

Java 常用类

目录

1 标准输入.....	2
2 标准输出.....	2
3 Arrays&对象数组.....	3
3.1 Arrays.fill().....	3
3.2 Arrays.sort().....	3
3.3 Arrays.copyOf()&Arrays.copyOfRange().....	4
3.4 对象数组.....	4
4 String.....	5
4.1 int length().....	5
4.2 char charAt(int index).....	5
4.3 查找字串.....	5
4.4 boolean equals(str).....	5
4.5 char[] toCharArray().....	6
4.6 String replace(oldchar,newchar).....	6
4.7 String toUpperCase() & String toLowerCase().....	6
4.8 String substring(begin) & String substring(begin,end).....	6
5 大数处理.....	6
5.1 大数输入.....	6
5.2 大数输出.....	7
5.3 大数的加减乘除求余等计算.....	7
6 日期时间.....	9
6.1 计算日期差.....	9
6.2 计算时间差.....	9

1 标准输入

使用 Scanner 类：

(1) 使用 java.util 包。

```
1. import java.util.*;
```

(2) 构造 Scanner 类对象，它隶属于标准输入流 System.in。

```
1. Scanner s = new Scanner(System.in);
```

(3) 常用的 next()方法系列：

nextInt():输入整数

nextDouble():输入双精度数

nextBigInteger():输入大整形

nextBigDecimal():输入大浮点型

next():输入字符串（以空格作为分隔符）

nextLine():输入字符串

(4) hasNext()判断输入是否结束

2 标准输出

(1) System.out.println(); 是最常用的输出语句，它会把括号里的内容转换成字符串输出到输出窗口（控制台），并且换行，当输出的是一个基本数据类型时，会自动转换成字符串，如果输出的是一个对象，会自动调用对象的 toString();方法，将返回值输出到控制台

(2) System.out.print(); 与第一个很相似，区别就是上一个输出后会换行，而这个命令输出后并不换行。

(3) System.out.printf(); 这个方法延续了 C 语言的输出方式，通过格式化文本和参数列表输出。

```
1. System.out.println(1111); //换行打印
2. System.out.print(1111); //不换行打印
3. System.out.write(2222); //字节输出
4. System.out.printf("%+8.3f\n", 3.14); //按格式输出
```

3 Arrays&对象数组

Arrays 类位于 java.util 包中，主要包含了操纵数组的各种方法。（注意，与 Array 类不同，是两个完全不同的类）

3.1 Arrays.fill()

Arrays.fill(Object[] array, Object obj)

用指定元素填充整个数组（会替换掉数组中原来的元素），其具体实现其实就是循环赋值，并不能提高运算效率

```
1. Integer[] data = {1, 2, 3, 4};
2. Arrays.fill(data, 9);
3. System.out.println(Arrays.toString(data)); // [9, 9, 9, 9]
```

Arrays.fill(Object[] array, int fromIndex, int toIndex, Object obj)

用指定元素填充数组，从起始位置到结束位置，取头不取尾（会替换掉数组中原来的元素）

```
1. Integer[] data = {1, 2, 3, 4};
2. Arrays.fill(data, 0, 2, 9);
3. System.out.println(Arrays.toString(data)); // [9, 9, 3, 4]
```

3.2 Arrays.sort()

Arrays.sort(Object[] array)

对数组元素进行排序（串行排序）

```
1. String[] data = {"1", "4", "3", "2"};
2. System.out.println(Arrays.toString(data)); // [1, 4, 3, 2]
3. Arrays.sort(data);
4. System.out.println(Arrays.toString(data)); // [1, 2, 3, 4]
```

Arrays.sort(Object[] array, int fromIndex, int toIndex)

对数组元素的指定范围进行排序（串行排序）

```
1. String[] data = {"1", "4", "3", "2"};
2. System.out.println(Arrays.toString(data)); // [1, 4, 3, 2]
3. // 对下标[0, 3)的元素进行排序，即对 1, 4, 3 进行排序，2 保持不变
4. Arrays.sort(data, 0, 3);
5. System.out.println(Arrays.toString(data)); // [1, 3, 4, 2]
```

Arrays.sort(T[] array, Comparator<? super T> comparator)

使用自定义比较器，对数组元素进行排序（串行排序）

Arrays.sort(T[] array, int fromIndex, int toIndex, Comparator<? super T> c)

