**Министерство науки и высшего образования Российской Федерации**

**Новосибирский государственный технический университет**

**Кафедра вычислительной техники**

**Лабораторная работа №1**

**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 особенностями технологии Java и изучить синтаксис языка. Изучить основные понятия и термины обработки событий в модели делегирования событий.

**Задание**: разработать программу. Основная задача – разработка упрощенной имитации поведения объектов (все последующие лабораторные работы будут расширять это задание). Объекты реализуются через наследование: абстрактный класс + интерфейс → наследники.

Рабочий цикл программы:

- запускается процесс симуляции по клавише, генерируются объекты классов согласно заданию;

- симуляция завершается по другой клавише, выводится статистическая информация.

Вариант 2

Объект – пчела. Бывают 2 видов: трутень и рабочий. Трутни рождаются каждые N1 секунд, если их количество менее K% от общего числа пчел, в противном случае – не рождаются вовсе. Рабочие рождаются каждые N2 секунд с вероятностью P.

**Ход работы**

1. Был разработан абстрактный класс, модель пчелы BaseBee согласно варианту задания, содержащая в себе координаты точки на экране и картинку.

private int x;  
private int y;  
  
protected Image beeImage;

1. Были сделаны наследники Worker, Drone абстрактного класса BaseBee, каждый из них определяет картинку своего вида.

this.beeImage = ImageIO.*read*(getClass().getResource("/drone.png"));  
this.beeImage = ImageIO.*read*(getClass().getResource("/Worker.png"));

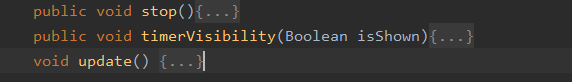
1. Объявлен интерфейс AbstractFactory, содержащий методы:

int getAmountOfBirth();  
void destroy();  
BaseBee birth(int x\_cord, int y\_cord, int time)throws IOException;

1. Класс ConcreteFactory имплементирует интерфейс AbstractFactory, содержит в себе поля для симуляции объектов с конкретной логикой. ( пчелы )

int timeBirthDrone;  
int timeBirthWorker;  
double probabilityJoWorker;  
double percentAllBees;  
int allBeesBirth;  
int allDronesBirth;

1. Разработан класс Habitat, отрисовывающий все объекты на экране, содержащий в себе методы:



1. Описан класс HabitatTask для запуска отрисовку среды с какой-то временной периодичностью.

@Override  
public void run() {  
 habitat.update();  
}

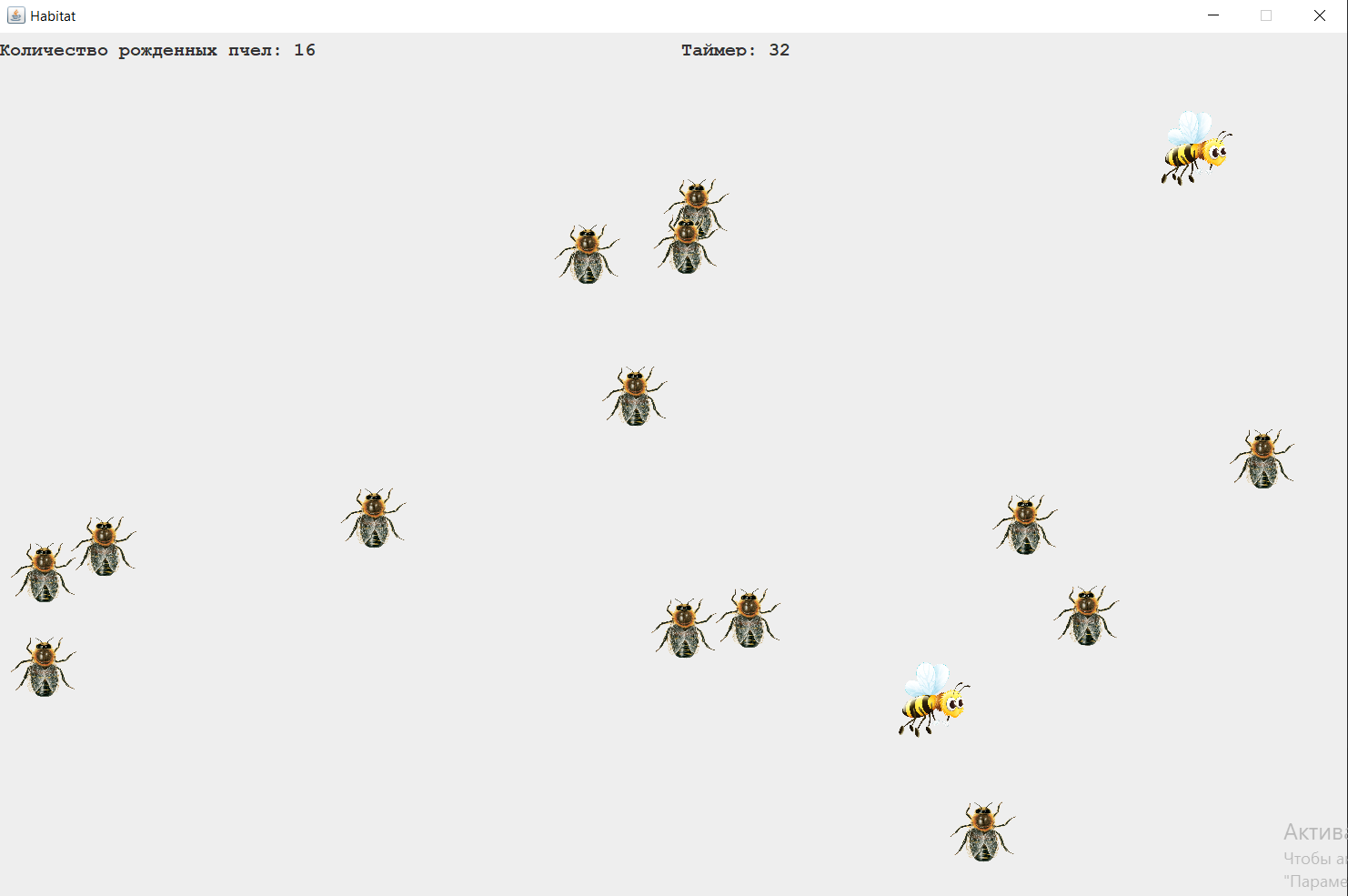
1. Реализован класс Simulation, запускающий симуляцию, и перехватывающий клавиатурные события:

