**Министерство науки и высшего образования Российской Федерации**

**Новосибирский государственный технический университет**

**Кафедра вычислительной техники**

**Лабораторная работа №2**

**по дисциплине:** Технология программирования

**на тему:** Основы программирования на Java.

Обработка событий. Механизм делегирования событий.

Преподаватель: Михайленко Д.А.

Студент: Быков Д.А.

Группа: АВТ-808

Новосибирск 2020

**Цель**: 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. Познакомиться с паттерном Singleton.

1. Для хранения генерируемых объектов использовать динамический массив объектов. Для обеспечения доступа всем элементам приложения и создания только одного экземпляра массива объектов используйте структурный паттерн Singleton.
2. При реализации шаблона Singleton используется класс Singleton. В этом классе определяется закрытый конструктор, имеется закрытая статическая переменная, в которой хранится ссылка на единственный экземпляр данного класса, а также определен статический метод доступа, возвращающий ссылку на этот экземпляр. Так же класс содержит объект коллекции и открытые методы доступа к ней. Таким образом можно в любом месте приложения получить ссылку на объект класса Singleton и доступ к коллекции объектов.
3. 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 в меню и панель инструментов, в которых продублировать основные команды вашего 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 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 комбобокс и список (шаг значений 10%);
   * Дополнить интерфейс поясняющими метками.

Вариант №2

**Ход работы**

1. Был разработан класс Singleton по одноимённому паттерну проектирования, содержащий в себе статичное поле и статичный метод получения состояния объекта, а также сам список.

private static Singleton *instance*;  
public final List<BaseBee> bees;

1. Главный JFrame – Habitat поделен на три панели:
   1. MenuButtons – содержит в себе меню и панель инструментов
   2. Information – панель, отображающая таймер и количество объектов
   3. DrawBee– панель, отвечающая за отрисовку списка объектов
2. Information – диалоговое окно, показывающее информацию после окончания симуляции(можно отключить в панели инструментов)
3. MenuButtons делится на 4 компонента:
   1. ShowInfButton – наследник JPanel, содержит в себе пункты меню

JCheckBox chb;  
Boolean bl;

* 1. TimerShowHide – наследник JPanel, содержит в себе пункты меню

ButtonGroup buttonGroup;  
JRadioButton show\_timer;  
JRadioButton hide\_timer;  
Boolean bool = true;

* 1. ComboboxProbability – наследник JPanel, содержит в себе пункты меню

JComboBox cb;  
JComboBox cb1;  
Double probabilityOfWorkers;  
Double percent;

* 1. TimerOfBirth– наследник JPanel, содержит в себе пункты меню

JTextField txt;  
JTextField txt2;  
Integer birthTimeWorkers;  
Integer birthTimeDrones;

1. Класс Buttons – наследник JPanel содержит в себе кнопки начала и конца симуляции.
2. В классе Simulation обрабатываются события пользователя с меню и панели инструментов.

**Результат:**



Рис.1 – Во время симуляции.



Рис.2 – Во время завершения симуляции.

Код:

AbstractFactory

package com.company.Factory;  
  
import com.company.Models.Abstract.BaseBee;  
import com.company.Panels.MenuButtons;  
  
  
import java.io.IOException;  
  
public interface AbstractFactory {  
 int getAmountOfBirth();  
 void destroy();  
 BaseBee birth(int x\_cord, int y\_cord, int time, MenuButtons menuButtons)throws IOException;  
 int getAmountOfDrones();  
 int getAmountOfWorkers();  
  
}

ConcreneFactory

package com.company.Factory;  
  
import com.company.Models.Abstract.BaseBee;  
import com.company.Models.Drone;  
import com.company.Models.Worker;  
import com.company.Panels.ComboBoxProbability;  
import com.company.Panels.MenuButtons;  
import com.company.Panels.TimeOfBirth;  
  
import java.io.IOException;  
import java.util.Random;  
  
public class ConcreteFactory implements AbstractFactory {  
  
  
 private int allBeesBirth;  
 private int allDronesBirth;  
 private int allWorkersBirth;  
 private Double probabilityOfWorkers;  
 private Double percent;  
 private Integer birthTimeWorkers;  
 private Integer birthTimeDrones;  
  
 public ConcreteFactory(){  
 allBeesBirth = 0;  
 allDronesBirth = 0;  
 allWorkersBirth = 0;  
 }  
  
