МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники



**Лабораторная Работа №5**

**по дисциплине:** *технология программирования*

**на тему:** *Потоки ввода-вывода.*

*Сериализация объектов в файл*

|  |  |
| --- | --- |
| Выполнили: | Проверил: |
| Студенты гр. *АВТ-710*, *АВТФ* | *ассистент каф. ВТ* |
| *Аконечников Е.Н.* | *Михайленко Д.А.* |
| «\_\_\_» \_\_\_\_\_\_ 20\_\_г. | «\_\_\_» \_\_\_\_\_\_ 20\_\_г. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| *Перминов А.А.* | (подпись) |
| «\_\_\_» \_\_\_\_\_\_ 20\_\_г. |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| (подпись) |  |

Цель

1. Изучить особенности реализации системы ввода-вывода в Java.
2. Доработать программу, созданную в лабораторной работе № 4:

Задание

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 основное окно программы 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 3***

Реализовать в консоли команды «Установить вероятность рождения золотых рыбок» и

«Получить вероятность рождения золотых рыбок». Как параметр в команду установки

должно передаваться значение вероятности. Полученная вероятность должна выводиться

на консоль.

Проектирование программы

Обсуждение основных идей алгоритма

Для программы создается 12 классов:

**Fish** — абстрактный класс объекта, реализующий интерфейс IBehaviour. Содержит:

* Переменные: координаты х и y, количество рыб, Id рыбы, время рождения
* Конструктор по умолчанию (Присваивает координаты рыбки равные (0, 0) и увеличивает количество рыбок на 1)
* Конструктор с параметрами (Приравнивает координаты рыбки переданным координатам, увеличивает количество рыбок на 1)

**Habitat** — класс, в котором содержится LinkedList рыбок, HashSet Id рыбок, TreeMap id и дату рождения рыбок, находит координаты рыбки, чтобы она попадала на экран.

* Update() — получает на вход время, прошедшее от начала симуляции. В данном методе генерируются новые объекты и помещаются в коллекцию объектов.
* addToCollections – добавляет элемент в колекции IdSet, HashMap, LinkedList, осуществляет уникальность Id. Осуществляется проверка нахождения в рабочей области.

**HabitatView** — класс среды, определяющий параметры рабочей области (конструктор HabitatView ()) и описывающий весь пользовательский интерфейс, реализуются функции, происходящие при нажатии клавиш и кнопок на панели интерфейса. Определяет интерфейс KeyListener. Помимо этого, содержит следующие методы:

* Update() — получает на вход время, прошедшее от начала симуляции. В данном методе генерируются новые объекты и помещаются в поле визуализации в случайном месте.
* paint() — выводит изображение в поле визуализации.
* startSim () — запуск симуляции, сбрасываются счетчики объектов, создается элемент класса Habitat.
* stopSim () — завершает симуляцию.
* SimulationLoop — класс, наследуемый от класса TimerTask. Содержит метод run(), в котором определяется текущее время работы программы и вызывается метод update() класса Habitat.
* show\_list\_object() – метод для отображения списка сгенерированных рыб и их ID
* load\_file() – метод для загрузки рыб, их координат и текущего времени из файла
* save\_file - метод для сохранения рыб, их координат и текущего времени в файл
* saveConfig – метод для сохранения текущих настроек программы.
* loadConfig – метод для загрузки настроек программы

**Gold** — класс, наследуемый от Fish и устанавливающий координаты золотой рыбки, количество, время жизни, скорость передвижения

* Конструктор с параметрами (Приравнивает координаты рыбки переданным координатам, увеличивает количество рыбок на 1)
* Функция для создания рыбки с определенным шансом, возвращает true или false в зависимости от результата.
* Метод для движения рыбки по области отображения

**Guppy** — класс, наследуемый от Fish и устанавливающий координаты рыбки гуппи, количество, время жизни.

* Конструктор с параметрами (Приравнивает координаты рыбки переданным координатам, увеличивает количество рыбок на 1)
* Функция для создания рыбки с определенным шансом, возвращает true или false в зависимости от результата.
* Метод для движения рыбки по области отображения

**IBehaviour** — реализуемый интерфейс, определяет основные параметры реализуемых его классов.

**Main** — основной класс работы программы, в котором создается объект класса Habitat и класс processKeyEvent который отвечает за обработку события — нажатие клавиши, также там реализованы все кнопки интерфейса и менюбара.

**BaseAI –** класс, наследуемый от потока Thread, необходим для реализации передвижения, в поточном режиме, потоки передвижения синхронизированы.

**GuppyAI -** класс, наследуемый от BaseAI, содержит метод Step, который проверяет, возможно ли движение рыбки - гуппи

**GoldAI -** класс, наследуемый от BaseAI, содержит метод Step, который проверяет, возможно ли движение золотой рыбки

**Console**  – класс, наследуемый от JDialog, содержит консольное окно, с поддержкой команд для задания значения шанса появления рыбки и его получения( вывод в консоль). Действия совершаются в поточном режиме, с проверкой правильности ввода команд.

**DocumentChangeListener** – абстрактный класс, реализующий интерфейс DocumentListener

Пользовательский интерфейс

Управление пользовательским интерфейсом осуществляется через нажатие на клавиши клавиатуры, а так же при помощи кнопок на панели интерфейса и кнопок менюбара. При первом запуске выводится информация о возможных действиях программы(рисунок 1).

