

JAVA 基础复习

1.JAVA 基础复习——计算机基础与环境变量配置

软件开发的了解

软件开发：

软件：一系列按照特定组织的计算机数据和指令的集合。

开发：制作软件。

程序：一系列有序指令的集合。

人机交互

人机交互的方式有两种：图形化界面和命令行方式。

图形化界面：简单直观易于操作。

命令行方式：需要一个控制台需要了解一些特定的指令，较为麻烦。

计算机语言

语言：是人与人之间沟通的一种方式。

计算机语言：是人与计算机交流的方式。

Java 语言概述

由 Sun 公司 1995 年推出的一门语言。

特点:简单易学，完全面向对象，跨平台。

Java 三大版本

J2SE：标准版，是 Java 语言的基础和核心，也是 J2EE 和 J2ME 技术的基础，主要用于开发桌面应用程序。

J2EE：企业版，提供了企业级应用开发的完整解决方案，主要用于 Web 程序的开发，是 Java 技术应用最广泛的领域。

J2ME：微缩版，主要用于嵌入式开发，开发移动端程序。

注:Java 5.0 版本后更名为 Java SE、Java EE 和 Java ME。

Java 虚拟机

Java 虚拟机简称 JVM，用来解析 Java 编写的程序的工具，是程序与平台（操作系统）的一个桥梁。

Java 语言跨平台原理

通过在不同平台（操作系统）安装不同版本的 JVM 来解析 Java 程序，从而实现了跨平台。

Java 环境变量搭建

JRE:Java 运行环境，包含 JVM 和 Java 程序所需的核心类库，用来运行 Java 程序。

JDK（包含 JRE）:JRE+Java 开发工具：编译器、打包工具等。

简单概述：用 JDK 开发完成程序交给 JRE 运行。

Java 环境变量的配置

Java 需要配置的环境变量：

PATH: Java 环境变量的主要配置，为了可以在任何地方对源文件进行编译。

JAVA_HOME: 为了在 JDK 路径有改动的时候不影响 PATH 变量单独为 JDK 新建一个变量，防止对其他系统变量的误操作。

CLASS_PATH:执行 class 文件时，查找 class 文件的位置的范围，Classpath 的目的，在于告诉 Java 执行环境，在哪些目录下可以找到您所执行的 Java 程序所需要的类或者包。

1.下载并安装 JDK

2.右键计算机——》属性——》高级配置——》环境变量——》系统变量

3.新建 JAVA_HOME 变量将 JDK 安装路径复制给 JAVA_HOME 变量，在改变 JDK 路径时 PATH 变量不会受到影响。

4.找到系统变量 PATH 进行编辑，在最前面加上 `%JAVA_HOME%\bin;` 路径，注意用；来与其他系统变量分隔（%%：用来引用某个系统变量，这里引用了 JAVA_HOME 的值）。

5.新建 CLASS_PATH 变

量 `.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar;`

验证 Java 环境变量配置是否成功

打开控制台输入 `javac` 显示如下界面表示配置成功

```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\haochunlu>
C:\Users\haochunlu>javac
用法: javac <options> <source files>
其中, 可能的选项包括:
-g 生成所有调试信息
-g:none 不生成任何调试信息
-g:{lines,vars,source} 只生成某些调试信息
-nowarn 不生成任何警告
-verbose 输出有关编译器正在执行的操作的消息
-deprecation 输出使用已过时的 API 的源位置
-classpath <路径> 指定查找用户类文件和注释处理程序的位置
-cp <路径> 指定查找用户类文件和注释处理程序的位置
-sourcepath <路径> 指定查找输入源文件的位置
-bootclasspath <路径> 覆盖引导类文件的位置
-extdirs <目录> 覆盖所安装扩展的位置
-endorseddirs <目录> 覆盖签名的标准路径的位置
-proc:{none,only} 控制是否执行注释处理和/或编译。
-processor <class1>[,<class2>,<class3>...] 要运行的注释处理程序的名称; 绕过默认
的搜索进程
-processorpath <路径> 指定查找注释处理程序的位置
-d <目录> 指定放置生成的类文件的位置
-s <目录> 指定放置生成的源文件的位置
-implicit:{none,class} 指定是否为隐式引用文件生成类文件
-encoding <编码> 指定源文件使用的字符编码
-source <发行版> 提供与指定发行版的源兼容性
-target <发行版> 生成特定 VM 版本的类文件
-version 版本信息
-help 输出标准选项的提要
-A关键字[=值] 传递给注释处理程序的选项
-X 输出非标准选项的提要
-J<标记> 直接将 <标记> 传递给运行时系统
-Werror 出现警告时终止编译
-e<文件名> 从文件读取选项和文件名

C:\Users\haochunlu>

半:
```

2.0 JAVA 基础复习——JAVA 语言的基础组成关键字和标识符

JAVA 语言的基础组成有:

- 1.关键字: 被赋予特殊含义的单词。
- 2.标识符: 用来标识的符号。
- 3.注释: 用来注释说明程序的文字。
- 4.常量和变量: 内存存储区域的表示。
- 5.运算符: 程序中用来运算的符号。
- 6.语句: 程序中常用的一些语句。
- 7.函数: 也叫做方法, 用来做一些特定的动作。

