Groovy Collections Cheat Sheet

Lists

1. **get(integer) / getAt(integer) –** Get element at i^{th} position
2. **getAt(range) / subList(start, end) –** Get a list of elements withing a range
3. **first()** - Get first element of list
4. **last() -** Get last element of list
5. **head()** - Get the head of list
6. **tail()** - Get all elements of a list except the head
7. **next()** - Get the next element given the current index
8. **unique() -** Remove all non-unique elements
9. **size()** - Returns size of list
10. **add(element) / add(collection)** – Adds element(s) to a list
11. **plus(index, collection)** – Adds a list of objects at specified location
12. **contains(value)** – Checks whether a list contains a specified value or not
13. **remove(idx / element) –** Remove element at index specified by "idx" or the specified element
14. **pop() -** Pop the last element of a list
15. **clear()** - Remove all elements of a list
16. **drop(num)** – Drop "num" number of elements from start of list if available
17. **flatten()** - Flattens a nested list
18. **removeAll(list)** – Remove all occurrences of elements in list
19. **tokenize(delimiter)** – Split string into a list with argument used as delimiter; uses white space as delimiter by default
20. **split(delimiter) –** Same as tokenize but can also accept regular expressions as delimiters; default delimiter is white space
21. **join(adhesive) –** Joins elements of a list using "adhesive" and returns a string
22. **collect(closure) –** Returns list of elements returned by the closure
23. **find(closure) –** Returns the first element matching the closure criteria
24. **findAll(closure)** – Returns all elements matching the closure criteria
25. **each(closure) –** Loops over a list executing the closure for each element
26. **eachWithIndex(closure) –** Loops over a list executing the closure for each element; also provides the index for each element in addition
27. **reverseEach(closure)** – Loops over a list in reverse order, executing the closure for each element
28. **sum(<closure>)** - Sum over all the elements of a list (in case of a simple list) or over the element specified in the closure (in case of a complex object)
29. **max(<closure>)** - Return the maximum of all elements of a list (in case of a simple list) or the maximum of the element specified in the closure (in case of a complex object)

30. **min(<closure>) -** Return the maximum of all elements of a list (in case of a simple list) or the maximum of the element specified in the closure (in case of a complex object)
31. **sort(<closure>)** - Sort the list in ascending order (in case of a simple list) or sort the list on the basis of a property specified in the closure(in case of complex objects)
32. **reverse(<boolean>)** - Reverse the list; does not alter original list unless specified

Maps

1. **put(key, value) –** Insert key-value pair in a map
2. **putAll(map) –** Insert map into another map
3. **get(key) –** Get value associated with a given key
4. **remove(key) –** Remove key value pair associated with a particular key
5. **containsKey(key) –** Checks if map contains a particular key
6. **keySet()** - Returns a set of keys in a map
7. **containsValue(value) –** Checks if map contains a particular value
8. **values() -** Returns set of values in map
9. **every(closure)** - Iterates over the entries of a map, and checks whether a predicate is valid for all entries.
10. **any(closure)** - Iterates over the entries of a map, and checks whether a predicate is valid for some.
11. **find(closure)** – Find first element of occurrence being searched
12. **findAll(closure)** – Find all occurences of element being searched
13. **each(closure)** – Iterates over the entries of a map executing the closure for each element
14. **eachWithIndex(closure)** – Iterates over each entry of a map allowing entries to be accessed via an index

Ranges

1. **from –** Get the first element of the range
2. **to** – Get the last element of the range
3. **contains(value)** – Checks if a value lies within a range
4. **isReverse()** - Checks if the range is reversed