 @Override  
 public BaseBee birth(int x, int y, int time,MenuButtons menuButtons) throws IOException {  
  
 ComboBoxProbability comboBoxProbability = menuButtons.return\_comboBox();  
 this.probabilityOfWorkers = comboBoxProbability.return\_probabilityOfWorkers();  
 this.percent = comboBoxProbability.return\_percent();  
  
 TimeOfBirth timeOfBirth = menuButtons.return\_timerOfBirth();  
 this.birthTimeWorkers = timeOfBirth.return\_birthTimeWorkers();  
 this.birthTimeDrones = timeOfBirth.return\_birthTimeDrones();  
  
  
/\*  
 // if (time % this.birthTimeWorkers == 0) {  
  
 if (time % birthTimeDrones == 0 && allDronesBirth < percent \* allBeesBirth) {  
 BaseBee baseBee = new Drone(x, y);  
 allDronesBirth++;  
 allBeesBirth++;  
 return baseBee;  
 }  
 if (time % birthTimeWorkers == 0) {  
 Random random = new Random();  
 double randomProbability = random.nextDouble();  
 if (probabilityOfWorkers >= randomProbability) {  
 BaseBee baseBee = new Worker(x, y);  
 allBeesBirth++;  
 allWorkersBirth++;  
 return baseBee;  
 }  
 }  
  
 return null;  
 }  
\*/  
 if (time % this.birthTimeWorkers == 0){ // % - остаток от деления. Т.о. каждые n секунд  
  
 Random random = new Random();  
 double randomProbability = random.nextDouble(); // float nextFloat() - возвращает следующее случайное значение типа float  
  
 if (randomProbability <= probabilityOfWorkers) {  
 BaseBee baseBee = new Worker(x, y);  
 allBeesBirth++;  
 allWorkersBirth++;  
 return baseBee;  
 }  
 }  
  
 if (time % this.birthTimeDrones == 0) {  
 if(allDronesBirth < allWorkersBirth\*percent) {  
 BaseBee baseBee = new Drone(x, y);  
 allBeesBirth++;  
 allDronesBirth++;  
 return baseBee;  
 }  
 }  
 return null;  
 }  
  
  
  
  
 @Override  
 public int getAmountOfDrones() {  
 return allDronesBirth;  
 }  
  
 @Override  
 public int getAmountOfWorkers() {  
 return allWorkersBirth;  
 }  
  
  
  
 @Override  
 public int getAmountOfBirth() {  
 return allBeesBirth;  
 }  
  
 @Override  
 public void destroy() {  
 allBeesBirth = 0;  
 allDronesBirth = 0;  
 allWorkersBirth = 0;  
 }  
}

Singleton

package com.company.Habitat.BeesArray;  
  
import com.company.Habitat.Habitat;  
import com.company.Habitat.HabitatTask;  
import com.company.Models.Abstract.BaseBee;  
  
import javax.swing.\*;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Timer;  
  
public class Singleton {  
  
 private static Singleton *instance*;  
 public final List<BaseBee> bees;  
 private Timer timer;  
 private HabitatTask habitatTask;  
 Boolean isStart = true;  
  
  
 public Singleton() {  
 bees = new ArrayList<>();  
 }  
  
 public static Singleton getInstance(){  
 if (*instance* == null) {  
 *instance* = new Singleton();  
 }  
 return *instance*;  
 }  
  
 public void clear() {  
 if (!*instance*.bees.isEmpty())  
 *instance*.bees.clear();  
 }  
  
