МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**Лабораторная работа №5**

по дисциплине «Технология программирования»

на тему: «Потоки ввода-вывода. Сериализация объектов в файл»

|  |  |
| --- | --- |
| Группа: АВТ-808 | Преподаватель: Михайленко Дмитрий Анатольевич |
| Студент: Кузьмина Александра |  |
| Вариант: 1 |  |
|  |  |

Новосибирск, 2020

**Цель работы**

Изучить особенности реализации системы ввода-вывода в Java.

**Задание**

Доработать программу, созданную в лабораторной работу № 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 1**

Реализовать в консоли команды «Старт» и «Стоп» симуляции. Кнопки в интерфейсе должны вести себя так же, как если бы нажимали их, а не исполняли команды (блокироваться по очереди).

**Ход работы**

В ходе решения поставленных задач доработки программы были добавлены новые классы и внесены соответствующие изменения в ранее разработанные:

1. Создан класс диалогового окна **Console**, представляющий собой консоль. Реализованы команды:

/help – выводит список всех доступных команд;

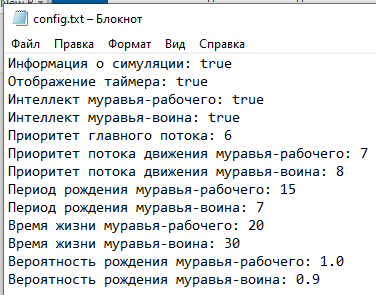
/Старт – запустить симуляцию

/Стоп – остановить симуляцию



2. Добавлен класс **Configuration**, отвечающий за сохранение и загрузку параметров симуляции в и из текстового файла. По умолчанию задан файл “config.txt”, но если он не будет найден, то пользователю будет предложено выбрать другой файл для работы с конфигурацией.





3. Создан класс **Serialization**, реализующий сериализацию и десериализацию по соответствующим кнопкам на панели управления.



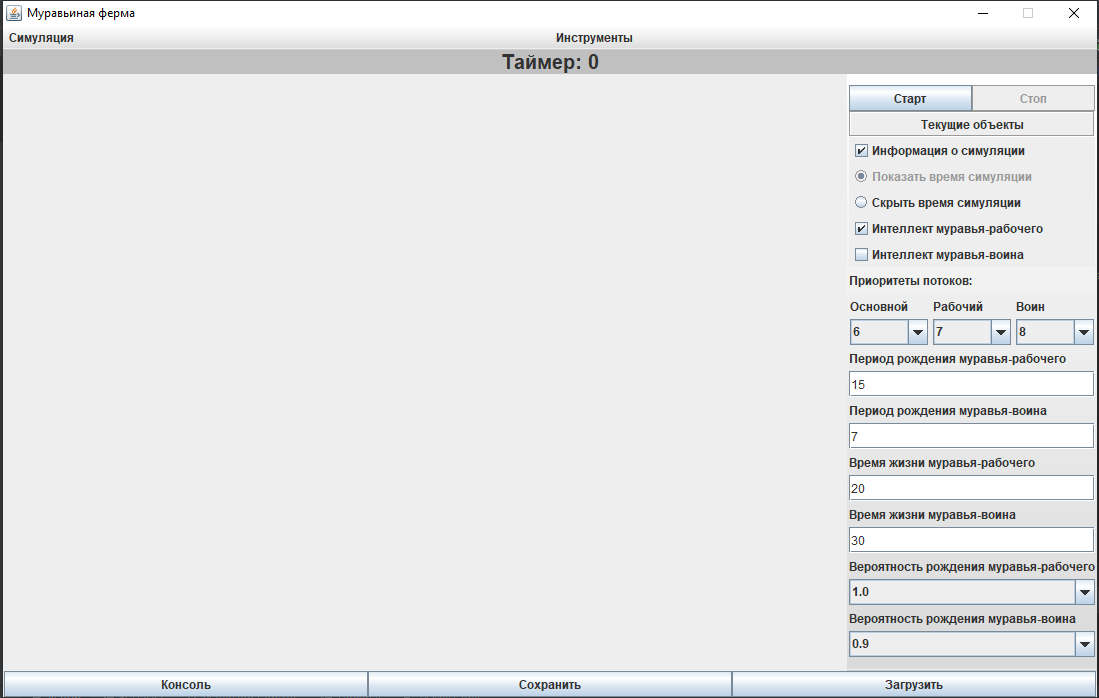
4. На панель управления добавлены кнопки “Консоль” – открывает консоль, “Сохранить” – выполняет сериализацию и “Загрузить” - десериализация.



