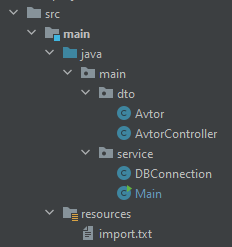
Структура проекта:



Класс DBConnection

Класс Avtor

Класс AvtorController

Изображение выглядит как текст, апельсин

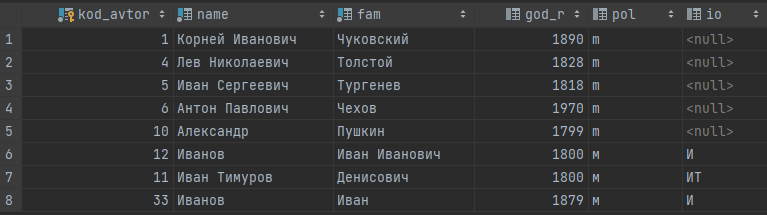
Автоматически созданное описаниеКласс Main – внутри создается экземпляр класса DBConnection, обеспечивающий подключение к БД, и экземпляр класса Avtor, создающий нового автора. Также вызываются функции класса AvtorController для манипулирования данными в БД.

Изображение выглядит как текст

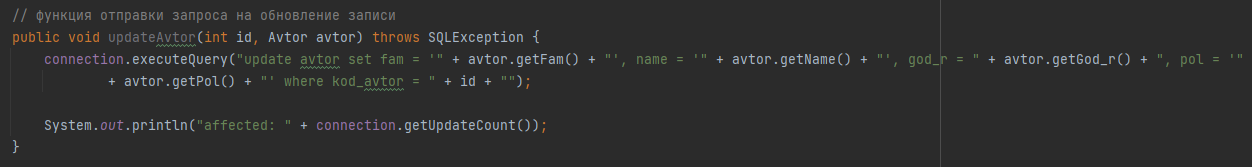
Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание







Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

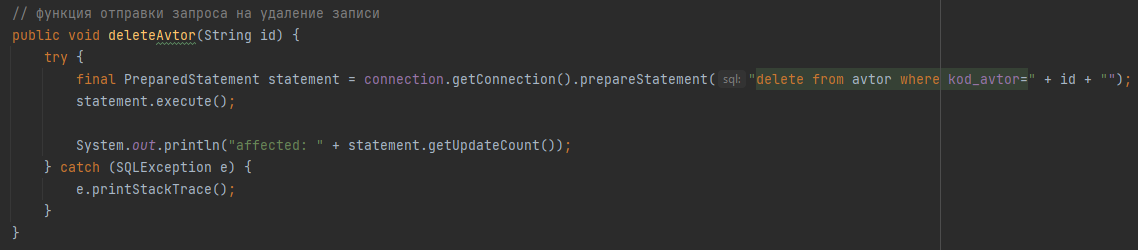
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

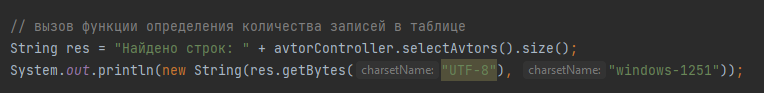


Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, электроника, экран, черный

Автоматически созданное описание

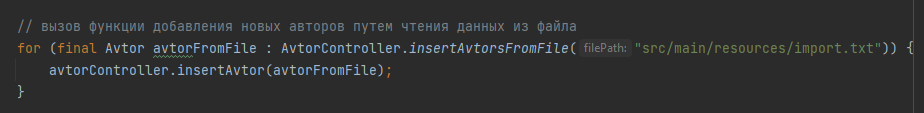


Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, электроника, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, в помещении, черный, снимок экрана

Автоматически созданное описание

ПРИЛОЖЕНИЕ

1. Код класса DBConnection

package main.service;  
import java.sql.\*;  
  
  
public class DBConnection {  
 private Statement statement;  
 private Connection connection;  
 ResultSet rSet = null;  
  
 // создание подключения  
 public DBConnection() {  
 System.*out*.println("Connecting to database ...\n");  
 final String url = "jdbc:postgresql://172.20.8.18:5432/db0092\_22";  
 try {  
 Class.*forName*("org.postgresql.Driver");  
  
 connection = DriverManager.*getConnection*(url, "st0092", "qwerty92");  
 statement = connection.createStatement();  
 } catch (SQLException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // функция, возвращающая подключение  
 public Connection getConnection() {  
 return connection;  
 }  
  
 // функция, исполняющая пришедший запрос  
 public void executeQuery(String query) throws SQLException {  
 statement.executeUpdate(query);  
 }  
  
 // функция, возвращающая количество обновленных строк  
 public int getUpdateCount() throws SQLException {  
 return statement.getUpdateCount();  
 }  
  
 // функция, возвращающая набор результатов по запросу  
 public ResultSet getRSet(String query) throws SQLException {  
 statement = connection.createStatement();  
 rSet = statement.executeQuery(query);  
  
 return rSet;  
 }  
}

1. Код класса Avtor

package main.dto;  
  
public class Avtor {  
  
 int id;  
 String fam;  
 String name;  
 Integer god\_r;  
 Character pol;  
  
 public Avtor(String fam, String name, Integer god\_r, Character pol) {  
 this.fam = fam;  
 this.name = name;  
 this.god\_r = god\_r;  
 this.pol = pol;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public String getFam() {  
 return fam;  
 }  
  
 public void setFam(String fam) {  
 this.fam = fam;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Integer getGod\_r() {  
 return god\_r;  
 }  
  
 public void setGod\_r(Integer god\_r) {  
 this.god\_r = god\_r;  
 }  
  
 public Character getPol() {  
 return pol;  
 }  
  
 public void setPol(Character pol) {  
 this.pol = pol;  
 }  
}

1. Код класса AvtorController

package main.dto;  
import main.service.DBConnection;  
import java.io.\*;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.List;  
  
public class AvtorController {  
  
 private final DBConnection connection;  
 public AvtorController(DBConnection connection) {  
 this.connection = connection;  
 }  
  
