Using High Level Functions to Simplify Code

Kevin Jones

@kevinrjones www.rocksolidknowledge.com

What are High Level Functions?

Functions as first class citizens

Can pass to and return from functions

Can store in collections

Strategy Pattern

Allows an algorithms behaviour to be selected at runtime

In OO the strategy patterns is often implemented using an Interface

But can just pass a function

Strategy Pattern - 1

```
interface Process { fun execute() }
fun calculateResult(a: Int, b: Int, object : Process {
    override fun execute(value: Int) {
        println(value)
    }
 })
```

Strategy Pattern - 1

```
fun calculateResult(a: Int, b: Int, func: (Int) -> Unit) {
    // calculate using a and b
    // use the result
    func(result)
}
```

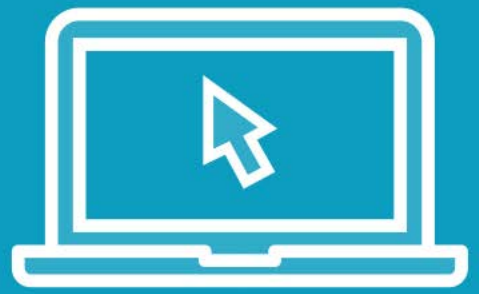
Strategy Pattern - 2

```
calculateResult(1, 2, { s -> print(s) })

calculateResult(1, 2) { s -> print(s) }
```

Strategy Pattern - 2

```
calculateResult(1, 2) { print(it) }

calculateResult(1, 2) { s -> print(s) }

calculateResult(1, 2, { print(it) })

calculateResult(1, 2, { s -> print(s) })

calculateResult(1, 2, ::print)
```

Demo



Using Lambdas

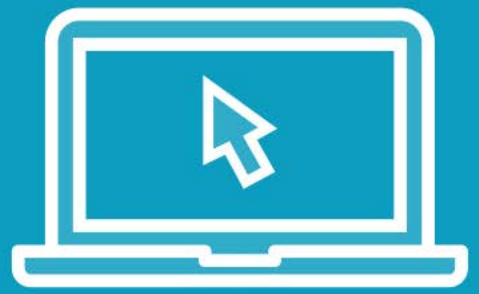
Closures

Kotlin lambdas can mutate values

Unlike Java 8 Lambdas

```
var total = 0

calculateResult(10) {total += it}

println(total)
```

Demo



Closures

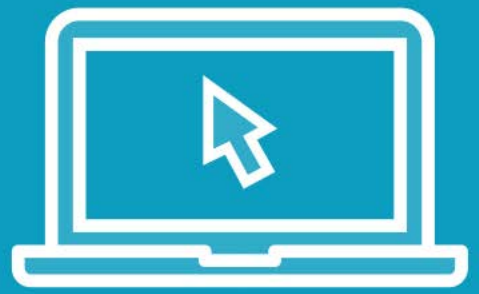
With and Apply

Used to make certain operations more natural

Look like language keywords

Actually use lambdas

Demo



Using With and Apply

Summary

- Kotlin has support for 'first-class' functions
- Can store functions in collections
- Can pass functions into other functions
- Makes using collections much easier