- При нажатии кнопки B, симуляция запускается.

- При нажатии кнопки T, скрывается или появляется таймер

-При нажатии кнопки E, симуляция завершается, выводится итоговая информация.

**Результат работы программы:**

  
Рис.1 – Во время работы программы.

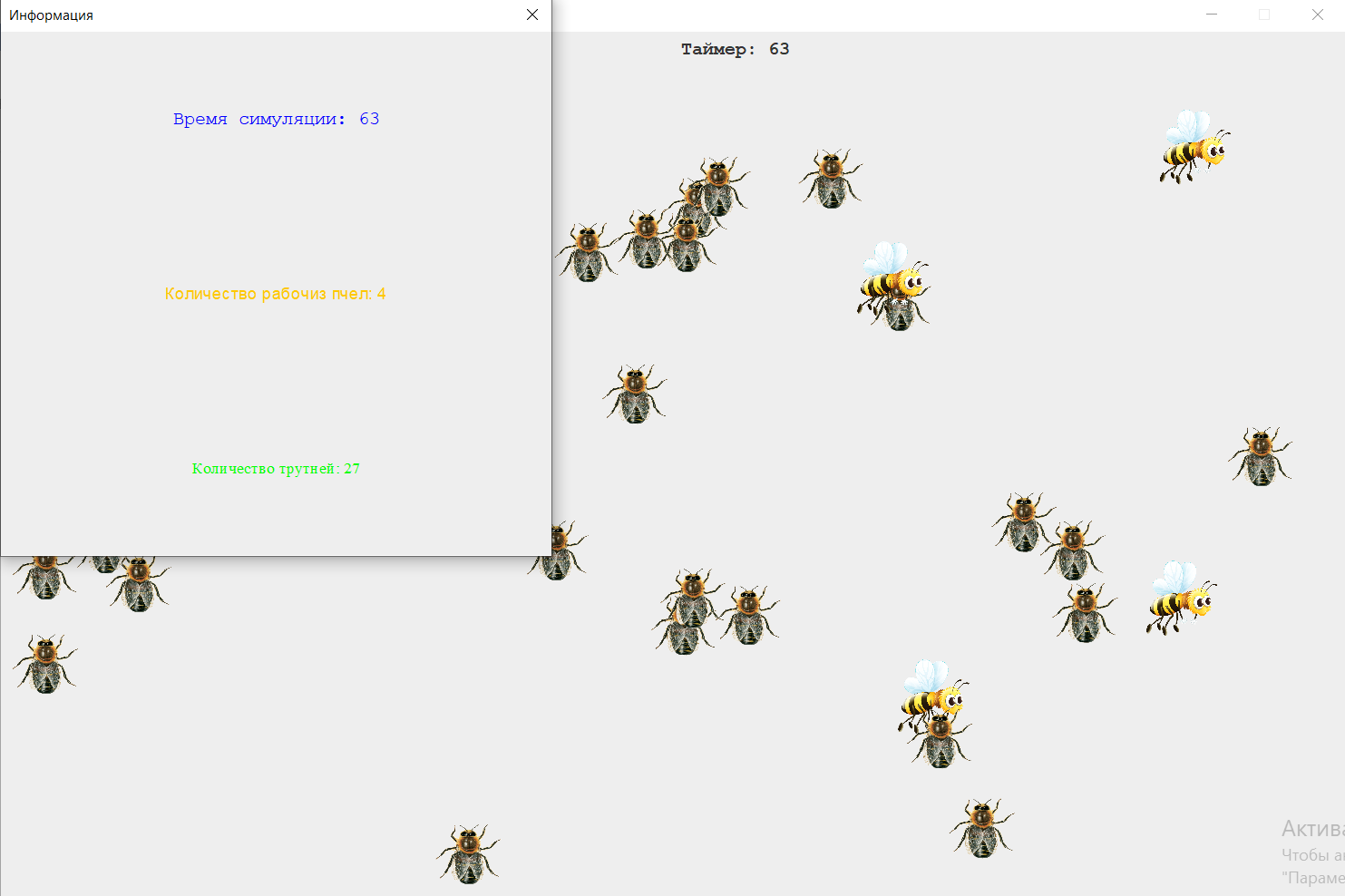


Рис.2 – Программа закончила работу.

**Вывод:** в данной работе была реализована программа симуляции рождения кроликов. Были изучены основные понятия и термины обработки событий в модели делегирования событий на Java. Использован базовый синтаксис языка Java, и графические объекты библиотеки AWT и Swing. Была продемонстрирована возможность перехватывать события пользователя с клавиатуры в программе.

**Код программы:**

AbstractFactory

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

ConcreteFactory

package com.company.Factory;  
  
import com.company.Models.Abstract.BaseBee;  
import com.company.Models.Drone;  
import com.company.Models.Worker;  
  
import java.io.IOException;  
import java.util.Random;  
  
public class ConcreteFactory implements AbstractFactory {  
  
 int timeBirthDrone;  
 int timeBirthWorker;  
 double probabilityJoWorker;  
 double percentAllBees;  
 int allBeesBirth;  
 int allDronesBirth;  
  
 public ConcreteFactory(int timeBirthDrone,int timeBirthWorker, double probabilityJoWorker,double percentAllBees){  
 this.timeBirthDrone = timeBirthDrone;  
 this.timeBirthWorker = timeBirthWorker;  
 this.probabilityJoWorker = probabilityJoWorker;  
 this.percentAllBees = percentAllBees;  
 }  
  
 @Override  
 public BaseBee birth(int x, int y, int time) throws IOException {  
 if(time%timeBirthDrone == 0 && allDronesBirth<percentAllBees\*allBeesBirth){  
 BaseBee baseBee = new Drone(x,y);  
 allDronesBirth++;  
 allBeesBirth++;  