8.数组：用来存储多个数据的集合。

JAVA 中的关键字：

JAVA 中的关键字不能用作变量名、方法名、参数、类名、包名，也就是不能作为标识符来应用。

JAVA 中的关键字有：

关键字	含义
-----	----

abstract ：	表明类或者成员方法具有抽象属性
-------------------	-----------------

assert ：	用来进行程序调试
-----------------	----------

boolean ：	基本数据类型之一，布尔类型
------------------	---------------

break ：	提前跳出一个块
----------------	---------

byte ：	基本数据类型之一，字节类型
---------------	---------------

case ：	用在 switch 语句之中，表示其中的一个分支
---------------	---------------------------------

catch ：	用在异常处理中，用来捕捉异常
----------------	----------------

char ：	基本数据类型之一，字符类型
---------------	---------------

class ：	类
----------------	---

const ：	保留关键字，没有具体含义
----------------	--------------

continue ：	回到一个块的开始处
-------------------	-----------

default ：	默认，例如，用在 switch 语句中，表明一个默认分支
------------------	-------------------------------------

do ：	用在 do-while 循环结构中
-------------	--------------------------

double ：	基本数据类型之一，双精度浮点数类型
-----------------	-------------------

else ：	用在条件语句中，表明当条件不成立时的分支
---------------	----------------------

enum ：	枚举
---------------	----

extends ：	表明一个类型是另一个类型的子类型，这里常见的类型有类和接口
------------------	-------------------------------

final ：	用来说明最终属性，表明一个类不能派生出子类，或者成员方法不能被覆盖，或者成员域的值不能被改变
----------------	--

finally ：	用于处理异常情况，用来声明一个基本肯定会被执行到的语句块
------------------	------------------------------

float ：	基本数据类型之一，单精度浮点数类型
----------------	-------------------

for ：	一种循环结构的引导词
--------------	------------

goto ：	保留关键字，没有具体含义
---------------	--------------

if ：	条件语句的引导词
-------------	----------

implements ：	表明一个类实现了给定的接口
---------------------	---------------

import ：	表明要访问指定的类或包
-----------------	-------------

instanceof ：	用来测试一个对象是否是指定类型的实例对象
---------------------	----------------------

int ：	基本数据类型之一，整数类型
--------------	---------------

interface ：	接口
--------------------	----

long ：	基本数据类型之一，长整数类型
---------------	----------------

native ：	用来声明一个方法是由与计算机相关的语言（如 C/C++/FORTRAN 语言）实现的
-----------------	--

new ：	用来创建新实例对象
--------------	-----------

null ：	用来标识一个不确定的对象
---------------	--------------

package ：	包
------------------	---

private ：	一种访问控制方式：私用模式
------------------	---------------

protected ：	一种访问控制方式：保护模式
--------------------	---------------

public : 一种访问控制方式: 共用模式
return: 从成员方法中返回数据
short: 基本数据类型之一,短整数类型
static : 表明具有静态属性
strictfp: 用来声明 **FP_strict** (单精度或双精度浮点数) 表达式遵循 IEEE 754 算术规范
super : 表明当前对象的父类型的引用或者父类型的构造方法
switch : 分支语句结构的引导词
synchronized: 表明一段代码需要同步执行
this: 指向当前实例对象的引用
throw : 抛出一个异常
throws: 声明在当前定义的成员方法中所有需要抛出的异常
transient : 声明不用序列化的成员域
try : 尝试一个可能抛出异常的程序块
void : 声明当前成员方法没有返回值
volatile: 表明两个或者多个变量必须同步地发生变化
while : 用在循环结构中

JAVA 中的标识符:

JAVA 中的标识符内容可以随便定义,但尽量要定义有意义的单词,标识符由字母、`_`、`$`、数字组成,标识符其实就是给类、方法、变量起个名字。

JAVA 标识符需要注意的规则:

- 1.标识符不能以数字开头,标识符不能使用关键字。
- 2.类名首字母需要大写面跟的单词大写 如: `MyTest{}`。
- 3.方法名小写后面跟的单词大写 如: `findName()`;
- 4.变量名小写后面跟的单词大写 如: `myName`
- 5.常量名全部大写 如: `HOUSE`
- 6.包名全部小写 如: `com.test.entity`

2.1JAVA 基础复习——JAVA 语言的基础组成注释和常量变量

JAVA 语言的基础组成有:

- 1.关键字: 被赋予特殊含义的单词。
- 2.标识符: 用来标识的符号。
- 3.注释: 用来注释说明程序的文字。
- 4.常量和变量: 内存存储区域的表示。
- 5.运算符: 程序中用来运算的符号。

- 6.语句：程序中常用的一些语句。
- 7.函数：也叫做方法，用来做一些特定的动作。
- 8.数组：用来存储多个数据的集合。

JAVA 中的注释：

注释还可以用来缩小程序错误的范围，方便查找错误。

// ：表示单行注释。

```
//这是一个单行注释
```

/**/：表示多行注释。

```
/*  
  这是一个  
  多行注释  
*/
```

/**/：java 中独有的多行注释用于文档的注释

```
/**  
  这是 java 中  
  独有的多行注释  
  一般用于文档注释  
*/
```

