МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОСИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**Лабораторная работа №7**

**По дисциплине «Технология программирования»**

**на тему**

**Организация в Java хранения данных в базе данных**

Группа: АВТ-808

Студент: Попов Д.А.

Вариант: 7

Новосибирск, 2020

***Задание:***

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

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

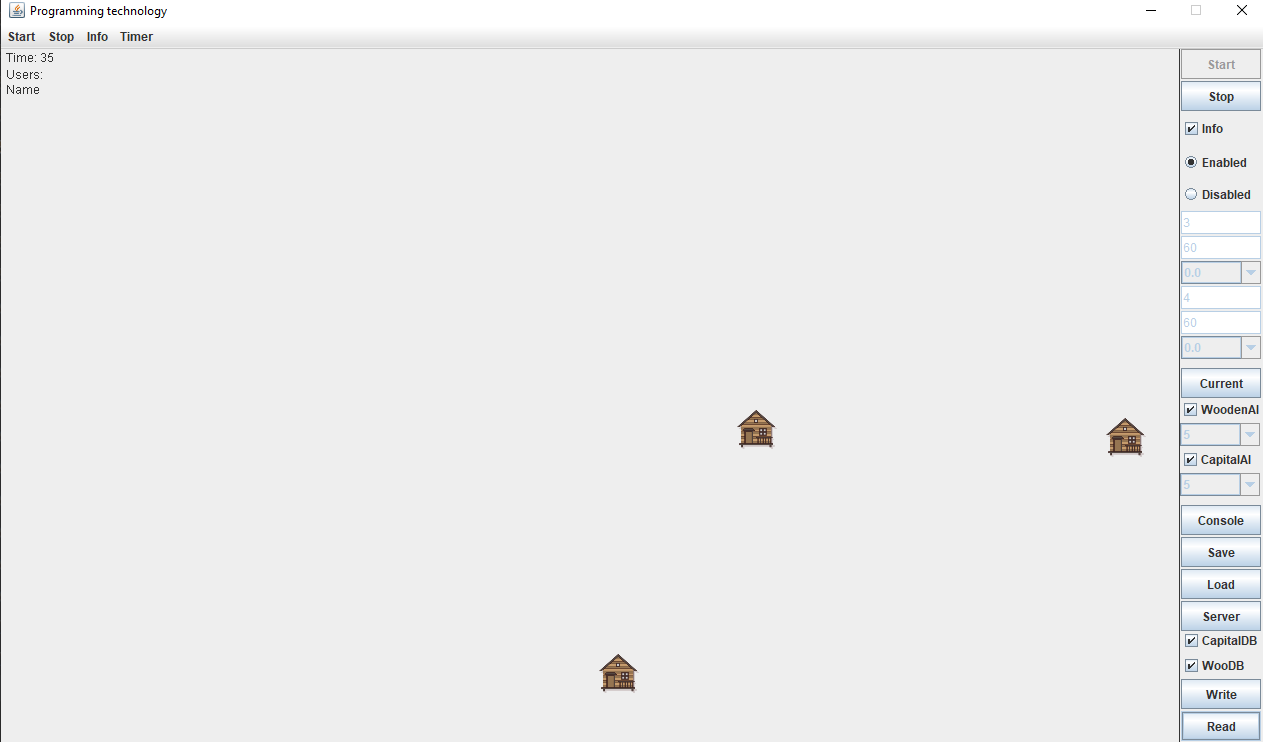
- Добавить в меню пункты сохранение и загрузку объектов в базу данных. Можно использовать любую СУБД.

