Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных технологий

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем

**ОТЧЕТ**

по лабораторной работе №5

по дисциплине «Объектно-ориентированное программирование»

**Оконные приложения и коллекции**

ОГУ 09.03.04.4023.980 ПЗ

|  |
| --- |
| Руководитель  канд. техн. наук, доцент  Н. А. Тишина  « » 2023 г.  Студент группы 22ПИнж(б)РПиС-2  К. Р. Безруков  « » 2023 г. |

Оренбург 2023

**Содержание**

1 Постановка задачи. Цель, задание 3

2 Практическая часть 6

3 Выводы 10

Приложение А. Исходный текст программы 11

**1 Постановка задачи. Цель, задание**

Целью лабораторной работы является изучение основных понятий и терминов обработки событий в Java, изучение коллекций и оконных приложений в языке программирования Java.

Необходимо разработать программу. Основная задача – разработка упрощенной имитации поведения объектов. Объекты реализуются через наследование.

Рабочий цикл программы:

* запускается процесс симуляции, генерируются объекты классов согласно заданию;
* симуляция завершается, выводится статистическая информация.

Для решения задачи необходимо:

* создать интерфейс IBehaviour, задающий поведение объекта;
* создать иерархию классов, определяющие объекты по варианту и реализующие интерфейс IBehaviour;
* создать класс Habitat, определяющий размер рабочей области и хранящий список объектов, с параметрами заданными вариантом. Предусмотреть в классе метод Update, вызывающийся по таймеру и получающий на вход время, прошедшее от начала симуляции. В данном методе должны генерироваться новые объекты и помещаться в поле визуализации в случайном месте. Визуализация объекта – схематично, плюсом будет, если объект будет похож на оригинал;
* рабочее окно программы – область визуализации среды обитания объектов;
* симуляция должна запускаться по клавише B и останавливаться по клавише E. При остановке симуляции список уничтожается. Время симуляции должно отображаться текстом в области визуализации и скрываться/показываться по клавише T;
* по завершению симуляции в поле визуализации должна выводиться информация о количестве и типе сгенерированных объектов, а также время симуляции. Текст должен быть форматирован, т.е. выводиться с использованием разных шрифтов и цветов.
* параметры симуляции задаются в классе Habitat.

В дальнейшем необходимо доработать программу таким образом, что:

* поделить рабочую область на 2 части. Визуализация переносится в левую часть окна, справа появляется панель управления;
* добавить кнопки «Старт» и «Стоп» в панель управления. Они должны запускать и останавливать симуляцию соответственно. Если симуляция остановлена, то кнопка «Стоп» должна блокироваться. Если симуляция идет, то блокируется кнопка «Старт». Клавиши B и E должны функционировать по-прежнему;
* добавить переключатель «Показывать информацию», который разрешает отображение модального диалога из 7 пункта задания;
* добавить группу из 2 исключающих переключателей: «Показывать время симуляции» и «Скрывать время симуляции». Клавиша T должна функционировать по-прежнему;
* используя различные менеджеры компоновки, сформировать интерфейс пользователя согласно индивидуальному заданию;
* добавить в программу главное в меню и панель инструментов, в которых продублировать основные команды вашего интерфейса пользователя;
* при остановке симуляции должно появляться модальное диалоговое окно (при условии, что оно разрешено) с информацией о количестве и типе сгенерированных объектов, а также времени симуляции. Вся информация выводится в элементе TextArea, недоступном для редактирования. В диалоговом окне должно быть 2 кнопки: «ОК» и «Отмена». При нажатии на «ОК» симуляции останавливается, а при нажатии на «Отмена», соответственно продолжается;
* предусмотреть проверку данных, вводимых пользователем. При вводе неверного значения обрабатывать исключительную ситуацию: выставлять значение по умолчанию и выводить диалоговое окно с сообщением об ошибке;

Необходимо реализовать следующие элементы управления:

* периоды рождения объектов – текстовые поля;
* для задания вероятностей рождения объектов комбобокс и список (шаг значений 10%);
* дополнить интерфейс поясняющими метками.

