**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ “НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”**

Лабораторная работа №4

по дисциплине «Технологии и методы программирования»

Многопотоковые приложения.

Группа: **АВТ-042**

Студенты: **Антонов А.Р.**

Преподаватель: **Булатов А.Д**

Новосибирск 2022

1. **Цель**

Изучить особенности реализации и работы потоков в Kotlin, управлением приоритетами потоков и синхронизацией потоков.

1. **Задание**

Вариант 10

1. Юр. лица двигаются в левую верхнюю четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке 0;0, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если запись сгенерировалась сразу в этой области, то она никуда не движется. По прибытии в конечную точку запись больше не движется.

Доработать программу, созданную в лабораторной работе № 3:

1. Создать абстрактный класс BaseAI, описывающий «интеллектуальное поведение» объектов. Класс должен создавать поток, обеспечивающий движения объектов коллекции;

2. Реализовать класс BaseAI для каждого из видов объекта, включив в него поведение, описанное в индивидуальном задании по варианту;

3. Синхронизовать работу потоков расчета интеллекта объектов, их рисования и генерации новых объектов. Рисование должно остаться в основном потоке;

4. Добавить в панель управления кнопки для остановки и возобновления работы интеллекта каждого вида объектов. Реализовать через засыпание/пробуждение потоков (методы wait() и notify());

5. Добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

1. **UML-диаграмма**

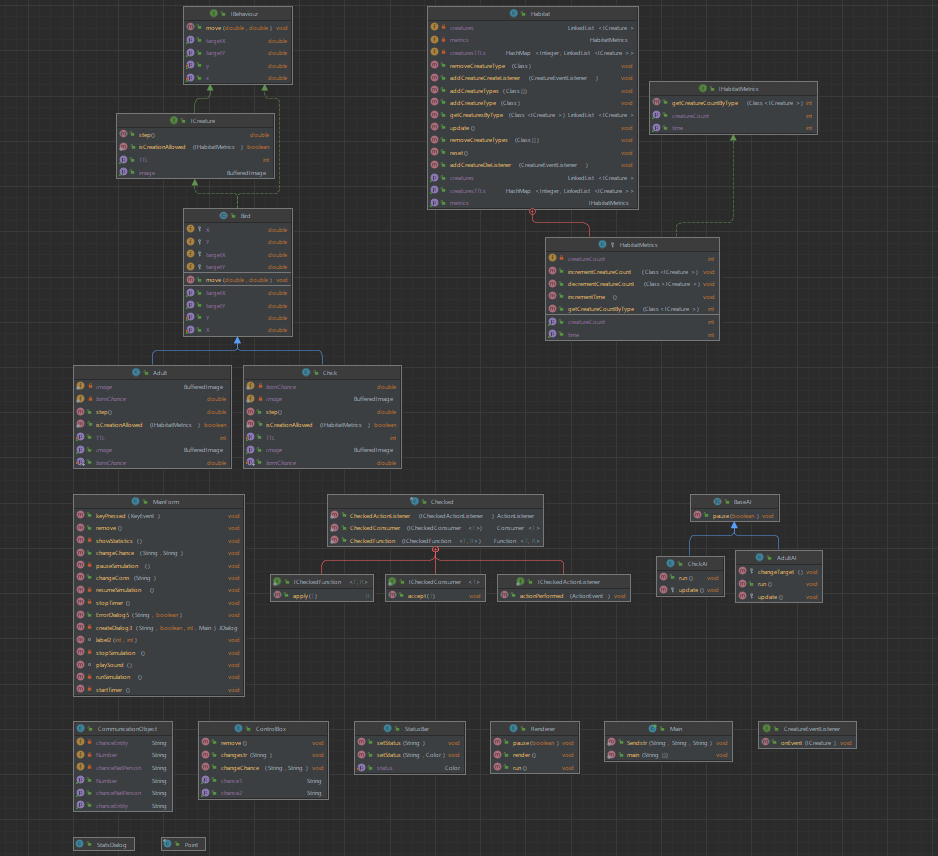


Рисунок 1 UML-диаграмма

1. **Пример работы программы**

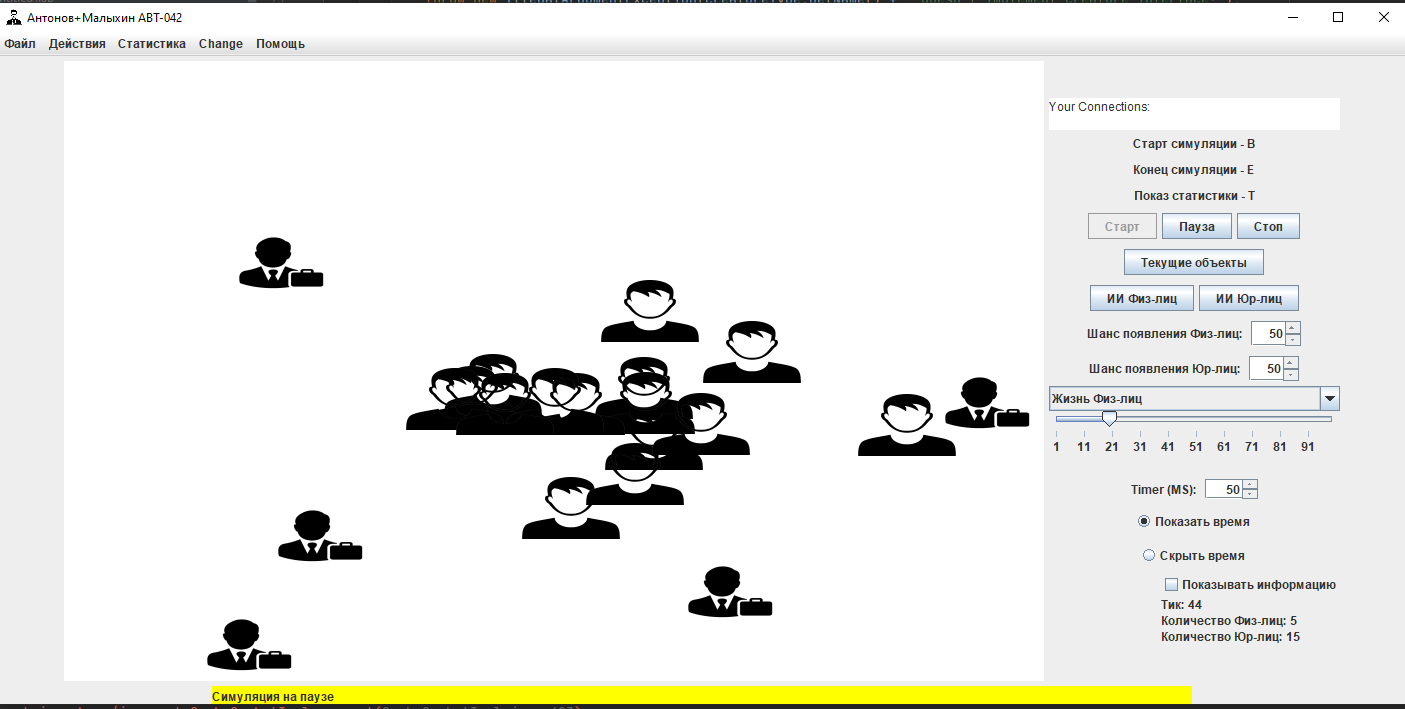


Рисунок 2 Пример работы программы

1. **Вывод**

Мы изучили особенности реализации и работы потоков в Kotlin, управление приоритетами потоков и синхронизацией потоков. Была доработана программа, созданная в лабораторной работу № 3. Были выполнены все требования согласно варианту.

**Приложение**

**class BaseAI**

package lab.habitat;  
  
import lab.Point;  
  
import java.util.Timer;  
  
  
public abstract class BaseAI extends Thread {  
 protected static double *targetX* = 0.5;  
 protected static double *targetY* = 0.5;  
 protected Timer timer;  
 protected boolean active = true;  
  
 public void pause(boolean b) {  
 active = !b;  
 }  
}

**class AdultAI**

package lab.habitat.ai;  
  
import lab.habitat.BaseAI;  
import lab.habitat.Habitat;  
import lab.habitat.ICreature;  
import lab.habitat.creatures.birds.Adult;  
  
import java.util.Timer;  
import java.util.TimerTask;  
  
  
public class AdultAI extends BaseAI {  
 private Habitat habitat;  
 private Timer changeTargetTimer;  
  
  
 public AdultAI(Habitat h) {  
 habitat = h;  
 }  
  
 public synchronized void run() {  
 timer = new Timer();  
 timer.scheduleAtFixedRate(new TimerTask() {  
 public void run() {  
 update();  
 }  
 }, 0, 200);  
 changeTargetTimer = new Timer();  
 changeTargetTimer.scheduleAtFixedRate(new TimerTask() {  
 public void run() {  
 changeTarget();  
 }  
 }, 0, 5000);  
  