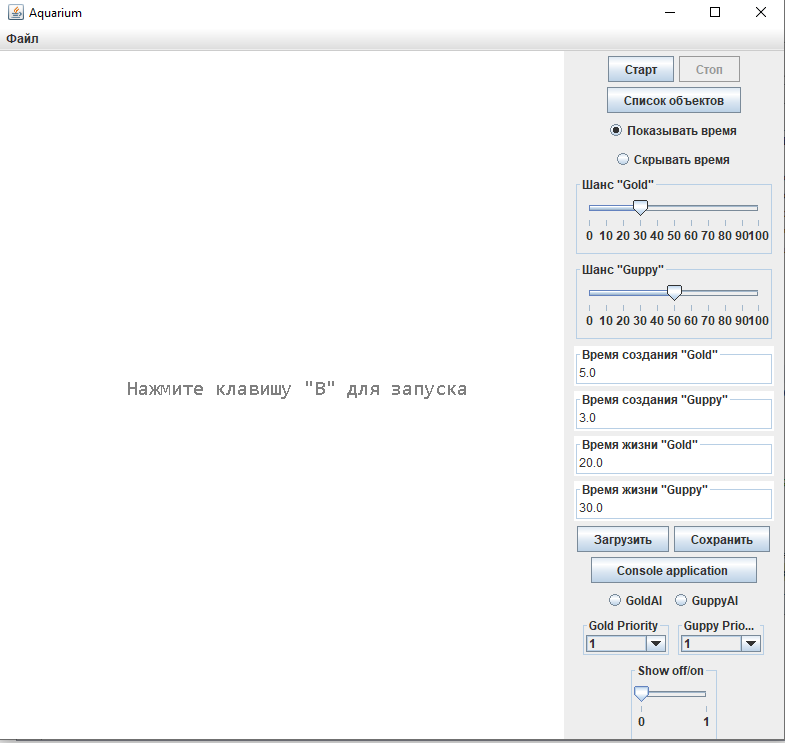


Рисунок 1. Начальная информация

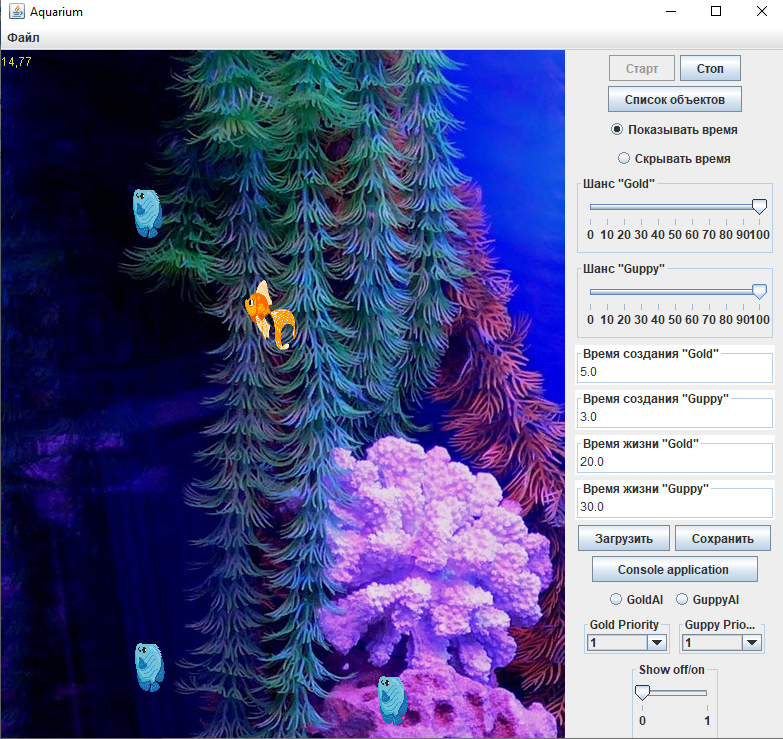


Рисунок 2. Графический интерфейс с панелью управления

Все действия, производимые в панели управления, воздействуют на графический интерфейс (Рисунок 2)

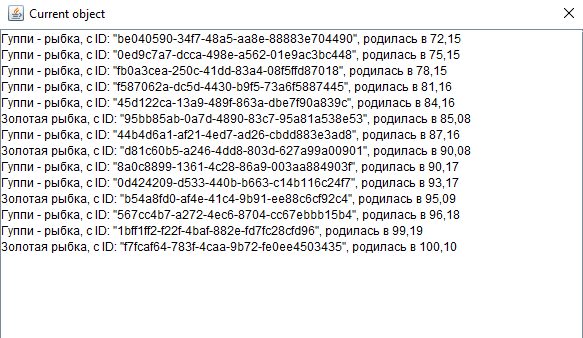


Рисунок 3. Список объектов

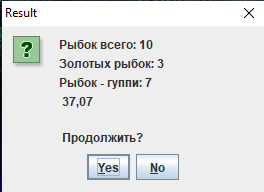


Рисунок 4. Вывод информации

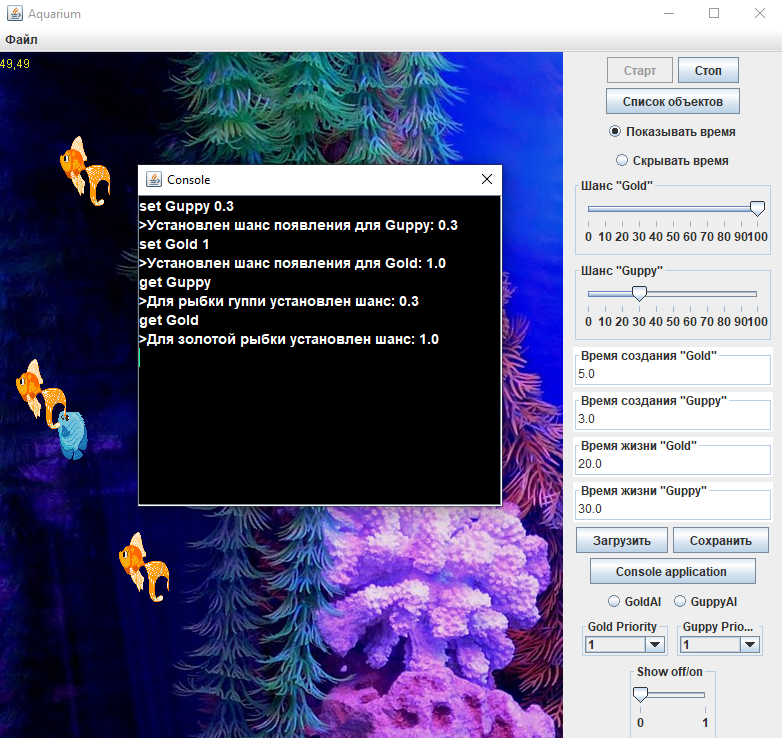


Рисунок 5. Работа с консолью

При включении отображения результата, при остановке симуляции, мы можем увидеть панель информации, в которой содержатся данные о генерации, необходимо выбрать, продолжить ли симуляцию с момента остановки или завершить ее. В списке объектов отображается разновидность рыбки, сгенерированный id, время рождения. Движение объектов синхронизировано, зависит от приоритета потока рыбки. Движение рыбки можно включать и выключать.

