Anton Dudakov, Vladimir Merkurev, 2023

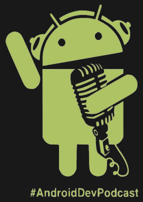
Coroutine puzzlers 🧩

Anton Dudakov

Software Engineer at **Ziina**
- mostly mobile (but sometimes backend) 💪
- mostly Android (but a lot of iOS also) 🍏

Previously:
- Sberbank (Devices)
- Yandex (Auto)

Also:
- Android Dev Podcast co-host
- MENA Mobile Meetup organiser



Владимир Меркурьев

Андроид разработчике в Ziina
Иногда backend разработчик в Ziina

До Ziina работал в

- Yandex(Auto)

Что нас ждёт?

https://www.crowd.live/ETYGW

Scope 🍕

Scope 1

Show me the code 👉

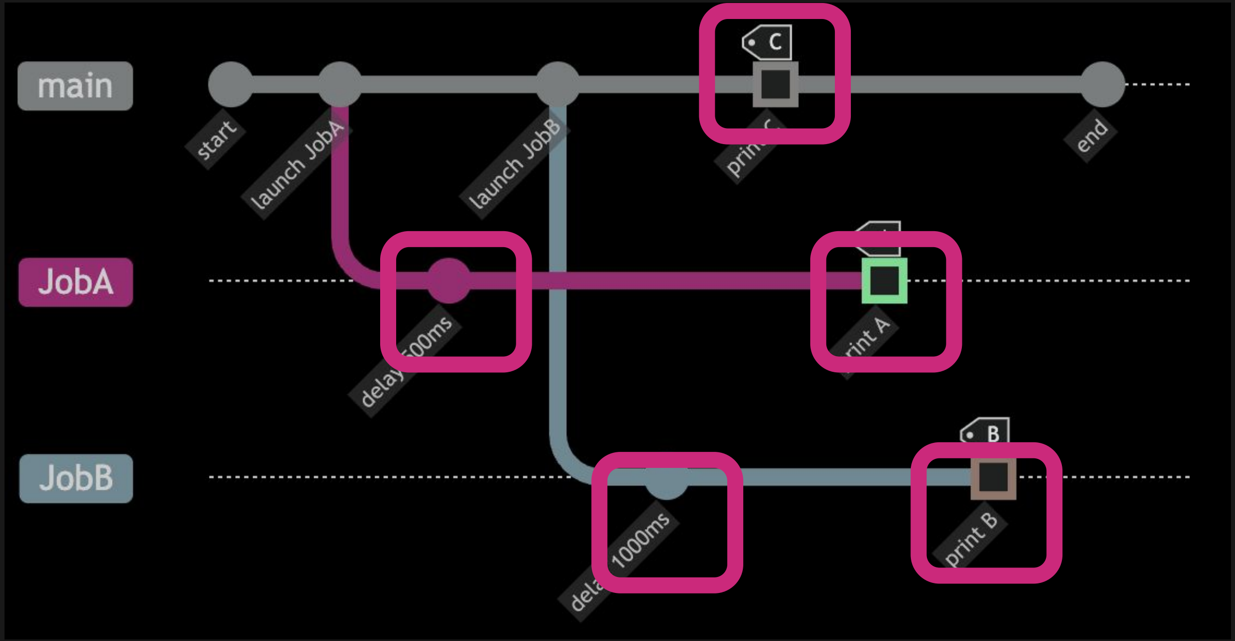
Scope 1

CAB

```
launch {
    delay(500)
    print("A")
}

launch {
    delay(1000)
    print("B")
}

print("C")
```



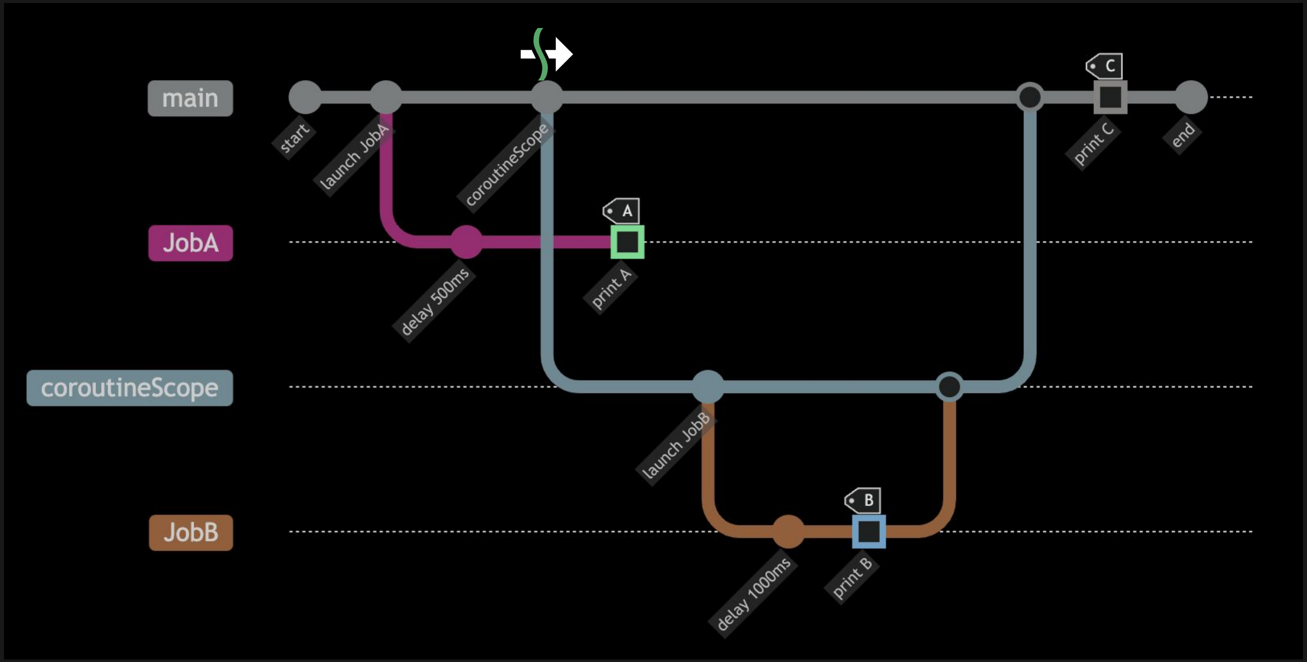
Scope 2

Show me the code 👉

Scope 2

ABC

```
//same as runBlocking,
//but suspend
coroutineScope {
    val JobB = launch {
        delay(1000)
        print("B")
    }
}
```



