Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Практическое занятие 12**»

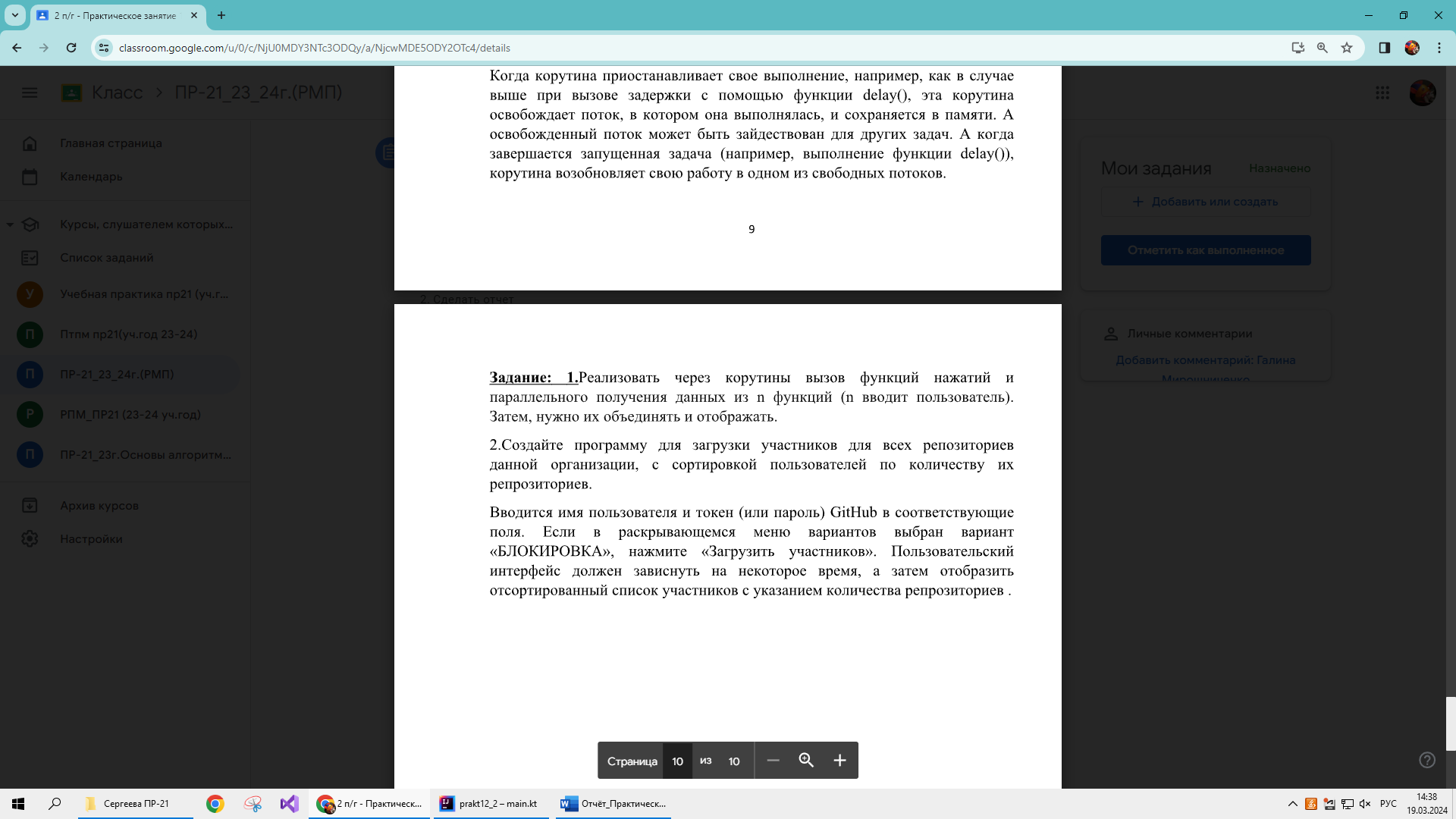
Выполнил: Сергеева Агата Андреевна

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2024 г.

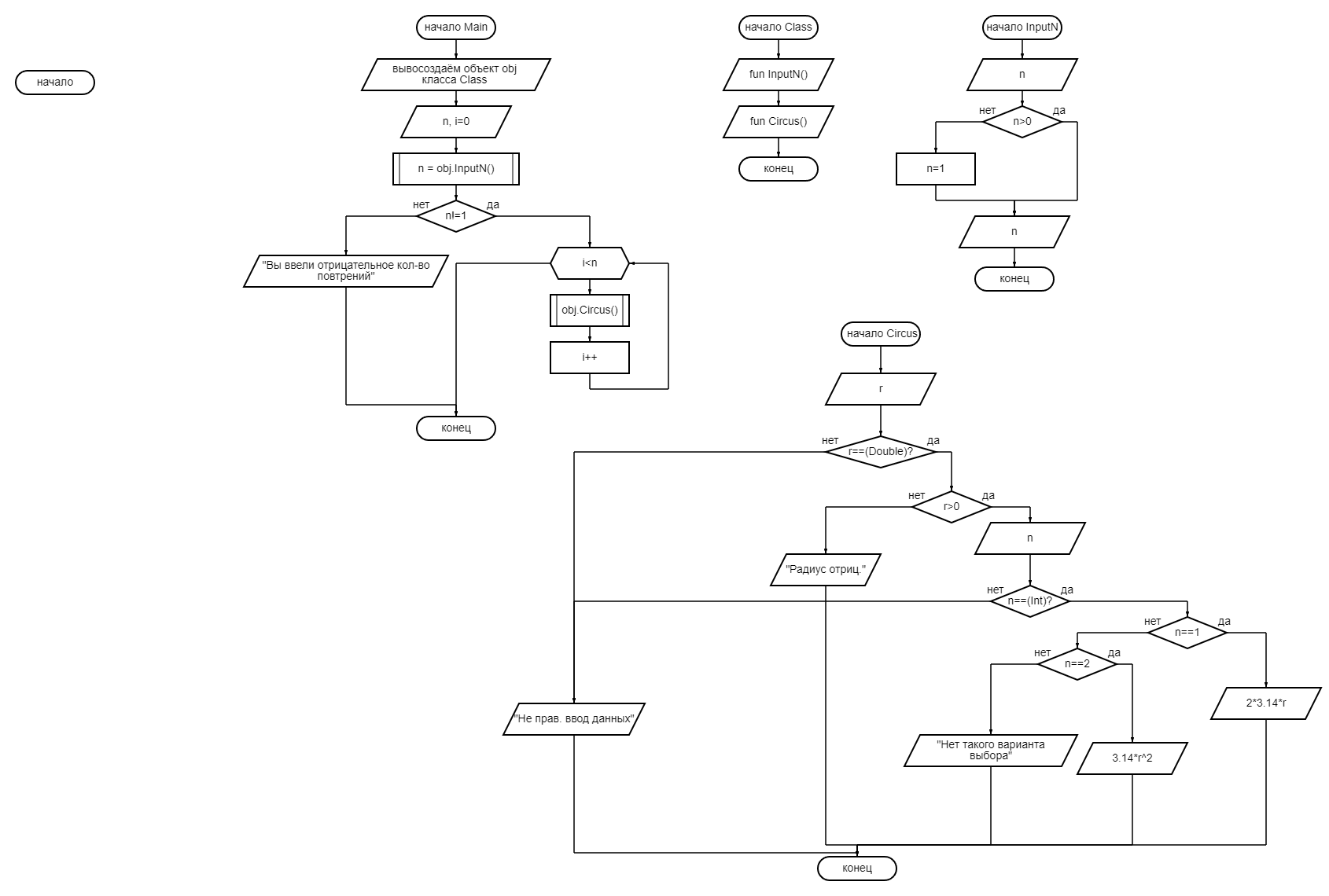
**Задание 1:**



**Входные данные:** n –количество запусков программы , int; r –радиус окружности, double; number – выбор номера операции, int.

