super();

		this.window = window;
		sections = new ArrayList<WindowPageSection>();

		createLayout(layoutRows, layoutColumns);
		addSections();
	}
```

```
	private void createLayout(int rows, int columns) {
		GridLayout layout = new GridLayout(rows, columns);
		setLayout(layout);
	}

	/**
	 * Responsible for adding the necessary {@link WindowPageSection}s to this
	 * page.
	 */
	protected abstract void addSections();

	/**
	 * Adds the given {@link WindowPageSection} to this window page.
	 *
	 * @param section
	 *            The window page section to add.
	 */
	protected final void addSection(WindowPageSection section) {
		add(section);
		sections.add(section);
	}

	/**
	 * Returns the {@link Window} this window page belongs to.
	 */
	public final Window getWindow() {
		return window;
	}

	/**
	 * Refreshes the contents of this page by refreshing all of it's sections.
	 */
	public final void refresh() {
		for (WindowPageSection section : sections) {
			section.refresh();
		}
	}

}
```

7.3.13 WindowPageSection.java

```
package de.hft.carrental.ui;

import java.awt.Font;

import javax.swing.BorderFactory;
import javax.swing.JPanel;
import javax.swing.border.TitledBorder;

/**
 * This class is like a {@link JPanel}. In addition, it has a title and a
 * border. The title is displayed at the top left of the border. A window page
 * section always belongs to a specific {@link WindowPage}. The contents of the
 * section can be refreshed using the method {@link #refresh()}.
 *
 * @author Alexander Weickmann
 */
public abstract class WindowPageSection extends JPanel {

	private static final long serialVersionUID = 7237705902963031893L;
```

```
	/**
	 * The {@link WindowPage} this window page section belongs to.
	 */
	private final WindowPage windowPage;

	/**
	 * @param windowPage
	 *            The {@link WindowPage} this window page sections belongs to.
	 * @param title
	 *            The title of this window page section that will be displayed
	 *            at the top left of the section's border.
	 */
	protected WindowPageSection(WindowPage windowPage, String title) {
		this.windowPage = windowPage;
		createBorder(title);
	}

	private void createBorder(String title) {
		TitledBorder border = BorderFactory.createTitledBorder(title + ":");
		border.setTitleFont(new Font("Arial", Font.BOLD, 11));
		setBorder(border);
	}

	/**
	 * Returns the {@link WindowPage} this window page section belongs to.
	 */
	public final WindowPage getWindowPage() {
		return windowPage;
	}

	/**
	 * Refreshes the contents of this window page section.
	 */
	protected abstract void refresh();

}
```

7.3.14 MainWindow.java

```
package de.hft.carrental.ui.main;

import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JPanel;

import de.hft.carrental.database.SessionManager;
import de.hft.carrental.domain.Customer;
import de.hft.carrental.ui.Window;
import de.hft.carrental.ui.main.bookings.CurrentBookingsPage;
import de.hft.carrental.ui.main.cars.CarsPage;
import de.hft.carrental.ui.main.personal.PersonalPage;
import de.hft.carrental.ui.splash.SplashWindow;
import de.hft.carrental.ui.util.GridBagUtil;

/**
 * The main window is the application's window that appears after the login was
 * successful. It provides a menu at the top of the window which allows
```

```
 * navigating to the different use cases. In addition, it provides a huge area
 * of space below that menu, where the contents of the current window page are
 * displayed.
 *
 * @author Alexander Weickmann
 */
public final class MainWindow extends Window implements ActionListener {

	private static final long serialVersionUID = -1064259514784128805L;

	private static final int MINIMUM_WIDTH = 940;

	private static final int MINIMUM_HEIGHT = 400;

	/** This action command triggers the 'Current Bookings' use case. */
	private static final String AC_CURRENT_BOOKINGS = "current_bookings";

	/** This action command triggers the 'Search Cars' use case. */
	private static final String AC_SEARCH_CARS = "search_cars";

	/** This action command triggers the 'Edit Personal Data' use case. */
	private static final String AC_EDIT_PERSONAL_DATA = "edit_personal_data";

	/** This action command triggers the logout. */
	private static final String AC_LOGOUT = "logout";

	private final CurrentBookingsPage currentBookingsPage;

	private final CarsPage searchCarsPage;

	private final PersonalPage editPersonalDataPage;

	private final Customer loggedInUser;

	public MainWindow(Customer user) {
		super();

		loggedInUser = user;

		String title = getTitle();
		setTitle(title + " [logged in as: " + user.getLoginName() + "]");

		currentBookingsPage = new CurrentBookingsPage(this);
		searchCarsPage = new CarsPage(this);
		editPersonalDataPage = new PersonalPage(this);

		createMenu();

		showCurrentBookingsPage();
		setVisible(true);
	}

	/**
	 * Creates the menu that is shown at the top of the window.
	 */
	private void createMenu() {
		JPanel menuPanel = new JPanel();

		JButton currentBookingsButton = new JButton("Current Bookings");
		currentBookingsButton.setActionCommand(AC_CURRENT_BOOKINGS);
		currentBookingsButton.addActionListener(this);
		currentBookingsButton.setIcon(new ImageIcon(
				"images/current_bookings.png"));
		menuPanel.add(currentBookingsButton);
```

```
        JButton searchCarsButton = new JButton("Search Cars");
        searchCarsButton.setActionCommand(AC_SEARCH_CARS);
        searchCarsButton.addActionListener(this);
        searchCarsButton.setIcon(new ImageIcon("images/search_cars.png"));
        menuPanel.add(searchCarsButton);

        JButton editPersonalDataButton = new JButton("Edit Personal Data");
        editPersonalDataButton.setActionCommand(AC_EDIT_PERSONAL_DATA);
        editPersonalDataButton.addActionListener(this);
        editPersonalDataButton.setIcon(new ImageIcon(
                    "images/edit_personal_data.png"));
        menuPanel.add(editPersonalDataButton);