JAVA 中的常量与变量：

常量：其值不可改变的量，只能定义一次，通常用来定义不变的量用 **final** 来修饰 如：

语法：final 数据类型 常量名 = 值；

```
public static final String str="常量"
```

变量：其值可以改变的量，内存空间中的一个存储区域，通常用来定义经常需要改变的量
如：

语法：数据类型 变量名 = 值；

```
int age = 18;  
age=19;  
//这时候 age 可以改变 age 的值覆盖掉了原来的值变成了 19
```

JAVA 中的数据类型：

JAVA 中的数据类型分为：基本数据类型和引用数据类型。

JAVA 中的基本数据类型有四类八种：整数类型、小数类型、字符类型、布尔类型。

整数类型：byte、short、int、long

占用字节： 1 2 4 8

小数类型：float、double

占用字节： 4 8

字符类型：char

占用字节： 2

布尔类型：boolean

占用字节：布尔类型只有 true 或 false 理论上只占用 1bit，所以按 1byte 处理。

JAVA 从小到大排列顺序：byte、short、char、int、float、long、double、boolean

JAVA 中的引用数据类型有：类(class)、接口(interface)、数组([])。

JAVA 中的类型转换：

JAVA 中的类型转换也有两种：一种是自动类型转换（隐式转换），另一种是强制类型转换（显示转换）。

自动类型转换：将一个数值 A 赋值给另一个数值 B，如果 A 的类型小于 B 类型，系统会自动提升 A 的类型使他与 B 类型一致，然后再赋值给 B

基本数据类型自动转换：

byte——》short、char——》int——》long

float——》double

int——》float

long——》double

强制类型转换：将一个数值 A 赋值给另一个数值 B，如果 A 的类型大于 B 类型，需要我们将强制转换 A 的类型使他与 B 类型一致，然后再赋值给 B

强制类型转换：需要注意的是强制类型转换可能会出现丢失精度的风险要慎用。

```
int a=10;
```

```
short b = (short)a;
```

JAVA 中如何将字符串数值转换成数值类型。

```
String s = "123";
```

```
int num = Integer.parseInt(s);
```

利用基本类型的封装类的.parseInt.....()方法来将字符串转换成相应的数值类型。

2.2JAVA 基础复习——JAVA 语言的基础组成运算符和语句

JAVA 语言的基础组成有：

- 1.关键字：被赋予特殊含义的单词。
- 2.标识符：用来标识的符号。
- 3.注释：用来注释说明程序的文字。
- 4.常量和变量：内存存储区域的表示。
- 5.运算符：程序中用来运算的符号。
- 6.语句：程序中常用的一些语句。
- 7.函数：也叫做方法，用来做一些特定的动作。
- 8.数组：用来存储多个数据的集合。

JAVA 中的运算符

- 1.算术运算符：用来进行一些数据算法的符号

算术运算符分为单目运算符、双目运算符、三目运算符。

单目运算符有：+（取正）-（取负）++（自增）--（自减）代码如下：

```
1 1 //算术运算符：单目运算符+(取正)-(取负)
2 2 int a = 5;
3 3 //取 a 的正数 b=5
4 4 int b = +a;
5 5 //取 a 的负数 c=-5
6 6 int c = -a;
7 7 //单目运算符++(自增)自减原理与自增一样故省略
8 8 int d = 0;
9 9 int e = 0;
10 10 //f 的结果为 0，++在后面会先将 d 的值赋给 f 然后 d 自增 1 结
果为 0
11 11 int f = d++;
12 12 //g 的结果为 1，++在前面会先将 e 的值+1 然后再赋值给 g 结
果为 1
```



```
13 13      int g = ++e;
```

双目运算符有：+（加）-（减）*（乘）/（除）%（取余）代码如下：

```
1 1      //算术运算符：双目运算符+(加)-(减)*(乘)/(除)%(取余)
2 2      int a = 5;
3 3      int b = 2;
4 4      //双目运算符可以对变量进行运算结果为：7
5 5      int c = a+b;
6 6      //也可以直接对数值进行运算结果为：5
7 7      int d = 10-5;
8 8      //结果为：10
9 9      int e = a*b;
10 10     //结果为：2 因为 java/默认的两个整数相除返回的也是整数
11 11     int f = a/b;
12 12     //结果为:1 因为 5/2 余 1%就是来去余数的所以为 1
13 13     int j = a%b;
```

三目运算符有：a>b?true:false 判断 a 是否大于 b 如果大于 b 返回 true 也就是：前的值否则返回 false（true 和 false）可以自己定义想要返回的值

如：a>b?a:b 判断 a 是否大于 b 如果大于 b 返回 a 的值否则返回 b 的值代码如下：

[View Code](#)

2.关系运算符:用来判断数据比较关系的符号

关系运算符有：==（等于）!=（不等于）>（大于）<（小于）>=（大于等于）<=（小于等于）

[View Code](#)

