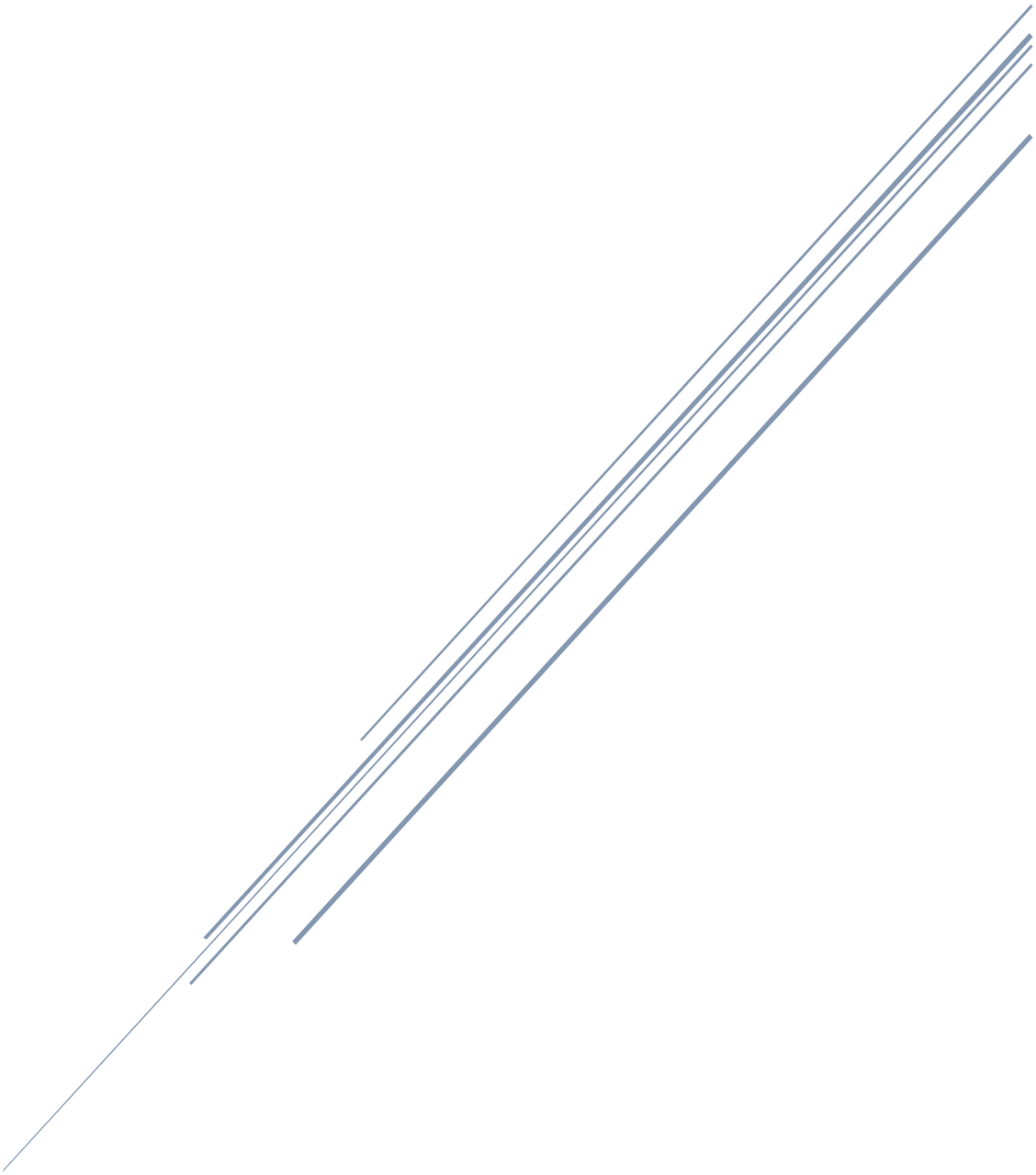
HAW HAMBURG

Software Engineering I Praktikum 4



HAW Hamburg
Steffen Giersch & Maria Lüdemann

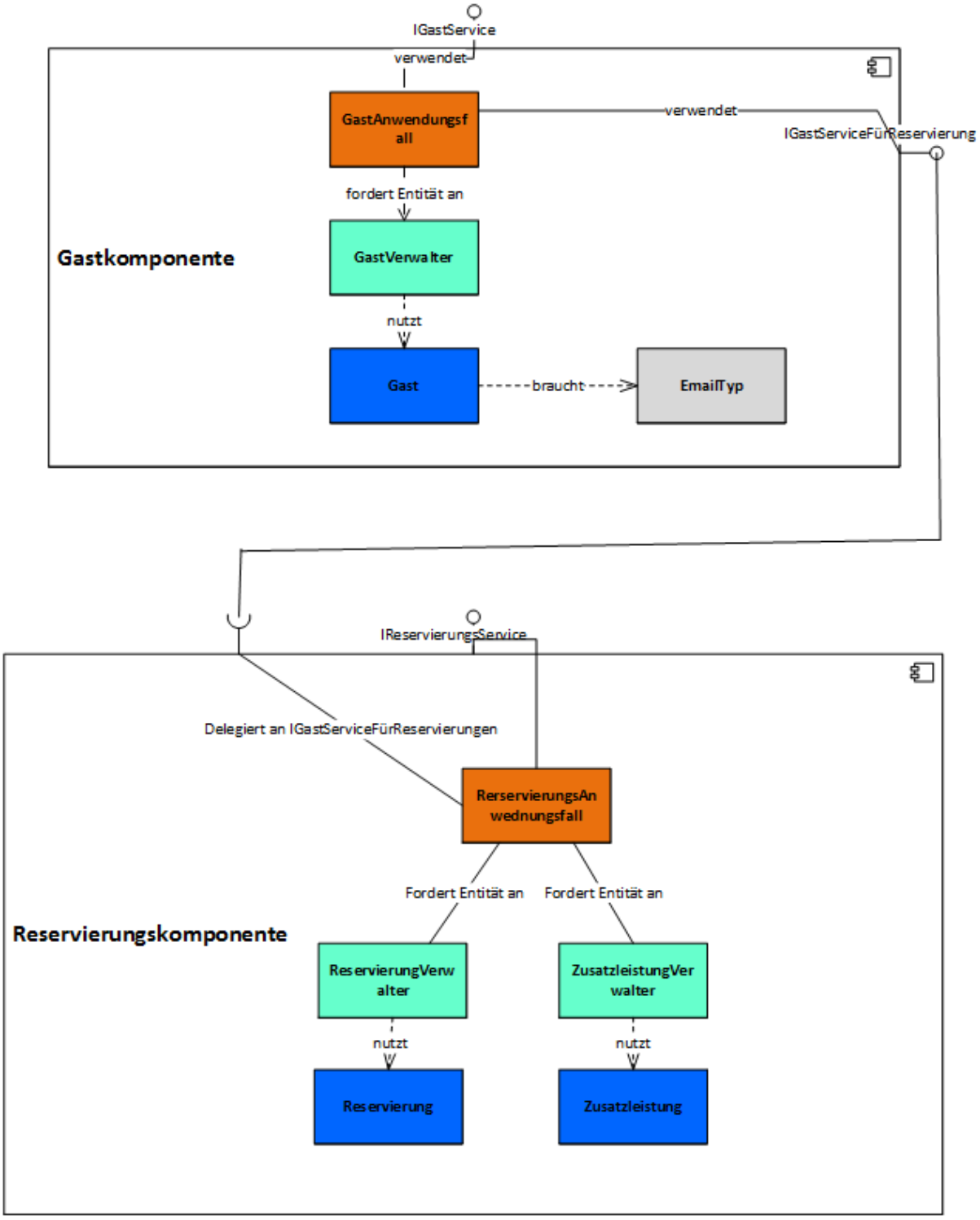
Steffen Giersch & Maria Lüdemann
HAW HAMBURG
16.01.2014

Inhaltsverzeichnis

Steffen Giersch & Maria Lüdemann
HAW HAMBURG
16.01.2014

Aufgabe 9 Implementierung

UML Innensicht



Steffen Giersch & Maria Lüdemann
HAW HAMBURG
16.01.2014

Quellcode

Datenbank Package

ConnectionException.java

```
package db;

import java.sql.SQLException;
/*
 * Eine Exception fuer den Fall, dass die Verbindung zur Datenbank fehlschlaegt
 */
public class ConnectionException extends Exception{

	public ConnectionException() {
		super("Connection to database has failed.");
	}

	public ConnectionException(Exception e){
		//super(e.getMessage().toString());
		super(e.fillInStackTrace());
	}
}
```

DbData.java

```
package db;

/**
 * Datenbank Login Daten
 *
 */
public class DbData {
    public static String USER = "abl128";
    public static String PASS = "
    public static String CON_STRING = "jdbc:oracle:thin:@oracle.informatik.haw-hamburg.de:1521:inf09";

}
```

DBFascade.java

```
package db;

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


public class DBFascade {

	private Connection conn;

	public DBFascade()throws ConnectionException {
		try {
			DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
			conn = DriverManager.getConnection(DbData.CON_STRING, DbData.USER,
DbData.PASS);
		} catch (SQLException e) {
			throw new ConnectionException(e);
		}
	}

	public Connection getConn() {
		return conn;
	}
}
```

Interface Package

Steffen Giersch & Maria Lüdemann
HAW HAMBURG
16.01.2014

Entity_ID.java

```
package ireservierung;

public abstract class Entity_ID {

	protected int idStand;

	public int getID(){
		idStand ++;
		return idStand;
	}

	public void refresh(){
		idStand = 0;
	}

	private int idStand(){
		return idStand;
	}
}
```

IGastAnwendungsfall.java

```
package ireservierung;

public interface IGastAnwendungsfall extends IGastServices, IGastServicesFuerReservierung{

}
```