Работа с консолью представлена в виде двух команд «Set» и «Get». Проверяется, соответствует ли первое слово одному из двух вариантов, далее должно следовать слово с видом рыбки, и далее число (если это команда «Set»).

При запуске программы загружается файл последней использованной конфигурации программы до ее отключения.

В программе есть функции сохранения текущего набора рыбок и их местоположения, а так же загрузка из файла.

Вывод:

В ходе лабораторной работы были изучены особенности реализации работы потоков в Java, управления настройками программы, работа с сериализацией, а также изучили работу консольного приложения.

Листинг программы

Habitat.java

package aquarium;

import java.util.HashSet;

import java.util.LinkedList;

import java.util.TreeMap;

import java.util.UUID;

public class Habitat {

public LinkedList<Fish> ObjCollection = new LinkedList<>();

//private FishList ObjCollection = new FishList();

public HashSet<UUID> idSet = new HashSet<>();

public TreeMap<UUID, Double> birthdayMap = new TreeMap<UUID, Double>();

float timerGold = 0;

float timerGuppy = 0;

public GoldAI ThreadGoldAI = null;

public GuppyAI ThreadGuppyAI = null;

public void start() {

// Запускаем потоки

if (ThreadGuppyAI == null) {

ThreadGuppyAI = new GuppyAI(ObjCollection);

ThreadGuppyAI.start();

}

if (ThreadGoldAI == null) {

ThreadGoldAI = new GoldAI(ObjCollection);

ThreadGoldAI.start();

}

}

public void stop() {

// Останавливаем потоки

if (ThreadGuppyAI != null) ThreadGuppyAI.going = false;

if (ThreadGoldAI != null) ThreadGoldAI.going = false;

}

void Update(double timer){

timerGold += timer;

timerGuppy += timer;

if (timerGold > Gold.N1) {

if (Gold.isCreated()) addToCollections(0);

timerGold = 0;

}

if (timerGuppy > Guppy.N2) {

if (Guppy.isCreated()) addToCollections(1);

timerGuppy = 0;

}

}

public void addToCollections(int i){

int hi = HabitatView.heigthImg;

int wi = HabitatView.widthImg;

int upBoundWidth = HabitatView.Width - wi;

int upBoundHeight = HabitatView.Height - hi;

if(i == 0) ObjCollection.add(Fish.Sum, new Gold((int) ( Math.random() \* upBoundWidth ), (int) ( Math.random() \* upBoundHeight)));

else ObjCollection.add(Fish.Sum, new Guppy((int) ( Math.random() \* upBoundWidth ), (int) (Math.random() \* upBoundHeight)));

UUID tmpID;

// Объект точно получит уникальный ИД

do{

tmpID = UUID.randomUUID();

ObjCollection.get(Fish.Sum-1).setID(tmpID);

} while (idSet.contains(tmpID));

idSet.add(tmpID);

birthdayMap.put(tmpID, ObjCollection.get(Fish.Sum-1).getBornTime());

}

}

Main.java

package aquarium;

import java.awt.BorderLayout;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.event.ComponentAdapter;

import java.awt.event.ComponentEvent;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import javax.swing.BorderFactory;

import javax.swing.ButtonGroup;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JPanel;

import javax.swing.JRadioButton;

import javax.swing.JSlider;

import javax.swing.JTextField;

public class Main extends JFrame{

HabitatView view;

static Main frame;

static JPanel panel;

static JButton button\_start = new JButton("Старт");

static JButton button\_stop = new JButton("Стоп");

static JButton show\_list = new JButton("Список объектов");

static JRadioButton rbutton\_show\_time = new JRadioButton("Показывать время");

static JRadioButton rbutton\_hide\_time = new JRadioButton("Скрывать время");

static JSlider slider\_chance\_Gold = new JSlider(JSlider.HORIZONTAL,0, 100, 0);

static JSlider slider\_chance\_Guppy = new JSlider(JSlider.HORIZONTAL,0, 100, 0);

static JTextField txt\_spawn\_time\_Gold = new JTextField();

static JTextField txt\_spawn\_time\_Guppy = new JTextField();

static JTextField txt\_life\_time\_Gold = new JTextField();

static JTextField txt\_life\_time\_Guppy = new JTextField();

static JMenuBar menuBar = new JMenuBar();

static JMenu menu = new JMenu("Файл");

static JMenuItem stop\_menu = new JMenuItem("Стоп");

static JMenuItem start\_menu = new JMenuItem("Старт");

static ButtonGroup button\_group = new ButtonGroup();

public static int Width;

public static int Height;

static JSlider slider\_show\_info = new JSlider(JSlider.HORIZONTAL,0, 1, 0);

static JRadioButton switchGoldAI = new JRadioButton("GoldAI");

static JRadioButton switchGuppyAI = new JRadioButton("GuppyAI");

static JComboBox priorityGoldAI = new JComboBox();

static JComboBox priorityGuppyAI = new JComboBox();

static JButton save\_files = new JButton("Сохранить");

static JButton load\_files = new JButton("Загрузить");

static JButton console = new JButton(" Console application ");

