KOTLIN DASTURLASH TILIDA KORUTINLAR BILAN ISHLASHNI TALABALARGA O‘RGATISH

X.SH.Musayev
o‘qituvchi, TATU Farg‘ona filiali, Farg‘ona

Z.Q.Ermatova
o‘qituvchi, TATU Farg‘ona filiali, Farg‘ona

Abstract. The article shows how to teach students how to work with Coroutines in the Kotlin programming language. Information about asynchronous and parallel calculations, coroutine forms in the Kotlin programming language is provided.

Key words and phrases: asynchronous and parallel calculation, coroutine, lib folder, suspend function, launch function.

Аннотация: В статье показано, как научить студентов работать с корутинами на языке программирования Kotlin. Приведена информация об асинхронных и параллельных вычислениях, формах сопрограмм в языке программирования Kotlin.

Ключевые слова и фразы: асинхронный и параллельный расчет, сопрограмма, папка lib, функция приостановки, функция запуска.

Annotasiya. Maqolada Kotlin dasturlash tilida Korutinlar bilan ishlashni talabalarga o‘rgatish usullari ko‘rsatilgan. Kotlin dasturlash tilida asinxron va parallel hisoblashlar, korutin shakllari haqida ma’lumotlar keltirilgan.

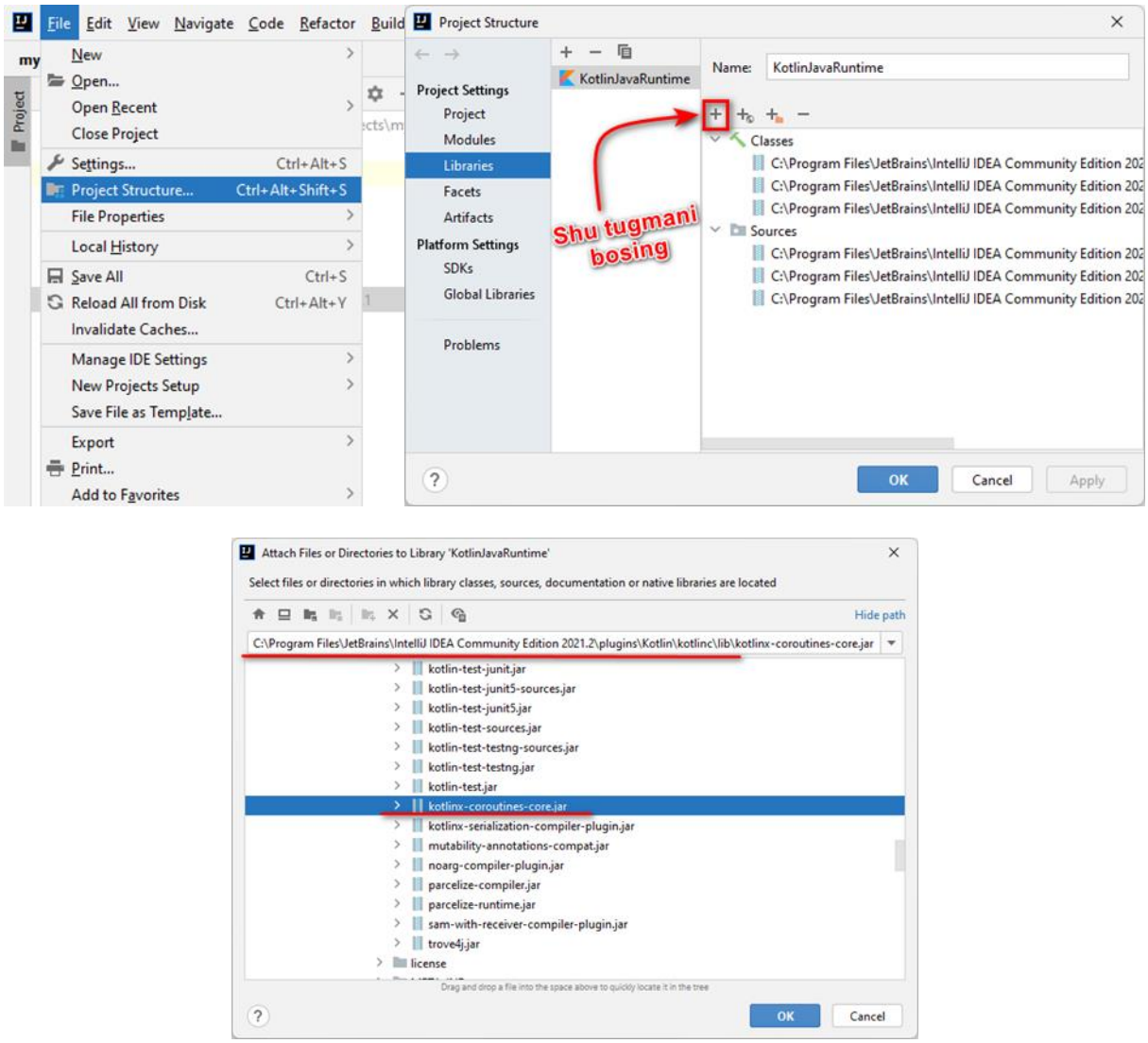
Tayanch so‘z va iboralar: asinxron va paralel hisoblash, korutin, lib nomli papka, **suspend** funksiyasi, **launch** funksiyasi.