По итогу программу необходимо снова дополнить:

* 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;
* вынести параметры времен 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;
* организовать коллекцию сгенерированных объектов по варианту. При генерации объекта происходит добавление его в коллекцию. При обновлении времени обойдите коллекцию и удалите все объекты, время жизни которых истекло;
* добавить генерируемым объектам уникальные целочисленные идентификаторы (случайные числа), которые назначаются при генерации объекта. Для хранения сгенерированных идентификаторов используйте коллекцию удобную для поиска по варианту;
* добавить в панель управления кпопку «Текущие объекты». По нажатию на эту кнопку появляется модальное диалоговое окно, содержащее список всех «живых» объектов на момент нажатия со временем их рождения (время рождения – ключ). В класс диалогового окна должна передаваться коллекция с хранением объектов по времени рождения. Типы коллекций задаются вариантом.

Объект – кролик. Бывают 2 видов: обыкновенный и альбинос. Обыкновенные кролики рождаются каждые N1 секунд с вероятностью P1. Альбиносы рождаются каждые N2 секунд, при условии, что их количество менее K% от общего числа кроликов, в противном случае – не рождаются вовсе.

Коллекция для хранения объектов: Vector

Коллекция для хранения и поиска уникальных идентификаторов: TreeSet

Коллекция для хранения объектов по времени рождения: HashMap

**2 Практическая часть**

Иерархию классов можно представить следующим образом:

* класс «Rabbit», состоящий из полей «id», «lifeTime». Реализует интерфейс IBehaviour, состоящий из методов gitLifeTime() и createRabbit();
* класс «AlbinoRabbit», наследуемый от класса «Rabbit», переопределяющий метод createRabbit();
* класс «DefaultRabbit», наследуемый от класса «Rabbit», переопределяющий метод createRabbit();

На рисунке 1 приведена диаграмма классов.

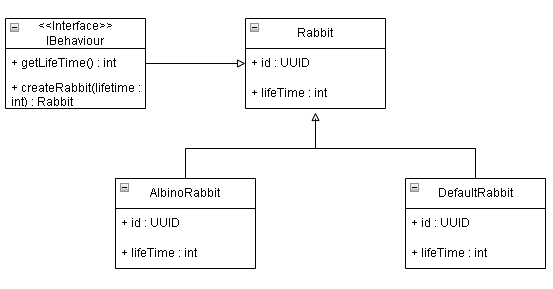


Рисунок 1 – Диаграмма классов предметной области «Кролики»

При запуске программы отображается окно симуляции, приведенное на рисунке 2. Оно состоит из текстовой метки таймера, панели инструментов (с кнопками старта, окончания симуляции и показа объектов), панели отображения объектов, текстовых полей (периоды создания объектов, время жизни объектов), выпадающие списки вероятностей, кнопки «Старт» и «Стоп», кнопка-переключатель отображения модальных окон, переключатель показа таймера (включен по умолчанию).

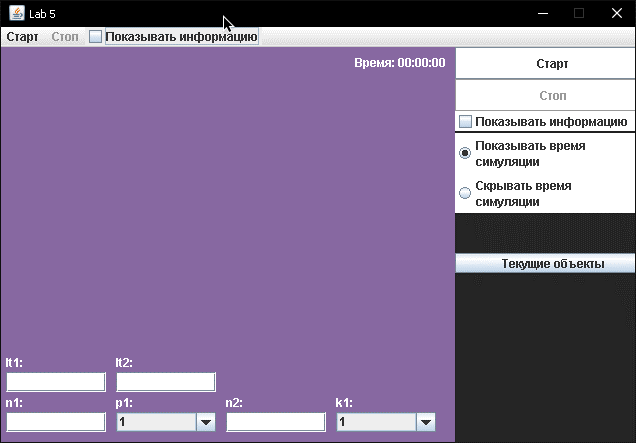


Рисунок 2 – Окно симуляции

Чтобы запустить процесс симуляции, надо ввести время порождения объектов, выбрать вероятности их создания, ввести периоды их жизни, и нажать кнопку «Старт» или клавишу B, после чего начнется процесс симуляции – если ничего не ввести в текстовые поля, будут приняты значения по умолчанию (5 секунд периода рождения для студентов, 2 секунды рождения объекта для студенток, 10 секунд времени студентов, 4 секунд времени жизни студенток). По истечении времени жизни объекты уничтожаются. Процесс симуляции приведен на рисунке 3.

