МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

*Лабораторная работа №2 по дисциплине «Технология программирования»*

*Вариант 2*

Группа: АВТ-610

Студенты: Князев Р.И., Дунаев Н.Ю.

Преподаватель: Васюткина И. А.

Новосибирск 2018

**Темы:**

Разработка графического интерфейса приложения.

**Задание:**

1. Познакомиться с основными графическими библиотеками Java - AWT и Swing и их основными компонентами. Изучить классы менеджеров компоновки, классы создания меню приложения, основных и диалоговых окон и т.д.

2. Для хранения генерируемых объектов использовать динамический массив объектов.

Доработать программу, созданную в лабораторной работе № 1:

1) поделить рабочую область окна приложения на 2 части. Визуализация переносится в одну часть окна, панель управления в другую;

2) 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;

3) добавить переключатель «Показывать информацию», который разрешает отображение модального диалога из 7 пункта задания;

4) добавить группу из 2 исключающих переключателей: «Показывать время симуляции» и «Скрывать время симуляции». Клавиша T должна функционировать по-прежнему;

5) используя различные менеджеры компоновки реализовать интерфейс пользователя согласно индивидуальному заданию;

6) добавить в программу главное меню и панель инструментов, в которых продублировать основные команды интерфейса пользователя;

7) 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 полностью останавливается, а при нажатии на «Отмена», соответственно продолжается;

8) предусмотреть проверку данных вводимых пользователем. При вводе неверного значения обрабатывать исключительную ситуацию: выставлять значение по умолчанию и выводить диалоговое окно с сообщением об ошибке;

9) Реализовать следующие элементы управления:

- Периоды рождения объектов – текстовые поля;

- Для задания вероятностей рождения объектов - JComboBox и JSlider (шаг значений

10) Дополнить интерфейс поясняющими метками.

***Описание структуры классов:***

Для реализации программы были изменены следующие классы:

* Habitat
* Window

**Habitat** *–* интерфейс, содержащий следующие методы:

**public void focusDisable(Window myWind)**– убирает возможность сфокурсироваться на какой-либо кнопке

**public void startSimulation()**– начало симуляции

**public void stopSimulation()**– конец симуляции

**Habitat()**– конструктор класса Habitat

**Window** – класс окна, в нем же добавляются всевозможные компоненты на это окно, изменены следующие методы:

**public void paint\_Statictics()**– отрисовка окна статистики, если она разрешена

**Window()**- конструктор класса Window

***Описание функционала программы:***

Добавленные элементы графического интерфейса, пользуясь уже существующими методами для управления процессом симуляции, наравне с клавишами контролируют процесс симуляции, так же используя механизм обработки событий. Для проверки верности введенных данных используется механизм обработки исключений.

public class Habitat {

// показывается ли время

private boolean showTime=true;

// показывается ли информация

private boolean showInfo = false;

//пауза или нет:?

private boolean isPaused = false;

public Habitat(Window myWind) {

//Некоторые компоненты класса окна перенесли в среду для работы с компонентами из класса среды

JPanel panel = myWind.panel;

JButton start = myWind.start;

JButton stop = myWind.stop;

JRadioButton show = myWind.show;

JRadioButton hide = myWind.hide;

JRadioButton infoShow = myWind.infoShow;

JButton ok = myWind.ok;

JButton cansel = myWind.cansel;

JDialog dialog = myWind.dialog;

JSlider slide = myWind.slide;

JSlider slide2 = myWind.slide2;

JTextField workTim = myWind.workerCreate;

JTextField trutTim = myWind.trutenCreate;

JButton workerbut = myWind.workerbut;

JButton trutbut = myWind.trutbut;

JDialog excep = myWind.excep;

JLabel lab1 = new JLabel(" Введены некорректные данные!");

JButton but1 = new JButton("ОК");

lab1.setVisible(true);

show.setEnabled(true);

tim.schedule(new Updater(habit), 1000, 100);

gr = wind.getGraphics();

//на кнопки добавили слушатели событий

trutbut.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

timeSpawnTruten = Integer.parseInt(trutTim.getText());

}catch (Exception ex){

timeSpawnTruten = 5;

excep.setLayout(new BorderLayout());

