МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Основы объектно-реляционного отображения (ORM)

(наименование темы проекта или работы)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе по дисциплине

Управление данными

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Балашова Т.И.\_

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Опарин С. А.

(подпись) (фамилия, и.,о.)

17-СБК

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород

2020 г.

**Задание:**

1. Создать maven-проект в среде IntelliJ IDEA EE

-Сконфигурировать pom.xml

-Добавить поддержку фреймворка Hibernate

-Сконфигурировать src/main/resources/hibernate.cfg.xml

-Создать java-класс, отображающий сущность из бд

-Заполнить класс сущности

-Добавить подключение к БД

1. Создать класс с реализацией CRUD-операций с таблицей (CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete)). Написать самим операции удаления записи, обновления записи, чтения по id объекта.

**Ход работы:**

Листинг программного кода:

**Class Dentist.java**

package com.Dentist.model;

import javax.persistence.\*;

import java.math.BigInteger;

import java.sql.Date;

import java.util.Objects;

@Entity

@Embeddable

@Table(name = "dentist")

public class Dentist {

@Id

@Column(name = "dentist\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private BigInteger dentistId;

@Column(name = "first\_name")

private String surname;

@Column(name = "last\_name")

private String name;

@Column(name = "birthday")

private Date birth;

@Column(name = "specialization")

private String spec;

@Column(name = "carier\_start\_date")

private Date st\_date;

@Column(name = "work\_phone")

private String w\_phone;

public BigInteger getDentistId() {

return dentistId;

}

public void setDentistId(BigInteger dentistId) {

this.dentistId = dentistId;

System.out.println("111");

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public Date getBirth() {

return birth;

}

public void setBirth(String birth) {this.birth = Date.valueOf(birth);}

public String getSpec() {

return spec;

}

public void setSpec(String spec){

this.spec = spec;

}

public Date getStDate(){

return st\_date;

}

public void setStDate(String start\_date){this.st\_date = Date.valueOf(start\_date);}

public String getWPhone(){

return w\_phone;

}

public void setWPhone(String phone){

this.w\_phone = phone;

}

public Dentist(BigInteger dentistId, String name, String surname, String birth, String spec, String date\_of\_start, String work\_phone) {

this.dentistId = dentistId;

this.name = name;

this.surname = surname;

this.birth = Date.valueOf(birth);

this.spec = spec;

this.st\_date = Date.valueOf(date\_of\_start);

this.w\_phone = work\_phone;

}

public Dentist() { }

@Override

public String toString() {

return "Dentist{" + "ID=" + dentistId + ", name: '" + name + '\'' + ", surname: '" + surname + '\'' + ", birth: '" + birth +'\'' + ", specialization: '" + spec + '\'' + ", work phone: '" + w\_phone + '\'' + ", start his/her career at: '" + st\_date + '\'' + '}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (!(o instanceof Dentist))

return false;

Dentist dentist = (Dentist) o;

return Objects.equals(getDentistId(), dentist.getDentistId());

}

@Override

public int hashCode() {

int result = dentistId.hashCode();

result = 31 \* result + name.hashCode();

result = 31 \* result + surname.hashCode();

result = 31 \* result + w\_phone.hashCode();

return result;

}

}

**class DentistCrud.java**

package com.Dentist;

import com.Dentist.model.Dentist;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

import org.junit.Test;

import java.math.BigInteger;

import java.util.List;

public class DentistCrud {

@Test

public void crud() {

SessionFactory sessionFactory = new Configuration().configure().buildSessionFactory();

Session session = sessionFactory.openSession();

create(session, "Andrey", "Ponin", "1989-04-10", "ortodontist","2010-02-06", "89109875734");

read\_by\_id(session, get\_cur\_id());

update\_by\_id(session, get\_cur\_id());

read(session);

delete\_by\_id(session, get\_cur\_id());

read(session);

session.close();

}

private Dentist dentist = null;

private void create(Session session, String d\_name, String d\_surname, String d\_birth,

String d\_specialization, String d\_start\_work\_date, String d\_work\_phone) {

Dentist dentist = new Dentist();

dentist.setName(d\_name);

dentist.setSurname(d\_surname);

dentist.setBirth(d\_birth);

dentist.setSpec(d\_specialization);

dentist.setStDate(d\_start\_work\_date);

dentist.setWPhone(d\_work\_phone);

session.beginTransaction();

session.save(dentist);

session.getTransaction().commit();

this.dentist = dentist;

}

private void read(Session session) {

System.out.println("All rows: ");

List<Dentist> dentistList = session.createQuery("SELECT s FROM Dentist s").list();

for (Dentist d : dentistList) {

System.out.println("List:" + d);

}

}

private long get\_cur\_id() {

if(this.dentist != null)

return this.dentist.getDentistId().longValue();

else

return -1;

}

private void get\_dent\_by\_id(Session session, long id) {

List<Dentist> dentistList = session.createQuery("SELECT s FROM Dentist s ").list();

for (Dentist d : dentistList) {

if (d.getDentistId().longValue() == id);

this.dentist = d;

}

}

private void read\_by\_id(Session session, long id){

List<Dentist> dentistList = session.createQuery("SELECT s FROM Dentist s ").list();

for (Dentist d : dentistList) {

if (d.getDentistId().longValue() == id)

