Министерство науки и высшего образования Российской ФедерацииФедеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательскийуниверситет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра «Теоретическая информатика и компьютерные технологии»

Отчёт о лабораторной работе № 4 по курсу

«Разработка параллельных и

распределенных программ»

Многопоточность

Студент: Белецкий В. С.

Группа: ИУ9-51Б

Преподаватель: Царев А.С.

Москва, 2022

Содержание

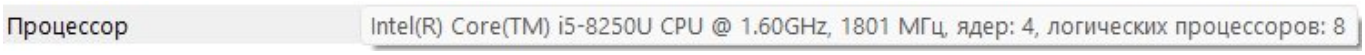
Постановка задачи	2
Практическая реализация	3
Результаты	12

Постановка задачи

Вариант 2. Задача о пяти обедающих философах.
Пять философов сидят за круглым столом. Они проводят жизнь, чередуя приёмы пищи и размышления. В центра стола находится большое блюдо спагетти. Чтобы съесть порцию, каждому философу нужно две вилки. Однако, вилок всего пять: между каждой парой рядом сидящих философов лежат по одной вилке, и каждый философ может пользоваться только теми вилками, которые лежат рядом с ним, слева и справа. Философ не может брать две вилки одновременно: сначала он тратит некоторое время на то, чтобы взять одну, затем вторую. Однако, он может одновременно положить их на место. Задача заключается в том, чтобы написать программу, моделирующую поведение философов. Очевидно, что раз вилок всего пять, то одновременно есть могут не более двух философов, и два сидящих рядом философа не могут есть одновременно. Для имитации периодов раздумий и приёмов пищи можно использовать генератор случайных чисел, позволяющий задавать времена их действий в определённом интервале. Имитация поведения каждого философа, по сути, разбивается на то, что в любой момент времени философ находится в одном из пяти состояний: размышляет, берёт левую вилку, берёт правую вилку, ест, кладёт вилки на место. Таким образом, вилки являются разделяемым ресурсом.
На программу накладываются условия:
1. Каждый философ, по сути, является потоком, и модель поведения у каждого из них должна быть одинаковой, кроме того, какие вилки они могут брать.
2. Накладывание блокировки по сути является действием по взятию вилки, поэтому накладывать блокировку сразу на обе вилки нельзя; последовательность действий должна быть «наложить блокировку – взять вилку – наложить вторую блокировку – взять вторую вилку».
3. Программа должна избегать ситуации взаимоблокировки: ситуации, в которой все философы голодны, то есть ни один из них не может взять себе две вилки (например, когда каждый держит по одной и не хочет её отдавать).
Запрограммировать остановку алгоритма по достижении контрольного времени (например, атомарной операцией над булевым флагом).

Практическая реализация

Лабораторная работа выполнялась на ноутбуке с установленной ОС Windows 11, процессор имеет характеристики, указанные на скриншоте:



Для реализации поставленной задачи использовался язык программирования Java.