IGastServices.java

```
package ireservierung;
import reservierung.*;

public interface IReservierungService {

	/**
	 * Eine Funktion die eine Zusatzdienstleistung erzeugt und in die Datenbank speichert
	 *
	 * @param name Der Name der gewünschten Dienstleistung
	 * @return liefert das erzeugte Zusatzdienstleistungs Objekt
	 */
	public Zusatzleistung erzeugeZusatzleistung(String name);

	/**
	 * Reserviert ein Zimmer wobei ein Reservierungsobjekt erzeugt und in die Datenbank gespeichert
wird
	 *
	 * @param gastNr Die ID des Gastes der reserviert
	 * @param zimmerNr Die Nummer des Zimmers das reserviert werden soll
	 * @return Das soeben erzeugte Reservierungsobjekt
	 */
	public Reservierung reserviereZimmer(int gastNr, int zimmerNr);
	/**
	 * Bucht eine Zusatzdienstleistung zu einer Reservierung
	 *
	 * @param reservierungsNr die Nummer der Reservierung auf der gebucht werden soll
	 * @param zusatzleistungNr die Nummer der Dienstleistung die gebucht werden soll
	 */
	public void bucheZusatzleistung(int reservierungsNr, int zusatzleistungNr);
}
```

IGastServicesFuerReservierung.java

```
package ireservierung;

public interface IGastServicesFuerReservierung {
```

```
        /**
         * Eine Funktion, die den angegebenen Gast in der Datenbank auf Stammkunde setzt
         *
         * @param nr Die Nummer des gewünschten Gastes
         */
        public void markiereGastAlsStammkunden(int nr);
}
```

GastKomponentenPackage

EmailTyp.java

```
package gast;
import java.util.regex.Pattern;

public class EmailTyp {

        private String name;
        private String server;
        private String domain;

        //Konstruktor für Strings
        public EmailTyp(String email){
                int name_char = email.indexOf('@');
                int domain_char = email.lastIndexOf('.');
                System.out.println("@" + name_char + "." + domain_char);
                System.out.println(email.substring(0, name_char));
                System.out.println(email.substring(name_char, domain_char));
                System.out.println(email.substring(domain_char, email.length()));

                new EmailTyp(email.substring(0, name_char), email.substring(name_char, domain_char),
email.substring(domain_char, email.length()));
        }
        //Konstruktor mit drei Feldern für die Datenbank. Umständliche Implementierung
        public EmailTyp(String name, String server, String domain) {
                if(name.length() <= 60 && name.length() >= 1 && (Pattern.matches( "[\\w._-]*", name))){
                        this.name = name;
                }else throw new IllegalArgumentException("Der Name ist zu lang");
                if(Pattern.matches( "@[\\w_.-]*", server) && server.length() <= 20 &&  server.length()
>= 1){
                        this.server = server;
                }else throw new IllegalArgumentException("@ Zeichen fehlt oder Server Name lang");

                if(Pattern.matches( ".\\w.*", domain) && domain.length() <= 20 &&  domain.length() >=
1){
                        this.domain = domain;
                }else throw new IllegalArgumentException(". Zeichen fehlt oder Domain Name lang");
        }
        //Getter
        public EmailTyp neueEmail(String name, String server, String domain){
                return new EmailTyp(name, server, domain);
        }

        public String getName(){
                return name;
        }
        public String getServer(){
                return server;
        }
        public String getDomain(){
                return domain;
        }
}
```

Gast,java

```
package gast;

public class Gast{
```

```
	private int nr;
	private String name;
	private EmailTyp email;
	private int stammkunde = 0;

	public Gast(int num,String name, EmailTyp mail) {
		this.nr = num;
		this.name = name;
		this.email = mail;
	}
	public Gast(int num,String name, EmailTyp mail, int istStammKunde) {
		this.nr = num;
		this.name = name;
		this.email = mail;
		this.stammkunde = istStammKunde;
	}

	public int getNr(){
		return nr;
	}

	public String getName(){
		return name;
	}

	public EmailTyp getEmail(){
		return email;
	}

	public void setStammkunde(){
		stammkunde = 1;
	}
	public int getStammkunde(){
		return stammkunde;
	}

}
```

GastAnwendungsfall.java

```
package gast;

import ireservierung.*;
import db.DBFascade;
import db.ConnectionException;


public final class GastAnwendungsfall implements IGastAnwendungsfall{

	//Initialisierung
	static final GastVerwalter verwalter = new GastVerwalter();
	static  DBFascade dbf;

	//Konstrukor
	public GastAnwendungsfall() throws ConnectionException, ClassNotFoundException {
		try{
			dbf = new DBFascade();
		}catch(ConnectionException e){
			throw new ConnectionException(e);
		}
	}

	//Methoden Anfang
	@Override
	public Gast erzeugeGast(int nr, String name, EmailTyp email) {
		Gast gast = verwalter.neuerGast(nr, name, email);
		try{
			verwalter.speichereGast(gast, dbf.getConn());
		}catch(ConnectionException e) {
			e.printStackTrace();
		}
		return gast;
	}
```

```
	@Override
	public Gast sucheGastNachName(String name) {
		try {
			return verwalter.sucheGastNachName(name, verwalter.getAllGast(dbf.getConn()));
		}catch (ConnectionException e) {
			e.printStackTrace();
			return null;
		}
	}

	@Override
	public void markiereGastAlsStammkunden(int nr) {
		try {
			verwalter.markiereGastStammkunde(nr,dbf.getConn());
		}catch (ConnectionException e) {
			e.printStackTrace();
		}
	}
}
```