Hozirgi zamonda asinxron va paralel hisoblash qo‘plab dasturlash tillarining ajralmas xususiyatiga aylanib bormoqda. Kotlin dasturlash tili ham bundan mustasno emas. Nima uchun asinxron va parallel hisoblash kerak? Parallel hisoblash bir vaqtning o‘zida bir nechta vazifalarni bajarishga imkon beradi, asinxron hisoblash esa uzoq vaqt talab qilinadigan vazifa bajarilayotganda asosiy dastur oqimini bloklamaslik uchun xizmat qiladi. Masalan, oddiy dastur yoki mobil ilova uchun grafik ma’lumotlarni chizishda. Bundan tashqari qandaydir tugmani bosib internet resursiga so‘rov yuborish va undan kelgan ma’lumotlarni tahlil qilishda. Bu ikki ko‘rsatilgan misollar uzoq vaqt talab qilishi mumkin. So‘rov yuborilganida ilova osilib qolmasligi uchun internet resurslariga so‘rovlar asinxron tarzda yuboriladi. Asinxron so‘rovlar yordamida ilova internet resursidan javob kelguniga qadar kutib turmaydi, dastur o‘z ishini davom etadi va javob kelganida unga tegishli bo‘lgan buyruqlar ketma–ketligi bajariladi.

Kotlin dasturlash tilida asinxron va parallel hisoblashlar korutin shaklida amalga oshiriladi. Korutin – bu kodning qolgan qismi bilan parallel ravishda ishlashi mumkin bo‘lgan kodlar blokidir. Korutinlar bilan bog‘liq bo‘lgan asosiy funksiyalar kotlinx.coroutines paketida joylashgan.

