Collection types

&

Standard collections

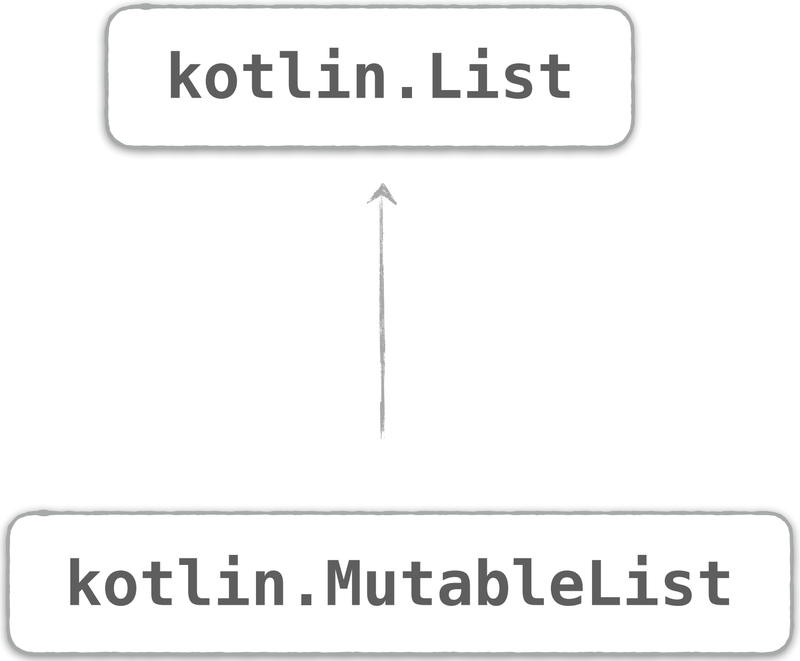
```
val set = hashSetOf(1, 7, 53)
val list = arrayListOf(1, 7, 53)
val map = hashMapOf(1 to "one",
        7 to "seven", 53 to "fifty-three")
```

```
println(set.javaClass)      class java.util.HashSet
println(list.javaClass)     class java.util.ArrayList
println(map.javaClass)      class java.util.HashMap
```

kotlin.List vs java.util.List

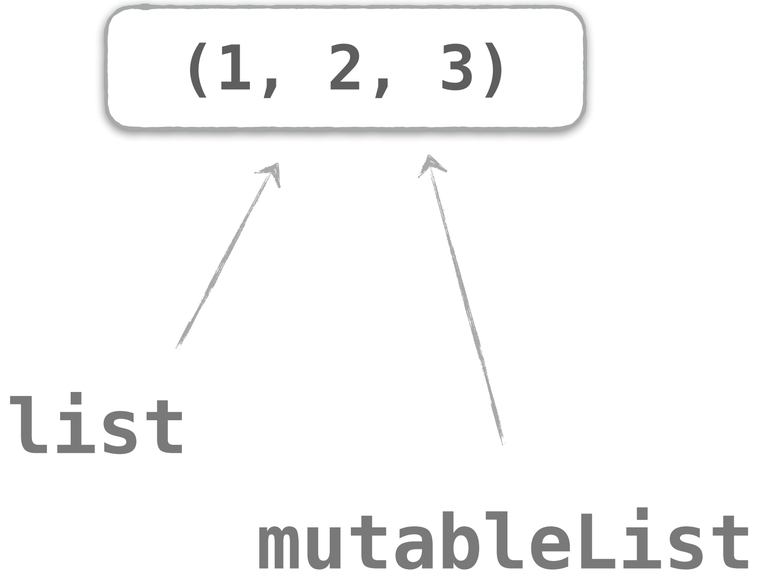
List & MutableList

- Two interfaces declared in `kotlin` package
- `MutableList` extends `List`



Read-only ≠ immutable

- Read-only interface just lacks mutating methods
- The actual list can be changed by another reference



What happens when running the following code?

```
#1  val mutableList = mutableListOf(1, 2, 3)
#2  val list: List<Int> = mutableList
#3  mutableList.add(4)
#4  println(list)
```

1. Line #2 doesn't compile
2. Line #3 doesn't compile
3. `[1, 2, 3]` is printed
4. `[1, 2, 3, 4]` is printed

What happens when running the following code?

```
#1  val mutableList = mutableListOf(1, 2, 3)
#2  val list: List<Int> = mutableList
#3  mutableList.add(4)
#4  println(list)
```