**Выходные данные:** значение выражения, вычисленное на основе выбора пользователя

**Блок-схема:**

****

**Листинг программы:**

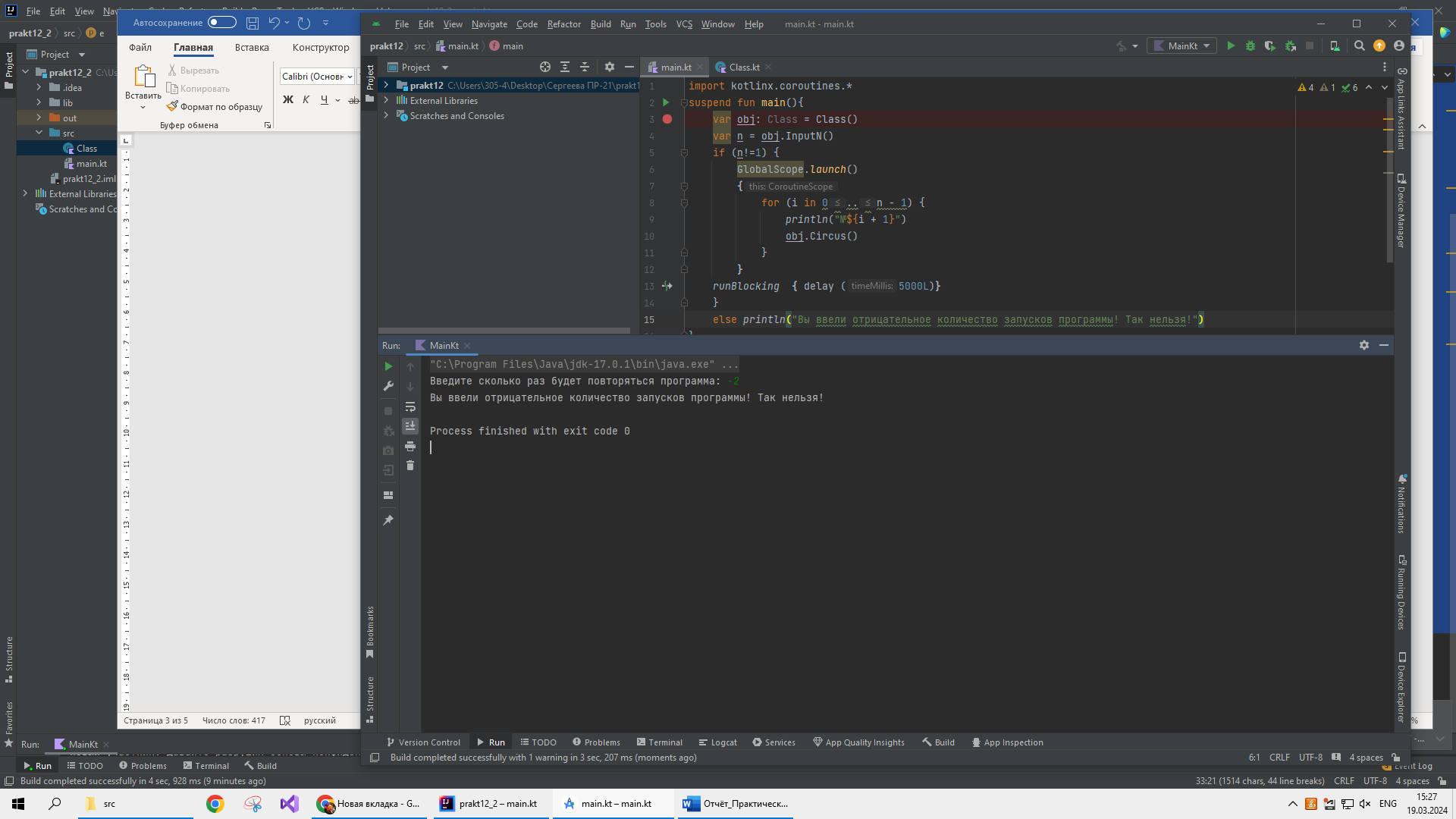
**Main:**

import kotlinx.coroutines.\*  
suspend fun main(){  
 var obj: Class = Class()  
 var n = obj.InputN()  
 if (n!=1) {  
 GlobalScope.*launch*()  
 **{** for (i in 0..n - 1) {  
 *println*("№${i + 1}")  
 obj.Circus()  
 }  
 **}** *runBlocking* **{** delay (30000L)**}** }  
 else *println*("Вы ввели отрицательное количество запусков программы! Так нельзя!")  
 *println*("THE END")  
}

