Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ***ИУК «Информатика и Управление»***

КАФЕДРА ***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

ЛАБОРАТОРНАЯ РАБОТА №1

ДИСЦИПЛИНА: «Кроссплатформенная разработка ПО»

Выполнил: студент гр. ИУК4-62Б _____ (Калашников А. С.)
(Подпись) (Ф.И.О.)

Проверил: _____ (Пчелинцева Н. Н.)
(Подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2023

Цель: Получить навык разработки приложения с использованием объектно-реляционного отображения при помощи фреймворка Hibernate на языке Java.
Задачи:
1. Разработать модель предметной области.
2. Получить навыки программирования на языке Java.
3. Освоить реализацию основных принципов ООП.
4. Разобраться и применить ORM-подход на базе фреймворка Hibernate.

Задание:

Отделение платных услуг в больнице. Пациент заключает договор на лечение в платном отделении больницы. Различные процедуры могут делать разные врачи. Учет процедур введется в карте пациента. Стоимость процедуры зависит от базовой цены и коэффициента врача. Стоимость договора рассчитывается из цен оказанных процедур. Минимальный набор сущностей: пациент, врач, карта пациента, договор.

Минимальный набор атрибутов: ФИО пациента, дата рождения, контактные данные; ФИО врача, должность, коэффициент, контактные данные; номер карты, наименование процедуры, базовая цена, врач, дата проведения; номер договора, дата, пациент, список процедур, итоговая стоимость.

Диаграмма UML:

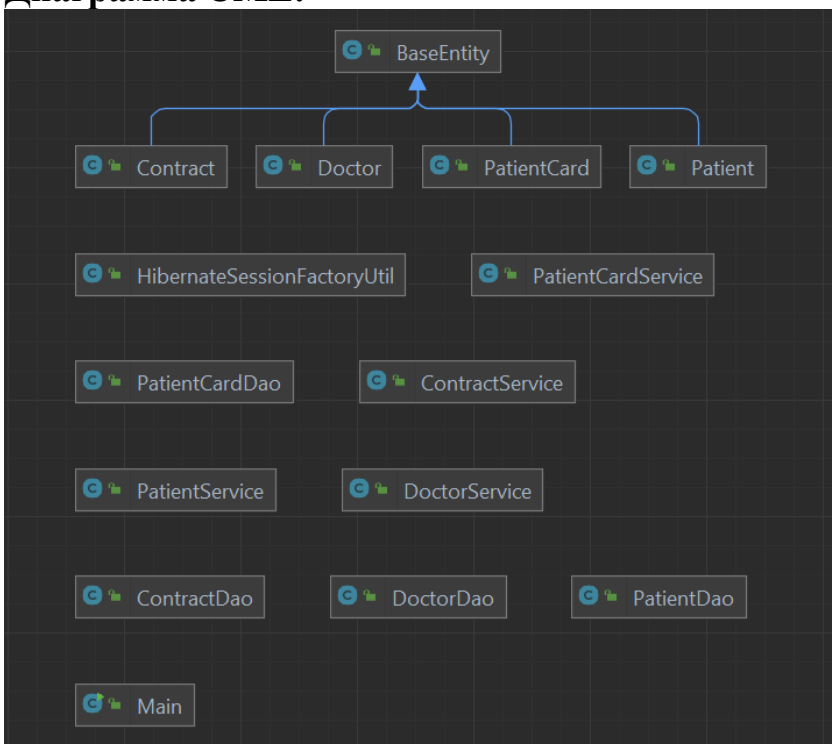
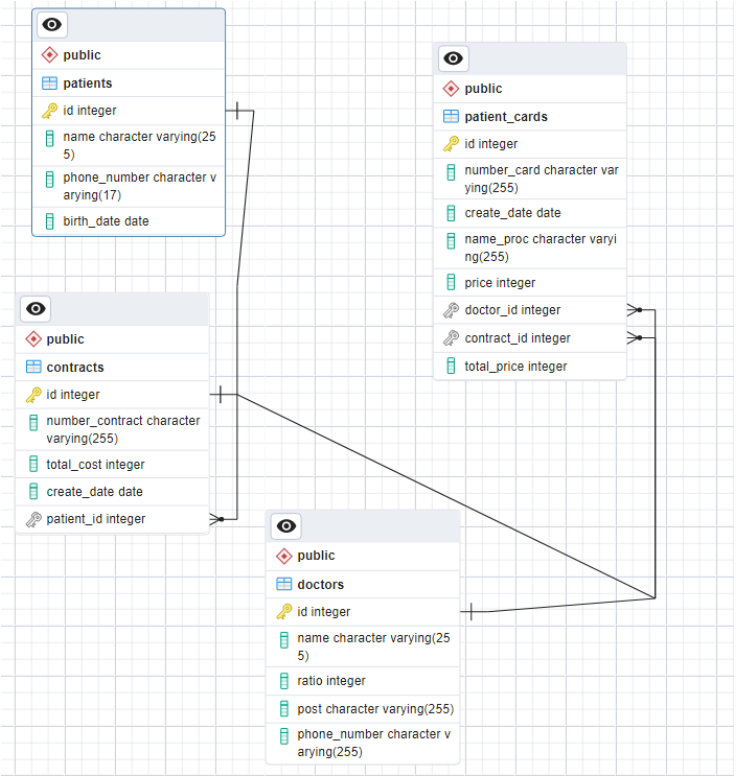


