МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

НГТУ

Лабораторная работа №4

по предмету «Технология программирования»

на тему «Многопотоковые приложения»

Факультет: АВТ Преподаватель:

Группа: АВТ-809 Михайленко Д.А.

Студент: Швыдко Е.А.

Новосибирск 2020

Задание:

***Вариант 11***

Кошки двигаются по периметру области симуляции со скоростью V1 . Собаки их догоняют со скоростью V2.

Цель работы:

1. Изучить особенности реализации и работы потоков в Java, управлением приоритетами потоков и синхронизацией потоков.
2. Доработать программу, созданную в лабораторной работу № 3:
3. создать абстрактный класс BaseAI, описывающий «интеллектуальное поведение» объектов. Класс должен создавать поток, обеспечивающий движения объектов коллекции;
4. реализовать класс BaseAI для каждого из видов объекта, включив в него поведение, описанное в индивидуальном задании по варианту;
5. синхронизовать работу потоков расчета интеллекта объектов, их рисования и генерации новых объектов. Рисование должно остаться в основном потоке;
6. добавить в панель управления кнопки для остановки и возобновления работы интеллекта каждого вида объектов. Реализовать через засыпание/пробуждение потоков (методы wait() и notify());
7. добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

Ход разработки:

* Был разработан абстрактный класс BaseAI
* Были разработаны классы-наследники Base AI: CatAI и DogAI, реализующие поведение объектов типа Cat и Dog соответственно:

**public** **class** CatAI **extends** BaseAI {

boolean isWait = **true**;

**@Override**

**public** void run() {

**while**(**true**){

**try** {

**Thread**.sleep(30);

} **catch** (**InterruptedException** e) {

e.printStackTrace();

}

**synchronized** (**this**) {

**try** {

**while**(isWait) wait();

} **catch** (**InterruptedException** e) {

e.printStackTrace();

}

}

move();

}

}

**@Override**

**public** **synchronized** void move() {

**for**(int i = 0; i < Singleton.getVector().getArray().size(); i++) {

Animal animal = Singleton.getVector().getArray().get(i);

**if** (animal **instanceof** Cat) {

((Cat)animal).move();

}

}

}

**public** boolean isWait() {**return** isWait;}

**synchronized** void mySuspend() {

isWait = **true**;

}

**synchronized** void myResume() {

isWait = **false**;

notify();

}

}

* В классах Cat и Dog были реализованы методы для осуществления движения объектов:

@Override

**public** void move(){

**if**(((**this**.X >= Habitat.WIDTH/2) && (**this**.Y >= Habitat.HEIGHT/2)) && (!**this**.isMoving)) **return**;

**if**(!(**this**.Y == **this**.targetY && **this**.X == **this**.targetX)) {

setMoving(**true**);

int distX = **this**.targetX - **this**.X;

int distY = **this**.Y - targetY;

int tempX = (int) (**this**.X + speed \* distX / (Math.sqrt(distX \* distX + distY \* distY)));

int tempY = (int) (**this**.Y - speed \* distY / (Math.sqrt(distX \* distX + distY \* distY)));

setX(tempX);

setY(tempY);

}

}

* В панель управления были добавлены выпадающие списки для выставления приоритетов каждого из потоков

JLabel catAIPrLabel = new JLabel("Приоритет кош.");

JLabel dogAIPrLabel = new JLabel("Приоритет соб.");

add(catAIPrLabel);

add(dogAIPrLabel);

catAIPrLabel.setBounds(250,45,130,13);

dogAIPrLabel.setBounds(385,45,130,13);

String[] catAIValues = {"1","2", "3", "4", "5", "6", "7", "8", "9", "10"};

JComboBox catAIPrComboBox = new JComboBox(catAIValues);

JComboBox dogAIPrComboBox = new JComboBox(catAIValues);

catAIPrComboBox.setSelectedIndex(4);

dogAIPrComboBox.setSelectedIndex(4);

add(catAIPrComboBox);

add(dogAIPrComboBox);

catAIPrComboBox.setBounds(250,60,130,20);

dogAIPrComboBox.setBounds(385,60,130,20);

catAIPrComboBox.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

int kk = **Integer**.parseInt((**String**) **Objects**.requireNonNull(catAIPrComboBox.getSelectedItem()));

catThread.setPriority(kk);

}

});

dogAIPrComboBox.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

int kk = **Integer**.parseInt((**String**) **Objects**.requireNonNull(dogAIPrComboBox.getSelectedItem()));

dogThread.setPriority(kk);

}

});

Выводы:

В результате выполнения работы были изучены особенности реализации работы потоков.