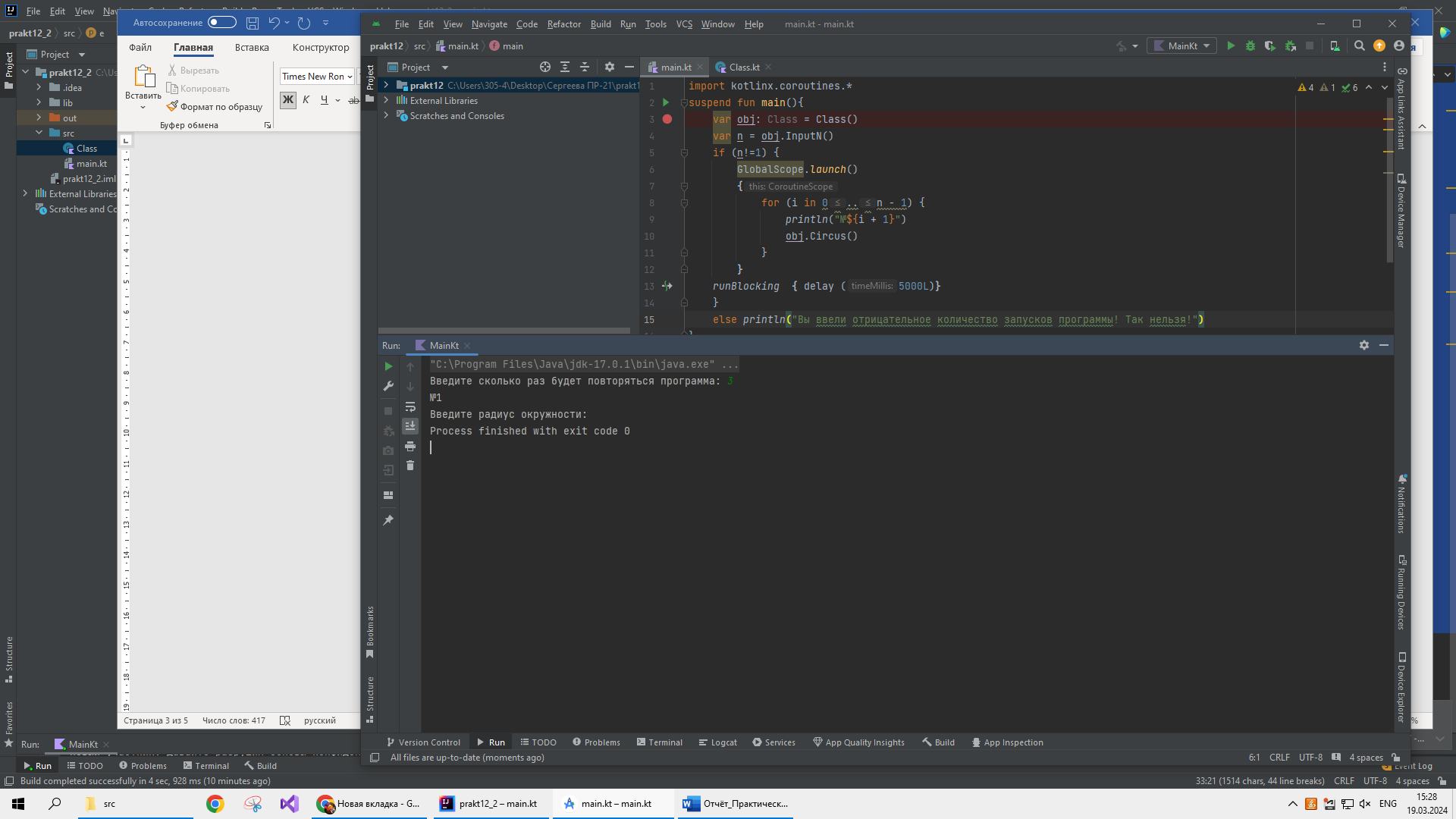
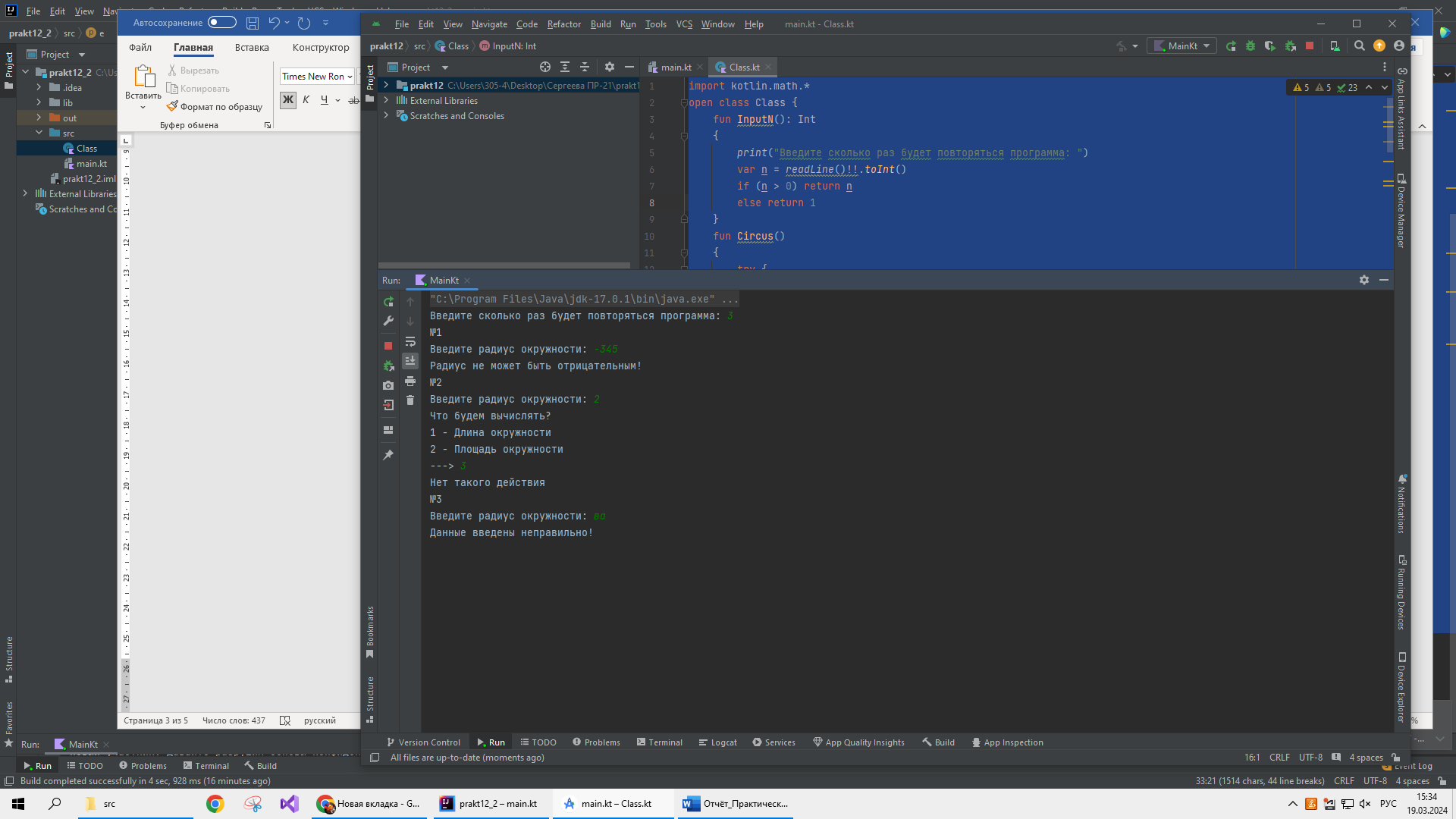
**Class:**

