Android App Development with Kotlin

Collections, various operators

Type checking/casting

1

1

Using Collections

- You can pretty much use any collection available in Java directly from within Kotlin. In addition, Kotlin offers a few **view interfaces** over Java collections.
  - `Pair, Triple, Sets, Maps`
- If the application is simple and single threaded, you may use **mutable interfaces**.
- For more complex behaviors, functional and asynchronous programming, the **immutable interfaces** are safer to use

2

2

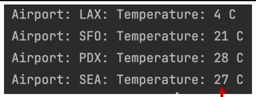
Using Pair and Triple

```
// Pair: a tuple of two values
// Triple: a triple of three values

// Create an instance of Pair using the constructor
println (Pair ("Tom","Jerry"))

// to() extension function, available on any object in Kotlin.
// create pairs of entries for a Map.

// function to() creates an instance of Pair,
// the first value is the key and the second is the value.

println (mapOf ("Tom" to "Cat", "Jerry" to "Mouse"))
println("$mapExample\n")
```

3

3

How to map airport code with temperature?



```
Airport: LAX: Temperature: 4 C
Airport: SFO: Temperature: 21 C
Airport: PDX: Temperature: 28 C
Airport: SEA: Temperature: 27 C
```

random temp

```
val airportCode = listOf("LAX", "SFO", "PDX", "SEA")
println (airportCode)
```

Collect the temperature values for different airport codes
Iterate over the collection airportCodes using the functional-style map(): `airportCode.map{ }`

Transform each airport code in the list to the pair of (code, temperature). The result is a list of Pair<String,String>.

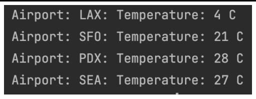
Loop through the values in the list of Pairs
For each Pair, obtain the two contained values using the first and second property

https://kotlinlang.org/docs/collection-transformations.html#map

4

4

How to map airport code with temperature?



```
Airport: LAX: Temperature: 4 C
Airport: SFO: Temperature: 21 C
Airport: PDX: Temperature: 28 C
Airport: SEA: Temperature: 27 C
```

```
val airportCode = listOf("LAX", "SFO", "PDX", "SEA")
println (airportCode)
```

```
val temperatures =
     airportCode.map {code -> code to getTemperatureAtAirport(code) }

for (temp in temperatures) {
    println ("Airport: ${temp.first}: Temperature: ${temp.second}")
}
```

```
fun getTemperatureAtAirport(code:String):String =
    "${(Math.random() * 30).roundToInt() + code.count()} C"
```

5

5

listOf() vs. mutableListOf()

```
val fruits: List<String> = listOf("Apple", "Banana", "Grape")
println (fruits)

// use .get() or []
println ("${fruits.get(1)}, ${fruits[2]}")
println(fruits.contains("Apple"))   // Java style
println("Apple" in fruits)          // Kotlin style
// fruits.add("Orange")    // ERROR

val fruits2 = fruits + "Orange"
println(fruits2)
val fruits3 = fruits2 - "Banana"
println(fruits3)
```

```
val fruitsMutable: MutableList<String> =
                            mutableListOf("Apple", "Banana", "Grape")
fruitsMutable.add("Tomato")
println (fruitsMutable)
```

6

6

Map from java.util.Map<T>

```
// Maps : associated dictionary or map of keys and values
// mapOf(), mutableMapOf() from java.util.Map<T>
// hashMapOf() from java.utill.HashMap<T>
// linkedMapOf() from java.utill.LinkedHashMap<T>
// sortedMapOf() from java.utill.SortedMap<T>

val sites: Map<String,String> =
         mapOf(”google" to "https://www.google.com",
               "agiledeveloper" to "https://www.agiledeveloper.com")
println(sites.size)

// Iterate Map: Java style
println(sites.containsKey("agiledeveloper"))
println(sites.containsValue("https://www.agiledeveloper.com"))

// Iterate Map: Kotlin style
println("agiledeveloper" in sites)

// false, only check keys, not values
println("https://www.agiledeveloper.com" in sites)
```

```
println(sites["google"])
// Type is String?, not String
val googleSite = sites["google"]
// Will not work because it returns a nullable type
val googleSite: String = sites["google"]
println(googleSite)

// The nullable types have a ? suffix
println(sites["Microsoft"]) // Will print null
```

```
// Alternatively,
// we can avoid the nullable reference type by providing
// an alternate, default value if the key doesn't exist
// Type is String (not String?)
val googleSite2 =
       sites.getOrDefault("Microsoft","http://www.bridgew.edu")
println("Will print default value $googleSite2\n")
```

```
val sitesWithExample = sites + ("example" to "http://www.example.com")
for (entry in sitesWithExample) {
    println ("${entry.key} ... ${entry.value}")
}


val withoutAgileDeveloper = sitesWithExample - "agiledeveloper"
for ((key, value) in withoutAgileDeveloper) {
    println ("$key ... $value")
}
```

