



Kotlin

Rafael Vieira Coelho
rafaelvc2@gmail.com

<https://kotlinlang.org>

1) Listas (MutableList ou List)

- Trata-se de uma coleção ordenada de items.
- Elas podem ser mutáveis (MutableList) ou apenas de leitura (List).
- Para criar uma mutável, usa-se o método `mutableListOf()`
- Para criar uma de leitura, usa-se o método `ListOf()`

1. Cria um MutableList

2. Cria um List (leitura)

3. Adiciona um item no MutableList

4. Retorna a List

5. Atualiza o MutableList

6. Obtém o tamanho da List

7. Itera a lista e mostra os elementos

8. Tenta escrever no List (apenas leitura)

```
1  val systemUsers: MutableList<Int> = mutableListOf(1, 2, 3)          // 1
2  val sudoers: List<Int> = systemUsers                                // 2
3
4  fun addSudoer(newUser: Int) {
5      systemUsers.add(newUser)
6  }
7
8  fun getSysSudoers(): List<Int> {                                         // 4
9      return sudoers
10 }
11 |
12 ➤ fun main() {
13     addSudoer( newUser: 4)                                                 // 5
14     println("Tot sudoers: ${getSysSudoers().size}")                         // 6
15     getSysSudoers().forEach {                                              // 7
16         i -> println("Some useful info on user $i")
17     }
18     // getSysSudoers().add(5) <- Error!                                     // 8
19 }
```

```
12 ► fun main() {  
13     addSudoer(newUser: 4) // 5  
14     println("Tot sudoers: ${getSysSudoers().size}") // 6  
15     getSysSudoers().forEach { // 7  
16         i -> println("Some useful info on user $i")  
17     }  
18     // getSysSudoers().add(5) <- Error! // 8  
19 }
```

```
main()
```

Run: Ex1ListKt   

```
"/Applications/IntelliJ IDEA CE.app/Contents/jbr/Contents/Home/bin/java" -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/intellij-idea-ce-agent.jar=port=5005  
Tot sudoers: 4  
Some useful info on user 1  
Some useful info on user 2  
Some useful info on user 3  
Some useful info on user 4  
  
Process finished with exit code 0
```

2) Conjuntos (Set ou MutableSet)

- Trata-se de uma coleção não-ordenada de elementos que não suporta elementos duplicados.
- Duas funções podem ser usadas para criar: `setOf()` ou `mutableSetOf()`

```
1  val openIssues: MutableSet<String> = mutableSetOf("uniqueDescr1",
2      "uniqueDescr2", "uniqueDescr3") // 1
3
4  fun addIssue(uniqueDesc: String): Boolean {
5      return openIssues.add(uniqueDesc) // 2
6  }
7
8  fun getStatusLog(isAdded: Boolean): String {
9      return if (isAdded) "registered correctly." else "marked as duplicate and rejected." // 3
10 }
11
12 ➤ fun main() {
13     val aNewIssue: String = "uniqueDescr4"
14     val anIssueAlreadyIn: String = "uniqueDescr2"
15
16     println("Issue $aNewIssue ${getStatusLog(addIssue(aNewIssue))}") // 4
17     println("Issue $anIssueAlreadyIn ${getStatusLog(addIssue(anIssueAlreadyIn))}") // 5
18 }
```

Conjuntos (Set)

1. Adiciona um elemento ao Set

2. Retorna um boolean que determina se o elemento foi adicionado.

```
1  val openIssues: MutableSet<String> = mutableSetOf("uniqueDescr1",
2      "uniqueDescr2", "uniqueDescr3") // 1
3
4  fun addIssue(uniqueDesc: String): Boolean {
5      return openIssues.add(uniqueDesc) // 2
6  }
7
8  fun getStatusLog(isAdded: Boolean): String {
9      return if (isAdded) "registered correctly." else "marked as duplicate and rejected." // 3
10 }
11
12 ➤ fun main() {
13     val aNewIssue: String = "uniqueDescr4"
14     val anIssueAlreadyIn: String = "uniqueDescr2"
15
16     println("Issue $aNewIssue ${getStatusLog(addIssue(aNewIssue))}") // 4
17     println("Issue $anIssueAlreadyIn ${getStatusLog(addIssue(anIssueAlreadyIn))}") // 5
18 }
```