Диаграмма БД:



Код:

ContractDao

```
package dao;
import models.Contract;
import models.Patient;
import models.PatientCard;
import org.hibernate.Session;
import org.hibernate.Transaction;
import utils.HibernateSessionFactoryUtil;
import java.util.List;
public class ContractDao {

    public Contract findById(int id) {
        return
HibernateSessionFactoryUtil.getSessionFactory().openSession().get(Contract.cl
ass, id);
    }

    public void save(Contract contract) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.save(contract);
        tx1.commit();
        session.close();
    }

    public void update(Contract contract) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.update(contract);
        tx1.commit();
        session.close();
```

```
    }

    public void delete(Contract contract) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.delete(contract);
        tx1.commit();
        session.close();
    }

    public List<Contract> findAll() {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        List<Contract> contracts = (List<Contract>)
session.createQuery("From Contract").list();
        session.close();
        return contracts;
    }

    public List<PatientCard> findAllPatientCard(Contract contract) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        var query = session.createQuery("FROM PatientCard WHERE
contract=:contract");
        query.setParameter("contract", contract);
        List<PatientCard> patientCards = (List<PatientCard>) query.list();

        session.close();
        return patientCards;
    }


}
```

DoctorDao

```
package dao;

import models.Doctor;
import org.hibernate.Session;
import org.hibernate.Transaction;
import utils.HibernateSessionFactoryUtil;

import java.util.List;

public class DoctorDao {

    public Doctor findById(int id) {
        return
HibernateSessionFactoryUtil.getSessionFactory().openSession().get(Doctor.clas
s, id);
    }

    public void save(Doctor doctor) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.save(doctor);
        tx1.commit();
        session.close();
```

```
    }

    public void update(Doctor doctor) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.update(doctor);
        tx1.commit();
        session.close();
    }

    public void delete(Doctor doctor) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.delete(doctor);
        tx1.commit();
        session.close();
    }

    public List<Doctor> findAll() {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        List<Doctor> doctors = (List<Doctor>)  session.createQuery("From
Doctor").list();
        session.close();
        return doctors;
    }


}
```

PatientDao

```
package dao;

import models.Contract;
import models.Patient;
import org.hibernate.Session;
import org.hibernate.Transaction;
import utils.HibernateSessionFactoryUtil;
import java.util.List;

public class PatientDao {

    public Patient findById(int id) {
        return
HibernateSessionFactoryUtil.getSessionFactory().openSession().get(Patient.cla
ss, id);
    }

    public void save(Patient patient) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.save(patient);
        tx1.commit();
        session.close();
    }

    public void update(Patient patient) {
        Session session =
```

```
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.update(patient);
        tx1.commit();
        session.close();
    }

    public void delete(Patient patient) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.delete(patient);
        tx1.commit();
        session.close();
    }

    public Contract findContractById(int id) {
        return
HibernateSessionFactoryUtil.getSessionFactory().openSession().get(Contract.cl
ass, id);
    }

    public List<Patient> findAll() {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        List<Patient> patients = (List<Patient>)  session.createQuery("FROM
Patient").list();
        session.close();
        return patients;
    }
}
```

