МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОСИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**Лабораторная работа №4**

**По дисциплине «Технология программирования»**

**на тему**

**Потоки ввода-вывода.**

**Сериализация объектов в файл**

Группа: АВТ-808

Студент: Попов Д.А.

Вариант: 7

Новосибирск, 2020

***Задание:***

1. Изучить особенности реализации системы ввода-вывода в Java.

2. Доработать программу, созданную в лабораторной работе № 4:

1) добавить в главное меню команду «Консоль». По этой команде должно появляться немодальное диалоговое окно с многострочным текстовым полем, занимающим всю область окна. В это окно можно вводить команды по варианту. В это же окно выводится реакция программы на команду;

2) для передачи команды в обрабатывающий поток использовать каналы ввода-вывода;

3) создать конфигурационный файл для программы. В конфигурационный файл должны сохраняться все настройки симуляции, т.е. все данные и состояния, которые задаются в панели управления программы. Конфигурационный файл должен читаться при запуске программы и записываться при выходе. Формат файла текстовый;

4) добавить в главное меню пункты «Загрузить» и «Сохранить». Команда «Сохранить» вызывает сериализацию всех «живых» объектов в ней. Команда «Загрузить» останавливает текущую симуляцию (если симуляция запущена) и загружает объекты из выбранного файла. Не забудьте скорректировать время рождения объектов. После открытия симуляцию можно запустить, загруженные объекты должны вести себя естественно;

5) использовать стандартные файловые диалоги.

Вариант 7

Реализовать в консоли команды «Установить вероятность генерации капитальных домов» и «Получить вероятность генерации капитальных домов». Как параметр в команду установки должно передаваться значение вероятности. Полученная вероятность должна выводиться на консоль.

***Описание доработки программы:***

Создан класс Console наследник класса JDIalog. Метод showConsole() открывает консоль. Реализованы команды:

/help – справка по остальным командам

/set Capital Probability <probability> - установить вероятность генерации капитальных домов

/get Capital Probability – получить вероятность генерации капитальных домов

package term4;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
  
public class Console extends JDialog {  
  
 JTextArea jTextArea = new JTextArea();  
 JTextField jTextAreaEnter = new JTextField();  
  
 JScrollPane scrollPane = new JScrollPane(jTextArea);  
  
  
 String text = "Use /help for a list of possible options\n";  
  
 public Console(JFrame owner) {  
 super(owner,"Console", true);  
 setBackground(Color.*BLACK*);  
 setBounds(650, 300, 600, 400);  
 setResizable(false);  
 jTextArea.setEditable(false);  
  
 jTextArea.setBackground(Color.*BLACK*);  
 jTextArea.setForeground(Color.*WHITE*);  
 jTextArea.setFont(new Font("Times New Romain",Font.*PLAIN*,16));  
 jTextArea.setLineWrap(true);  
 jTextArea.setText(text);  
  
 jTextAreaEnter.setBackground(Color.*BLACK*);  
 jTextAreaEnter.setForeground(Color.*WHITE*);  
 jTextAreaEnter.setFont(new Font("Times New Romain",Font.*PLAIN*,16));  
  
 setLayout(new FlowLayout(FlowLayout.*LEFT*));  
 scrollPane.setPreferredSize(new Dimension(575,320));  
 add(scrollPane);  
 jTextAreaEnter.setPreferredSize(new Dimension(575,25));  
 add(jTextAreaEnter);  
 }  
  
 public void showConsole(){  
  
 jTextAreaEnter.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
 if(e.getKeyCode() == KeyEvent.*VK\_ENTER*){  
 text += jTextAreaEnter.getText() + "\n";  
 jTextArea.setText(text);  
  
 switch (jTextAreaEnter.getText()){  
 case "/help" -> {  
 text += """  
 /set Capital Probability <probability> - Установить вероятность генерации  
 капитальных домов  
 /get Capital Probability - Получить вероятность генерации капитальных домов  
 """;  
 jTextAreaEnter.setText("");  
 }  
 case "/get Capital Probability" ->{  
 text += CapitalHouse.*getProbability*() + "\n";  
 jTextAreaEnter.setText(null);  
 }  
 default -> {  
 if(isSet()){  
 text += "\n";  
 System.*out*.println("Set");  
 }  
 else {  
 text += """  
  