 return baseBee;  
 }  
 if(time%timeBirthWorker == 0){  
 Random random = new Random();  
 double randomProbability = random.nextDouble();  
 if(probabilityJoWorker > randomProbability){  
 BaseBee baseBee = new Worker(x,y);  
 allBeesBirth++;  
 return baseBee;  
 }  
 }  
 return null;  
 }  
  
 @Override  
 public int getAmountOfBirth() {  
 return allBeesBirth;  
 }  
  
 @Override  
 public void destroy() {  
 allBeesBirth = 0;  
 allDronesBirth = 0;  
 }  
  
  
}

Habitat

package com.company.Habitat;  
  
import com.company.Factory.AbstractFactory;  
import com.company.Factory.ConcreteFactory;  
import com.company.Models.Abstract.BaseBee;  
import com.company.Models.Drone;  
import com.company.Panels.DrawBee;  
import com.company.Panels.Information;  
import com.company.Panels.MenuPanel;  
  
import javax.swing.\*;  
  
import java.util.Random;  
  
public class Habitat extends JFrame {  
  
 private int i = 0;  
 private BaseBee [] bees;  
 private int time;  
 private AbstractFactory factory;  
 private DrawBee drawBee;  
 private MenuPanel menuPanel;  
 private Integer DronesAmount;  
  
 public Habitat(String title,int timeBirthDrone,int timeBirthWorker, double probabilityJoWorker,double percentAllBees){  
 super(title);  
  
 int width = 1200;  
 int height = 800;  
 setSize(width, height);  
 setResizable(false);  
 setLayout(null);  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 this.time = 0;  
 bees = new BaseBee[500];  
 factory = new ConcreteFactory(timeBirthDrone,timeBirthWorker,probabilityJoWorker,percentAllBees);  
 menuPanel = new MenuPanel();  
 menuPanel.setBounds(0,0, width,30);  
 drawBee = new DrawBee(bees);  
 drawBee.setBounds(0,21, width, height -21);  
 add(menuPanel);  
 add(drawBee);  
 DronesAmount=0;  
 }  
  