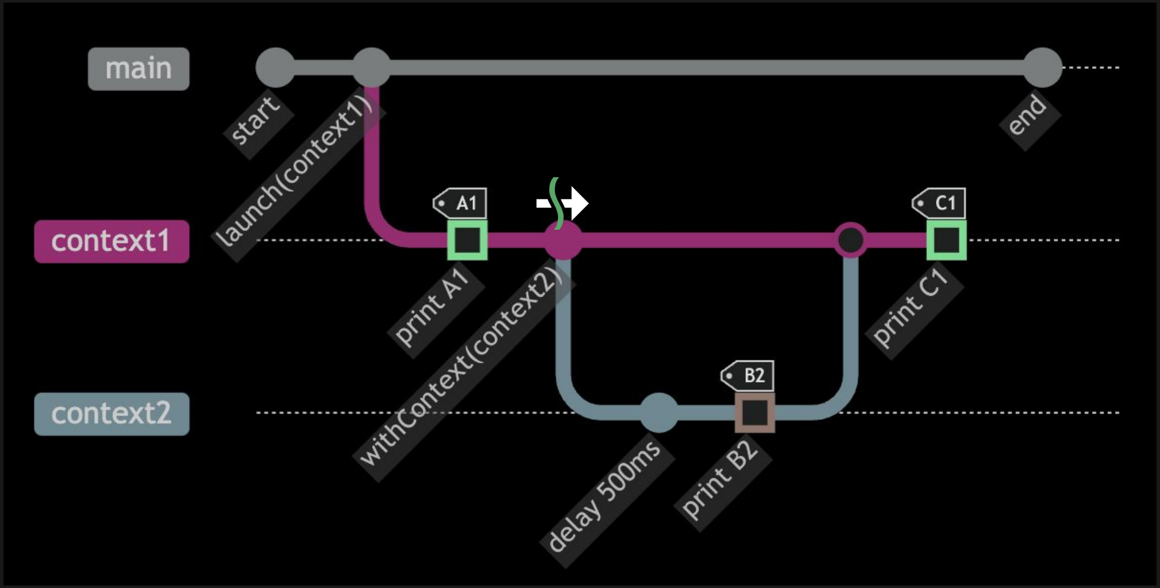
Scope 3

Show me the code 👉

Scope 3

A1 B2 C1

```
//suspend function
withContext(context2) {
    print("B$threadName ")
}
```



Cancellation □

Cancellation 1

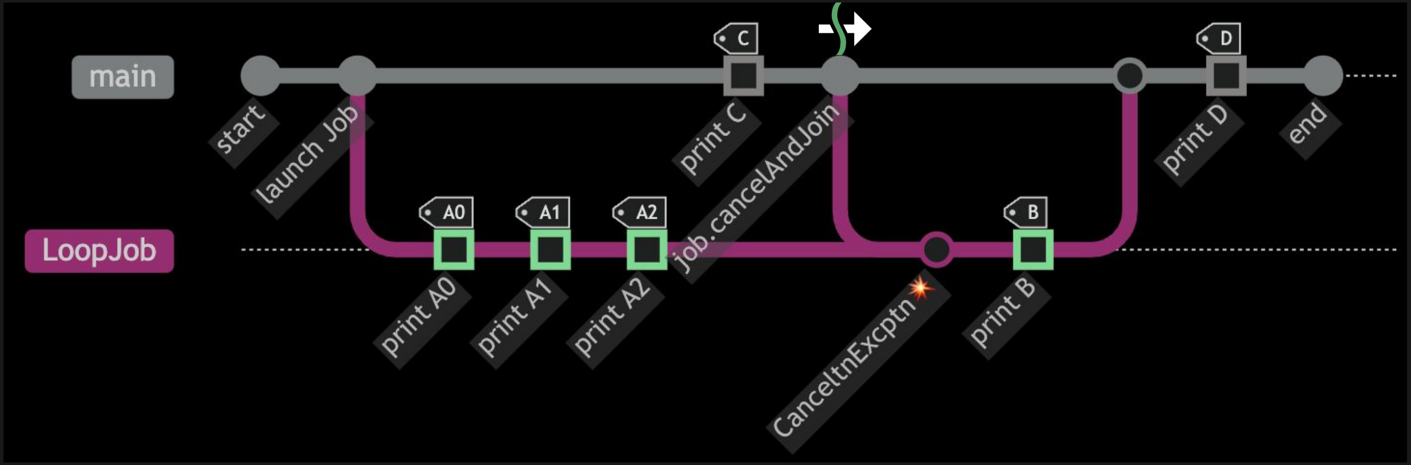
Show me the code 👉

Cancellation 1

A0A1A2CBD

```
val job = launch {
  try {
    repeat(5) { i ->
      print("A$i")
      delay(100) // 💥 CancellationException
    }
  } finally {
    print("B")
  }
}

delay(250)
print("C")
job.cancelAndJoin()
print("D")
```