JPanel pan1 = new JPanel();

excep.setSize(230,90);

excep.setLocation(550,300);

excep.add(pan1);

pan1.add(lab1, BorderLayout.NORTH);

pan1.add(but1, BorderLayout.SOUTH);

excep.setVisible(true);

pan1.setVisible(true);

pan1.setFocusable(true);

}

trutTim.setText("");

}

});

but1.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

excep.dispose();

}

});

workerbut.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

timeSpawnWorker = Integer.parseInt(workTim.getText());

}catch (Exception ex){

timeSpawnWorker = 3;

excep.setLayout(new BorderLayout());

JPanel pan1 = new JPanel();

excep.setSize(230,90);

excep.setLocation(550,300);

excep.add(pan1);

pan1.add(lab1, BorderLayout.NORTH);

pan1.add(but1, BorderLayout.SOUTH);

excep.setVisible(true);

}

workTim.setText("");

}

});

slide2.addChangeListener(new ChangeListener() {

@Override

public void stateChanged(ChangeEvent e) {

JSlider source = (JSlider) e.getSource();

trutenPersent = (float)source.getValue()/100;

}

});

slide.addChangeListener(new ChangeListener() {

@Override

public void stateChanged(ChangeEvent e) {

JSlider source = (JSlider) e.getSource();

workerCreateChance = (float)source.getValue()/100;

}

});

ok.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

System.exit(0);

}

});

cansel.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

wind.setSimulationState(true);

dialog.dispose();

focusDisable(wind);

startSimulation(wind);

}

});

infoShow.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

showInfo = !showInfo;

focusDisable(wind);

wind.showInfo = showInfo;

}

});

show.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (!showTime) wind.changeTimeState(true);

showTime=true;

focusDisable(wind);

}

});

hide.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if(showTime) wind.changeTimeState(false);

showTime=false;

focusDisable(wind);

}

});

start.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if(!isStarted) {

startSimulation(wind);

}

}

});

stop.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (isStarted)

stopSimulation(wind);

}

});

wind.addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyCode.B.getCode())

{

if (!isStarted) {

startSimulation(wind);

}

}

if (e.getKeyCode() == KeyCode.E.getCode())

{

if (isStarted) {

stopSimulation(wind);

}

}

if (e.getKeyCode() == KeyCode.T.getCode())

{

showTime=!showTime;

wind.changeTimeState(showTime);

}

}

});

}

//начало симуляции

public void startSimulation(Window myWind)

{

wind.setMas(mas);

isStarted = true;

wind.setSimulationState(isStarted);

myWind.start.setEnabled(false);

myWind.stop.setEnabled(true);

focusDisable(wind);

isPaused = false;

}

//остановить симуляцию

public void stopSimulation(Window myWind)

{

isStarted = false;

wind.setSimulationState(isStarted);

wind.paint(gr);

focusDisable(wind);

isPaused = true;

}

//убрать фокус с кнопок

public void focusDisable(Window myWind)

{

myWind.start.setFocusable(false);

myWind.stop.setFocusable(false);

myWind.hide.setFocusable(false);

myWind.show.setFocusable(false);

myWind.infoShow.setFocusable(false);

myWind.ok.setFocusable(false);

myWind.cansel.setFocusable(false);

}

//Изменение: если стоит пауза, к elapsed не прибавляется время

public void update(double elapsed)

{

if (isStarted)

{

secForTrut += elapsed;

secForWorker += elapsed;

time += elapsed;

if (secForTrut > timeSpawnTruten) {

if ((float) BeeTruten.trNumber / Bee.n < trutenPersent) {

mas.add(new BeeTruten((int)(Math.random() \* 720), 40 + (int)(Math.random() \* 380)));

}

secForTrut -= timeSpawnTruten;

}

if (secForWorker > timeSpawnWorker) {

if (Math.random() >= (1 - workerCreateChance)) {

mas.add(new BeeWorker((int) (Math.random() \* 720),40 + (int)(Math.random() \* 380)));

}

secForWorker -= timeSpawnWorker;

}

wind.setTextTime(time);

}

wind.paint(gr);

}

}

public class Window extends JFrame {

ArrayList<Bee> mas; // Заменили статический массив на ArrayList

JButton start = new JButton("Старт");

//панель для компановки кнопок

static JPanel panel = new JPanel();

//добавили кнопок для реализации интерфейса, радио кнопки объединили в группы, добавили чекбокс показывать информацию

JButton stop = new JButton("Стоп");

JRadioButton show = new JRadioButton("Показывать время");

