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

«Java. Hibernate»

ДИСЦИПЛИНА: «Кроссплатформенная разработка ПО»

Выполнил: студент гр. ИУК4-62Б _____ (Карельский М.К.)
(Подпись)

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

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:

Разработать модель предметной области, определить сущности и их атрибуты. Реализовать консольное приложение на языке Java с использованием фреймворка Hibernate. При разработке приложения использовать принципы ООП. Для хранения данных использовать реляционную базу данных. Для доступа к данным в приложении использовать технологию ORM на базе фреймворка Hibernate.

Приложение должно обеспечить возможность чтения, добавления, обновления информации в БД; производить обработку и вычисление необходимых атрибутов сущности модели предметной области; взаимодействовать с пользователем посредством меню.

Вариант 8

Типография занимается выпуском печатной продукции. Для выполнения работ типография заключает договор с клиентом. Стоимость работ зависит от вида продукции и тиража издания. При составлении договора учитываются персональные скидки клиентов.

Минимальный набор сущностей: клиент, продукция, договор.

Минимальный набор атрибутов: ФИО клиента, контактные данные, персональная скидка; наименование продукции, стоимость за единицу; номер договора, дата, клиент, продукция, тираж, итоговая стоимость.

Описание и UML-диаграмма предметной области:

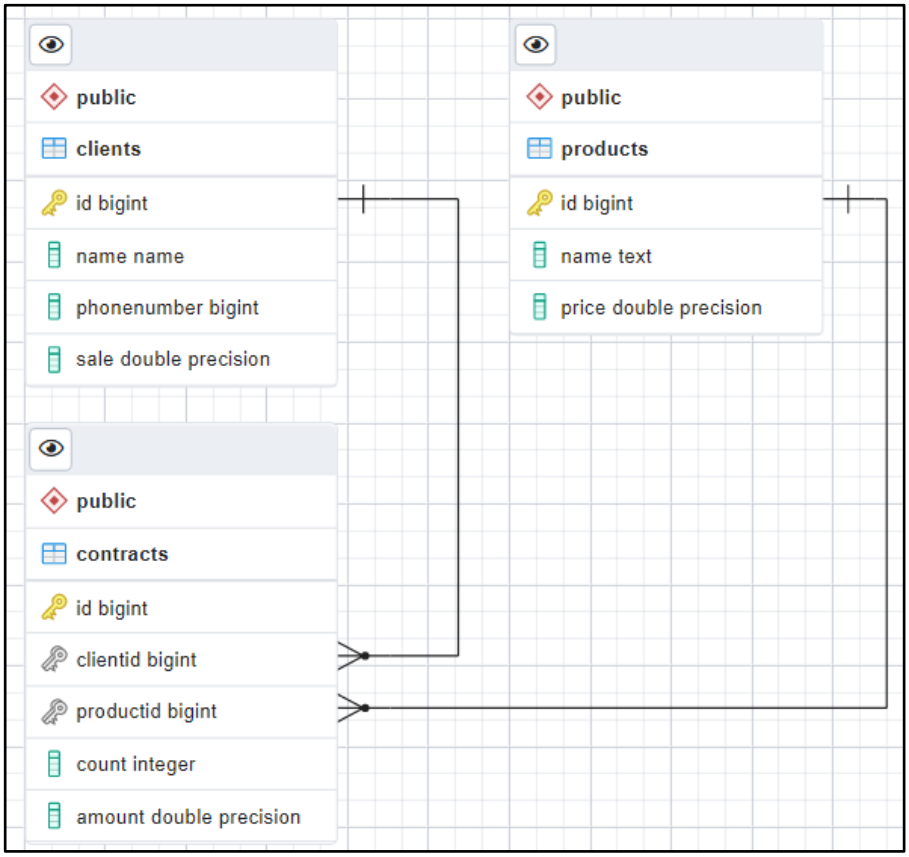

Рис. 1. База данных

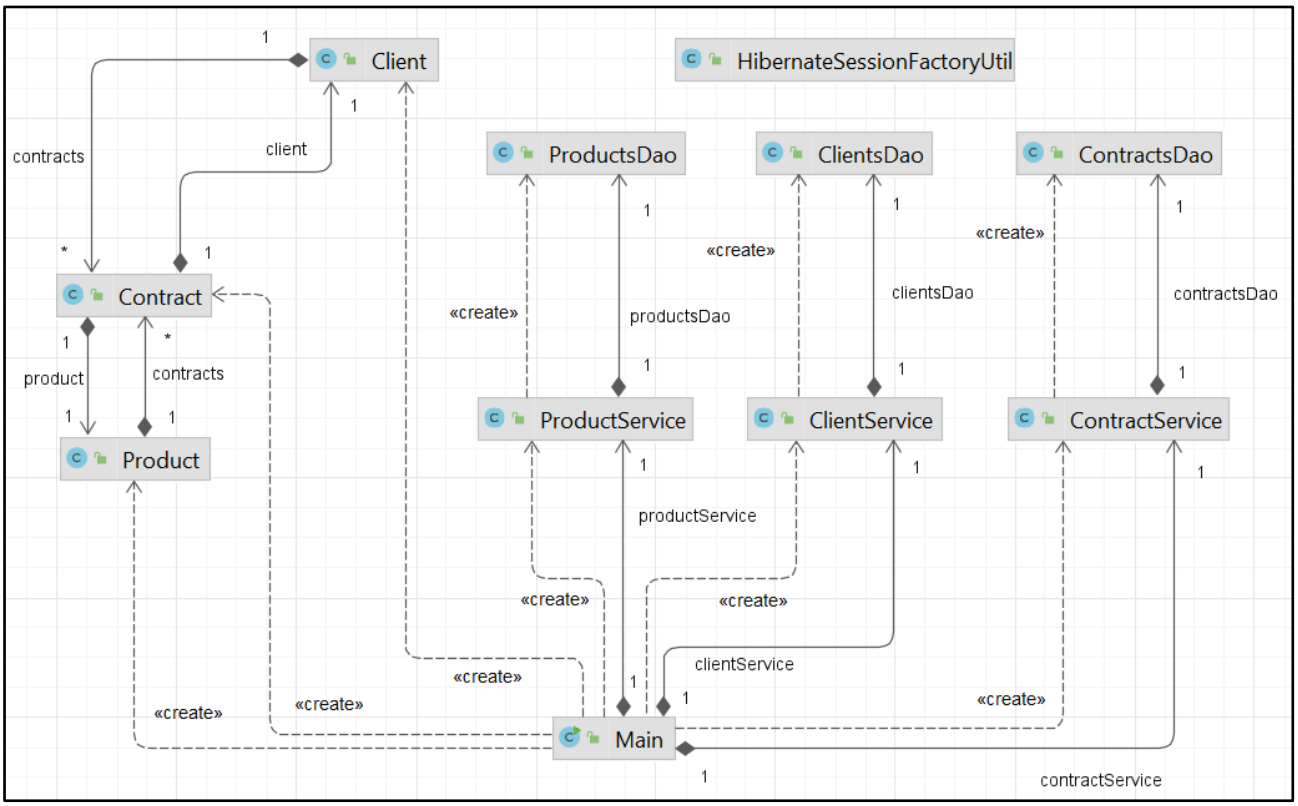

Рис. 2. Структура классов

Листинг:

Таблица clients

```
CREATE TABLE IF NOT EXISTS public.clients
(
    id bigint NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 9223372036854775807 CACHE 1 ),
    name name COLLATE pg_catalog."C" NOT NULL,
    phonenumber bigint,
    sale double precision NOT NULL DEFAULT 0,
    CONSTRAINT "Client_pkey" PRIMARY KEY (id)
)
```