Cancellation 2

Show me the code 👉

Cancellation 2

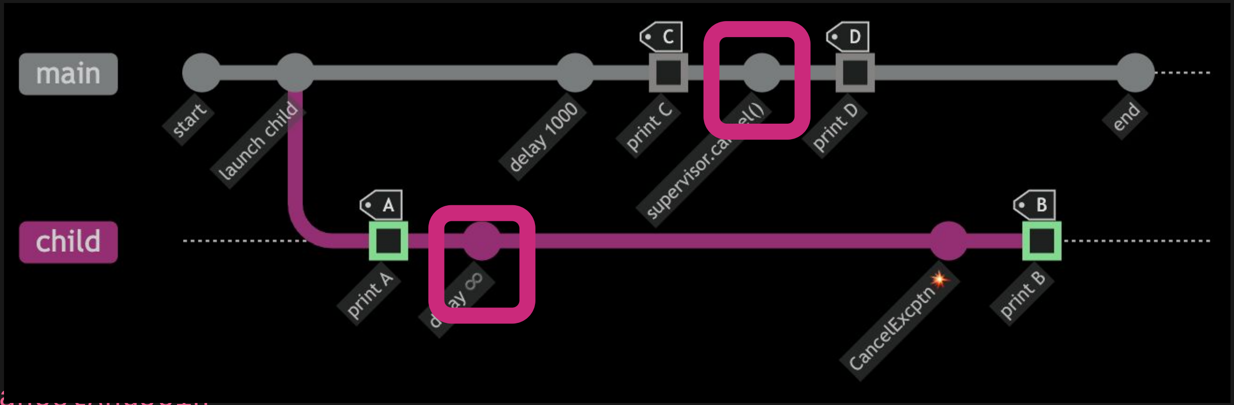
ACDB

```
with(/*..*/) { //not withContext
  val child = launch {
    try {
      print("A")
      delay(Long.MAX_VALUE)
    } finally {
      print("B")
    }
  }



  delay(1000)
  print("C")
 supervisor.cancel() //not ca
}

print("D")
```



Exceptions 🍬

Exceptions 1

Show me the code 👉

Exceptions 1

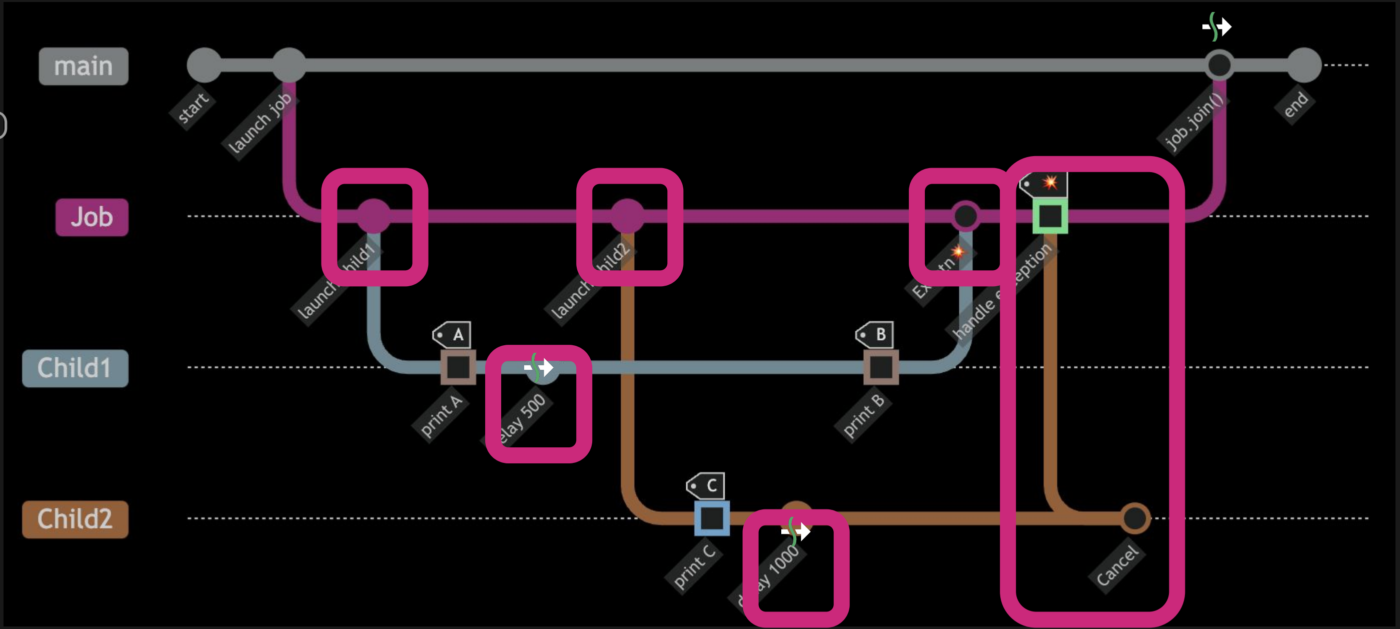
ACB💥

```
val exceptionHandler = { print("💥") }

val job = GlobalScope.launch(exceptionHandler) {
  launch {
    print("A")
    delay(500)
    print("B")
    throw Exception()
  }

  launch {
    print("C")
    delay(1000)
    print("🍬")
  }
}

job.join()
```



Exceptions 2

Show me the code 👉

Exceptions 2

ACB💥

```
val job = GlobalScope.launch(exceptionHandler) { /*..*/ }

// 🤔 difference? No!

val scope = CoroutineScope(Job())

val job1 = scope.launch(exceptionHandler) { /*..*/ }
```

Exceptions - How not to fail?