 public void addBees(BaseBee bee) {  
 *instance*.bees.add(bee);  
 }  
  
 public static List<BaseBee> getBees(){  
 return *instance*.bees;  
 }  
   
 public void Start(Habitat habitat, JButton startButton, JButton stopButton ){  
 isStart = false;  
 timer = new Timer();  
 habitatTask = new HabitatTask(habitat);  
 timer.schedule(habitatTask,0,1000);  
 stopButton.setEnabled(true);  
 startButton.setFocusable(false);  
 }  
  
 public void Stop(Habitat habitat, JButton startButton, JButton stopButton){  
 isStart = true;  
 timer.cancel();  
 habitatTask.cancel();  
 timer.purge();  
 habitat.stop();  
 stopButton.setEnabled(true);  
 startButton.setFocusable(false);  
 }  
  
 public Boolean getIsStart(){  
 return isStart;  
 }  
}

Habitat

package com.company.Habitat;  
  
import com.company.Factory.AbstractFactory;  
import com.company.Factory.ConcreteFactory;  
import com.company.Habitat.BeesArray.Singleton;  
import com.company.Models.Abstract.BaseBee;  
import com.company.Models.Drone;  
import com.company.Panels.\*;  
  
import javax.swing.\*;  
  
  
import java.awt.\*;  
import java.util.Random;  
  
public class Habitat extends JFrame {  
  
 private final Singleton bees;  
 private int time;  
 private AbstractFactory factory;  
 private DrawBee drawBee;  
 private MenuPanel menuPanel;  
 private Buttons button;  
 private MenuButtons menuButtons;  
  
 JButton startButton;  
 JButton stopButton;  
  
 public Habitat(){  
  
  
 int width = 1200;  
 int height = 800;  
 setSize(width, height);  
 setResizable(false);  
  
 setLayout(new BoxLayout(getContentPane(),BoxLayout.*Y\_AXIS*));  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 bees = Singleton.*getInstance*();  
 this.time = 0;  
  
  
 factory = new ConcreteFactory();  
  
 menuPanel = new MenuPanel();  
 menuPanel.setMaximumSize(new Dimension(width,50));  
 drawBee = new DrawBee();  
 drawBee.setBounds(0,21, width, height -21);  
 menuButtons = new MenuButtons(this);  
 menuButtons.setMaximumSize(new Dimension(width, 90));  
 button = new Buttons(this);  
 button.setMaximumSize(new Dimension(75,50));  
  
 startButton = button.returnStart();  
 stopButton = button.returnStop();  
  
 menuButtons.add(button);  
 add(drawBee);  
 add(menuPanel);  
 add(menuButtons);  
 setFocusable(true);  
 }  
  
 public JButton returnStart(){  
 return startButton;  
 }  
  
 public JButton returnStop() {  
 return stopButton;  
 }  
  
 public void stop(){//остановить отрисовку среды  
  
 Boolean b = true;  
 Boolean bl = menuButtons.return\_chb();  
  
 if (bl) {  
 Information informationPanel = new Information(this, "Информация", time, factory.getAmountOfWorkers(), factory.getAmountOfDrones());  
 informationPanel.setVisible(true);  
 b = informationPanel.return\_B();  
 }  
  
 if(b) {  
 time = 0;  
 bees.clear();  
 drawBee.repaint();  
 factory.destroy();  
 }  
 }  
  
 public void Timer\_show\_hide(Boolean bool){  
 if(!bool) menuPanel.TimerShowHide(false);  
 else menuPanel.TimerShowHide(true);  
 }  
  
 public Boolean return\_bool(){  
 Boolean bool = menuButtons.return\_bool();  
 menuButtons.return2\_bool();  
 return bool;  
 }  
  
  
 void update() {//продолжить симуляцию  
 time++;  
 menuPanel.setTimer(time);  
 Random coordinatesRandom = new Random();  
 int x\_cord = coordinatesRandom.nextInt(drawBee.getWidth()-100);  
 int y\_cord = coordinatesRandom.nextInt(drawBee.getHeight()-100);  
  
 requestFocus();//фокус на поле  
  
 try{  
 BaseBee bee = factory.birth(x\_cord,y\_cord,time, menuButtons);  
 menuPanel.setBeesAmount(factory.getAmountOfBirth());  
 if(bee != null) {  
 bees.addBees(bee);  
 drawBee.repaint();  
 }  
 }  
 catch (Exception ex){  
 ex.printStackTrace();  
 }  
 }  
}