使用自定义比较器，对数组元素的指定范围进行排序（串行排序）

比较器例子：

```
1. class myComparator implements Comparator<Integer> {
2.     @Override
3.     public int compare(Integer o1,Integer o2) {
4.         return o1-o2; //将数组按数字从小到大排序
5.     }
```

3.3 Arrays.copyOf()&Arrays.copyOfRange()

Arrays.copyOf(T[] original, int newLength)

拷贝数组，其内部调用了 System.arraycopy() 方法，从下标 0 开始，如果超过原数组长度，会用 null 进行填充

```
1. Integer[] data1 = {1, 2, 3, 4};
2. Integer[] data2 = Arrays.copyOf(data1, 2);
3. System.out.println(Arrays.toString(data2)); // [1, 2]
4. Integer[] data2 = Arrays.copyOf(data1, 5);
5. System.out.println(Arrays.toString(data2)); // [1, 2, 3, 4, null]
```

Arrays.copyOfRange(T[] original, int from, int to)

拷贝数组，指定起始位置和结束位置，如果超过原数组长度，会用 null 进行填充

```
1. Integer[] data1 = {1, 2, 3, 4};
2. Integer[] data2 = Arrays.copyOfRange(data1, 0, 2);
3. System.out.println(Arrays.toString(data2)); // [1, 2]
4. Integer[] data2 = Arrays.copyOfRange(data1, 0, 5);
5. System.out.println(Arrays.toString(data2)); // [1, 2, 3, 4, null]
```

3.4 对象数组

对象数组创建出来只是引用，每一位索引还要单独再实例化

```
1. class Test{
2.     int i;
```

```
3. }
4.
5. public static void main(String[] args){
6.     Test[] test=new Test[10];
7.     for(int i=0;i<10;i++){
8.         test[i]=new Test();
9.     }
10. }
```

4 String

4.1 int length()

获取字符串长度

4.2 char charAt(int index)

获取 index 位置上的字符

4.3 查找字符串

1. `int indexOf(int ch)`: 返回的是 `ch` 在字符串中第一次出现的位置。
2. `int indexOf(int ch, int fromIndex)`: 从 `fromIndex` 指定位置开始, 获取 `ch` 在字符串中出现的位置。
- 3.
4. `int indexOf(String str)`: 返回的是 `str` 在字符串中第一次出现的位置。
5. `int indexOf(String str, int fromIndex)`: 从 `fromIndex` 指定位置开始, 获取 `str` 在字符串中出现的位置。
- 6.
7. `int lastIndexOf(String str)`: 反向索引。
- 8.
9. `boolean contains(str)`: 字符串中是否包含某一个子串。

4.4 boolean equals(str)

判断两个字符串是否相等

4.5 char[] toCharArray()

将字符串转换为字符数组

4.6 String replace(oldchar,newchar)

替换字符串中的字符串

4.7 String toUpperCase() & String toLowerCase()

大小写转换

4.8 String substring(begin) & String substring(begin,end)

获取字符串

5 大数处理

首先我们需要导包，即 BigInteger 类和 BigDecimal 类所在的包

```
1. import java.math.*;
```

*就代表导入包 math 里面所有的类，如果你不喜欢看到 *

那么你也可以写 import java.math.BigInteger; import java.math.BigDecimal;

5.1 大数输入

```
1. public class Main
2. {
3.     public static void main(String[] args)
4.     {
5.         Scanner cin = new Scanner ( System.in);
6.         BigInteger c;
7.         a = cin.nextInt();
8.         b = cin.nextDouble();
9.         c = cin.nextBigInteger();
```

```
10.    }
11. }
```

5.2 大数输出

```
1. public class Main
2. {
3.     public static void main(String[] args)
4.     {
5.         /*
6.         规格化的输出:
7.         函数: 这里 0 指一位数字, #指除 0 以外的数字(如果是 0, 则不显示), 四舍五入.
8.         */
9.         DecimalFormat fd = new DecimalFormat("#.00#");
10.        DecimalFormat gd = new DecimalFormat("0.000");
11.        System.out.println("x =" + fd.format(x));
12.        System.out.println("x =" + gd.format(x));
13.    }
14. }
```