Код программы :

```
import java.util.Date;
import java.util.concurrent.locks.ReentrantLock;

public class Philosophers {

   static int philosopherNum = 5;
   static Philosopher[] philosophers = new Philosopher[philosopherNum];
   static Fork[] Forks = new Fork[philosopherNum];

   static class Fork {
      public ReentrantLock mutex = new ReentrantLock();
      void grab() {
         try {
            mutex.tryLock();
         }
         catch (Exception e) {
            e.printStackTrace(System.out);
         }
```

```
    }

    void release() {
        mutex.unlock();
    }

    boolean isFree() {
        return !mutex.isLocked();
    }
}


static class Philosopher extends Thread {
    public int number;
    public Fork leftFork;
    public Fork rightFork;

    Philosopher(int num, Fork left, Fork right) {
        number = num;
        leftFork = left;
        rightFork = right;
    }

    public void run() {
        while (true) {
```

```
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            int state = (int)(Math.random() * 5);
            switch (state) {
                case 0:
                    if (!(leftFork.mutex.isHeldByCurrentThread()) && leftFork.isFree()) {
                        leftFork.grab();
                        System.out.println(System.nanoTime() + ": " + "Философ №" + (number + 1)
+ " берёт левую вилку.");
                    }
                case 1:
                    if (!(rightFork.mutex.isHeldByCurrentThread()) && rightFork.isFree()) {
                        rightFork.grab();
                        System.out.println(System.nanoTime() + ":  " + "Философ №" + (number + 1)
+ " берёт правую вилку.");
                    }
                case 2:     // если есть обе вилки, то есть
                    if (leftFork.mutex.isHeldByCurrentThread() &&
rightFork.mutex.isHeldByCurrentThread()) {
                        int sleepTime = (int)(Math.random() * 3000);
                        System.out.println(System.nanoTime() + ":  " + "Философ №" + (number+1)
+ " собирается есть в течение " + sleepTime +"мс");
                        try {
                            Thread.sleep(sleepTime);
```

```
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
                break;
            case 3:
                if (leftFork.mutex.isHeldByCurrentThread()) {
                    leftFork.release();
                    System.out.println(System.nanoTime() + ":  " + "Философ №" + (number + 1)
+ " кладёт левую вилку на место.");
                }
                if (rightFork.mutex.isHeldByCurrentThread()) {
                    rightFork.release();
                    System.out.println(System.nanoTime() + ":  " + "Философ №" + (number + 1)
+ " кладёт правую вилку на место.");
                }
                break;
            case 4:
                int sleepTime = (int)(Math.random() * 2000);
                System.out.println(System.nanoTime() + ":  " + "Философ №" + (number+1) + "
собирается размышлять в течение " + sleepTime +"мс");
                try {
                    Thread.sleep(sleepTime);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```

```
                break;
            }
        }
    }
}


public static void main(String[] args) {

    for (int i = 0; i < philosopherNum; i++) {
        Forks[i] = new Fork();
    }
    for (int i = 0; i < philosopherNum; i++) {
        philosophers[i] = new Philosopher(i, Forks[i], Forks[(i + 1) % philosopherNum]);
        philosophers[i].start();
    }
    long startTime = System.currentTimeMillis();
    long elapsedTime = 0L;
    while (elapsedTime < 30 * 1000) {
        try {
            elapsedTime = (new Date()).getTime() - startTime;
            Thread.sleep(1000);
            boolean deadlock = true;
            for (Fork fs : Forks) {
                if (fs.isFree()) {
```

