

ГУАП

КАФЕДРА № 44

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

старший преподаватель  
\_\_\_\_\_  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

В. А. Ушаков  
\_\_\_\_\_  
инициалы, фамилия

### ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Многопоточное программирование

по курсу: ИТ-модуль "Разработка мобильных приложений"

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4142

\_\_\_\_\_  
подпись, дата

К.С. Некрасов  
\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2023

## Вариант 21

$$(4142 + 14) \% 22 + 1 = 21$$

### Задание

Суть алгоритма поликлиники заключается в том, что при выборе потоком себе номера, он не обращается к общему счетчику, а выбирает наибольший среди тех, что имеются у других потоков. При определении следующей очереди, так же как и в предыдущем алгоритме, ее получает поток с наименьшим номером.

### Листинг кода

#### Main.kt

```
import java.util.concurrent.ExecutorService
import java.util.concurrent.Executors
import java.util.concurrent.TimeUnit
import kotlin.random.Random

fun main() {
    // Создаём поликлинику
    val polyclinic = Polyclinic()

    // Создаём пациентов
    val patients = mutableListOf(
        Patient("Кирилл"),
        Patient("Алексей"),
        Patient("Даниил"),
        Patient("Дмитрий"),
        Patient("Максим"),
        Patient("Андрей"),
        Patient("Артем"),
        Patient("Иван"),
        Patient("Михаил"),
    )

    println("Пациенты пришли в поликлинику: $patients")

    // Создаём пул потоков (по одному потоку на каждого пациента + один п
    val executor: ExecutorService =
        Executors.newFixedThreadPool(patients.size + 1)
```

```

// Потоки пациентов, которые получают номера
for (i in 0..<patients.size) {
    val patient = patients[i]
    executor.submit {
        val randomTime = Random.nextLong(0, 3000)
        Thread.sleep(randomTime)
        var number = 0
        for (j in 0..<patients.size) {
            if (patients[j].getNumber() > number) {
                number = patients[j].getNumber()
            }
        }
        patient.setNumber(number + 1)
    }
}

// Поток поликлиники, который обрабатывает пациентов
executor.submit {
    Thread.sleep(3000)
    while (patients.isNotEmpty()) {
        var minPat = patients[0]
        for (i in 0..<patients.size) {
            if (patients[i].getNumber() < minPat.getNumber()) {
                minPat = patients[i]
            }
        }
        polyclinic.process(minPat)
        patients.remove(minPat)
        Thread.sleep(1000)
    }
}

// Ждём пока все потоки завершатся
executor.shutdown()
try {
    executor.awaitTermination(
        Long.MAX_VALUE,
        TimeUnit.NANOSECONDS
    )
}

```

```

        } catch (e: InterruptedException) {
            e.printStackTrace()
        }
    }
}

```

### **Patient.kt**

```

class Patient(val name: String) {

    private var number: Int = 0;

    fun setNumber(number: Int) {
        println("Пациент ${name} получил номер: ${number}")
        this.number = number;
    }

    fun getNumber(): Int {
        return number;
    }

    override fun toString(): String {
        return "$name[$number]"
    }
}

```

### **Polyclinic.kt**

```

class Polyclinic {
    fun process(patient: Patient) {
        println("Принимаем пациента $patient...")
    }
}

```

## Результат работы программы

```
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Пациенты пришли в поликлинику: [Кирилл[0], Алексей[0], Даниил[0], Дмитрий[0], Максим[0], Андрей[0], Артем[0], Иван[0], Михаил[0]]
Пациент Дмитрий получил номер: 1
Пациент Андрей получил номер: 2
Пациент Даниил получил номер: 3
Пациент Иван получил номер: 4
Пациент Кирилл получил номер: 5
Пациент Максим получил номер: 6
Пациент Михаил получил номер: 7
Пациент Артем получил номер: 8
Пациент Алексей получил номер: 9
Принимаем пациента Дмитрий[1]...
Принимаем пациента Андрей[2]...
Принимаем пациента Даниил[3]...
Принимаем пациента Иван[4]...
Принимаем пациента Кирилл[5]...
Принимаем пациента Максим[6]...
Принимаем пациента Михаил[7]...
Принимаем пациента Артем[8]...
Принимаем пациента Алексей[9]...

Process finished with exit code 0
```