Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

# Отчёт по программе «Практическое занятие 12 (корутины и потоки).»

Выполнил: Азаренко Михаил Владимирович

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2024

**Задание:**

Задание: 1.Реализовать через корутины вызов функций нажатий и

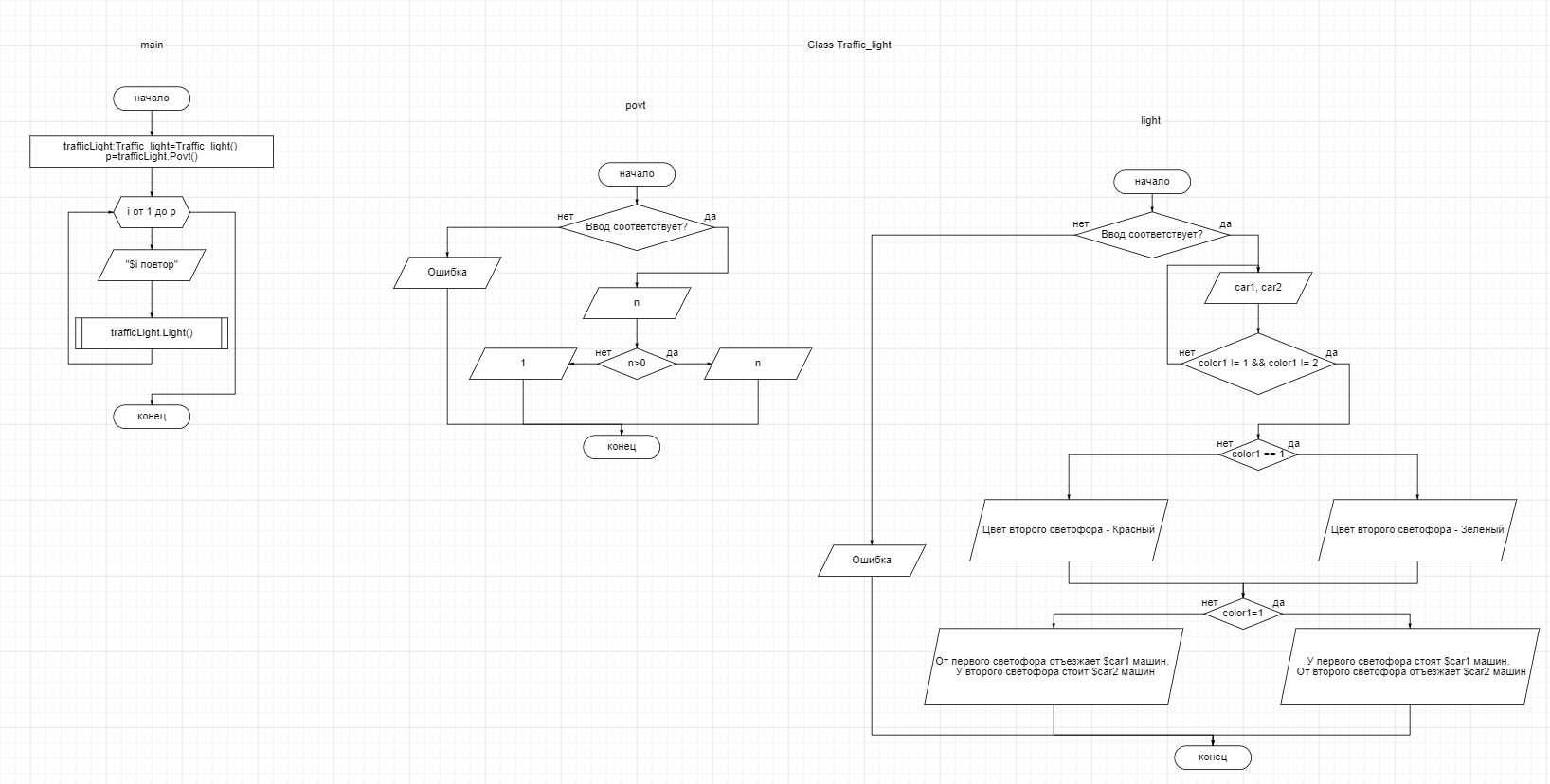
параллельного получения данных из n функций (n вводит пользователь).

Затем, нужно их объединять и отображать.