Conjuntos (Set)

3. Retorna uma String indicando se foi cadastrado ou rejeitado por ser duplicada.

4. Mostra mensagem de sucesso da adição de um novo elemento no Set.

```
1  val openIssues: MutableSet<String> = mutableSetOf("uniqueDescr1",
2      "uniqueDescr2", "uniqueDescr3") // 1
3
4  fun addIssue(uniqueDesc: String): Boolean {
5      return openIssues.add(uniqueDesc) // 2
6  }
7
8  fun getStatusLog(isAdded: Boolean): String {
9      return if (isAdded) "registered correctly." else "marked as duplicate and rejected." // 3
10 }
11
12 ➤ fun main() {
13     val aNewIssue: String = "uniqueDescr4"
14     val anIssueAlreadyIn: String = "uniqueDescr2"
15
16     println("Issue $aNewIssue ${getStatusLog(addIssue(aNewIssue))}") // 4
17     println("Issue $anIssueAlreadyIn ${getStatusLog(addIssue(anIssueAlreadyIn))}") // 5
18 }
```

Conjuntos (Set)

5. Mostra mensagem de falha, indicando que o elemento já foi inserido previamente no Set.

Conjuntos (Set)

```
12 ► fun main() {  
13     val aNewIssue: String = "uniqueDescr4"  
14     val anIssueAlreadyIn: String = "uniqueDescr2"  
15  
16     println("Issue $aNewIssue ${getStatusLog(addIssue(aNewIssue))}")           // 4  
17     println("Issue $anIssueAlreadyIn ${getStatusLog(addIssue(anIssueAlreadyIn))}") // 5  
18 }
```

Run: **Ex2SetKt** 

```
► ↑ "/Applications/IntelliJ IDEA CE.app/Contents/jbr/Contents/Home/bin/java" -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=55178:/Applications/IntelliJ IDEA CE.app/Contents/bin  
Issue uniqueDescr4 registered correctly.  
Issue uniqueDescr2 marked as duplicate and rejected.  
Process finished with exit code 0
```

3) Mapa (Map ou MutableMap)

- Trata-se uma coleção de pares de valores (chave/valor)
- Funções de criação: mapOf() ou mutableMapOf()

```
1 const val POINTS_X_PASS: Int = 15
2 val EZPassAccounts: MutableMap<Int, Int> = mutableMapOf(1 to 100, 2 to 100, 3 to 100) // 1
3 val EZPassReport: Map<Int, Int> = EZPassAccounts // 2
4
5 fun updatePointsCredit(accountId: Int) {
6     if (EZPassAccounts.containsKey(accountId)) { // 3
7         println("Updating $accountId...")
8         EZPassAccounts[accountId] = EZPassAccounts.getValue(accountId) + POINTS_X_PASS // 4
9     } else {
10        println("Error: Trying to update a non-existing account (id: $accountId)")
11    }
12 }
13 }
```

Mapa

1. Cria um mapa mutável

2. Cria um mapa de apenas leitura

```
1 const val POINTS_X_PASS: Int = 15
2 val EZPassAccounts: MutableMap<Int, Int> = mutableMapOf(1 to 100, 2 to 100, 3 to 100) // 1
3 val EZPassReport: Map<Int, Int> = EZPassAccounts // 2
4
5 fun updatePointsCredit(accountId: Int) {
6     if (EZPassAccounts.containsKey(accountId)) { // 3
7         println("Updating $accountId...")
8         EZPassAccounts[accountId] = EZPassAccounts.getValue(accountId) + POINTS_X_PASS // 4
9     } else {
10        println("Error: Trying to update a non-existing account (id: $accountId)")
11    }
12 }
13 }
```

Mapa

3. Verifica se a chave está dentro do mapa

4. Obtém o valor correspondente a chave e o incrementa em 15.

```
14  fun accountsReport() {  
15      println("EZ-Pass report:")  
16      EZPassReport.forEach {  
17          k, v -> println("ID $k: credit $v")  
18      }  
19  }  
20  |  
21 >  fun main() {  
22      accountsReport()  
23      updatePointsCredit( accountId: 1) // 6  
24      updatePointsCredit( accountId: 1)  
25      updatePointsCredit( accountId: 5)  
26      accountsReport()  
27  }
```