/\*\*

\* @param args the command line arguments

\*/

Main(String name){

super(name);

panel = new JPanel(new FlowLayout());

panel.setPreferredSize(new Dimension(220,600));

view = new HabitatView();

setPreferredSize(new Dimension(1000, 800));

setMinimumSize(new Dimension(800, 800));

setVisible(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

add(view);

setJMenuBar(menuBar);

slider\_show\_info.setBorder(BorderFactory.createTitledBorder("Show off/on"));

slider\_show\_info.setPreferredSize(new Dimension(90,80));

slider\_show\_info.setMajorTickSpacing(1);

slider\_show\_info.setMinorTickSpacing(1);

slider\_show\_info.setPaintTicks(true);

slider\_show\_info.setPaintLabels(true);

panel.add(slider\_show\_info);

slider\_show\_info.setVisible(true);

add(panel, BorderLayout.EAST);

addComponentListener(new ComponentAdapter(){

public void componentResized(ComponentEvent e) {

HabitatView.Width = getWidth()-235;

HabitatView.Height = getHeight()-50;

}

});

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

try {

HabitatView.saveConfig();

} catch (Exception ee) {

System.out.println("Ошибка сохранения");

}

System.exit(0);

}

});

}

public static void main(String[] args) {

frame = new Main("Aquarium");

// TODO code application logic here

}

}

Fish.java

package aquarium;

import java.util.UUID;

public abstract class Fish implements IBehaviour,Serializable {

private int x;

private int y;

public static int Sum=0;

private UUID ID;

private double bornTime;

private boolean isUpGoing;

public Fish(){

x = 0;

y = 0;

Sum++;

}

public Fish(int x1, int y1){

x = x1;

y = y1;

Sum++;

}

public int getX(){

return x;

}

public int getY(){

return y;

}

public void setX(int x1){

x = x1;

}

public void setY(int y1){

y = y1;

}

public void setBornTime(double bornTime1){

bornTime = bornTime1;

}

public double getBornTime(){

return bornTime;

}

public void setID(){

ID = UUID.randomUUID();

}

public void setID(String id){

ID = UUID.fromString(id);

}

public void setID(UUID id){

ID = id;

}

public UUID getID(){

return ID;

}

public boolean get\_going(){

return isUpGoing;

}

public void set\_going(boolean go){

isUpGoing = go;

}

public void move(int windowWidth, int windowHeight){

}

}

Gold

package aquarium;

public class Gold extends Fish {

public static float P1 = 0.3f;

public static float N1 = 5;

public static int Sum1 = 0;

public static double life\_time = 20;

double v = 1;

Gold(int x, int y){

super(x,y);

Sum1++;

super.setBornTime(HabitatView.ElapsedTime);

super.setID();

}

public void move(int windowWidth, int windowHeight){

setX((int) (getX() + (get\_going() ? v : -v)));

if (getX() < 0 || getX() > windowWidth-50) {

set\_going(!get\_going());

}

}

static boolean isCreated() {

if (P1 >= Math.random())

return true;

else

return false;

}

@Override

public int getx() {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public int gety() {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void setx(int x) {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void sety(int y) {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

Guppy

package aquarium;

public class Guppy extends Fish{

public static float P2 = 0.5f;

public static float N2 = 3;

public static int Sum2 = 0;

public static double life\_time =30;

double v = 1;

Guppy(int x, int y){

super(x,y);

Sum2++;

super.setBornTime(HabitatView.ElapsedTime);

super.setID();

}

static boolean isCreated() {

if (P2 >= Math.random())

return true;

else

return false;

}

@Override

public void move(int windowWidth, int windowHeight){

setY((int) (getY() + (get\_going() ? v : -v)));

if (getY() < 0 || getY() > windowHeight-70) {

set\_going(!get\_going());

}

}

@Override

public int getx() {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public int gety() {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void setx(int x) {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

@Override

public void sety(int y) {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

IBehaviour.java

package aquarium;

public interface IBehaviour {

void move(int windowWidth, int windowHeight);

int getx();

int gety();

void setx(int x);

void sety(int y);

}

HabitatView.java

﻿/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package aquarium;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.FlowLayout;

import java.awt.Font;

import java.awt.Frame;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.Timer;

import java.util.TimerTask;

import java.util.UUID;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.imageio.ImageIO;

import javax.swing.BorderFactory;

import javax.swing.JDialog;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JSlider;

import javax.swing.JTextArea;

import javax.swing.JTextField;

/\*\*

\*

\* @author Dexp

\*/

public class HabitatView extends JPanel {

static Habitat habitat;

public static int widthImg = 50;

public static int heigthImg = 70;

private double startTime;

private double pauseTime;

public static double ElapsedTime;

static Timer timer;

private double deltaTime;

static boolean show = true;

static boolean simulating = false;

static boolean firstRun = true;

Image img1;

Image img2;

Image imgBG;

int show\_info;

int show\_obj = 1;

public static int Width;

public static int Height;

private Frame frame\_list;

private float \_refTime = 5F;

private float \_refTimer = \_refTime;

Image imgGuppy\_up;

Image imgGuppy\_down;

Image imgGold\_right;

Image imgGold\_left;

//JPanel panel;

public HabitatView() {

try {

img1 = ImageIO.read(new File("Gold.png"));

img2 = ImageIO.read(new File("Guppy.png"));

imgBG = ImageIO.read(new File("BG.jpg"));

imgGuppy\_up = ImageIO.read(new File("Guppy\_up.png"));

imgGuppy\_down = ImageIO.read(new File("Guppy\_down.png"));

imgGold\_left = ImageIO.read(new File("Gold\_left.png"));

imgGold\_right = ImageIO.read(new File("Gold\_right.png"));

} catch (IOException ex) {

Logger.getLogger(HabitatView.class.getName()).log(Level.SEVERE, null, ex);

}

imgGuppy\_up = imgGuppy\_up.getScaledInstance(widthImg, heigthImg, Image.SCALE\_DEFAULT);

imgGuppy\_down = imgGuppy\_down.getScaledInstance(widthImg, heigthImg, Image.SCALE\_DEFAULT);

imgGold\_left = imgGold\_left.getScaledInstance(widthImg, heigthImg, Image.SCALE\_DEFAULT);

imgGold\_right = imgGold\_right.getScaledInstance(widthImg, heigthImg, Image.SCALE\_DEFAULT);

img1 = img1.getScaledInstance(widthImg, heigthImg, Image.SCALE\_DEFAULT);

img2 = img2.getScaledInstance(widthImg, heigthImg, Image.SCALE\_DEFAULT);

initComponents();

repaint();

