Министерство науки и высшего образования РФ

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Кафедра Вычислительной техники

Новосибирск 2020

|  |  |
| --- | --- |
| Факультет: АВТФ  Группа: АВТ-814  Студент: Альжанов В.А. | Преподаватель:  Васюткина Ирина Александровна |

Лабораторная работа 4

«Многопотоковые приложения»

по дисциплине «Технологии программирования»

**Цель работы**

1. Изучить особенности реализации и работы потоков в Java, управление приоритетами потоков и синхронизацией потоков.
2. Доработать программу, созданную в лабораторной работе №3.

**Задание**

Вариант 2

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 управление монитором (методы wait() и notify());
5. добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

1. Пчелы-рабочие двигаются в один из углов области их обитания (например, [0;0]) по прямой со скоростью V, а затем возвращаться обратно в точку своего рождения с той же скоростью.

2. Трутни двигаются хаотично со скоростью V. Хаотичность достигается случайной сменой направления движения раз в N секунд.

**Демонстрация работы программы**

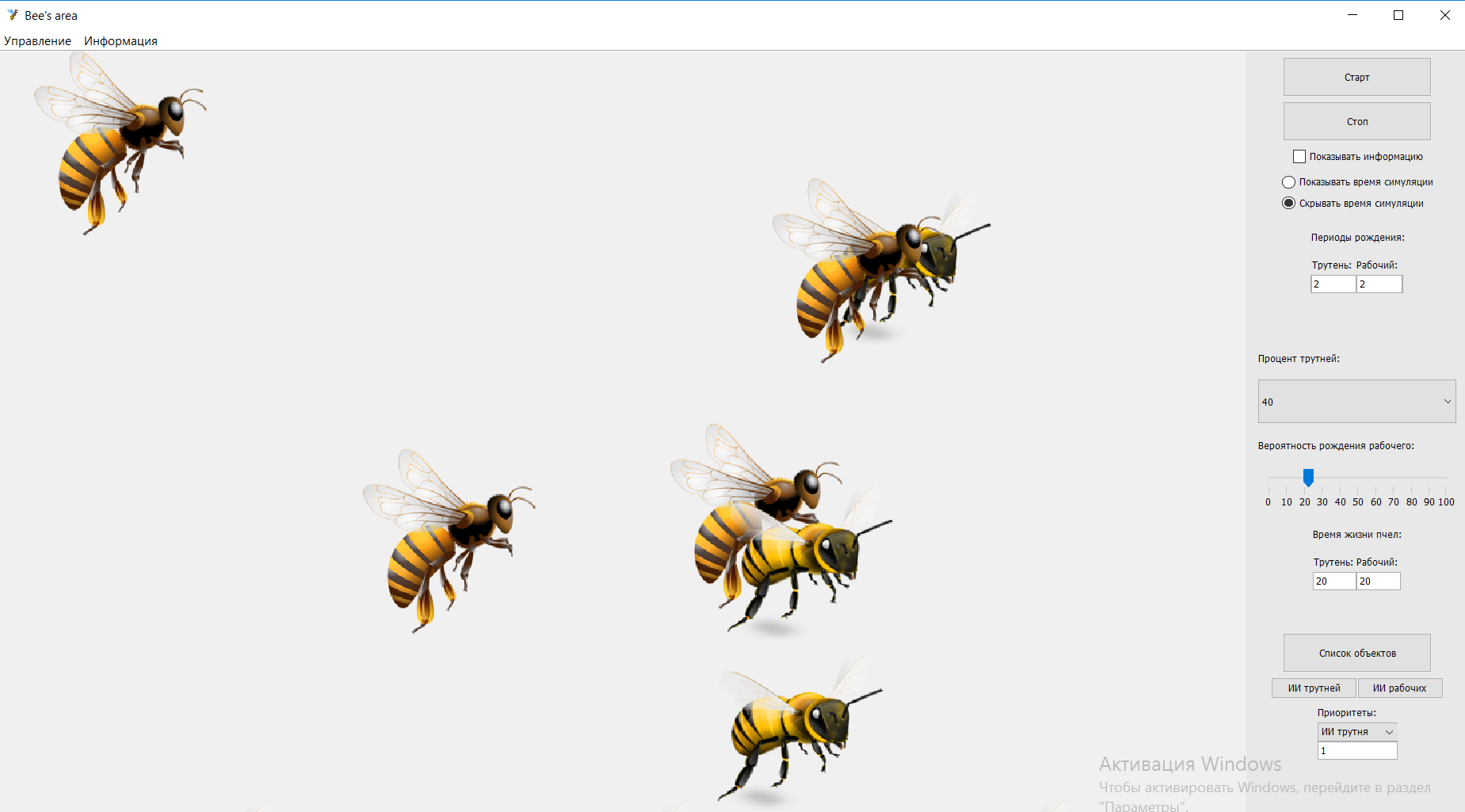


Рис. 1 – Движение объектов по области симуляции

**Описание блоков программы**

class BaseAI – абстрактный класс, содержащий статические переменные int priority (изменение приоритета потока) и boolean move (при ее значении, равном true, происходит активация движения объектов). Класс унаследован от класса Thread, который реализует потоки.

abstract class BaseAI extends Thread {  
 static int *priority* = 1;  
 static boolean *move* = true;  
}

