

Creating Collections

Arrays

Simple Array	<code>val intArray: Array< Int> = arrayOf(1, 2, 3)</code>
Copy of Array	<code>val copyOf Array: Array< Int> = intArr ay.c op yOf()</code>
Partial copy of Array	<code>val partia lCo pyO fArray: Array< Int> = intArr ay.c op yOf Ran ge(0, 2)</code>

Lists

Simple List	<code>val intList: List<I nt> = listOf(1, 2, 3)</code>	<code>Or arrayL ist Of(1,2,3)</code>
Empty List	<code>val emptyList: List<I nt> = emptyL ist()</code>	<code>Or listOf()</code>
List with no null elements	<code>val listWi thN onN ull Ele ments: List<I nt> = listOf Not Null(1, null, 3)</code>	<code>same as List(1,3)</code>

Sets

Simple Set	<code>val aSet: Set<In t> = setOf(1)</code>	<code>Or hashSe tOf(1) /linked Ser Of(1)</code>
Empty Set	<code>val emptySet: Set<In t> = emptySet()</code>	<code>Or setOf() /hashSe tOf() /linked SetOf()</code>

Maps

Simple Map	<code>val aMap: Map<St ring, Int> = mapOf(" hi" to 1, " hel lo" to 2)</code>	<code>Or mapOf(Pai r("h i", 1) /hashMa pOf ("hi " t o 1) /linked Map Of(" - hi" to 1)</code>
Empty Map	<code>val emptyMap: Map<St ring, Int> = emptyMap()</code>	<code>Or mapOf() /hashMa pOf() /linked MapOf()</code>

Black sheep, mutables

Simple Mutable List	<code>val mutabl eList: Mutabl eLi st< Int> = mutabl eLi stOf(1, 2, 3)</code>
Simple Mutable Set	<code>val mutabl eSet: Mutabl eSe t<I nt> = mutabl eSe tOf(1)</code>
Simple Mutable Map	<code>var mutabl eMap: Mutabl eMa p<S tring, Int> = mutabl eMa pOf ("hi " to 1, " hel lo" to 2)</code>

We will be using these collections throughout the cheat sheet.



By **Xantier**
cheatography.com/xantier/

Published 14th June, 2017.
 Last updated 14th June, 2017.
 Page 1 of 15.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Operators

Method	Example	Result	Explanation
<i>Iterables</i>			
Plus	<code>intList + 1</code>	<code>[1, 2, 3, 1]</code>	Returns a new iterables with old values + added one
Plus (Iterable)	<code>intList + listOf(1, 2, 3)</code>	<code>[1, 2, 3, 1, 2, 3]</code>	Return a new iterables with old values + values from added iterable
Minus	<code>intList - 1</code>	<code>[2, 3]</code>	Returns a new iterables with old values - subtracted one
Minus (Iterable)	<code>intList - listOf(1, 2)</code>	<code>[3]</code>	Returns a new iterables with old values - values from subtracted iterable
<i>Maps</i>			
Plus	<code>aMap + Pair("H i", 2)</code>	<code>{hi=1, hello=2, Goodbye=3}</code>	Returns new map with old map values + new Pair. Updates value if it differs
Plus (Map)	<code>aMap + mapOf(Pair("h ell o", 2), Pair("G ood by e ", 3))</code>	<code>{hi=1, hello=2, Goodbye=3}</code>	Returns new map with old map values + Pairs from added map. Updates values if they differ.
Minus	<code>aMap - Pair("H i", 2)</code>	<code>{Hi=2}</code>	Takes in a key and removes if found
Minus (Map)	<code>aMap - listOf("hello", "hi")</code>	<code>{}</code>	Takes in an iterable of keys and removes if found
<i>Mutables</i>			
Minus Assign	<code>mutableList -= 2</code>	<code>[1, 3]</code>	Mutates the list, removes element if found. Returns boolean
Plus Assign	<code>mutableList += 2</code>	<code>[1, 3, 2]</code>	Mutates the list, adds element. Returns boolean



By Xantier
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 2 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Operators (cont)

Minus Assign (MutableMap)	<code>mutableMap { hi=1 }</code>	Takes in key and removes if that is found from the mutated map. Returns boolean. Same as --
Plus Assign (MutableMap)	<code>mutableMap { hi=1, Goodbye=3 }</code>	Takes in key and adds a new pair into the mutated map. Returns boolean. Same as +=