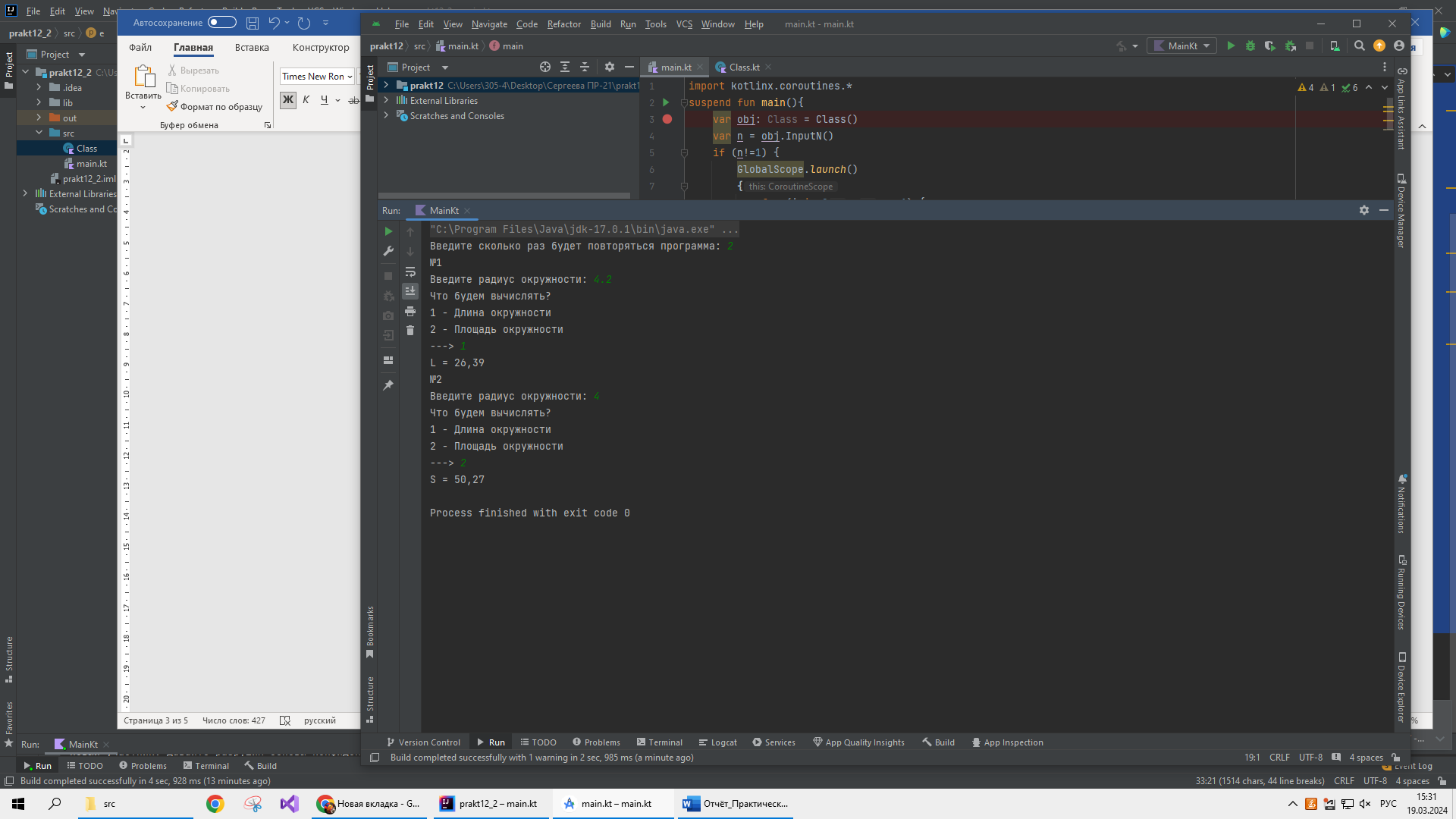
import kotlin.math.\*  
open class Class {  
 fun InputN(): Int  
 {  
 *print*("Введите сколько раз будет повторяться программа: ")  
 var n = *readLine*()!!.*toInt*()  
 if (n > 0) return n  
 else return 1  
 }  
 fun Circus()  
 {  
 try {  
 *print*("Введите радиус окружности: ")  
 var r = *readLine*()!!.*toDouble*()  
 if (r>0)  
 {  
 *print*("Что будем вычислять?\n1 - Длина окружности\n2 - Площадь окружности\n---> ")  
 var number = *readLine*()!!.*toInt*()  
 if (number == 1) *println*("L = ${"%.2f".*format*(2 \* *PI* \* r)}")  
 else if (number == 2) *println*("S = ${"%.2f".*format*(*PI* \* r.*pow*(2))}")  
 else *println*("Нет такого действия")  
 }  
 else *println*("Радиус не может быть отрицательным!")  
 }  
 catch (e : Exception)  
 {  
 *println*("Данные введены неправильно!")  
 }  
 }  
}

**Тестовые ситуации:**

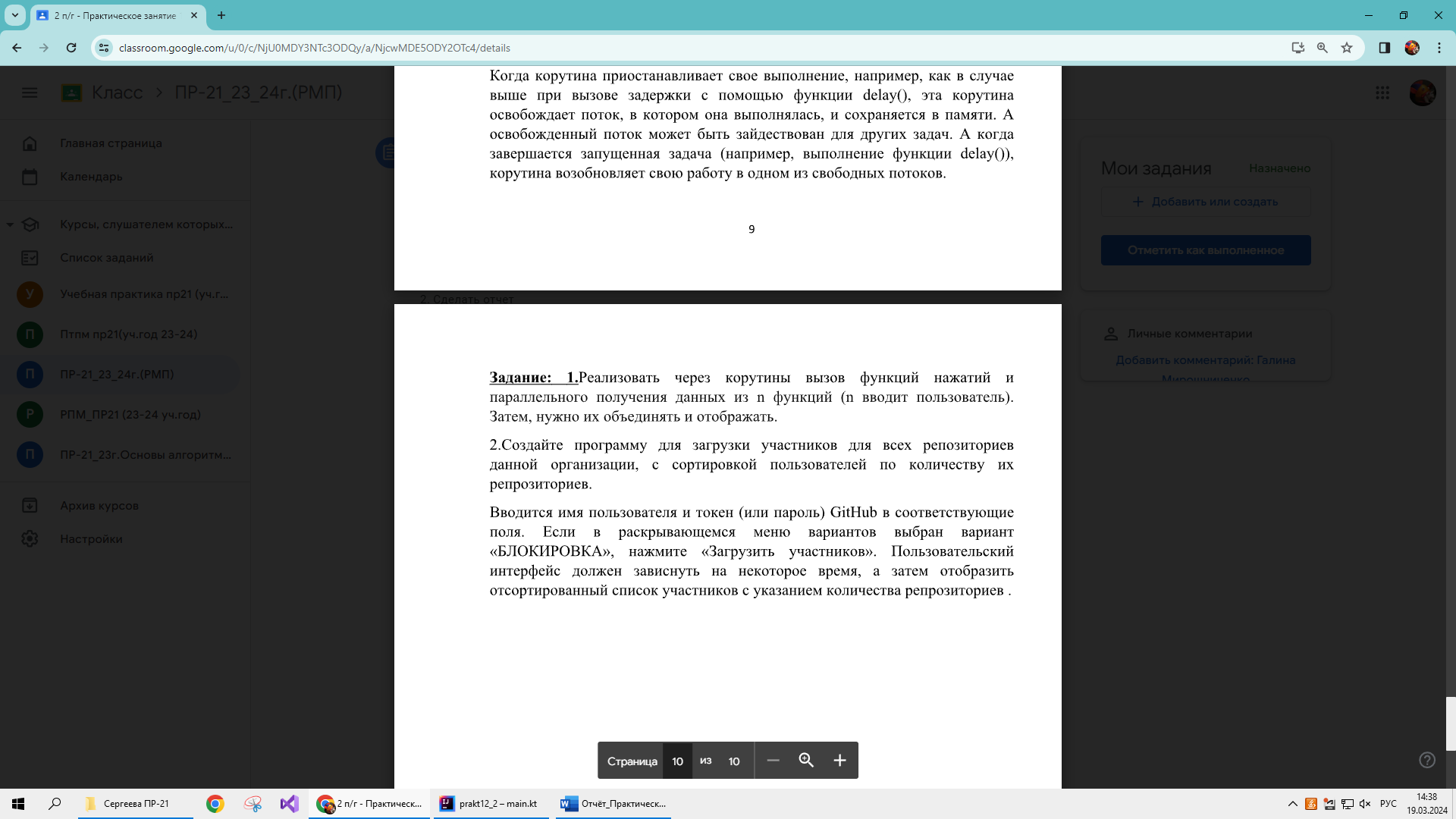


(пользователь не ввёл данные за 30 секунд, поэтому программа завершилась)