        JButton logoutButton = new JButton("Logout");
        logoutButton.setActionCommand(AC_LOGOUT);
        logoutButton.addActionListener(this);
        logoutButton.setIcon(new ImageIcon("images/logout.png"));
        menuPanel.add(logoutButton);

        add(menuPanel, GridBagUtil.createGridBagConstraints(0, 0, 1, 0,
                    GridBagConstraints.BOTH, new Insets(0, 0, 0, 0),
                    GridBagConstraints.FIRST_LINE_START, 0, 0));
    }

    public void showCurrentBookingsPage() {
        switchPageTo(currentBookingsPage);
    }

    public void showSearchCarsPage() {
        switchPageTo(searchCarsPage);
    }

    public void showEditPersonalDataPage() {
        switchPageTo(editPersonalDataPage);
    }

    @Override
    protected int getMinHeight() {
        return MINIMUM_HEIGHT;
    }

    @Override
    protected int getMinWidth() {
        return MINIMUM_WIDTH;
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        if (command.equals(AC_CURRENT_BOOKINGS)) {
            showCurrentBookingsPage();
        } else if (command.equals(AC_SEARCH_CARS)) {
            showSearchCarsPage();
        } else if (command.equals(AC_EDIT_PERSONAL_DATA)) {
            showEditPersonalDataPage();
        } else if (command.equals(AC_LOGOUT)) {
            logout();
        }
    }

    public Customer getLoggedInUser() {
        return loggedInUser;
    }
```

```
	/**
	 * Logs the current user out and shows the login screen yet again.
	 */
	private void logout() {
		SessionManager.getInstance().closeSession();
		setVisible(false);
		new SplashWindow();
	}
}
```

7.3.15 MainWindowPage.java

```
package de.hft.carrental.ui.main;

import de.hft.carrental.domain.Customer;
import de.hft.carrental.ui.WindowPage;

/**
 * A page belonging to the {@link MainWindow}. In addition to a normal window
 * page, this class provides a method to retrieve the currently logged in user.
 *
 * @author Alexander Weickmann
 */
public abstract class MainWindowPage extends WindowPage {

	private static final long serialVersionUID = 2652629706275110110L;

	protected MainWindowPage(MainWindow mainWindow, int layoutRows,
			int layoutColumns) {
		super(mainWindow, layoutRows, layoutColumns);
	}

	protected final Customer getLoggedInUser() {
		return ((MainWindow) getWindow()).getLoggedInUser();
	}

}
```

7.3.16 MainWindowPageSection.java

```
package de.hft.carrental.ui.main;

import de.hft.carrental.domain.Customer;
import de.hft.carrental.ui.WindowPageSection;

/**
 * A window page section that belongs to a {@link MainWindowPage}. In addition
 * to a normal page it offers a method that enables subclasses to retrieve the
 * currently logged in user.
 *
 * @author Alexander Weickmann
 */
public abstract class MainWindowPageSection extends WindowPageSection {

	private static final long serialVersionUID = -120418255996426304 8L;

	protected MainWindowPageSection(MainWindowPage mainWindowPage, String
title) {
		super(mainWindowPage, title);
	}
```

```
	protected final Customer getLoggedInUser() {
		return ((MainWindowPage) getWindowPage()).getLoggedInUser();
	}

}
```

7.3.17 TableSection.java

```
package de.hft.carrental.ui.main;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;

import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;

import de.hft.carrental.ui.util.GridBagUtil;

/**
 * This abstract class provides a section that contains a table. This table can
 * be configured as necessary by subclasses, for example which columns the table
 * has.
 *
 * @author Alexander Weickmann
 */
public abstract class TableSection extends MainWindowPageSection {

	private static final long serialVersionUID = 8789403383980546612L;

	/** The Swing table UI widget. */
	private JTable table;

	/** The table model that manages the data of the table. */
	private DefaultTableModel tableModel;

	/**
	 * @param page
	 *            The window page this section belongs to.
	 * @param title
	 *            The title for this section that will be shown in the title
	 *            area of the section.
	 * @param columnNames
	 *            The names of the columns this table shall have.
	 * @param columnWidths
	 *            The width of each column.
	 */
	protected TableSection(MainWindowPage page, String title,
			String[] columnNames, int[] columnWidths) {

		super(page, title);

		createLayout();
		beforeCreateTable();
		createTable(columnNames, columnWidths);
	}

	protected void beforeCreateTable() {
		// Empty default implementation.
	}
```

```
	private void createTable(String[] columnNames, int[] columnWidths) {
		tableModel = new DefaultTableModel();
		for (String columnName : columnNames) {
			tableModel.addColumn(columnName);
		}
		table = new JTable(tableModel);
		int columnMargin = 5;
		table.getColumnModel().setColumnMargin(columnMargin);
		for (int i = 0; i < columnWidths.length; i++) {

	table.getColumn(columnNames[i]).setPreferredWidth(columnWidths[i]);
		}
		JTableHeader tableHeader = table.getTableHeader();
		tableHeader.getColumnModel().setColumnMargin(columnMargin);
		add(tableHeader, GridBagUtil.createGridBagConstraints(0, 1, 1, 0,
				GridBagConstraints.HORIZONTAL, new Insets(0, 0, 0, 0),
				GridBagConstraints.FIRST_LINE_START, 0, 0));
		add(table, GridBagUtil.createGridBagConstraints(0, 2, 1, 1,
				GridBagConstraints.HORIZONTAL, new Insets(0, 0, 0, 0),
				GridBagConstraints.FIRST_LINE_START, 0, 0));
	}

	protected void createLayout() {
		GridBagLayout layout = new GridBagLayout();
		setLayout(layout);
	}

	/**
	 * Clears the table by removing all data from the table so that it is
empty
	 * after a call to this operation.
	 */
	protected final void clearTable() {
		tableModel.getDataVector().clear();
	}

	/**
	 * Adds one row of data to the table.
	 *
	 * @param rowData
	 *            The array containing the row data.
	 */
	protected final void addDataRow(Object[] rowData) {
		tableModel.addRow(rowData);
	}
}
```

7.3.18 CurrentBookingsPage.java

