

Math Library

Math is a final class in java	Which means it cannot be inherited
Math.abs(-1)	1 //returns the absolute value
Math.ceil(-1.9)	returns: -1.0 //Returns the smallest integer that is greater than or equal to the specified number.
Math.floor(-1.9)	returns: -2.0 // Returns the largest integer that is less than or equal to the specified number.
Math.max(1, 2)	Returns the larger of the two specified numbers.
Math.min(1, 2)	returns: 1
Math.pow(2, 3)	returns: 8.0, Returns the value of the first argument raised to the power of the second argument.
Math.random()	returns: 0.0 <= x < 1.0, Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
Math.round(1.1)	returns: 1, Returns the closest long or int, as indicated by the method's return type, to the argument
Math.round(1.5)	returns: 2,
Math.sqrt(4)	returns: 2.0, Returns the correctly rounded positive square root of a double value.

Treeset

Set<Integer> treeSet = new TreeSet<Integer>();	Set interface does not have ceiling() method
TreeSet<Integer> treeSet = new TreeSet<Integer>();	if we use Set interface rather than TreeSet class, we cannot use ceiling() method
treeSet.add(10);	adds the element to the treeset
treeSet2.addAll(treeSet);	
System.out.println("treeSet2 after adding elements" + treeSet2);	print treeSet
treeSet.remove(20);	removes the specific element from the treeset
for (Integer item : treeSet) { System.out.println(item); }	Iterating over tree set items
Iterator<Integer> iterator = treeSet.iterator();	Iterating using iterator
Iterator<Integer> descendingIterator = treeSet.descendingIterator();	returns an iterator over the elements in this set in descending order.
int value1 = treeSet.ceiling(25);	returns the least element in this set greater than or equal to the given element
int value2 = treeSet.floor(25);	returns the greatest element in this set less than or equal to the given element



By [shravya_kavali2](#)
cheatography.com/shravya-kavali2/

Not published yet.
 Last updated 6th August, 2023.
 Page 1 of 5.

Sponsored by [CrosswordCheats.com](#)
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>

Treeset (cont)

<code>Set<Integer> descendingSet = treeSet.descendingSet();</code>	returns a reverse order view of the elements contained in this set.
<code>Set<Integer> headSet = treeSet.headSet(30);</code>	returns a view of the portion of this set whose elements are strictly less than toElement.
<code>Set<Integer> headSet2 = treeSet.headSet(30, true);</code>	returns a view of the portion of this set whose elements are less than or equal to toElement.
<code>Set<Integer> tailSet = treeSet.tailSet(30);</code>	returns a view of the portion of this set whose elements are greater than or equal to fromElement.
<code>Set<Integer> tailSet2 = treeSet.tailSet(30, false);</code>	returns a view of the portion of this set whose elements are strictly greater than fromElement.
<code>Set<Integer> subSet = treeSet.subSet(20, 40);</code>	returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive.
<code>Set<Integer> subSet2 = treeSet.subSet(20, true, 40, true);</code>	returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, inclusive.
<code>Integer pollFirst = treeSet.pollFirst();</code>	retrieves and removes the first (lowest) element, or returns null if this set is empty.
<code>Integer pollLast = treeSet.pollLast();</code>	retrieves and removes the last (highest) element, or returns null if this set is empty.
<code>boolean remove = treeSet.remove(20);</code>	returns true if this set contained the specified element
<code>boolean isEmpty = treeSet.isEmpty();</code>	returns true if this set contains no elements.
<code>int size = treeSet.size();</code>	returns the number of elements in this set (its cardinality).
<code>boolean contains = treeSet.contains(20);</code>	returns true if this set contains the specified element.
<code>treeSet.clear();</code>	

TreeSet is implemented using a tree structure (red-black tree in algorithm book)

TreeSet is ordered, sorted, and navigable

TreeSet does not allow duplicate elements

TreeSet allows null elements

Primitive Data types

byte	Size: 1 byte	Range: -128 to 127
short	Size: 2 bytes	Range: -32,768 to 32,767
int	Size: 4 bytes	Range: -2,147,483,648 to 2,147,483,647
long	Size: 8 bytes	Range: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Size: 4 bytes	Range: 3.4e-038 to 3.4e+038
double	Size: 8 bytes	Range: 1.7e-308 to 1.7e+308
boolean	Size: 1 bit	Range: true or false
char	Size: 2 bytes	Range: 0 to 65,536