PatientCardDao

```
package dao;

import models.PatientCard;
import org.hibernate.Session;
import org.hibernate.Transaction;
import utils.HibernateSessionFactoryUtil;

import java.util.List;

public class PatientCardDao {

    public PatientCard findById(int id) {
        return
HibernateSessionFactoryUtil.getSessionFactory().openSession().get(PatientCard
.class, id);
    }

    public void save(PatientCard patientCard) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.save(patientCard);
        tx1.commit();
        session.close();
    }

    public void update(PatientCard patientCard) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
```

```
        Transaction tx1 = session.beginTransaction();
        session.update(patientCard);
        tx1.commit();
        session.close();
    }

    public void delete(PatientCard patientCard) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx1 = session.beginTransaction();
        session.delete(patientCard);
        tx1.commit();
        session.close();
    }

    public List<PatientCard> findAll() {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        List<PatientCard> patientCards = (List<PatientCard>)
session.createQuery("From PatientCard").list();
        session.close();
        return patientCards;
    }


}
```

BaseEntity

```
package models;
import jakarta.persistence.*;

import java.sql.Date;


@MappedSuperclass
public class BaseEntity {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    protected int id;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

Contract

```
package models;
import java.util.List;
import jakarta.persistence.*;
import java.sql.Date;

@Entity
@Table(name = "contracts")
public class Contract extends BaseEntity{

    @Column (name = "number_contract")
    private String numberContract;
```

```
    //можно не указывать Column name, если оно совпадает с названием столбца
в таблице
    @Column (name = "total_cost")
    private int totalCost;

    @Column (name = "create_date")
    private Date createDate;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "patient_id")
    private Patient patient;

    @OneToMany(mappedBy = "contract")
    private List<PatientCard> patientCards;

    public Contract(int id, String numberContract, int totalCost, Date
createDate, Patient patient, List<PatientCard> patientCards) {
        this.id = id;
        this.numberContract = numberContract;
        this.totalCost = totalCost;
        this.createDate = createDate;
        this.patient = patient;
    }
    public Contract(String numberContract, int totalCost, Date createDate,
Patient patient, List<PatientCard> patientCards) {
        this.numberContract = numberContract;
        this.totalCost = totalCost;
        this.createDate = createDate;
        this.patient = patient;
    }

    public Contract() {
    }

    public Date getCreateDate() {
        return createDate;
    }

    public void setCreateDate(Date createDate) {
        this.createDate = createDate;
    }

    public String getNumberContract() {
        return numberContract;
    }

    public void setNumberContract(String numberContract) {
        this.numberContract = numberContract;
    }

    public int getTotalCost() {
        return totalCost;
    }

    public void setTotalCost(int totalCost) {
        this.totalCost = totalCost;
    }

    public Patient getPatient() {
        return patient;
    }
```

```
    public void setPatient(Patient patient) {
        this.patient = patient;
    }

    public List<PatientCard> getPatientCards() {
        return patientCards;
    }

    public void setPatientCard(List<PatientCard> patientCards) {
        this.patientCards = patientCards;
    }
    @Override
    public String toString() {
        return numberContract + " " + totalCost;
    }


}
```

Doctor

```
package models;

import jakarta.persistence.*;
@Entity
@Table (name = "doctors")
public class Doctor extends BaseEntity{


    @Column(name="name")
    private String name;

    @Column(name="post")
    private String post;