```
package de.hft.carrental.ui.main.bookings;

import de.hft.carrental.ui.main.MainWindow;
import de.hft.carrental.ui.main.MainWindowPage;

/**
 * This page belongs to the {@link MainWindow} and enables the user to review
 * his current bookings.
 *
 * @author Alexander Weickmann
 */
public final class CurrentBookingsPage extends MainWindowPage {
```

```
	private static final long serialVersionUID = 5392467214213264243L;

	public CurrentBookingsPage(MainWindow mainWindow) {
		super(mainWindow, 1, 1);
	}

	@Override
	protected void addSections() {
		addSection(new CurrentBookingsTableSection(this));
	}

}
```

7.3.19 CurrentBookingsTableSection.java

```
package de.hft.carrental.ui.main.bookings;

import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.Date;
import java.util.Iterator;
import java.util.Set;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import org.hibernate.Session;
import org.hibernate.Transaction;

import de.hft.carrental.database.SessionManager;
import de.hft.carrental.domain.Booking;
import de.hft.carrental.domain.BranchAddress;
import de.hft.carrental.ui.main.TableSection;
import de.hft.carrental.ui.util.GridBagUtil;

/**
 * This section belongs to the {@link CurrentBookingsPage}. It shows a table
 * with all the information relevant to the user's current bookings.
 *
 * @author Alexander Weickmann
 */
public final class CurrentBookingsTableSection extends TableSection implements
		ActionListener {

	private static final long serialVersionUID = 6099129396844699288L;

	private static final String AC_CANCEL = "Cancel";

	private static final int[] COLUMN_WIDTHS = new int[] { 55, 65, 65, 80, 90,
			90, 160 };
```

```
	private static final String[] COLUMN_NAMES = new String[] { "Booking Nr.",
			"Booking Date", "Return Date", "Car Type", "Registration
Number",
			"Agency", "Location" };

	private JTextField cancelField;

	/**
	 * @param currentBookingsPage
	 *            The {@link CurrentBookingsPage} this section belongs to.
	 */
	protected CurrentBookingsTableSection(
			CurrentBookingsPage currentBookingsPage) {

		super(currentBookingsPage, "Current Bookings", COLUMN_NAMES,
				COLUMN_WIDTHS);
	}

	@Override
	protected void refresh() {
		clearTable();
		fillTableWithData();
	}

	@Override
	protected void beforeCreateTable() {
		JPanel cancelPanel = new JPanel();
		cancelPanel.setLayout(new FlowLayout());

		JLabel cancelLabel = new JLabel("Cancel Booking: ");
		cancelLabel.setFont(new Font("Arial", Font.BOLD, 11));
		cancelPanel.add(cancelLabel);

		cancelField = new JTextField(10);
		cancelField.setText("Booking Nr.");
		cancelField.addFocusListener(new FocusListener() {
			@Override
			public void focusGained(FocusEvent e) {
				cancelField.setText("");
			}

			@Override
			public void focusLost(FocusEvent e) {
				if (cancelField.getText().length() == 0) {
					cancelField.setText("Booking Nr.");
				}
			}
		});
		cancelField.addKeyListener(new KeyListener() {
			@Override
			public void keyPressed(KeyEvent e) {
				// Nothing to do.
			}

			@Override
			public void keyReleased(KeyEvent e) {
				if (e.getKeyCode() == KeyEvent.VK_ENTER) {
					cancelBooking();
				}
			}

			@Override
			public void keyTyped(KeyEvent e) {
				// Nothing to do.
```

```
            }
        });
        cancelPanel.add(cancelField);

        JButton cancelButton = new JButton("Cancel!");
        cancelButton.addActionListener(this);
        cancelButton.setActionCommand(AC_CANCEL);
        cancelPanel.add(cancelButton);

        add(cancelPanel, GridBagUtil.createGridBagConstraints(0, 0, 1, 0,
                    GridBagConstraints.BOTH, new Insets(0, 0, 0, 0),
                    GridBagConstraints.FIRST_LINE_START, 0, 0));
    }

    private void fillTableWithData() {
        Iterator<Booking> it = getLoggedInUser().getBookings().iterator();
        for (int i = 0; it.hasNext(); i++) {
            Object[] rowData = new Object[7];
            Booking booking = it.next();
            rowData[0] = booking.getBookingNumber();

            Date bookingDate = booking.getBookingDate();
            String bookingDateString = bookingDate.toString();
            rowData[1] = bookingDateString.substring(0, bookingDateString
                        .indexOf(" "));

            Date returnDate = booking.getReturnDate();
            String returnDateString = returnDate.toString();
            rowData[2] = returnDateString.substring(0, returnDateString
                        .indexOf(" "));
            rowData[3] = booking.getCar().getCarType().getName();
            rowData[4] = booking.getCar().getRegistrationNumber();
            rowData[5] = booking.getAgency().getName();

            BranchAddress branchAddress = booking.getCar().getBranch()
                        .getBranchAddress();
            rowData[6] = branchAddress.getPostalCode() + " "
                        + branchAddress.getCityName() + ", "
                        + branchAddress.getStreetName() + " "
                        + branchAddress.getStreetNumber();

            addDataRow(rowData);
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        String command = e.getActionCommand();
        if (command.equals(AC_CANCEL)) {
            cancelBooking();
        }
    }

    private void cancelBooking() {
        String nrToCancel = cancelField.getText();
        for (Character c : nrToCancel.toCharArray()) {
            if (!(Character.isDigit(c))) {
                return;
            }
        }
        Integer bookingNumber = Integer.valueOf(nrToCancel);
        Booking toDelete = null;
        Set<Booking> bookings = getLoggedInUser().getBookings();
        for (Booking booking : bookings) {
```

```
                if (booking.getBookingNumber().equals(bookingNumber)) {
                        toDelete = booking;
                        break;
                }
            }
            if (toDelete == null) {
                return;
            }
            bookings.remove(toDelete);
            Session session = SessionManager.getInstance().openSession();
            Transaction transaction = session.beginTransaction();
            session.delete(toDelete);
            transaction.commit();
            refresh();
        }
}
```

7.3.20 CarsPage.java

