МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «Белгородский государственный национальный исследовательский университет»

Лабораторная работа №7
дисциплина «Современные методы, среды и языки программирования»
по теме «Разработка простых многопоточных программ в Java»

Выполнил: студент группы 12002135 Макаров Д.С.
Проверил:

Белгород 2021

Лабораторная работа №7

«Разработка простых многопоточных программ в Java»

Цель работы:Научиться создавать программный код решения типовых задач на языке Java с применением принципа многопоточного исполнения команд.

Задание I

Автостоянка. Доступно несколько машиномест. На одном месте может находиться только один автомобиль. Если все места заняты, то автомобиль не станет ждать больше определенного времени и уедет на другую стоянку.

Пусть время между появлением нового автомобиля на стоянке определяется по показательному закону распределения СВ с интенсивностью $\lambda = 0.02$.

Ход работы

Пример вывода программы

```
[mda@mda-notebook lab7]$ go run main.go
car ab2a9128-f665-4358-9286-9c432670b1f9, patiente 261, reserve 37
ab2a9128-f665-4358-9286-9c432670b1f9 take place 1
create parking place 1car 50c0848e-6bb7-4c93-8176-5193cc39dc6b, patiente 19, reserve 245
50c0848e-6bb7-4c93-8176-5193cc39dc6b take place 2
create parking place 2car 0e550c12-483e-480e-bf03-70080d931be3, patiente 289, reserve 138
0e550c12-483e-480e-bf03-70080d931be3 take place 3
create parking place 3car f3b6ec10-6573-4f8a-82df-b25616a040f3, patiente 169, reserve 46
f3b6ec10-6573-4f8a-82df-b25616a040f3 take place 1
ab2a9128-f665-4358-9286-9c432670b1f9 leave place 1
car 71afa6f7-eed0-47e2-b19e-372adabcbda1, patiente 77, reserve 74
50c0848e-6bb7-4c93-8176-5193cc39dc6b leave place 2
71afa6f7-eed0-47e2-b19e-372adabcbda1 take place 2
car 6e106c47-ce0f-406a-941a-0382336a8168, patiente 91, reserve 286
6e106c47-ce0f-406a-941a-0382336a8168 take place 3
0e550c12-483e-480e-bf03-70080d931be3 leave place 3
car 206f73c4-2a09-4e06-b47c-f56a69a58459, patiente 81, reserve 160
206f73c4-2a09-4e06-b47c-f56a69a58459 take place 1
f3b6ec10-6573-4f8a-82df-b25616a040f3 leave place 1
car de15e2f4-03e6-498f-b683-9b08ad5a91d1, patiente 17, reserve 98
de15e2f4-03e6-498f-b683-9b08ad5a91d1 take place 2
71afa6f7-eed0-47e2-b19e-372adabcbda1 leave place 2
car 2b7d13e2-bb23-43ab-961c-9c0919d31f2e, patiente 15, reserve 180
2b7d13e2-bb23-43ab-961c-9c0919d31f2e leave, end of patience
car 842d019c-52a9-4895-bc4e-b0dc62303af7, patiente 84, reserve 100
842d019c-52a9-4895-bc4e-b0dc62303af7 take place 3
6e106c47-ce0f-406a-941a-0382336a8168 leave place 3
```

Приложение

Содержимое файла settings.go

```
package settings

import "time"

const TickDuration time.Duration = 100 * time.Millisecond

const AutoLambda float64 = 0.02
```

Содержимое файла parking.go

```
package parking

import (
        "fmt"
        "math"
        "math/rand"
        "time"

        "github.com/google/uuid"
        expRand "golang.org/x/exp/rand"
)

type Car struct {
        id       string
        patience time.Duration
        reserve  time.Duration
}

func (c Car) Process(parking chan int, stop chan struct{}) {
        endOfpatience := make(chan struct{})

        go func() {
                defer close(endOfpatience)
                time.Sleep(time.Millisecond * c.patience)
        }()

        select {
        case place := <-parking:
                fmt.Printf("%s take place %d\n", c.id, place)
                time.Sleep(time.Millisecond * c.reserve)
                parking <- place
                fmt.Printf("%s leave place %d\n", c.id, place)
        case <-endOfpatience:
                fmt.Printf("%s leave, end of patience\n", c.id)
        case <-stop:
                fmt.Printf("parking is closed, %s leaves\n", c.id)
        }
}

func CreateTraffic(lambda float64, stop chan struct{}) <-chan Car {
        traffic := make(chan Car)
        go func() {
                defer close(traffic)
                for {
                        select {
                        case <-stop:
                                break
                        default:
                                traffic <- CreateMockCar()
                                newDuration := int64(math.RoundToEven(expRand.ExpFloat64() / lambda))
                                time.Sleep(time.Duration(newDuration) * time.Millisecond)
                        }
                }
        }()
        return traffic
}

func InitPlaces(parking chan int, stop chan struct{}, count int) {
        for i := 1; i <= count; i++ {
                select {
                case <-stop:
                        break
                default:
```

```
                        parking <- i
                        fmt.Printf("create parking place %d", i)
                }
        }
}

func CreateMockCar() Car {
        rand.Seed(time.Now().UTC().UnixNano())
        id, err := uuid.NewRandom()
        if err != nil {
                panic(err)
        }
        patiente := time.Duration(rand.Int63n(300) + 1)
        reserve := time.Duration(rand.Int63n(300) + 1)
        result := Car{id: id.String(), patience: patiente, reserve: reserve}
        fmt.Printf("car %s, patiente %d, reserve %d\n", result.id, result.patience, result.reserve)
        return result
}
```

Содержимое файла main.go

```
package main

// Автостоянка. Доступно несколько машиномест. На одном месте
// может находиться только один автомобиль. Если все места заняты, то
// автомобиль не станет ждать больше определенного времени и уедет на другую
// стоянку.

// Пусть время между появлением нового автомобиля на стоянке определяется по показательному закону распределения СВ

import (
        "lab7/pkg/parking"
        "lab7/pkg/settings"
)

func main() {
        p := make(chan int)
        stop := make(chan struct{})
        cars := parking.CreateTraffic(settings.AutoLambda, stop)
        go parking.InitPlaces(p, stop, 3)
        for {
                select {
                case car := <-cars:
                        go car.Process(p, stop)
                }
        }
}
```