МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОСИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

**Лабораторная работа №4**

**По дисциплине «Технология программирования»**

**на тему**

**Многопотоковые приложения**

Группа: АВТ-808

Студент: Попов Д.А.

Новосибирск, 2020

***Задание:***

1. Изучить особенности реализации и работы потоков в Java, управлением приоритетами потоков и синхронизацией потоков.

2. Доработать программу, созданную в лабораторной работу № 3:

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 засыпание/пробуждение потоков (методы wait() и notify());

5) добавить в панель управления выпадающие списки для выставления приоритетов каждого из потоков.

Вариант 7

1. Капитальные дома двигаются (в городах будущего и не такое возможно) в левую верхнюю четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке 0;0, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.

2. Деревянные дома после генерации начинают двигаться в нижнюю правую четверть области симуляции (т.е. прямоугольник с верхним-левым углом в точке w/2;h/2, шириной/длиной = (w/2;h/2), где w и h – ширина и длина области симуляции) со скоростью V по прямой. Конечная точка движения – случайная точка в пределах этой области. Если дом сгенерировался сразу в этой области, то он никуда не движется. По прибытии в конечную точку дом больше не движется.

***Описание доработки программы:***

Создан абстрактный класс BaseAI, наследник класса Thread:

public abstract class BaseAI extends Thread {  
 protected boolean isMoving;  
 protected Thread thread;  
  
 public synchronized void startAI() {}  
 public void stopAI() {}  
 public void setTheadPriority(int priority){}  
}

Созданы наследники класса BaseAI – CapitalHouseAI и WoodenHouseAI:

public class CapitalHouseAI extends BaseAI {  
  
 public CapitalHouseAI() {  
 thread = new Thread(this);  
 isMoving = true;  
 thread.start();  
 }  
  
 @Override  
 public void run() {  
 while (true){  
 synchronized (this){  
 while (!isMoving){  
 try {  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 for(int i = 0; i < Singleton.*getSingleton*().getHouseVector().size(); i++){  
 if(Singleton.*getSingleton*().getHouse(i) instanceof CapitalHouse){  
 Singleton.*getSingleton*().getHouse(i).move();  
 }  
 }  
 try {  
 Thread.*sleep*(100);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
  
 @Override  
 public synchronized void startAI() {  
 isMoving = true;  
 notify();  
 }  
  
 @Override  
 public void stopAI() { isMoving = false; }  
  
 @Override  
 public void setTheadPriority(int priority) { thread.setPriority(priority); }  
}

В классы CapitalHouse и WoodenHouse добавлен метод move() для движения объектров:

@Override  
public void move() {  
 if(inArea()){  
 return;  
 }  
 if(getX() >= 70 && getY() >= 70){  
 setPosition(getX() - speed, getY() - speed);  
 }  
 else if (getX() >= 70){  
 setX(getX() - speed);  
 }  
 else if(getY() >= 70){  
 setY(getY() - speed);  
 }  
}

Метод inArea() проверяет находится ли объект в своем конечном квадрате:

@Override  
public boolean inArea() { return getX() < 1178 / 2 && getY() < 720 / 2; }

Для каждого объекта скорость рассчитывается по формуле:

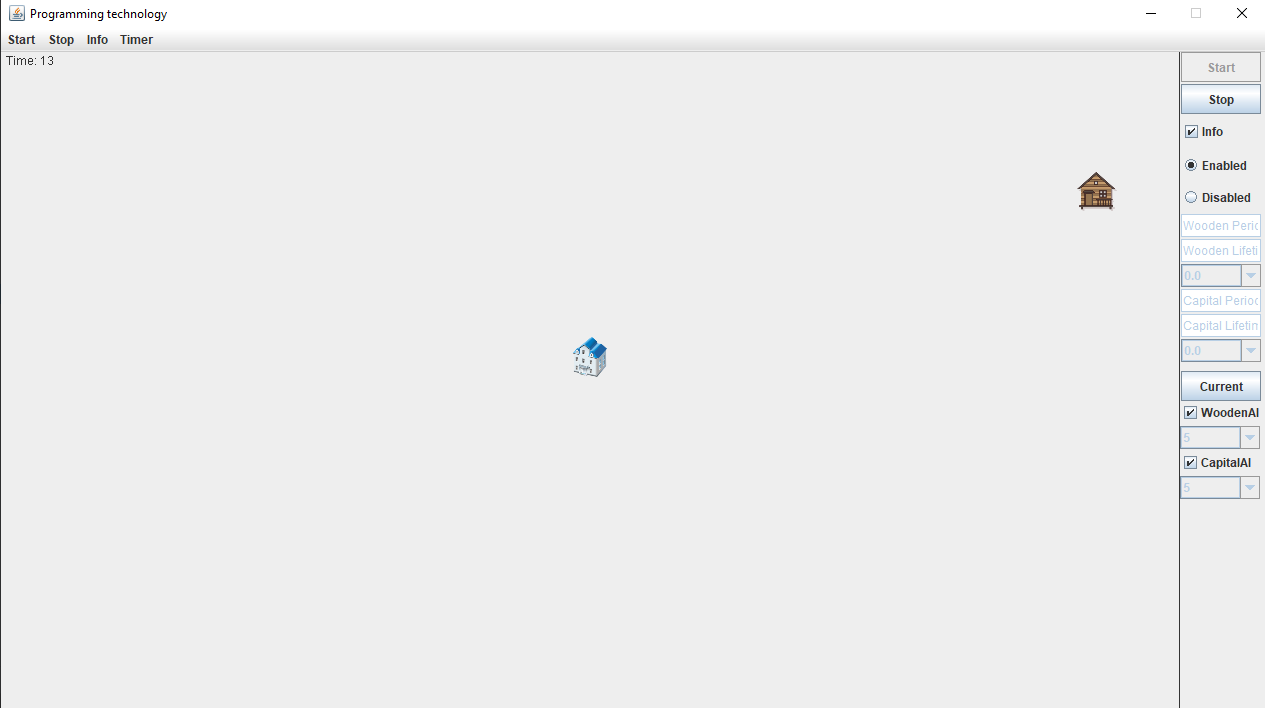
speed = (int) (Math.*random*() \* ((Math.*random*() \* 10) % 4) \* 10);

Добавлены чекбоксы для управления движением объектов:

private JCheckBox jCheckBoxWoodenHouseAI = new JCheckBox("WoodenAI", true);  
private JCheckBox jCheckBoxCapitalHouseAI = new JCheckBox("CapitalAI", true);

Добавлены комбобоксы для задания приоритета потоков:

private Integer[] selectionPriorityStep = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
  
private JComboBox<Integer> jComboBoxWoodenPriority = new JComboBox<>(selectionPriorityStep);  
private JComboBox<Integer> jComboBoxCapitalPriority = new JComboBox<>(selectionPriorityStep);



*Рис.1 – Демонстрация работы программы*

**Вывод:**

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

**Программный код:**

https://github.com/AVT-808/AVT-808/tree/Popov-Dmitry-7