 }  
  
 protected void update() {  
 if(!active)  
 return;  
  
 for(ICreature c : habitat.getCreaturesByType(Adult.class)) {  
 if(c.getTargetX() != *targetX* && c.getTargetY() != *targetY*) {  
 double dx = c.step() \* Math.*signum*(*targetX* - c.getX());  
 double dy = c.step() \* Math.*signum*(*targetY* - c.getY());  
 c.setX(c.getX() + dx);  
 c.setY(c.getY() + dy);  
 }  
 }  
 }  
  
 protected void changeTarget() {  
 *targetX* = Math.*random*();  
 }  
}

**class ChickAI**

package lab.habitat.ai;  
  
import lab.habitat.BaseAI;  
import lab.habitat.Habitat;  
import lab.habitat.ICreature;  
import lab.habitat.creatures.birds.Chick;  
  
import java.util.Timer;  
import java.util.TimerTask;  
  
  
public class ChickAI extends BaseAI {  
 private Habitat habitat;  
  
  
 public ChickAI(Habitat h) {  
 habitat = h;  
 }  
  
  
 public synchronized void run() {  
 timer = new Timer();  
 timer.scheduleAtFixedRate(new TimerTask() {  
 public void run() {  
 update();  
 }  
 }, 0, 200);  
 }  
  
 protected void update() {  
 if(!active)  
 return;  
  
 for(ICreature c : habitat.getCreaturesByType(Chick.class)) {  
 if(c.getTargetX() != *targetX* && c.getTargetY() != *targetY*) {  
 double dx = c.step() \* Math.*signum*(*targetX* - c.getX());  
 double dy = c.step() \* Math.*signum*(*targetY* - c.getY());  
 c.setX(c.getX() + dx);  
 c.setY(c.getY() + dy);  
 }  
 }  
 }  
}

**class MainForm**

package lab.frames;  
  
import javax.sound.sampled.\*;  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.KeyEvent;  
import java.io.\*;  
import java.time.Duration;  
import java.time.Instant;  
import java.util.Timer;  
import java.util.TimerTask;  
  
import lab.Main;  
import lab.Renderer;  
import lab.frames.components.ControlBox;  
import lab.frames.components.StatusBar;  
import lab.habitat.Habitat;  
import lab.habitat.ICreature;  
import lab.habitat.ai.AdultAI;  
import lab.habitat.ai.ChickAI;  
import lab.habitat.creatures.birds.Adult;  
import lab.habitat.creatures.birds.Chick;  
  
import static java.awt.event.KeyEvent.\*;  
import static lab.Checked.\*;  
  
  
public class MainForm extends JFrame {  
 private JFileChooser fileChooser;  
  
 private JMenuBar menuBar = new JMenuBar();  
 private JMenuItem Conn1=new JMenuItem("Show Your Connections");  
 private JPanel frame = new JPanel();  
 private JPanel creatureBox = new JPanel(true);  
 private ControlBox controlBox = new ControlBox();  
 private StatusBar statusBar = new StatusBar();  
  
 private Habitat habitat;  
 private Timer timer;  
 private Instant startPoint, stopPoint, pausePoint;  
 private Duration pauseTime = Duration.*ZERO*;  
  
 private Renderer renderer;  
 private AdultAI adultAI;  
 private ChickAI chickAI;  
 private String conne;  
  
 private boolean running = false;  
 private boolean showingInfo = true;  
 private boolean adultAI\_active = true;  
 private boolean chickAI\_active = true;  
 private boolean showingStats = false;  
  
  
 public MainForm(Habitat h, Main a) {  
  
 habitat = h;  
 renderer = new Renderer(creatureBox, habitat);  
 adultAI = new AdultAI(h);  
 chickAI = new ChickAI(h);  
  