Transformers

Method	Example	Result	Explanation
Associate	<code>intList.associate { Pair(it.toString(), it) }</code>	<code>{1=1, 2=2, 3=3}</code>	Returns a Map containing key-value pairs created by lambda
Map	<code>intList.map { it + 1 }</code>	<code>[2, 3, 4]</code>	Returns a new list by transforming all elements from the initial Iterable.
MapNotNull	<code>intList.mapNotNull { null }</code>	<code>[]</code>	Returned list contains only elements that return as not null from the lambda
MapIndexed	<code>intList.mapIndexed { idx, value -> if (idx == 0) value + 1 else value + 2 }</code>	<code>[2, 4, 5]</code>	Returns a new list by transforming all elements from the initial Iterable. Lambda receives an index as first value, element itself as second.
MapIndexedNotNull	<code>intList.mapIndexedNotNull { idx, value -> if (idx == 0) null else value + 2 }</code>	<code>[4, 5]</code>	Combination of Map, MapIndexed & MapIndexedNotNull
MapKeys	<code>aMap.mapKeys { pair -> pair.key + ", mate" }</code>	<code>{hi, mate=1, hello, mate=2}</code>	Transforms all elements from a map. Receives a Pair to lambda, lambda return value is the new key of original value



By Xantier
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 3 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopad.com>

Transformers (cont)

MapValues	<code>aMap.m apV alues { pair -> pair.value + 2 }}</code>	<code>{hi=3, hello=4}</code>	Transforms all elements from a map. Receives a Pair to lambda, lambda return value is the new value for the original key.
Reversed	<code>intLis t.r eve rsed()</code>	<code>[3,2,1]</code>	
Partition	<code>intLis t.p art ition { it > 2 }}</code>	<code>Pair([1,2], [3])</code>	Splits collection into to based on predicate
Slice	<code>intLis t.s lic e(1..2)</code>	<code>[2,3]</code>	Takes a range from collection based on indexes
Sorted	<code>intLis t.s ort ed()</code>	<code>[1,2,3]</code>	
SortedByD- escending	<code>intLis t.s ort edB yDe sce nding { it }</code>	<code>[3,2,1]</code>	Sorts descending based on what lambda returns. Lambda receives the value itself.
SortedWith	<code>intLis t.s ort edW ith (Co mpa rat or< Int> { x, y -> when { x == 2 -> 1 y == 2 -> -1 else -> y - x } })</code>	<code>[3,1,2]</code>	Takes in a Comparator and uses that to sort elements in Iterable.
Flatten	<code>listOf (in tList, aSet).f la tten()</code>	<code>[2,3,4,1]</code>	Takes elements of all passed in collections and returns a collection with all those elements
FlatMap with just return	<code>listOf (in tList, aSet).f latMap { it }</code>	<code>[2,3,4,1]</code>	Used for Iterable of Iterables and Lambdas that return Iterables. Transforms elements and flattens them after transformation.



By **Xantier**
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 4 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopadd.com>

Transformers (cont)

FlatMap with transform	<pre>listOf (in tList, aSet).flatMap { iterable: Iterable< Int> -> iterable.map { it + 1 } }</pre>	[2,3,4,2]	FlatMap often used with monads to flatten nested collections to flatten nested collections to handle context and errors and side effects.
Zip	<pre>listOf(3, 4).zip (in tList)</pre>	[(3,1), (4,2)]	Creates list of Pairs from two Iterable. As many pairs as values shorter the original Iterable
Zip with predicate	<pre>listOf(3, 4).zip (in tList) { firstElem, secondElem -> Pair(firstElem - 2, secondElem + 2) }</pre>	[(1,3), (2,4)]	Creates list of Pairs from two Iterable. As many pairs as values shorter the original Iterable. Lambda receive both it on that index from Iterable
Unzip	<pre>listOf (Pair(" hi", 1), Pair("h ell o", 2)).un zip()</pre>	Pair([hi, hello], [1,2])	Reverses the operation from zip. Takes in an Iterable. Pairs are returned as Pair of Lists.