GastAnwendungsfallMock.java

```
package gast;

import db.ConnectionException;
import db.DBFascade;
import ireservierung.IGastAnwendungsfall;

public class GastAnwendungsfallMock implements IGastAnwendungsfall {

	//Initialisierung
		static final GastVerwalter verwalter = new GastVerwalter();
		static  DBFascade dbf;

		//Konstrukor
		public GastAnwendungsfallMock() throws ConnectionException, ClassNotFoundException {
			try{
				dbf = new DBFascade();
			}catch(ConnectionException e){
				throw new ConnectionException(e);
			}
		}

		//Methoden Anfang
		@Override
		public Gast erzeugeGast(int nr, String name, EmailTyp email) {
			Gast gast = verwalter.neuerGast(nr, name, email);
			try{
				verwalter.speichereGast(gast, dbf.getConn());
			}catch(ConnectionException e) {
				e.printStackTrace();
			}
			return gast;
		}

		@Override
		public Gast sucheGastNachName(String name) {
			try {
				return verwalter.sucheGastNachName(name,
verwalter.getAllGast(dbf.getConn()));
			}catch (ConnectionException e) {
				e.printStackTrace();
				return null;
			}
		}

	@Override
	public void markiereGastAlsStammkunden(int nr) {
		throw new SucessException();

	}
```

```
	public class SucessException extends Error{

		public SucessException() {
			super();
			// TODO Auto-generated constructor stub
		}

		public SucessException(String message, Throwable cause, boolean enableSuppression,
boolean writableStackTrace) {
			super(message, cause, enableSuppression, writableStackTrace);
			// TODO Auto-generated constructor stub
		}

		public SucessException(String message, Throwable cause) {
			super(message, cause);
			// TODO Auto-generated constructor stub
		}

		public SucessException(String message) {
			super(message);
			// TODO Auto-generated constructor stub
		}

		public SucessException(Throwable cause) {
			super(cause);
			// TODO Auto-generated constructor stub
		}


	}

}
```

GastVerwalter.java

```
package gast;

import db.ConnectionException;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class GastVerwalter {

	//Konstruktor
	public GastVerwalter() {

	}

//	public GastVerwalter getGastVerwalter() {
//		return new GastVerwalter();
//	}

	/*
	 * Der Konstrukor der nach oben weitergegeben wird ohne Stammkunde
	 */
	public Gast neuerGast(int num, String name, EmailTyp mail) {
		return new Gast(num, name, mail);
	}
	/*
	 * Der Konstruktor für die Datenbank mit Stammkunde
	 */
	public Gast neuerGast(int num, String name, EmailTyp mail, int istStammKunde) {
		return new Gast(num, name, mail, istStammKunde);
	}

	//Getter Anfang
```

```
	public String getName(Gast gast) {
		return gast.getName();
	}

	public void setStammkunde(Gast gast) {
		gast.setStammkunde();
	}
	//Getter Ende

	//Methoden Anfang
	/*
	 * Holt eine Liste aller Gäste aus der Datenbank
	 */
	public List<Gast> getAllGast(Connection conn) throws ConnectionException {
		try {
			List<Gast> gastListe = new ArrayList<Gast>();
			Statement stmt = conn.createStatement();
			String findAllGastSQL = "SELECT gastID, gastName, emailName, emailServer,
emaildomain, istStammKunde FROM gast";
			ResultSet rset = stmt.executeQuery(findAllGastSQL);
			while (rset.next()) {
				EmailTyp email = new EmailTyp(rset.getString("emailName"),
						rset.getString("emailServer"),
						rset.getString("emailDomain"));
				Gast gast = neuerGast(rset.getInt("gastID"),
						rset.getString("gastName"), email,
						rset.getInt("istStammKunde"));
				gastListe.add(gast);
			}
			return gastListe;
		} catch (Exception e) {
			throw new ConnectionException(e);
		}
	}

	/*
	 * Speichert den übergebenen Gast in die Datenbank
	 * Hier erfolgt keinerlei Verifizierung
	 */
	public void speichereGast(Gast gast, Connection conn)throws ConnectionException {
		try {
			String newGastSQL = "INSERT INTO GAST (gastID, gastName, emailName,
emailServer, emailDomain, istStammkunde) VALUES ("
					+ +gast.getNr()
					+ ", '"
					+ gast.getName()
					+ "', '"
					+ gast.getEmail().getName()
					+ "', '"
					+ gast.getEmail().getServer()
					+ "', '"
					+ gast.getEmail().getDomain()
					+ "', "
					+ gast.getStammkunde()
					+ ")";

			PreparedStatement speichereGast = conn.prepareStatement(newGastSQL);
			speichereGast.execute();

		} catch (SQLException e) {
			throw new ConnectionException(e);
		}
	}

	/*
	 * Sucht in der Datenbank nach dem Gast mit dem übergebenen Namen
	 * und liefert ihn zurück
	 *
	 * Bei nicht finden wird ein Fehler geworfen
	 */
	public Gast sucheGastNachName(String name, List<Gast> gastList) {
		Gast akku = null;

		for (Gast g : gastList) {
```

```
                        if (g.getName().equals(name)) {
                                akku = g;
                                break;
                        }
                }
                if (akku == null) {
                        throw new IllegalArgumentException("Gast nicht gefunden");
                }
                return akku;
        }

        /*
         * Sucht in der Datenbank nach einem Gast mit der übergebenen ID
         *
         * Bei nicht finden wird ein Fehler geworfen
         */
        public Gast sucheGastNachNr(int nr, List<Gast> gastList) {
                Gast akku = null;

                for (Gast g : gastList) {
                        if (g.getNr() == nr) {
                                akku = g;
                                break;
                        }
                }
                if (akku == null) {
                        throw new IllegalArgumentException("Gast nicht gefunden");
                }
                return akku;
        }

        /*
         * Sucht in der Datenbank nach dem Gast mit der übergebenen ID
         * setzt ihn auf Stammkunde und updated den Datenbank eintrag auf Stammkunde
         */
        public void markiereGastStammkunde(int nr, Connection conn) throws ConnectionException {
                try{
                Gast gast = sucheGastNachNr(nr, getAllGast(conn));
                gast.setStammkunde();
                String updateGastSQL = "UPDATE GAST SET istStammkunde = "+gast.getStammkunde()+"  WHERE
gastID = " +nr;

                PreparedStatement speichereGast = conn.prepareStatement(updateGastSQL);
                speichereGast.execute();

                } catch (SQLException e) {
                        throw new ConnectionException(e);
                }
        }
}
```