 // habitat settings  
 habitat.addCreatureCreateListener(e -> {  
 if(e instanceof Adult)  
 controlBox.cBig.setText("Количество Юр-лиц: " + h.getMetrics().getCreatureCountByType(Adult.class));  
 if(e instanceof Chick)  
 controlBox.cLit.setText("Количество Физ-лиц: " + h.getMetrics().getCreatureCountByType(Chick.class));  
 });  
 habitat.addCreatureDieListener(e -> {  
 if(e instanceof Adult)  
 controlBox.cBig.setText("Количество Юр-лиц: " + h.getMetrics().getCreatureCountByType(Adult.class));  
 if(e instanceof Chick)  
 controlBox.cLit.setText("Количество Физ-лиц: " + h.getMetrics().getCreatureCountByType(Chick.class));  
 });  
  
 KeyboardFocusManager.*getCurrentKeyboardFocusManager*().addKeyEventPostProcessor(  
 e -> {  
 if(KeyEvent.*KEY\_PRESSED* == e.getID())  
 keyPressed(e);  
 return false;  
 }  
 );  
  
 // frame settings  
 setFocusable(true);  
 setFocusTraversalKeysEnabled(false);  
 setLayout(new BorderLayout());  
 setIconImage(new ImageIcon("src/lab/assets/icon256.png").getImage());  
 setJMenuBar(menuBar);  
 add(frame, BorderLayout.*CENTER*);  
  
 // frame settings  
 frame.add(creatureBox, BorderLayout.*CENTER*);  
 frame.add(controlBox, BorderLayout.*EAST*);  
 frame.add(statusBar, BorderLayout.*SOUTH*);  
  
 // statusBar settings  
 statusBar.setStatus("Симуляция ещё не началась", Color.*RED*);  
 statusBar.setPreferredSize(new Dimension(980,20));  
  
 // creatureBox settings  
 creatureBox.setLayout(null);  
 creatureBox.setBackground(Color.*WHITE*);  
// Image img = Toolkit.getDefaultToolkit().createImage("src/img/background.png");  
  
 creatureBox.setPreferredSize(new Dimension(980,620));  
  
 // controlBox settings  
 controlBox.run.addActionListener(e -> runSimulation());  
 controlBox.pause.setEnabled(false);  
 controlBox.pause.addActionListener(e -> {  
 if(running) pauseSimulation(); else resumeSimulation();  
 });  
 controlBox.stop.setEnabled(false);  
 controlBox.stop.addActionListener(e -> stopSimulation());  
 controlBox.aiAdult.addActionListener(e -> adultAI.pause(adultAI\_active = !adultAI\_active));  
 controlBox.aiChick.addActionListener(e -> chickAI.pause(chickAI\_active = !chickAI\_active));  
 controlBox.showTimeOn.setSelected(true);  
 controlBox.showTimeOn.addActionListener(e -> controlBox.ticks.setVisible(true));  
 controlBox.showTimeOff.addActionListener(e -> controlBox.ticks.setVisible(false));  
 controlBox.adultChance.addChangeListener(e -> Adult.*setBornChance*((Integer) controlBox.adultChance.getValue() / 100.0));  
 controlBox.chickChance.addChangeListener(e -> Chick.*setBornChance*((Integer) controlBox.adultChance.getValue() / 100.0));  
 controlBox.sliderLifeBig.addChangeListener(e -> Adult.*setTTL*(controlBox.sliderLifeBig.getValue()));  
 controlBox.sliderLifeLit.addChangeListener(e -> Chick.*setTTL*(controlBox.sliderLifeLit.getValue()));  
 controlBox.showStats.addActionListener(e -> showingStats = !showingStats);  
 controlBox.lifecreaturesBox.addActionListener(e-> {  
 if (controlBox.lifecreaturesBox.getSelectedIndex() == 0) {  
 controlBox.sliderLifeLit.setVisible(true);  
 controlBox.sliderLifeBig.setVisible(false);  
 }  
  
 if (controlBox.lifecreaturesBox.getSelectedIndex() == 1) {  
 controlBox.sliderLifeLit.setVisible(false);  
 controlBox.sliderLifeBig.setVisible(true);  
 }  
 });  
  
 controlBox.current.addActionListener(e -> {  
 pauseSimulation();  
  
 StringBuffer sb = new StringBuffer();  
 var creatureTTLs = habitat.getCreaturesTTLs();  
 for (Integer time : creatureTTLs.keySet()) {  
 for (ICreature c : creatureTTLs.get(time)) {  
 sb.append(String.*format*("<p>Тип: %s, Умрет: %d</p>", c.getClass().getSimpleName(), time));  
 }  
 }  
  