 // функция отправки запроса на добавление записи  
 public void insertAvtor(Avtor avtor) throws SQLException {  
 connection.executeQuery("insert into avtor (name, fam, god\_r, pol) values ('" + avtor.getName() + "', '" + avtor.getFam() + "', " +  
 avtor.getGod\_r() + ", '" + avtor.getPol() + "')");  
  
 System.*out*.println("affected: " + connection.getUpdateCount());  
 }  
  
 // функция отправки запроса на обновление записи  
 public void updateAvtor(int id, Avtor avtor) throws SQLException {  
 connection.executeQuery("update avtor set fam = '" + avtor.getFam() + "', name = '" + avtor.getName() + "', god\_r = " + avtor.getGod\_r() + ", pol = '"  
 + avtor.getPol() + "' where kod\_avtor = " + id + "");  
  
 System.*out*.println("affected: " + connection.getUpdateCount());  
 }  
  
 // функция отправки запроса на удаление записи  
 public void deleteAvtor(String id) {  
 try {  
 final PreparedStatement statement = connection.getConnection().prepareStatement("delete from avtor where kod\_avtor=" + id + "");  
 statement.execute();  
  
 System.*out*.println("affected: " + statement.getUpdateCount());  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // функция возврата списка всех записей таблицы  
 public List<Avtor> selectAvtors() {  
 final List<Avtor> list = new ArrayList<>();  
  
 try {  
 final ResultSet set = connection.getRSet("select \* from avtor");  
 while (set.next()) {  
 final String fam = set.getString("fam");  
 final String name = set.getString("name");  
 final int god\_r = set.getInt("god\_r");  
 final Character pol = set.getString("pol").charAt(0);  
  
 list.add(new Avtor(fam, name, god\_r, pol));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 return list;  
 }  
  
 // функция вывода всей информации о авторе по id  
 public void absoluteSelect(int id) {  
 try {  
 final ResultSet set = connection.getRSet("select \* from avtor where kod\_avtor=" + id);  
 set.next();  
  
 final String name = set.getString("name");  
 final String fam = set.getString("fam");  
 final int god\_r = set.getInt("god\_r");  
 final Character pol = set.getString("pol").charAt(0);  
  
 String res = name + " " + fam + " " + god\_r + " " + pol;  
 System.*out*.println(new String(res.getBytes("UTF-8"), "windows-1251"));  
  
 } catch (SQLException | UnsupportedEncodingException e) {  
 e.printStackTrace();  
 }  
 }  
  
 // функция возвращающая данные об авторе путем чтения файла  
 public static List<Avtor> insertAvtorsFromFile(String filePath) {  
 final List<Avtor> list = new ArrayList<>();  
  
 try {  
 final BufferedReader reader = new BufferedReader(  
 new InputStreamReader(  
 new FileInputStream(filePath), "UTF8"));  
 String line = reader.readLine();  
 while (line != null) {  
 final String[] args = line.split(",");  
 final String name = args[0];  
 final String fam = args[1];  
 final int god\_r = Integer.*parseInt*(args[2]);  
 final Character pol = args[3].charAt(0);  
  
 line = reader.readLine();  
 list.add(new Avtor(name, fam, god\_r, pol));  
 }  
 } catch (NumberFormatException | IOException e) {  
 e.printStackTrace();  
 }  
  
 return list;  
 }  
}

1. Код класса Main

package main.service;  
import main.dto.Avtor;  
import main.dto.AvtorController;  
import java.io.UnsupportedEncodingException;  
import java.sql.SQLException;  
  
public class Main {  
  
 public static void main(String[] args) throws SQLException, UnsupportedEncodingException {  
 // создание подключения к БД  
 DBConnection dbConn = new DBConnection();  
 // создание нового автора  
 Avtor avtor = new Avtor("Иван", "Иванов", 1879, 'м');  
  
 // добавление подключения к классу, содержащему управляющие функции  
 AvtorController avtorController = new AvtorController(dbConn);  
  
 // вызов функции добавления нового автора  
 avtorController.insertAvtor(avtor);  
 // вызов функции обновления автора по id  
 avtorController.updateAvtor(33, new Avtor("Романова", "Елизавета", 1900, 'ж'));  
 // вызов функции вывода всей информации об авторе по id  
 avtorController.absoluteSelect(33);  
 // вызов функции удаления автора по id  
 avtorController.deleteAvtor("35");  
  
  
 // вызов функции добавления новых авторов путем чтения данных из файла  
 for (final Avtor avtorFromFile : AvtorController.*insertAvtorsFromFile*("src/main/resources/import.txt")) {  
 avtorController.insertAvtor(avtorFromFile);  
 }  
  
 // вызов функции определения количества записей в таблице  
 String res = "Найдено строк: " + avtorController.selectAvtors().size();  
 System.*out*.println(new String(res.getBytes("UTF-8"), "windows-1251"));  
 }  
}