Частное учреждение образования

«Колледж бизнеса и права»

Тема: «Параллельное программирование на языке Java»

Учащийся М.Н. Гончаров.

группа: Т-992

2022

Выполните задания:

Задание No1: Об “обедающих философах”

Для реализации потребуется пять процессоров. Суть задачи

следующая: пять философов сидят за круглым столом. Они проводят жизнь,

чередуя приемы пищи и размышления. В центре стола находится большое

блюдо спагетти. Философам, чтобы съесть порцию спагетти, требуется две

вилки. Вилок всего пять: между каждой парой философов лежит по одной

вилке. Каждому философу дозволительно пользоваться только вилками,

которые лежат рядом с ним (слева и справа).

Задача – написать программу, моделирующую поведение философов.

Программа должна избегать ситуации, в которой все философы голодны, то

есть ни один из них не может взять себе две вилки (например, когда каждый

философ держит по одной вилке и не хочет отдавать ее). Раз вилок всего

пять, то одновременно могут есть не более, чем двое философов. Два

сидящих рядом философа не могут есть одновременно. Предположим, что

периоды раздумий и приемов пищи различны – для их имитации в программе

можно использовать генератор случайных чисел. Имитация поведения

каждого философа может быть разбита на следующие блоки: поразмыслить,

взять вилки, поесть, отдать вилки. Вилки являются разделяемым ресурсом.

// main

///////////////////////// Предположительно что в правую руку не попадает то нико и не кушает, нужно найти в чем причина ++ РЕШЕНО

package core;

import java.util.ArrayList;

////////////////////////////////////////////// Например ID потока будет ID философа - Thread.currentThread().getId()

import java.util.Random;

class Philosopher implements Runnable{

private boolean left\_hand, right\_hand;

public boolean isRight\_hand() {

return right\_hand;

}

public void setRight\_hand(boolean right\_hand) {

this.right\_hand = right\_hand;

}

// Совойства

public boolean isLeft\_hand() {

return left\_hand;

}

public void setLeft\_hand(boolean left\_hand) {

this.left\_hand = left\_hand;

}

public Philosopher(boolean left\_hand, boolean right\_hand)

{

this.left\_hand = left\_hand;

this.right\_hand = right\_hand;

}

@Override

public void run(){

try

{

do

{

// Рандомное время ожидания

Random rnd = new Random();

Thread.sleep(rnd.nextInt(4000));

if (left\_hand == true && right\_hand == true) // Две вилки в руках

{

// Например тут по ID можно посчитать какой конкретно философ и сколько раз кушал. Через флаг.

System.out.println("Философ ID = " + Thread.currentThread().getId() + " Кушает спагетти");

}

else if (left\_hand == false && right\_hand == false) // Ноль вилок в руках

{

System.out.println("Философ ID = " + Thread.currentThread().getId() + " Думает");

}

else if (left\_hand == true && right\_hand == false) // Вилка в левой руке

{

System.out.println("Философ ID = " + Thread.currentThread().getId() + " Держит вилку в левой руке");

}

else if (left\_hand == false && right\_hand == true) // Вилка в левой руке

{

System.out.println("Философ ID = " + Thread.currentThread().getId() + " Держит вилку в правой руке");

}

}while(true); // Можно например закончить когда каждый поест более чем 5 раз

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

public class Class\_main

{

public static void main(String[] args)

{

var list\_philosophs = new ArrayList<Philosopher>();

list\_philosophs.add(new Philosopher(true,true));

list\_philosophs.add(new Philosopher(false,true));

list\_philosophs.add(new Philosopher(false,false));

list\_philosophs.add(new Philosopher(true,true));

list\_philosophs.add(new Philosopher(false,false));

// Запускаю потоки философов

(new Thread (list\_philosophs.get(0))).start();

(new Thread (list\_philosophs.get(1))).start();

(new Thread (list\_philosophs.get(2))).start();

(new Thread (list\_philosophs.get(3))).start();

(new Thread (list\_philosophs.get(4))).start();

// Отдельный поток для выдачи вилок (через проверку в массиве) / и тут идет левая правая левая правая и т.д

Thread myThread\_1 = new Thread(new Runnable()

{

public void run()

{

boolean[] array\_eating = {false,false,false,false,false,false,false,false,false,false};

// Грубо говоря зануляю для новой итерации + к этому чтобы если в if не войдет то не писать else и записывать true