```
                deadlock = false;
                break;
            }
        }
        if (deadlock) {
            for (Fork fs : Forks) {
                if (fs.mutex.isHeldByCurrentThread()) fs.mutex.unlock();
            }
            System.out.println(System.nanoTime() + ":  " + "Все кладут вилки на стол");
        }
    }
    catch (Exception e) {
        e.printStackTrace(System.out);
    }
  }
  System.exit(0);
}
```

}

Ниже приведен лог работы программы:

76967: Философ 3 берёт левую вилку

76967: Философ 4 берёт левую вилку

76972: Философ 4 берёт правую вилку

76972: Философ 4 ест в течение 1365мс

77167: Философ 1 размышляет в течение 312мс

77299: Философ 4 ест в течение 399мс

77367: Философ 2 берёт левую вилку

77367: Философ 5 размышляет в течение 2010мс

77372: Философ 3 размышляет в течение 2092мс

77392: Философ 1 берёт левую вилку

77467: Философы кладут вилки на стол

77531: Философ 4 кладёт левую вилку

77531: Философ 4 кладёт правую вилку

77753: Философ 5 размышляет в течение 2155мс

77762: Философ 3 берёт правую вилку

77762: Философ 3 ест в течение 1921мс

77792: Философ 1 размышляет в течение 1666мс

77967: Философ 2 размышляет в течение 2098мс

78134: Философ 3 ест в течение 233мс

78153: Философ 5 берёт левую вилку

78168: Философы кладут вилки на стол

78268: Философы кладут вилки на стол

78353: Философ 3 кладёт левую вилку

78353: Философ 3 кладёт правую вилку

78358: Философ 2 кладёт левую вилку

78536: Философ 1 кладёт левую вилку

78553: Философ 3 берёт правую вилку

78554: Философ 5 берёт правую вилку

78554: Философ 5 ест в течение 2817мс

78558: Философ 2 размышляет в течение 501мс

78736: Философ 1 берёт правую вилку

78931: Философ 4 размышляет в течение 1244мс

78953: Философ 3 берёт левую вилку

78953: Философ 3 ест в течение 2002мс

78969: Философы кладут вилки на стол

79010: Философ 5 ест в течение 2312мс

79069: Философы кладут вилки на стол

79136: Философ 1 кладёт правую вилку

79335: Философ 3 размышляет в течение 1432мс

79396: Философ 2 берёт левую вилку

79426: Философ 5 ест в течение 1172мс

79469: Философы кладут вилки на стол

79569: Философы кладут вилки на стол

79596: Философ 2 кладёт левую вилку

79639: Философ 4 размышляет в течение 344мс

79666: Философ 3 ест в течение 1822мс

79732: Философ 5 кладёт левую вилку

79732: Философ 5 кладёт правую вилку

79736: Философ 1 берёт левую вилку

79736: Философ 1 берёт правую вилку

79736: Философ 1 ест в течение 2083мс

79868: Философ 4 берёт правую вилку

79869: Философы кладут вилки на стол

79932: Философ 5 размышляет в течение 822мс

79969: Философы кладут вилки на стол

80041: Философ 3 ест в течение 761мс

80068: Философ 4 кладёт правую вилку

80127: Философ 1 ест в течение 2412мс

80196: Философ 2 размышляет в течение 733мс

80200: Философ 5 размышляет в течение 1822мс

80303: Философ 3 размышляет в течение 1916мс

80556: Философ 1 кладёт левую вилку

80556: Философ 1 кладёт правую вилку

80570: Философ 5 размышляет в течение 892мс

80668: Философ 4 берёт правую вилку

80683: Философ 3 ест в течение 6мс

80854: Философ 5 берёт правую вилку

80855: Философ 2 размышляет в течение 1613мс

80869: Философ 4 размышляет в течение 1526мс

80883: Философ 3 ест в течение 2128мс

81156: Философ 1 размышляет в течение 821мс

81214: Философ 2 размышляет в течение 291мс

81286: Философ 3 кладёт левую вилку

81286: Философ 3 кладёт правую вилку

81412: Философ 4 берёт левую вилку

81412: Философ 4 ест в течение 1112мс

81636: Философ 2 берёт правую вилку

81705: Философ 4 размышляет в течение 1022мс

81836: Философ 2 кладёт правую вилку

81854: Философ 5 кладёт правую вилку

81886: Философ 3 размышляет в течение 191мс

82002: Философ 4 ест в течение 1899мс

82036: Философ 2 берёт левую вилку

82036: Философ 2 берёт правую вилку

82036: Философ 2 ест в течение 2835мс

82054: Философ 5 берёт правую вилку

82070: Философы кладут вилки на стол

82170: Философы кладут вилки на стол

82254: Философ 5 размышляет в течение 1365мс

82270: Философы кладут вилки на стол

82370: Философы кладут вилки на стол

82386: Философ 4 ест в течение 1231мс

82470: Философы кладут вилки на стол

82512: Философ 2 кладёт левую вилку

82512: Философ 2 кладёт правую вилку

82583: Философ 5 кладёт правую вилку

82699: Философ 4 размышляет в течение 271мс

82712: Философ 2 берёт правую вилку

Результаты

В ходе выполнения лабораторной работы были изучены способы создания многопоточных программ. Благодаря использованию нескольких потоков была создана симуляция случайных действий нескольких человек. Были изучены методы работы с deadlock ситуациями.