```
package de.hft.carrental.ui.main.cars;

import de.hft.carrental.ui.main.MainWindow;
import de.hft.carrental.ui.main.MainWindowPage;

// TODO RM, PS: Class yet to be implemented.
public final class CarsPage extends MainWindowPage {

    private static final long serialVersionUID = 8416397855771759503L;

    public CarsPage(MainWindow mainWindow) {
        super(mainWindow, 2, 1);
    }

    @Override
    protected void addSections() {
        addSection(new SearchCarsSection(this));
        addSection(new CarsTableSection(this));
    }

}
```

7.3.21 CarsTableSection.java

```
package de.hft.carrental.ui.main.cars;

import de.hft.carrental.ui.main.MainWindowPage;
import de.hft.carrental.ui.main.TableSection;

public final class CarsTableSection extends TableSection {

    private static final long serialVersionUID = 6923046504449113618L;

    protected CarsTableSection(MainWindowPage page) {
        super(page, "INSERT TITLE", new String[] {}, new int[] {});
    }

    @Override
    protected void refresh() {
        // TODO RM, PS: Method yet to be implemented.
    }
```

```
}
```

7.3.22 SearchCarsSection.java

```
package de.hft.carrental.ui.main.cars;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;

import de.hft.carrental.ui.main.MainWindowPageSection;

//TODO RM, PS: Class yet to be implemented.
public final class SearchCarsSection extends MainWindowPageSection {

	private static final long serialVersionUID = -8607244737443476481L;

	// Labels for search page
	private final JLabel CarName = new JLabel("Car Name:");
	private final JLabel Pickup = new JLabel("Pick up Location:");
	private final JLabel FromDate = new JLabel("From:");
	private final JLabel ToDate = new JLabel("To:");

	// Textfields for search page
	private final JTextField CarNameField = new JTextField();
	private final JTextField PickupField = new JTextField();
	private final JTextField FromDateField = new JTextField();
	private final JTextField ToDateField = new JTextField();

	// Button for search
	private final JButton searchButton = new JButton("Search");

	protected SearchCarsSection(CarsPage carsPage) {
		super(carsPage, "Search Cars");
	}

	@Override
	protected void refresh() {
		// CarNameField.setText(user.getCarName());
		// PickupField.setText(user.getPickup());
		// FromDateField.setText(user.getFromDate());
		// ToDateField.setText(user.getToDate());
		//
		// DateFormat df = DateFormat.getInstance();
		// FromDateField.setText(df.format(user.getFromDate()));
		// ToDateField.setText(df.format(user.getToDate()));
		// searchButton.setEnabled(false);
	}

	private void addListeners() {
		// searchButton.addActionListener(this);
	}

	// @Override
	// public void actionPerformed(ActionEvent e) {
	// searchButton.setEnabled(True);
	// CarsTableSection();
	// }
}
```

7.3.23 AddAddressDialog.java

```
package de.hft.carrental.ui.main.personal;

import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JButton;

import de.hft.carrental.domain.Customer;
import de.hft.carrental.domain.CustomerAddress;

public class AddAddressDialog extends BaseAddressDialog implements KeyListener,
            ActionListener {

      private static final long serialVersionUID = 1L;

      private JButton addButton = new JButton("Add");

      private JButton closeButton = new JButton("Close");

      private boolean addressAdded = false;

      private Customer customer;

      private static final String AC_ADD_ADDRESS = "add_address";

      private static final String AC_CLOSE_DIALOG = "close_dialog";

      public AddAddressDialog(Customer customer) {
            this.customer = customer;

            setTitle("Add new address:");

            addButtons();
            addListeners();

            pack();
            int posX = Toolkit.getDefaultToolkit().getScreenSize().width / 2
                        - getWidth() / 2;
            int posY = Toolkit.getDefaultToolkit().getScreenSize().height / 2
                        - getHeight() / 2;
            setLocation(posX, posY);

            addButton.setEnabled(false);
            setVisible(true);
      }

      public CustomerAddress getNewCustomerAddress() {
            CustomerAddress newAddress = new CustomerAddress();
            newAddress.setStreetName(streetField.getText());
            newAddress.setStreetNumber(numberField.getText());
            newAddress.setPostalCode(postalField.getText());
            newAddress.setCityName(cityField.getText());
            newAddress.setCountry(countryField.getText());
            newAddress.setPhoneNumber(phoneField.getText());
            newAddress.setCustomer(customer);

            return newAddress;
      }
```

```
	public boolean addressAdded() {
		return addressAdded;
	}

	private void addButtons() {
		add(addButton, "align left");
		add(closeButton, "align right");
	}

	private void addListeners() {
		addButton.setActionCommand(AC_ADD_ADDRESS);
		addButton.addActionListener(this);
		closeButton.setActionCommand(AC_CLOSE_DIALOG);
		closeButton.addActionListener(this);

		streetField.addKeyListener(this);
		numberField.addKeyListener(this);
		postalField.addKeyListener(this);
		cityField.addKeyListener(this);
		countryField.addKeyListener(this);
		phoneField.addKeyListener(this);
	}

	@Override
	public void keyPressed(KeyEvent e) {
		/* nothing to do */
	}

	@Override
	public void keyReleased(KeyEvent e) {
		/* nothing to do */
	}

	@Override
	public void keyTyped(KeyEvent e) {
		if (allFieldsFilled()) {
			addButton.setEnabled(true);
		} else {
			addButton.setEnabled(false);
		}
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		String actionCommand = e.getActionCommand();

		if (actionCommand.equals(AC_ADD_ADDRESS)) {
			addressAdded = true;
			setVisible(false);
		} else if (actionCommand.equals(AC_CLOSE_DIALOG)) {
			setVisible(false);
		}
	}
}
```

7.3.24 AddressesSection.java

```
package de.hft.carrental.ui.main.personal;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.HashSet;
```