HabitatTask

package com.company.Habitat;  
  
  
import java.util.TimerTask;  
  
public class HabitatTask extends TimerTask {  
 private final Habitat habitat;  
  
 public HabitatTask(Habitat habitat) {  
 this.habitat = habitat;  
 }  
  
 @Override  
 public void run() {  
 habitat.update();  
 }  
}

BaseBee

package com.company.Models.Abstract;  
  
import com.company.Models.Behaviour.IBehaviour;  
  
import java.awt.\*;  
  
public abstract class BaseBee implements IBehaviour {  
 private int x;  
 private int y;  
  
 protected Image beeImage;  
  
 protected BaseBee(int x, int y) {  
 this.x = x;  
 this.y = y;  
 }  
  
 @Override  
 public int getX() {  
 return x;  
 }  
  
 @Override  
 public void setX(int x) {  
 this.x = x;  
 }  
  
 @Override  
 public int getY() {  
 return y;  
 }  
  
 @Override  
 public void setY(int y) {  
 this.y = y;  
 }  
  
 @Override  
 public Image getBeeImage() {  
 return beeImage;  
 }  
  
 @Override  
 public void setBeeImage(Image beeImage) {  
 this.beeImage = beeImage;  
 }  
}

IBehaviour

package com.company.Models.Behaviour;  
  
import java.awt.\*;  
  
public interface IBehaviour {  
 int getX();  
  
 void setX(int x);  
  
 int getY();  
  
 void setY(int y);  
  
 Image getBeeImage();  
  
 void setBeeImage(Image beeImage);  
}

Drone

package com.company.Models;  
  
import com.company.Models.Abstract.BaseBee;  
  
import javax.imageio.ImageIO;  
import java.io.IOException;  
  
public class Drone extends BaseBee {  
 public Drone(int x, int y) throws IOException {  
 super(x, y);  
 this.beeImage = ImageIO.*read*(getClass().getResource("/drone.png"));  
 }  
}

Worker

package com.company.Models;  
  
import com.company.Models.Abstract.BaseBee;  
  
import javax.imageio.ImageIO;  
import java.io.IOException;  
  
public class Worker extends BaseBee {  
 public Worker(int x, int y) throws IOException {  
 super(x, y);  
 this.beeImage = ImageIO.*read*(getClass().getResource("/Worker.png"));  
 }  
}

Buttons

package com.company.Panels;  
  
import com.company.Habitat.BeesArray.Singleton;  
import com.company.Habitat.Habitat;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Buttons extends JPanel {  
 private final JButton startButton;  
 private final JButton stopButton;  
 Singleton st;  
  
 public Buttons(Habitat habitat){  
 setLayout(new GridLayout(2,1));  
 setFocusable(false);  
  
 startButton = new JButton("Старт");  
 // startButton.setFocusable(true);  
 startButton.setMnemonic('B');  
 startButton.setFocusable(false);  
 startButton.setEnabled(true);  
  
  
 stopButton = new JButton("Стоп");  
 stopButton.setMnemonic('E');  
 stopButton.setEnabled(false);  
  
 add(startButton);  
 startButton.setVisible(true);  
 add(stopButton);  
 stopButton.setVisible(true);  
  
 st = Singleton.*getInstance*();  
  
 startButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 st.Start(habitat,startButton,stopButton);  
 startButton.setEnabled(false);  
 stopButton.setEnabled(true);  
 setFocusable(false);  
 }  
 });  
  
 stopButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 st.Stop(habitat,startButton,stopButton);  
 stopButton.setEnabled(false);  
 startButton.setEnabled(true);  
 setFocusable(false);  
 }  
 });  
 stopButton.setFocusable(false);  
 startButton.setFocusable(false);  
 setFocusable(false);  
 }  
 public JButton returnStart() {  
 return startButton;  
 }  
  
 public JButton returnStop() {  
 return stopButton;  
 }  
}

ComboboxProbability

package com.company.Panels;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class ComboBoxProbability extends JPanel {  
  