Nullable Reference

- If a function that returns an object does not have anything to return at times, then the solution in Java is to return `null`.
  - This, unfortunately, leads to an `NullPointerException` **if programmer forgets to check `null`.**
- Refer the following site for details
  - https://kotlinlang.org/docs/null-safety.html

```
// Kotlin compiler complain because
// 1) you assigned null to a non-nullable reference or
// 2) returning null where the reference type is non-nullable
println(nickName0(null))    //ERROR
fun nickName0(name: String): String {
    if (name != null)
        return name.reversed()
    return null //ERROR
}
```

Similar code in Java would have compiled and then failed at runtime

```
// Kotlin compiler complain because
// 1) you assigned null to a non-nullable reference or
// 2) returning null where the reference type is non-nullable
println(nickName0(null))    //ERROR
fun nickName0(name: String): String {
    if (name != null)
        return name.reversed()
    return null //ERROR
}
```

Similar code in Java would have compiled and then failed at runtime

Fix the issues, starting to return null

```
println(nickName1("William"))
println(nickName1("Jacob"))
println(nickName1(null))

fun nickName1(name: String?): String?{
    if (name != null)
        return name.reversed()
    return null
}
```

But checking null is noisy

13

13

Safe-Call Operator ?

```
if (name != null)
  return name.reversed()
return null
```

Return `name?.reversed()`

```
println(nickName2("William"))
println(nickName2("Jacob"))
println(nickName2(null))

fun nickName2(name:String?): String? = name?.reversed()
```

Safe-call operator returns null if the target is null

14

14

Safe-Call Operator ?

```
if (name != null)
  return name.reversed()
return null
```

Return `name?.reversed()`

```
println(nickName2("William"))
println(nickName2("Jacob"))
println(nickName2(null))

fun nickName2(name:String?): String? = name?.reversed()
```

Safe-call operator returns null if the target is null

```
println(nickName3("William"))
println(nickName3("Jacob"))
println(nickName3(null))

fun nickName3(name:String?): String? = name?.reversed()?.uppercase()
```

15

15

Safe-Call Operator ?

```
if (name != null)
  return name.reversed()
return null
```

Return `name?.reversed()`

```
println(nickName2("William"))
println(nickName2("Jacob"))
println(nickName2(null))

fun nickName2(name:String?): String? = name?.reversed()
```

Safe-call operator returns null if the target is null

```
println(nickName3("William"))
println(nickName3("Jacob"))
println(nickName3(null))

fun nickName3(name:String?): String? = name?.reversed()?.uppercase()
```

But, what if we want to return something else, a non-null?
we can combine multiple calls to the safe-call operator

16

16

```
// takes String? to allow null arguments,
// but the return type is String.

fun nickName4(name:String?): String{
    val result = name?.reversed()?.uppercase()

    if (result == null)
        return "Joker"
    else
        return result
}
```

17

17

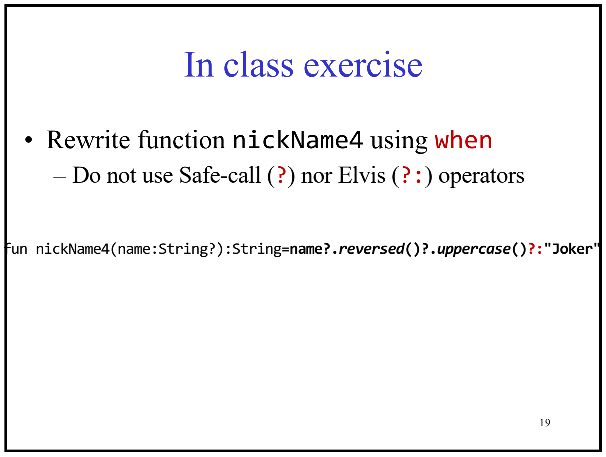
Elvis Operator ?:

```
// takes String? to allow null arguments,
// but the return type is String.

fun nickName4(name:String?): String{
    val result = name?.reversed()?.uppercase()

    if (result == null)
        return "Joker"
    else
        return result
}
```

```
fun nickName4(name:String?):String=name?.reversed()?.uppercase()?:"Joker
```

18

18



In class exercise

- Rewrite function `nickName4` using `when`
  - Do not use Safe-call (`?`) nor Elvis (`?:`) operators

```
fun nickName4(name:String?):String=name?.reversed()?.uppercase()?:"Joker"
```

19

19