Aggregators

Method	Example	Result	Explanation
Folds And Reduces			
Fold	<pre>intList.fold(10) { accumulator, value -> accumulator + value }</pre>	16 $(10+1+2+3)$	Accumulates values starting with initial and applying operation from left to right. Lambda receives accumulated value and current value.
FoldIndexed	<pre>intList.foldIndexed(10) { idx, accumulator, value -> if (idx == 2) accumulator else accumulator + value }</pre>	13 $(10+1+2)$	Accumulates values starting with initial and applying operation from left to right. Lambda receives index as the first value.



By **Xantier**
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 5 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Aggregators (cont)

FoldRight	<pre>intList t.foldRight(10) { accumulator, value -> accumulator + value }</pre>	16 <i>(10+3+2+1)</i>	Accumulates values starting with initial and applying operation from right to left. Lambda receives accumulated value and current value.
FoldRight-Indexed	<pre>intList t.foldRightIndexed(10) { idx, accumulator, value -> if (idx == 2) accumulator else accumulator + value }</pre>	16 <i>(10+3+2+1)</i>	
Reduce	<pre>intList t.reduce { accumulator, value -> accumulator + value }</pre>	6 <i>(1+2+3)</i>	Accumulates values starting with first value and applying operation from left to right. Lambda receives accumulated value and current value.
Reduce-Right	<pre>intList t.reduceRight { accumulator, value -> accumulator + value }</pre>	6 <i>(3+2+1)</i>	Accumulates values starting with first value and applying operation from right to left. Lambda receives accumulated value and current value.
Reduce-Indexed	<pre>intList t.reduceIndexed { idx, accumulator, value -> if (idx == 2) accumulator else accumulator + value }</pre>	3 <i>(1+2)</i>	
Reduce-Right-Indexed	<pre>intList t.reduceRightIndexed { idx, accumulator, value -> if (idx == 2) accumulator else accumulator + value }</pre>	3 <i>(2+1)</i>	

Grouping



By Xantier
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 6 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Aggregators (cont)

```
groupBy      intList.t.groupBy { value -> 2 }
```

```
groupBy      intList.t.groupBy({ it }, { it + 1 })  
(With new  
values)
```

```
groupByTo    val mutableStringToListMap = mapOf( " first" to 1, " second " to 2)  
mutableStringToListMap.values.groupByTo(mutableMapOf<Int, MutableList<Int >>(), { value: Int -  
    })
```

```
GroupingBy  intList.groupingBy { it }
-> FoldTo    .foldTo(mutableMapOf<Int, Int>(), 0) { accumulator, element ->
                accumulator + element      }
```

```
Grouping >  intList.groupingBy { "key" }
Aggregate   .aggregate({ key, accumulator: String?, element, isFirst ->
                    when (accumulator) {
                        null -> "element"
                        else -> accumulator + "element"
                    }
                })
```

Aggregating



By **Xantier**
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 7 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish
Yours!
<https://apollopad.com>

Aggregators (cont)

Count	<code>intList.count()</code>	3	AKA size
Count (with Lambda)	<code>intList.count { it == 2 }</code>	1	Count of elements satisfying the predicate
Average	<code>intList.average()</code>	2.0 $((1+2+3)/3 = 2.0)$	Only for numeric Iterables
Max	<code>intList.max()</code>	3	Maximum value in the list. Only for Iterables of Comparables.
MaxBy	<code>intList.maxBy { it * 3 }</code>	3	Maximum value returned from lambda. Only for Lambdas returning Comparables.
MaxWith	<code>intList.maxWith (on eOr Larger)</code>	1	Maximum value defined by passed in Comparator
Min	<code>intList.min()</code>	1	Minimum value in the list. Only for Iterables of Comparables.
MinBy	<code>intList.minBy { it * 3 }</code>	1	Minimum value returned from lambda. Only for Lambdas returning Comparables.
MinWith	<code>intList.minWith (on eOr Larger)</code>	3	Minimum value defined by passed in Comparator
Sum	<code>intList.sum()</code>	6	Summation of all values in Iterable. Only numeric Iterables.
SumBy	<code>intList.sumBy { if(it == 3) 6 else it }</code>	9 $(1+2+6)$	Summation of values returned by passed in lambda. Only for lambdas returning numeric values.
SumByDouble	<code>intList.sumByDouble { it.toDouble() }</code>	6.0	Summation to Double values. Lambda receives the value and returns a Double.

```

val oneOrLarger = Comparator<Int> { x, y ->
    when{
        x == 1 -> 1
        y == 1 -> -1
        else -> y - x
    }
}