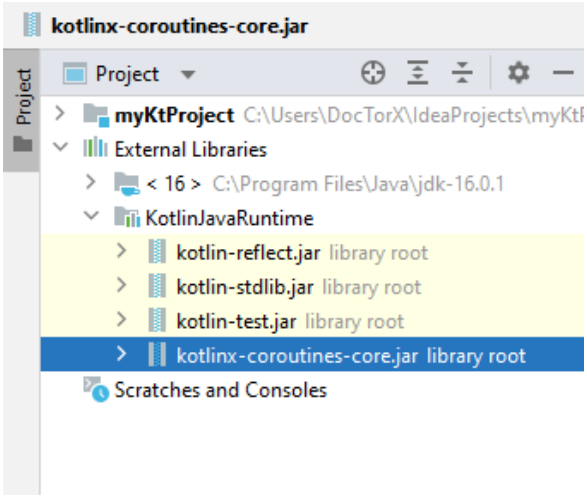
Korutin paketini dasturga qo‘shish

Korutin bilan ishlovchi paket yaratilayotgan dasturga to‘g‘ridan–to‘g‘ri bog‘lanmagan bo‘lib, bu paketni dasturchining o‘zi yaratayotgan dasturiga qo‘shib oladi. Agar dasturchi IntelliJ IDEA muhitidan foydalanayotgan bo‘lsa, unda yaratilgan loyiha uchun bu paketni qo‘shish quyidagicha amalga oshiriladi. File menyusidagi Project Structure bandi tanlanadi. Bu band tanlanganida Project Structure nomli oyna hosil bo‘ladi. Hosil bo‘lgan oynadagi Project Settings bo‘limidagi Libraries bandi tanlanadi. Libraries bandi tanlanganda oynaning o‘ng qismidagi + tugmasi bosilib, kerakli fayl qo‘shib olinadi. Qo‘shib olinadigan faylning nomi kotlinx-coroutines-core.jar bo‘lib, bu fayl Kotlin dasturlash tilining kompilyatori joylashgan papkaning ichidagi lib nomli papka joylangan bo‘ladi. (9.1.1–rasm).



9.1.1–rasm: kotlinx-coroutines-core.jar faylini ilovaga qo‘shish

Rasmda keltirilgan ketma–ketlik bajarilganida tashqi kutubxonalarni ro‘yxatga oluvchi External Libraries/KotlinJavaRuntime dagi mavjud kutubxonalar bilan ro‘yxatga olinadi (9.1.2–rasm).



9.1.2–rasm: External Libraries/KotlinJavaRuntime bo‘limi

Boshqa turdagi muhitlarda kutubxonalarni bog‘lash bu ko‘rinishda bo‘lmasligi mumkin.

suspend ko‘rinishdagi funksiyani aniqlash

Korutinlar bilan ishlashda buyruqlar ketma–ketligini kutib turish vazifasini berish uchun suspend ko‘rinishidagi funksiya yaratish kerak bo‘ladi. Quyida kkeltirilgan dasturda funksiya suspend ko‘rinishida e’lon qilingan bo‘lsa ham korutindan foydalanilmagan hol ko‘rsatilgan.

```
import kotlinx.coroutines.*
suspend fun main(){
  for(i in 0..5){
          delay(400L)
          print("$i, ")
  }
  println("\nHello Coroutines")
}
```

Bu yerda asosiy funksiya 0 dan 5 gacha bo‘lgan sonlar ketma–ketligini konsol oynasiga chiqarish keltirib o‘tilgan. Sonlarni konsol oynasiga chiqarish vaqtidagi simulyasiya uchun kotlinx.coroutines paketidagi maxsus delay() funksiyasidan foydalanilgan. Bu funksiya kechiktirish ma’nosini anglatib, kechiktirish millisekundlarda ifodalaniladi. Funksiyaga ko‘rsatilayotgan qiymat Long turiga mansub bo‘lishi kerak. Yuqoridagi dasturda funksiya ketma–ketlik elementlarini chiqarish uchun har bir murojaatda 400 millisekund kechiktirishni amalga oshiradi. Takrorlanish operatori ishini yakunlaganidan so‘ng ekranga “Hello Coroutines” qatorini chiqariladi.

delay() funksiyasini asosiy funksiyada ishlatish uchun asosiy funksiya suspend kalit so‘zi bilan e’lon qilingan bo‘lishi kerak. Buyruqlar ketma–ketligi bajarilishini to‘xtatib turadigan va ma’lum bir vaqtdan keyin davom etadigan funksiyalar albatta suspend kalit so‘zi yordamida aniqlangan bo‘lishi kerak. delay() funksiyasini ham suspend kalit so‘zi yordamida aniqlangan bo‘lib, suspend kalit

so‘zi bilan aniqlangan har qanday funksiya shu kalit so‘z bilan aniqlangan funksiya bilan chaqirilishi lozim. Agar dasturni ishga tushirilsa, konsolda quyidagi ko‘rinishda natija hosil bo‘ladi.