3.逻辑运算符：用来判断数据逻辑关系的符号

逻辑运算符有：与（&&）或（||）非（!）

[View Code](#)

4.位运算符：用来对二进制位进行操作的符号（位运算比算术运算符效率高）

位运算符有：与（&）或（|）非（~）异或（^）<<（左移）>>（右移）>>>（无符号右移）

 View Code


5.赋值运算符：用来赋值的运算符将右边的值赋给左边

赋值运算符有：=、+=、-=、*=、/=、%=、&=、|=、^=、<<=、>>=、>>>=

 View Code

JAVA 中的常用语句

if 语句语法：

```

1      /*
2      * if 语句：
3      *      语法 1：if(条件){
4      *          执行体
5      *      }
6      *
7      *      语法 2：if(条件){
8      *          执行体
9      *      }else{
10     *          执行体
11     *      }
12     *
13     *      语法 3：if(条件){
14     *          执行体
15     *      }else if(条件){
16     *          执行体
17     *      }else{
18     *          执行体
19     *      }
20     */
21     int a = 2;
22     int b = 3;
23     //语法 1 示例：条件成立输出 a 小于 b 不成立则不输出
```

```

24         if(a<b) {
25             System.out.println("a 小于 b");
26         }
27         //语法 2 示例：条件成立输出 a 小于 b 不成立输出 a 小于 b
28         if(a>b) {
29             System.out.println("a 大于 b");
30         }else{
31             System.out.println("a 小于 b");
32         }
33         //语法 3 示例：条件成立输出 a 小于 b 不成立继续判断条件成立输出执行体不成立输出 else 的执行体
34         if(a>b) {
35             System.out.println("a 大于 b");
36         }else if(a<b) {
37             System.out.println("a 小于 b");
38         }else{
39             System.out.println("a 等于 b");
40         }
41         运行结果:1. a 小于 b
                2. a 小于 b
                3. a 小于 b

```



switch 语句



```

/*
 * switch 语句:
 *     需要注意的是 jdk1.7 以下版本 key 只能是 int 或能自动转换成 int 类型的值如: byte、short、char 和枚举 enum 类型
 *     jdk1.7 后支持字符串 case 可多个
 * 语法:
 *     switch (key) {
 *         case value:
 *             执行体
 *             break;
 *
 *         default:
 *             没有符合条件执行体
 *             break;
 *     }
 * */
//如果 week=1 输出星期一

```

```
int week = 0;
switch (week) {
case 1:
    System.out.println("星期一");
    break;
case 2:
    System.out.println("星期二");
    break;
case 3:
    System.out.println("星期三");
    break;
case 4:
    System.out.println("星期四");
    break;
case 5:
    System.out.println("星期五");
    break;
case 6:
    System.out.println("星期六");
    break;
case 7:
    System.out.println("星期日");
    break;

default:
    System.out.println("输入有误");
    break;
}
//运行结果： 输入有误
```



Java 中的循环：

1.while 循环： 符合条件执行循环不符合条件跳出循环

2.do—while 循环: 先执行一次循环，在判断条件是否符合，如果符合继续循环，不符合跳出循环

3.for 循环: 在明确循环次数时用 for 循环，判断条件是否符合，如果符合继续循环，不符合跳出循环

while 循环：



```
/*
 * while 语法:
```

```

*    while(条件){
*        执行体
*    }
*
* */

```



```

int c = 0;
while (c<5) {
    System.out.println(c);
    c++;
}
//运行结果:0、1、2、3、4

```



do-while 循环:

```

/*
* do-while 语法:
*    do{
*        执行体
*    }while(条件);
* */

```

```

int d = 6;
do {
    System.out.println(d);
    d++;
} while (d<5);
//运行结果: 6

```

for 循环:



```

/*
* for 语法:
*    for(条件 1; 条件 2; 条件 3){
*        执行体
*    }
* */
int[] a = new int[] {1, 12, 55, 33};

```

```
for (int i = 0; i < a.length; i++) {  
    System.out.println(a[i]);  
}
```



2.3 JAVA 基础复习——JAVA 语言的基础组成函数

JAVA 语言的基础组成有：

- 1.关键字：被赋予特殊含义的单词。
- 2.标识符：用来标识的符号。
- 3.注释：用来注释说明程序的文字。
- 4.常量和变量：内存存储区域的表示。
- 5.运算符：程序中用来运算的符号。
- 6.语句：程序中常用的一些语句。
- 7.函数：也叫做方法，用来做一些特定的动作。
- 8.数组：用来存储多个数据的集合。

JAVA 中的函数：

函数的定义：

函数就是定义在勒种具有独特功能的一段独立的小程序，函数也成为方法。

函数的格式：

```
访问修饰符 返回值类型 函数名（参数类型 形式参数 1， 参数类型 形式参数  
2， .....） {  
    方法体;  
    return 返回值;  
}
```