 String[] options = { "Ok" };  
 JOptionPane.*showOptionDialog*(  
 this,  
 "<html>" + sb + "</html>",  
 "Текущие объекты",  
 JOptionPane.*DEFAULT\_OPTION*,  
 JOptionPane.*PLAIN\_MESSAGE*,  
 new ImageIcon("src/img/icon64.png"),  
 options,  
 options[0]  
 );  
 resumeSimulation();  
 });  
  
 // menuBar settings  
 JMenu fileMenu = new JMenu("Файл");  
  
  
  
  
 fileMenu.add(new JMenuItem("Сохранить"));  
//// {{  
// addActionListener(e -> {  
// try {  
// saveFile();  
// } catch (IOException ex) {  
// ex.printStackTrace();  
// }  
// });  
// }  
  
// private void saveFile() throws IOException {  
// fileChooser.showSaveDialog(this);  
// File selFile = fileChooser.getSelectedFile();  
// if (selFile == null) return;  
//  
// ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(selFile));  
// oos.writeLong(ticks);  
// oos.writeObject(Habitat.birds);  
// oos.close();  
// }  
// });  
//  
 fileMenu.add(new JMenuItem("Загрузить"));  
// {{  
// addActionListener(e -> {  
// try {  
// loadFile();  
// } catch (IOException | ClassNotFoundException ex) {  
// ex.printStackTrace();  
// }  
// });  
// }  
//  
// private void loadFile() throws IOException, ClassNotFoundException{  
// fileChooser.showOpenDialog(this);  
// File selFile = fileChooser.getSelectedFile();  
// if (selFile == null) return;  
// ObjectInputStream ois = new ObjectInputStream(new FileInputStream(selFile));  
// }  
// });  
  
 fileMenu.addSeparator();  
 fileMenu.add(new JMenuItem("Выйти"){{  
 addActionListener(e -> System.*exit*(0));  
 }});  
 menuBar.add(fileMenu);  
  
 JMenu actionMenu = new JMenu("Действия");  
 actionMenu.add(new JMenuItem("Запустить симуляцию"){{  
 addActionListener(*CheckedActionListener*(  
 e -> {  
 setText(running ? "Остановить симуляцию" : "Запустить симуляцию");  
 if(running) stopSimulation(); else runSimulation();  
 }  
 ));  
 }});  
 menuBar.add(actionMenu);  
  
 JMenu statMenu = new JMenu("Статистика");  
 statMenu.add(new JMenuItem("Скрыть"){{  
 addActionListener(*CheckedActionListener*(  
 e -> {  
 setText(showingInfo ? "Показать" : "Скрыть");  
 controlBox.showInfoPanel.setVisible(showingInfo = !showingInfo);  
 }  
 ));  
 }});  
 menuBar.add(statMenu);  
 JMenu show= new JMenu("Change");  
 JMenuItem connections = new JMenuItem("Change Chance of Receive");  
 JMenuItem mlg = new JMenuItem("MLGGGG1!!!1111!!!");  
  
 connections.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 JDialog dialog = createDialog3("EnterYourClient(by number of connection)", false,2, a);  
 dialog.setVisible(true);  
 }  
 });  
 mlg.addActionListener(new ActionListener() {  
 public void actionPerformed(ActionEvent e) {  
 int x=0;  
 int y=0;  
 playSound();  
 for(int i=0;i<17;i++){  
 label2(x,y);  
 if(i<5)x=x+400;  
 if(i>=5&&i<11){y=300;  
 x=x-400;}  
 if(i>=11&&i<16){y=600;  
 x=x+400;}  
 }  
 }});  
  
 menuBar.add(show);  
 show.add(connections);  
 show.add(mlg);  
  
