Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Лабораторная работа №7 по дисциплине  
«Технология программирования» на тему

*Организация в Java хранения данных в базе данных*

Факультет: АВТФ Преподаватель:

Группа: АВТ-810 Михайленко Д. А.

Студенты: Степаненко Владислав

Новосибирск

2020

#### Практические задания

**ЦЕЛЬ И ЗАДАНИЕ**

# Цель работы:

1. Изучить особенности реализации хранения данных в базе данных в приложениях Java.

2. Доработать программу, созданную в лабораторной работе № 5:

# Задание

- Добавить в меню пункты сохранение и загрузку объектов в базу данных. Можно использовать любую СУБД.

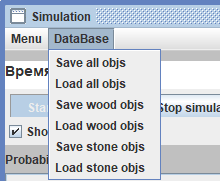
- Загрузку и сохранение объектов только заданного типа.

БД – postgresql.

Был создан класс HouseBD для управления базой данных

Разные методы класса загружают и сохраняют определенные типы объектов в базы данных.

Для выбора команд для работы с базой данных были добавлены пункты меню.



Код БД:

package singleton;  
import objects.\*;  
import java.sql.\*;  
import java.util.Vector;  
  
public class HouseDB  
{  
 public String url = "jdbc:postgresql://localhost:5432/HouseDB";  
 public static String *login* = "postgres";  
 public static String *password* = "123321Qs";  
 public static String *DB\_Driver* = "org.postgresql.Driver";  
 public static int *count* = 0;  
 public HouseDB()  
 {  
 try  
 {  
 Class.*forName*("org.postgresql.Driver");  
 System.*out*.println("Connection to DB");  
 Connection connection = DriverManager.*getConnection*(url,*login*,*password*);  
 System.*out*.println("Connected to DB");  
 connection.close();  
 } catch (ClassNotFoundException | SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 public void insertAll(Vector<House> obj) {  
 delete("House");  
 try {  
 Connection connection = DriverManager.*getConnection*(url, *login*, *password*);  
 PreparedStatement ps = connection.prepareStatement("INSERT INTO House (ID,TYPE,X,Y,LIFEPERIOD,APPTIME,VX,VY,TACT,LASTTACT) VALUES (?,?,?,?,?,?,?,?,?,?)");  
  
 for (int i = 0; i < obj.size(); i++) {  
 ps.setInt(1, obj.get(i).id);  
 if ((obj.get(i) instanceof Wood)) {  
 ps.setString(2, "Wood");  
 } else {  
 ps.setString(2, "Stone");  
 }  
 ps.setDouble(3, obj.get(i).getX());  
 ps.setDouble(4, obj.get(i).getY());  
 ps.setInt(5, obj.get(i).lifePeriod);  
 ps.setInt(6, obj.get(i).appTime);  
 ps.setDouble(7, obj.get(i).vx);  
 ps.setDouble(8, obj.get(i).vy);  
 ps.setInt(9, obj.get(i).tact);  
 ps.setInt(10, obj.get(i).lasttact);  
 ps.addBatch();  
 }  
  
 ps.executeBatch();  
 ps.close();  
 connection.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 public void insertWood(Vector<House> woodVector) {  
 delete("Wood");  
 try {  
 Connection connection = DriverManager.getConnection(url, login, password);  
 PreparedStatement ps = connection.prepareStatement("INSERT INTO Wood (ID,X,Y,LIFEPERIOD,APPTIME,VX,VY,TACT,LASTTACT) VALUES (?,?,?,?,?,?,?,?,?)");  
  
 for (int i = 0; i < woodVector.size(); i++) {  
 ps.setInt(1, woodVector.get(i).id);  
 ps.setDouble(2, woodVector.get(i).getX());  
 ps.setDouble(3, woodVector.get(i).getY());  
 ps.setInt(4, woodVector.get(i).lifePeriod);  
 ps.setInt(5, woodVector.get(i).appTime);  
 ps.setDouble(6, woodVector.get(i).vx);  
 ps.setDouble(7, woodVector.get(i).vy);  
 ps.setInt(8, woodVector.get(i).tact);  
 ps.setInt(9, woodVector.get(i).lasttact);  
 ps.addBatch();  
 }  
 ps.executeBatch();  
 ps.close();  
 connection.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 public void insertStone(Vector<House> stoneVector) {  
 delete("Stone");  
 try {  
 Connection connection = DriverManager.getConnection(url, login, password);  
 PreparedStatement ps = connection.prepareStatement("INSERT INTO Stone (ID,X,Y,LIFEPERIOD,APPTIME,VX,VY,TACT,LASTTACT) VALUES (?,?,?,?,?,?,?,?,?)");  
  