SupervisorJob()

```
/**
* Creates a supervisor job object in an active state.
* Children of a supervisor job can fail independently of each other.
* …
**/
fun SupervisorJob(parent: Job? = null): CompletableJob
```

Exceptions 3

Show me the code 👉

Exception 3

ACB💥

```
val scope = CoroutineScope(SupervisorJob())
val job = scope.launch(exceptionHandler) {
  launch {
    /*..*/
  }
  launch {
    /*..*/
  }
}
```

Exceptions 4

Show me the code 👉

Exception 4

ACB💥

```
val scope = CoroutineScope(Job())
val job = scope.launch(exceptionHandler + SupervisorJob()) {
  launch {
    /*..*/
  }
  launch {
    /*..*/
  }
}
```

Exceptions 5

Show me the code 👉

Exceptions 5

ACB💥🍬

```
val scope = CoroutineScope(Job())
val job = scope.launch(exceptionHandler) {
    supervisorScope { //this: CoroutineScope
        val child1 = launch {/*..*/}
        val child2 = launch {/*..*/}
        joinAll(child1, child2)
    }
}
```

Exceptions 6

Show me the code 👉

Exceptions 6

ACB💥🍬

```
val scope = CoroutineScope(SupervisorJob() + exceptionHandler)

val job1 = scope.launch {/*..*/}

val job2 = scope.launch {/*..*/}
```

Exceptions 7

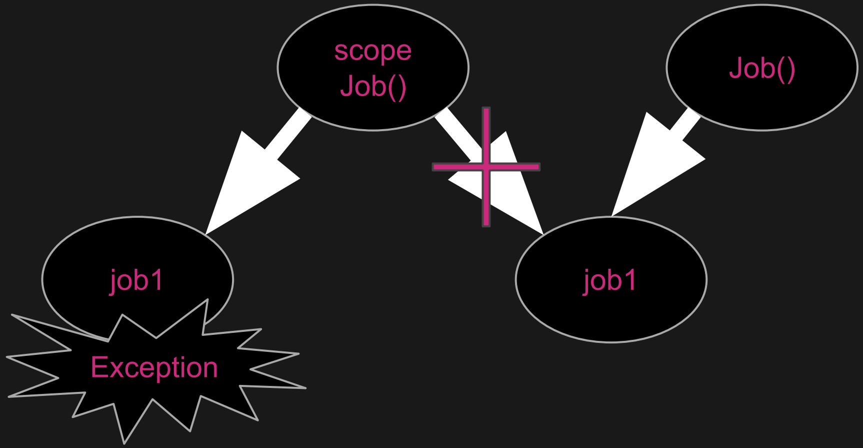
Show me the code 👉

Exception 7

ACB ...java.lang.Exception 🍬

```
val scope = CoroutineScope(Job())

val job1 = scope.launch { //this child of scope's Job
  /*..*/
}
val job2 = scope.launch(Job()) { //this is NOT child of scope's Job
  /*..*/
}
```



Timeout 🍻

Timeout 1

Show me the code 👉

Timeout 1

012 Crash

```
/**
* Runs a given suspending block of code inside a coroutine
* with a specified timeout and
* throws a TimeoutCancellationException
* if the timeout was exceeded.
**/
public suspend fun <T> withTimeout(/**/): T
```

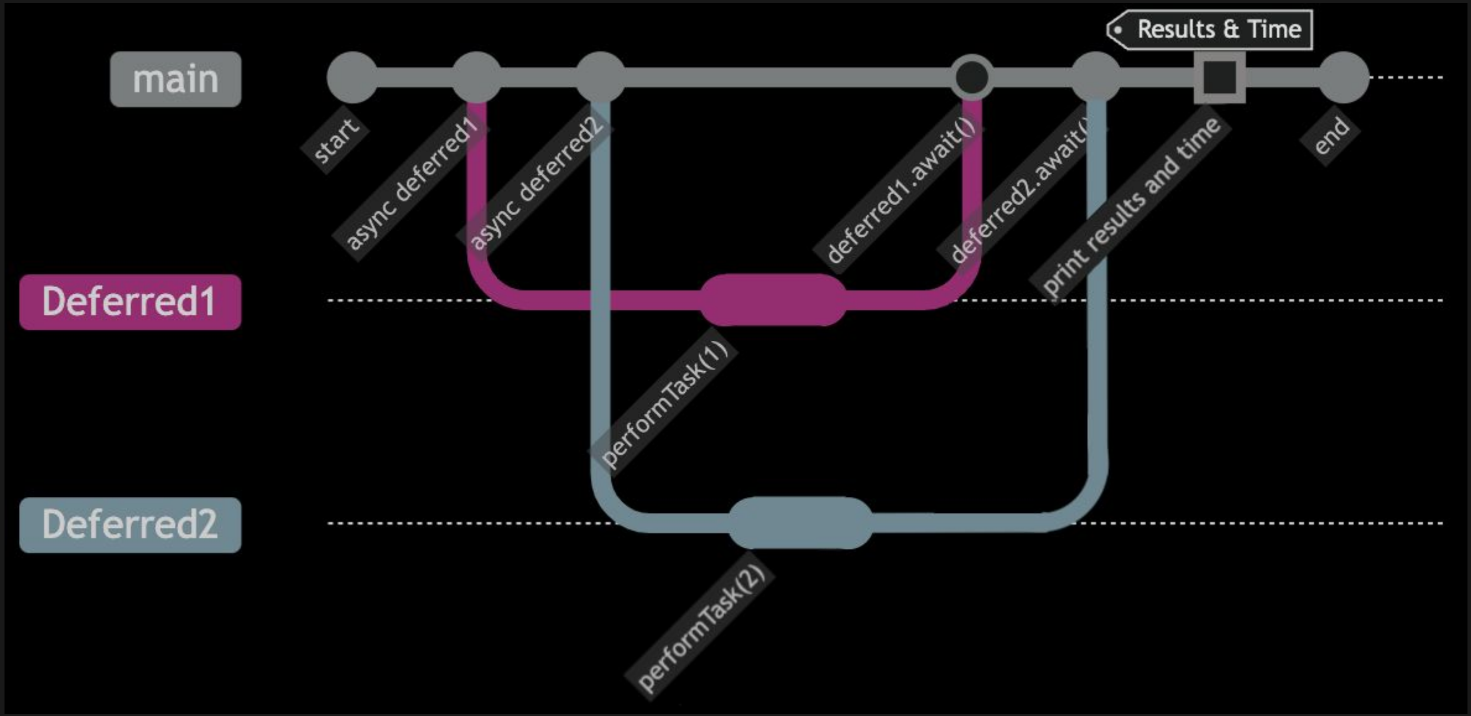
Async/Await 🥠

Async/Await

Show me the code 👉

Async/Await

A 2 4 1.021s



Flow

Flow 1

Show me the code 👉

Flow 1

```kotlin
suspend fun performRequest(request: Int): String {
    delay(100)
    if (request == 2) throw RuntimeException("💥")
    return "$request"
}
fun requestFlow() = flow {
    for (i in 1..3) {
        emit(i)
    }
}
fun main() = runBlocking {
    requestFlow()
        .map { request -> performRequest(request) }
        .catch { e -> emit( e.localizedMessage) }
        .collect { response -> print(response) }
}
```