**Входные и выходные данные**

**Вход**

n-количество повторений цикла-int, car1-кол-во машин у первого светофора-int, car2- кол-во машин у второго светофора -int, color1-цвет первого светофора-int

**Блок-схема**  ****

**Листинг программы**

**Class Traffic\_light**

open class Traffic\_light {  
 fun Povt():Int{  
 try {  
 *println*("Сколько повторений вы хотите выполнить?")  
 var n=*readLine*()!!.*toInt*();  
 if (n>0)  
 return n  
 else  
 *println*("Количество повторений не может быть меньше или равно нулю. Автоматически выставлено значение = 1")  
 return 1  
 }catch (e:Exception){  
 *println*("Автоматически выставлено значение = 1")  
 return 1}  
 }  
  
 fun Light()  
 {  
 try {  
  
 var color1:Int  
 *println*("Введите количество машин у первого светофора")  
 var car1 = *readLine*()!!.*toInt*()  
 *println*("Введите количество машин у второго светофора")  
 var car2 = *readLine*()!!.*toInt*()  
 do {  
 *println*("Какой цвет у первого светофора?\n1-Красный\n2-Зелёный")  
 color1 = *readLine*()!!.*toInt*()  
 if (color1 != 1 && color1 != 2)  
 *println*("Неверный ввод, выполняется повтор действия.")  
 }while(color1!=1 && color1!=2)  
 if (color1 == 1) {  
 *println*("Цвет второго светофора - Зелёный\n")  
 } else {  
 *println*("Цвет второго светофора - Красный\n")  
 }  
 when (color1) {  
 1->*println*("У первого светофора стоят $car1 машин.\n От второго светофора отъезжает $car2 машин\n")  
 2->*println*("От первого светофора отъезжает $car1 машин.\n У второго светофора стоит $car2 машин\n")  
 }  
 }catch (e:Exception){  
 *println*("Ошибка")}

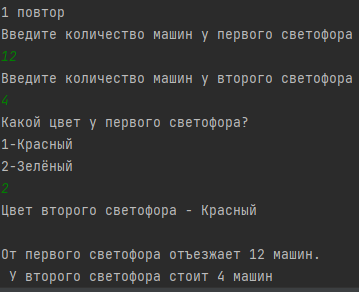
}

}

**Main**

import kotlinx.coroutines.\*  
suspend fun main(){  
 var trafficLight:Traffic\_light=Traffic\_light()  
 var p=trafficLight.Povt()  
 GlobalScope.*launch* **{** for(i in 1..p){  
 *println*("$i повтор")  
 trafficLight.Light()  
 }  
 **}** *runBlocking* **{** delay(10000L) **}**}

**Тестовые ситуации**

****

**Задание:**

2.Создайте программу для загрузки участников для всех репозиториев

данной организации, с сортировкой пользователей по количеству их

репозиториев.

Вводится имя пользователя и токен (или пароль) GitHub в соответствующие

поля. Если в раскрывающемся меню вариантов выбран вариант

«БЛОКИРОВКА», нажмите «Загрузить участников». Пользовательский

интерфейс должен зависнуть на некоторое время, а затем отобразить

отсортированный список участников с указанием количества репозиториев.

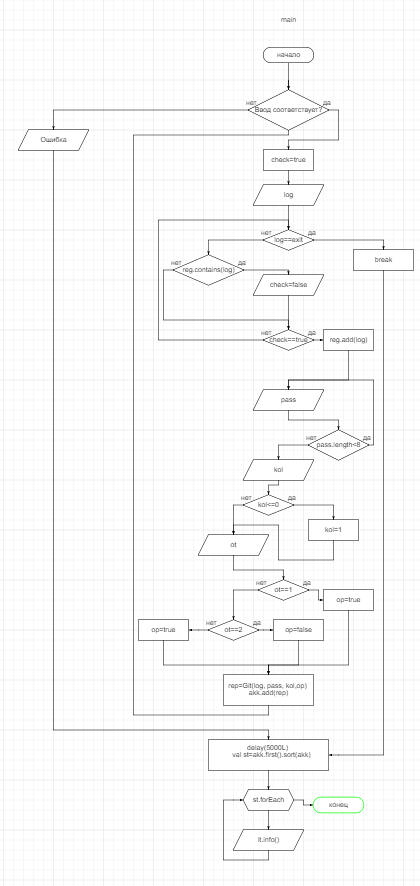
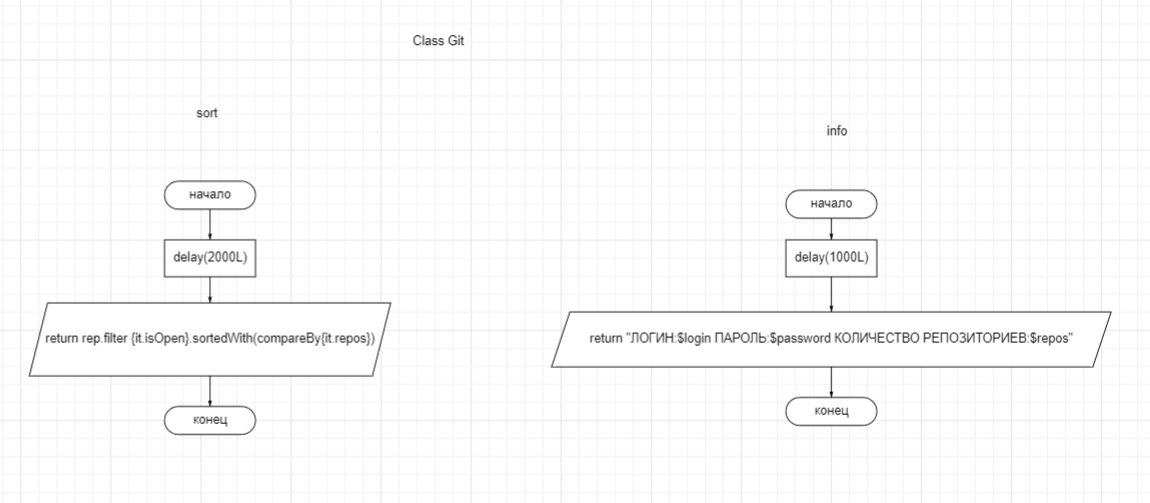
**Входные и выходные данные**