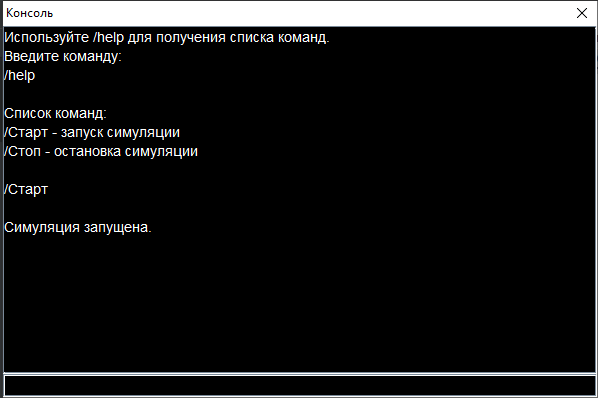
Остальные классы программы были переработаны в соответствии с новыми задачами.

**Скриншоты работы программы**

Окно до запуска симуляции:



Демонстрация работы консоли:



**Вывод**

При выполнении данной работы были изучены особенности реализации системы ввода-вывода в Java. Была реализована консоль, с помощью которой можно контролировать симуляцию, конфигурационный файл с параметрами симуляции и возможность сериализации и десериализации объектов.

**Листинг программы**

**Configuration**

package AntFarm;  
  
import javax.swing.\*;  
import java.io.\*;  
import java.util.LinkedHashMap;  
import java.util.Map;  
  
public class Configuration  
{  
 File file = new File ("C:\\Users\\Турбопупс\\IdeaProjects\\t4.lab\\src\\config.txt");  
 FileReader fileReader;  
 FileWriter fileWriter;  
 JFileChooser jFileChooser = new JFileChooser();  
 Map<String, String> config = new LinkedHashMap<>();  
  
 Configuration()  
 {  
 try  
 {  
 fileReader = new FileReader(file);  
 }  
 catch (FileNotFoundException e)  
 {  
 int chooser = jFileChooser.showDialog(null, "Выбрать файл");  
 if (chooser == JFileChooser.*APPROVE\_OPTION*)  
 {  
 file = jFileChooser.getSelectedFile();  
 try  
 {  
 fileReader = new FileReader(file);  
 }  
 catch (FileNotFoundException fe)  
 {  
 fe.printStackTrace();  
 }  
 }  
 }  
 }  
  
 public void load(Habitat antFarm) {  
 char[] c = new char[512];  
 String[] s1, s2;  
 try  
 {  
 fileReader.read(c);  
 s1 = new String(c).split("\n");  
 for (String s : s1)  
 {  
 s2 = s.split(": ");  
 config.put(s2[0], s2[1]);  
 }  
 fileReader.close();  
 }  
 catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 setConfig(antFarm);  
 config.forEach((key, value) -> System.*out*.println(key + ": " + value));  
 }  
  
 public void save(Habitat antFarm) {  
 try  
 {  
 fileWriter = new FileWriter(file);  
 fileWriter.write("Информация о симуляции: " + antFarm.getInformationVisibility().isSelected() +  
 "\nОтображение таймера: " + antFarm.getTimerIsVisible().isSelected() +  
 "\nИнтеллект муравья-рабочего: " + antFarm.getWorkerAI().isSelected() +  
 "\nИнтеллект муравья-воина: " + antFarm.getWarriorAI().isSelected() +  
 "\nПриоритет главного потока: " + antFarm.getMainThread().getSelectedItem() +  
 "\nПриоритет потока движения муравья-рабочего: " + antFarm.getWorkerThread().getSelectedItem() +  
 "\nПриоритет потока движения муравья-воина: " + antFarm.getWarriorThread().getSelectedItem() +  
 "\nПериод рождения муравья-рабочего: " + antFarm.getWorkerPeriod().getText() +  
 "\nПериод рождения муравья-воина: " + antFarm.getWarriorPeriod().getText() +  
 "\nВремя жизни муравья-рабочего: " + antFarm.getWorkerLifetime().getText() +  
 "\nВремя жизни муравья-воина: " + antFarm.getWarriorLifetime().getText() +  
 "\nВероятность рождения муравья-рабочего: " + antFarm.getWorkerProbability().getSelectedItem() +  
 "\nВероятность рождения муравья-воина: " + antFarm.getWarriorProbability().getSelectedItem());  
 fileWriter.close();  
 }  
 catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
  