// Вернул чтобы поновой не генерился а то не логично

Random rnd = new Random();

while(true)

{

try

{

Thread.sleep(5000);

System.out.println("\n=======================NEW\_EATING=========================");

// Тут генерю в массиве как по условию чтобы два рядом сидяших не могли есть

// например генерю число и от него отталкиваюсь

// Еще интересно может получиться если поочередно а не сначала левые руки заполнить, хотя там все равно рандом

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////// Левая рука / Потом Правая и чередуется

int x1 = rnd.nextInt(2); // Было чисто в проверкае, но для доставерности решил сделать для каждого философа и его рук и разбил по x1, x2 и т.д

if (x1 == 1 && array\_eating[2] != true && array\_eating[array\_eating.length-2-1] != true) // чтобы последний философ в круге не держалвилку, но -2 т.к это леваю рука по массиву

{

array\_eating[0] = true;

}

else

{

// если правая и левая уже были true то зануляю

if (array\_eating[0] == true && array\_eating[1] == true)

{

array\_eating[0] = false;

}

}

if (x1 == 1 && array\_eating[array\_eating.length-1] != true && array\_eating[3] != true)

{

array\_eating[1] = true;

}

else

{

if (array\_eating[1] == true && array\_eating[2] == true)

{

array\_eating[1] = false;

}

}

int x2 = rnd.nextInt(2);

if (x2 == 1 && array\_eating[0] != true && array\_eating[4] != true) // тут и веззде чтобы 2 соседних не смогли помешать

{

array\_eating[2] = true;

}

else

{

if (array\_eating[2] == true && array\_eating[3] == true)

{

array\_eating[2] = false;

}

}

if (x2 == 1 && array\_eating[1] != true && array\_eating[5] != true)

{

array\_eating[3] = true;

}

else

{

if (array\_eating[3] == true && array\_eating[4] == true)

{

array\_eating[3] = false;

}

}

int x3 = rnd.nextInt(2);

if (x3 == 1 && array\_eating[2] != true && array\_eating[6] != true)

{

array\_eating[4] = true;

}

else

{

if (array\_eating[4] == true && array\_eating[5] == true)

{

array\_eating[4] = false;

}

}

if (x3 == 1 && array\_eating[3] != true && array\_eating[7] != true)

{

array\_eating[5] = true;

}

else

{

if (array\_eating[5] == true && array\_eating[6] == true)

{

array\_eating[5] = false;

}

}

int x4 = rnd.nextInt(2);

if (x4 == 1 && array\_eating[4] != true && array\_eating[8] != true)

{

array\_eating[6] = true;

}

else

{

if (array\_eating[6] == true && array\_eating[7] == true)

{

array\_eating[6] = false;

}

}

if (x4 == 1 && array\_eating[5] != true && array\_eating[9] != true)

{

array\_eating[7] = true;

}

else

{

if (array\_eating[7] == true && array\_eating[8] == true)

{

array\_eating[7] = false;

}

}

int x5 = rnd.nextInt(2);

if (x5 == 1 && array\_eating[6] != true && array\_eating[0] != true)

{

array\_eating[8] = true;

}

else

{

if (array\_eating[8] == true && array\_eating[9] == true)

{

array\_eating[8] = false;

}

}

if (x5 == 1 && array\_eating[7] != true && array\_eating[array\_eating.length-2-1] != true)

{

array\_eating[9] = true;

}

else

{

if (array\_eating[9] == true && array\_eating[array\_eating.length-2-1] == true)

{

array\_eating[9] = false;

}

}

// А тут просто меняю значения в свойстве

list\_philosophs.get(0).setLeft\_hand(array\_eating[0]);

list\_philosophs.get(0).setRight\_hand(array\_eating[1]);

list\_philosophs.get(1).setLeft\_hand(array\_eating[2]);

list\_philosophs.get(1).setRight\_hand(array\_eating[3]);

list\_philosophs.get(2).setLeft\_hand(array\_eating[4]);

list\_philosophs.get(2).setRight\_hand(array\_eating[5]);

list\_philosophs.get(3).setLeft\_hand(array\_eating[6]);

list\_philosophs.get(3).setRight\_hand(array\_eating[7]);

list\_philosophs.get(4).setLeft\_hand(array\_eating[8]);

list\_philosophs.get(4).setRight\_hand(array\_eating[9]);

}

catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

});

myThread\_1.start();

}

}