 Unknown command. Please use /help for a list of possible options  
 """;  
 }  
 jTextAreaEnter.setText(null);  
 }  
 }  
 jTextArea.setText(text);  
 }  
 }  
 });  
  
  
 setVisible(true);  
 }  
  
 private boolean isSet(){  
 String[] strings = jTextAreaEnter.getText().split(" ", 4);  
  
 if (strings.length == 4 && strings[0].equalsIgnoreCase("/set") && strings[1].equalsIgnoreCase("Capital")  
 && strings[2].equalsIgnoreCase("Probability")) {  
  
 try {  
 if (Double.*parseDouble*(strings[3]) >= 0 && Double.*parseDouble*(strings[3]) <= 1){  
 CapitalHouse.*setProbability*(Double.*parseDouble*(strings[3]));  
 }  
 else {  
 text += "Error! The probability should be in the range from 0 to 1.\n";  
 }  
 return true;  
 } catch (RuntimeException re) {  
 return false;  
 }  
 }  
 return false;  
 }  
}

Создан класс Config для чтения и записи конфигурационного файла. Чтение производится при открытии программы, запись при закрытии. В случае, если не найден файл config.txt пользователю предлагается выбрать его с помощью диалога выбора файла.

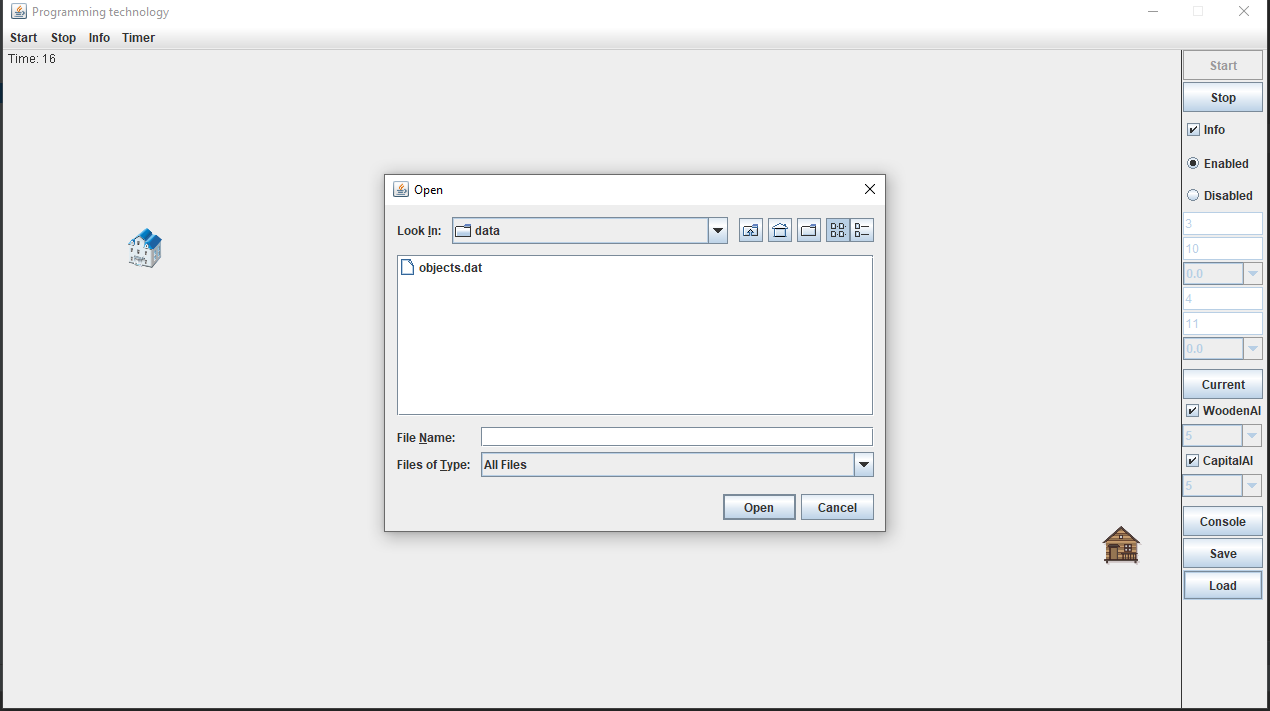
package term4;  
  
import javax.swing.\*;  
import java.io.\*;  
import java.util.LinkedHashMap;  
import java.util.Map;  
  
public class Config {  
 File file = new File("config.txt");  
 FileReader fileReader;  
 FileWriter fileWriter;  
 JFileChooser jFileChooser = new JFileChooser();  
  