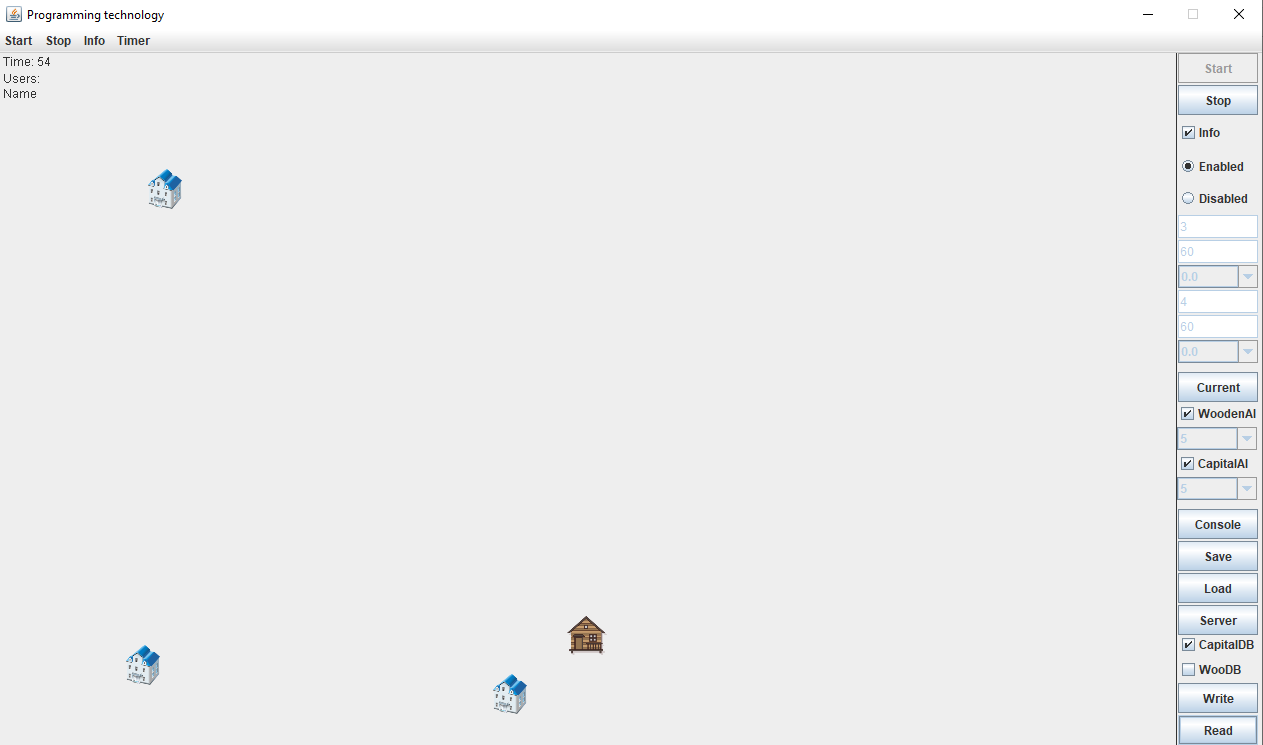
- Загрузку и сохранение объектов только заданного типа.

***Описание доработки программы:***

Создан класс DB, в котором реализованы методы для подключения к БД, создания БД, а также чтения из БД и записи в БД, и метод закрывающий соединения.

package term4;  
  
import java.sql.\*;  
  
public class DB {  
 private static Connection *connection*;  
 private static Statement *statement*;  
 private static ResultSet *resultSet*;  
  
 private static final int *CAPITAL* = 1;  
 private static final int *WOODEN* = 2;  
  
 public static void connect() throws ClassNotFoundException, SQLException {  
 final String URL = "jdbc:mysql://localhost:3306/houses";  
 Class.*forName*("com.mysql.jdbc.Driver");  
 *connection* = DriverManager.*getConnection*(URL, "root", "1337");  
 }  
  
 public static void create() throws SQLException {  
 *statement* = *connection*.createStatement();  
 *statement*.execute("CREATE TABLE IF NOT EXISTS houses (" +  
 "id DOUBLE, " +  
 "type INT, " +  
 "x INT, " +  
 "y INT, " +  
 "speed INT);");  
  
 }  
  
 public static void write() throws SQLException {  
 *statement*.execute("DELETE FROM houses");  
 int type = -1;  
 for (int i = 0; i < Singleton.*getSingleton*().getHouseVector().size(); i++) {  
 if(Singleton.*getSingleton*().getHouse(i).getTypeOfHouse() == TypeOfHouse.*CAPITAL*){  
 if(Main.*habitat*.getMyComponent().getjCheckBoxCapitalDB().isSelected()) {  
 type = *CAPITAL*;  
 *statement*.execute("INSERT INTO houses " +  
 "(id, type, x, y, speed) VALUES" +  
 " ('" + Singleton.*getSingleton*().getHouse(i).getId() + "', '" + type + "', '" + Singleton.*getSingleton*().getHouse(i).getX() + "','" +  
 Singleton.*getSingleton*().getHouse(i).getY() + "','" + Singleton.*getSingleton*().getHouse(i).getSpeed() + "');");  
  