Таблица products

```
CREATE TABLE IF NOT EXISTS public.products
(
    id bigint NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 9223372036854775807 CACHE 1 ),
    name text COLLATE pg_catalog."default" NOT NULL,
    price double precision NOT NULL DEFAULT 0,
    CONSTRAINT "Product_pkey" PRIMARY KEY (id)
)
```

Таблица contracts

```
CREATE TABLE IF NOT EXISTS public.contracts
(
    id bigint NOT NULL GENERATED BY DEFAULT AS IDENTITY ( INCREMENT 1 START 1
MINVALUE 1 MAXVALUE 9223372036854775807 CACHE 1 ),
    clientid bigint NOT NULL,
    productid bigint NOT NULL,
    count integer NOT NULL DEFAULT 1,
    amount double precision,
    CONSTRAINT "Contracts_pkey" PRIMARY KEY (id),
    CONSTRAINT "ClientId" FOREIGN KEY (clientid)
        REFERENCES public.clients (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "ProductId" FOREIGN KEY (productid)
        REFERENCES public.products (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

Триггер update_amount_by_client

```
CREATE CONSTRAINT TRIGGER update_amount_by_client
    AFTER UPDATE OF sale
    ON public.clients
    FOR EACH ROW
    EXECUTE FUNCTION public.update_amount();
```

Триггер update_amount_by_product

```
CREATE CONSTRAINT TRIGGER update_amount_by_product
    AFTER UPDATE OF price
    ON public.products
    FOR EACH ROW
    EXECUTE FUNCTION public.update_amount();
```

Триггер update_amount_by_contract

```
CREATE CONSTRAINT TRIGGER update_amount_by_contract
    AFTER INSERT OR UPDATE OF productid, count
    ON public.contracts
    FOR EACH ROW
    EXECUTE FUNCTION public.update_amount();
```

Триггерная функция update_amount

```
CREATE OR REPLACE FUNCTION public.update_amount()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    UPDATE public.Contracts SET Amount = (
        SELECT Count * prc * (100 - sl) * 0.01 FROM (
            SELECT Price as prc, Sale as sl
            FROM public.Products
            JOIN public.Clients ON public.Clients.Id = ClientId
            WHERE public.Products.Id = ProductId
        ) AS f
    );
    return NEW;
END;
$BODY$;
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>Typography</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>19</maven.compiler.source>
        <maven.compiler.target>19</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.5.3</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>6.2.0.CR2</version>
        </dependency>
        <dependency>
            <groupId>jakarta.persistence</groupId>
            <artifactId>jakarta.persistence-api</artifactId>
            <version>3.1.0</version>
        </dependency>
```

```
    </dependencies>
</project>
```

hibernate.cfg.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
        <property
name="connection.driver_class">org.postgresql.Driver</property>
        <property
name="connection.url">jdbc:postgresql://localhost:5432/postgres</property>
        <property name="connection.username">postgres</property>
        <property name="connection.password">2458173671</property>
        <property
name="dialect">org.hibernate.dialect.PostgreSQLDialect</property>
        <property name="hibernate.show_sql">false</property>
        <property name="hibernate.format_sql">false</property>
        <property name="hibernate.enable_lazy_load_no_trans">true</property>
    </session-factory>
</hibernate-configuration>
```

HibernateSessionFactoryUtil.java

```
package utils;

import models.Client;
import models.Contract;
import models.Product;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class HibernateSessionFactoryUtil {
    private static SessionFactory sessionFactory;

    private HibernateSessionFactoryUtil() {}

    public static SessionFactory getSessionFactory() {
        if (sessionFactory == null) {
            try {
                Configuration configuration = new Configuration().configure();

                configuration.addAnnotatedClass(Client.class);
                configuration.addAnnotatedClass(Product.class);
                configuration.addAnnotatedClass(Contract.class);

                StandardServiceRegistryBuilder builder =
                        new
StandardServiceRegistryBuilder().applySettings(configuration.getProperties());

                sessionFactory =
configuration.buildSessionFactory(builder.build());
            } catch (Exception e) {
                System.out.println("Error: " + e);
            }
        }

        return  sessionFactory;
    }
}
```