Operators

a * b	Multiplication
a + b	
a - b	
a / b	Division
a % b	Modulo
a++	Postincrement
++a	Preincrement
b--	Postdecrement
--b	Predecrement
a operator= 3	+, -, *, /, %, <<, >>, >>>
a operator 0b1010	&, , ^
a relationalOperator b	==, !=, <, <=, >, >=
a logicalOperator b	&&, ,
!x	not operator
TODO: ternary	
TODO: shift operator	

loops

```
for (int number : numbers) { System.out.println("Element: " + number); }  
for (int i = 0; i < numbers.size(); i++) { int element = numbers.get(i); System.out.println("Element at index " + i + ": " + element); }
```



Set

The set interface is present in java.util package	which is extended from collection interface
Interfaces extended from Set interface:	SortedSet, NavigableSet
Classes that implements Set Interface	HashSet, EnumSet, LinkedHashSet, TreeSet
SortedSet Interface	contains the methods inherited from the Set interface and adds a feature that stores all the elements in this interface to be stored in a sorted manner
Navigable Interface:	provides the implementation to navigate through the Set.
Treeset	implementation of a self-balancing tree
<p>set is implemented using hash table</p> <p>set is unordered</p> <p>set does not allow duplicate elements</p> <p>set allows null elements</p>	

LinkedHashSet

create a linkedHashSet of characters	<code>Set<Character> linkedHashSet = new LinkedHashSet<Character>();</code>
add element	<code>linkedHashSet.add('a');</code>
remove the element	<code>linkedHashSet.remove('a');</code>
check if the element is present	<code>linkedHashSet.contains('a')</code>
size	<code>linkedHashSet.size()</code>
is empty	<code>linkedHashSet.isEmpty()</code>
Iterator	<code>Iterator iterator = linkedHashSet.iterator();</code>
to Array	<code>linkedHashSet.toArray()</code>
to string	<code>linkedHashSet.toString()</code>
equals	<code>linkedHashSet.equals(linkedHashSet)</code>
hashCode()	<code>linkedHashSet.hashCode()</code>

linkedHashSet is a subclass of HashSet
 linkedHashSet maintains insertion order
 linkedHashSet is slower than HashSet
 linkedHashSet is faster than TreeSet
 when to use: when you want to maintain insertion order
 when not to use: when you want to sort the elements

HashSet

Creating the hashSet	<code>Set<String> hashSet = new HashSet<String>();</code>
Adding element	<code>hashSet.add("element1");</code>
Print hashSet	<code>System.out.println("hashset after adding elements" + hashSet);</code>
remove element	<code>hashSet.remove("element2");</code>



HashSet (cont)

Iterating over hash set items	<code>for (String item : hashSet) {System.out.println(item);}</code>
// Iterating using iterator	<code>Iterator<String> iterator = hashSet.iterator();</code>
checking if hashSet contains an element	<code>hashSet.contains("element1")</code>
checking if hashSet is empty	<code>hashSet.isEmpty()</code>
checking size of hashSet	<code>hashSet.size()</code>
clearing hashSet	<code>hashSet.clear();</code>

Hashset is implemented using hash table
 Hashset is unordered
 Hashset does not allow duplicate elements
 Hashset allows null elements

Enum hashset

create enumSet	<code>EnumSet<Days> hashSet = EnumSet.of(Days.MONDAY, Days.TUESDAY, Days.WEDNESDAY, Days.THURSDAY, Days.FRIDAY);</code>
[MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY]	<code>EnumSet<Days> workingDays = EnumSet.range(Days.MONDAY, Days.FRIDAY);</code>
[SATURDAY, SUNDAY]	<code>EnumSet<Days> weekEnds = EnumSet.complementOf(workingDays);</code>
[MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY]	<code>EnumSet<Days> allDays = EnumSet.allOf(Days.class);</code>
[]	<code>EnumSet<Days> noDays = EnumSet.noneOf(Days.class);</code>
add	<code>noDays.add(Days.MONDAY);</code>
remove	<code>noDays.remove(Days.MONDAY);</code>
check if element is present	<code>noDays.contains(Days.MONDAY)</code>
check if enum hashset are equal	<code>weekDays.equals(noDays)</code>
iterator	<code>Iterator iterator = noDays.iterator();</code>
get the size	<code>noDays.size()</code>
	<code>noDays.toArray()</code>
	<code>noDays.clear();</code>

when to use: when you want to use a set of enum constants
 when not to use: when you want to use a set of non-enum constants



By [shravya_kavali2](#)
cheatography.com/shravya-kavali2/

Not published yet.
 Last updated 6th August, 2023.
 Page 5 of 5.

Sponsored by [CrosswordCheats.com](#)
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>