```java
import java.util.Set;

import javax.swing.JButton;

import org.hibernate.Session;
import org.hibernate.Transaction;

import de.hft.carrental.database.SessionManager;
import de.hft.carrental.domain.CustomerAddress;
import de.hft.carrental.ui.main.MainWindowPage;
import de.hft.carrental.ui.main.TableSection;

public final class AddressesSection extends TableSection implements
		ActionListener {

	private static final String AC_ADD_ADDRESS = "add_address";

	private static final String AC_EDIT_ADDRESS = "edit_address";

	private static final long serialVersionUID = -1627894704897348854L;

	private static final String[] COLUMN_NAMES = { "Street", "Number",
			"Postal code", "City", "Country", "Phone number"

	};

	private static final int[] COLUMN_WIDTHS = new int[] { 110, 100, 100, 130,
			140, 150 };

	private JButton add = new JButton("Add");

	private JButton edit = new JButton("Edit");

	protected AddressesSection(MainWindowPage page) {
		super(page, "Address(es)", COLUMN_NAMES, COLUMN_WIDTHS);

		add(add);
		add(edit);
		addListeners();
	}

	@Override
	protected void refresh() {
		clearTable();
		fillTable();
	}

	private void addListeners() {
		add.setActionCommand(AC_ADD_ADDRESS);
		add.addActionListener(this);
		edit.setActionCommand(AC_EDIT_ADDRESS);
		edit.addActionListener(this);
	}

	private void fillTable() {
		for (CustomerAddress address :
getLoggedInUser().getCustomerAddresses()) {
			Object[] row = new Object[6];
			row[0] = address.getStreetName();
			row[1] = address.getStreetNumber();
			row[2] = address.getPostalCode();
			row[3] = address.getCityName();
			row[4] = address.getCountry();
			row[5] = address.getPhoneNumber();
```

```
                addDataRow(row);
            }
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            String actionCommand = e.getActionCommand();

            if (actionCommand.equals(AC_ADD_ADDRESS)) {
                AddAddressDialog ad = new AddAddressDialog(getLoggedInUser());

                if (ad.addressAdded()) {
                    Session session =
SessionManager.getInstance().openSession();
                    Transaction transaction = session.beginTransaction();
                    getLoggedInUser().getCustomerAddresses().add(
                            ad.getNewCustomerAddress());
                    session.save(getLoggedInUser());
                    transaction.commit();
                    refresh();
                }
            } else if (actionCommand.equals(AC_EDIT_ADDRESS)) {
                EditAddressDialog ed = new EditAddressDialog(getLoggedInUser()
                        .getCustomerAddresses());

                if (ed.getAddresses().size() == getLoggedInUser()
                        .getCustomerAddresses().size()) {
                    Session session =
SessionManager.getInstance().openSession();
                    Transaction transaction = session.beginTransaction();

        getLoggedInUser().setCustomerAddresses(ed.getAddresses());
                    transaction.commit();
                } else {
                    Session session =
SessionManager.getInstance().openSession();
                    Transaction transaction = session.beginTransaction();
                    Set<CustomerAddress> customerAddresses = new
HashSet<CustomerAddress>(
                            getLoggedInUser().getCustomerAddresses());
                    customerAddresses.removeAll(ed.getAddresses());
                    CustomerAddress[] array = customerAddresses
                            .toArray(new CustomerAddress[1]);
                    CustomerAddress address = array[0];

        getLoggedInUser().getCustomerAddresses().remove(address);
                    transaction.commit();
                }
                refresh();
            }
        }
}
```

7.3.25 BaseAddressDialog.java

```
package de.hft.carrental.ui.main.personal;

import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JTextField;
```

```
import net.miginfocom.swing.MigLayout;

public abstract class BaseAddressDialog extends JDialog {
	private static final long serialVersionUID = 1L;

	private JLabel streetLabel = new JLabel("Street:");
	private JLabel numberLabel = new JLabel("Street Number:");
	private JLabel postalLabel = new JLabel("Postal code:");
	private JLabel cityLabel = new JLabel("City:");
	private JLabel countryLabel = new JLabel("Country:");
	private JLabel phoneLabel = new JLabel("Phone number");

	protected JTextField streetField = new JTextField(45);
	protected JTextField numberField = new JTextField(5);
	protected JTextField postalField = new JTextField(10);
	protected JTextField cityField = new JTextField(45);
	protected JTextField countryField = new JTextField(45);
	protected JTextField phoneField = new JTextField(45);

	protected BaseAddressDialog() {
		setModalityType(ModalityType.APPLICATION_MODAL);
		buildLayout();
	}

	private void buildLayout() {
		setLayout(new MigLayout("", "[][grow]"));

		add(streetLabel);
		add(streetField, "growx, wrap");

		add(numberLabel);
		add(numberField, "growx, wrap");

		add(postalLabel);
		add(postalField, "growx, wrap");

		add(cityLabel);
		add(cityField, "growx, wrap");

		add(countryLabel);
		add(countryField, "growx, wrap");

		add(phoneLabel);
		add(phoneField, "growx, wrap");
	}

	protected boolean allFieldsFilled() {
		if ((streetField.getText().length() == 0)
				|| (numberField.getText().length() == 0)
				|| (postalField.getText().length() == 0)
				|| (cityField.getText().length() == 0)
				|| (countryField.getText().length() == 0)
				|| (phoneField.getText().length() == 0)) {
			return false;
		}

		return true;
	}
}
```

7.3.26 EditAddressDialog.java

```
package de.hft.carrental.ui.main.personal;
```

```
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import javax.swing.JButton;

import de.hft.carrental.domain.CustomerAddress;

public class EditAddressDialog extends BaseAddressDialog implements
            ActionListener, KeyListener {

      private static final long serialVersionUID = 1L;

      private JButton close = new JButton("Close");

      private JButton previous = new JButton("Previous");

      private JButton next = new JButton("Next");

      private JButton save = new JButton("Save");

      private JButton delete = new JButton("Delete");

      private static final String AC_CLOSE_DIALOG = "close_dialog";

      private static final String AC_PREVIOUS_ADDRESS = "previous_address";