 JMenu helpMenu = new JMenu("Помощь");  
 JMenuItem helpFile = new JMenu("Как написать лабу?");  
 helpFile.add(new JMenuItem("1 лабораторная работа"){{  
 addActionListener(*CheckedActionListener*(  
 e -> Desktop.*getDesktop*().open(new File("out/production/lab/lab/assets/1.pdf"))  
 ));  
 }});  
 helpFile.add(new JMenuItem("2 лабораторная работа"){{  
 addActionListener(*CheckedActionListener*(  
 e -> Desktop.*getDesktop*().open(new File("out/production/lab/lab/assets/2.pdf"))  
 ));  
 }});  
 helpFile.add(new JMenuItem("3 лабораторная работа"){{  
 addActionListener(*CheckedActionListener*(  
 e -> Desktop.*getDesktop*().open(new File("out/production/lab/lab/assets/3.pdf"))  
 ));  
 }});  
 helpFile.add(new JMenuItem("4 лабораторная работа"){{  
 addActionListener(*CheckedActionListener*(  
 e -> Desktop.*getDesktop*().open(new File("out/production/lab/lab/assets/4.pdf"))  
 ));  
 }});  
 helpFile.add(new JMenuItem("5 лабораторная работа"){{  
 addActionListener(*CheckedActionListener*(  
 e -> Desktop.*getDesktop*().open(new File("out/production/lab/lab/assets/5.pdf"))  
 ));  
 }});  
 helpFile.add(new JMenuItem("6 лабораторная работа"){{  
 addActionListener(*CheckedActionListener*(  
 e -> Desktop.*getDesktop*().open(new File("out/production/lab/lab/assets/6.pdf"))  
 ));  
 }});  
 helpMenu.add(helpFile);  
 menuBar.add(helpMenu);  
  
 adultAI.pause(adultAI\_active);  
 chickAI.pause(chickAI\_active);  
 renderer.run();  
 adultAI.run();  
 chickAI.run();  
 }  
 private JDialog createDialog3(String title, boolean modal, int i, Main client)  
 {  
 final JDialog[] dialog = {new JDialog(this, title, modal)};  
 dialog[0].setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 dialog[0].setSize(300, 90);  
 JTextField text = new JTextField(20);  
 JPanel contents = new JPanel(new FlowLayout(FlowLayout.*LEFT*));  
 JButton button=new JButton("Send");  
 button.addActionListener(e -> {  
 String value = text.getText();;  
 // client.Sendstr(controlBox.getChance1(),controlBox.getChance2(),value);  
 dialog[0] = (JDialog)button.getRootPane().getParent();  
 dialog[0].dispose();  
 });  
 contents.add(text);  
 contents.add(button);  
 dialog[0].add(contents);  
 return dialog[0];  
 }  
 void label2(int height,int weight) {  
 JDialog dialog = new JDialog(this,"MLGGGGG",false);  
 dialog.setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 dialog.setSize(400, 400);  
 dialog.setBounds(height,weight,400,400);  
 JLabel label2= new JLabel();  
 Image image = Toolkit.*getDefaultToolkit*().createImage("./src/lab/assets/Jaba1.gif");  
 ImageIcon imageIcon = new ImageIcon(image.getScaledInstance(400,400,Image.*SCALE\_DEFAULT*));  
 imageIcon.setImageObserver(label2);  
 label2.setIcon(imageIcon);  
 dialog.add(label2);  
 dialog.setVisible(true);  
 }  
 public void ErrorDialog5(String title, boolean modal)  
 {  
 JDialog dialog = new JDialog(this, "ErrorWindow", modal);  
 dialog.setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 dialog.setSize(300, 90);  
 JTextArea text = new JTextArea("Error: "+ title);  
 Font font = text.getFont();  
 float size = 20.0f;  
 text.setFont(font.deriveFont(size));  
 JPanel contents = new JPanel(new FlowLayout(FlowLayout.*LEFT*));  
 contents.add(text);  
 dialog.add(contents);  
 dialog.setVisible(true);  
 }  
 public void remove(){  
 //controlBox.remove();  
 }  
  
 private void runSimulation() {  
 running = true;  
 habitat.reset();  
  
 startTimer();  
 startPoint = Instant.*now*();  
  
 controlBox.run.setEnabled(false);  
 controlBox.pause.setEnabled(true);  
 controlBox.stop.setEnabled(true);  
  
 statusBar.setStatus("Симуляция запущена", Color.*GREEN*);  
 }  
  
 private void pauseSimulation() {  
 running = false;  
  
 stopTimer();  
 pausePoint = Instant.*now*();  
 adultAI.pause(true);  
 chickAI.pause(true);  
  
 statusBar.setStatus("Симуляция на паузе", Color.*YELLOW*);  
 }  
  
 public void changeChance(String value, String value1){  
 // controlBox.changeChance(value,value1);  
 }  
  
