МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники



**Лабораторная Работа №3**

**по дисциплине:** *технология программирования*

**на тему:** *Классы-коллекции.*

|  |  |
| --- | --- |
| Выполнил: | 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 классов-коллекций в Java.
2. Доработать программу, созданную в лабораторной работе № 2:

Задание

1. Добавить генерируемым объектам понятия «время рождения» и «время жизни». Время рождения устанавливается в момент генерации объекта и по значению соответствует времени, прошедшему от начала симуляции. Время жизни – время, через которое объект должен исчезнуть, считая от времени рождения;
2. Вынести установку параметров времени жизни объектов в пользовательский интерфейс. Для каждого типа объекта должно задаваться собственное время. Рекомендуется использовать текстовые поля, но следуют помнить о проверке на ввод некорректных данных;
3. Добавить генерируемым объектам уникальные целочисленные идентификаторы (случайные числа), которые назначаются при генерации объекта. Для хранения сгенерированных идентификаторов используйте коллекцию удобную для поиска по варианту;
4. Использовать коллекции по варианту. При генерации объекта происходит добавление его в коллекцию (в класс добавить поле идентификатора), а во вторую коллекцию: идентификаторы существующих объектов, в третью идентификатор + время рождения. При возникновении события по таймеру обойдите коллекцию и удалите все объекты, время жизни которых истекло, а также все данные во вспомогательных коллекциях;
5. Добавьте в панель управления кнопку «Текущие объекты». По нажатию на эту кнопку появляется модальное диалоговое окно, содержащее список всех «живых» объектов на момент нажатия со временем их рождения (тип объекта, время рождения, идентификатор). В класс диалогового окна должна передаваться коллекция с хранением объектов по времени рождения. Типы коллекций задаются вариантом.

***Вариант 3***

Коллекция для хранения объектов: LinkedList

Коллекция для хранения и поиска уникальных идентификаторов: HashSet

Коллекция для хранения времени рождения объектов: TreeMap

Проектирование программы

Обсуждение основных идей алгоритма

Для программы создается семь 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() – функция для отображения списка сгенерированных рыб и их ID

**Gold** — класс, наследуемый от Fish и устанавливающий координаты золотой рыбки, 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.

**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.

**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.

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 и (рисунок 1).