System.out.println("New row with ID = " + id + ": " + d.toString());

}

}

private void update\_by\_id(Session session, long id){

if(id == -1)

System.out.println("ID doesn't exist");

else{

get\_dent\_by\_id(session, id);

if (id == this.dentist.getDentistId().longValue()) {

get\_dent\_by\_id(session, id);

System.out.println("Updating ID " + id);

dentist.setBirth("1989-04-11");

System.out.println("Birthday was updated on " + dentist.getBirth().toString());

Transaction tx = session.beginTransaction();

session.merge(dentist);

tx.commit();

}

}

}

private void delete\_by\_id(Session session, long id){

if (id == -1)

System.out.println("Wrong ID");

else {

get\_dent\_by\_id(session, id);

if (id == this.dentist.getDentistId().longValue()) {

get\_dent\_by\_id(session, id);

System.out.println("Deleting ID " + id);

Transaction tx = session.beginTransaction();

session.delete(dentist);

tx.commit();

}

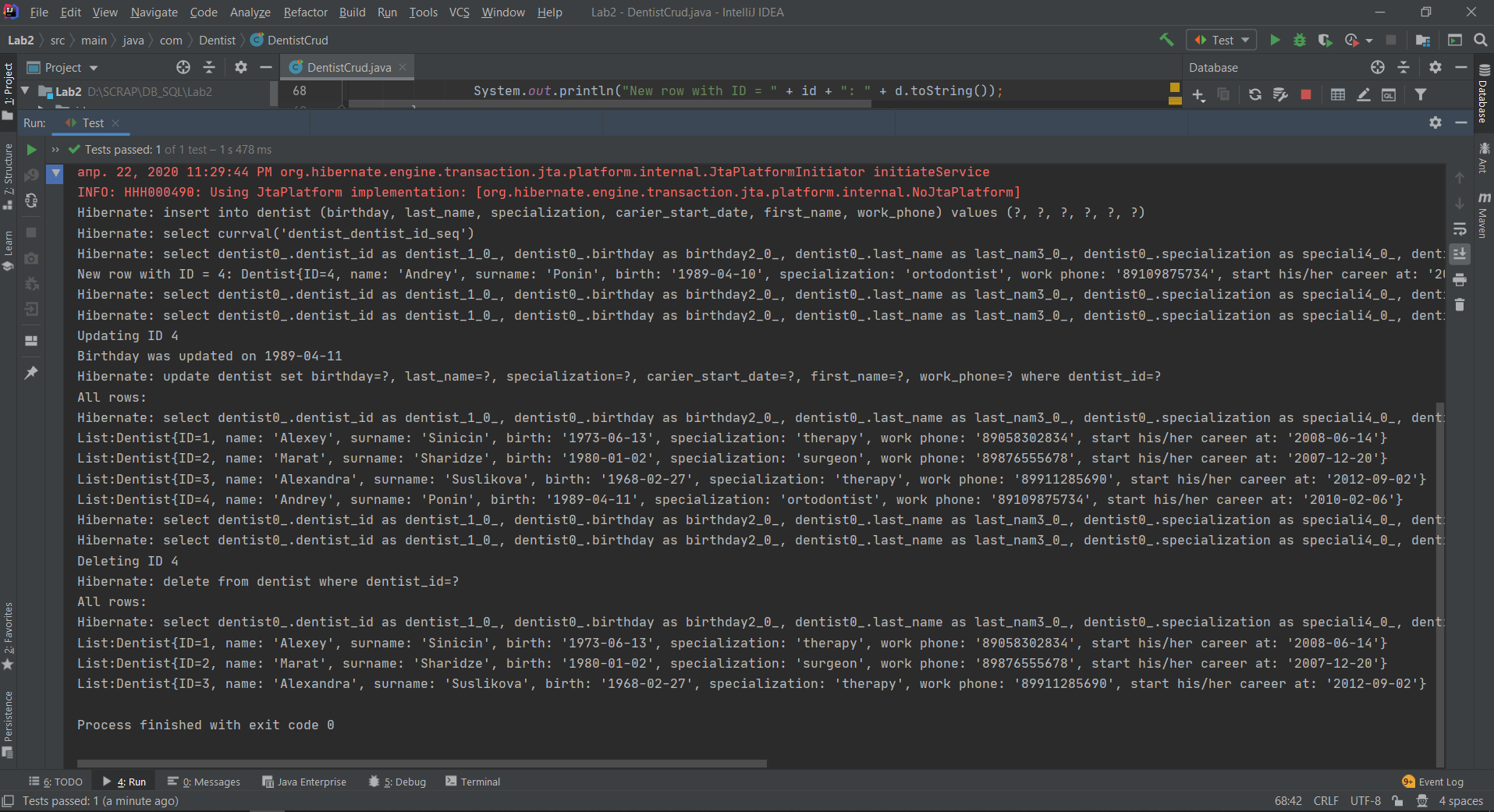
}

}

}

Работа программы:

Тестируемый метод crud класса DentistCrud.java прошёл тест.



Вывод: таким образом, мы создали и сконфигурировали Maven-проект, сделали Java-классы, отображающие сущности из БД, а также создали тестовый класс с реализацией CRUD.