Client.java

```
package models;

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

@Entity
@Table(name = "Clients")
public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Id")
    private long id;
    @Column(name = "Name")
    private String name;
    @Column(name = "PhoneNumber")
    private long phoneNumber;
    @Column(name = "Sale")
    private double sale;
    @OneToMany(mappedBy = "client", cascade = CascadeType.ALL, orphanRemoval =
true)
    private List<Contract> contracts;

    public Client() {}
    public Client(String name, long phoneNumber, double sale){
        this.name = name;
        this.phoneNumber = phoneNumber;
        this.sale = sale;
        contracts = new ArrayList<>();
    }

    public void AddContract(Contract contract) {
        contract.SetClient(this);
        contracts.add(contract);
    }
    public void RemoveContract(Contract contract) {
        contracts.remove(contract);
    }

    public long GetId() { return id; }
    public String GetName() { return name; }
    public long GetPhoneNumber() { return phoneNumber; }
    public double GetSale() { return sale; }
    public List<Contract> GetContracts() { return contracts; }

    public void SetName(String name) { this.name = name; }
    public void SetPhoneNumber(long phoneNumber) { this.phoneNumber =
phoneNumber; }
    public void SetSale(double sale) { this.sale = sale; }
    public void SetContracts(List<Contract> contracts) { this.contracts =
contracts; }
}
```

Product.java

```
package models;

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

@Entity
@Table(name = "Products")
```

```
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Id")
    private long id;
    @Column(name = "Name")
    private String name;
    @Column(name = "Price")
    private double price;
    @OneToMany(mappedBy = "product", cascade = CascadeType.ALL, orphanRemoval =
true)
    private List<Contract> contracts;

    public Product() {}
    public Product(String name, double price) {
        this.name = name;
        this.price = price;
        contracts = new ArrayList<>();
    }

    public void AddContract(Contract contract) {
        contract.SetProduct(this);
        contracts.add(contract);
    }
    public void RemoveContract(Contract contract) {
        contracts.remove(contract);
    }

    public long GetId() { return id; }
    public String GetName() { return name; }
    public double GetPrice() { return price; }
    public List<Contract> GetContracts() { return contracts; }

    public void SetName(String name) { this.name = name; }
    public void SetPrice(double price) { this.price = price; }
    public void SetContracts(List<Contract> contracts) { this.contracts =
contracts; }
}
```

Contract.java

```
package models;

import jakarta.persistence.*;

@Entity
@Table(name = "Contracts")
public class Contract {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Id")
    private long id;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "ClientId")
    private Client client;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "ProductId")
    private Product product;
    @Column(name = "Count")
    private int count;
    @Column(name = "Amount")
    private double amount;

    public Contract() {}
    public Contract(Client client, Product product, int count) {
```

```
        this.client = client;
        this.product = product;
        this.count = count;
    }

    public long GetId() { return id; }
    public Client GetClient() { return client; }
    public Product GetProduct() { return product; }
    public int GetCount() { return count; }
    public double GetAmount() { return amount; }

    public void SetClient(Client client) { this.client = client; }
    public void SetProduct(Product product) { this.product = product; }
    public void SetCount(int count) { this.count = count; }
}
```

ClientsDao.java

```
package dao;

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

public class ClientsDao {
    public Client FindById(long id) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Client client = session.get(Client.class, id);
        session.close();
        return client;
    }

    public List<Client> FindAll() {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        List<Client> clients = (List<Client>) session.createQuery("From
Client").list();
        session.close();
        return clients;
    }

    public void Save(Client client) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.save(client);
        tx.commit();
        session.close();
    }

    public void Update(Client client) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.update(client);
        tx.commit();
        session.close();
    }

    public void Delete(Client client) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
```

```
        Transaction tx = session.beginTransaction();
        session.delete(client);
        tx.commit();
        session.close();
    }
}
```