 private void setConfig(Habitat antFarm) {  
 antFarm.getInformationVisibility().setSelected(Boolean.*parseBoolean*(config.get("Информация о симуляции")));  
  
 boolean timer = Boolean.*parseBoolean*(config.get("Отображение таймера"));  
 if (timer)  
 {  
 antFarm.getTimerIsVisible().setSelected(true);  
 antFarm.getTimerIsVisible().setEnabled(false);  
 }  
 else {  
 antFarm.getTimerNotVisible().setSelected(true);  
 antFarm.getTimerNotVisible().setEnabled(false);  
 }  
 antFarm.timerVisibility(!timer);  
  
 antFarm.getWorkerAI().setSelected(Boolean.*parseBoolean*(config.get("Интеллект муравья-рабочего")));  
 antFarm.getWarriorAI().setSelected(Boolean.*parseBoolean*(config.get("Интеллект муравья-воина")));  
  
 antFarm.getMainThread().setSelectedItem(Integer.*parseInt*(config.get("Приоритет главного потока")));  
 antFarm.getWorkerThread().setSelectedItem(Integer.*parseInt*(config.get("Приоритет потока движения муравья-рабочего")));  
 antFarm.getWarriorThread().setSelectedItem(Integer.*parseInt*(config.get("Приоритет потока движения муравья-воина")));  
  
 antFarm.getWorkerPeriod().setText(config.get("Период рождения муравья-рабочего"));  
 antFarm.getWarriorPeriod().setText(config.get("Период рождения муравья-воина"));  
  
 antFarm.getWorkerLifetime().setText(config.get("Время жизни муравья-рабочего"));  
 antFarm.getWarriorLifetime().setText(config.get("Время жизни муравья-воина"));  
  
 antFarm.getWorkerProbability().setSelectedItem(Double.*parseDouble*(config.get("Вероятность рождения муравья-рабочего")));  
 antFarm.getWarriorProbability().setSelectedItem(Double.*parseDouble*(config.get("Вероятность рождения муравья-воина")));  
 }  
}

**Console**

package AntFarm;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
  
public class Console extends JDialog  
{  
 JTextArea commands = new JTextArea();  
 JTextField commandLine = new JTextField();  
 JScrollPane scrollPane = new JScrollPane(commands);  
  
 String string = "Используйте /help для получения списка команд.\nВведите команду:\n";  
  
 public Console(JFrame jFrame) {  
 super(jFrame, "Консоль", false);  
 setSize(600,400);  
 setResizable(false);  
 commands.setEditable(false);  
  
 commands.setBackground(Color.*BLACK*);  
 commands.setForeground(Color.*WHITE*);  
 commands.setFont(new Font("Times New Romain", Font.*PLAIN*, 14));  
 commands.setLineWrap(true);  
 commands.setText(string);  
  
 commandLine.setBackground(Color.*BLACK*);  
 commandLine.setForeground(Color.*WHITE*);  
 commandLine.setFont(new Font("Times New Romain", Font.*PLAIN*, 14));  
  
 setLayout(new BorderLayout());  
 add(scrollPane, BorderLayout.*CENTER*);  
 add(commandLine, BorderLayout.*PAGE\_END*);  
 commandLine.setFocusable(true);  
 commandLine.requestFocus();  
 }  
  
 public void ConsoleOperation(Main m){  
 commandLine.addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyPressed(KeyEvent e) {  
 if(e.getKeyCode() == KeyEvent.*VK\_ENTER*) {  
 string += commandLine.getText() + "\n";  
 commands.setText(string);  
 String getTextString = commandLine.getText();  
 if (getTextString.equals("/help"))  
 {  
 string += "\nСписок команд:\n/Старт - запуск симуляции\n/Стоп - остановка симуляции\n\n";  
 }  
 else  
 {  
 if (getTextString.equals("/Старт"))  
 {  
 if (Main.*isStarted*)  
 string += "\nОшибка! Симуляция уже запущена!\n";  
 else  
 {  
 string += "\nСимуляция запущена.\n";  
 m.startSimulation();  
 }  
 }  
 else  
 {  
 if(getTextString.equals("/Стоп"))  
 {  
 if (Main.*isStarted*)  
 {  
 string += "\nСимуляция остановлена\n";  
 m.stopSimulation();  
 }  
 else  
 string+="\nОшибка! Симуляция еще не запущена\n";  
 }  
 else  
 {  
 string += "\nНеизвестная команда. Используйте /help для получения списка команд.\n";  
 }  
 }  
 }  
 commands.setText(string);  
 commandLine.setText(null);  
 commandLine.requestFocus();  
 }  
 }  
 });  
 }  
}