 Map<String, String> config = new LinkedHashMap<>();  
  
 public Config() {  
 try{  
 fileReader = new FileReader(file);  
 }  
 catch (FileNotFoundException e){  
 int res = jFileChooser.showDialog(null, "Choose a configuration file");  
 if(res == JFileChooser.*APPROVE\_OPTION*){  
 file = jFileChooser.getSelectedFile();  
 try {  
 fileReader = new FileReader(file);  
 }  
 catch (FileNotFoundException fe){  
 fe.printStackTrace();  
 }  
 }  
 }  
 }  
  
 public void loadConfig(){  
 char[] cfg = new char[512];  
 String[] s1, s2;  
  
 try {  
 fileReader.read(cfg);  
 s1 = new String(cfg).split("\n");  
  
 for (String s : s1) {  
 s2 = s.split(": ");  
 config.put(s2[0], s2[1]);  
 }  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
  
 try {  
 fileReader.close();  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
  
 init();  
 config.forEach((k, v) -> System.*out*.println(k + ": " + v));  
 }  
  
 public void saveConfig(){  
 try {  
 fileWriter = new FileWriter(file);  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
  
 try{  
 fileWriter.write("Information: " + Main.*habitat*.getMyComponent().getjCheckBox().isSelected() +  
 "\nTimer: " + Main.*habitat*.getMyComponent().getjRadioButtonEnabled().isSelected() +  
 "\nWooden House Period: " + WoodenHouse.*getPeriod*() +  
 "\nWooden House Lifetime: " + (WoodenHouse.*getLifetime*() + 1) +  
 "\nWooden House Probability: " + WoodenHouse.*getProbability*() +  
 "\nCapital House Period: " + CapitalHouse.*getPeriod*() +  
 "\nCapital House Lifetime: " + (CapitalHouse.*getLifetime*() + 1) +  
 "\nCapital House Probability: " + CapitalHouse.*getProbability*() +  
 "\nWooden House AI: " + Main.*habitat*.getMyComponent().getjCheckBoxWoodenHouseAI().isSelected() +  
 "\nWooden House AI threads priority: " + Main.*habitat*.getMyComponent().getjComboBoxWoodenPriority()  
 .getSelectedItem() +  
 "\nCapital House AI: " + Main.*habitat*.getMyComponent().getjCheckBoxCapitalHouseAI().isSelected() +  
 "\nCapital House AI threads priority: " + Main.*habitat*.getMyComponent().getjComboBoxCapitalPriority()  
 .getSelectedItem() +  
 "\nEOF: true");  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
  
 try{  
 fileWriter.close();  
 }  
 catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
  
 private void init(){  
 Main.*habitat*.getMyComponent().getjCheckBox().setSelected(Boolean.*parseBoolean*(config.get("Information")  
 .replaceAll("\\s+", "")));  
  
 Main.*setIsTimeVisible*(Boolean.*parseBoolean*(config.get("Timer").replaceAll("\\s+", "")));  
 if(Boolean.*parseBoolean*(config.get("Timer").replaceAll("\\s+", ""))){  
 Main.*habitat*.getMyComponent().getjRadioButtonEnabled().setSelected(true);  
 }  
 else {  
 Main.*habitat*.getMyComponent().getjRadioButtonDisabled().setSelected(true);  
 }  
  