HabitatView.Width = this.getWidth();

HabitatView.Height = this.getHeight();

System.out.println(Height);

System.out.println(Width);

addMouseListener(new PanelMouseListener());

addKeyListener((KeyListener) new processKeyEvent());

}

private void initComponents() {

Main.rbutton\_show\_time.setSelected(true);

Main.slider\_chance\_Gold.setValue((int) (Gold.P1 \* 100));

Main.slider\_chance\_Guppy.setValue((int) (Guppy.P2 \* 100));

Main.txt\_spawn\_time\_Gold.setText(String.valueOf(Gold.N1));

Main.txt\_spawn\_time\_Guppy.setText(String.valueOf(Guppy.N2));

Main.txt\_life\_time\_Gold.setText(String.valueOf(Gold.life\_time));

Main.txt\_life\_time\_Guppy.setText(String.valueOf(Guppy.life\_time));

Main.button\_start.addActionListener(e -> startSim());

Main.panel.add(Main.button\_start);

Main.button\_stop.addActionListener(e -> stopSim());

Main.panel.add(Main.button\_stop);

Main.show\_list.addActionListener(e -> show\_list\_object());

Main.panel.add(Main.show\_list);

Main.rbutton\_show\_time.addActionListener(e -> show = true);

Main.rbutton\_hide\_time.addActionListener(e -> show = false);

Main.button\_group.add(Main.rbutton\_show\_time);

Main.button\_group.add(Main.rbutton\_hide\_time);

Main.panel.add(Main.rbutton\_show\_time);

Main.panel.add(Main.rbutton\_hide\_time);

Main.slider\_chance\_Gold.setBorder(BorderFactory.createTitledBorder("Шанс \"Gold\""));

Main.slider\_chance\_Gold.setPreferredSize(new Dimension(200, 80));

Main.slider\_chance\_Gold.setMajorTickSpacing(10);

Main.slider\_chance\_Gold.setPaintTicks(true);

Main.slider\_chance\_Gold.setPaintLabels(true);

Main.slider\_chance\_Gold.addChangeListener(e -> Gold.P1 = (((JSlider) e.getSource()).getValue()) / 100f);

Main.panel.add(Main.slider\_chance\_Gold);

Main.slider\_show\_info.addChangeListener(e -> show\_info = (((JSlider) e.getSource()).getValue()));

Main.slider\_chance\_Guppy.setBorder(BorderFactory.createTitledBorder("Шанс \"Guppy\""));

Main.slider\_chance\_Guppy.setPreferredSize(new Dimension(200, 80));

Main.slider\_chance\_Guppy.setMajorTickSpacing(10);

Main.slider\_chance\_Guppy.setPaintTicks(true);

Main.slider\_chance\_Guppy.setPaintLabels(true);

Main.slider\_chance\_Guppy.addChangeListener(e -> Guppy.P2 = (((JSlider) e.getSource()).getValue()) / 100f);

Main.panel.add(Main.slider\_chance\_Guppy);

Main.txt\_spawn\_time\_Gold.setName("Spawn\_Gold");

Main.txt\_spawn\_time\_Gold.setBorder(BorderFactory.createTitledBorder("Время создания \"Gold\""));

Main.txt\_spawn\_time\_Gold.setPreferredSize(new Dimension(200, 40));

Main.txt\_spawn\_time\_Gold.addActionListener(new Txt\_name\_ActionListener());

Main.panel.add(Main.txt\_spawn\_time\_Gold);

Main.txt\_spawn\_time\_Guppy.setName("Spawn\_Guppy");

Main.txt\_spawn\_time\_Guppy.setBorder(BorderFactory.createTitledBorder("Время создания \"Guppy\""));

Main.txt\_spawn\_time\_Guppy.setPreferredSize(new Dimension(200, 40));

Main.txt\_spawn\_time\_Guppy.addActionListener(new Txt\_name\_ActionListener());

Main.panel.add(Main.txt\_spawn\_time\_Guppy);

Main.txt\_life\_time\_Gold.setName("life\_time\_Gold");

Main.txt\_life\_time\_Gold.setBorder(BorderFactory.createTitledBorder("Время жизни \"Gold\""));

Main.txt\_life\_time\_Gold.setPreferredSize(new Dimension(200, 40));

Main.txt\_life\_time\_Gold.addActionListener(new Txt\_name\_ActionListener());

Main.panel.add(Main.txt\_life\_time\_Gold);

Main.txt\_life\_time\_Guppy.setName("life\_time\_Guppy");

Main.txt\_life\_time\_Guppy.setBorder(BorderFactory.createTitledBorder("Время жизни \"Guppy\""));

Main.txt\_life\_time\_Guppy.setPreferredSize(new Dimension(200, 40));

Main.txt\_life\_time\_Guppy.addActionListener(new Txt\_name\_ActionListener());

Main.panel.add(Main.txt\_life\_time\_Guppy);

Main.menu.add(Main.start\_menu);

Main.menu.add(Main.stop\_menu);

Main.start\_menu.addActionListener(e -> startSim());

Main.stop\_menu.addActionListener(e -> stopSim());

Main.menuBar.add(Main.menu);

Main.start\_menu.setEnabled(true);

Main.button\_start.setEnabled(true);

Main.stop\_menu.setEnabled(false);

Main.button\_stop.setEnabled(false);

Main.load\_files.addActionListener(e -> load\_file());

Main.panel.add(Main.load\_files);

Main.save\_files.addActionListener(e -> save\_file());

Main.panel.add(Main.save\_files);

Main.console.addActionListener(e-> new Console(Main.frame));

Main.panel.add( Main.console);

Main.switchGoldAI.addActionListener(e -> {

if (Main.switchGoldAI.isSelected() == true) {

synchronized (habitat.ThreadGoldAI.locker) {

habitat.ThreadGoldAI.sleeping = false;

habitat.ThreadGoldAI.locker.notify();

}

} else {

habitat.ThreadGoldAI.sleeping = true;

}

});