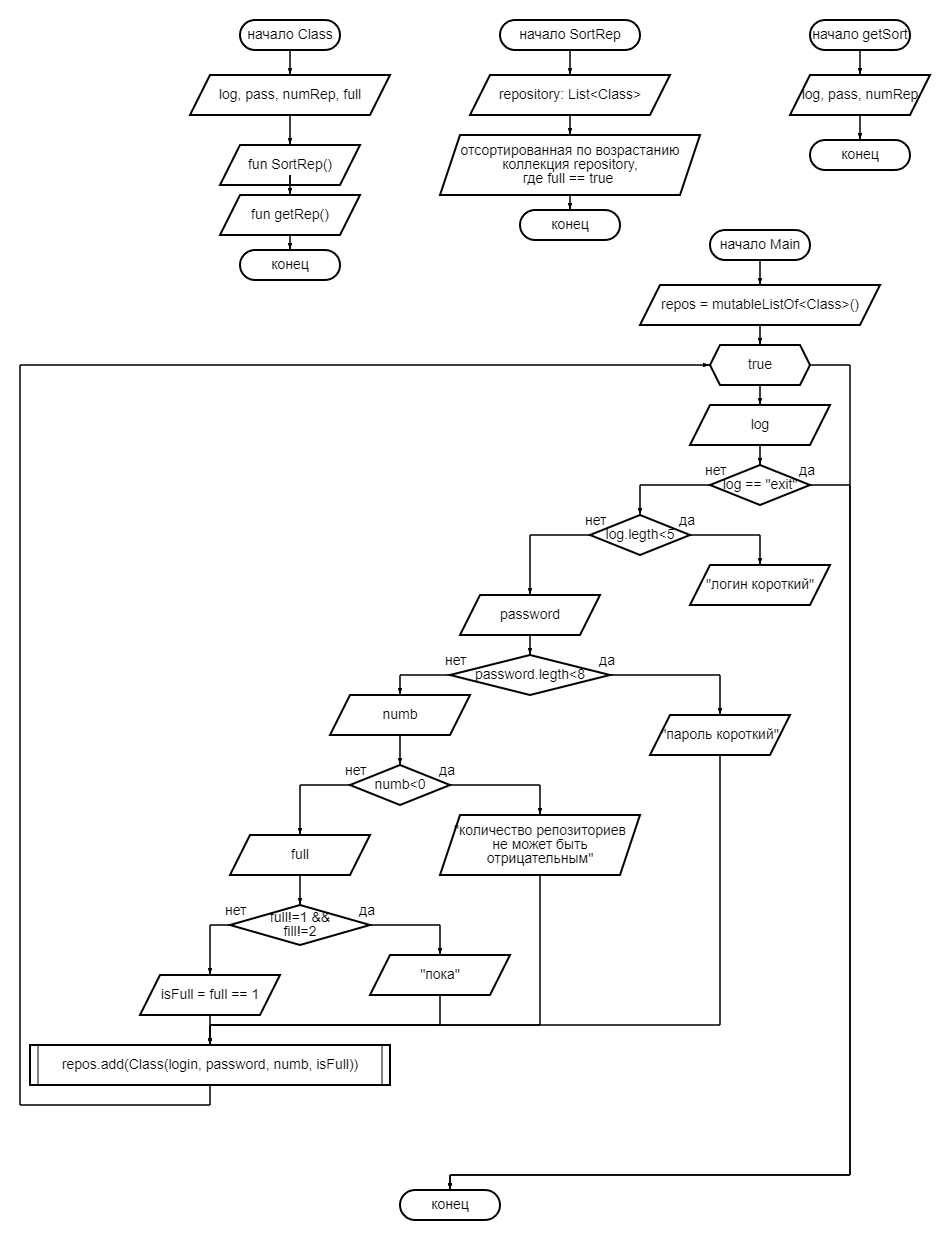
**Задание 2:**



**Входные данные:** login – логин аккаунта, string; password – пароль от аккаунта, string; numb – количество репозиториев, int; full – заполненность репозиториев(1 – репозиторий заполнен, 2 – репозиторий пустой), int

**Выходные данные:** конфиденциальные данные пользователей, у которых репозитории не пустые, отсортированные по возрастанию количества репозиториев

