Java 快速上手基础突击

版本	时间	作者
1.0	2020-09-27	张新强

目录

Java 基础概念

方法

方法：就是完成特定功能的代码块。注意：在很多语言里面有函数的定义，而在 Java 中，函数被称为方法。

格式：

```
修饰符 返回值类型 方法名(参数类型 参数名1,参数类型 参数名2...) {
        方法体语句;
        return 返回值;
}
```

修饰符 public static private protectd default void

返回值类型：就是功能结果的数据类型

方法名：就是起了一个名字，方便我们调用该方法。

参数类型：就是参数的数据类型

参数名：就是变量

参数分类：

实参：实际参与运算的数据

形参：方法上定义的，用于接收实际参数的变量

方法体语句：就是完成功能的代码块

return：结束方法

返回值：就是功能的结果，由 return 带给调用者。

权限修饰符作用范围：

访问修饰符\作用范围	所在类	同一包内其他类	其他包内子类	其他包内非子类
private	可以访问	不可以	不可以	不可以
缺省	可以	可以	不可以	不可以
protected	可以	可以	可以	不可以
public	可以	可以	可以	可以

参数传递

Java 中只有值传递。

基本类型：形式参数的改变不影响实际参数

引用类型：形式参数的改变直接影响实际参数

This

代表当前类的引用对象

哪个对象调用方法，该方法内部的 this 就代表那个对象

Final

是最终的意思，可以修饰类，方法，变量。

它修饰的类，不能被继承。

它修饰的方法，不能被重写。

它修饰的变量，是一个常量。

Java 的各种类的定义

类：class

接口：interfact

枚举：enum

注解：@interfact

实现接口用：implements　接口可以多实现

继承类使用：extends 继承只能单继承

常用类介绍

String 类

多个字符组成的一串数据。

其实它可以和字符数组进行相互转换。

(1)多个字符组成的一串数据。

其实它可以和字符数组进行相互转换。

(2)构造方法:

A:public String()

B:public String(byte[] bytes)

C:public String(byte[] bytes,int offset,int length)

D:public String(char[] value)

E:public String(char[] value,int offset,int count)

F:public String(String original)

下面的这一个虽然不是构造方法，但是结果也是一个字符串对象

G:String s = "hello";

(3)字符串的特点

A:字符串一旦被赋值，就不能改变。

注意：这里指的是字符串的内容不能改变，而不是引用不能改变。

(4)字符串的功能

A:判断功能

boolean equals(Object obj)

boolean equalsIgnoreCase(String str)

```
boolean contains(String str)
boolean startsWith(String str)
boolean endsWith(String str)
boolean isEmpty()
```

B:获取功能

```
int length()
char charAt(int index)
int indexOf(int ch)
int indexOf(String str)
int indexOf(int ch,int fromIndex)
int indexOf(String str,int fromIndex)
String substring(int start)
String substring(int start,int end)
```

C:转换功能

```
byte[] getBytes()
char[] toCharArray()
static String valueOf(char[] chs)
static String valueOf(int i)
String toLowerCase()
String toUpperCase()
String concat(String str)
```

D:其他功能

a:替换功能

```
String replace(char old,char new)
```

```
String replace(String old,String new)
```

b:去空格功能

```
String trim()
```

c:按字典比较功能

```
int compareTo(String str)
int compareToIgnoreCase(String str)
```

StringBuilder

用字符串 sz 拼接，比较耗时并且也耗内存，而这种拼接操作又是比较常见的，为了解决这个问题，Java 就提供了一个字符串缓冲区类。StringBuilder 供我们使用。

(1)StringBuilder 的构造方法

A: StringBuilder ()

B: StringBuilder (int size)

C: StringBuilder (String str)

(2) StringBuilder 的常见功能

A:添加功能

```
public StringBuilderappend(object obj);
```

B:删除功能

```
public StringBuilder delete(int start,int end);
public StringBuilder deleteCharAt(int index);
```

C:替换功能

```
public StringBuilder replace(int start,int end,String str);
```

D:反转功能

```
public StringBuilder reverse();
```

E:截取功能(注意这个返回值)

```
public String substring(int start,int end);
```

ArrayList

可调整大小的数组的实现 List 接口。 实现所有可选列表操作，并允许所有元素，包括 null 。

(1)构造方法

ArrayList()