Main.panel.add(Main.switchGoldAI);

Main.switchGuppyAI.addActionListener(e -> {

if (Main.switchGuppyAI.isSelected() == true) {

synchronized (habitat.ThreadGuppyAI.locker) {

habitat.ThreadGuppyAI.sleeping = false;

habitat.ThreadGuppyAI.locker.notify();

}

} else {

habitat.ThreadGuppyAI.sleeping = true;

}

});

Main.panel.add(Main.switchGuppyAI);

//Выпадающие списки для здания приоритетов потоков

Main.priorityGoldAI.setBorder(BorderFactory.createTitledBorder("Gold Priority"));

Main.priorityGoldAI.setPreferredSize(new Dimension(90, 40));

for (int i = 0; i < 10; i++) {

Main.priorityGoldAI.addItem(i + 1);

}

Main.priorityGoldAI.addActionListener(e -> habitat.ThreadGoldAI.setPriority(Main.priorityGoldAI.getSelectedIndex() + 1));

Main.panel.add(Main.priorityGoldAI);

Main.priorityGuppyAI.setBorder(BorderFactory.createTitledBorder("Guppy Priority"));

Main.priorityGuppyAI.setPreferredSize(new Dimension(90, 40));

for (int i = 0; i < 10; i++) {

Main.priorityGuppyAI.addItem(i + 1);

}

Main.priorityGuppyAI.addActionListener(e -> habitat.ThreadGuppyAI.setPriority(Main.priorityGuppyAI.getSelectedIndex() + 1));

Main.panel.add(Main.priorityGuppyAI);

}

class processKeyEvent extends KeyAdapter {

public void keyReleased(KeyEvent e) {

super.keyReleased(e);

if (e.getKeyCode() == KeyEvent.VK\_T) {

show = !show;

}

if (e.getKeyCode() == KeyEvent.VK\_B) {

if (!simulating) {

startSim();

}

}

if (e.getKeyCode() == KeyEvent.VK\_E) {

stopSim();

}

}

}

class PanelMouseListener extends MouseAdapter {

public void mousePressed(MouseEvent e) {

super.mouseClicked(e);

requestFocusInWindow();

}

}

public class Txt\_name\_ActionListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

JTextField tmp = (JTextField) e.getSource();

float value;

try {

value = Float.parseFloat(tmp.getText());

} catch (NumberFormatException ee) {

value = 0;

JOptionPane.showMessageDialog(HabitatView.this, "Неверный ввод. Возвращено значение по умолчанию.", "ERROR404", JOptionPane.ERROR\_MESSAGE);

}

if (value <= 0) {

value = 1;

}

System.out.println(value);

switch (tmp.getName()) {

case ("Spawn\_Gold"):

Gold.N1 = value;

break;

case ("Spawn\_Guppy"):

Guppy.N2 = value;

break;

case ("life\_time\_Gold"):

Gold.life\_time = value;

break;

case ("life\_time\_Guppy"):

Guppy.life\_time = value;

break;

}

tmp.setText(String.valueOf(value));

}

}

//Слушатель нажатия клавиш

private void update() {

habitat.Update(deltaTime);

\_refTimer -= deltaTime;

if (\_refTimer <= 0) {

\_refTimer = \_refTime;

synchronized (habitat.ObjCollection) {

for (Iterator<Fish> iterator = habitat.ObjCollection.iterator(); iterator.hasNext();) {

Fish temp = iterator.next();

if (temp instanceof Gold && (ElapsedTime - temp.getBornTime() > Gold.life\_time)) {

--Gold.Sum1;

} else if (temp instanceof Guppy && (ElapsedTime - temp.getBornTime() > Guppy.life\_time)) {

--Guppy.Sum2;

}

if (Fish.Sum != Gold.Sum1 + Guppy.Sum2) {

UUID tmpID = temp.getID();

habitat.idSet.remove(tmpID);

habitat.birthdayMap.remove(tmpID);

iterator.remove();

Fish.Sum--;

}

}

}

}

repaint();

}

//Отрисовка наших объектов и таймера при помощи буфера

public void paint(Graphics g) {

super.paint(g);

clearViewScreen(g);

if (simulating) {

g.drawImage(imgBG, 0, 0, this);

//Отрисовывыаем объекты

for (Fish temp : habitat.ObjCollection) {

if (temp instanceof Gold && (ElapsedTime - temp.getBornTime() < Gold.life\_time)) {

if (!temp.get\_going()) {

g.drawImage(imgGold\_left, temp.getX(), temp.getY(), this);

} else {

g.drawImage(imgGold\_right, temp.getX(), temp.getY(), this);

}

} else if (temp instanceof Guppy && (ElapsedTime - temp.getBornTime() < Guppy.life\_time)) {

if (temp.get\_going()) {

g.drawImage(imgGuppy\_down, temp.getX(), temp.getY(), this);

} else {

g.drawImage(imgGuppy\_up, temp.getX(), temp.getY(), this);

}

}

}

//Отрисовывваем таймер

if (show) {

g.setColor(new Color(222, 222, 30, 255));

g.drawString(String.format("%.2f", ElapsedTime), 0, 16);

}

//освобождаем ресуры

g.dispose();

} else {

if (firstRun) {

g.setColor(Color.WHITE);

g.fillRect(0, 0, getWidth(), getHeight());

g.setFont(new Font("Consolas", Font.PLAIN, 20));

g.setColor(Color.GRAY);

g.drawString("Нажмите клавишу \"B\" для запуска", (getWidth() - 31 \* 10) / 2, getHeight() / 2);

}

}

}

private void clearViewScreen(Graphics g) {

g.setColor(Color.GRAY);

g.fillRect(0, 0, getWidth(), getHeight());

}

//вспмогательяные методы пауза. старт, стоп, продолжить

void startSim() {

firstRun = false;

simulating = true;

try {

loadConfig();

} catch (FileNotFoundException ex) {

Logger.getLogger(HabitatView.class.getName()).log(Level.SEVERE, null, ex);

}

Main.start\_menu.setEnabled(false);

Main.button\_start.setEnabled(false);