访问修饰符：用来修饰函数的作用范围。

返回值类型：函数运行后的结果的数据类型。

参数类型：是形式参数的数据类型。

形式参数：是一个变量，用于存储调用函数时传递给函数的实际参数。

实际参数：传递给形式参数的具体数值。

return：用于结束函数。

返回值：该值会返回给调用者。

函数的特点：

- a) 定义函数可以将功能代码进行封装
- b) 便于对该功能进行复用
- c) 函数只有被调用才会被执行
- d) 函数的出现提高了代码的复用性
- e) 对于函数没有具体返回值的情况，返回值类型用关键字 `void` 表示，那么该函数中的 `return` 语句如果在最后一行可以省略不写，系统会帮你自动加上。

注：

- a) 函数中只能调用函数，不可以在函数内部定义函数。
- b) 定义函数时，函数的结果应该返回给调用者，交由调用者处理。
- c) 当函数运算后，没有具体的返回值时，这是返回值类型用一个特殊的关键字来标识该关键字就是 `void`，`void`：代表的是函数没有具体返回值的情况。
- d) 当函数的返回值类型是 `void` 时，函数中的 `return` 语句可以省略不写。

如何定义一个函数：

函数其实就是一个功能，定义函数就是实现功能，通过两个明确来完成：

- 1)、明确该功能的运算完的结果，其实是在明确这个函数的返回值类型。
- 2)、在实现该功能的过程中是否有未知内容参与了运算，其实就是在明确这个函数的参数列表(参数类型&参数个数)。

函数的作用：

- 1)、用于定义功能。
 - 2)、用于封装代码提高代码的复用性。
- 注意：函数中只能调用函数，不能定义函数。

为什么要定义函数的名称：

- 1)、为了对该功能进行标示，方便于调用。
- 2)、为了通过名称就可以明确函数的功能，为了增加代码的阅读性。

主函数：

- 1)、保证该类的独立运行。
- 2)、因为它是程序的入口。
- 3)、因为它在被 `jvm` 调用。

函数的应用：

- 1) 两个明确
 - a) 明确要定义的功能最后的结果是什么？
 - b) 明确在定义该功能的过程中，是否需要未知内容参与运算。

示例 1



```
1 class FunctionDemo
```

```

2 {
3 public static void main(String[] args)
4 {
5     int x = 4;
6     System.out.println(x*3+5);
7     x = 6;
8     System.out.println(x*3+5);
9     int y = 4*3+5;
10    int z = 6*3+5;
11        System.out.println(y);
12        System.out.println(z);
13 }
14

```

如上面的代码我们需要多次用到方法中同一个运算为了提高代码的复用性，我们就可以定义函数来简化代码，

首先我们要明确返回值得类型，因为结果我们需要一个 `int` 类型的值所以返回值定义为 `int` 类型，再来看一下参数，

由上图代码可以知道 `*3+5` 是不变的 `x` 的值是变化的需要我们指定一个参数，参数类型为 `int` 类型，返回值类型为

`int` 类型所以我们要 `return` 一个 `int` 类型的值代码如下：

```

1 public int getValue(int x) {
2     int a = x*3+5;
3     return a;
4 }
5 public static void main(String[] args) {
6     Demao1 d = new Demao1();
7     int i = d.getValue(5);
8     System.out.println(i);
9 }

```

提取方法后，当我们需要时调用方法只需要传入一个参数就可以得到我们想要的结果，简化了代码提高了代码的复用性。

函数的重载：

重载定义：函数名相同参数列表不同，与返回值类型无关。多态的一种

```

1 public int getValue(int x) {
2     int a = x*3+5;
3     return a;

```



```

4    }
5    public double getValue(double x) {
6        double a = x*3+5;
7        return a;
8    }
9    public int getValue(int x,int z) {
10       int a = x*z+5;
11       return a;
12    }

```

由上面代码可以看出方法名相同参数列表不同，调用方法时会根据传入的参数来调用相同的方法。

函数的重写：

重写定义：子类重写父类的方法，方法名参数列表都相同。

父类代码：

```

1 package com.jdbc.test;
2
3 public class ParentClass {
4     public int getValue(int a,int b){
5         return a+b;
6     }
7 }

```

子类代码：

```

1 package com.jdbc.test;
2
3
4 public class ChildrenClass extends ParentClass {
5     public int getValue(int a,int b){
6         return a*b;
7     }
8 }

```

由上面的代码可以看出子类继承了父类的方法，当我们调用子类的方法时会将父类的getValue（）方法覆盖掉。

2.4JAVA 基础复习——JAVA 语言的基础组成数组

JAVA 语言的基础组成有：

- 1.关键字：被赋予特殊含义的单词。
- 2.标识符：用来标识的符号。
- 3.注释：用来注释说明程序的文字。
- 4.常量和变量：内存存储区域的表示。
- 5.运算符：程序中用来运算的符号。
- 6.语句：程序中常用的一些语句。
- 7.函数：也叫做方法，用来做一些特定的动作。
- 8.数组：用来存储多个数据的集合。

JAVA 中的数组：


数组定义：

用来存储同一类型的容器。


数组定义语法：

- 1.类型[] 数组名称 = new 类型[长度];
- 2.类型[] 数组名称 = new 类型[]{value1,value2,.....};