**Блок-схема:**

****

**Листинг программы:**

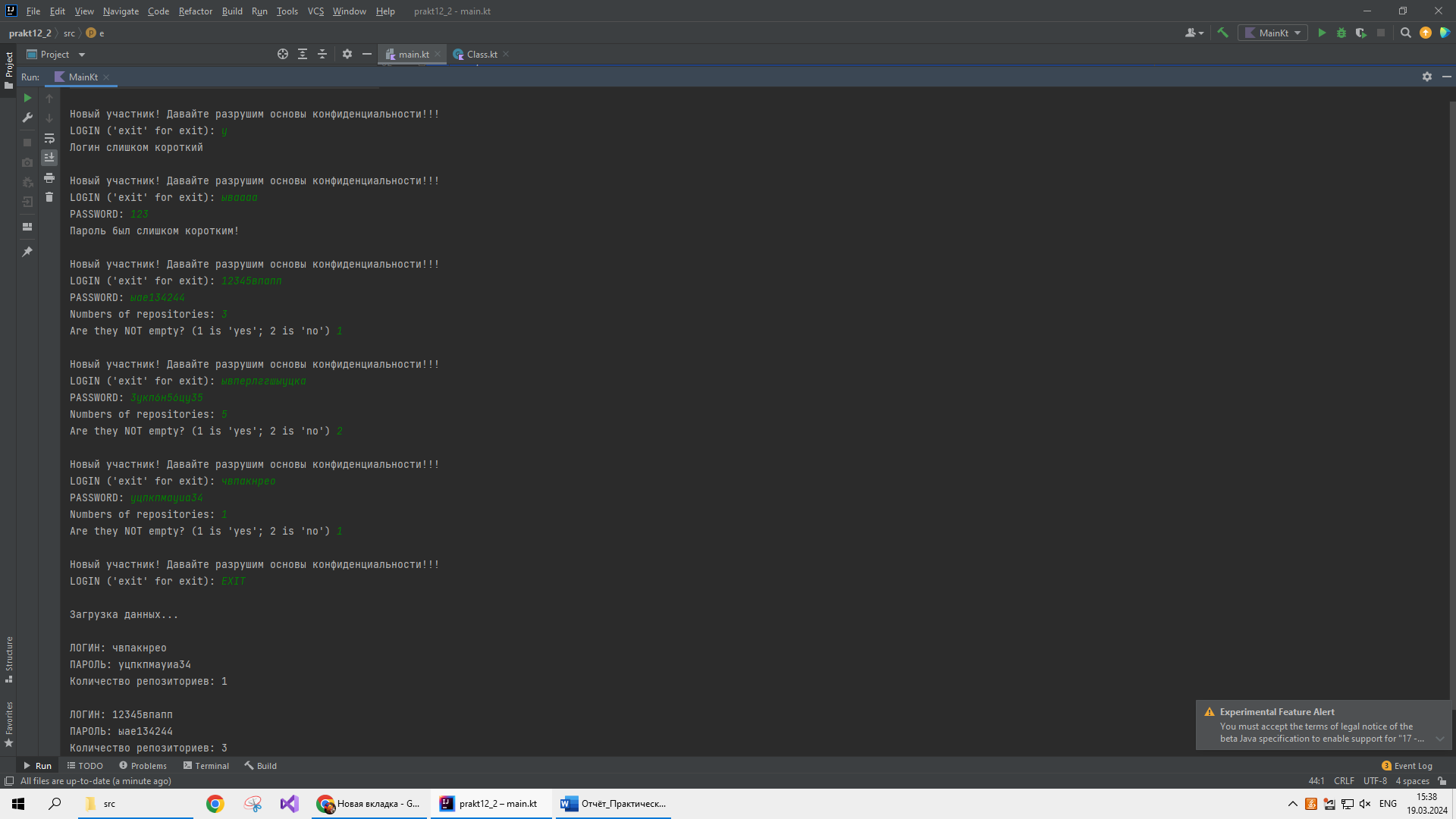
**Main:**

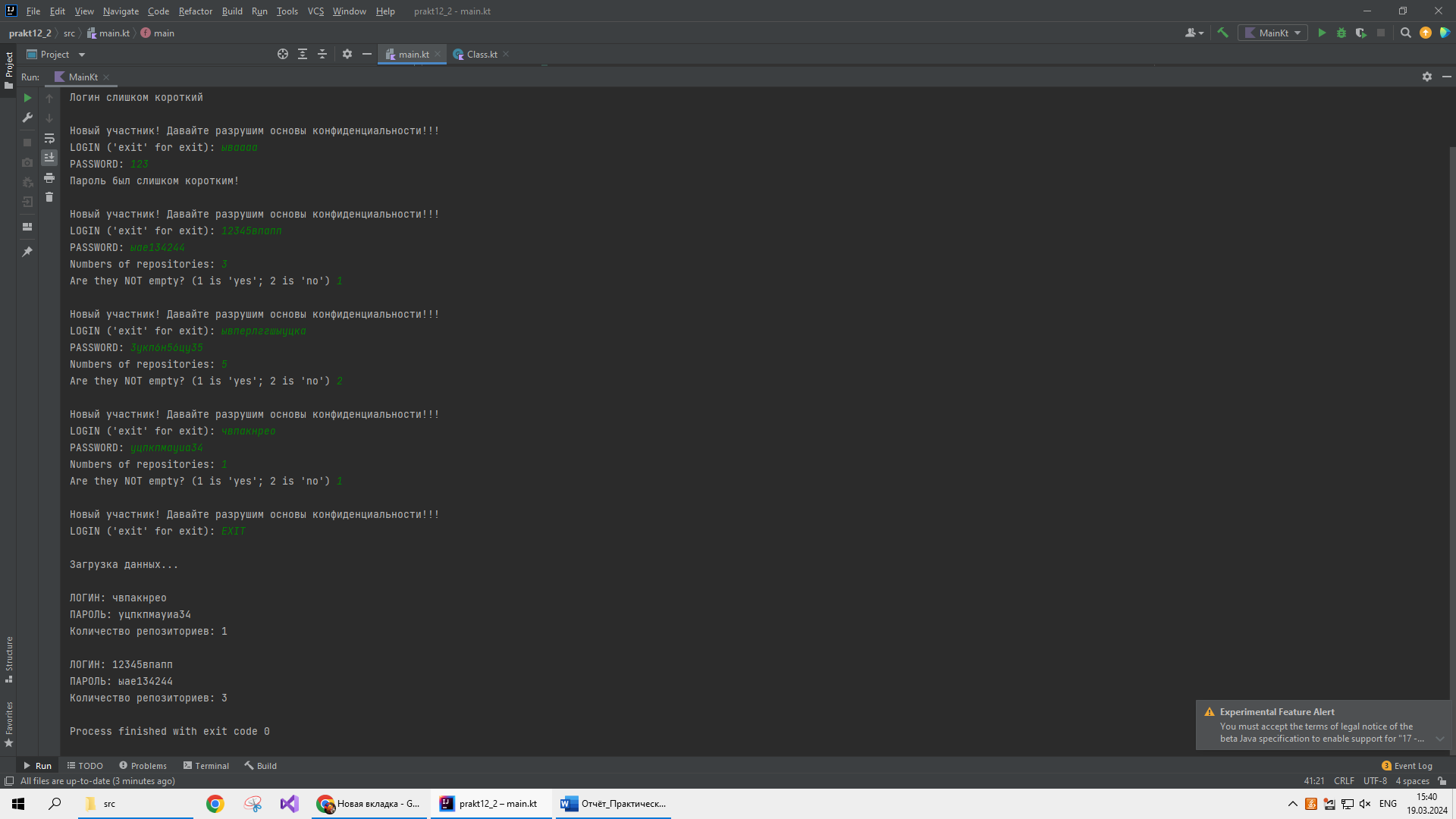
import kotlinx.coroutines.\*  
fun main() = *runBlocking***{** var repos = *mutableListOf*<Class>()  
 while(true)  
 {  
 *println*("\nНовый участник! Давайте разрушим основы конфиденциальности!!!")  
 try{  
 *print*("LOGIN ('exit' for exit): ")  
 var login = *readLine*()!!.toString()  
 if (login == "exit" || login == "Exit" || login == "EXIT") break  
 if (login.length>4)  
 {  
 *print*("PASSWORD: ")  
 var password = *readLine*()!!.toString()  
 if (password.length > 7) {  
 *print*("Numbers of repositories: ")  
 var numb = *readLine*()!!.*toInt*()  
 if (numb>0)  
 {  
 *print*("Are they NOT empty? (1 is 'yes'; 2 is 'no') ")  
 var full = *readLine*()!!.*toInt*()  
 if (full != 1 && full != 2) throw Exception()  
 val isFull: Boolean = full == 1  
  
 repos.add(Class(login, password, numb, isFull))  
 }  
 else *println*("Количество репозиториев не может быть отрицательным!")  
 }  
 else *println*("Пароль был слишком коротким!")  
 }  
 else *println*("Логин слишком короткий")  
 }  
 catch(e : Exception)  
 {  
 *println*("Был произведён неправильный ввод данных")  
 }  
 }  
 if (repos.size >0) {  
 *println*("\nЗагрузка данных...")  
 delay(5000L)  
 val sortRep = repos.*first*().sortRep(repos)  
 sortRep.*forEach* **{** *println*(**it**.getRep())  
 **}** }  
 else *println*("THE END")  
**}**