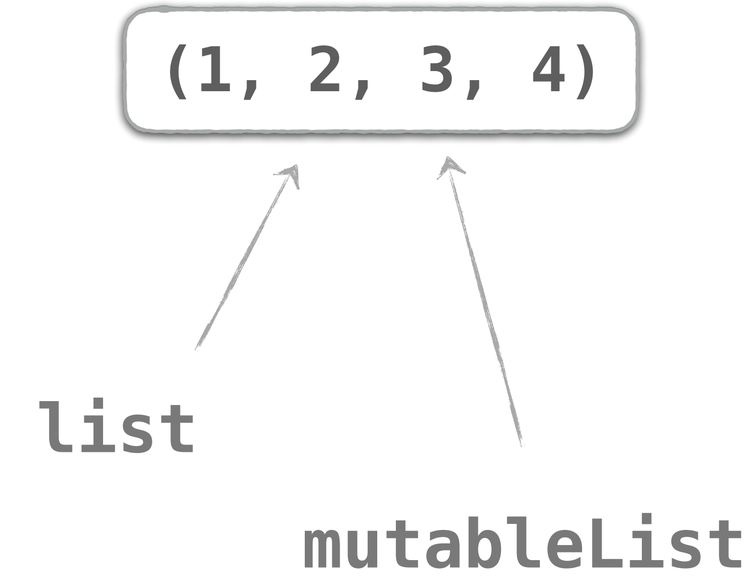
1. Line #2 doesn't compile
2. Line #3 doesn't compile
3. `[1, 2, 3]` is printed
4. `[1, 2, 3, 4]` is printed

Read-only ≠ immutable

```
val mutableList = mutableListOf(1, 2, 3)
val list: List<Int> = mutableList

println(list)  // [1, 2, 3]
list.add(4)

mutableList.add(4)
println(list)  // [1, 2, 3, 4]
```



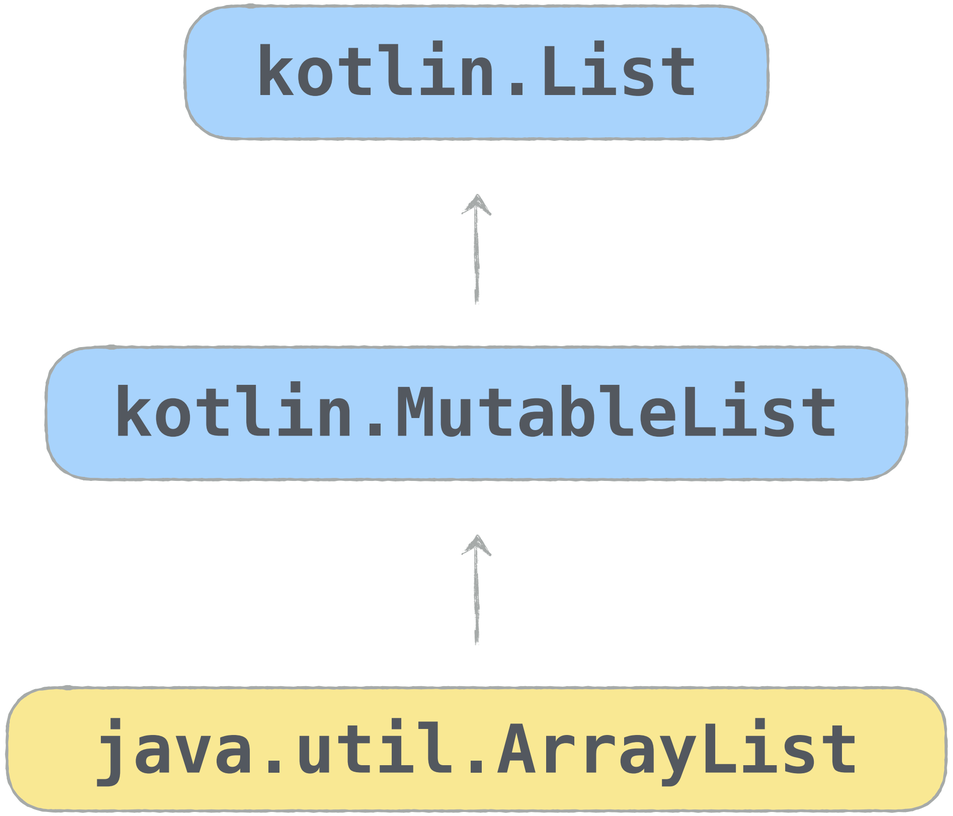
Under the hood

```
fun getNames(): List<String>
fun getNames(): MutableList<String>
```

Both functions are compiled to:

```
java.util.List<String> getNames();
```

Kotlin uses `java.util.ArrayList` under the hood



Platform type for Java collection

notation, not syntax

`List<String>` → `(Mutable)List<String!>`

type that came from Java

collection type of “unknown” mutability

Read-only interfaces improve API

```
object Shop {
    private val customers = mutableListOf<Customer>()
    fun getCustomers(): List<Customer> = customers
}
```

```
val customers = Shop.getCustomers()
customers.add()
```

you can’t shoot yourself
in the foot any longer

Collection platform types: summary

Good compromise between
safety and convenience