5.3 大数的加减乘除求余等计算

```
1. /*
2. 大数的加减运算不同于普通整数的加减乘除运算
3. 加— a+b: a=a.add(b);
4. 减— a-b: a=a.subtract(b);
5. 乘— a*b: a=a.multiply(b);
6. 除— a/b: a=a.divide(b);
7. 求余—a%b: a=a.mod(b);
8. 转换—a=b: b=BigInteger.valueOf(a);
9. 比较 if (ans.compareTo(x) == 0)//比较
10.    System.out.println("相等");
11. System.out.println("a + b = "+ans_add); // 这里的‘+’ （第二个） 是连接的意思
12. */
13.
14. import java.util.*;
15. import java.math.*;
16. public class Main {
17.     public static void main(String args[]) {
```

```

18. Scanner cin = new Scanner(System.in);
19. BigInteger a,b,x,y;
20. BigInteger ans_add,ans_sub,ans_mul,ans_div,ans_mod;
21. a=cin.nextBigInteger();
22. b=cin.nextBigInteger();
23. ans_add = a.add(b); //a+b
24. ans_sub = a.subtract(b); //a-b
25. ans_mul = a.multiply(b); //a*b
26. ans_div = a.divide(b); //a/b
27. ans_mod = a.mod(b); //a%b
28. x=BigInteger.valueOf(1); //转换
29. System.out.println("a + b = "+ans_add);
30. System.out.println("a - b = "+ans_sub);
31. System.out.println("a * b = "+ans_mul);
32. System.out.println("a / b = "+ans_div);
33. System.out.println("a % b = " + ans_mod);
34. System.out.println(x);
35. if (a.compareTo(b) == 0) //比较
36.     System.out.println("相等");
37. else
38.     System.out.println("不相等");
39. }
40. }

```

关于 BigDecimal 的用法大致上和 BigInteger 一样。

不过这里需要提一下，在进行大浮点数运算的时候，小数点后面可能会含有多余的后导 0

比如 0.5000，在题目要求中可能只需要输出 0.5

当然，有的题目可能还会要求小数点前面的 0 也要去掉，输入.5

这时候我们就需要去除掉后导 0

转化成 字符型的

方法如下：

```

1. String str;
2. str=ans.stripTrailingZeros().toPlainString(); //去除所有后导 0，并且转化成字符
   型
3. //ans 为大浮点数运算后得到的答案
4. //如果小数点前面的 0 也需要去掉，那么输出的时候处理一下即可：
5. if(str.charAt(0)=='0') //如果以 0 开头
6.     System.out.println(str.substring(1)); //返回以位置 1 开头的字符串
7. else
8.     System.out.println(str);

```


6 日期时间

6.1 计算日期差

```
1. import java.time.LocalDate;
2. import java.time.Month;
3. import java.time.Period;
4. import java.time.temporal.ChronoUnit;
5.
6. public class Date {
7.     public static void main(String[] args) {
8.         LocalDate startDate = LocalDate.of(1993, Month.OCTOBER, 19);
9.         System.out.println("开始时间 : " + startDate);
10.
11.         LocalDate endDate = LocalDate.of(2017, Month.JUNE, 16);
12.         System.out.println("结束时间 : " + endDate);
13.
14.         long daysDiff = ChronoUnit.DAYS.between(startDate, endDate);
15.         System.out.println("两天之间的差在天数 : " + daysDiff);
16.
17.         Period p = Period.between(startDate, endDate);
18.         System.out.printf("两天之间的差 : %d 年 %d 月 %d 日", p.getYears(), p.getMonths(), p.getDays());
19.     }
20. }
```

输出结果：

```
1. 开始时间 : 1993-10-19
2. 结束时间 : 2017-06-16
3. 两天之间的差在天数 : 8641
4. 两天之间的差 : 23 年 7 月 28 日
```

6.2 计算时间差

```
1. import java.time.Duration;
2. import java.time.Instant;
3.
4. public class Time {
5.
6.     public static void main(String[] args) {
7.         Instant inst1 = Instant.now();
```

```
8.      System.out.println("Inst1 : " + inst1);
9.      Instant inst2 = inst1.plus(Duration.ofSeconds(10));
10.     System.out.println("Inst2 : " + inst2);
11.
12.     System.out.println("Difference in milliseconds : " + Duration.between(
    n(inst1, inst2).toMillis());
13.
14.     System.out.println("Difference in seconds : " + Duration.between(inst1, inst2).getSeconds());
15. }
16. }
```

输出结果：

```
1. Inst1 : 2019-05-06T14:24:21.037Z
2. Inst2 : 2019-05-06T14:24:31.037Z
3. Difference in milliseconds : 10000
4. Difference in seconds : 10
```