String

Declaration

```
String s = "Hello world";
String s = new String("Hello world!");
String s = new String(charArray);
```

Methods

```
char c = s.charAt(index);
int index = s.indexOf("world");
String subs = s.substring(startIdx, endIdx);
bool hasRed = s.contains("red");
String newStr = s.replace('a', 'b');
String lowerCaseStr = s.toLowerCase();
String upperCaseStr = s.toUpperCase();
```

Convert

```
String s = String.valueOf(number);
String[ ] arrayOfStr = s.split(",");
char[ ] charArray = s.toCharArray();
int number = Integer.parseInt("123");
```

ArrayList

Declaration

```
ArrayList<String> cars = new ArrayList<>();
ArrayList<Integer> nums= new ArrayList<>(Arrays.asList(1, 2));
```

Methods

```
cars.add("Volvo");
cars.get(0);
cars.set(0, "Opel");
cars.remove(0);
cars.clear(); //remove all
cars.size();
int firstIndex = cars.indexOf("Volvo");
```

Sort & Iterate

```
Collections.sort(cars);
Collections.reverse(cars);
for ( int i = 0; i < cars.size(); i++ ) { }
for ( String car : cars ) { }
```

ArrayList (cont)

```
Iterator it = cars.iterator();
while ( it.hasNext() ) { it.next(); };
```

Convert to array

```
Integer[] arr = new Integer[arrayList.size()];
arr = arrayList.toArray(arr);
```

Array

Declaration

```
int[ ] array= new int[]{1, 2, 3, 4, 5, 6};
int[ ] array = new int[10];
int[ ][ ] twoDArray = new int[n][m];
```

Methods

```
int n = array.length;
```

Sort & Iterate

```
Arrays.sort(array);
Arrays.sort(array, (a,b) -> Integer.compare(a,b))
```

Character

Declaration & Methods

```
char myGrade = 'B';
(char) (1 + 'a') -> 'b'
```

Convert to other types

```
int a = ch - '0';
char myVar1 = 65; // myVar1 = 'A'

Character c1 = new Character('r');
String s1 = c1.toString();
String s2 = String.valueOf(c1);
```

LinkedList

Declaration

```
LinkedList<String> cars = new LinkedList<String>();
```

Method

```
cars.add("Volvo");
cars.addFirst("Mazda");
cars.addLast("Ford");
cars.removeFirst();
cars.removeLast();
```

By **nafanya**
cheatography.com/nafanya/

Published 20th April, 2024.
Last updated 8th March, 2025.

LinkedList (cont)

```
cars.getFirst();
cars.getLast();
cars.size();
```

Iterate

```
for ( int i = 0; i < cars.size(); i++ ) { }
for ( String car: cars ) { }
```

HashMap

Declaration

```
HashMap<String, Integer> map = new HashMap<>();
```

Methods

```
map.put("key1", 10);
map.containsKey("key2")
map.containsValue(2));
map.get("key1");
map.getOrDefault("key1", 0);
map.remove("key1");
map.clear(); //remove all items
map.put("key1", 5); //update key1
map.size();
```

Iterate

```
for ( String key : map.keySet() ) { }
for ( Integer i : map.values() ) { }
```

Queue (LinkedList implementation)

Declaration

```
Queue<String> queue = new LinkedList<>();
```

Methods

```
queue.add("apple");
queue.remove(); // remove first element
queue.peek(); //show first element
queue.size();
```

LinkedList vs PriorityQueue

LinkedList preserves the insertion order, PriorityQueue does not. The elements of the priority queue are ordered according to their natural ordering, or by a Comparator provided at queue construction time.

Stack

Declaration

```
Stack<Integer> stack = new Stack<Integer>();
stack.push(1);
stack.peek();
stack.pop();
stack.isEmpty()
int index = stack.search(2);
```

Iterate

```
for ( Integer item : stack ) { }
Iterator it = stack.iterator();
while ( it.hasNext() ) { it.next(); };
```

HashSet

Declaration

```
HashSet<String> cars = new HashSet<String>();
```

Methods

```
cars.add("Volvo");
cars.contains("Mazda");
cars.remove("Volvo");
cars.clear();
cars.size();
```

Iterate & Convert

```
for ( String car : cars ) { }
Iterator<Integer> it = cars.iterator();
while ( it.hasNext() ) { it.next();};
String[] array = cars.toArray(
                    new String[cars.size()])
```

Queue (PriorityQueue implementation)

Declaration

```
Queue<Integer> pQueue = new PriorityQueue<Integer>();
```

Methods

```
pQueue.add(10);
pQueue.peek(); //return top element
pQueue.poll(); //return and remove
pQueue.remove(2); //remove first 2
```

Iterate

```
Iterator it = pQueue.iterator();
while ( it.hasNext() ) { it.next(); };
```

Comparartor

Comparator examples

Bitwise Operators

OR	0101 | 0111 = 0111
AND	0101 & 0111 = 0101
XOR	0101 ^ 0111 = 0010
NOT	~ 0101 = 1010
Left shift	000110 << 1 = 001100
Right shift	10110101 >> 1 = 11011010
Unsigned Right Shift	10110101 >>> 1 = 01011010

Other

Min integer	Integer.MIN_VALUE
Max integer	Integer.MAX_VALUE