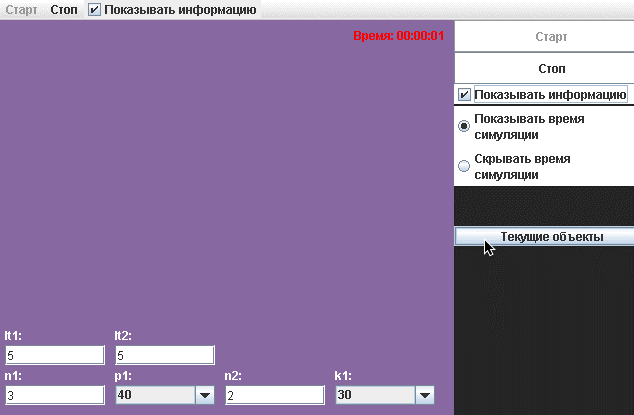


Рисунок 3 – Процесс симуляции

Переключение переключателя отображения таймера и нажатие клавиши T показывает или скрывает таймер (нажатие клавиши T отображаются на переключателе), а кнопка переключения отображения модальных окон имеет два режиме – «вкл.» и «выкл.» что показано на рисунке 4.

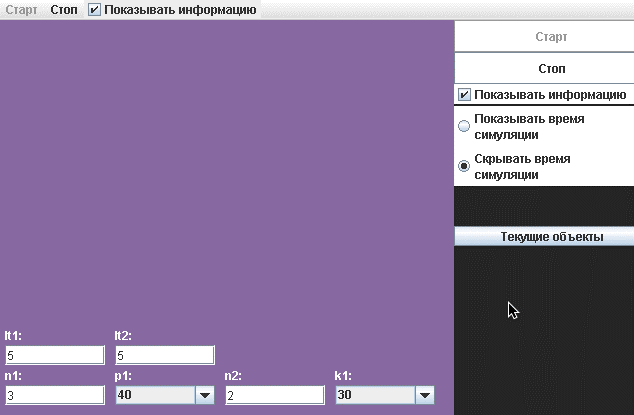


Рисунок 4 – Управление симуляцией

По нажатию клавиши E или одной из кнопок «Стоп» откроется диалоговое окно с выводом информации о количестве созданных объектов и прошедшем времени (если разрешен показ модальных окон), а сама симуляция будет поставлена на паузу. Если показ модальных окон запрещен – процесс симуляции завершится. На рисунке 5 приведено модальное окно, отображающее количество созданных объектов.

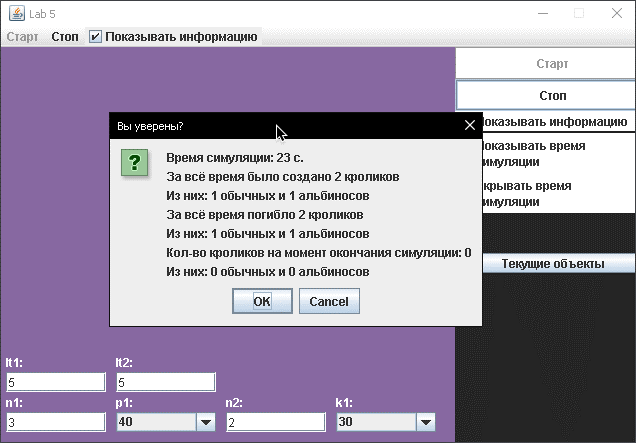


Рисунок 5– Модальное окно отображения основной информации

Если нажать кнопку «Ок», процесс симуляции завершится, а если кнопку «Отмена» - продолжится. При нажатии кнопки «Показать текущие объекты» (доступна при разрешенных модальных окнах), откроется модальное окно со списком объектов каждого типа (симуляция при этом будет поставлена паузу, если она не была остановлена), приведенное на рисунке 6.

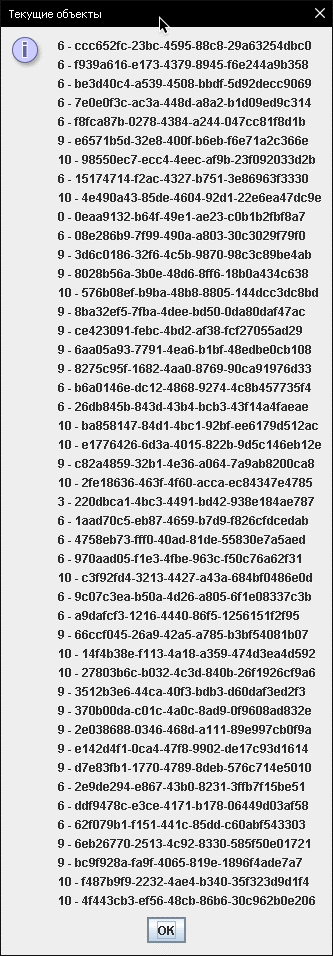


Рисунок 6 – Список созданных объектов

Если нажать кнопку «Отмена» (или закрыть окно), симуляция продолжится, если на кнопку «Ок» - симуляция остановится. На рисунке 7 приведено окно симуляции после завершения симуляции.