 public void stop(){//остановить отрисовку среды  
 Information informationPanel = new Information(this,"Информация",time,factory.getAmountOfBirth() - DronesAmount,DronesAmount);  
 informationPanel.viewInformation();  
 factory.destroy();  
 for (int j=0; j<500; j++) // "Обнуление" массива  
 { bees[j] = null; }  
 i = 0;  
 time = 0;  
 menuPanel.setBeesAmount(factory.getAmountOfBirth());  
 menuPanel.setTimer(time);  
 drawBee.repaint();  
 }  
 public void timerVisibility(Boolean isShown){//скрыть таймер  
 if (isShown)  
 menuPanel.setTimerVisibility(false);  
 else  
 menuPanel.setTimerVisibility(true);  
 }  
 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);  
  
 try{  
 BaseBee bee = factory.birth(x\_cord,y\_cord,time);  
 menuPanel.setBeesAmount(factory.getAmountOfBirth());  
 if(bee != null)  
 {  
 if (bee.getClass() == Drone.class)  
 DronesAmount++;  
 bees[i] = bee;  
 i++;  
 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"));  
 }  
}

DrawBee

package com.company.Panels;  
  
import com.company.Models.Abstract.BaseBee;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
  
public class DrawBee extends JPanel {  
 private BaseBee [] bees;  
  
  
 public DrawBee(BaseBee[] bees){  
 this.bees = bees;  
 }  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 if (bees[0] == null)  
 return;  
 for (int i=0; bees [i] != null; i++)  
 if (i < 500)  
 g.drawImage(bees[i].getBeeImage(),bees[i].getX(),bees[i].getY(),70,70,null);  
 }  
 }

Information

package com.company.Panels;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class Information extends JDialog {  
 private JLabel timerResult;  
 private JLabel WorkerResult;  
 private JLabel DroneResult;  
  
 public Information(JFrame frame, String title, Integer time, Integer WorkerResult, Integer DroneResult){  
 super(frame, title, false);  
  
 this.timerResult = new JLabel("Время симуляции: " + time);  
 this.timerResult.setFont(new Font("Courier New", Font.*PLAIN*,16));  
 this.timerResult.setForeground(Color.*BLUE*);  
  
 this.WorkerResult = new JLabel("Количество рабочиз пчел: " + WorkerResult);  
 this.WorkerResult.setFont(new Font("Arial", Font.*PLAIN*,15));  
 this.WorkerResult.setForeground(Color.*ORANGE*);  
  
 this.DroneResult = new JLabel("Количество трутней: " + DroneResult);  
 this.DroneResult.setFont(new Font("Times New Roman", Font.*PLAIN*,14));  
 this.DroneResult.setForeground(Color.*GREEN*);  
  
 setResizable(false);  
 setDefaultCloseOperation(WindowConstants.*DISPOSE\_ON\_CLOSE*);  
 setSize(500, 500);  
 setLayout(new GridLayout(3,1));  
  
 this.timerResult.setHorizontalAlignment(SwingConstants.*CENTER*);  
 add(this.timerResult);  
  
 this.WorkerResult.setHorizontalAlignment(SwingConstants.*CENTER*);  
 add(this.WorkerResult);  
  
 this.DroneResult.setHorizontalAlignment(SwingConstants.*CENTER*);  
 add(this.DroneResult);  
  
 this.setModal(true);  
  
 }  
  
 public void viewInformation(){  
 this.timerResult.setVisible(true);  
 this.WorkerResult.setVisible(true);  
 this.DroneResult.setVisible(true);  
 setVisible(true);  
 }  
}

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);  
 }  
}

Main package com.company;  
  
import com.company.Habitat.Habitat;  
import com.company.Habitat.HabitatTask;  
  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
import java.util.Timer;  
  
public class Main extends KeyAdapter {  
  
 private Timer timer;  
 private HabitatTask habitatTask;  
 private final Habitat habitat;  
 private Boolean isStarted;  
 private Boolean isShown;  
  
 private Main() {  
 habitat = new Habitat("Habitat", 2,2,0.9f,0.9f);  
 habitat.setVisible(true);  
 habitat.addKeyListener(this);  
 isStarted = false;  
 isShown = true;  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 super.keyPressed(e);  
 int key = e.getKeyCode();  
 if(key == KeyEvent.*VK\_B* && !isStarted){  
 isStarted = true;  
 timer = new Timer();  
 habitatTask = new HabitatTask(habitat);  
 timer.schedule(habitatTask,0,1000);  
 }  
 if (key == KeyEvent.*VK\_E* && isStarted) {  
 timer.cancel();  
 habitatTask.cancel();  
 timer.purge();  
 habitat.stop();  
 isStarted = false;  
 }  
 if(key == KeyEvent.*VK\_T*) {  
 isShown = !isShown;  
 habitat.timerVisibility(isShown);  
 }  
 }  
  
 public static void main(String[] args) {  
 new Main();  
 }  
}