      private static final String AC_NEXT_ADDRESS = "next_address";

      private static final String AC_SAVE_CHANGES = "save_changes";

      private static final String AC_DELETE_ADDRESS = "delete_address";

      private List<CustomerAddress> addressList = new
ArrayList<CustomerAddress>();

      private int pos = 0;

      public EditAddressDialog(Set<CustomerAddress> addresses) {
            addressList.addAll(addresses);

            setTitle("Edit address details:");
            addButtons();
            addListeners();
            fillFields(pos);

            if (addressList.size() == 1) {
                  delete.setEnabled(false);
            }

            pack();
            int posX = Toolkit.getDefaultToolkit().getScreenSize().width / 2
                        - getWidth() / 2;
            int posY = Toolkit.getDefaultToolkit().getScreenSize().height / 2
                        - getHeight() / 2;
            setLocation(posX, posY);

            save.setEnabled(false);
```

```
		setVisible(true);
	}

	public Set<CustomerAddress> getAddresses() {
		Set<CustomerAddress> tmp = new HashSet<CustomerAddress>();
		tmp.addAll(addressList);
		return tmp;
	}

	private void addButtons() {
		add(previous, "split 2, align left");
		add(next);
		add(save, "split 3, align right");
		add(delete);
		add(close);
	}

	private void fillFields(int pos) {
		streetField.setText(addressList.get(pos).getStreetName());
		numberField.setText(addressList.get(pos).getStreetNumber());
		postalField.setText(addressList.get(pos).getPostalCode());
		cityField.setText(addressList.get(pos).getCityName());
		countryField.setText(addressList.get(pos).getCountry());
		phoneField.setText(addressList.get(pos).getPhoneNumber());

		checkPreviousNext();
	}

	private void checkPreviousNext() {
		if (pos - 1 < 0) {
			previous.setEnabled(false);
		} else {
			previous.setEnabled(true);
		}

		if (pos + 1 == addressList.size()) {
			next.setEnabled(false);
		} else {
			next.setEnabled(true);
		}
	}

	private void addListeners() {
		close.setActionCommand(AC_CLOSE_DIALOG);
		close.addActionListener(this);
		previous.setActionCommand(AC_PREVIOUS_ADDRESS);
		previous.addActionListener(this);
		next.setActionCommand(AC_NEXT_ADDRESS);
		next.addActionListener(this);
		save.setActionCommand(AC_SAVE_CHANGES);
		save.addActionListener(this);
		delete.setActionCommand(AC_DELETE_ADDRESS);
		delete.addActionListener(this);

		streetField.addKeyListener(this);
		numberField.addKeyListener(this);
		postalField.addKeyListener(this);
		cityField.addKeyListener(this);
		countryField.addKeyListener(this);
		phoneField.addKeyListener(this);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
```

```
            String actionCommand = e.getActionCommand();

            if (actionCommand.equals(AC_CLOSE_DIALOG)) {
                  setVisible(false);
            } else if (actionCommand.equals(AC_DELETE_ADDRESS)) {
                  addressList.remove(pos);
                  setVisible(false);
            } else if (actionCommand.equals(AC_NEXT_ADDRESS)) {
                  fillFields(++pos);
            } else if (actionCommand.equals(AC_PREVIOUS_ADDRESS)) {
                  fillFields(--pos);
            } else if (actionCommand.equals(AC_SAVE_CHANGES)) {
                  addressList.get(pos).setStreetName(streetField.getText());
                  addressList.get(pos).setStreetNumber(numberField.getText());
                  addressList.get(pos).setPostalCode(postalField.getText());
                  addressList.get(pos).setCityName(cityField.getText());
                  addressList.get(pos).setCountry(countryField.getText());
                  addressList.get(pos).setPhoneNumber(phoneField.getText());

                  save.setEnabled(false);
            }

      }

      @Override
      public void keyPressed(KeyEvent e) {
            /* nothing to do */
      }

      @Override
      public void keyReleased(KeyEvent e) {
            /* nothing to do */
      }

      @Override
      public void keyTyped(KeyEvent e) {
            if (allFieldsFilled()) {
                  save.setEnabled(true);
            } else {
                  save.setEnabled(false);
            }
      }
}
```

7.3.27 GeneralInfoSection.java

```
package de.hft.carrental.ui.main.personal;

import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.text.DateFormat;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;

import net.miginfocom.swing.MigLayout;
```

```java
import org.hibernate.Session;
import org.hibernate.Transaction;

import de.hft.carrental.database.SessionManager;
import de.hft.carrental.domain.Customer;
import de.hft.carrental.ui.main.MainWindowPage;
import de.hft.carrental.ui.main.MainWindowPageSection;

public final class GeneralInfoSection extends MainWindowPageSection implements
		ActionListener, KeyListener {

	private static final long serialVersionUID = 2921841683848149881L;

	private final JLabel loginLabel = new JLabel("Login name:");
	private final JLabel registerDateLabel = new JLabel("Register date:");
	private final JLabel emailLabel = new JLabel("Email:");
	private final JLabel firstNameLabel = new JLabel("First name:");
	private final JLabel surNameLabel = new JLabel("Surname:");
	private final JLabel birthDateLabel = new JLabel("Date of birth:");
	private final JLabel companyNameLabel = new JLabel("Company name: ");

	private final JTextField loginField = new JTextField();
	private final JTextField registerField = new JTextField();
	private final JTextField emailField = new JTextField();
	private final JTextField firstNameField = new JTextField();
	private final JTextField surNameField = new JTextField();
	private final JTextField birthDateField = new JTextField();
	private final JTextField companyNameField = new JTextField();

	private final JButton saveChangesButton = new JButton("Save changes");

	private static final String AC_SAVE_CHANGES = "save_changes";

	private final Customer user;

	protected GeneralInfoSection(MainWindowPage page) {
		super(page, "Personal details");
		setLayout(new MigLayout("", "[][grow][][grow]", ""));
		user = getLoggedInUser();

		if (user.getCustomerType().equals(Customer.CUSTOMER_TYPE_PRIVATE)) {
			createPrivateUserContents();
		} else {
			createCompanyUserContents();
		}

		addListeners();