```



By Xantier
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 8 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Filtering and other predicates + simple HOFs

Method	Example	Result	Notes
Filtering			
Filter	<code>intLis t.f ilter { it > 2 }</code>	<code>[3]</code>	Filter-in
FilterKeys	<code>aMap.f ilt erKeys { it != " hel lo" }</code>	<code>{hi=1}</code>	
Filter-Values	<code>aMap.f ilt erV alues { it == 2 }</code>	<code>{hello=2}</code>	
Filter-Indexed	<code>intLis t.f ilt erI ndexed { idx, value -> idx == 2 value == 2 }</code>	<code>[2,3]</code>	
FilterIsInstance	<code>intLis t.f ilt erI sIn sta nce <St rin g>()</code>	<code>[]</code>	All of them are ints

Taking and Dropping

Take	<code>intLis t.t ake (2)</code>	<code>[1,2]</code>	Take n elements from Iterable. If passed in number larger than list, full list is returned.
TakeWhile	<code>intLis t.t ake While { it < 3 }</code>	<code>[1,2]</code>	
TakeLast	<code>intLis t.t ake Last(2)</code>	<code>[2,3]</code>	
TakeLastWhile	<code>intLis t.t ake Las tWhile { it < 3 }</code>	<code>[]</code>	Last element already satisfies this condition --> empty
Drop	<code>intLis t.d rop (2)</code>	<code>[3]</code>	Drop n elements from the start of the Iterable.
DropWhile	<code>intLis t.d rop While { it < 3 }</code>	<code>[3]</code>	
DropLast	<code>intLis t.d rop Last(2)</code>	<code>[1]</code>	
DropLastWhile	<code>intLis t.d rop Las tWhile { it > 2 }</code>	<code>[1, 2]</code>	

Retrieving individual elements

Component	<code>intLis t.c omp one nt1()</code>	<code>1</code>	There are 5 of these --> <code>comp on ent1(), compon ent2(), compon ent3(), compon ent4(), compon ent5()</code>
ElementAt	<code>intLis t.e lem ent At(2)</code>	<code>3</code>	Retrieve element at his index. Throws <code>IndexOutOfBoundsException</code> if element index doesn't exist
ElementAtOrElse	<code>intLis t.e lem ent AtO rEl se(13) { 4 }</code>	<code>4</code>	Retrieve element at his index or return lambda value if element index doesn't exist.



By **Xantier**
cheatography.com/xantier/

Published 14th June, 2017.
 Last updated 14th June, 2017.
 Page 9 of 15.

Sponsored by **ApolloPad.com**
 Everyone has a novel in them. Finish Yours!
<https://apollopapad.com>

Filtering and other predicates + simple HOFs (cont)

ElementAtOrNull	<code>intList.elementAtOrNull(66)</code>	66	Retrieve element at his index or return null if element index doesn't exist.
Get (clumsy syntax)	<code>intList.get(2)</code>	3	Get element by index
Get	<code>intList[2]</code>	3	Shorthand and preferred way for the one above
GetOrElse	<code>intList.getOrElse(14) { 42 }</code>	42	Get element or return lambda value if it doesn't exist.
Get from Map (clumsy syntax)	<code>aMap.get(" hi")</code>	1	
Get from Map	<code>aMap["h i"]</code>	1	
GetValue	<code>aMap.getValue("hi")</code>	1	Get value or throw NoSuchElementException
GetOrDefault	<code>aMap.getOrDefault(" HI", 4)</code>	4	Get value or return the value returned from lambda
GetOrPut	<code>mutableMap.getOrPut("H I") { 5 }</code>	5	MutableMap only. Returns the the value if it exist, otherwise puts it and returns put value.

Finding

BinarySearch	<code>intList.binarySearch(2)</code>	1	Does a binary search through the collection and returns the index of the element if found. Otherwise returns negative index.
Find	<code>intList.find { it > 1 }</code>	2	First element satisfying the condition or null if not found
FindLast	<code>intList.findLast { it > 1 }</code>	3	Last element satisfying the condition or null if not found
First	<code>intList.first()</code>	1	First element of Iterable or throws NoSuchElementException
First with predicate	<code>intList.first { it > 1 }</code>	2	Same as find but throws NoSuchElementException if not found
FirstOrNull	<code>intList.firstOrNull()</code>	1	Throw safe version of first().