Main.stop\_menu.setEnabled(true);

Main.button\_stop.setEnabled(true);

habitat = new Habitat();

habitat.start();

timer = new Timer();

timer.schedule(new SimulationLoop(), 0, 10);

Fish.Sum = 0;

Gold.Sum1 = 0;

Guppy.Sum2 = 0;

}

void stopSim() {

simulating = false;

if (show\_info == 1) {

int result = JOptionPane.showConfirmDialog(HabitatView.this, String.format("Рыбок всего: %d\nЗолотых рыбок: %d\nРыбок - гуппи: %d\n %.2f\n\n Остановить?", Fish.Sum, Gold.Sum1, Guppy.Sum2, ElapsedTime), "Result", JOptionPane.OK\_CANCEL\_OPTION);

if (result == JOptionPane.OK\_OPTION) {

firstRun = true;

Main.start\_menu.setEnabled(true);

Main.button\_start.setEnabled(true);

Main.stop\_menu.setEnabled(false);

Main.button\_stop.setEnabled(false);

ElapsedTime = 0;

repaint();

habitat.stop();

return;

}

simulating = true;

return;

}

Main.start\_menu.setEnabled(true);

Main.button\_start.setEnabled(true);

Main.stop\_menu.setEnabled(false);

Main.button\_stop.setEnabled(false);

firstRun = true;

ElapsedTime = 0;

habitat.stop();

repaint();

}

public void show\_list\_object() {

simulating = false;

// if (show\_obj==1){

JDialog container = new JDialog(frame\_list, "Current object", true);

JTextArea textArea = new JTextArea();

JScrollPane scrollPane = new JScrollPane(textArea);

container.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent e) {

simulating = true;

container.dispose();

}

});

StringBuilder str = new StringBuilder();

for (Fish tmp : habitat.ObjCollection) {

if (tmp instanceof Gold && (ElapsedTime - tmp.getBornTime() < Gold.life\_time)) {

str.append(String.format("Золотая рыбка, с ID: \"%s\", родилась в %.2f\n", tmp.getID(), tmp.getBornTime()));

} else if (tmp instanceof Guppy && (ElapsedTime - tmp.getBornTime() < Guppy.life\_time)) {

str.append(String.format("Гуппи - рыбка, с ID: \"%s\", родилась в %.2f\n", tmp.getID(), tmp.getBornTime()));

}

}

textArea.setText(str.toString());

textArea.setDisabledTextColor(Color.BLACK);

container.setPreferredSize(new Dimension(600, 200));

container.add(scrollPane);

textArea.setEnabled(false);

container.pack();

container.setVisible(true);

// }

}

void load\_file() {

synchronized (habitat.ObjCollection) {

habitat.ObjCollection.clear();

habitat.idSet.clear();

habitat.birthdayMap.clear();

Fish.Sum = 0;

Gold.Sum1 = 0;

Guppy.Sum2 = 0;

LinkedList<Fish> tmp = null;

try {

ObjectInputStream oin = new ObjectInputStream(new FileInputStream("data.dat"));

ElapsedTime = (double) oin.readObject();

tmp = (LinkedList<Fish>) oin.readObject();

double currentTime = System.currentTimeMillis();

startTime = currentTime - ElapsedTime \* 1000;

System.out.println(ElapsedTime);

System.out.println(currentTime);

System.out.println(startTime);

for (Fish temp : tmp) {

temp.setBornTime(ElapsedTime);

Fish.Sum++;

habitat.idSet.add(temp.getID());

habitat.birthdayMap.put(temp.getID(), temp.getBornTime());

if (temp instanceof Gold) {

Gold.Sum1++;

} else {

Guppy.Sum2++;

}

}

habitat.ObjCollection = tmp;

habitat.ThreadGoldAI.set\_list(habitat.ObjCollection);

habitat.ThreadGuppyAI.set\_list(habitat.ObjCollection);

} catch (Exception e) {

e.printStackTrace();

}

}

}

void save\_file() {

synchronized (habitat.ObjCollection) {

ObjectOutputStream oos;

try {

oos = new ObjectOutputStream(new FileOutputStream("data.dat"));

oos.writeObject(ElapsedTime);

oos.writeObject(habitat.ObjCollection);

oos.close();

} catch (IOException ex) {

Logger.getLogger(HabitatView.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

static void saveConfig() {

BufferedWriter out;

try {

out = new BufferedWriter(new FileWriter("Config.txt"));

out.write(String.valueOf((Gold.P1)));

out.write('\n');

out.write(String.valueOf((Gold.N1)));

out.write('\n');

out.write(String.valueOf((Gold.life\_time)));

out.write('\n');

out.write(String.valueOf((Guppy.P2)));

out.write('\n');

out.write(String.valueOf((Guppy.N2)));

out.write('\n');

out.write(String.valueOf((Guppy.life\_time)));

out.close();

} catch (IOException ex) {

Logger.getLogger(HabitatView.class.getName()).log(Level.SEVERE, null, ex);

}

}

static void loadConfig() throws FileNotFoundException {

BufferedReader in;

in = new BufferedReader(new FileReader("Config.txt"));

try {

Gold.P1 = Float.parseFloat(in.readLine());

Gold.N1 = Float.parseFloat(in.readLine());

Gold.life\_time = Float.parseFloat(in.readLine());

Guppy.P2 = Float.parseFloat(in.readLine());

Guppy.N2 = Float.parseFloat(in.readLine());

Guppy.life\_time = Float.parseFloat(in.readLine());

Main.txt\_spawn\_time\_Gold.setText(String.valueOf(Gold.N1));

Main.txt\_spawn\_time\_Guppy.setText(String.valueOf(Guppy.N2));

Main.txt\_life\_time\_Gold.setText(String.valueOf(Gold.life\_time));

Main.txt\_life\_time\_Guppy.setText(String.valueOf(Guppy.life\_time));

Main.slider\_chance\_Gold.setValue( (int)(Gold.P1\*100));

Main.slider\_chance\_Guppy.setValue((int)(Guppy.P2\*100));

in.close();