		Image image = new ImageIcon("images/save.png").getImage();
		image = image.getScaledInstance(20, 20, Image.SCALE_SMOOTH);
		saveChangesButton.setIcon(new ImageIcon(image));
	}

	@Override
	protected void refresh() {
		loginField.setText(user.getLoginName());
		firstNameField.setText(user.getFirstName());
		surNameField.setText(user.getSurname());
		companyNameField.setText(user.getCompanyName());
		emailField.setText(user.getEmail());

		DateFormat df = DateFormat.getInstance();
		registerField.setText(df.format(user.getRegisterDate()));
		birthDateField.setText(df.format(user.getDateOfBirth()));
```

```
		saveChangesButton.setEnabled(false);
	}

	private void createPrivateUserContents() {
		add(loginLabel);
		add(loginField, "growx");
		loginField.setEditable(false);
		add(firstNameLabel);
		add(firstNameField, "growx, wrap");
		firstNameField.setEditable(false);

		add(registerDateLabel);
		add(registerField, "growx");
		registerField.setEditable(false);
		add(surNameLabel);
		add(surNameField, "growx, wrap");
		surNameField.setEditable(false);

		add(emailLabel);
		add(emailField, "growx");
		add(birthDateLabel);
		add(birthDateField, "growx, wrap");
		birthDateField.setEditable(false);

		add(saveChangesButton, "span 4, align right");
	}

	private void createCompanyUserContents() {
		add(loginLabel, "span 1 3, growx");
		add(loginField);
		loginField.setEditable(false);

		add(registerDateLabel);
		add(registerField, "growx");
		registerField.setEditable(false);

		add(emailLabel);
		add(emailField, "growx, wrap");

		add(companyNameLabel);
		add(companyNameField, "growx");
		companyNameField.setEditable(false);
	}

	private void addListeners() {
		saveChangesButton.setActionCommand(AC_SAVE_CHANGES);
		saveChangesButton.addActionListener(this);
		emailField.addKeyListener(this);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		String actionCommand = e.getActionCommand();

		if (actionCommand.equals(AC_SAVE_CHANGES)) {
			Session session = SessionManager.getInstance().openSession();
			Transaction transaction = session.beginTransaction();
			user.setEmail(emailField.getText());
			session.save(user);
			transaction.commit();
			saveChangesButton.setEnabled(false);
		}
	}
```

```
	@Override
	public void keyPressed(KeyEvent e) {
		/* nothing to do */
	}

	@Override
	public void keyReleased(KeyEvent e) {
		/* nothing to do */
	}

	@Override
	public void keyTyped(KeyEvent e) {
		saveChangesButton.setEnabled(true);
	}

}
```

7.3.28 PersonalPage.java

```
package de.hft.carrental.ui.main.personal;

import de.hft.carrental.ui.main.MainWindow;
import de.hft.carrental.ui.main.MainWindowPage;

public final class PersonalPage extends MainWindowPage {

	private static final long serialVersionUID = -5215876430603142614L;

	public PersonalPage(MainWindow mainWindow) {
		super(mainWindow, 2, 1);
	}

	@Override
	protected void addSections() {
		addSection(new GeneralInfoSection(this));
		addSection(new AddressesSection(this));
	}

}
```

7.3.29 SplashWindow.java

```
package de.hft.carrental.ui.splash;

import de.hft.carrental.ui.Window;
import de.hft.carrental.ui.splash.login.LoginPage;

/**
 * The splash window is the application window that is shown right after the
 * program starts. Basically it allows the user to login.
 *
 * @author Alexander Weickmann
 */
public final class SplashWindow extends Window {

	private static final long serialVersionUID = -5210789835246067684L;

	private static final int MINIMUM_WIDTH = 300;

	private static final int MINIMUM_HEIGHT = 120;
```

```
	private LoginPage loginPage;

	public SplashWindow() {
		super();

		loginPage = new LoginPage(this);

		showLoginPage();
		setVisible(true);
	}

	public void showLoginPage() {
		switchPageTo(loginPage);
	}

	@Override
	protected int getMinHeight() {
		return MINIMUM_HEIGHT;
	}

	@Override
	protected int getMinWidth() {
		return MINIMUM_WIDTH;
	}

}
```

7.3.30 LoginPage.java

```
package de.hft.carrental.ui.splash.login;

import de.hft.carrental.ui.WindowPage;
import de.hft.carrental.ui.splash.SplashWindow;

/**
 * This page is used by the {@link SplashWindow} to enable the user to login.
 *
 * @author Alexander Weickmann
 */
public final class LoginPage extends WindowPage {

	private static final long serialVersionUID = -2590619285921957633L;

	/**
	 * @param splashWindow
	 *            The {@link SplashWindow} this page belongs to.
	 */
	public LoginPage(SplashWindow splashWindow) {
		super(splashWindow, 1, 1);
	}

	@Override
	protected void addSections() {
		addSection(new LoginSection(this));
	}

}
```

7.3.31 LoginSection.java

```
package de.hft.carrental.ui.splash.login;

import java.awt.Toolkit;
import java.awt.Dialog.ModalityType;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JTextField;

import net.miginfocom.swing.MigLayout;

import org.hibernate.Session;
import org.hibernate.Transaction;

import de.hft.carrental.database.SessionManager;
import de.hft.carrental.domain.Customer;
import de.hft.carrental.ui.WindowPageSection;
import de.hft.carrental.ui.main.MainWindow;

/**
 * This section contains a text field allowing the user to type in his login
 * name. It also shows a button that allows the user to eventually perform the
 * login.
 *
 * @author Alexander Weickmann
 */
public final class LoginSection extends WindowPageSection implements
            ActionListener {

      private static final long serialVersionUID = 10645211197356544437L;

      private static final String TITLE = "Login";

      /** Action command that is triggered upon activation of the login button.
*/
      private static final String AC_LOGIN = "login";

      private JTextField loginTextField;