ProductsDao.java

```
package dao;

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

public class ProductsDao {
    public Product FindById(long id) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Product product = session.get(Product.class, id);
        session.close();
        return product;
    }

    public List<Product> FindAll() {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        List<Product> products = (List<Product>) session.createQuery("From
Product").list();
        session.close();
        return products;
    }

    public void Save(Product product) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.save(product);
        tx.commit();
        session.close();
    }

    public void Update(Product product) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.update(product);
        tx.commit();
        session.close();
    }

    public void Delete(Product product) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.delete(product);
        tx.commit();
        session.close();
    }
}
```

ContractsDao.java

```
package dao;

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

import java.util.List;

public class ContractsDao {
    public Contract FindById(long id) {
        Session session =
HibernateSessionFactoryUtil.getSessionFactory().openSession();
        Contract contract = session.get(Contract.class, id);
        session.close();
        return contract;
    }

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

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

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

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

ClientService.java

```
package services;

import dao.ClientsDao;
import models.Client;
import java.util.List;
```

```
public class ClientService {
    private ClientsDao clientsDao = new ClientsDao();

    public ClientService() {}

    public Client FindClient(long id) { return clientsDao.FindById(id); }
    public List<Client> FindAllClients() { return clientsDao.FindAll(); }
    public void SaveClient(Client client) { clientsDao.Save(client); }
    public void DeleteClient(Client client) { clientsDao.Delete(client); }
    public void UpdateClient(Client client) { clientsDao.Update(client); }
}
```

ProductService.java

```
package services;

import dao.ProductsDao;
import models.Product;
import java.util.List;

public class ProductService {
    private ProductsDao productsDao = new ProductsDao();

    public ProductService() {}

    public Product FindProduct(long id) { return productsDao.FindById(id); }
    public List<Product> FindAllProducts() { return productsDao.FindAll(); }
    public void SaveProduct(Product product) { productsDao.Save(product); }
    public void DeleteProduct(Product product) { productsDao.Delete(product); }
    public void UpdateProduct(Product product) { productsDao.Update(product); }
}
```

ContractService.java

```
package services;

import dao.ContractsDao;
import models.Contract;
import java.util.List;

public class ContractService {
    private ContractsDao contractsDao = new ContractsDao();

    public ContractService() {}

    public Contract FindContract(long id) { return contractsDao.FindById(id); }
    public List<Contract> FindAllContracts() { return contractsDao.FindAll(); }
    public void SaveContract(Contract contract) { contractsDao.Save(contract); }
    public void DeleteContract(Contract contract) {
contractsDao.Delete(contract); }
    public void UpdateContract(Contract contract) {
contractsDao.Update(contract); }
}
```

Main.java

```
package org.example;

import models.Client;
import models.Contract;
import models.Product;
import services.ClientService;
import services.ContractService;
import services.ProductService;
```

```
import java.sql.SQLException;
import java.util.Collections;
import java.util.Scanner;

public class Main {
    static ClientService clientService = new ClientService();
    static ProductService productService = new ProductService();
    static ContractService contractService = new ContractService();

    static long InputLong() {
        Scanner scanner = new Scanner(System.in);
        return scanner.nextLong();
    }

    static double InputDouble() {
        Scanner scanner = new Scanner(System.in);
        return scanner.nextDouble();
    }

    static int InputInt() {
        Scanner scanner = new Scanner(System.in);
        return scanner.nextInt();
    }

    static String InputString() {
        Scanner scanner = new Scanner(System.in);
        return scanner.nextLine();
    }

    static void ShowClients() {
        System.out.printf("%10s %40s %20s %20s",
                "ID |",
                "ФИО |",
                "Номер телефона |",
                "Персональная скидка");
        System.out.println();
        System.out.print(String.join("", Collections.nCopies(93, "-")));
        System.out.println();
        for (var client : clientService.FindAllClients()) {
            System.out.printf("%10s %40s %20s %20s",
                    client.GetId() + " |",
                    client.GetName()  + " |",
                    client.GetPhoneNumber()  + " |",
                    client.GetSale() + "%");
            System.out.println();
        }
    }

    static void ShowProducts() {
        System.out.printf("%10s %40s %20s",
                "ID |",
                "Наименование |",
                "Стоимость за единицу");
        System.out.println();
        System.out.print(String.join("", Collections.nCopies(72, "-")));
        System.out.println();
        for (var product : productService.FindAllProducts()) {
            System.out.printf("%10s %40s %20s",
                    product.GetId() + " |",
                    product.GetName()  + " |",
                    product.GetPrice() + "p");
            System.out.println();
        }
    }
```