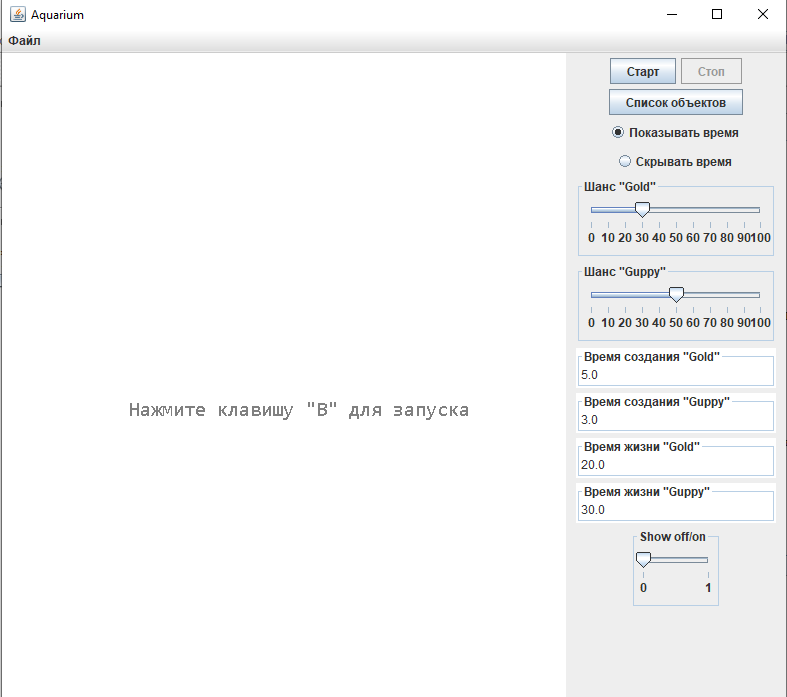


Рисунок 1. Начальная информация

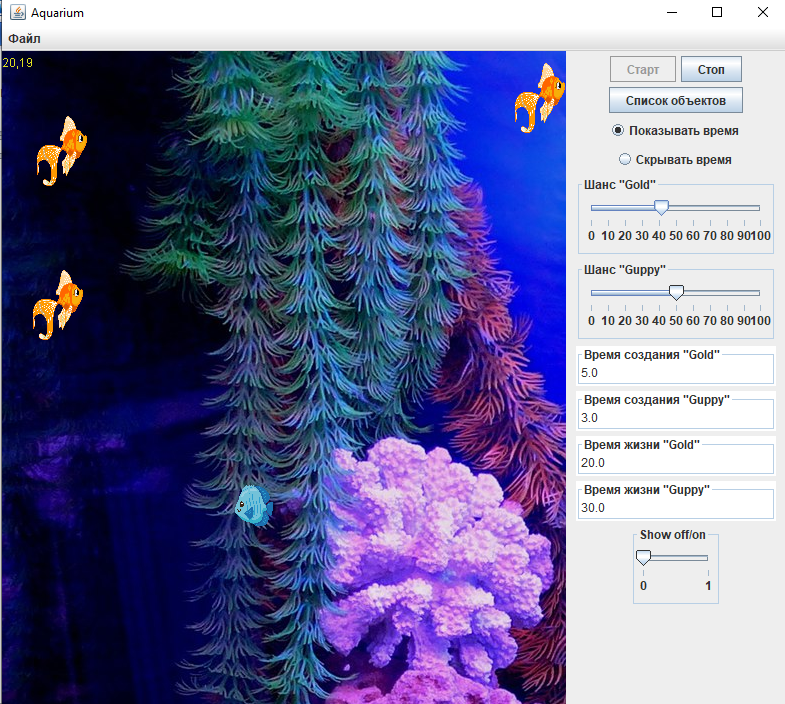


Рисунок 2. Графический интерфейс с панелью управления

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

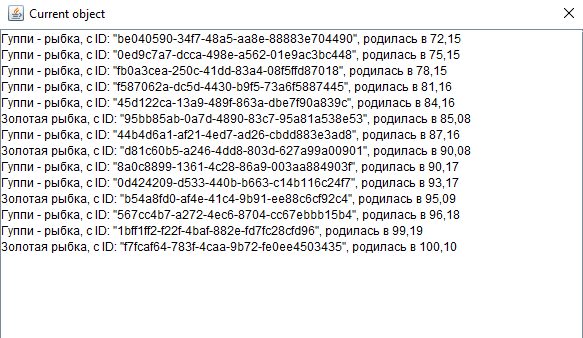


Рисунок 3. Список объектов

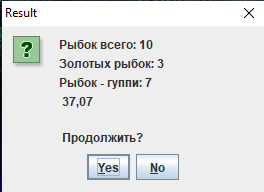


Рисунок 4. Вывод информации

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.

Вывод:

В ходе лабораторной работы были изучены особенности реализации классов-коллекций в Java. Была доработана учебная программа.

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;

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.Dimension;

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 javax.swing.JFrame;

public class Main extends JFrame{

HabitatView view;

static Main frame;

Main(String name){

super(name);

view = new HabitatView();

setPreferredSize(new Dimension(1000, 800));

setMinimumSize(new Dimension(800, 800));

setVisible(true);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

add(view);

addComponentListener(new ComponentAdapter(){

public void componentResized(ComponentEvent e) {

HabitatView.Width = getWidth();

HabitatView.Height = getHeight();

}

});

addKeyListener((KeyListener) new processKeyEvent());

}

public static void main(String[] args) {

frame = new Main("Aquarium");

// TODO code application logic here

}

class processKeyEvent extends KeyAdapter {

public void keyReleased(KeyEvent e) {

super.keyReleased(e);

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

view.show=!view.show;

}

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

if (!view.simulating)

view.startSim();

}

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

view.stopSim();

}

}

}

}

Fish.java

package aquarium;

import java.util.UUID;

public abstract class Fish implements IBehaviour {

private int x;

private int y;

public static int Sum=0;

private UUID ID;

private double bornTime;

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;

}

}

Gold

package aquarium;

public class Gold extends Fish {

public static float P1 = 0.3f;

public static float N1 = 5;

public static int Sum1 = 0;

Gold(int x, int y){

super(x,y);

Sum1++;

}

public void move(){

}

static boolean isCreated() {

if (P1 >= Math.random())

return true;

else

return false;

}

}

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;

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() {

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

}

@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 getx();

int gety();

void setx(int x);

void sety(int y);

}

HabitatView.java

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.File;

import java.io.IOException;

import java.util.Iterator;

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;

public class HabitatView extends JPanel {

static Habitat habitat;

public static int widthImg = 50;

public static int heigthImg = 70;

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;

//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"));

} catch (IOException ex) {

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

}

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);

}

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)) g.drawImage(img1, temp.getX(), temp.getY(), this);

else if (temp instanceof Guppy && (ElapsedTime - temp.getBornTime() < Guppy.life\_time)) g.drawImage(img2, 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;

Main.start\_menu.setEnabled(false);

Main.button\_start.setEnabled(false);

Main.stop\_menu.setEnabled(true);

Main.button\_stop.setEnabled(true);

habitat = new Habitat();

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();

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;

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);

// }

}

//класс таймера

private class SimulationLoop extends TimerTask {

private double pauseTime;

private double startTime;

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;

}

}

}

}