 JComboBox cb;  
 JComboBox cb1;  
 Double probabilityOfWorkers;  
 Double percent;  
  
 public ComboBoxProbability(){  
 setLayout(new BoxLayout(this, BoxLayout.*X\_AXIS*));  
 Double[] Items = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0};  
  
 ActionListener actionListener = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 probabilityOfWorkers = (Double) cb.getSelectedItem();  
 cb.setFocusable(false);  
 }  
 };  
 ActionListener actionListener1 = new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 percent = (Double) cb1.getSelectedItem();  
 cb1.setFocusable(false);  
 }  
 };  
  
 cb = new JComboBox(Items);  
 cb.addActionListener(actionListener);  
 cb1 = new JComboBox(Items);  
 cb1.addActionListener(actionListener1);  
  
 JLabel text = new JLabel("Вер.рожд.трутней");  
 text.setFont(new Font("Courier New", Font.*BOLD*,11));  
 text.setForeground(Color.*BLACK*);  
 text.setHorizontalAlignment(SwingConstants.*CENTER*);  
  
 JLabel text2 = new JLabel("% отн. общ. числа");  
 text2.setFont(new Font("Courier New", Font.*BOLD*,11));  
 text2.setForeground(Color.*BLACK*);  
 text2.setHorizontalAlignment(SwingConstants.*CENTER*);  
  
 JPanel jpanel = new JPanel();  
 jpanel.setLayout(new GridLayout(2, 1));  
  
 JPanel jpanel2 = new JPanel();  
 jpanel2.setLayout(new GridLayout(2,1));  
  
 jpanel.add(text);  
 jpanel.add(cb);  
  
 jpanel2.add(text2);  
 jpanel2.add(cb1);  
  
 this.add(jpanel);  
 add(Box.*createRigidArea*(new Dimension(2,0))); // Отступ в расположении  
 this.add(jpanel2);  
 add(Box.*createRigidArea*(new Dimension(3,0)));  
  
 setVisible(true);  
  
 setFocusable(false);  
  
 }  
 public Double return\_probabilityOfWorkers(){  
 return probabilityOfWorkers;  
 }  
  
 public Double return\_percent(){  
 return percent;  
 }  
  
}

DrawBee

package com.company.Panels;  
  
import com.company.Habitat.BeesArray.Singleton;  
import com.company.Models.Abstract.BaseBee;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
  
public class DrawBee extends JPanel {  
 private final Singleton bees;  
  
 public DrawBee() {  
 bees = Singleton.*getInstance*();  
 }  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 if(bees.*getBees*().isEmpty())  
 return;  
 for (BaseBee bee : bees.*getBees*()) {  
 setBackground(Color.*PINK*);  
 g.drawImage(bee.getBeeImage(),bee.getX(),bee.getY(),40,40,null);  
 }  
 }  
}