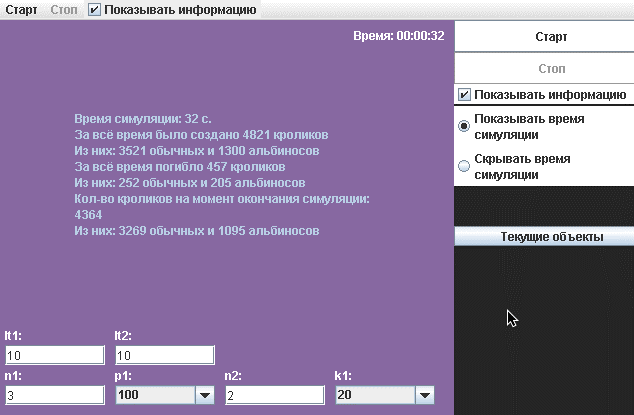


Рисунок 7 – Завершенная симуляция

Исходный текст программы приведен в приложении А.

**3 Выводы**

В ходе лабораторной работы были получены навыки работы с оконными приложениями, коллекциями и интерфейсами.

В ходе задачи был создан интерфейс поведения объектов, иерархия классов для, поведение которых симулировалось при работе программы.

Был разработан отчёт по лабораторной работе, содержащий следующие пункты:

* постановка задачи;
* практическая часть;
* исходный текст программы (приведён в приложении А);
* выводы.

**Приложение А**

***(обязательное)***

**Исходный текст программы**

public class Main {  
 public static void main(String[] args) {  
 new AdvancedJFrame(650, 450);  
  
 }  
}

package classes;  
  
public interface IBehaviour {  
 Rabbit createRabbit(int lifeTime);  
 int getLifeTime();  
}

package classes;  
  
import java.util.UUID;  
  
public abstract class Rabbit implements IBehaviour {  
 private final int lifeTime;  
 private final UUID id;  
  
 public Rabbit(int lifeTime) {  
 this.lifeTime = lifeTime;  
 id = UUID.randomUUID();  
 }  
 public UUID getId() {  
 return id;  
 }  
  
 @Override  
 public int getLifeTime() {  
 return lifeTime;  
 }  
}

package classes;  
  
public class DefaultRabbit extends Rabbit {  
 public DefaultRabbit(int lifeTime) {  
 super(lifeTime);  
 }  
  
 @Override  
 public Rabbit createRabbit(int lifeTime) {  
 return new DefaultRabbit(lifeTime);  
 }  
}

package classes;  
  
public class AlbinoRabbit extends Rabbit {  
 public AlbinoRabbit(int lifeTime) {  
 super(lifeTime);  
 }  
  
 @Override  
 public Rabbit createRabbit(int lifeTime) {  
 return new AlbinoRabbit(lifeTime);  
 }  
}

package classes;  
  
import java.util.\*;  
import java.util.stream.Collectors;  
  
public class Habitat {  
 public Vector<Rabbit> rabbits;  
 public TreeSet<Rabbit> rabbitsSet;  
 public HashMap<Rabbit, Integer> rabbitsMap;  
 private final Random random = new Random();  
 private boolean isStop;  
 private long startTime;  
 private double duration;  
 private final int periodToBirthForDefaultRabbits;  
 private final float probabilityToBirthForDefaultRabbits;  
 private final int periodToBirthForAlbinoRabbits;  
 private final int conditionForAlbinoRabbits;  
 private final int lifeTimeForDefaultRabbits;  
 private final int lifeTimeForAlbinoRabbits;  
 private int countCreatedDefault;  
 private int countCreatedAlbino;  
 private int countDeathDefault;  
 private int countDeathAlbino;  
 private String report;  
  