0, 1, 2, 3, 4, 5,
Hello Coroutines

Yuqoridagi dasturda “Hello Coroutines” qatori takrorlanish tugashini kutadi. Ammo bunday ko‘rinish internet resurs bilan ishlash vaqtida xatolik keltirib chiqarishi mumkin yoki ma’lumotlar uzatilishidagi kamchilik ko‘rinib qoladi.

Ma’lumotlar uzatilishidagi kamchiliklarni bartaraf etish uchun suspend ko‘rinishdagi funksiyaga odatiy buyruqlar ketma–ketligi yozmasdan quyidagi ko‘rinishda buyruqlar yozilishi kerak bo‘ladi.

```
import kotlinx.coroutines.*
suspend fun main() = coroutineScope{
  launch{
        for(i in 0..5){
              delay(400L)
              print("$i, ")}}
  println("Hello Coroutines")
}
```

Birinchi navbatda korutinni aniqlash va bajarish uchun korutinning tana qismini aniqlab olish kerak, chunki korutinni faqat korutin doirasi ishlatish yoki chaqirish mumkin. Korutin doirasi coroutineScope() funksiyasining tana qismi hisoblanadi va bu funksiyaning tanasida korutin joylashadi. Bundan tashqari bu funksiyaning tanasida aniqlangan barcha korutinlarni bajarilishini kutadi. Shuni ta’kidlab o‘tish kerakki, coroutineScope() funksiyasi faqat asosiy funksiya (main) ga ishlatiladi.

Korutin quruvchisi bo‘lgan launch funksiyasi yordamida quriladi va ishga tushiriladi. Bu funksiya kodlar blokidan foydalanilgan holda korutin yaratadi. Yuqorida ko‘rsatilgan dasturda korutin yaratuvchi launch funksiyasi tomonidan quyidagi korutin yaratilgan:

```
{
  for(i in 0..5){
        delay(400L)
        print("$i, ")}}
```

Bu korutin dasturda berilgan boshqa kodlar bilan parallel ravishda bajariladi. Ya’ni bu korutin dasturning asosiy funksiyasida aniqlangan boshqa kodlardan mustaqil ravishda ishlash xususiyatiga ega. Natijada dasturni bajarish uchun buyruq berilganda konsolga quyidagi ma’lumotlar chiqariladi.

Hello Coroutines
0, 1, 2, 3, 4, 5,

Bu dasturda “Hello Coroutines” qatorini chiqarish uchun dastur takrorlanuvchi operator ishini tugatishiga qarab turmaydi. Balki u bilan parallel ravishda bajariladi.

Yuqorida ko‘rsatib o‘tilgan dasturlarda korutinlar asosiy funksiyaning tana qismida aniqlangan. Korutinni alohida funksiyada ham aniqlash mumkin.

```
import kotlinx.coroutines.*
suspend fun main()= coroutineScope{
  launch{ doWork() }
  println("Hello Coroutines")
}
suspend fun doWork(){
  for(i in 0..5){
          print("$i, ")
          delay(400L)
  }
}
```

Bu dasturda korutin buyruqlar ketma–ketligi doWork() funksiyasida aniqlangan. Bu funksiyada delay() funksiyasidan foydalanganligi sababli funksiya suspend kalit so‘zi yordamida aniqlangan. Dasturning asosiy funksiyasi main() ham suspend kalit so‘zi bilan aniqlangan. Bu asosiy funksiyaning tana qismida launch funksiyasi yozilgan.

Korutin maydoni. Korutin faqat ma’lum bir korutin maydoni (coroutine scope)da bajarilishi mumkin. Korutin maydoni korutinlar ishlaydigan bo‘shliqni ifodalaydi. Korutin maydoni ma’lum bir hayotiy sikli ega. Bu maydonning ichida aniqlangan korutinlar hayot siklini o‘zi boshqarib boradi.