示例：



```
1 //数组定义 1:类型[] 数组名称 = new 类型[长度];通过下标来赋值
2 int[] intArray = new int[5];
3 intArray[0]=2;
4 intArray[1]=20;
5 intArray[2]=1;
6 intArray[3]=45;
7 intArray[4]=3;
8 //数组定义 2: 2.类型[] 数组名称 = new 类型
  []{value1,value2,.....};直接进行赋值
9 int[] a = new int[] {5, 10, 20, 30, 1};
```



注：

定义数组时如果采用第一种定义数组方式应注意赋值不能超出指定的长度否则会报数组越界异常。

数组中的数据类型类型必须是一致的。

取数组中的元素：

数组中的元素我们可以通过下标来取出，需要注意的是数组的下标从 0 开始如：


```
1      int[] a = new int[] {5, 10, 20, 30, 1};
2      System.out.println(a[2]);
3      System.out.println(a[4]);
4      //a[2]结果为: 20
5      //a[4]结果为: 1;
```

获取数组的长度：


当我们想知道一个数组中有多少个数据的时候用.length 来获取数组的长度如：

```
1      int[] a = new int[] {5, 10, 20, 30, 1};
2      System.out.println(a.length);
3      //结果为: 5
```


获取数组中的最大最小值：



```
1      int[] a = new int[] {5, 10, 20, 30, 1};
2      int max = 0;
3      int min = a[0];
4      for (int i = 0; i < a.length; i++) {
5          if(max<a[i]) {
6              max=a[i];
7          }
8          if(min>a[i]) {
9              min=a[i];
10         }
11     }
12     //结果: max=30 min=1;
```



获取数组按从大到小或从小到大排序：



```
1      int[] a = new int[] {5, 10, 20, 30, 1};
2      for (int i = 0; i < a.length; i++) {
3          for (int j = 0; j < a.length-1; j++) { //控制比较次数
4              int x=0;
5              if(a[i]>a[j]) {
6                  x = a[i];
7                  a[i] = a[j];
8                  a[j]=x;
9              }
10         }
11     }
```



也可以通过 `Arrays.sort()` 方法来进行排序

3.JAVA 基础复习——JAVA 中的类与对象

什么是对象：

就是现实中真实的实体，对象与实体是一一对应的，现实中每一个实体都是一个对象在。

JAVA 中的对象：

Java 中通过 `new` 关键字来创建对象。

类：

用 JAVA 语言对现实生活中的事物进行描述，通过类的形式来体现，类是用来描述对象的

类描述对象通常只关注两个方面：

一个是属性，一个是行为。

如何定义一个类：

只要明确该事物的属性和行为并定义在类中即可。

类与对象之间的关系：

类用来描述对象的，对象是该类事物的实体。

类中的属性：

定义在类中的属性称为成员变量。

定义在方法中的属性称为局部变量。

区别：

成员变量定义在类中，作用范围为整个类，默认有初始值，存在于堆内存的对象中，随着 对象的创建而存在，对象的消失而消失。

局部变量定义在方法、语句、局部代码块中，只在所属的区域有效，默认没有初始值，存 在于栈内存的方法中，存着所属区域的执行而存在，随着所属区域的结束而消失。

匿名对象：

没有名字的对象，是定义对象的简写格式如：`new house()`；当只用到对象一次的时候可以用匿名对象，用到多次时不可以用匿名对象，

什么时候使用匿名对象：

1.当对象对方法仅做一次调用的时候，就可以简化成匿名对象。

如：`new house().getHouse()`；

2.匿名对象可以作为实际参数进行传递。

如：`show(new house())`；

参数传递有：

1.基本类型数据参数传递

如：`getValue(int x)`；

2.引用类型数据参数传递


如：`getPerson(Person p)`；


4. JAVA 基础复习——JAVA 中的构造函数与 this 关键字

构造函数：构建创建对象时调用的函数

特点：

1. 函数名与类名相同。
2. 不用定义返回值类型。
3. 没有具体的返回值。

```
public class Demo {  
    private int age;  
    private String name;  
    //省略 get、set 方法  
    public Demo() { //无参构造不写时系统将默认添加一个无参构造  
    }  
}
```



作用：

给对象初始化，创建对象时会先调用构造方法对对象进行初始化。

注意：

1. 默认构造函数的特点。不写构造函数时系统会自动给加一个默认的构造函数，如果自己定义了一个构造函数，系统不会再给添加一个默认的构造函数。
2. 多个构造函数是以重载的形式存在的。
3. 创建对象都必须要通过构造函数初始化。

```
1 public class Demo {  
2     private int age;  
3     private String name;  
4     //省略 get、set 方法  
5     public Demo() { //无参构造不写时系统将默认添加一个无参构造  
6     }  
7     public Demo(int age) { //有参构造多个构造函数以重载的形式存在  
8         this.age = age;  
9     }  
10    public Demo(String name) {  
11        this.name = name;  
12    }  
13    public Demo(String name, int age) {  
14        this(age);  
15        this.name = name;  
16    }  
17  
18    public static void main(String[] args) {
```

```
19      Demo d = new Demo(10); //创建对象时会先去找与之参数对应的构造函数进行初始化
20    }
```