 public Habitat(int periodToBirthForDefaultRabbits, float probabilityToBirthForDefaultRabbits, int periodToBirthForAlbinoRabbits, int conditionForAlbinoRabbits, int lifeTimeForDefaultRabbits, int lifeTimeForAlbinoRabbits) {  
 rabbits = new Vector<>();  
 rabbitsSet = new TreeSet<>(Comparator.comparing(Rabbit::getId));  
 rabbitsMap = new HashMap<>();  
 this.conditionForAlbinoRabbits = conditionForAlbinoRabbits;  
 this.periodToBirthForAlbinoRabbits = periodToBirthForAlbinoRabbits;  
 this.periodToBirthForDefaultRabbits = periodToBirthForDefaultRabbits;  
 this.probabilityToBirthForDefaultRabbits = probabilityToBirthForDefaultRabbits;  
 this.lifeTimeForDefaultRabbits = lifeTimeForDefaultRabbits;  
 this.lifeTimeForAlbinoRabbits = lifeTimeForAlbinoRabbits;  
 isStop = true;  
 countCreatedDefault = 0;  
 countDeathDefault = 0;  
 countCreatedAlbino = 0;  
 countDeathAlbino = 0;  
 }  
 public int getDuration() {  
 return (int)duration;  
 }  
 private Vector<DefaultRabbit> getDefaultRabbits() {  
 return rabbits.stream()  
 .filter(DefaultRabbit.class::isInstance)  
 .map(DefaultRabbit.class::cast)  
 .collect(Collectors.toCollection(Vector::new));  
 }  
  
 private Vector<AlbinoRabbit> getAlbinoRabbits() {  
 return rabbits.stream()  
 .filter(AlbinoRabbit.class::isInstance)  
 .map(AlbinoRabbit.class::cast)  
 .collect(Collectors.toCollection(Vector::new));  
 }  
  
 public void Start() {  
 isStop = false;  
 startTime = System.currentTimeMillis();  
 rabbits.add(new DefaultRabbit(lifeTimeForDefaultRabbits));  
 rabbits.add(new AlbinoRabbit(lifeTimeForAlbinoRabbits));  
 rabbits.forEach(rabbit -> rabbitsMap.put(rabbit, 0));  
 countCreatedDefault = 1;  
 countDeathDefault = 0;  
 countCreatedAlbino = 1;  
 countDeathAlbino = 0;  
 }  
  
 public void Update() {  
 duration = (double)(System.currentTimeMillis() - startTime) / 1000;  
 if(duration % periodToBirthForDefaultRabbits == 0) {  
 getDefaultRabbits().forEach(rabbit -> {  
 if (random.nextFloat() <= probabilityToBirthForDefaultRabbits) {  
 Rabbit daughter = rabbit.createRabbit(lifeTimeForDefaultRabbits);  
 rabbits.add(daughter);  
 rabbitsSet.add(daughter);  
 rabbitsMap.put(daughter, (int)duration);  
 countCreatedDefault++;  
 }  
 if(rabbit.getLifeTime() + rabbitsMap.get(rabbit) <= duration) {  
 rabbitsMap.remove(rabbit);  
 rabbitsSet.remove(rabbit);  
 rabbits.remove(rabbit);  
 countDeathDefault++;  
 }  
 });  
 }  
 if(duration % periodToBirthForAlbinoRabbits == 0) {  
 Vector<AlbinoRabbit> albinoRabbits = getAlbinoRabbits();  
 albinoRabbits.forEach(rabbit -> {  
 if (((float) albinoRabbits.size() / rabbits.size()) < ((float) conditionForAlbinoRabbits / 100)) {  
 Rabbit daughter = rabbit.createRabbit(lifeTimeForAlbinoRabbits);  
 rabbits.add(daughter);  
 rabbitsSet.add(daughter);  
 rabbitsMap.put(daughter, (int)duration);  
 countCreatedAlbino++;  
 }  
 if(rabbit.getLifeTime() + rabbitsMap.get(rabbit) <= duration) {  
 rabbitsMap.remove(rabbit);  
 rabbitsSet.remove(rabbit);  
 rabbits.remove(rabbit);  
 countDeathAlbino++;  
 }  
 });  
 }  
 }  
 public void Stop() {  
 isStop = true;  
 long endTime = System.currentTimeMillis();  
 duration = (double)(endTime - startTime)/1000;  
 report = this.toString();  
 rabbits = new Vector<>();  
 rabbitsSet = new TreeSet<>(Comparator.comparing(Rabbit::getId));  
 rabbitsMap = new HashMap<>();  
 }  
  
