Частное учреждение образования

«Колледж бизнеса и права»

Лабораторная работа № 10

Учащийся Е. А. Чужавко

группа: Т-992

2022

**Задание.**

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.

Результат программы показан на рисунке 1.

public class Main {

public static void main(String[] args) {

Philosopher[] philosophers = new Philosopher[5];

Object[] forks = new Object[philosophers.length];

int c=1;

System.out.println("Количество 'кругов' ужина: "+c);

for (int i = 0; i < forks.length; i++) {

forks[i] = new Object();

}

for (int i = 0; i < philosophers.length; i++) {

Object leftFork = forks[i];

Object rightFork = forks[(i + 1) % forks.length];

if (i == philosophers.length - 1) {

philosophers[i] = new Philosopher(c, rightFork, leftFork);

}

else {

philosophers[i] = new Philosopher(c, leftFork, rightFork);

}

Thread t = new Thread(philosophers[i], "Philosopher " + (i + 1));

t.start();

}

}

}

public class Philosopher implements Runnable {

private Object leftFork;

private Object rightFork;

private int cc;

public Philosopher(int c,Object leftFork, Object rightFork) {

this.leftFork = leftFork;

this.rightFork = rightFork;

cc=c;

}

private void doAction(String action) throws InterruptedException {

System.out.println( Thread.currentThread().getName() + ": " + action);

Thread.sleep(((int) (Math.random() \* 100)));

}

@Override

public void run() {

try {

int p=0;

while (p<cc) {

doAction("Думает...");

synchronized (leftFork) {

doAction("Взял левую вилку.");

synchronized (rightFork) {

doAction("Взял правую вилку - ест...");

doAction("Положил правую вилку.");

}

doAction("Положил левую вилку. Опять думает...");

}

p++;

}

}

catch (InterruptedException e) {

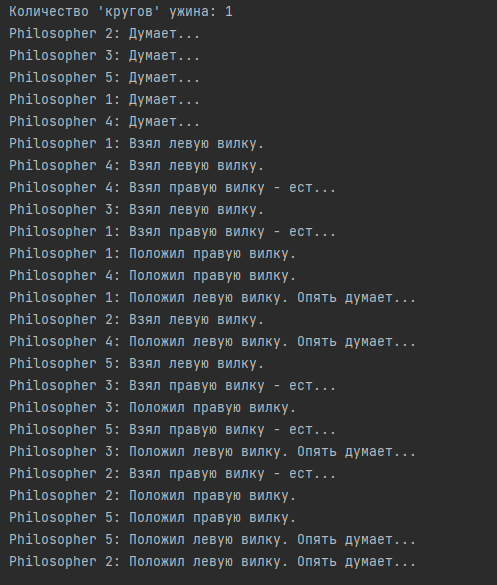
Thread.currentThread().interrupt();

return;

}

}

}



*Рисунок 1.*