JRadioButton hide = new JRadioButton("Скрывать время");

ButtonGroup group = new ButtonGroup();

JRadioButton infoShow = new JRadioButton("Показывать информацию");

JButton ok = new JButton("Ок");

JButton cansel = new JButton("Отмена");

JDialog dialog = new JDialog(this, "Информация", true);

JPanel pan = new JPanel();

TextArea txt = new TextArea(10,10);

//добавили ползунки для задания вертоятностей рождения пчелок

JSlider slide = new JSlider(JSlider.HORIZONTAL,0,100,0);

JSlider slide2 = new JSlider(JSlider.HORIZONTAL, 0, 100, 0);

JLabel infoAboutWorker = new JLabel("Вероятность рождения рабочего");

JLabel getInfoAboutTruten = new JLabel("Допустимая доля трутней");

JLabel slidLab = new JLabel();

JLabel empty = new JLabel(" ");

JLabel empty1 = new JLabel(" ");

//тектсовые поля для ввода информации о периодах рождения пчел

JLabel timeWorkerInfo = new JLabel("Время рождения рабочего");

JLabel timeTrutenInfo = new JLabel("Время рождения трутня");

JTextField workerCreate = new JTextField(4);

JTextField trutenCreate = new JTextField(4);

JButton workerbut = new JButton();

JButton trutbut = new JButton();

JDialog excep = new JDialog(this, "",true);

Window() {

//на фрейм добавили панель. На панель добавили все остальные компоненты. На фрейм добавили меню, на меню панель инструментов.

panel.setPreferredSize(new Dimension(270,500));

panel.add(start);

panel.add(stop);

group.add(show);

group.add(hide);

setFocusable(true);

panel.add(show);

panel.add(hide);

panel.add(infoShow);

show.doClick();

//worker chance

panel.add(empty);

panel.add(infoAboutWorker);

slidLab.setOpaque(true);

slide.setMajorTickSpacing(10);

slide.setMinorTickSpacing(0);

slide.setPaintLabels(true);

slide.setPaintTicks(true);

panel.add(slidLab);

panel.add(slide);

//truten chance

panel.add(empty);

panel.add(getInfoAboutTruten);

slidLab.setOpaque(true);

slide2.setMajorTickSpacing(10);

slide2.setMinorTickSpacing(0);

slide2.setPaintLabels(true);

slide2.setPaintTicks(true);

panel.add(slidLab);

panel.add(slide2);

//worker create

workerbut.setPreferredSize(new Dimension(20,20));

panel.add(empty1);

panel.add(timeWorkerInfo);

panel.add(workerCreate);

panel.add(workerbut);

//truten create

trutbut.setPreferredSize(new Dimension(20,20));

panel.add(empty);

panel.add(timeTrutenInfo);

panel.add(trutenCreate);

panel.add(trutbut);

}

public void setMas(ArrayList<Bee> mas) // Массив заменен на ArrayList

{

this.mas = mas;

}

public void changeTimeState(boolean is)

{

timeState = is;

}

@Override

public void paint(Graphics gr)

{

super.paint(gr);

if (simulationState)

paintSimulation(gr);

else

paint\_Statictics();

//paintStatictics(gr);

}

public void paint\_Statictics()

{

//если разрешено показывать инфо, выводим статистику, иначе пишем, что инфомрация недоступна

dialog.setLayout(new BorderLayout());

dialog.add(pan);

dialog.setLocation(550,300);

txt.setEditable(false);

pan.add(ok);

pan.add(cansel);

pan.setVisible(true);

if(showInfo)

txt.setText("Всего пчел: " + Bee.n + "\n" + "Трутней: " + BeeTruten.trNumber + "\n" + "Рабочих: "

+(Bee.n - BeeTruten.trNumber) + "\n" + "Время симуляции " + time);

else

txt.setText("Информация недоступна!");

dialog.add(txt, BorderLayout.NORTH);

dialog.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

dialog.setSize(300, 250);

dialog.setVisible(true);

//setSimulationState(true);

}

**Вывод:**

В ходе выполнения лабораторной работы мы познакомились с особенностями разработки графического интерфейса, изучили возможности библиотек Swing и Awt. Научились компоновать элементы графического интерфейса необходимым образом и совмещать разные типы управления процессом. Так же познакомились с ArrayList и обработкой исключений в Java.