 Main.*habitat*.getMyComponent().getjTextFieldWoodenPeriod().setText(config.get("Wooden House Period")  
 .replaceAll("\\s+", ""));  
 Main.*habitat*.getMyComponent().getjTextFieldWoodenLifetime().setText(config.get("Wooden House Lifetime")  
 .replaceAll("\\s+", ""));  
  
 if(Double.*parseDouble*(config.get("Wooden House Probability").replaceAll("\\s+", "")) >= 0 &&  
 Double.*parseDouble*(config.get("Wooden House Probability").replaceAll("\\s+", "")) <= 1) {  
 WoodenHouse.*setProbability*(Double.*parseDouble*(config.get("Wooden House Probability")  
 .replaceAll("\\s+", "")));  
 }  
  
 Main.*habitat*.getMyComponent().getjTextFieldCapitalPeriod().setText(config.get("Capital House Period")  
 .replaceAll("\\s+", ""));  
 Main.*habitat*.getMyComponent().getjTextFieldCapitalLifetime().setText(config.get("Capital House Lifetime")  
 .replaceAll("\\s+", ""));  
  
 if(Double.*parseDouble*(config.get("Capital House Probability").replaceAll("\\s+", "")) >= 0 &&  
 Double.*parseDouble*(config.get("Capital House Probability").replaceAll("\\s+", "")) <= 1)  
 CapitalHouse.*setProbability*(Double.*parseDouble*(config.get("Capital House Probability")  
 .replaceAll("\\s+", "")));  
  
 Main.*habitat*.getMyComponent().getjCheckBoxWoodenHouseAI().setSelected(Boolean.*parseBoolean*(  
 config.get("Wooden House AI").replaceAll("\\s+", "")));  
 Main.*habitat*.getMyComponent().getjComboBoxWoodenPriority().setSelectedIndex(Integer.*parseInt*(  
 config.get("Wooden House AI threads priority").replaceAll("\\s+", "")) - 1);  
  
 Main.*habitat*.getMyComponent().getjCheckBoxCapitalHouseAI().setSelected(Boolean.*parseBoolean*(  
 config.get("Capital House AI").replaceAll("\\s+", "")));  
 Main.*habitat*.getMyComponent().getjComboBoxCapitalPriority().setSelectedIndex(Integer.*parseInt*(  
 config.get("Capital House AI threads priority").replaceAll("\\s+", "")) - 1);  
  
 System.*out*.println(Boolean.*parseBoolean*(config.get("Wooden House AI").replaceAll("\\s+", "")));  
 System.*out*.println(Main.*habitat*.getMyComponent().getjCheckBoxWoodenHouseAI().isSelected());  
 }  
}

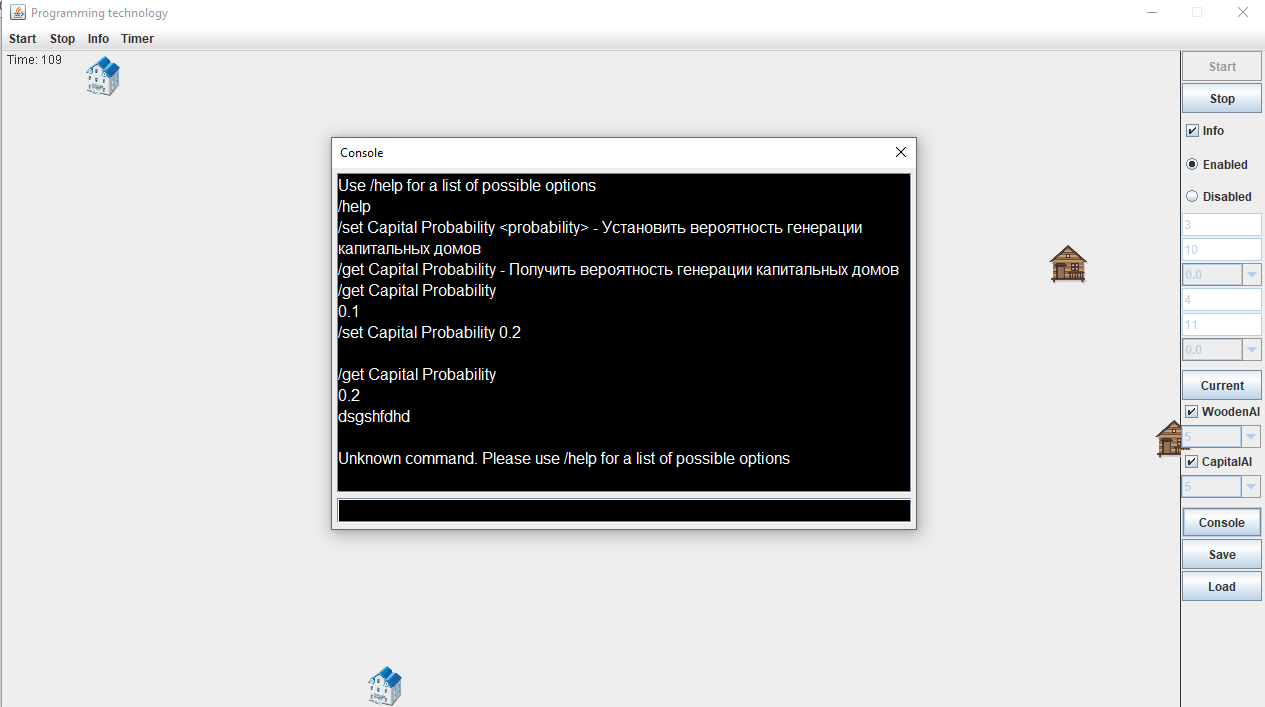
Класс House теперь имплементирует интерфейс Serializable.