**Вход**

Log-логин(string), pass(string)-пароль, kol(int)-количество репозиториев, ot(int)-ответ на вопрос “Открыты ли репозитории”

**Выход**

Akk- отсортированный лист с внесенными данными о пользователях

**Блок-схема**  **** ****

**Листинг программы (если есть)**

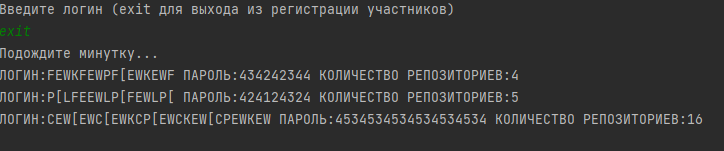
Class Git

class Git(var login:String, var password:String, var repos:Int, var isOpen:Boolean) {  
 suspend fun sort(rep:List<Git>):List<Git>{  
 delay(2000L)  
 return rep.*filter* **{it**.isOpen**}**.*sortedWith*(*compareBy***{it**.repos**}**)  
 }  
  
 suspend fun info():String  
 {  
 delay(1000L)  
 return "ЛОГИН:$login ПАРОЛЬ:$password КОЛИЧЕСТВО РЕПОЗИТОРИЕВ:$repos"  
 }  
  
}

Main

fun main()= *runBlocking***{** var reg= ArrayList<String>()  
 var akk= *mutableListOf*<Git>()  
  
 try{  
 var pass=""  
 var kol=0  
 var ot=0  
 var op=true  
 var check=true  
 while (true) {  
 check=true  
 *println*("Введите логин (exit для выхода из регистрации участников)")  
 val log = *readln*()  
 if (log == "exit") {break; check=false}  
 else  
 if (reg.contains(log)) {  
 *println*("Данный логин уже существует")  
 check=false  
 }  
 if (check==true){  
 reg.add(log)  
 *println*("Введите пароль")  
 pass= *readln*()  
 if(pass.length<8)  
 {while (pass.length<8)  
 {  
 *println*("Пароль слишком короткий, попробуйте ещё раз")  
 *println*("Введите пароль")  
 pass= *readln*()  
 }  
 }  
 *println*("Введите количество репозиториев")  
 kol=*readLine*()!!.*toInt*()  
 if (kol<=0) {  
 *println*("Недопустимая величина. Выставлено значение 1")  
 kol=1  
 }  
 *println*("Они открытые?\n1-Yes\n2-No")  
 ot= *readln*()!!.*toInt*()  
 when (ot)  
 {  
 1->{op=true}  
 2->{op=false}  
 else->{*println*("Неверный ввод, выставлено значение true")  
 op=true}  
 }  
 val rep=Git(log, pass, kol,op)  
 akk.add(rep)  
 }  
 }  
  
 }catch (e:Exception){*println*("Ошибка ввода")}  
 *println*("Подождите минутку...")  
 delay(5000L)  
 val st=akk.*first*().sort(akk)  
 st.*forEach* **{** *println*(**it**.info())  
 **}  
}**

**Тестовые ситуации**

****