ReservierungsKomponentenpackage

Reservierung.java

```
package reservierung;

public class Reservierung {


        private int nr;
        private int zimmerNr;

        public Reservierung(int num, int zNr) {
                nr = num;
                zimmerNr = zNr;
        }

        public int getNr(){
                return nr;
        }
```

```
        public int getZimmerNr(){
                return zimmerNr;
        }

}
```

ReservierungsAnwendungsfall.java

```
package reservierung;
import java.util.List;

import db.ConnectionException;
import db.DBFascade;
import gast.GastAnwendungsfall;
import ireservierung.*;

public class ReservierungAnwendungsfall implements IReservierungService {

        //Initialisierung
        static final ReservierungsVerwalter reservierungsVerwalter = new ReservierungsVerwalter();
        static final ZusatzleistungVerwalter zusatzleistungsVerwalter = new ZusatzleistungVerwalter();
        private IGastAnwendungsfall gastAnwendungsfall;

        private Entity_ID resId;
        private Entity_ID zusatzId;
        static  DBFascade dbf;

        //Konstruktor
        public ReservierungAnwendungsfall() throws ConnectionException, ClassNotFoundException {
                this(new GastAnwendungsfall(), new ReservierungID(), new ZusatzID());
        }


        public ReservierungAnwendungsfall(IGastAnwendungsfall gastAnwendungsfall, Entity_ID resId,
Entity_ID zusatzId) throws ConnectionException, ClassNotFoundException {
                super();
                this.gastAnwendungsfall = gastAnwendungsfall;
                this.resId = resId;
                this.zusatzId = zusatzId;
                try{
                        dbf = new DBFascade();
                }catch(ConnectionException e){
                        throw new ConnectionException(e);
                }
        }


        //Methoden Anfang
        @Override
        public Zusatzleistung erzeugeZusatzleistung(String name) {
                Zusatzleistung zusatz =
zusatzleistungsVerwalter.erzeugeZusatzleistung(zusatzId.getID(), name);
                try{
                        zusatzleistungsVerwalter.speicherZusatzleistung(zusatz, dbf.getConn());
                }catch(ConnectionException e) {
                        e.printStackTrace();
                }
                return zusatz;
        }

        @Override
        public Reservierung reserviereZimmer(int gastNr, int zimmerNr){
                Reservierung res = reservierungsVerwalter.neueReservierung(resId.getID(), zimmerNr);
                try{
                        reservierungsVerwalter.speicherReservierung(res,gastNr,dbf.getConn());
                        if( reservierungsVerwalter.countRes(gastNr, dbf.getConn()) >= 5){
                                gastAnwendungsfall.markiereGastAlsStammkunden(gastNr);
                        }
                }catch(Exception e){
                        e.printStackTrace();
                }
```

```
                return res;
        }

        @Override
        public void bucheZusatzleistung(int reservierungsNr, int zusatzleistungNr){
                try{
                        zusatzleistungsVerwalter.speichereZusatzBuchung(reservierungsNr,
zusatzleistungNr, dbf.getConn());
                        int gastNr = reservierungsVerwalter.getGastRes(reservierungsNr,
dbf.getConn());
                        List<Integer> resList = reservierungsVerwalter.getGastReservierungen(gastNr,
dbf.getConn());

                        if(zusatzleistungsVerwalter.countZusatz(resList, dbf.getConn()) >= 3){
                                gastAnwendungsfall.markiereGastAlsStammkunden(gastNr);
                        }
                }catch(ConnectionException e){
                        e.printStackTrace();
                }

        }

}
```

