

Лабораторная работа №4

Технология программирования

*«Многопотоковые приложения»*

Студент: Молнер В.С.

Группа: АВТ-814

Вариант: 4

Преподаватель: Михайленко Д.А.



НОВОСИБИРСК 2020

Оглавление

[Задание 1](#_Toc38223051)

[Ход работы 1](#_Toc38223052)

[Вывод 1](#_Toc38223053)

# Задание

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

* создать абстрактный класс BaseAI, описывающий «интеллектуальное поведение» объектов. Класс должен создавать поток, обеспечивающий движения объектов коллекции;
* реализовать класс BaseAI для каждого из видов объекта, включив в него поведение, описанное в индивидуальном задании по варианту;
* синхронизовать работу потоков расчета интеллекта объектов, их рисования и генерации новых объектов. Рисование должно остаться в основном потоке;
* добавить в панель управления кнопки для остановки и возобновления работы интеллекта каждого вида объектов.
* добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

***Вариант 4***

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

Альбиносы двигаются по оси X от одного края области обитания до другого со скоростью V.

# Ход работы

Был создан абстрактный класс BaseAI:

public class BaseAI extends Thread{  
 protected int Speed = 1;  
}

Затем были созданы два дочерних класса описывающие движение каждого из кроликов по заданию варианта.

Обыкновенные:

public class OrdinaryAI extends BaseAI{  
  
 private int N = 300, cumN = 0;  
 private int period = 5;  
  
 public boolean isAlive = false;  
  
 Timer timer = new Timer();  
  
 @Override  
 public void run() {  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 synchronized (Singleton.*getVector*()) {  
 if (isAlive) {  
 for (Rabbit rabbit : Singleton.*getVector*()) {  
 if (rabbit instanceof Ordinary) {  
 if (cumN % N == 0) {  
 ((Ordinary) rabbit).targetX = (int) (Math.*random*() \* Habitat.*gui*.jPanelImage.getWidth());  
 ((Ordinary) rabbit).targetY = (int) (Math.*random*() \* Habitat.*gui*.jPanelImage.getHeight());  
 cumN = 0;  
 }  
 int dx = (((Ordinary) rabbit).targetX - rabbit.getX());  
 int dy = (((Ordinary) rabbit).targetY - rabbit.getY());  
 int length = (int) (Math.*sqrt*(Math.*pow*(dx, 2) + Math.*pow*(dy, 2)));  
 int time = length / (Speed \* 2); //за такое кол-во секунд будет достигнута цель  
  
 rabbit.setX(rabbit.getX() + (time == 0 ? 0 : dx / time));  
 rabbit.setY(rabbit.getY() + (time == 0 ? 0 : dy / time));  
 }  
 }  
 cumN += period;  
 }  
 }  
 }  
 },0,period);  
 }  
}

И для альбиносов:

public class AlbinoAI extends BaseAI{  
  
 boolean isActive = false;  
  
 Timer timer = new Timer();  
  
 @Override  
 public void run() {  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 synchronized (Singleton.*getVector*()) {  
 if (isActive) {  
 for (Rabbit rabbit : Singleton.*getVector*()) {  
 if (rabbit instanceof Albino) {  
 if (((Albino) rabbit).DirectionIsToRight) {  
 if (rabbit.getX() + Speed < Habitat.*gui*.jPanelImage.getWidth()) {  
 int newX = rabbit.getX() + Speed;  
 rabbit.setX(newX);  
 } else {  
 ((Albino) rabbit).DirectionIsToRight = false;  
 }  
 } else {  
 if (rabbit.getX() - Speed > 0) {  
 int newX = rabbit.getX() - Speed;  
 rabbit.setX(newX);  
 } else {  
 ((Albino) rabbit).DirectionIsToRight = true;  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
 },0,5);  
 }  
}

Движение объектов возникает за счёт того, что каждому кролику задаются новые координаты на отрисовку.

Пришлось исправить все классы: изначально координаты каждого из объекта задавились переменной типа double, теперь int.

Также были добавлены кнопки в панель управления:

jButtonAlbinoAI.addActionListener(e -> {  
 albinoAI.setPriority(jComboBoxAlbinoPriority.getSelectedIndex());  
 albinoAI.isActive = !albinoAI.isActive;  
});  
  
jButtonOrdinaryAI.addActionListener(e -> {  
 ordinaryAI.setPriority(jComboBoxOrdinaryPriority.getSelectedIndex());  
 ordinaryAI.isAlive = !ordinaryAI.isAlive;  
});

И выпадающие списки для задания приоритета потока.

for (int i = 0; i < 11; i++) {  
 jComboBoxOrdinaryPriority.addItem(i);  
 jComboBoxAlbinoPriority.addItem(i);  
}  
jComboBoxOrdinaryPriority.setSelectedIndex(6);  
jComboBoxAlbinoPriority.setSelectedIndex(2);

# Вывод

В процессе выполнения лабораторной работы были изучены принципы многопоточного программирования и синхронизация потоков. Был доработан проект из предыдущих лабораторных работ.