      /**
       * @param loginPage
       *            The {@link LoginPage} this section belongs to.
       */
      public LoginSection(LoginPage loginPage) {
            super(loginPage, TITLE);
            createContents();
      }

      private void createContents() {
            loginTextField = new JTextField(18);
            loginTextField.requestFocusInWindow();
            loginTextField.addKeyListener(new KeyListener() {
                  @Override
                  public void keyPressed(KeyEvent e) {
                        // Nothing to do.
                  }
```

```
			@Override
			public void keyReleased(KeyEvent e) {
				if (e.getKeyCode() == KeyEvent.VK_ENTER) {
					performLogin();
				}
			}

			@Override
			public void keyTyped(KeyEvent e) {
				// Nothing to do.
			}
		});
		add(loginTextField);

		JButton loginButton = new JButton("Login!");
		loginButton.setActionCommand(AC_LOGIN);
		loginButton.addActionListener(this);
		loginButton.setIcon(new ImageIcon("images/login.png"));
		add(loginButton);
	}

	@Override
	public void actionPerformed(ActionEvent e) {
		String command = e.getActionCommand();
		if (command.equals(AC_LOGIN)) {
			performLogin();
		}
	}

	/**
	 * Performs the login. Shows the main window on success and displays an
	 * appropriate error message on failure.
	 */
	private void performLogin() {
		Customer user = null;
		String username = loginTextField.getText();

		Session session = SessionManager.getInstance().openSession();
		if (session == null) {
			showErrorDialog("No connection to database.");
			return;
		}

		Transaction tr = session.beginTransaction();

		String query = "from Customer where loginName = '" + username + "'";
		Object result = session.createQuery(query).uniqueResult();

		if (result == null) {
			showErrorDialog("Username not found.");
			return;
		}
		user = (Customer) result;

		tr.commit();

		getWindowPage().getWindow().setVisible(false);
		new MainWindow(user);
	}

	@Override
	protected void refresh() {
		// Nothing to do.
```

```
	}

	private void showErrorDialog(String errorText) {
		final JDialog errorDialog = new JDialog();
		errorDialog.setTitle("Error");

		MigLayout layout = new MigLayout("", "[center]", "[center]");

		errorDialog.setLayout(layout);
		errorDialog.setModalityType(ModalityType.APPLICATION_MODAL);

		JLabel errorLabel = new JLabel(errorText);
		errorDialog.add(errorLabel, "spanx, wrap");

		final JButton closeButton = new JButton("Ok");
		closeButton.addActionListener(new ActionListener() {

			@Override
			public void actionPerformed(ActionEvent e) {
				errorDialog.setVisible(false);
			}
		});
		errorDialog.add(closeButton);

		errorDialog.pack();
		errorDialog.setResizable(false);
		int posX = Toolkit.getDefaultToolkit().getScreenSize().width / 2;
		int posY = Toolkit.getDefaultToolkit().getScreenSize().height / 2;
		errorDialog.setLocation(posX, posY);

		errorDialog.setVisible(true);

		loginTextField.setText("");
		loginTextField.requestFocus();
	}

}
```

7.3.32 GridBagUtil.java

```
package de.hft.carrental.ui.util;

import java.awt.GridBagConstraints;
import java.awt.Insets;

/**
 *
 *
 * @author Alexander Weickmann
 */
public final class GridBagUtil {

	/**
	 * Convenience method provided to subclasses allowing for rapid
	 * {@link GridBagConstraints} creation.
	 *
	 * @param gridx
	 *            The zero-based x-position of the component inside the grid.
	 * @param gridy
	 *            The zero-based y-position of the component inside the grid.
	 * @param weightx
	 *            Value between 0.0 and 1.0 indicating how much priority the
```

```
         *            component has when it comes to filling up empty horizontal
         *            space.
         * @param weighty
         *            Value between 0.0 and 1.0 indicating how much priority the
         *            component has when it comes to filling up empty vertical
         *            space.
         * @param fill
         *            Indicates whether additional space should be used by the
         *            component (both, horizontal, vertical or none).
         * @param insets
         *            Specifies the external padding of the component.
         * @param anchor
         *            Specifies where to anchor the component.
         * @param ipadx
         *            Specifies the internal padding in x direction.
         * @param ipady
         *            Specifies the internal padding in y direction.
         */
        public static GridBagConstraints createGridBagConstraints(int gridx,
                    int gridy, int weightx, int weighty, int fill, Insets insets,
                    int anchor, int ipadx, int ipady) {

              GridBagConstraints constraints = new GridBagConstraints();
              constraints.gridx = gridx;
              constraints.gridy = gridy;
              constraints.weightx = weightx;
              constraints.weighty = weighty;
              constraints.fill = fill;
              constraints.insets = insets;
              constraints.anchor = anchor;
              constraints.ipadx = ipadx;
              constraints.ipady = ipady;
              return constraints;
        }

}
```

7.3.33 hibernate.cfg.xml

```
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
      <session-factory>
            <mapping class="de.hft.carrental.domain.Agency" />
            <mapping class="de.hft.carrental.domain.Booking" />
            <mapping class="de.hft.carrental.domain.Branch" />
            <mapping class="de.hft.carrental.domain.BranchAddress" />
            <mapping class="de.hft.carrental.domain.Car" />
            <mapping class="de.hft.carrental.domain.CarType" />
            <mapping class="de.hft.carrental.domain.Customer" />
            <mapping class="de.hft.carrental.domain.CustomerAddress" />
      </session-factory>
</hibernate-configuration>
```

7.3.34 hibernate.properties

```
hibernate.dialectorg.hibernate.dialect.MySQLInnoDBDialect
hibernate.connection.driver_class com.mysql.jdbc.Driver
hibernate.connection.url jdbc:mysql:///carrental
```

```
hibernate.connection.username root
hibernate.connection.password root
hibernate.show_sql true
hibernate.cache.provider_class org.hibernate.cache.HashtableCacheProvider
```

7.4 Google Code Source Repository URL

https://code.google.com/p/carrentalhft/