 @Override  
 public String toString() {  
 return ("Время симуляции: %s с.\nЗа всё время было создано %s кроликов\nИз них: %s обычных и %s альбиносов\n" +  
 "За всё время погибло %s кроликов\nИз них: %s обычных и %s альбиносов\n" +  
 "Кол-во кроликов на момент окончания симуляции: %s\nИз них: %s обычных и %s альбиносов").formatted((int)duration, countCreatedDefault+ countCreatedAlbino,  
 countCreatedDefault, countCreatedAlbino, countDeathDefault+ countDeathAlbino, countDeathDefault, countDeathAlbino, rabbits.size(), getDefaultRabbits().size(),  
 getAlbinoRabbits().size());  
 }  
  
 public boolean isStop() {  
 return isStop;  
 }  
  
 public String getReport() {  
 return report;  
 }  
}

import classes.Habitat;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.awt.event.MouseAdapter;  
import java.awt.event.MouseEvent;  
import java.time.LocalTime;  
import java.util.stream.Collectors;  
  
public class AdvancedJFrame extends JFrame {  
 private final JMenuBar menuBar;  
 private final JMenu startMenuButton;  
 private final JMenu stopMenuButton;  
 private final JPanel mainPanel;  
 private final JPanel menuPanel;  
 private final JButton startButton;  
 private final JButton stopButton;  
 private final JCheckBox showInfoCBMenu;  
 private final JCheckBox showInfoCBPanel;  
 private final JLabel timerLabel;  
 private final JRadioButton showTimerRB;  
 private final JRadioButton hideTimerRB;  
 private final JTextArea reportArea;  
 private final JTextField n1;  
 private final JComboBox p1;  
 private final JTextField n2;  
 private final JComboBox k1;  
 private final JTextField lt1;  
 private final JTextField lt2;  
 private Habitat habitat;  
 private final MouseAdapter startClick = new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 if(startButton.isEnabled())  
 start();  
 }  
 };  
 private final MouseAdapter stopClick = new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 if(stopButton.isEnabled())  
 stop();  
 }  
 };  
 private final KeyAdapter keyHandler = new KeyAdapter() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 if(e.getKeyChar() == 't' || e.getKeyChar() == 'T' || e.getKeyChar() == 'е' || e.getKeyChar() == 'Е') {  
 switchTimerVisibility();  
 } else if (e.getKeyChar() == 'b' || e.getKeyChar() == 'B' || e.getKeyChar() == 'и' || e.getKeyChar() == 'И') {  
 if(!stopButton.isEnabled())  
 start();  
 } else if (e.getKeyChar() == 'e' || e.getKeyChar() == 'E' || e.getKeyChar() == 'у' || e.getKeyChar() == 'У') {  
 if(!startButton.isEnabled())  
 stop();  
 }  
 }  
 };  
  
 private final KeyAdapter onlyDigital = new KeyAdapter() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 if(!Character.isDigit(e.getKeyChar()))  
 e.consume();  
 }  
 };  
  
 public AdvancedJFrame(int width, int height) {  
 this.setTitle("Lab 5");  
 this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 Toolkit toolkit = Toolkit.getDefaultToolkit();  
 Dimension dimension = toolkit.getScreenSize();  
 this.setBounds(dimension.width/2 - width/2, dimension.height/2 - height/2, width, height);  
 this.setLayout(null);  
 this.setResizable(false);  
  
 menuBar = new JMenuBar();  
 startMenuButton = new JMenu("Старт");  
 startMenuButton.addMouseListener(startClick);  
 stopMenuButton = new JMenu("Стоп");  
 stopMenuButton.addMouseListener(stopClick);  
 stopMenuButton.setEnabled(false);  
 showInfoCBMenu = new JCheckBox("Показывать информацию");  
 showInfoCBPanel = new JCheckBox("Показывать информацию");  
 showInfoCBMenu.addItemListener(e -> showInfoCBPanel.setSelected(showInfoCBMenu.isSelected()));  
 menuBar.add(startMenuButton);  
 menuBar.add(stopMenuButton);  
 menuBar.add(showInfoCBMenu);  
 menuBar.setBounds(0,0, width, 20);  
  
 mainPanel = new JPanel();  
 mainPanel.setBounds(0,20, (int)(width\*0.7), height-20);  
 mainPanel.setOpaque(true);  
 mainPanel.setBackground(Color.decode("#8768a1"));  
 mainPanel.setLayout(null);  
 mainPanel.addKeyListener(keyHandler);  
 mainPanel.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 mainPanel.requestFocus();  
 }  
 });  
  
 menuPanel = new JPanel();  
 menuPanel.setBounds(mainPanel.getWidth(), 20, width-mainPanel.getWidth(), height-20);  
 menuPanel.setOpaque(true);  
 menuPanel.setBackground(Color.decode("#252525"));  
 menuPanel.setLayout(null);  
 menuPanel.addKeyListener(keyHandler);  
 menuPanel.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 menuPanel.requestFocus();  
 }  
 });  
  