    @Column(name="ratio")
    private int ratio;
    @Column(name="phone_number", length=16, nullable=true)
    private String phoneNumber;

    public Doctor(int id, String name,String post, int ratio, String
phoneNumber) {
        this.id = id;
        this.name = name;
        this.post=post;
        this.ratio = ratio;
        this.phoneNumber = phoneNumber;

    }
    public Doctor(String name,String post, int ratio, String phoneNumber) {
        this.name = name;
        this.post=post;
        this.ratio = ratio;
        this.phoneNumber = phoneNumber;

    }
    public Doctor() {}

    public String getName() {
        return name;
    }
```

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

    public String getPhoneNumber() {
        return phoneNumber;
    }

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

    public String getPost() {
        return post;
    }

    public void setPost(String post) {
        this.post = post;
    }

    public int getRatio() {
        return ratio;
    }

    public void setRatio(int ratio) {
        this.ratio = ratio;
    }

}
```

Patient

```
package models;

import java.util.ArrayList;
import java.util.List;
import jakarta.persistence.*;
import java.sql.Date;

@Entity
@Table (name = "patients")
public class Patient extends BaseEntity{

    @Column(name = "name")
    private String name;
    @Column(name="phone_number", length=16, nullable=true)
    private String phoneNumber;
    @Column(name = "birth_date", nullable=false)
    private Date birthDate;

    public Patient() {}

    public Patient(int id,String name,String phoneNumber,Date birthDate) {
        this.id =id;
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
    }
    public Patient(String name,String phoneNumber,Date birthDate) {
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.birthDate = birthDate;
```

```
    }

    public void addContract(Contract contract) {
        contract.setPatient(this);
    }


    public String getName() {
        return name;
    }

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

    public String getPhoneNumber() {
        return phoneNumber;
    }

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

    public Date getBirthDate() {
        return birthDate;
    }

    public void setBirthDate(Date birthDate){
        this.birthDate = birthDate;
    }

    @Override
    public String toString() {
        return "models.User{" +
                "id=" + id +
                ", name='" + name + '\''+ '}';
    }
}
```

PatientCard

```
package models;

import jakarta.persistence.*;

import java.sql.Date;

@Entity
@Table(name = "patient_cards")
public class PatientCard extends BaseEntity{

    @Column(name = "number_card")
    private String numberCard;

    @Column (name = "create_date")
    private Date createDate;

    @Column (name = "name_proc")
    private String nameProc;

    @Column (name = "price")
    private int price;
```

```
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "doctor_id")
    private Doctor doctor;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "contract_id")
    private Contract contract;

    @Column(name = "total_price")
    private int totalPrice;

    public PatientCard() {}

    public PatientCard(int id, String numberCard,String nameProc,Date
createDate,int price,Doctor doctor, Contract contract) {
        this.id = id;
        this.numberCard = numberCard;
        this.nameProc = nameProc;
        this.createDate = createDate;
        this.price = price;
        this.doctor = doctor;
        this.contract=contract;
        this.totalPrice = price * doctor.getRatio();


    }
    public PatientCard(String numberCard,String nameProc,Date createDate,int
price,Doctor doctor, Contract contract) {
        this.numberCard = numberCard;
        this.nameProc = nameProc;
        this.createDate = createDate;
        this.price = price;
        this.doctor = doctor;
        this.contract=contract;
        this.totalPrice = price * doctor.getRatio();


    }

    public int getTotalPrice() {
        return totalPrice;
    }

    public void setTotalPrice() {this.totalPrice = this.price *
this.doctor.getRatio();}

    public String getNumberCard() {
        return numberCard;
    }

    public void setNumberCard(String numberCard) {
        this.numberCard = numberCard;
    }

    public String getNameProc() {
        return nameProc;
    }

    public void setNameProc(String nameProc) {
        this.nameProc = nameProc;
    }
```

```
    public Date getCreateDate() {
        return createDate;
    }

    public void setCreateDate(Date createDate) {
        this.createDate = createDate;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public Doctor getDoctor() {
        return doctor;
    }

    public void setDoctor(Doctor doctor) {
        this.doctor = doctor;
    }

    public Contract getContract() {
        return contract;
    }

    public void setContract(Contract contract) {
        this.contract = contract;
    }


}
```