一般函数和构造函数的区别：

构造函数：

对象创建时，就会调用与之对应的构造函数，对对象进行初始化。

对象创建时，会调用且只调用一次

一般函数：

对象创建后，需要函数功能时才调用。

对象创建后，可以调用多次。

什么时候定义构造函数：

在描述事物时，该事物一存在就具备一些内容，这些内容都定义在构造函数中。

this 关键字：

作用：

1.当成员变量和局部变量重名时，可以用 this 关键字来区分

```
1 public class Demo {
2     private int age; //成员变量
3     private String name;
4     public Demo(int age) { //参数为局部变量
5         this.age = age; //this.age 表示当前对象的 age 用来跟局部变量 age 区分
6     }
7 }
```

2.当构造函数需要调用已有的构造函数时可以有 this 关键字，需要注意的是需要将调用的构造函数放在第一行。

```
1 public class Demo {
2     private int age;
3     private String name;
4     public Demo(int age) {
5         this.age = age;
6     }
7     public Demo(String name, int age) {
```

```

8         this(age); //调用已有的构造函数，注意调用的构造函数要放在第
一行
9         this.name=name;
10    }
11 }

```

this:

代表当前对象。

是所在函数所属对象的引用。

简单来说：哪个对象调用了 this 所在的函数，this 就代表哪个对象。

5.JAVA 基础复习——JAVA 中的 static 关键字作用与用法

static 关键字：

特点：

- 1.static 是一个修饰符，用于修饰成员。（成员变量，成员函数）static 修饰的成员变量称之为静态变量或类变量。
- 2.static 修饰的成员被所有的对象共享。
- 3.static 优先于对象存在，因为 static 的成员随着类的加载就已经存在。
- 4.static 修饰的成员多了一种调用方式，可以直接被类名所调用，（类名.静态成员）。
- 5.static 修饰的数据是共享数据，对象中的存储的是特有的数据。

```

1    private static int age; //用 static 修饰的成员变量静态变量或者叫
做类变量
2    private String name;    //成员变量
3    public static void show() { //静态函数 可直接用类来调用
4        System.out.println("showStatic");
5    }
6    public void showDemo() { //成员函数 需要创建对象才可以调用
7        System.out.println("showDemo");
8    }

```

成员变量和静态变量的区别：

1.生命周期的不同：

成员变量随着对象的创建而存在随着对象的回收而释放。

静态变量随着类的加载而存在随着类的消失而消失。

2.调用方式不同：

成员变量只能被对象调用。

静态变量可以被对象调用，也可以用类名调用。（推荐用类名调用）

3.别名不同:

成员变量也称为实例变量。

静态变量称为类变量。

4.数据存储位置不同:

成员变量数据存储堆内存的对象中，所以也叫对象的特有数据。

静态变量数据存储方法区（共享数据区）的静态区，所以也叫对象的共享数据。

按 Ctrl+C 复制代码



按 Ctrl+C 复制代码

静态使用时需要注意的事项:

- 1.静态方法只能访问静态成员。（非静态既可以访问静态，又可以访问非静态）
- 2.静态方法中不可以使用 `this` 或者 `super` 关键字。
- 3.主函数是静态的。

```
1 public class Demo {
2     private static int age; //用 static 修饰的成员变量静态变量或者叫做类变量
3     private String name;    //成员变量也叫做实例变量
4
5     public static void show() { //静态函数
6         Demo d = new Demo(); //因为静态先于对象加载如果需要访问必须要创建对象才能访问
7         d.name = "张三"; //静态方法不能直接访问非静态的成员变量
8         d.showDemo(); //静态方法不能直接访问非静态的成员函数
9         System.out.println(d.name);
10    }
11    public void showDemo() { //成员函数
12        age = 10; //可以直接访问静态变量
13        show(); //也可以直接访问静态函数
14        System.out.println(age);
15    }
16 }
```

什么时候使用 `static` 来修饰

- 1.静态变量:

当分析对象中所具备的成员变量的值都是相同的。这时这个成员就可以被静态修饰。只要是数据在对象中都是不同的，就是对象的特有数据，必须存储在对象中，是非静态的。

如果是相同的数据，对象不需要做修改，只需要使用即可，不需要存储在对象中，是静态的。

2.静态函数。

函数是否用静态修饰，就参考一点，就是该函数功能是否有访问到对象中特有的数据。


简单来说，从源代码看，该功能是否需要访问非静态的成员变量，如果需要，该功能就是非静态的。如果不需要，就可以将该功能定义成静态的。当然，也可以定义成非静态，但是非静态需要被对象调用，而仅创建对象是没有意义的。