Information

package com.company.Panels;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class Information extends JDialog {  
  
 private final JTextArea textArea;  
 private final JButton ok;  
 private final JButton cancel;  
 Boolean b = true;  
  
 public Information(JFrame frame, String title, int time, int allWorkersBirth, int allDronesBirth) {  
 super(frame, "Информация", false);  
  
 setLayout(new BoxLayout(getContentPane(), BoxLayout.*Y\_AXIS*));  
  
 textArea = new JTextArea();//создаем текстовое окно  
  
 String s\_time = "Время симуляции: " + time + "\n";  
 String s\_allWorkersBirth = "Количество рабочих пчел: " + allWorkersBirth + "\n";  
 String s\_allDronesBirth = "Количество трутней: " + allDronesBirth + "\n";  
  
 textArea.append(s\_time);  
 textArea.append(s\_allWorkersBirth);  
 textArea.append(s\_allDronesBirth);  
 textArea.setEditable(false); // Не редактируемое поле  
  
 setResizable(false); // Не изменять размеры окна  
 setDefaultCloseOperation(WindowConstants.*DISPOSE\_ON\_CLOSE*);  
  
 setSize(new Dimension(300, 300));  
 textArea.setFont(new Font("Dialog", Font.*BOLD*, 14));  
 textArea.setTabSize(10);  
 add(textArea);  
  
 textArea.setMaximumSize(new Dimension(300, 280));  
 this.setModal(true); // Блокирует остальные окна  
  
 ok = new JButton("Ок");  
 cancel = new JButton("Отмена");  
  
 ok.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 b = true;  
 setVisible(false);  
 }  
 });  
  
 cancel.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 b = false;  
 setVisible(false);  
 }  
 });  
  
 JPanel jPanel = new JPanel();  
 jPanel.setLayout(new GridLayout(1, 2));  
  
  
 jPanel.setLayout(new GridLayout(1, 2));  
 jPanel.add(ok);  
 jPanel.add(cancel);  
  
 add(jPanel);  
 jPanel.setMaximumSize(new Dimension(180, 20));  
  
 jPanel.setVisible(true);  
  
 }  
  
 public Boolean return\_B(){  
 return b;  
 }  
}

MenuButtons

package com.company.Panels;  
  
import com.company.Habitat.Habitat;  
  
import javax.swing.\*;  
  
  
public class MenuButtons extends JPanel {  
  
 ShowInfButton chb;  
 TimerShowHide timerShowHide;  
 ComboBoxProbability comboBoxProbability;  
 TimeOfBirth timeOfBirth;  
  
 public MenuButtons(Habitat habitat){  
  
 setLayout(new BoxLayout(this, BoxLayout.*X\_AXIS*));  
  
 chb = new ShowInfButton();// показать инфу да/нет  
 add(chb);  
  
 timerShowHide = new TimerShowHide(habitat);//скрыть/показать таймер  
 add(timerShowHide);  
  
 comboBoxProbability = new ComboBoxProbability();//вероятность рождения пчел  
 add(comboBoxProbability);  
  
 timeOfBirth = new TimeOfBirth();//время рождения тех или других пчел  
 add(timeOfBirth);  
  
 setVisible(true);  
 setFocusable(false);  
 }  
  
  
 public Boolean return\_chb() {  
 Boolean bl = chb.return\_chb();  
 return bl;  
 }  
  
 public Boolean return\_bool(){  
 Boolean bool = timerShowHide.return\_bool();  
 return bool;  
 }  
 public void return2\_bool(){  
 timerShowHide.return2\_bool();  
 }  
  
 public ComboBoxProbability return\_comboBox() {  
 return comboBoxProbability;  
 }  
  
 public TimeOfBirth return\_timerOfBirth(){  
 return timeOfBirth;  
 }  
}

MenuPanel

package com.company.Panels;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class MenuPanel extends JPanel {  
  
 private final JLabel beesAmount;  
 private final JLabel timer;  
  
  
 public MenuPanel()  
 {  
 setLayout(new GridLayout(1,2));  
 beesAmount = new JLabel("Количество рожденных пчел: 0");  
 beesAmount.setFont(new Font("Courier New", Font.*BOLD*,16));  
 timer = new JLabel("Таймер: 0");  
 timer.setFont(new Font("Courier New", Font.*BOLD*,16));  
  
 add(beesAmount);  
 add(timer);  
 }  
  
 public void setBeesAmount(int beesAmount) {  
 this.beesAmount.setText("Количество рожденных пчел: " + beesAmount);  
 }  
  
 public void setTimer(Integer timer) {  
 this.timer.setText("Таймер: " + timer);  
 }  
  
