Cheatography

Java Collections (OCA) Cheat Sheet

by Jianmin Feng (taotao) via cheatography.com/79308/cs/19511/

What's collection

a framework/architecture(a set of classes /interface) to store and manipulation group(-single unit) of objcts
sorting, searching, insert, delete, iterate etc.
many interfaces: List, Set, Queue,Dequeue
many classes: ArrayList,Vector, LinkedList,PriorityQueue,HashSet,TreeSet etc

Collection framework hierarchy

iterable --> collection -->List,Queue/Deque,Set/SortedSet
list->ArrayList,LinkedList,Vector <-Sack
Queue ->PriorityQueue
Deque ->ArrayDeque,LinkedList
SortedSet->TreeSet
Set->HashSet,LinkedHashSet

Collection Methods

public boolean add(E e)	append an item
public boolean addAll(Collection<? extends E> c)	addAll
public boolean remove(Object element)	remove 1
public boolean removeAll(Collection<?> c)	removeAll
default boolean removeIf(Predicate<? super E> filter)	removeIf

Collection Methods (cont)

public boolean retainAll(Collection<?> c)	retainAll
public int size()	size()
public void clear()	clear
public boolean isEmpty()	isEmpty
public boolean contains(Object element)	contains
public boolean containsAll(Collection<?> c)	containsAll
public Iterator iterator()	iterator
public Object[] toArray()	toArray
public <T> T[] toArray(T[] a)	toArray type
public boolean equals(Object element)	equals
public int hashCode()	hashcode
default Stream<E> parallelStream()	
default Stream<E> stream()	
default Spliterator<E> spliterator()	

Iterator interface

public boolean hasNext()
public Object next()
public void remove()
enumeration hasMoreElement(), nextElement(), but no remove()

Iterable interface

top of collection	
Only one method:	
Iterator<T> iterator()	return the iterator over the items of type T
4 way to iterate	
1. iterator	hasNext(), next()
2. for loop	size()
3. for each loop	
4.lambda expression forEach()	list.forEach(name->name.charAt(0)='h')
mapAscii.forEach((key, value)	can be used to iterate map

List Interface -Index plays important role

Duplicable	
ArrayList	random access, add/remove expensive(shift),not ordered
LinkedList	sequence access,add/remove cheap(no shift), ordered
Vector	like ArrayList,but synchronized,more methods
Stack	extends Vector, LIFO, more methods
	boolean push(),boolean peek(),boolean push(obj)

Cheatography

Java Collections (OCA) Cheat Sheet
by Jianmin Feng (taotao) via cheatography.com/79308/cs/19511/

Queue interface

FIFO	first in first out
Ordered list of item to be processed	
PriorityQueue	no null item, ordered by priority
Deque	interface, doubled ended queue
ArrayDeque	add/remove from both end, faster than ArrayList and Stack

Set

unordered	no duplicate, at most one null
Hashset	
LinkedListHashSet	maintain insertion order, permit nulls
SortedSet interface	sorted ascending/descending/natual ordering
TreeSet	ascending order, faster access

Java Collections

java.util.Collections	Static methods
max()	min()
sort()	shuffle()
binarySearch()	copy()
reverse()	synchronizedCollection()
disjoin(): split into 3 collection w/o commons	

Comparable and Comparator interfaces

Comparator	equals(), Compare()
Comparable	compareTo()

Java Map

key value pairs	not iterable
NoSuchElementException	ClassCastException
NullPointerException	UnsupportedOperationException
Object put(Object k, Object v)	add
void putAll(Map m)	addAll
Object remove(Object k)	remvoe
Object get(Object k)	get
boolean containsKey(Object k)	ContainsKey
boolean containsValue(Object v)	containsValue
Set entrySet( )	value->set
Set keySet( )	key->set
Collection values( )	value->collection
int size( )	size
void clear( )	clear
boolean isEmpty( )	isEmpty
boolean equals(Object obj)	equals
int hashCode( )	hashcode

iterate on map

No iterator	
1 for each loop	for (Map.Entry<String,String> e:myMap.entrySet()){}
	for (String k:myMap.keySet()){}
	for (String v:myMap.value()){}
2 indrect iterator	Oterator<Map.Entry<String,String>> itr=myMao.entrySet().iterator()
3 stand for loop	size()
4 forEach(lambdas)	myMap.foreach((k,v)->...)
5 iterator on key	set value myMap.get(key)
not efficient, not practical	

HashMap,Treemap and Hashable

HashMap:	unique key, dup values;allow null values and null keys
TreeMap	ordered object
HashTable	synchonized, no nulls