ReservierungID.java

```
package reservierung;

import ireservierung.Entity_ID;

public class ReservierungID extends Entity_ID{


        public ReservierungID() {
                idStand = 0;
        }
}
```

ReservierungsVerwalter.java

```
package reservierung;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

import db.ConnectionException;

public class ReservierungsVerwalter {

        //Konstruktor
        public Reservierung neueReservierung(int num, int zNr){
                return new Reservierung(num, zNr);
        }
        //Getter Anfang
        public int getReservierungNr(Reservierung res){
                return res.getNr();
        }

        public int getReservierungZimmerNr(Reservierung res){
                return res.getZimmerNr();
        }

        //Getter Ende

        //Anfang Methode
        /*
         * Holt alle Reservierungen aus der Datenbank und gibt sie in einer Liste zurück
```

```
	 */
	public List<Reservierung> getAllReservierung(Connection conn) throws ConnectionException {
		try {
			List<Reservierung> reservierungListe = new ArrayList<Reservierung>();
			Statement stmt = conn.createStatement();
			String findAllResSQL = "SELECT reservierungID, zimmerNr FROM reservierung";
			ResultSet rset = stmt.executeQuery(findAllResSQL);
			while (rset.next()) {
				Reservierung res = neueReservierung(rset.getInt("reservierungID"),
rset.getInt("zimmerNr"));
				reservierungListe.add(res);
			}
			return reservierungListe;
		} catch (Exception e) {
			throw new ConnectionException(e);
		}
	}

	/*
	 * Speichert die übergebene Reservierung in die Datenbank und fügt den gastNr Fremdschlüssen
hinzu
	 * hierbei wird keinerlei Validierung vorgenommen
	 */
	public void speicherReservierung(Reservierung res, int gastNr,Connection conn) throws
ConnectionException {

		try {String newResSQL = "INSERT INTO RESERVIERUNG (reservierungID, zimmerNr, fGastNr)
VALUES ("
					+ +res.getNr()
					+ ", "
					+ res.getZimmerNr()
					+ ","
					+ gastNr
					+ ")";

				PreparedStatement speicherReservierung =
conn.prepareStatement(newResSQL);
				speicherReservierung.execute();

		} catch (SQLException e) {
			throw new ConnectionException(e);
		}
	}

		/*
		 * Sucht in der Datenbank nach einer Reservierung mit der übergebenen ID
		 */
		public Reservierung sucheReservierungNachNr(int nr, List<Reservierung> resList) {
			Reservierung akku = null;

			for (Reservierung res : resList) {
				if (res.getNr() == nr) {
					akku = res;
					break;
				}
			}
			if (akku == null) {
				throw new IllegalArgumentException("Reservierung nicht gefunden");
			}
			return akku;
		}

		/*
		 * Sucht in der Datenbank nach der Reservierung mit der übergebenen Nummer und gibt
		 * die ID des Gastes zurück zu der sie gehört
		 */
		public int getGastRes(int resNr, Connection conn)throws ConnectionException{
			int gastNr = -1;
			try {
				Statement stmt = conn.createStatement();

				String findAllResSQL = "SELECT fGastNr FROM Reservierung WHERE
reservierungID = "+ resNr;
				ResultSet rset = stmt.executeQuery(findAllResSQL);
```

```
                                rset.next();
                                gastNr = rset.getInt("fGastNr");

                        } catch (Exception e) {
                                throw new ConnectionException(e);
                        }
                        return gastNr;
                }

                /*
                 * Sucht alle Reservierungen heraus die ein Gast getätigt hat und liefert die IDs in
einer Liste zurück
                 */
                public List<Integer> getGastReservierungen(int gastNr, Connection conn)throws
ConnectionException{
                        try {
                                List<Integer> resListe = new ArrayList<Integer>();
                                Statement stmt = conn.createStatement();
                                String findAllResSQL = "SELECT reservierungID FROM Reservierung WHERE
fGastNr = "+gastNr;
                                ResultSet rset = stmt.executeQuery(findAllResSQL);

                                while (rset.next()) {
                                        resListe.add(rset.getInt("reservierungID"));
                                }
                                return resListe;

                        } catch (Exception e) {
                                throw new ConnectionException(e);
                        }
                }

                /*
                 * Zählt wie viele Reservierungen ein Gast getätigt hat
                 */
                public int countRes(int gastNr, Connection conn)throws ConnectionException{
                        int count = 0;

                        try{
                                Statement stmt = conn.createStatement();
                                String newCountSQL = "SELECT COUNT(*) AS amount FROM Reservierung
WHERE fgastNr = " + gastNr;

                                ResultSet countset = stmt.executeQuery(newCountSQL);
                                countset.next();
                                count = countset.getInt("amount");

                        }catch(SQLException e){
                                throw new ConnectionException(e);
                        }
                        return count;
                }

        }
```