By Xantier
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 10 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Filtering and other predicates + simple HOFs (cont)

FirstOrNull with predicate	<code>intList t.firstOrNull { it > 1 }</code>	2	Throw safe version of <code>first()</code> -> <code>Boolean</code> .
IndexOf	<code>intList t.indexOf(1)</code>	0	
IndexOfFirst	<code>intList t.indexOfFirst { it > 1 }</code>	1	
IndexOfLast	<code>intList t.indexOfLast { it > 1 }</code>	2	
Last	<code>intList t.last()</code>	3	Throws <code>NoSuchElementException</code> if empty Iterable
Last with predicate	<code>intList t.last { it > 1 }</code>	3	Throws <code>NoSuchElementException</code> if none found satisfying the condition.
LastIndexOf	<code>intList t.lastIndexOf(2)</code>	1	
LastOrNull	<code>intList t.lastOrNull()</code>	3	Throw safe version of <code>last()</code>
LastOrNull with predicate	<code>intList t.lastOrNull { it > 1 }</code>	3	Throw safe version of <code>last()</code> -> <code>Boolean</code> .

Unions, distincts, intersections etc.

Distinct	<code>intList t.distinct()</code>	[1, 2, 3]	
DistinctBy	<code>intList t.distinctBy { if (it > 1) it else 2 }</code>	[1, 3]	
Intersect	<code>intList t.intersection(listOf(1, 2))</code>	[1, 2]	
MinusElement	<code>intList t.minusElement(2)</code>	[1, 3]	
MinusElement with collection	<code>intList t.minusElement(listOf(1, 2))</code>	[3]	
Single	<code>listOf("One Element").single()</code>	One Element	Returns only element or throws.
SingleOrNull	<code>intList t.singleOrNull()</code>	null	Throw safe version of <code>single()</code> .
OrEmpty	<code>intList t.orEmpty()</code>	[1, 2, 3]	Returns itself or an empty list if itself is null.
Union	<code>intList t.union(listOf(4, 5, 6))</code>	[1, 2, 3, 4, 5, 6]	
Union (infix notation)	<code>intList union listOf(4, 5, 6)</code>	[1, 2, 3, 4, 5, 6]	



By [Xantier](#)
cheatography.com/xantier/

Published 14th June, 2017.
 Last updated 14th June, 2017.
 Page 11 of 15.

Sponsored by [ApolloPad.com](#)
 Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>

Checks and Actions

Method	Example	Result
Acting on list elements		
	<pre>val listOf Functions = listOf({ print(" first ") }, { print(" second ") })</pre>	
ForEach	<pre>listOf Functions.forEach { it() }</pre>	first
ForEachIndexed	<pre>listOf Functions.forEachIndexed { idx, fn -> if (idx == 0) fn() else print(" Won't do it") }</pre>	first
OnEach	<pre>intList.forEach { print(it) }</pre>	123
Checks		
All	<pre>intList.all { it < 4 }</pre>	true
Any	<pre>intList.any()</pre>	true
Any with predicate	<pre>intList.any { it > 4 }</pre>	false
Contains	<pre>intList.contains(3)</pre>	true
ContainsAll	<pre>intList.containsAll(listOf(2, 3, 4))</pre>	false
Contains (Map)	<pre>aMap.containsKey("Hello")</pre>	false
ContainsKey	<pre>aMap.containsKey("Hello")</pre>	true
ContainsValue	<pre>aMap.containsValue(2)</pre>	true
None	<pre>intList.none()</pre>	false
None with predicate	<pre>intList.none { it > 5 }</pre>	true
IsEmpty	<pre>intList.isEmpty()</pre>	false
IsNotEmpty	<pre>intList.isNotEmpty()</pre>	true

<3 Kotlin

Github repository with all code examples:

<https://github.com/Xantier/Kollections>

PDF of this cheat sheet:

<http://jussi.hallila.com/Kollections/>

Created with <3 by Jussi Hallila



By Xantier
cheatography.com/xantier/

Published 14th June, 2017.
Last updated 14th June, 2017.
Page 12 of 15.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
<https://apollopadd.com>