ContractService

```
package services;

import java.util.List;

import dao.ContractDao;
import models.Contract;
import models.PatientCard;


public class ContractService {

    private ContractDao contractsDao = new ContractDao();

    public ContractService() {
    }

    public Contract findContract(int id) {
        return contractsDao.findById(id);
    }

    public void saveContract(Contract contract) {
        contractsDao.save(contract);
    }

    public void deleteContract(Contract contract) {
```

```
        contractsDao.delete(contract);
    }

    public void updateContract(Contract contract) {
        contractsDao.update(contract);
    }

    public List<Contract> findAllContracts() {
        return contractsDao.findAll();
    }

    public List<PatientCard> findAllPatientCard(Contract contract) {
        return contractsDao.findAllPatientCard(contract);
    }


}
```

PatientService

```
package services;

import dao.PatientDao;
import models.Contract;
import models.Patient;

import java.util.List;

public class PatientService {

    private static PatientDao patientsDao = new PatientDao();

    public PatientService() {
    }

    public Patient findPatient(int id) {
        return patientsDao.findById(id);
    }

    public void savePatient(Patient patient) {
        patientsDao.save(patient);
    }

    public void deletePatient(Patient patient) {
        patientsDao.delete(patient);
    }

    public void updatePatient(Patient patient) {
        patientsDao.update(patient);
    }

    public List<Patient> findAllPatients() {
        return patientsDao.findAll();
    }

//    public Contract findPatientById(int id) {
//        return patientsDao.findContractById(id);
//    }


}
```

DoctorService

```
package services;

import dao.DoctorDao;
import models.Doctor;

import java.util.List;


public class DoctorService {

    private DoctorDao doctorsDao = new DoctorDao();

    public DoctorService() {
    }

    public Doctor findDoctor(int id) {
        return doctorsDao.findById(id);
    }

    public void saveDoctor(Doctor doctor) {
        doctorsDao.save(doctor);
    }

    public void deleteDoctor(Doctor doctor) {
        doctorsDao.delete(doctor);
    }

    public void updateDoctor(Doctor doctor) {
        doctorsDao.update(doctor);
    }

    public List<Doctor> findAllDoctors() {
        return doctorsDao.findAll();
    }


}
```

PatientCardService

```
package services;

import dao.PatientCardDao;
import models.PatientCard;

import java.util.List;


public class PatientCardService {

    private PatientCardDao patientCardsDao = new PatientCardDao();

    public PatientCardService() {
    }

    public PatientCard findPatientCard(int id) {
        return patientCardsDao.findById(id);
    }

    public void savePatientCard(PatientCard patientCard) {
        patientCardsDao.save(patientCard);
```

```
    }

    public void deletePatientCard(PatientCard patientCard) {
        patientCardsDao.delete(patientCard);
    }

    public void updatePatientCard(PatientCard patientCard) {
        patientCardsDao.update(patientCard);
    }

    public List<PatientCard> findAllPatientCards() {
        return patientCardsDao.findAll();
    }


}
```

Решение:

```
Поликлиника

1) Просмотр таблицы
2) Добавить данные
3) Изменить данные:
4) Удалить данные:
0) Выйти
-> 1


Вывод таблицы

1) Просмотр таблицы пациента
2) Просмотр таблицы доктора
3) Просмотр таблицы карты пациента
4) Просмотр таблицы контракта
0) Выйти

-> 1
Hibernate: select p1_0.id,p1_0.birth_date,p1_0.name,p1_0.phone_number from patients p1_0

Текущие пациенты
-----------------------------------------------------------------------------------------------
       ID                                    ФИО        Дата рождения       Номер телефона
-----------------------------------------------------------------------------------------------
        1                                   Alex          2001-12-31         +79106055606

Нажмите для продолжения ...
```