 timerLabel = new JLabel("Время: 00:00:00");  
 timerLabel.setBounds(mainPanel.getWidth()-100, 5, 100, 20);  
 timerLabel.setForeground(Color.WHITE);  
  
 reportArea = new JTextArea();  
 reportArea.setEnabled(false);  
 reportArea.setBounds((int) (mainPanel.getWidth()/2 - mainPanel.getWidth()/(1.5\*2)), mainPanel.getHeight()/2 - 125, (int) (mainPanel.getWidth()/1.5), 200);  
 reportArea.setOpaque(true);  
 reportArea.setForeground(Color.WHITE);  
 reportArea.setBackground(Color.decode("#8768a1"));  
 reportArea.setFont(new Font(reportArea.getFont().getFontName(), Font.BOLD, 12));  
 reportArea.setWrapStyleWord(true);  
 reportArea.setLineWrap(true);  
  
 n1 = new JTextField();  
 n1.setBounds(5, mainPanel.getHeight()-65, 100, 20);  
 n1.addKeyListener(onlyDigital);  
 n1.setToolTipText("Период времени для рождения обычных кроликов");  
 JLabel n1Label = new JLabel("n1:");  
 n1Label.setBounds(5, mainPanel.getHeight()-85, 100, 20);  
 n1Label.setForeground(Color.WHITE);  
  
 p1 = new JComboBox(new Integer[]{1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100});  
 p1.setBounds(115, mainPanel.getHeight()-65, 100, 20);  
 p1.setToolTipText("Вероятность рождения обычных кроликов");  
 JLabel p1Label = new JLabel("p1:");  
 p1Label.setBounds(115, mainPanel.getHeight()-85, 100, 20);  
 p1Label.setForeground(Color.WHITE);  
  
 n2 = new JTextField();  
 n2.setBounds(225, mainPanel.getHeight()-65, 100, 20);  
 n2.addKeyListener(onlyDigital);  
 n2.setToolTipText("Период времени для рождения кроликов альбиносов");  
 JLabel n2Label = new JLabel("n2:");  
 n2Label.setBounds(225, mainPanel.getHeight()-85, 100, 20);  
 n2Label.setForeground(Color.WHITE);  
  
 k1 = new JComboBox(new Integer[]{1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100});  
 k1.setBounds(335, mainPanel.getHeight()-65, 100, 20);  
 k1.setToolTipText("Максимальная часть кроликов альбиносов от общего кол-ва для рождения");  
 JLabel k1Label = new JLabel("k1:");  
 k1Label.setBounds(335, mainPanel.getHeight()-85, 100, 20);  
 k1Label.setForeground(Color.WHITE);  
  
 lt1 = new JTextField();  
 lt1.setBounds(5, mainPanel.getHeight()-105, 100, 20);  
 lt1.addKeyListener(onlyDigital);  
 lt1.setToolTipText("Время жизни обычных кроликов");  
 JLabel lt1Label = new JLabel("lt1:");  
 lt1Label.setBounds(5, mainPanel.getHeight()-125, 100, 20);  
 lt1Label.setForeground(Color.WHITE);  
  
 lt2 = new JTextField();  
 lt2.setBounds(115, mainPanel.getHeight()-105, 100, 20);  
 lt2.addKeyListener(onlyDigital);  
 lt2.setToolTipText("Время жизни кроликов альбиносов");  
 JLabel lt2Label = new JLabel("lt2:");  
 lt2Label.setBounds(115, mainPanel.getHeight()-125, 100, 20);  
 lt2Label.setForeground(Color.WHITE);  
  
 startButton = new JButton("Старт");  
 startButton.addMouseListener(startClick);  
 startButton.setBounds(0,0, menuPanel.getWidth(), 32);  
 startButton.setOpaque(true);  
 startButton.setBackground(Color.white);  
  
 stopButton = new JButton("Стоп");  
 stopButton.addMouseListener(stopClick);  
 stopButton.setBounds(0,32, menuPanel.getWidth(), 32);  
 stopButton.setOpaque(true);  
 stopButton.setBackground(Color.white);  
 stopButton.setEnabled(false);  
  
 showInfoCBPanel.addItemListener(e -> showInfoCBMenu.setSelected(showInfoCBPanel.isSelected()));  
 showInfoCBPanel.setBounds(0, 64, menuPanel.getWidth(), 20);  
 showInfoCBPanel.setOpaque(true);  
 showInfoCBPanel.setBackground(Color.white);  
  