构造一个初始容量为十的空列表。

ArrayList(Collection<? extends E> c)

构造一个包含指定集合的元素的列表，按照它们由集合的迭代器返回的顺序。

ArrayList(int initialCapacity)

构造具有指定初始容量的空列表。

(2)常用方法

A：将指定的元素追加到此列表的末尾。

```
public boolean add(E e)
```

B：在此列表中的指定位置插入指定的元素。 将当前位于该位置的元素（如果有）和任何后续元素（向其索引添加一个）移动。

```
public void add(int index,E element)
```

C：按指定集合的 Iterator 返回的顺序将指定集合中的所有元素追加到此列表的末尾。

```
public boolean addAll(Collection<? extends E> c)
```

D：从列表中删除所有元素。 此呼叫返回后，列表将为空。

```
public void clear()
```

E：如果此列表包含指定的元素，则返回 true 。

```
public boolean contains(Object o)
```

F：返回此列表中指定位置的元素。

public E get(int index)

G：删除该列表中指定位置的元素。

public E remove(int index)

H：返回此列表中的元素数。

public int size()

特别注意：

并发修改异常

A:出现的现象

迭代器遍历集合，集合修改集合元素

B:原因

迭代器是依赖于集合的，而集合的改变迭代器并不知道。

C:解决方案

a:迭代器遍历，迭代器修改(ListIterator)

元素添加在刚才迭代的位置

b:集合遍历，集合修改(size()和 get())

元素添加在集合的末尾

HashSet

此类实现 Set 接口，由哈希表（实际为 HashMap 实例）支持。 对集合的迭代次序不作任何保证; 特别是，它不能保证订单在一段时间内保持不变。 这个类允许 null 元素。

(1)特别说明

HashSet 底层有 HashMap 实现，HashSet 不可以存储重复元素，拥有去重功能，HashSet 不保证取出元素时的顺序与存入时一致。

(2)构造方法

HashSet()

构造一个新的空集合; 背景 HashMap 实例具有默认初始容量（16）和负载因子（0.75）。

HashSet(Collection<? extends E> c)

构造一个包含指定集合中的元素的新集合。

HashSet(int initialCapacity)

构造一个新的空集合; 背景 HashMap 实例具有指定的初始容量和默认负载因子（0.75）。

HashSet(int initialCapacity, float loadFactor)

构造一个新的空集合; 背景 HashMap 实例具有指定的初始容量和指定的负载因子。

注意：负载因子是指，当实际使用容量到达指定容量*负载因子的值时，会自动扩容。

(3)常用方法

A：将指定的元素添加到此集合（如果尚未存在）。 更正式地，将指定的元素 e 添加到此集合，如果此集合不包含元素 e2 ，使得(e==null ? e2==null : e.equals(e2)) 。 如果该集合已经包含该元素，则该呼叫将保持不变，并返回 false 。

public boolean add(E e)

B：如果存在，则从该集合中删除指定的元素。 更正式地，删除一个元素 e ，使(o==null ? e==null : o.equals(e)) ，如果这个集合包含这样一个元素。如果此集合包含元素（或等效地，如果该集合由于调用而更改），则返回 true 。 （一旦调用返回，此集合将不包含该元素。）

public boolean remove(Object o)

C：其他方法参考 Set 集合接口

HashSet 使用频率并不是很高，需要时查找 api

HashSet 底层

A:底层数据结构是哈希表(是一个元素为链表的数组)

B:哈希表底层依赖两个方法：hashCode()和 equals()

执行顺序：

首先比较哈希值是否相同

相同：继续执行 equals()方法

返回 true：元素重复了，不添加

返回 false：直接把元素添加到集合

不同：就直接把元素添加到集合

C:如何保证元素唯一性的呢?