 for (int i = 0; i < stoneVector.size(); i++) {  
 ps.setInt(1, stoneVector.get(i).id);  
 ps.setDouble(2, stoneVector.get(i).getX());  
 ps.setDouble(3, stoneVector.get(i).getY());  
 ps.setInt(4, stoneVector.get(i).lifePeriod);  
 ps.setInt(5, stoneVector.get(i).appTime);  
 ps.setDouble(6, stoneVector.get(i).vx);  
 ps.setDouble(7, stoneVector.get(i).vy);  
 ps.setInt(8, stoneVector.get(i).tact);  
 ps.setInt(9, stoneVector.get(i).lasttact);  
 ps.addBatch();  
 }  
  
 ps.executeBatch();  
 ps.close();  
 connection.close();  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 public Vector<House> loadAll() {  
 Vector<House> vector = new Vector<>();  
 try {  
 Connection connection = DriverManager.getConnection(url, login, password);  
 Statement st = connection.createStatement();  
 ResultSet rs = st.executeQuery("SELECT \* FROM House");  
 while (rs.next()) {  
 if (rs.getString("TYPE").equals("Wood"))  
 {  
 Wood newWood = new Wood();  
 newWood.id = rs.getInt("ID");  
 newWood.x = rs.getInt("X");  
 newWood.y = rs.getInt("Y");  
 newWood.lifePeriod =rs.getInt("LIFEPERIOD");  
 newWood.appTime = rs.getInt("APPTIME");  
 newWood.vx =rs.getInt("VX");  
 newWood.vy = rs.getInt("VY");  
 newWood.tact = rs.getInt("TACT");  
 newWood.lasttact = rs.getInt("LASTTACT");  
 vector.add(newWood);  
 }  
 else {  
 Stone newStone = new Stone();  
 newStone.id = rs.getInt("ID");  
 newStone.x = rs.getInt("X");  
 newStone.y = rs.getInt("Y");  
 newStone.lifePeriod =rs.getInt("LIFEPERIOD");  
 newStone.appTime = rs.getInt("APPTIME");  
 newStone.vx =rs.getInt("VX");  
 newStone.vy = rs.getInt("VY");  
 newStone.tact = rs.getInt("TACT");  
 newStone.lasttact = rs.getInt("LASTTACT");  
 vector.add(newStone);  
 }  
 }  
  
 } catch (  
 SQLException e) {  
 e.printStackTrace();  
 }  
 return vector;  
 }  
  
 public Vector<House> loadWood() {  
 Vector<House> vector = new Vector<>();  
 try {  
 Connection connection = DriverManager.getConnection(url, login, password);  
 Statement st = connection.createStatement();  
 ResultSet rs = st.executeQuery("SELECT \* FROM Wood");  
 while (rs.next())  
 {  
 Wood newWood = new Wood();  
 newWood.id = rs.getInt("ID");  
 newWood.x = rs.getInt("X");  
 newWood.y = rs.getInt("Y");  
 newWood.lifePeriod =rs.getInt("LIFEPERIOD");  
 newWood.appTime = rs.getInt("APPTIME");  
 newWood.vx =rs.getInt("VX");  
 newWood.vy = rs.getInt("VY");  
 newWood.tact = rs.getInt("TACT");  
 newWood.lasttact = rs.getInt("LASTTACT");  
 vector.add(newWood);  
 }  
  
 } catch (  
 SQLException e) {  
 e.printStackTrace();  
 }  
 return vector;  
 }  
  
 public Vector<House> loadStone() {  
 Vector<House> vector = new Vector<>();  
 try {  
 Connection connection = DriverManager.getConnection(url, login, password);  
 Statement st = connection.createStatement();  
 ResultSet rs = st.executeQuery("SELECT \* FROM Stone");  
 while (rs.next())  
 {  
 Stone newStone = new Stone();  
 newStone.id = rs.getInt("ID");  
 newStone.x = rs.getInt("X");  
 newStone.y = rs.getInt("Y");  
 newStone.lifePeriod =rs.getInt("LIFEPERIOD");  
 newStone.appTime = rs.getInt("APPTIME");  
 newStone.vx =rs.getInt("VX");  
 newStone.vy = rs.getInt("VY");  
 newStone.tact = rs.getInt("TACT");  
 newStone.lasttact = rs.getInt("LASTTACT");  
 vector.add(newStone);  
 }  
  
 } catch (  
 SQLException e) {  
 e.printStackTrace();  
 }  
 return vector;  
 }  
  
 public void delete(String str) {  
 try {  
 Connection con = DriverManager.getConnection(url, login, password);  
 PreparedStatement stmt = con.prepareStatement(  
 "DELETE FROM " + str);  
 stmt.execute();  
 stmt.close();  
 con.close();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
}