Создан класс Serializer для сериализации и десериализации объектов. В случае, если при десериализации по стандартному пути не находится файл objects.dat, пользователю предлагается его выбрать с помощью диалога выбора файла.

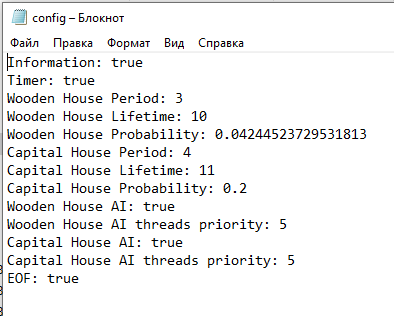
package term4;  
  
import javax.swing.\*;  
import java.io.\*;  
import java.util.Vector;  
  
public class Serializer {  
 File file = new File("data/objects.dat");  
 ObjectOutputStream objectOutputStream;  
 ObjectInputStream objectInputStream;  
 FileOutputStream fileOutputStream;  
 FileInputStream fileInputStream;  
 JFileChooser jFileChooser = new JFileChooser();  
  
 public void serialization(){  
 try {  
 fileOutputStream = new FileOutputStream(file);  
 } catch (FileNotFoundException e) {  
 try {  
 if(file.createNewFile()){  
 System.*out*.println("File created");  
 }  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 try {  
 objectOutputStream = new ObjectOutputStream(fileOutputStream);  
 objectOutputStream.writeObject(Singleton.*getSingleton*().getHouseVector());  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 try {  
 objectOutputStream.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 public void deserialization(){  
 try {  
 fileInputStream = new FileInputStream(file);  
 } catch (FileNotFoundException e) {  
 int res = jFileChooser.showDialog(null, "Open");  
 if(res == JFileChooser.*APPROVE\_OPTION*){  
 file = jFileChooser.getSelectedFile();  
 try {  
 fileInputStream = new FileInputStream(file);  
 } catch (FileNotFoundException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
  
 try {  
 objectInputStream = new ObjectInputStream(fileInputStream);  
  
 Singleton.*getSingleton*().clearArea();  
 Singleton.*getSingleton*().setHouseVector((Vector<House>) objectInputStream.readObject());  
 MyComponent.*setCount*(Singleton.*getSingleton*().getHouseVector().size());  
 for(int i = 0; i < Singleton.*getSingleton*().getHouseVector().size(); i++){  
 Singleton.*getSingleton*().getHouse(i).setTimeOfBirth(MyComponent.*getTime*());  
 }  
 } catch (IOException | ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 try {  
 objectInputStream.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}







*Рис.1-3 – Скриншоты работы программы*



*Рис.4 – Файл config.txt*

**Вывод:**

В ходе лабораторной работы были изучены особенности реализации системы ввода-вывода в Java.

**Программный код:**

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