Class DroneAI – класс-наследник класса BaseAI, описывающий интеллектуальное поведение объектов класса Drone. Содержит константный объект типа Object, необходимый для синхронизации выполнения потока, а также переопределенный метод run().

class DogAI – класс-наследник класса BaseAI, описывающий интеллектуальное поведение объектов класса Dog. Содержит константный объект типа Object, необходимый для синхронизации выполнения потока, а также переопределенный метод run().

public class DroneAI extends BaseAI {  
  
 final Object tmp = new Object();  
  
 @Override  
 public void run() {  
 while (*move*) {  
 synchronized (tmp) {  
 if (GUI.*stopDroneAI*) {  
 try {  
 tmp.wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 for (int i = 0; i < Habitat.*array*.size(); i++) {  
 if (Habitat.*array*.get(i) instanceof Drone) {  
 Habitat.*array*.get(i).move();  
 }  
 }  
 }  
  
 try {  
 Thread.*sleep*(10);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

class Bee был изменен. в него была добавлена целочисленная переменная speed, отвечающая за скорость движения объектов (на сколько пикселей сдвинется объект за период выполнения задачи таймером).

class Drone был изменен. Теперь в нем появились булевые переменные border и

borderY;, значение которых изменяется по мере достигания объектом типа Drone границ экрана а также переменная end которая обнуляет цикл движения трутней. Кроме того, в классе появился переопределенный метод move(), унаследованный от интерфейса IBehaviour. Метод описывает траекторию движения объектов по периметру области симуляции.

import java.awt.\*;  
import java.util.Random;  
  
public class Drone extends Bee {  
 private int Vx,  
 Vy,  
 rx,ry;  
 private boolean  
 end =true,  
 borderX=true,  
 borderY=true;  
 Drone(Image img, int x, int y, long timeOfBirth,int id) {  
 this.img = img;  
 this.x = x;  
 this.y = y;  
 this.timeOfBirth = timeOfBirth;  
 this.id=id;  
 V = 1; // скорость   
 Vx=V;  
 Vy=V;  
 }  
  
 @Override  
 public void move() {  
 if (end == true)  
 {  
 rx = new Random().nextInt(2);  
 ry = new Random().nextInt(2);  
 if (rx == 0) Vx \*= -1;  
 if (ry == 0) Vy \*= -1;  
 end = false;  
  
 }  
 if (borderX) {  
 if (Vx > 0) {  
 if (this.x <= Habitat.*areaSizeX*) {  
 this.x += Vx;  
 } else borderX = false;  
 }  
 if (Vx < 0) {  
 if (this.x >= 0) {  
 this.x += Vx;  
 } else borderX = false;  
 }  
 }  
  
 if (borderY) {  
 if (Vy > 0) {  
 if (this.y <= Habitat.*areaSizeY*) {  
 this.y += Vy;  
 } else borderY = false;  
 }  
 if (Vy < 0) {  
 if (this.y >= 0) {  
 this.y += Vy;  
 } else borderY = false;  
 }  
 }  
  
 if ( borderX==false && borderY==false) {  
  
 if (Vx < 0) {  
 if (this.x <= Habitat.*areaSizeX*/2) {  
 this.x -= Vx;  
 }  
 }  
  
 if (Vx > 0) {  
 if (this.x >= Habitat.*areaSizeX*/2) {  
 this.x -= Vx;  
 }  
 }  
  
 if (Vy < 0) {  
 if (this.y <= Habitat.*areaSizeY*/2) {  
 this.y -= Vy;  
 }  
 }  
  
 if (Vy > 0) {  
 if (this.y >= Habitat.*areaSizeY*/2) {  
 this.y -= Vy;  
 }  
 }  
 }  
  
 if (((Vy < 0 && this.y >= Habitat.*areaSizeY* / 2) || (Vy > 0 && this.y <= Habitat.*areaSizeY* / 2)) &&  
 ((Vx < 0 && this.x >= Habitat.*areaSizeX* / 2) || (Vx > 0 && this.x <= Habitat.*areaSizeX* / 2))) {  
 borderY =true;  
 borderX = true;  
 end=true;  
 }  
  
 }  
}

class Worker был изменен. Теперь в нем появился переопределенный метод move(), унаследованный от интерфейса IBehaviour. Метод описывает траекторию движения объектов класса .

import java.awt.\*;  
import java.util.Random;  
  
  
public class Worker extends Bee {  
 long time;  
 int N;  
 int direction;  
 Worker(Image img, int x, int y, long timeOfBirth,int id) {  
 this.img = img;  
 this.x = x;  
 this.y = y;  
 this.timeOfBirth = timeOfBirth;  
 this.id=id;  
 V = 1; // скорость   
 N=1;//меняется направление раз в N секунд  
 time=0;  
 }  
  
 @Override  
 public void move() {  
 time++;  
  
 if (time % (N\*100) == 0) {  
 direction=new Random().nextInt(4);  
   
 }  
 if (direction == 0 && this.x <= Habitat.*areaSizeX*) {  
  
 this.x += V;  
 }  
 if (direction == 1 && this.x > 0) {  
  
 this.x -= V;  
 }  
  
 if (direction == 2 && this.y <= Habitat.*areaSizeY*) {  
  
 this.y += V;  
 }  
 if (direction == 3 && this.y > 0) {  
  
 this.y -= V;  
 }  
 }  
}

**Выводы**

В ходе выполнения работы были изучены особенности реализации и работы потоков в Java, управление приоритетами потоков и синхронизацией потоков.