 // public void setTimerVisibility(Boolean isVisible) {  
 // timer.setVisible(isVisible);  
 //}  
  
 /\* public Boolean return\_bool() {  
 return bool;  
 }  
  
 public void return2\_bool() {  
  
 }\*/  
  
 public void TimerShowHide(boolean b) {  
 timer.setVisible(b);  
 }  
}

ShowInfPanel

package com.company.Panels;  
  
import javax.swing.\*;  
  
  
public class ShowInfButton extends JPanel {  
 JCheckBox chb;  
 Boolean bl;  
  
 public ShowInfButton() {  
 chb = new JCheckBox("Показать информацию", true);  
 add(chb);  
 chb.setFocusable(false);  
 }  
  
 public Boolean return\_chb() {  
 bl = chb.isSelected(); // Стоит ли галочка да\нет return d;  
 return bl;  
 }  
}

TimeOfBirth

package com.company.Panels;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class TimeOfBirth extends JPanel {  
  
 JTextField txt;  
 JTextField txt2;  
 Integer birthTimeWorkers;  
 Integer birthTimeDrones;  
  
 public TimeOfBirth(){  
 setLayout(new BoxLayout(this, BoxLayout.*X\_AXIS*));  
  
 JLabel text = new JLabel("Время рождения раб.");  
 text.setFont(new Font("Courier New", Font.*BOLD*,11));  
 text.setForeground(Color.*BLACK*);  
 text.setHorizontalAlignment(SwingConstants.*CENTER*);  
  
  
 JLabel text2 = new JLabel("Время рождения трут.");  
 text2.setFont(new Font("Courier New", Font.*BOLD*,11));  
 text2.setForeground(Color.*BLACK*);  
 text2.setHorizontalAlignment(SwingConstants.*CENTER*);  
  
  
 JPanel panel = new JPanel();  
 panel.setLayout(new GridLayout(2, 1));  
  
  
 JPanel panel2 = new JPanel();  
 panel2.setLayout(new GridLayout(2,1));  
  
 panel.add(text);  
 panel2.add(text2);  
  
 txt = new JTextField(9);  
 txt2 = new JTextField(9);  
  
 txt.setHorizontalAlignment(JTextField.*CENTER*);  
 //textArea\_small.setToolTipText("Длиное поле"); // Установка метки  
 txt2.setHorizontalAlignment(JTextField.*CENTER*);  
  
 txt.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try { birthTimeWorkers = Integer.*parseInt*(txt.getText());}  
 catch (NumberFormatException e1) {  
 birthTimeWorkers = 2;  
 JFrame frame = new JFrame("Внимание");  
 frame.setSize(200,200);  
 JTextArea textArea = new JTextArea();  
 String string1 = "Введены некорректные данные\n";  
 String string2 = "Период рожд. пт. = 2\n";  
 String string3 = "(значение по умолчанию)\n";  
 textArea.append(string1);  
 textArea.append(string2);  
 textArea.append(string3);  
 textArea.setEditable(false);  
 frame.add(textArea);  
 frame.setVisible(true);  
 txt.setText("2");  
 }  
 txt.setEnabled(false);  
 txt.setFocusable(false);  
 }  
 });  
  
 txt2.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 try { birthTimeDrones = Integer.*parseInt*(txt2.getText());}  
 catch (NumberFormatException e1) {  
 birthTimeDrones = 2;  
 JFrame frame = new JFrame("Внимание");  
 frame.setSize(200,200);  
 JTextArea textArea = new JTextArea();  
 String string1 = "Введены некорректные данные\n";  
 String string2 = "Период рожд. пт. = 5\n";  
 String string3 = "(значение по умолчанию)\n";  
 textArea.append(string1);  
 textArea.append(string2);  
 textArea.append(string3);  
 textArea.setEditable(false);  
 frame.add(textArea);  
 frame.setVisible(true);  
 txt2.setText("2");  
 }  
 txt2.setEnabled(false);  
 txt2.setFocusable(false);  
 }  
 });  
 panel.add(txt);  
 panel2.add(txt2);  
  
 add(panel);  
 add(Box.*createRigidArea*(new Dimension(5,0))); // Разделитель  
 add(panel2);  
  
 setVisible(true);  
  