Рис.2 Вывод данных таблицы

```
Удалить данные из таблицы

1 - Карты пациента
2 - Пациенты
3 - Доктора
4 - Контракты
0 - Вернуться назад

-> 2

Hibernate: select p1_0.id,p1_0.birth_date,p1_0.name,p1_0.phone_number from patients p1_0
Удаление пациента
Выберите пациента для удаления
Текущие пациенты
-----------------------------------------------------------------------------------------------
        ID                                    ФИО        Дата рождения       Номер телефона
-----------------------------------------------------------------------------------------------
         1                                  Artem           2001-12-31         +79106055606
         2                                  Artem           2001-12-31         +79106055606

Введите ID пациента: 2
Hibernate: delete from patients where id=?

Нажмите для продолжения ...
```

Рис.3 Добавление новых данных

```
Обновление данных таблицы

1 - Карты пациента
2 - Пациенты
3 - Доктора
4 - Контракты
0 - Вернуться назад

-> 2

Обновление текущих пациентов
Выберите пациента для обновления
Текущие пациенты
-----------------------------------------------------------------------------------------------
        ID                                    ФИО        Дата рождения       Номер телефона
-----------------------------------------------------------------------------------------------
         1                                  Artem           2001-12-31         +79106055606

Введите ID пациента: 1
Введите параметр, который хотите обновить: ФИО
Введите новое ФИО: Alex
Hibernate: update patients set birth_date=?, name=?, phone_number=? where id=?

Нажмите для продолжения ...
```

Рис.4 Изменение данных

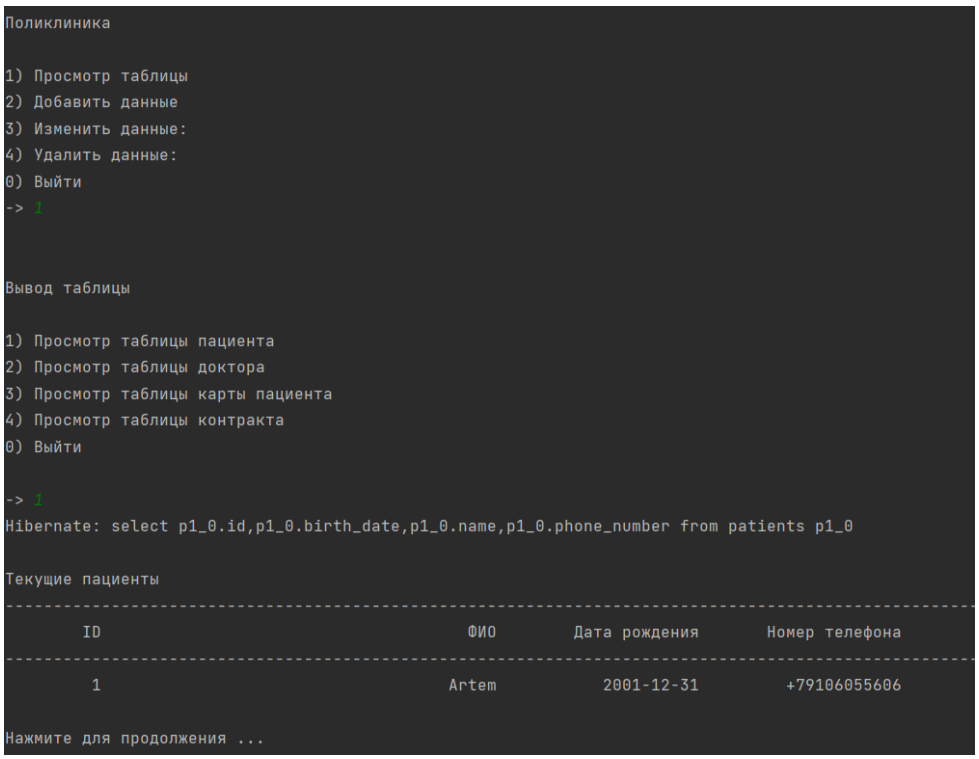


Рис.5 Удаление данных

Вывод: в ходе выполнения лабораторной работы были получены навыки разработки приложения с использованием объектно-реляционного отображения при помощи фреймворка Hibernate на языке Java..