 showTimerRB = new JRadioButton("<html>Показывать время симуляции</html>");  
 showTimerRB.setBounds(0, 86, menuPanel.getWidth(), 40);  
 showTimerRB.setSelected(true);  
 showTimerRB.setOpaque(true);  
 showTimerRB.setBackground(Color.white);  
 showTimerRB.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 switchTimerVisibility();  
 }  
 });  
 hideTimerRB = new JRadioButton("<html>Скрывать время симуляции</html>");  
 hideTimerRB.setBounds(0, 126, menuPanel.getWidth()-10, 40);  
 hideTimerRB.setOpaque(true);  
 hideTimerRB.setBackground(Color.white);  
 hideTimerRB.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 switchTimerVisibility();  
 }  
 });  
  
 JButton currentObjectsButton = new JButton("Текущие объекты");  
 currentObjectsButton.setBounds(0, 206, menuPanel.getWidth(), 20);  
 currentObjectsButton.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 if(habitat != null && !habitat.isStop())  
 JOptionPane.showMessageDialog(null, habitat.rabbitsMap.keySet().stream().map(rabbit -> "%s - %s".formatted(habitat.rabbitsMap.get(rabbit), rabbit.getId())).collect(Collectors.joining("\n")), "Текущие объекты", JOptionPane.INFORMATION\_MESSAGE);  
 }  
 });  
  
 mainPanel.add(timerLabel);  
 mainPanel.add(reportArea);  
 mainPanel.add(n1);  
 mainPanel.add(n1Label);  
 mainPanel.add(p1);  
 mainPanel.add(p1Label);  
 mainPanel.add(n2);  
 mainPanel.add(n2Label);  
 mainPanel.add(k1);  
 mainPanel.add(k1Label);  
 mainPanel.add(lt1);  
 mainPanel.add(lt1Label);  
 mainPanel.add(lt2);  
 mainPanel.add(lt2Label);  
  
 menuPanel.add(startButton);  
 menuPanel.add(stopButton);  
 menuPanel.add(showInfoCBPanel);  
 menuPanel.add(showTimerRB);  
 menuPanel.add(hideTimerRB);  
 menuPanel.add(currentObjectsButton);  
  
 this.add(menuBar);  
 this.add(mainPanel);  
 this.add(menuPanel);  
 this.setVisible(true);  
 }  
  
 private void switchTimerVisibility() {  
 timerLabel.setVisible(!timerLabel.isVisible());  
 showTimerRB.setSelected(timerLabel.isVisible());  
 hideTimerRB.setSelected(!timerLabel.isVisible());  
 }  
  
 private void start() {  
 if(n1.getText().isEmpty() || n2.getText().isEmpty() || lt1.getText().isEmpty() || lt2.getText().isEmpty()) {  
 JOptionPane.showMessageDialog(this, "Заполнить поля!", "Внимание", JOptionPane.WARNING\_MESSAGE);  
 return;  
 }  
 reportArea.setText("");  
 timerLabel.setForeground(Color.RED);  
 startButton.setEnabled(false);  
 startMenuButton.setEnabled(false);  
 stopButton.setEnabled(true);  
 stopMenuButton.setEnabled(true);  
 habitat = new Habitat(Integer.parseInt(n1.getText()), ((float)((Integer)p1.getSelectedItem())/100), Integer.parseInt(n2.getText()), (Integer) k1.getSelectedItem()  
 ,Integer.parseInt(lt1.getText()), Integer.parseInt(lt2.getText()));  
 habitat.Start();  
 Thread thread = new Thread(() -> {  
 while(!habitat.isStop()) {  
 habitat.Update();  
 timerLabel.setText("Время: %s".formatted(habitat.getDuration() == 0 ? "00:00:00" : LocalTime.of(0,0,habitat.getDuration())));  
 }  
 });  
 thread.start();  
 }  
  
 private void stop() {  
 if(showInfoCBMenu.isSelected()) {  
 if(JOptionPane.showConfirmDialog(this, habitat.toString(), "Вы уверены?", JOptionPane.OK\_CANCEL\_OPTION) == 2)  
 return;  
 }  
 timerLabel.setForeground(Color.WHITE);  
 startButton.setEnabled(true);  
 startMenuButton.setEnabled(true);  
 stopButton.setEnabled(false);  
 stopMenuButton.setEnabled(false);  
 habitat.Stop();  
 reportArea.setText(habitat.getReport());  
 }  
}