} catch (IOException ex) {

Logger.getLogger(HabitatView.class.getName()).log(Level.SEVERE, null, ex);

}

}

//класс таймера

private class SimulationLoop extends TimerTask {

private double lastTime;

public SimulationLoop() {

startTime = System.currentTimeMillis();

lastTime = startTime;

pauseTime = 0;

}

@Override

public void run() {

if (simulating) {

double currentTime = System.currentTimeMillis();

// Время, прошедшее от начала, в секундах

ElapsedTime = (currentTime - startTime - pauseTime) / 1000;

// Время, прошедшее с последнего обновления, в секундах

deltaTime = (currentTime - lastTime) / 1000;

lastTime = currentTime;

// Вызываем обновление

update();

} else {

double currentTime = System.currentTimeMillis();

pauseTime = (currentTime - startTime - ElapsedTime \* 1000);

lastTime = currentTime;

}

}

}

}

GuppyAI.java

package aquarium;

import java.util.LinkedList;

public class GuppyAI extends BaseAI {

private LinkedList<Fish> list;

GuppyAI(LinkedList<Fish> temp) {

list = temp;

}

void set\_list(LinkedList<Fish> temp) {

list = temp;

}

public synchronized void step() {

for (Fish temp : list) {

if (temp instanceof Guppy && (HabitatView.ElapsedTime - temp.getBornTime() < Guppy.life\_time)) {

temp.move(HabitatView.Width, HabitatView.Height);

}

}

try {

Thread.sleep(2);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

GoldAI.java

package aquarium;

import java.util.LinkedList;

public class GoldAI extends BaseAI {

private LinkedList<Fish> list;

GoldAI(LinkedList<Fish> temp){

list = temp;

}

void set\_list(LinkedList<Fish> temp){

list = temp;

}

public synchronized void step(){

for(Fish temp : list)

if (temp instanceof Gold && (HabitatView.ElapsedTime - temp.getBornTime() < Gold.life\_time)){

temp.move(HabitatView.Width,HabitatView.Height);

}

try {

Thread.sleep(2);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

BaseAI.java

package aquarium;

public abstract class BaseAI extends Thread {

boolean going = true;

boolean sleeping = true;

public static Object locker = new Object();

public void run() {

while (going) {

synchronized (locker) {

if (sleeping) {

try {

System.out.println("own:: Waiting");

locker.wait();

System.out.println("own:: Running again");

} catch (InterruptedException e) {

System.err.println("own:: Interrupted: " + e.getMessage());

}

}

step();

}

}

}

public void step() {

}

}

Console.java

package aquarium;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.PipedInputStream;

import java.io.PipedOutputStream;

import java.nio.charset.StandardCharsets;

public final class Console extends JDialog {

private PipedInputStream inputStr = new PipedInputStream();

final JTextArea consoleText = new JTextArea("");

public Console(Main owner) {

super(owner, false);

setTitle("Console");

setSize(380, 350);

final JScrollPane scroll = new JScrollPane(consoleText);

consoleText.setFont(new Font("Arial", Font.BOLD, 14));

consoleText.setBackground(new Color(255, 255, 255));

consoleText.setForeground(new Color(0, 0, 0));

consoleText.setSelectedTextColor(new Color(255, 163, 26));

consoleText.setSelectionColor(Color.WHITE);

consoleText.setCaretColor(new Color(25, 21, 12));

add(scroll);

setVisible(true);

setLocationRelativeTo(null);

PipedOutputStream pos = new PipedOutputStream();

try {

pos.connect(inputStr);

consoleText.getDocument().addDocumentListener(new DocumentChangeListener() {

@Override

public void changeText() {

try {

pos.write(consoleText.getText().getBytes());

pos.flush();

} catch (IOException e) {

e.printStackTrace();

}

}

});

Thread thread = new StreamThread(owner);

thread.start();

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

super.windowClosing(e);

System.out.println("close");

thread.interrupt();

try {

inputStr.close();

} catch (IOException ignored) {

}

}

});

} catch (IOException ioe) {

ioe.printStackTrace();

}

}

class StreamThread extends Thread {

private final Main frame;

private StreamThread(Main frame1) {

frame = frame1;

}

@Override

public void run() {

try {

while (!isInterrupted()) {

ByteArrayOutputStream bytesOutput = new ByteArrayOutputStream();

byte[] buffer = new byte[1024];

int length = inputStr.read(buffer);

if (length != -1) {

bytesOutput.write(buffer, 0, length);

String result = bytesOutput.toString(String.valueOf(StandardCharsets.UTF\_8));

String[] lines = result.split("\n");

String [] words= lines[lines.length-1].split(" ");

for(int i=0;i<words.length;i++)

System.out.println(words[i]);

if (result.charAt(result.length() - 1) == '\n') {

switch (words[0].toLowerCase()) {

case "get":

if(words[1].equals("Gold"))

consoleText.append(String.valueOf(Gold.P1));

if(words[1].equals("Guppy"))

consoleText.append(String.valueOf(Guppy.P2));

break;

case "set":

if(Float.parseFloat(words[2])> 1 ||Float.parseFloat(words[2])<0 )

consoleText.append("Неверные данные");

else {

if(words[1].equals("Gold")){

Gold.P1 =Float.parseFloat(words[2]);

System.out.println(words[2]);

Main.slider\_chance\_Gold.setValue( (int)(Gold.P1\*100));

}

if(words[1].equals("Guppy")){

Guppy.P2 =Float.parseFloat(words[2]);

Main.slider\_chance\_Guppy.setValue( (int)(Guppy.P2\*100));

}

}

break;

}

}

}

}

} catch (Exception ignored) {

}

}

}

}

DocumentChangeListener.java

package aquarium;

import javax.swing.event.DocumentEvent;

import javax.swing.event.DocumentListener;

public abstract class DocumentChangeListener implements DocumentListener {

@Override

public void insertUpdate(DocumentEvent e) {

changeText();

}

@Override

public void removeUpdate(DocumentEvent e) {

changeText();

}

@Override

public void changedUpdate(DocumentEvent e) {

changeText();

}

protected abstract void changeText();

}