java lang&api cheetsheet

Number

```
Integer.MAX_VALUE
Integer.MIN_VALUE
Integer valueOf(int i)
Integer valueOf(String s)
Integer valueOf(String s, int radix)
String toString()
String toString(int i)
int parseInt(String s)
int parseInt(String s, int radix)
```

Math

```
Math.abs()
Math.ceil(Double/Float)
Math.floor(Double/Float)
long round(double d)
int round(float f)
double exp(double d)
double log(double d)
double pow(double base, double exponent)
Double E, PI
```

拷贝ArrayList

```
ArrayList<Integer> b = new ArrayList<>(a);
ArrayList<Integer> b =(ArrayList<Integer>) a.clone();
```

Array

```
int data[] = new int[3];
data[0] = 0;
data[1] = 1;
data[2] = 2;
int data[] = new int[3] {1, 2, 3};
for(int i = 0; i < data.length; i++) {...};
int data[][] = new int[3][3] {
  {1,2,3},{4,5,6},{7,8,9}
};
// System.arraycopy(源数组名称,源数组复制开始索引,目标数组名称,目标数组赋值开始索引,
长度), 下面的语句就是从dataA数组的第4位(包含第4位)的3位数复制到dataB的第2位
```

```
System.arraycopy(dataA, 4, dataB, 2, 3);
Arrays.sort(data);
```

String

```
String s = "Hello.";
String s = new("Hello.");
String s =  new String(new char[2]{'a', 'b'} ) //char数组转String
char c = s.chatAt(0);
char[] arr = s.toCharArray();
boolean b = s.euqals(s);
boolean b = s.euqalsIgnoreCase(s);
int i = s.compareTo(s);
boolean b = s.contains(s);
int i = s.indexOf(s); //s:String
int i = s.indexOf(s, 1); //s:String 从字符串第1位开始
int i = s.lastIndexOf(s); //s:String
boolean b = s.starsWith(s);
boolean b = s.starsWith(s, 1);
boolean b = s.endsWith(s);
Sting a = s.replaceAll(s);
Sting a = s.replaceFirst(s);
String s = s.substring(1);
String s = s.substring(1, 2);
String[] sr = s.split(String regex);
String s = s.concat(s); //+
String s = s.toLowerCase();
String s = s.toUpperCase();
String s = s.trim();
int len = s.length();
boolean b = s.isEmpty(); //验证是否为""而不是null
boolean matches(String regex)
```

List

```
List<String> l = new ArrayList<>();
l.add("1");
int len = l.size();
String s = l.get(0);
boolean b = l.contains("1");
l.remove("1");
l.clear(); //清空集合,根元素为null
boolean b = l.isEmpty();
String arr[] = l.toArray();
l.add(0, tmp);//将tmp元素加入到列表的第0位
List<List<Integer>> ans = new ArrayList<>();

LinkedList<String> list = new LinkedList<>();
list.add("hello");
```

```
Iterator<String> iter = list.iterator();
while(iter.hasNext())
System.out.println(iter.next)
```

Regex

```
String pattern = "(.*(\\d+))";
Pattern r = Pattern.compile(pattern);
String line = "lalala";
Matcher m = r.matches(line);
if(m.find()) line = line.replaceAll("-","");
```

Set

```
// TreeSet, HashSet
Set<Integer> s = new HashSet<>();
s.add(1);
boolean b = s.contains(1);
```

Map

```
// HashMap, TreeMap
Map<Integer, Integer> m = new HashMap<>();
m.put(1, 2);
Integer n = m.get(1);
Set<Map.Entry<Integer, Integer>> s = map.entrySet();
for(Integer i: map.keySet()) {...}
Integer k = Map.Entry<Integer, Integer>.getKey();
Integer k = Map.Entry<Integer, Integer>.getValue();
for(Map.Entry e: map.entrySet()) {
  Value v = e.getValue();
  Key k = e.getKey();
}
Map<String, String> map = new LinkedHashMap<String, String>
(16,0.75f,true); // LinkedHashMap可以保证Map中保留插入元素的顺序
```

Stack

```
Stack<Integer> stack = new Stack<>();
stack.push(1);
stack.push(2);
stack.peek();
stack.pop();
stack.search(1) // 返回的是栈中距离栈顶最近的目标数与栈顶的距离 top:[1, 2, 3] 此时
search(3)则返回3
```

```
Deque<Integer> stack = new ArrayDeque<Integer>();
```

Queue

```
Queue<E> queue = new LinkedList<E>();
queue.offer(1); //入队
queue.peek();
queue.poll(); //出队

// 双向队列
Deque<Integer> deque = new LinkedList<>();
deque.offer(1);
deque.offer(2);
deque.offerFirst(3); // [3, 1, 2]
deque.pollLast(); // [3, 1, 2]

// 优先队列
PriorityQueue<Integer> queue = new PriorityQueue<>(new Comparator<Integer>
(){
  @Override
  public int compare(Integer o1, Integer o2) {
    return o1.compareTo(o2);
  }
});
queue.add(1);
queue.poll();
queue.peek();
boolean queue.contains()
queue.size()
Object[] queue.toArray()
Comparator<Employee> nameSorter = Comparator.comparing(Employee::getName);
PriorityQueue<Employee> priorityQueue = new PriorityQueue<>( nameSorter );
```

StringBuilder和StringBuffer

```
StringBuilder append(char c)
StringBuilder deleteCharAt(int index)
int length()
Collections.sort(arrayList)
```

Read with Scanner

```
Scanner sc = new Scanner(System.in);
while(sc.hasNextLine()) {}
System.out.println(sc.nextLine());
```

Read with BufferedReader

```
String line = null;
try {
  BufferedReader br = new BufferedReader("filelocation");
  while((line = br.readLine()) != null) {
    System.out.println(line);
  }
} catch(Exception e) {
  e.printStackTrace();
}
```

Arrays类

```
boolean b = Arrays.equals(int[] a, int[] b);
Arrays.fill(int[] a, int val);
Arrays.sort(int[] a);
Arrays.binarySearch(int[] a, int key);
Arrays.toString(int[] a);
Arrays.sort(B, Comparator.comparingInt(Math::abs));
Arrays.sort(books, (o1, o2) -> {
  if(o1.getPrice() > o2.getPrice()) {
    return 1;
  } else if(...) {
    ...;
  } else {
    ...;
  }
});
```

Collections类

```
List<String> all = new ArrayList<>();
Collecions.addAll(all, "hello", "world");
Collections.reverse(all);
Collections.copy(dest, src);
```

Tricks

- 在常用char的情况下我们可以使用int[]来作为map而不需要创建HashMap