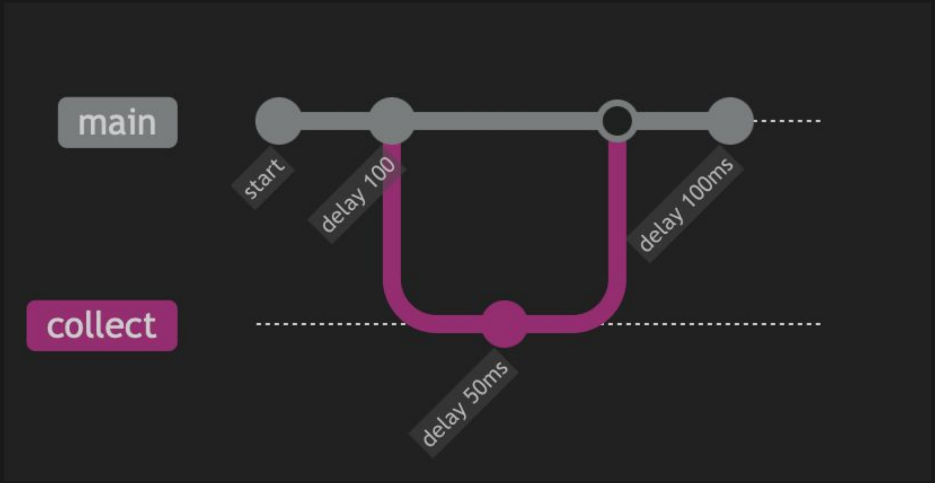
Flow 2

Show me the code 👉

Flow 2

```
fun numberFlow(): Flow<Int> = flow {
   repeat(3) {
       delay(100)
       emit(Random.nextInt(100))
   }
}

fun main(): Unit = runBlocking {
   withTimeoutOrNull(250) {
       numberFlow().collect {
           delay(50)
           println("$it ")
       }
   }
}
```



Flow 3

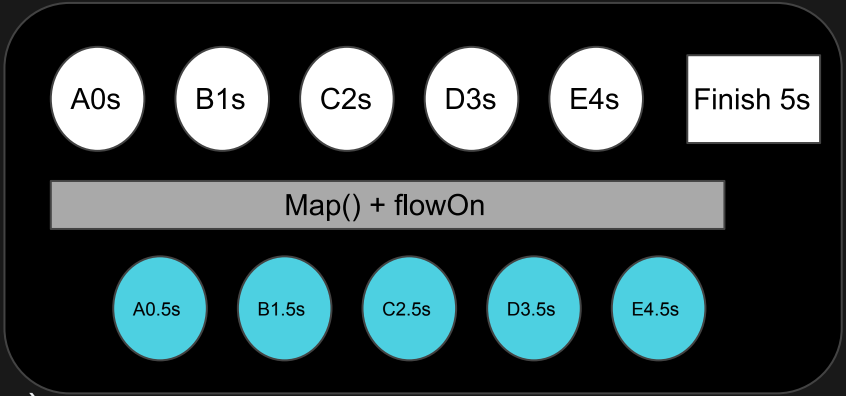
Show me the code 👉

Flow 3

```kotlin
private fun stringFlow(): Flow<String> = flow {
    ('A'..'E').forEach { char ->
        emit("$char->")
        delay(1000)
    }
}
fun main() = runBlocking {
    val time = now()
    var result = ""

    stringFlow().flowOn(Dispatchers.IO).map { item ->
        delay(500)
        item
    }.flowOn(Dispatchers.Default).collect { item ->
        result += item
    }

    print("Result: $result    ${time.passed}")
}
```



Flow 4

Show me the code 👉

Flow 4

```
private fun stringFlow(): Flow<String> = flow {
    ('A'..'E').forEach { char ->
        emit("$char->")
        delay(50)
    }
}

@OptIn(FlowPreview::class)
fun main() = runBlocking {
    val time = now()
    var result = ""
    stringFlow().flatMapMerge { value ->
        flow {
            withContext(Dispatchers.IO) {
                delay(100)
                emit(value)
            }
        }
    }.collect { item ->
        result+=item
    }

    print(result + " ${time.passed}")
}
```

```
Exception in thread "main" java.lang.IllegalStateException: Flow invariant is violated:
            Flow was collected in [StandaloneCoroutine{Active}@55b5ce3b, BlockingEventLoop@358
            but emission happened in [DispatchedCoroutine{Active}@2d18e68f, Dispatchers.IO].
            Please refer to 'flow' documentation or use 'flowOn' instead
```