由 hashCode()和 equals()保证的

HashMap

基于哈希表的实现的 Map 接口。 此实现提供了所有可选的地图操作，并允许 null 的值和 null 键。 （ HashMap 类大致相当于 Hashtable ，除了它是不同步的，并允许 null）。这个类不能保证地图的顺序; 特别是，它不能保证订单在一段时间内保持不变。也就是说，HashMap 的存储顺序，随时有可能改变。

构造方法

HashMap()

构造一个空的 HashMap ，默认初始容量（16）和默认负载系数（0.75）。

HashMap(int initialCapacity)

构造一个空的 HashMap 具有指定的初始容量和默认负载因子（0.75）。

HashMap(int initialCapacity, float loadFactor)

构造一个空的 HashMap 具有指定的初始容量和负载因子。

HashMap(Map<? extends K,? extends V> m)

构造一个新的 HashMap 与指定的相同的映射 Map 。

常用方法

A：清空 map

public void clear()

B：判断是否包含某个键

public boolean containsKey(Object key)

C：向 map 添加值

public V put(K key,V value)

D：删除指定键值对

public V remove(Object key)

E：将所有键整理成 Set 集合并返回

public Set<K> keySet()

F：将所有键值对提取，并用 Set 集合返回

public Set<Map.Entry<K,V>> entrySet()

泛型

是一种把明确类型的工作推迟到创建对象或者调用方法的时候才去明确的特殊的类型。

写法:

〈数据类型〉
注意：该数据类型只能是引用类型。

好处:

把运行时期的问题提前到了编译期间

避免了强制类型转换

优化了程序设计，解决了黄色警告线问题，让程序更安全

举例:

List<String> stringList=new ArrayList<>();

这样 stringList 中只要存储了非 String 类型，就会报错。在获取元素时，也会直接返回 String 类型的元素。

可变参数

使用时机:

如果我们在写方法的时候，参数个数不明确，就应该定义可变参数。

写法:

```
修饰符 返回值类型 方法名(数据类型... 变量) {}
public class ChangeDemo {
	public static void main(String[] args) {
		int a=10;
		int b=10;
		int c=10;
		System.out.println(sum(a,b,c,29));
	}
	//此方法使用了可变参数。
	public static int sum(int...a){
		int res=0;
		for(int x=0;x<a.length;x++){
			res+=a[x];
		}
		return res;
	}
}
```

注意:

该变量其实是一个数组名
如果一个方法有多个参数，并且有可变参数，可变参数必须在最后

IO 流

IO 用于在设备间进行数据传输的操作

分类:

A:流向

输入流　　读取数据 InputStream　Reader （都是抽象类）

输出流　　写出数据 OutputStream Writer　（都是抽象类）

B:数据类型

字节流

字节输入流　InputStream （是抽象类）

字节输出流　OutputStream（是抽象类）

字符流

字符输入流　Reader （是抽象类）

字符输出流　Writer　（是抽象类）

注意：

a:如果我们没有明确说明按照什么分，默认按照数据类型分。

b:除非文件用 windows 自带的记事本打开我们能够读懂，才采用字符流，否则建议使用字节流，字节流都可以处理。

常用字节流:

字节输入流：BufferedInputStream FileInputStream 前者是带缓冲区的流，包装后者

字节输出流：BufferedOutputSteam FileOutputSteam 前者时代缓冲区的流，包装后者

通过包装类，提高流的输入输出效率。

IO 常用流知识图谱:

IO 流

|--字节流

|--字节输入流

InputStream

int read():一次读取一个字节

int read(byte[] bys):一次读取一个字节数组

|--FileInputStream

|--BufferedInputStream

|--字节输出流

OutputStream

void write(int by):一次写一个字节

void write(byte[] bys,int index,int len):一次写一个字节数组的一部分

|--FileOutputStream

|--BufferedOutputStream

|--字符流

|--字符输入流

Reader

int read():一次读取一个字符

int read(char[] chs):一次读取一个字符数组

|--InputStreamReader

|--FileReader

|--BufferedReader

String readLine():一次读取一个字符串

|--字符输出流

Writer

void write(int ch):一次写一个字符

void write(char[] chs,int index,int len):一次写一个字符数组的一部分

|--OutputStreamWriter

|--FileWriter

|--BufferedWriter

void newLine():写一个换行符

void write(String line):一次写一个字符串