ZusatzID.java

```
package reservierung;

import ireservierung.Entity_ID;

public class ZusatzID extends Entity_ID{

        public ZusatzID() {
                idStand = 0;
        }

}
```

Zusatzleistung.java

```
package reservierung;

public class Zusatzleistung {

	private int nr;
	private String leistungsArt;

	public Zusatzleistung(int num, String lArt) {
		nr = num;
		leistungsArt = lArt;

	}

	public int getNr(){
		return nr;
	}

	public String getLeistungsArt(){
		return leistungsArt;
	}
}
```

MockPackage

GastAnwendungsFallMock.java

```
package mocks;

import db.ConnectionException;
import db.DBFascade;
import gast.EmailTyp;
import gast.Gast;
import gast.GastVerwalter;
import ireservierung.IGastAnwendungsfall;

public class GastAnwendungsfallMock implements IGastAnwendungsfall {

	//Initialisierung
		static final GastVerwalter verwalter = new GastVerwalter();
		static  DBFascade dbf;

		//Konstrukor
		public GastAnwendungsfallMock() throws ConnectionException, ClassNotFoundException {
			try{
				dbf = new DBFascade();
			}catch(ConnectionException e){
				throw new ConnectionException(e);
			}
		}

		//Methoden Anfang
		@Override
		public Gast erzeugeGast(int nr, String name, EmailTyp email) {
			Gast gast = verwalter.neuerGast(nr, name, email);
			try{
				verwalter.speichereGast(gast, dbf.getConn());
			}catch(ConnectionException e) {
				e.printStackTrace();
			}
			return gast;
		}

		@Override
		public Gast sucheGastNachName(String name) {
			try {
				return verwalter.sucheGastNachName(name,
verwalter.getAllGast(dbf.getConn()));
			}catch (ConnectionException e) {
				e.printStackTrace();
```

```
                                return null;
                        }
                }

        @Override
        public void markiereGastAlsStammkunden(int nr) {
                throw new SucessException();

        }


        public class SucessException extends Error{

                public SucessException() {
                        super();
                        // TODO Auto-generated constructor stub
                }

                public SucessException(String message, Throwable cause, boolean enableSuppression,
boolean writableStackTrace) {
                        super(message, cause, enableSuppression, writableStackTrace);
                        // TODO Auto-generated constructor stub
                }

                public SucessException(String message, Throwable cause) {
                        super(message, cause);
                        // TODO Auto-generated constructor stub
                }

                public SucessException(String message) {
                        super(message);
                        // TODO Auto-generated constructor stub
                }

                public SucessException(Throwable cause) {
                        super(cause);
                        // TODO Auto-generated constructor stub
                }
        }
}
```

Entity_ID_Mock.java

```
package mocks;

import ireservierung.Entity_ID;

public class Entity_ID_Mock extends Entity_ID {

        public Entity_ID_Mock(int start) {
                super();
                this.idStand = start;
        }
}
```