Mapa

5. Itera com o `forEach` o mapa e mostra na tela os pares chave/valor.

6. Obtém os valores antes das atualizações.

9. Mostra os valores após as atualizações de valor (linhas 23 a 25).

Mapa

```
21 > fun main() {  
22     accountsReport()  
23     updatePointsCredit(accountId: 1)
```

Run: **Ex3MapKt** ×

```
EZ-Pass report:  
ID 1: credit 100  
ID 2: credit 100  
ID 3: credit 100  
Updating 1...  
Updating 1...  
Error: Trying to update a non-existing account (id: 5)  
EZ-Pass report:  
ID 1: credit 130  
ID 2: credit 100  
ID 3: credit 100
```

Process finished with exit code 0

4) Filtro (filter)

- A função filter permite filtrar coleções para restringi-las com base em um predicado (função lambda) que é aplicado para cada elemento da coleção.
- Somente os elementos que retornaram true ao predicado são adicionados na coleção retornada.

Filtros

```
1 ► fun main() {  
2     val numbers = listOf(1, -2, 3, -4, 5, -6)          // 1  
3  
4     val positives = numbers.filter { x -> x > 0 }    // 2  
5  
6     val negatives = numbers.filter { it < 0 }           // 3  
7  
8     println(numbers)  
9     println(positives)  
10    println(negatives)  
11 }
```

1. Define a coleção de números.
2. Obtém os valores positivos.
3. Obtém os valores negativos a partir da notação reduzida `it`.

Filtros

```
1 ► fun main() {  
2     val numbers = listOf(1, -2, 3, -4, 5, -6)           // 1  
3  
4     val positives = numbers.filter { x -> x > 0 }    // 2  
5  
6     val negatives = numbers.filter { it < 0 }          // 3  
7  
8     println(numbers)  
9     println(positives)  
10    println(negatives)  
11 }
```

Run:  Ex4FilterKt ×

▶ 
"/Applications/IntelliJ IDEA CE.ap
[1, -2, 3, -4, 5, -6]

◀ 
[1, 3, 5]

✖ 
[-2, -4, -6]

▶ 
Process finished with exit code 0

5) Função de Extensão map

- Permite aplicar uma transformação (definida em um lambda) em todos os elementos de uma coleção.

```
1 ► fun main() {  
2     val numbers = listOf(1, -2, 3, -4, 5, -6)      // 1  
3  
4     val doubled = numbers.map { x -> x * 2 }      // 2  
5  
6     val tripled = numbers.map { it * 3 }           // 3  
7  
8     println(numbers)  
9     println(doubled)  
10    println(tripled)  
11 }
```

Run:  Ex5mapKt ×

"/Applications/IntelliJ IDEA CE.app"
[1, -2, 3, -4, 5, -6]
[2, -4, 6, -8, 10, -12]
[3, -6, 9, -12, 15, -18]

Process finished with exit code 0

Função de Extensão map

6) Ordenação

```
1 ► fun main() {  
2     val shuffled = listOf(5, 4, 2, 1, 3)      // 1  
3  
4     val natural = shuffled.sorted()          // 2  
5  
6     val inverted = shuffled.sortedBy { -it } // 3  
7  
8     println(shuffled)  
9     println(natural)  
10    println(inverted)  
11 }
```

Run: Ex6SortedKt

Icon	Action
▶	Run
↑	Up
↓	Down
⟳	Reset
⤵	Stop
⤷	Step Into
⤸	Step Over
⤹	Step Out
🖨	Print

/Applications/IntelliJ IDEA CE.ap
[5, 4, 2, 1, 3]
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]

Process finished with exit code 0

1. Define a coleção de números desordenados.

2. Ordena pela ordem natural.

3. Usa a notação reduzida it e ordena inversamente os elementos.



[**https://github.com/rafael-vieira-coelho/kotlin/tree/Aula-10---
Coleções**](https://github.com/rafael-vieira-coelho/kotlin/tree/Aula-10---Coleções)

rafaelvc2@gmail.com