Kotlin dasturlash tilida korutin maydonini yaratish uchun CoroutineScope interfeysi ob’yektini yaratuvchi bir nechta funksiyalardan foydalanish mumkin. Bu funksiyalardan biri – coroutineScope() nomli funksiya hisoblanadi. Bu funksiyani har qanday funksiya uchun qo‘llash mumkin, masalan:

```
import kotlinx.coroutines.*
suspend fun main(){
  doWork()
  println("Hello Coroutines")
}
suspend fun doWork()= coroutineScope{
  launch{
          for(i in 0..5){
                  print("$i, ")
                  delay(400L)}}}
```

Bir nechta korutinlarni ishga tushirish

Bir funksiyaning o‘zida bir vaqtda bir nechta korutinlardan foydalanish va ularni ishga tushirish mumkin. Bunda barcha korutinlar bir vaqtning o‘zida ishga tushib natija beradi. Masalan,

```
import kotlinx.coroutines.*
suspend fun main()= coroutineScope{
  launch{
```

```
        for(i in 0..5){
            delay(400L)
            println("$i – birinchi")
        }
    }
    launch{
        for(i in 6..10){
            delay(400L)
            println("$i - ikkinchi")
        }
    }
    println("Hello Coroutines")
}
```

coroutineScope() funksiyasi korutin maydonini yaratadi va bu maydonda aniqlangan barcha korutinlar bajarilishini nazarda tutadi. Yuqoridagi dasturda ikki korutin berilgan. Bu korutinlar faoliyatini tugatgandan so‘ng asosiy dastur o‘z ishini yakunlaydi. Bu dastur konsolga quyidagicha natija beradi.

```
Hello Coroutines
6 - ikkinchi
0 – birinchi
7 - ikkinchi
1 – birinchi
8 - ikkinchi
2 – birinchi
9 - ikkinchi
3 – birinchi
10 - ikkinchi
4 – birinchi
5 – birinchi
```

Ichki korutinlar

Bir korutin boshqa bir yoki bir nechta korutinlarni o‘z ichiga olishi mumkin:

```
import kotlinx.coroutines.*
suspend fun main() = coroutineScope{
    launch{
        println("Tashqi korutin")
        launch{
            println("Ichki korutin")
            delay(400L)
        }
    }
    println("Asosiy dastur oxiri")}
```

Ichki korutinlar tashqi korutinlar doirasida aniqlanadi. Bunda tashqi korutin bilan ichki korutin o‘z vazifasini bir vaqtda boshlaydi.

Foydalanilgan adabiyotlar ro‘yxati

1. R. Zulunov. Preparing the educational process for the era of artificial intelligence. The journal of integrated education and research, Volume 1, issue 4, September 2022, p.261-263
2. R. Zulunov. Use of artificial intelligence technologies in the educational process. Web of Scientist: International Scientific Research Journal (WoS), Volume 3, Issue 10, Oct., 2022, p. 764-770.
3. R. Zulunov. Подготовка образовательного процесса к эпохе искусственного интеллекта. Periodica Journal of Modern Philosophy, Social Sciences and Humanities, 2022, Oct., 11, p. 81-83.
4. Солиев Б. Н. Проблемы моделирования электронных торговых процессов на основе местных характеристик //Исследования молодых ученых. – 2020. – С. 8-11.
5. Halimovich T. T. et al. Monte Carlo method for constructing an unbelised assessment of diffusion problems //European science review. – 2020. – №. 1-2. – С. 7-12.
6. Tozhiev, Tokhirjon Halimovich, et al. "MONTE CARLO METHOD FOR CONSTRUCTING AN UNBELISED ASSESSMENT OF DIFFUSION PROBLEMS." European Science Review 1-2 (2020): 7-12.
7. Muminjonovich, Hoshimov Bahodirjon, and Uzokov Barhayot Muhammadiyevich. "Teaching Children to Programming on the Example of the Scratch Program." Eurasian Scientific Herald 9 (2022): 131-134.