Aufgabe 10 Test

TestPackage

ReserverungsAnwendungsfallTest.java

```
package Test;

import static org.junit.Assert.*;

import gast.EmailTyp;
import gast.Gast;
import gast.GastAnwendungsfall;
import ireservierung.IGastAnwendungsfall;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;

import mocks.Entity_ID_Mock;
import mocks.GastAnwendungsfallMock;

import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;

import reservierung.Reservierung;
import reservierung.ReservierungAnwendungsfall;
import reservierung.ReservierungsVerwalter;
import reservierung.Zusatzleistung;
import reservierung.ZusatzleistungVerwalter;
import db.ConnectionException;
import db.DbData;

public class ReservierungAnwendungsfallTest {
	static Connection conn = null;
	static GastAnwendungsfall gastAn = null;
	static ReservierungAnwendungsfall resAn = null;

	@BeforeClass
	public static void setUpBeforeClass() {
		try {
			DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
			conn = DriverManager.getConnection(DbData.CON_STRING, DbData.USER,
DbData.PASS);
			gastAn = new GastAnwendungsfall();
			resAn = new ReservierungAnwendungsfall();

		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

	@Before
	public void setUp() throws ConnectionException {
		try {
			DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
			Connection conn = DriverManager.getConnection(DbData.CON_STRING, DbData.USER,
DbData.PASS);

		} catch (SQLException e) {
			throw new ConnectionException(e);
		}
	}

	@After
	public void tearDown() throws Exception {
		conn.close();
	}

	@Test(expected = GastAnwendungsfallMock.SucessException.class)
	public void testZusatsStammgast() {
		IGastAnwendungsfall gastAn = null;
		ReservierungAnwendungsfall resAn = null;
		try {
			gastAn = new GastAnwendungsfallMock();
			resAn = new ReservierungAnwendungsfall(gastAn, new Entity_ID_Mock(1000),new
Entity_ID_Mock(1000));
		} catch (Exception e) {
			e.printStackTrace();
		}

		EmailTyp email = new EmailTyp("timzain", "@yahoo", ".de");
```

```
		Gast gast = gastAn.erzeugeGast(1, "Tom Tompson", email);

		assertEquals(0, gast.getStammkunde());

		Zusatzleistung sauna = resAn.erzeugeZusatzleistung("Sauna");
		Zusatzleistung vollPension = resAn.erzeugeZusatzleistung("Vollpension");
		Zusatzleistung wlan = resAn.erzeugeZusatzleistung("WLAN");

		Reservierung res = resAn.reserviereZimmer(gast.getNr(), 22);
		resAn.bucheZusatzleistung(res.getNr(), sauna.getNr());
		res = resAn.reserviereZimmer(gast.getNr(), 11);
		resAn.bucheZusatzleistung(res.getNr(), vollPension.getNr());
		res = resAn.reserviereZimmer(gast.getNr(), 66);
		resAn.bucheZusatzleistung(res.getNr(), wlan.getNr());

		gast = gastAn.sucheGastNachName("Tom Tompson");

		assertEquals(1, gast.getStammkunde());
	}

	@Test(expected = GastAnwendungsfallMock.SucessException.class)
	public void testReservierungStammgast() {
		IGastAnwendungsfall gastAn = null;
		ReservierungAnwendungsfall resAn = null;
		try {
			gastAn = new GastAnwendungsfallMock();
			resAn = new ReservierungAnwendungsfall(gastAn, new Entity_ID_Mock(500), new
Entity_ID_Mock(500));
		} catch (Exception e) {
			e.printStackTrace();
		}

		EmailTyp email1 = new EmailTyp("paulchen", "@yahoo", ".de");
		Gast gast1 = gastAn.erzeugeGast(2, "Paul Port", email1);
		assertEquals(gast1.getStammkunde(), 0);

		resAn.reserviereZimmer(gast1.getNr(), 22);
		resAn.reserviereZimmer(gast1.getNr(), 11);
		resAn.reserviereZimmer(gast1.getNr(), 66);
		resAn.reserviereZimmer(gast1.getNr(), 33);
		resAn.reserviereZimmer(gast1.getNr(), 44);
	}

	// Integrationstest
	@Test
	public void integrationsTest() {
		// Zusatz in die Datenbank
		Zusatzleistung zusatz = resAn.erzeugeZusatzleistung("Sektempfang");

		// Gast in die Datenbanl
		EmailTyp email1 = new EmailTyp("tonyTob", "@yahoo", ".de");
		Gast gast = gastAn.erzeugeGast(3, "Tony Tobago", email1);

		// Reservierung in die Datenbank
		Reservierung res = resAn.reserviereZimmer(gast.getNr(), 11);
		resAn.bucheZusatzleistung(res.getNr(), zusatz.getNr());

		// Gast richtig in die Datenbank?
		Gast gast1 = gastAn.sucheGastNachName("Tony Tobago");
		assertEquals(3, gast1.getNr());
		assertEquals("Tony Tobago", gast1.getName());
		assertEquals("tonyTob", gast1.getEmail().getName());
		assertEquals("@yahoo", gast1.getEmail().getServer());
		assertEquals(".de", gast1.getEmail().getDomain());
		assertEquals(0, gast1.getStammkunde());

		// Reservierung richtig in die Datenbank?
		try {
			ReservierungsVerwalter resVer = new ReservierungsVerwalter();
			Reservierung res1 = resVer.sucheReservierungNachNr(0,
resVer.getAllReservierung(conn));

			assertEquals(0, res1.getNr());
			assertEquals(11, res1.getZimmerNr());
```

Steffen Giersch & Maria Lüdemann
HAW HAMBURG
16.01.2014

```
                        // Zusatz richtig in die Datenbank?
                        ZusatzleistungVerwalter zuVer = new ZusatzleistungVerwalter();
                        Zusatzleistung zusatz1 = zuVer.sucheZusatzleistungNachArt("Sektempfang",
conn);

                        assertEquals(0, zusatz.getNr());
                        assertEquals("Sektempfang", zusatz1.getLeistungsArt());

                } catch (Exception e) {
                        e.getStackTrace();
                }

                //Stammgast
                resAn.reserviereZimmer(gast.getNr(), 22);
                resAn.reserviereZimmer(gast.getNr(), 66);
                resAn.reserviereZimmer(gast.getNr(), 33);
                resAn.reserviereZimmer(gast.getNr(), 44);

                Gast gast2 = gastAn.sucheGastNachName("Tony Tobago");
                assertEquals(1, gast2.getStammkunde());
        }
}
```