```
static void ShowContracts() {
    System.out.printf("%10s %15s %15s %10s %10s",
            "ID |",
            "ID клиента |",
            "ID продукции |",
            "Тираж |",
            "Стоимость");
    System.out.println();
    System.out.print(String.join("", Collections.nCopies(64, "-")));
    System.out.println();
    for (var contract : contractService.FindAllContracts()) {
        System.out.printf("%10s %15s %15s %10s %10s",
                contract.GetId() + " |",
                contract.GetClient().GetId()  + " |",
                contract.GetProduct().GetId()  + " |",
                contract.GetCount()  + " |",
                contract.GetAmount() + "р");
        System.out.println();
    }
}

static Client ChooseClient()
{
    ShowClients();
    System.out.print("ID: ");
    long id = InputLong();
    return clientService.FindClient(id);
}

static Product ChooseProduct()
{
    ShowProducts();
    System.out.print("ID: ");
    long id = InputLong();
    return productService.FindProduct(id);
}

static Contract ChooseContract()
{
    ShowContracts();
    System.out.print("ID: ");
    long id = InputLong();
    return contractService.FindContract(id);
}

static void AddClient() {
    System.out.print("ФИО: ");
    String name = InputString();
    System.out.print("Телефон: ");
    long phoneNumber = InputLong();
    System.out.print("Скидка: ");
    double sale = InputDouble();

    Client client = new Client(name, phoneNumber, sale);
    clientService.SaveClient(client);
}

static void AddProduct() {
    System.out.print("Название: ");
    String name = InputString();
    System.out.print("Цена: ");
    double price = InputDouble();

    Product product = new Product(name, price);
    productService.SaveProduct(product);
}
```

```
static void AddContract() {
    System.out.println("Клиент");
    Client client = ChooseClient();
    System.out.println("Продукция");
    Product product = ChooseProduct();
    System.out.print("Тираж: ");
    int count = InputInt();

    Contract contract = new Contract(client, product, count);
    contractService.SaveContract(contract);
}

static void EditClient() {
    Client client = ChooseClient();
    System.out.println();

    System.out.println("Выберите поле");
    System.out.println("1. ФИО");
    System.out.println("2. Телефон");
    System.out.println("3. Скидка");
    System.out.print(">>> ");
    int code = InputInt();
    System.out.println();

    System.out.print("Новое значение: ");
    switch (code) {
        case 1 -> client.SetName(InputString());
        case 2 -> client.SetPhoneNumber(InputLong());
        case 3 -> client.SetSale(InputDouble());
        default -> {
            return;
        }
    }

    clientService.UpdateClient(client);
}

static void EditProduct() {
    Product product = ChooseProduct();
    System.out.println();

    System.out.println("Выберите поле");
    System.out.println("1. Название");
    System.out.println("2. Цена");
    System.out.print(">>> ");
    int code = InputInt();
    System.out.println();

    System.out.print("Новое значение: ");
    switch (code) {
        case 1 -> product.SetName(InputString());
        case 2 -> product.SetPrice(InputDouble());
        default -> {
            return;
        }
    }

    productService.UpdateProduct(product);
}

static void EditContract() {
    Contract contract = ChooseContract();
    System.out.println();

    System.out.println("Выберите поле");
```

```
        System.out.println("1. Клиент");
        System.out.println("2. Продукция");
        System.out.println("3. Тираж");
        System.out.print(">>> ");
        int code = InputInt();
        System.out.println();