 }  
 }  
 else {  
 if(Main.*habitat*.getMyComponent().getjCheckBoxWoodenDB().isSelected()) {  
 type = *WOODEN*;  
 *statement*.execute("INSERT INTO houses " +  
 "(id, type, x, y, speed) VALUES" +  
 " ('" + Singleton.*getSingleton*().getHouse(i).getId() + "', '" + type + "', '" + Singleton.*getSingleton*().getHouse(i).getX() + "','" +  
 Singleton.*getSingleton*().getHouse(i).getY() + "','" + Singleton.*getSingleton*().getHouse(i).getSpeed() + "');");  
  
 }  
 }  
 }  
 }  
  
 public static void read() throws SQLException {  
 Singleton.*getSingleton*().clearArea();  
 MyComponent.*setCount*(0);  
 if(Main.*habitat*.getMyComponent().getjCheckBoxWoodenDB().isSelected() &&  
 Main.*habitat*.getMyComponent().getjCheckBoxCapitalDB().isSelected()) {  
 *resultSet* = *statement*.executeQuery("SELECT \* FROM houses");  
 }  
 else if(Main.*habitat*.getMyComponent().getjCheckBoxCapitalDB().isSelected()){  
 *resultSet* = *statement*.executeQuery("SELECT \* FROM houses WHERE type = '" + *CAPITAL* + "'");  
 }  
 else if(Main.*habitat*.getMyComponent().getjCheckBoxWoodenDB().isSelected()){  
 *resultSet* = *statement*.executeQuery("SELECT \* FROM houses WHERE type = '" + *WOODEN* + "'");  
 }  
 else if(!Main.*habitat*.getMyComponent().getjCheckBoxWoodenDB().isSelected() &&  
 !Main.*habitat*.getMyComponent().getjCheckBoxCapitalDB().isSelected()) {  
 return;  
 }  
  
 while (*resultSet*.next()){  
 if (*resultSet*.getInt("type") == *CAPITAL*) {  
  
 AbstractFactory abstractFactory = ConcreteFactory.*concreteFactory*(TypeOfHouse.*CAPITAL*);  
 House house = abstractFactory.createHouse();  
 house.setTimeOfBirth(MyComponent.*getTime*());  
 Singleton.*getSingleton*().getHouseVector().add(house);  
 house.setTimeOfBirth(MyComponent.*getTime*());  
 house.setPosition(*resultSet*.getInt("x"), *resultSet*.getInt("y"));  
 house.setSpeed(*resultSet*.getInt("speed"));  
 MyComponent.*setCount*(Singleton.*getSingleton*().getHouseVector().size());  
 }  
 else if(*resultSet*.getInt("type") == *WOODEN*){  
 AbstractFactory abstractFactory = ConcreteFactory.*concreteFactory*(TypeOfHouse.*WOODEN*);  
 House house = abstractFactory.createHouse();  
 house.setTimeOfBirth(MyComponent.*getTime*());  
 Singleton.*getSingleton*().getHouseVector().add(house);  
 house.setTimeOfBirth(MyComponent.*getTime*());  
 house.setPosition(*resultSet*.getInt("x"), *resultSet*.getInt("y"));  
 house.setSpeed(*resultSet*.getInt("speed"));  
 MyComponent.*setCount*(Singleton.*getSingleton*().getHouseVector().size());  
 }  
 }  
 }  
  
 public static void close() throws SQLException {  
 *connection*.close();  
 *statement*.close();  
 if(*resultSet* != null){  
 *resultSet*.close();  
 }  
 }  
}





*Рис. 1-2 – Скриншоты работы программы*

**Вывод:**

В ходе лабораторной работы были изучены особенности реализации хранения данных в базе данных в приложениях Java.

**Программный код:**

https://github.com/AVT-808/AVT-808/tree/Popov-Dmitry-7