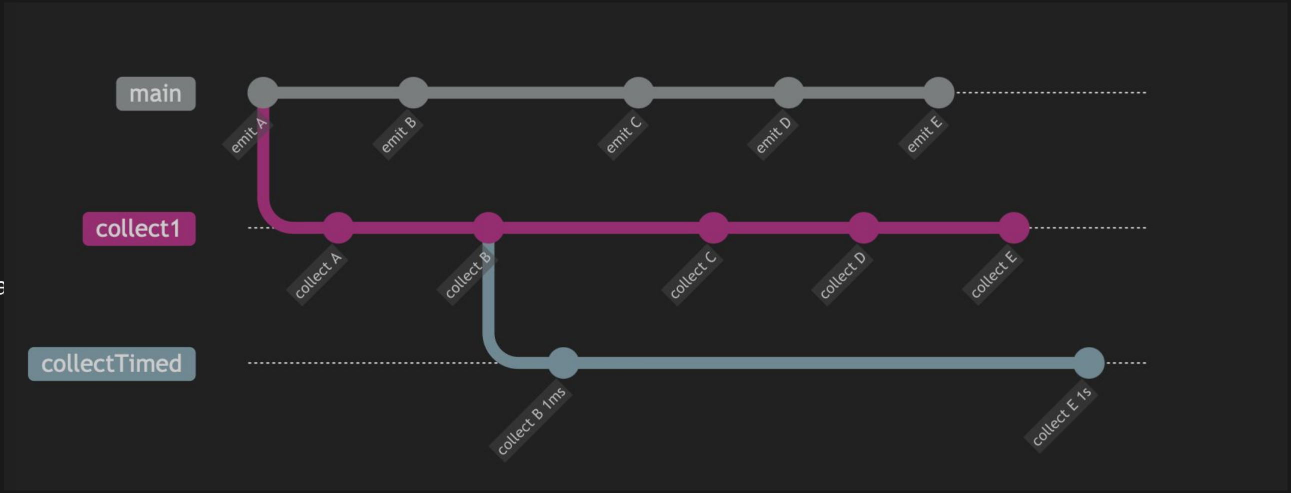
Flow 5

Show me the code 👉

Flow 5

```
fun Flow<String>.timed(delay: Duration): Flow<String> = flow {
    var time: Duration = Duration.ZERO
    buffer(1, BufferOverflow.DROP_OLDEST).collect { item ->
        if (time == Duration.ZERO) {
            time = now()
        }
        emit("${time.passed} $item")
        delay(delay)
    }
}

fun main(): Unit = runBlocking {
    val sharedFlow = stringFlow().sha
    launch {
        sharedFlow.collect {
            print(it)
        }
    }
    launch {
        sharedFlow
            .timed(1000.milliseconds)
            .collect {
                print(it)
            }
    }
}
```