        System.out.print("Новое значение: ");
        if (code == 1 || code == 2) System.out.println();
        switch (code) {
            case 1 -> contract.SetClient(ChooseClient());
            case 2 -> contract.SetProduct(ChooseProduct());
            case 3 -> contract.SetCount(InputInt());
            default -> {
                return;
            }
        }

        contractService.UpdateContract(contract);
    }

    static void DeleteClient() {
        clientService.DeleteClient(ChooseClient());
    }

    static void DeleteProduct() {
        productService.DeleteProduct(ChooseProduct());
    }

    static void DeleteContract() {
        contractService.DeleteContract(ChooseContract());
    }

    static boolean ShowMenu() {
        System.out.println("1. Просмотреть");
        System.out.println("2. Добавить");
        System.out.println("3. Изменить");
        System.out.println("4. Удалить");
        System.out.println("0. Выход");
        System.out.print(">>> ");
        int action = InputInt();

        if (action == 0) return false;
        if (action < 0 || action > 4) {
            System.out.println("Ошибка ввода");
            return true;
        }

        System.out.println();
        System.out.println("1. Клиентов");
        System.out.println("2. Продукцию");
        System.out.println("3. Договоры");
        System.out.print(">>> ");
        int table = InputInt();

        if (table < 1 || table > 3) {
            System.out.println("Ошибка ввода");
            return true;
        }

        System.out.println();
        if (action == 1) {
            switch (table) {
                case 1 -> ShowClients();
                case 2 -> ShowProducts();
                case 3 -> ShowContracts();
```

```
            }
        } else if (action == 2) {
            switch (table) {
                case 1 -> AddClient();
                case 2 -> AddProduct();
                case 3 -> AddContract();
            }
        } else if (action == 3) {
            switch (table) {
                case 1 -> EditClient();
                case 2 -> EditProduct();
                case 3 -> EditContract();
            }
        } else if (action == 4) {
            switch (table) {
                case 1 -> DeleteClient();
                case 2 -> DeleteProduct();
                case 3 -> DeleteContract();
            }
        } else {
            System.out.println("Ошибка ввода");
        }
        System.out.println();

        return true;
    }

    public static void main(String[] args) throws SQLException {
        while (ShowMenu());
    }
}
```

Результат:

```
1. Просмотреть
2. Добавить
3. Изменить
4. Удалить
0. Выход
>>> 1

1. Клиентов
2. Продукцию
3. Договоры
>>> 1

      ID |                                  ФИО |     Номер телефона |  Персональная скидка
---------------------------------------------------------------------------------------------
       3 |                  Иванов Иван Иванович |        88005553535 |                 3.5%
       4 |                  Петров Петр Петрович |        88002344842 |                 5.5%
```

Рис. 3. Пример просмотра таблицы

```
1. Просмотреть
2. Добавить
3. Изменить
4. Удалить
0. Выход
>>> 2

1. Клиентов
2. Продукцию
3. Договоры
>>> 1

ФИО: Никитов Никита Никитич
Телефон: 74842536892
Скидка: 0
```

Рис. 4. Пример добавления

```
1. Просмотреть
2. Добавить
3. Изменить
4. Удалить
0. Выход
>>> 3

1. Клиентов
2. Продукцию
3. Договоры
>>> 2

      ID |                          Наименование | Стоимость за единицу
-------------------------------------------------------------------------
       2 |                              Визитки |                 4.0р
ID: 2

Выберите поле
1. Название
2. Цена
>>> 2

Новое значение: 3
```

Рис. 5. Пример изменения

```
1. Просмотреть
2. Добавить
3. Изменить
4. Удалить
0. Выход
>>> 4

1. Клиентов
2. Продукцию
3. Договоры
>>> 3

      ID |    ID клиента |  ID продукции |    Тираж |  Стоимость
-----------------------------------------------------------------
       2 |             3 |             2 |      100 |     289.5р
ID: 2
```

Рис. 6. Пример удаления

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