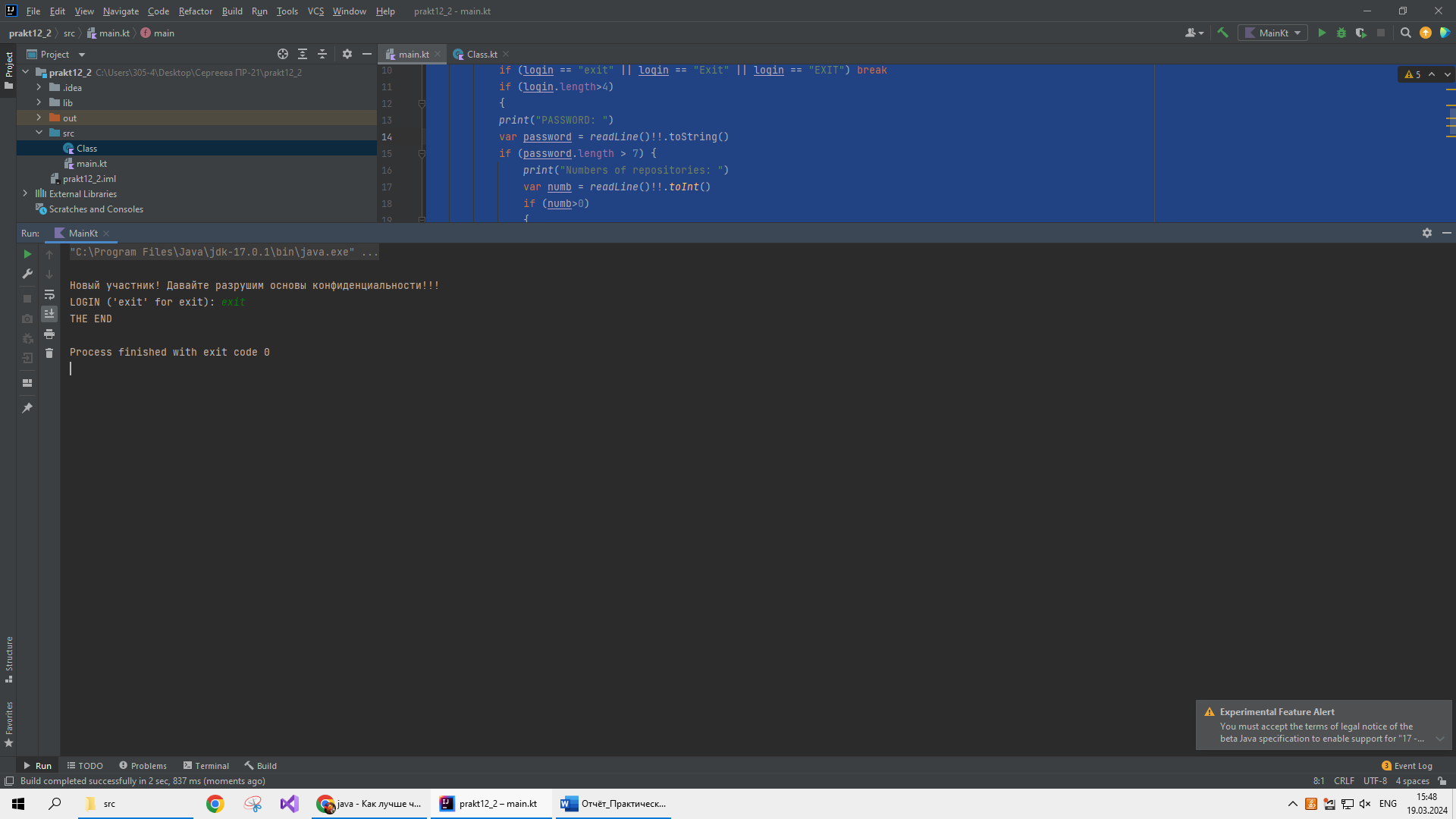
**Class:**

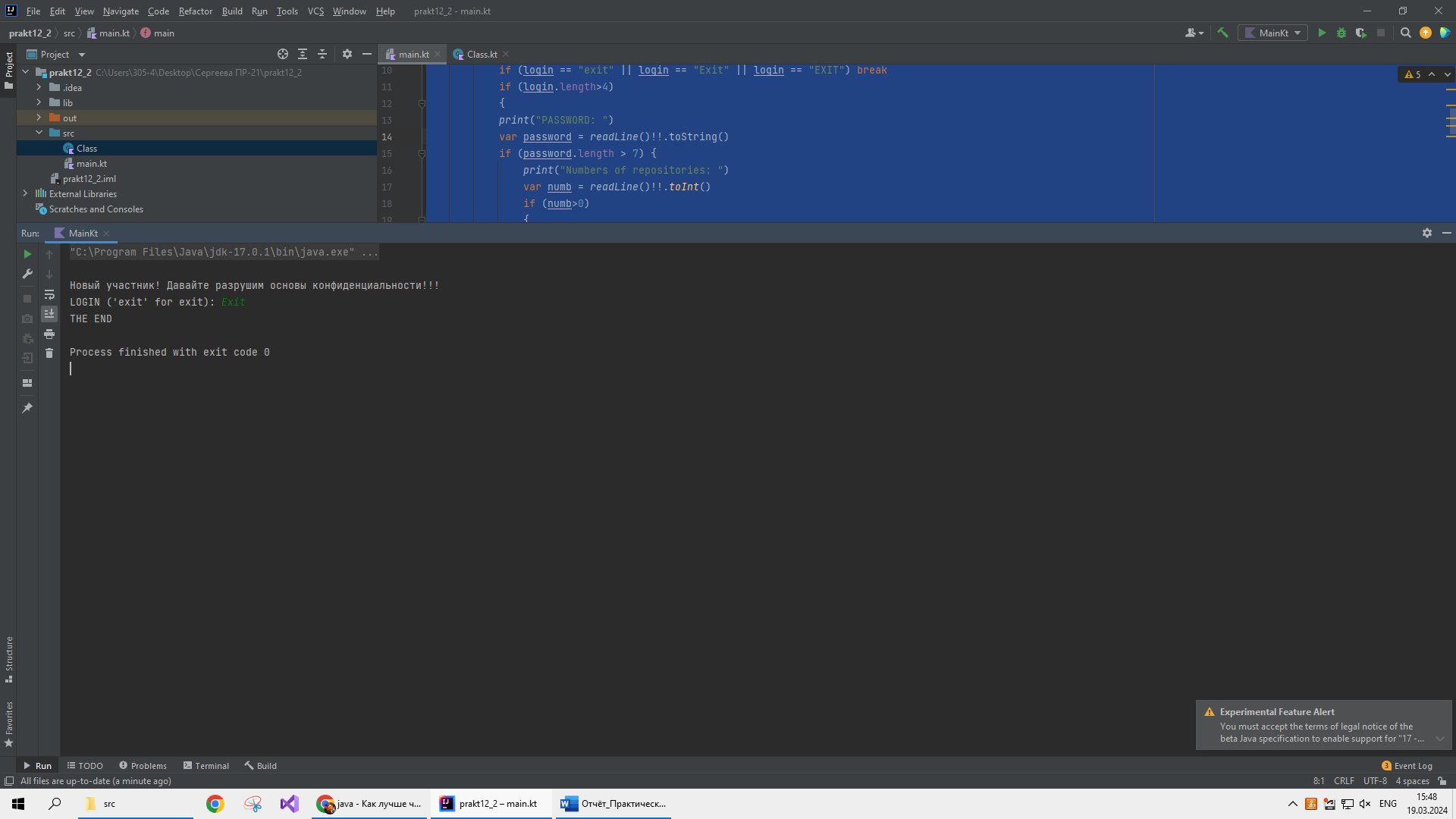
import kotlinx.coroutines.delay  
  
class Class (var log: String, var pass: String,var numRep: Int, var Full: Boolean) {  
 suspend fun sortRep(repository: List<Class>): List<Class>  
 {  
 delay(7000L)  
 return repository.*filter* **{it**.Full**}**.*sortedWith*(*compareBy* **{it**.numRep**}**)  
 }  
 suspend fun getRep(): String  
 {  
 return "\nЛОГИН: ${log}\nПАРОЛЬ: ${pass}\nКоличество репозиториев: ${numRep}"  
 }  
}