 }  
 public Integer return\_birthTimeDrones() {  
 return birthTimeDrones;  
 }  
  
 public Integer return\_birthTimeWorkers() {  
 return birthTimeWorkers;  
 }  
  
}

TimerShowHide

package com.company.Panels;  
  
import com.company.Habitat.Habitat;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class TimerShowHide extends JPanel {  
 ButtonGroup buttonGroup;  
 JRadioButton show\_timer;  
 JRadioButton hide\_timer;  
 Boolean bool = true;  
  
 public TimerShowHide(Habitat habitat){  
 buttonGroup = new ButtonGroup();//объединение в группу  
  
 setLayout(new GridLayout(2, 1));  
  
 show\_timer = new JRadioButton("Показать таймер",true);  
 hide\_timer = new JRadioButton("Скрыть таймер",true);  
  
 buttonGroup.add(show\_timer);  
 buttonGroup.add(hide\_timer);  
 add(show\_timer);  
 add(hide\_timer);  
 setVisible(true);  
 show\_timer.setFocusable(false);  
 hide\_timer.setFocusable(false);  
 setFocusable(false);  
  
 show\_timer.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 habitat.Timer\_show\_hide(true);  
 bool = true;  
 show\_timer.setFocusable(false);  
 }  
 });  
 hide\_timer.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 habitat.Timer\_show\_hide(false);  
 bool = false;  
 hide\_timer.setFocusable(false);  
 }  
 });  
  
 }  
  
 public Boolean return\_bool(){  
 return bool;  
 }  
  
 public void return2\_bool(){  
 this.bool =!this.bool;  
 }  
}

Main

package com.company;  
  
import com.company.Habitat.BeesArray.Singleton;  
import com.company.Habitat.Habitat;  
  
import javax.swing.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
  
public class Main extends KeyAdapter{  
  
 private final Habitat habitat;  
 Singleton st;  
 JButton startButton;  
 JButton stopButton;  
 private Boolean bool;  
 private Boolean isStarted;  
  
  
 private Main() {  
 habitat = new Habitat();  
 habitat.setVisible(true);  
 habitat.addKeyListener(this);  
 startButton = habitat.returnStart();  
 stopButton = habitat.returnStop();  
 st = Singleton.*getInstance*();  
 bool = true;  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 if (e.getKeyCode() == KeyEvent.*VK\_B*) {  
 isStarted = st.getIsStart();//isStarted присваиваем то, какое оно в Singleton  
 if(isStarted) { //isStarted = true  
 st.Start(habitat, startButton, stopButton);  
 habitat.requestFocus();  
 startButton.setEnabled(false);  
 stopButton.setEnabled(true);  
 isStarted = false;//кнопка на клавиатуре нажата и если нажмем еще раз, она работать не будет  
 }  
 }  
  
 if (e.getKeyCode() == KeyEvent.*VK\_E* ){  
 isStarted = st.getIsStart();//isStarted присваиваем true через метод getIsStart который возвращает isStart  
 if(!isStarted) {//isStarted = false  
 st.Stop(habitat, startButton, stopButton);  
 habitat.requestFocus();  
 stopButton.setEnabled(false);  
 startButton.setEnabled(true);  
 isStarted = true;  
 }  
 }  
 if(e.getKeyCode() == KeyEvent.*VK\_T*) {  
 bool = habitat.return\_bool();  
 bool=!bool;  
 habitat.Timer\_show\_hide(bool);  
 }  
 }  
  
 public static void main(String[] args) {  
 new Main();  
 }  
  
}