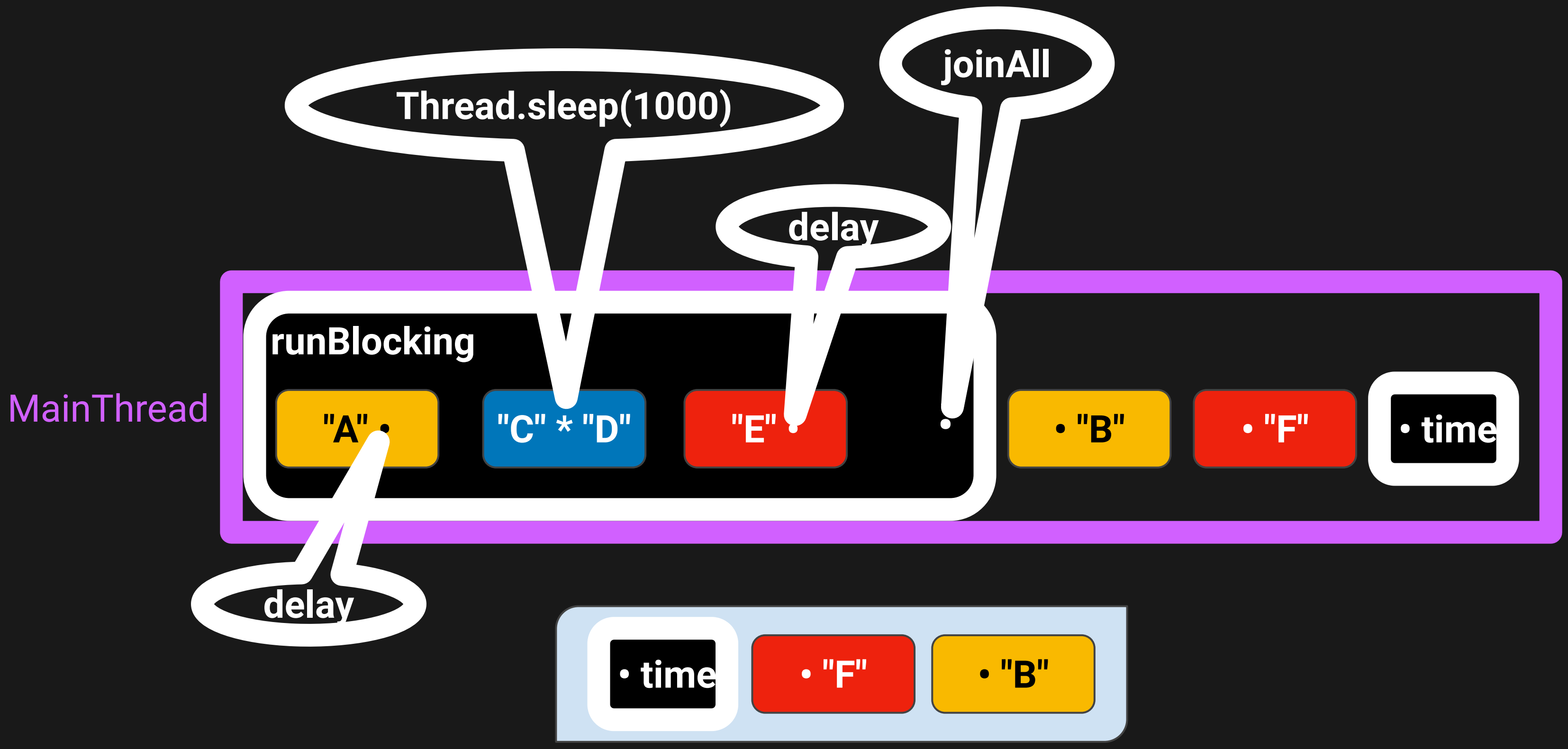
Time and Durations 🍽️

Time 1

Show me the code 👉

Time 1

ACDEBF 3.025s



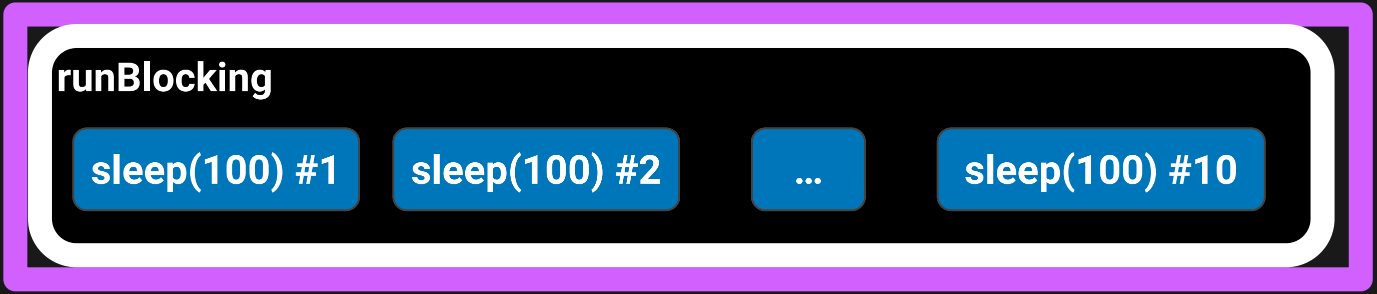
Time 2

Show me the code 👉

Time 2

Total: ~1.0s

```
repeat(10) {
    launch {
        sleep(100)
        print()
    }
}
```



Time 3

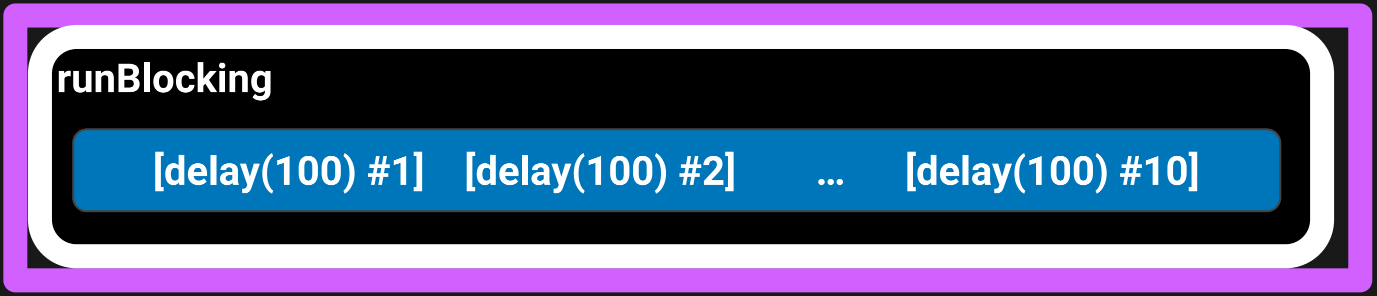
Show me the code 👉

Time 3

Total: ~1.0s

```
repeat(10) {
    launch {
        sleep(100)
        print()
    }
}
```

```
launch {
    repeat(10) {
        delay(100)
        print()
    }
}
```



Schedulers 🍱

Schedulers

```
private suspend fun heavyComputation(taskId: Int): Int {
   println("Task $taskId started")
   delay(1000L)
   println("Task $taskId comp
   return taskId
}
fun main() = runBlocking {
   val customDispatcher = newFixedThreadPoolContext(
       nThreads = 1,
       name = "CustomDispatcher"
   )
   val time = measureTimeMillis {
       val task1 = async(customDispatcher) {
           heavyComputation(1)
       }

       val task2 = async(customDispatcher) {
           heavyComputation(2)
       }

       val task3 = async(customDispatcher) {
           heavyComputation(3)
       }
       println("Result: ${task1.await() + task2.await() + task3.await()}")
   }
   println("Total time: $time ms")
}
```

Что-то ваш heavy computation не такой уж и heavy

Schedulers решение

```
private suspend fun heavyComputation(taskId: Int): Int {
   println("Task $taskId started")
   sleep(1000L)
   println("Task $taskId completed")
   return taskId
}
fun main() = runBlocking {
   val customDispatcher = newFixedThreadPoolContext(
       nThreads = 1,
       name = "CustomDispatcher"
   )
   val time = measureTimeMillis {
       val task1 = async(customDispatcher) {
           heavyComputation(1)
       }
       val task2 = async(customDispatcher) {
           heavyComputation(2)
       }
       val task3 = async(customDispatcher) {
           heavyComputation(3)
       }
       println("Result: ${task1.await() + task2.await() + task3.await()}")
   }
   println("Total time: $time ms")
}
```

Race 🧃

Race 1

Show me the code 👉

Race 1

~950_000

```
val customDispatcher = newFixedThreadPoolContext(
  nThreads = 2,
  name = "CustomDispatcher"
)
```

```
fun increment() {
 counter.count++
}
```

→

```
fun increment() {
  val oldValue = counter.count
  val newValue = oldValue + 1
  counter.count = newValue + 1
}
```

Race 2

Show me the code 👉

Race 2

~1_100

```
suspend fun increment() {
    val oldValue = counter.count
    val newValue = oldValue + 1
    delay(nextLong(0, 2))
    counter.count = newValue
}
```

Race 3

Show me the code 👉

Race 3

~950_000

```
repeat(1_000) {
   synchronized(this) {
        jobs += launch(customDispatcher) {
            repeat(1_000) {
                increment()
            }
        }
    }
}
```

Race 4

Show me the code 👉

Race 4

Final count: ~950_000

```
repeat(1_000) {
    jobs += launch(customDispatcher) { /*this: CoroutineScope*/
        repeat(1_000) {
            synchronized(this) {
                increment()
            }
        }
    }
}
```

Race 5

Show me the code 👉

Race 5

Final count: 1_000_000 in 58ms

🥂

Race 6

Show me the code 👉

Race 6

Final count: ~988_905 in 86ms

```
val semaphore = Semaphore(permits: 2)
```

Race 7

Show me the code 👉

Race 7

Final count: 1_000_000 in 439ms

🥂

Race 8

Show me the code 👉

Race 8

Final count: 1_000_000 in 472ms

🥂

Race 9

Show me the code 👉

Race 9

Final count: 998_301 in 56ms

```
repeat(1_000) {
    mutex.lock()
    jobs += launch(customDispatcher) {
        repeat(1_000) {
            increment()
        }
    }
    mutex.unlock()
}
```