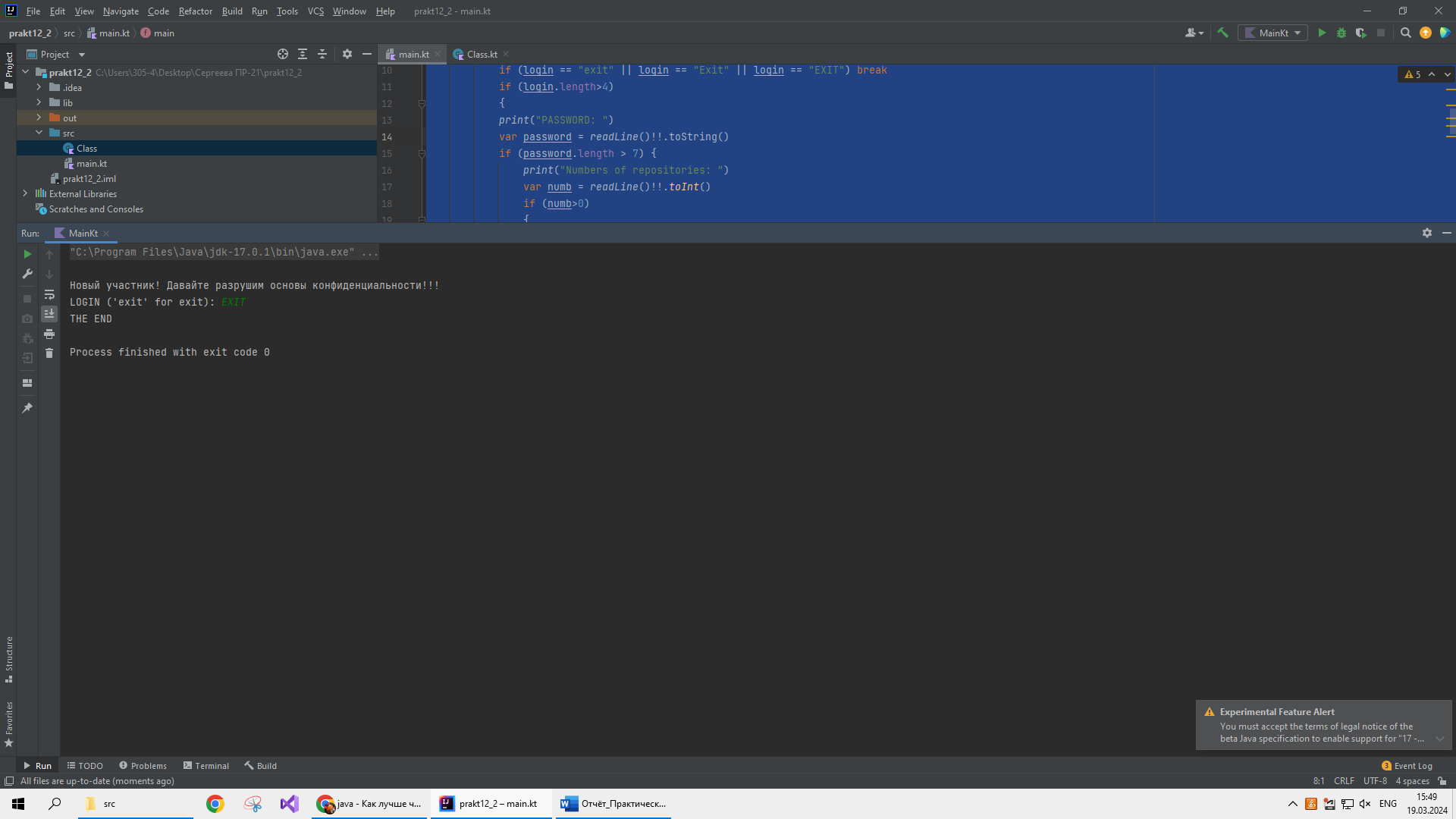
**Тестовые ситуации:**

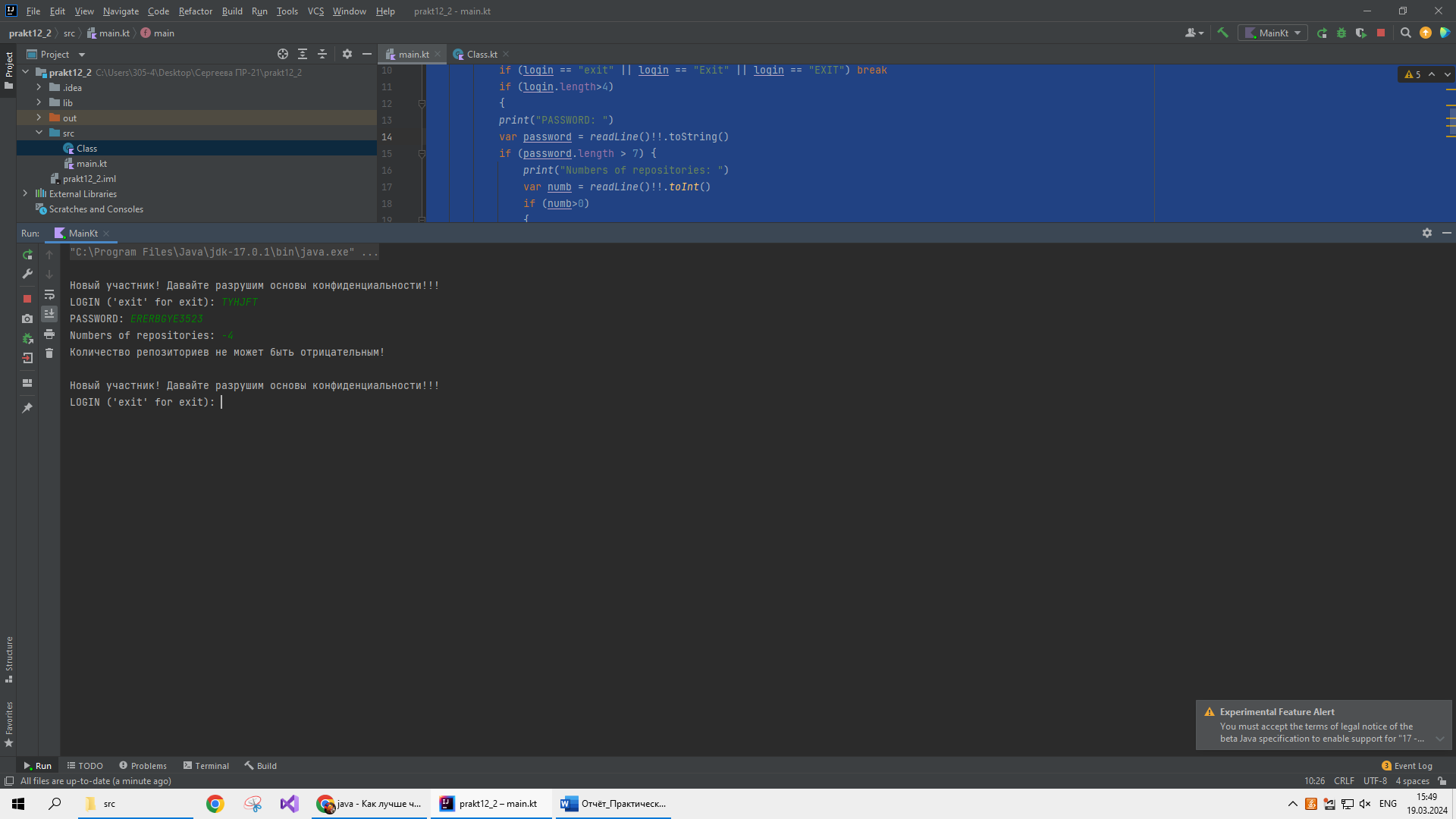












**Вывод:** познакомились с корутинами языка Kotlin