 // public void changeConn(String value){  
 //controlBox.changestr(value);  
 // }  
 void playSound(){  
 try{  
 File soundFile = new File("./src/lab/assets/MLG1.wav"); //Звуковой файл  
 AudioInputStream ais = AudioSystem.*getAudioInputStream*(soundFile);  
 Clip clip = AudioSystem.*getClip*();  
 clip.open(ais);  
 FloatControl vc = (FloatControl) clip.getControl(FloatControl.Type.*MASTER\_GAIN*);  
 vc.setValue(6);  
 clip.setFramePosition(0);  
 clip.start();  
 } catch (IOException | UnsupportedAudioFileException | LineUnavailableException exc) {  
 exc.printStackTrace();  
 }  
  
 }  
 private void resumeSimulation() {  
 running = true;  
  
 startTimer();  
 pauseTime.plus(Duration.*between*(pausePoint, Instant.*now*()));  
 adultAI.pause(adultAI\_active);  
 chickAI.pause(chickAI\_active);  
  
 statusBar.setStatus("Симуляция запущена", Color.*GREEN*);  
 }  
  
 private void stopSimulation() {  
 if(!running) pauseTime.plus(Duration.*between*(pausePoint, Instant.*now*()));  
  
 running = false;  
  
 stopTimer();  
 stopPoint = Instant.*now*();  
 controlBox.run.setEnabled(true);  
 controlBox.pause.setEnabled(false);  
 controlBox.stop.setEnabled(false);  
  
 controlBox.ticks.setText("Тик: 0");  
 controlBox.cBig.setText("Количество Юр-лиц: 0");  
 controlBox.cLit.setText("Количество Физ-лиц: 0");  
  
 statusBar.setStatus("Симуляция ещё не началась", Color.*RED*);  
  
 if(showingStats) showStatistics();  
 habitat.reset();  
 }  
  
 public void keyPressed(KeyEvent e) { // Бинд кнопок  
 switch (e.getKeyCode()) {  
 case *VK\_B*: { // Бинд кнопки "В"  
 if(!running) {  
 stopSimulation();  
 runSimulation();  
 }  
 break;  
 }  
 case *VK\_E*: { // Бинд кнопки "Е"  
 stopSimulation();  
 break;  
 }  
 case *VK\_T*: { // Бинд кнопки "Т"  
 controlBox.showInfoPanel.setVisible(showingInfo = !showingInfo);  
 break;  
 }  
 case *VK\_P*: { // Бинд кнопки "Р"  
 if(running)  
 pauseSimulation();  
 else  
 resumeSimulation();  
 break;  
 }  
 case *VK\_I*: { // Бинд кнопки "I"  
// if(showInfo.isEnabled())  
// showInfo.doClick();  
 break;  
 }  
 case *VK\_C*: { // Бинд кнопки "C"  
// if(showObjectsInfo.isEnabled())  
// showObjectsInfo.doClick();  
 break;  
 }  
 }  
 }  
  
 private void startTimer() {  
 if(timer != null)  
 timer.cancel();  
  
 timer = new Timer();  
 timer.scheduleAtFixedRate(new TimerTask() {  
 public void run() {  
 habitat.update();  
 controlBox.ticks.setText("Тик: " + habitat.getMetrics().getTime());  
 }  
 }, 0, 200);  
 }  
  
 private void stopTimer() {  
 if(timer == null)  
 return;  
  
 timer.cancel();  
 }  
  
 private void showStatistics() {  
 long secs = Duration.*between*(startPoint, stopPoint).minus(pauseTime).getSeconds();  
 StringBuffer sb = new StringBuffer();  
  
 int adultCount = habitat.getMetrics().getCreatureCountByType(Adult.class);  
 int chickCount = habitat.getMetrics().getCreatureCountByType(Chick.class);  
  
 sb.append(String.*format*("<p> Время симуляции: %dс </p>", secs));  
 sb.append(String.*format*("<p> Количество Юр-лиц: %d </p>", adultCount));  
 sb.append(String.*format*("<p> Количество Физ-лиц: %d </p>", chickCount));  
  
 String[] options = { "Ok" };  
 JOptionPane.*showOptionDialog*(  
 this,  
 "<html>" + sb + "</html>",  
 "Текущие объекты",  
 JOptionPane.*DEFAULT\_OPTION*,  
 JOptionPane.*PLAIN\_MESSAGE*,  
 new ImageIcon("src/img/icon64.png"),  
 options,  
 options[0]  
 );  
 }  
}