Inline function

- بنستخدم الـ inline function علشان نحسن الـ performance لـ function اللى من نوع higher order function

- لو فرضنا مثلا ان عندى function بالشكل ده :

```
fun myFunction(fn:()->Unit) {
    fn()
    println("code after lambda function")
}
```

- الـ compiler لما يجى يعمل compile للكود ده الـ function اللى هي fn() هيعمل object خاص بيها ولو عملت call لـ fn() كذا مرة داخل الـ function الأساسية اللى هي myFunction() فكل مرة هيعمل object جديد وده طبعاً بيستهلك resources وبيقلل الـ performance والحل إنى أستخدم الـ inline بالشكل ده :

```
inline fun myFunction(fn:()->Unit) {
    fn()
    println("code after lambda function")
}
```

- في الحالة دى الـ compile هياخد copy لـ function اللى هي fn() ويحطها في مكان الـ call وبالتالي مش هيعمل object جديد وبكده تكون الـ performance أعلى.

Non-local returns

```
fun main() {
    processRecords( ...records: "Eslam", "Ahmed", "Ali", "Mohamed")
}

fun processRecords(vararg records: String) {
    for (record in records) {
        executeAndMeasure(record) {
            save(record)
        }
    }
}

fun executeAndMeasure(label: String, block: () -> Unit) {
    val start = System.nanoTime()
    block()
    val end = System.nanoTime()
    println("Duration for $label: ${(end - start) / 1000000} ms")
}
```

الكود ده وظيفتة إنه ياخد مجموعة Strings ويعمل loop عليهم وكل واحد منهم يحسب وقت ال save بتاعة ويطبعه

```
Duration for Eslam: 2537 ms
Duration for Ahmed: 2510 ms
Duration for Ali: 1515 ms
Duration for Mohamed: 3530 ms
```

- لو عملت return في الـ lambda بالشكل ده :

```
fun processRecords(vararg records: String) {
    for (record in records) {
        executeAndMeasure(record) {
            if (record.startsWith( prefix: "A")) return
            save(record)
        }
    }
}
```

هلاحظ إنة ظهر error لانه مش عارف هيعمل هـ return من أنة function من الـ processRecords() ولا من الـ lambda

- فهعمل الـ return بالشكل ده :

```
fun processRecords(vararg records: String) {
    for (record in records) {
        executeAndMeasure(record) {
            if (record.startsWith( prefix: "A")) return@executeAndMeasure
            save(record)
        }
    }
}
```

هلاقى الـ output ظهر بالشكل ده :

```
Duration for Eslam: 2592 ms
Duration for Ahmed: 0 ms
Duration for Ali: 0 ms
Duration for Mohamed: 3532 ms
```

هلاحظ هنا إن الـ string اللى بيبدأ بـ A الـ time بتاعة zero والسبب إنة عمل return ولكن من الـ lambda وليس من الـ function الأساسية اللى هي executeAndMeasure()

- ولكن دلوقتى عايز أعمل return من الميثود processRecords() :

```
fun processRecords(vararg records: String) {
    for (record in records) {
        executeAndMeasure(record) {
            if (record.startsWith( prefix: "A")) return@processRecords
            save(record)
        }
    }
}
```

لو كتبت الـ return بالشكل دة هيحصل error ودة إسمة non-local return ومش هعرف أعمل ال return ده مع الميثود العادية ولكن أقدر أعملة مع ال inline function

```
fun processRecords(vararg records: String) {
    for (record in records) {
        executeAndMeasure(record) {
            if (record.startsWith( prefix: "A")) return@processRecords
            save(record)
        }
    }
}

inline fun executeAndMeasure(label: String, block: () -> Unit) {
    val start = System.nanoTime()
    block()
    val end = System.nanoTime()
    println("Duration for $label: ${(end - start) / 1000000} ms")
}
```

بالشكل ده هيعمل return من ال function اللى هي processRecords() وممكن أكتب return فقط من غير ال annotation وهو هيفهم

Crossinline

- لو عندى ميثود inline مينفعش أستخدم الـ lambda function parameter الخاصة بالميثود دى في أي scope بالشكل ده :

```
inline fun executeAndMeasure(label: String, block: () -> Unit) {
    val start = System.nanoTime()
    Thread{
        block()
    }
    val end = System.nanoTime()
    println("Duration for $label: ${(end - start) / 1000000} ms")
}
```

- هنا هيحصل error لان إستخدمت lambda function داخل scope اللى هو الـ Thread في ميثود inline
- أو حتى لو عملتها داخل Block بالشكل ده :

```
val x = { block() }
```

هيحصل error برضوا ولكن ممكن أعملها call في inline fun تانية زى الـ coroutine او الـ Scope function بالشكل ده :

```
inline fun executeAndMeasure(label: String, block: () -> Unit) {
    val start = System.nanoTime()
    run { block() }
    val end = System.nanoTime()
    println("Duration for $label: ${(end - start) / 1000000} ms")
}
```

- ولو مضطر إنى أعملها call في Scope معين في الحالة دى هستخدم modifier إسمة crossinline بالشكل ده :

```
inline fun executeAndMeasure(label: String,crossinline block: () -> Unit) {
    val start = System.nanoTime()
    Thread{
        block()
    }
    val end = System.nanoTime()
    println("Duration for $label: ${(end - start) / 1000000} ms")
}
```

- أنا استخدمت هنا الـ lambda داخل scope والـ scope هو ليس inline ولكن مش هيحصل error لانى مستخدم crossinline ولكن في حالة إستخدام الـ crossinline مش هينفع أستخدم ال non-local return يعنى مش هينفع أعمل return

noninline

- لما يكون عندى inline function أي lambda parameter لـ function دى بتكون هي كمان inline
- فلو عندى ميثود وكانت inline وفيها أكثر من lambda function وعايز اخلى مثلاً واحدة منهم تكون inline والثانية تكون عادية في الحالة دى هستخدم الـ modifier ده noninline بالشكل ده :

```
inline fun executeAndMeasure(label: String, block: () -> Unit, noinline block2: () -> Unit) {
    val start = System.nanoTime()
    block()
    block2()
    val end = System.nanoTime()
    println("Duration for $label: ${(end - start) / 1000000} ms")
}
```

- بالشكل ده الـ lambda الأولى inline والثانية noinline معنى كدة إنى ممكن أعملها call داخل scope مثلاً بالشكل ده :

```
inline fun executeAndMeasure(label: String, block: () -> Unit, noinline block2: () -> Unit) {
    val start = System.nanoTime()
    block()
    val x = { block2() }
    val end = System.nanoTime()
    println("Duration for $label: ${(end - start) / 1000000} ms")
}
```

ولكن مش هقدر أعمل return يعنى مش هعرف أستخدم non-local return لان دى خاصية موجودة في الـ inline فقط