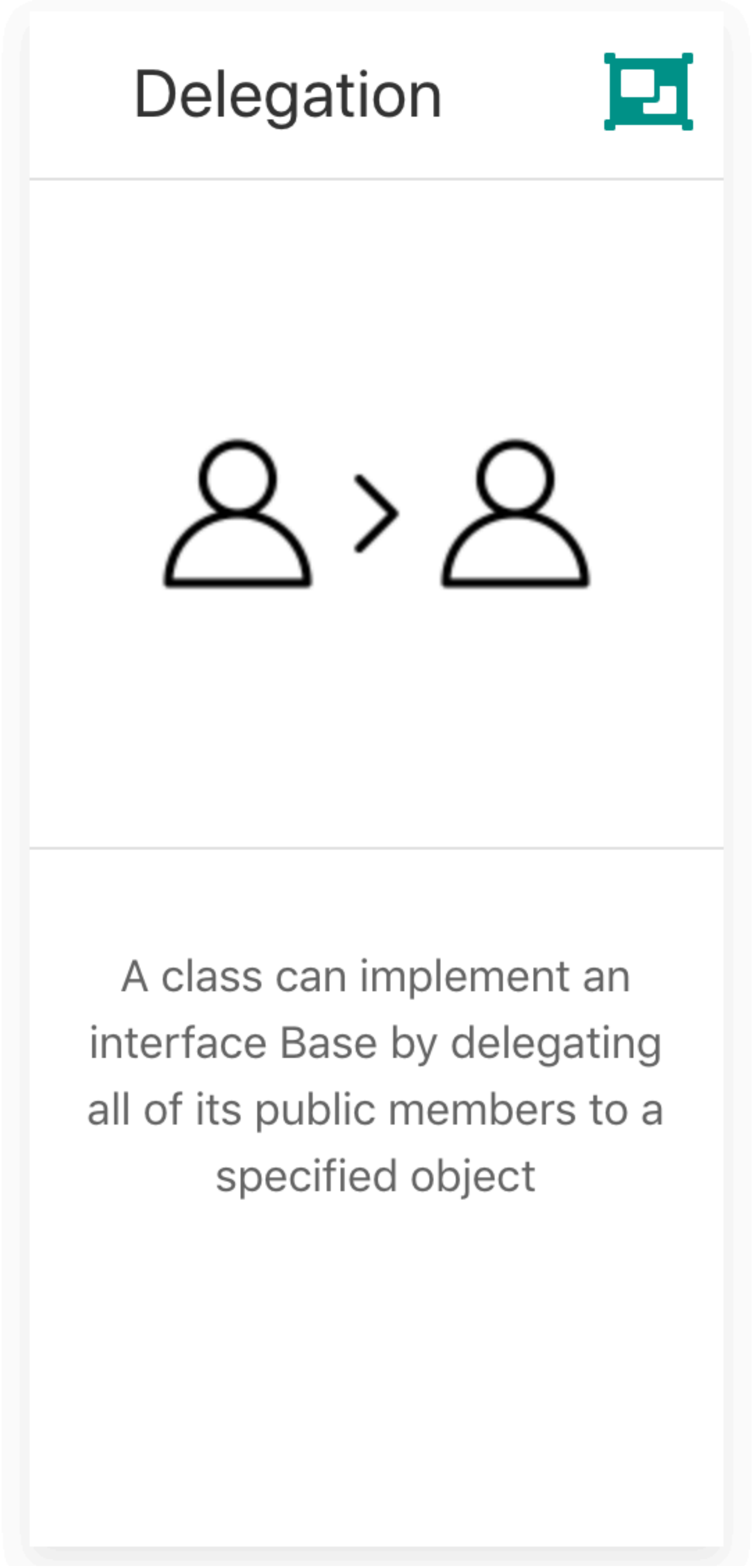
Delegation



Delegation

A class can implement an interface Base by delegating all of its public members to a specified object

Implementation by Delegation

The Delegation pattern has proven to be a good alternative to implementation inheritance, and Kotlin supports it natively requiring zero boilerplate code. A class `Derived` can implement an interface `Base` by delegating all of its public members to a specified object:

```kotlin
interface Base {
    fun print()
}

class BaseImpl(val x: Int) : Base {
    override fun print() { print(x) }
}

class Derived(b: Base) : Base by b

fun main() {
    val b = BaseImpl(10)
    Derived(b).print()
}
```

Target platform: JVM Running on kotlin v. 1.3.10

The by-clause in the supertype list for `Derived` indicates that `b` will be stored internally in objects of `Derived` and the compiler will generate all the methods of `Base` that forward to `b`.

Overriding a member of an interface implemented by delegation

Overrides work as you might expect: the compiler will use your `override` implementations instead of those in the delegate object. If we were to add `override fun printMessage() { print("abc") }` to `Derived`, the program would print "abc" instead of "10" when `printMessage` is called:

```kotlin
interface Base {
    fun printMessage()
    fun printMessageLine()
}

class BaseImpl(val x: Int) : Base {
    override fun printMessage() { print(x) }
    override fun printMessageLine() { println(x) }
}

class Derived(b: Base) : Base by b {
    override fun printMessage() { print("abc") }
}

fun main() {
    val b = BaseImpl(10)
    Derived(b).printMessage()
    Derived(b).printMessageLine()
}
```

Note, however, that members overridden in this way do not get called from the members of the delegate object, which can only access its own implementations of the interface members:

```kotlin
interface Base {
    val message: String
    fun print()
}

class BaseImpl(val x: Int) : Base {
    override val message = "BaseImpl: x = $x"
    override fun print() { println(message) }
}

class Derived(b: Base) : Base by b {
    // This property is not accessed from b's implementation of `print`
    override val message = "Message of Derived"
}

fun main() {
    val b = BaseImpl(10)
    val derived = Derived(b)
    derived.print()
    println(derived.message)
}
```