Race 10

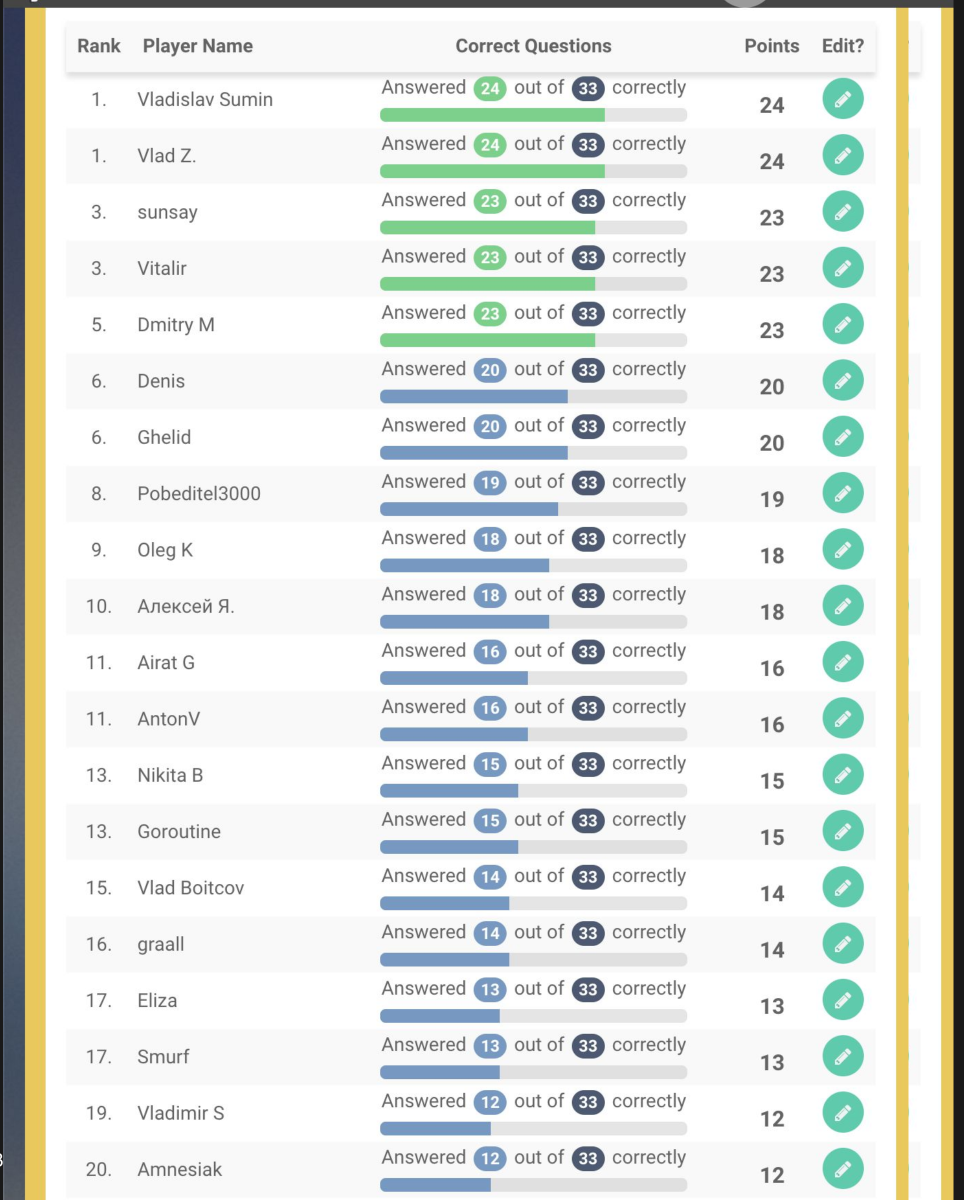
Show me the code 👉

Race 10

TimeoutCancellationException: Timed out waiting for 10000 ms

```
delay(Random.nextLong(0, 2)) // AVG = 1ms
// 1 * 1_000_000 = 1_000_000ms = 1000s ~ 16min
```

Объявление победителей



Rank	Player Name	Correct Questions	Points	Edit?
1.	Vladislav Sumin	Answered 24 out of 33 correctly	24	
1.	Vlad Z.	Answered 24 out of 33 correctly	24	
3.	sunsay	Answered 23 out of 33 correctly	23	
3.	Vitalir	Answered 23 out of 33 correctly	23	
5.	Dmitry M	Answered 23 out of 33 correctly	23	
6.	Denis	Answered 20 out of 33 correctly	20	
6.	Ghelid	Answered 20 out of 33 correctly	20	
8.	Pobeditel3000	Answered 19 out of 33 correctly	19	
9.	Oleg K	Answered 18 out of 33 correctly	18	
10.	Алексей Я.	Answered 18 out of 33 correctly	18	
11.	Airat G	Answered 16 out of 33 correctly	16	
11.	AntonV	Answered 16 out of 33 correctly	16	
13.	Nikita B	Answered 15 out of 33 correctly	15	
13.	Goroutine	Answered 15 out of 33 correctly	15	
15.	Vlad Boitcov	Answered 14 out of 33 correctly	14	
16.	graall	Answered 14 out of 33 correctly	14	
17.	Eliza	Answered 13 out of 33 correctly	13	
17.	Smurf	Answered 13 out of 33 correctly	13	
19.	Vladimir S	Answered 12 out of 33 correctly	12	
20.	Amnesiak	Answered 12 out of 33 correctly	12	

https://github.com/ziina-co/CoroutinePuzzlers

Thank you! Questions?

🙏 & 🙋

Anton Dudakov

Владимир Меркурьев

bwdude

antondudakov

vmerc

vladimir-merkurev