**Serialization**

package AntFarm;  
  
import javax.imageio.ImageIO;  
import javax.swing.\*;  
import java.io.\*;  
import java.util.Vector;  
  
public class Serialization  
{  
 File file = new File("C:\\Users\\Турбопупс\\IdeaProjects\\t4.lab\\src\\objects.ser");  
 ObjectOutputStream objectOutputStream;  
 ObjectInputStream objectInputStream;  
 FileOutputStream fileOutputStream;  
 FileInputStream fileInputStream;  
 JFileChooser jFileChooser = new JFileChooser();  
  
 public void serialize()  
 {  
 try  
 {  
 fileOutputStream = new FileOutputStream(file);  
 }  
 catch (FileNotFoundException e)  
 {  
 try  
 {  
 if(file.createNewFile())  
 System.*out*.println("Файл не найден, поэтому для сохранения объектов был создан новый файл.");  
 }  
 catch (IOException ex)  
 {  
 ex.printStackTrace();  
 }  
 }  
 try  
 {  
 objectOutputStream = new ObjectOutputStream(fileOutputStream);  
 objectOutputStream.writeObject(Singleton.*getAnts*());  
 }  
 catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 try  
 {  
 objectOutputStream.close();  
 }  
 catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
  
 public void deserialize(Habitat antFarm)  
 {  
 try  
 {  
 fileInputStream = new FileInputStream(file);  
 }  
 catch (FileNotFoundException e)  
 {  
 int res = jFileChooser.showDialog(null, "Выбрать файл");  
 if(res == JFileChooser.*APPROVE\_OPTION*){  
 file = jFileChooser.getSelectedFile();  
 try  
 {  
 fileInputStream = new FileInputStream(file);  
 }  
 catch (FileNotFoundException ex)  
 {  
 ex.printStackTrace();  
 }  
 }  
 }  
 try {  
 int workers = 0; int warriors = 0;  
 objectInputStream = new ObjectInputStream(fileInputStream);  
 Singleton.*clearAnts*();  
 Singleton.*setAnts*((Vector<Ant>)objectInputStream.readObject());  
 if (!Singleton.*getAnts*().isEmpty())  
 {  
 for (int i=0; i<Singleton.*getArraySize*();i++)  
 {  
 Singleton.*getAnts*().get(i).birthTime = antFarm.getTime();  
 if (Singleton.*getAnts*().get(i).getClass() == AntWorker.class)  
 {  
 Singleton.*getAnts*().get(i).setImage(ImageIO.*read*(getClass().getResource("/AntWorker.png")));  
 Singleton.*getAnts*().get(i).deathTime = Singleton.*getAnts*().get(i).birthTime +  
 Integer.*parseInt*(antFarm.getWorkerLifetime().getText());  
 workers++;  
 }  
 else  
 {  
 Singleton.*getAnts*().get(i).setImage(ImageIO.*read*(getClass().getResource("/AntWarrior.png")));  
 Singleton.*getAnts*().get(i).deathTime = Singleton.*getAnts*().get(i).birthTime +  
 Integer.*parseInt*(antFarm.getWarriorLifetime().getText());  
 warriors++;  
 }  
 Singleton.*addIds*(Singleton.*getAnts*().get(i).id);  
 Singleton.*addBirthTime*(Singleton.*getAnts*().get(i).id, Singleton.*getAnts*().get(i).birthTime);  
 }  
 antFarm.setAntsAmount(Singleton.*getArraySize*());  
 antFarm.setWorkersAmount(workers);  
 antFarm.setWarriorsAmount(warriors);  
 }  
 }  
 catch (IOException | ClassNotFoundException e)  
 {  
 e.printStackTrace();  
 }  
 try  
 {  
 objectInputStream.close();  
 }  
 catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}