静态代码块：


随着类的调用或创建实例而执行，而且只执行一次。

作用：

用于给类进行初始化。



```
1 public class Demo { //如果想让此类成为一个静态类而类中需要用到一些
参数需要初始化就需要静态代码块
2     private static int age;
3     private static String name;
4     //省略 get、set 方法
5     static { //当类第一次调用或创建实例时给属性初始化且只执行一次。
6         age = 10;
7         name = "张三";
8     }
9     public static void showNoen() { //调用此方法 age 为 10 name 为张三
(默认值)
10         System.out.println("年龄: "+age+"姓名: "+name);
11     }
12     public static void show(int age, String name) { //调用此方法会覆
盖掉默认值
13         Demo.age = age;
14         Demo.name = name;
15         System.out.println("年龄: "+Demo.age+"姓名: "+Demo.name);
16     }
17     public static void main(String[] args) {
18         Demo.showNoen(); //结果为: 年龄: 10 姓名: 张三
19         Demo.show(50, "赵四"); //结果为: 年龄: 50 姓名: 赵四
20     }
21 }
```



主函数特殊之处：

1. 格式固定。
2. 被 jvm 所识别和调用。

```
public static void main(String[] args){  
  
}
```

public: 因为权限必须是最大的。

static: jvm 在调用时是不需要对象的，直接用主函数所属的类名调用即可。

void: 主函数没有具体的返回值。

main: 函数名，不是关键字，只是一个 jvm 识别的固定名字。

String[] args: 这是主函数的参数列表，是一个数组类型的参数，而且参数都是字符串类型

5.1 JAVA 基础复习——JAVA 中的静态代码块、构造代码块、构造函数、局部代码块区别

构造代码块：

在类中定义可以给所有对象进行初始化。

局部代码块：

在方法中定义属性的生命周期。

静态代码块：


在类中定义用于给类调用时属性的初始化

构造函数与构造代码块的区别：

构造函数是给对应的对象进行针对性的初始化。

构造代码块是给所有对象进行初始化

代码如下：



```
public class Demo {  
    private static int age;  
    private static String name;  
    //静态代码块：给类的属性进行初始化  
    static {  
        age = 10;  
        name = "旺财";  
    }  
    //构造代码块：给所有的对象初始化定义在类中方法外。  
    {  
        show();  
    }  
    public Demo(int age, String name) { //有参构造函数  
        Demo.age = age;  
    }  
}
```

```

        Demo.name = name;
    }
    public Demo() { //无参构造函数

    }
    public void show() { //普通函数
        //局部代码块：定义属性的生命周期
        {
            char sex = '男'; //变量 sex 出了 {} 后就会被销毁
            System.out.println("sex="+sex);
        }
        Demo.name = "张三";
    }
    public static void main(String[] args) {
        /*当有静态代码块构造代码块构造函数时程序执行流程
        * 1. 类被调用或创建实例时会执行静态代码块进行初始化执行后
age=10 name=旺财。
        * 2. 创建实例会执行构造代码块调用 show 方法执行后 sex = 男 ,
age 不变, name = 张三。
        * 3. 调用有参构造函数传入参数执行后 age = 50, name=姚明。
        * 执行顺序为:
        * 1、静态代码块 : 给类的属性进行初始化。
        * 2、构造代码块: 给所有的对象进行初始化。
        * 3、有参或无参构造 (根据实例来定): 给指定的对象进行初始
化。
        * 结果为: sex=男 注意这个是在构造代码块中输出的
        *          50    姚明
        * */
        Demo d = new Demo(50, "姚明");
        System.out.println(d.age+"\t"+d.name);
    }
}

```

6.JAVA 基础复习——JAVA 中文档注释与帮助文档的生成

java 中的文档注释：用于说明该类的功能作用方便他人使用

关键词前需要加@符

用于类的注释

@author name 作者

@version v1.0 版本

.....

用于函数的注释

@param parameter 参数

@return value 返回值

.....

首先要给类加上帮助文档注释/** */用于类、函数的说明



```
1 package com.jdbc.test;
2
3 /**
4  * 数组工具类,用于数组的一些常用方法
5  * @author 张三
6  * @version v1.0
7  */
8 public class ArrayTool {
9
10     private ArrayTool() {
11
12     }
13
14     /**
15      * 获取数组的长度
16      * @param arr 传入一个 int 类型的数组
17      * @return 返回数组长度
18      */
19     public static int getLength(int[] arr) {
20         return arr.length;
21     }
22
23     /**
24      * 获取数组中最大的值
25      * @param arr 传入一个 int 类型的数组
26      * @return 返回最大值
27      */
28     public static int getMax(int[] arr) {
29         int max = arr[0];
30         for (int i = 1; i < arr.length; i++) {
31             if(max<arr[i]){
32                 max = arr[i];
33             }
34         }
35         return max;
36     }
37     /**
38      * 数组降序排序函数
```

```

39     * @param arr 传入一个 int 类型的数组
40     */
41     public static void arraySort(int[] arr) {
42         for (int i = 0; i < arr.length; i++) {
43             for (int j = 0; j < arr.length-1; j++) {
44                 compare(arr, i, j);
45             }
46         }
47     }
48     /**
49     * 查看数组中的元素
50     * @param arr 传入一个 int 类型的数组
51     * @return 返回数组中的所有元素
52     */
53     public static String selectArray(int[] arr) {
54         String str = "[";
55         for (int i = 0; i < arr.length; i++) {
56             str+=arr[i]+",";
57         }
58         return str+"]";
59     }
60     /**
61     * 根据数组下标查找元素
62     * @param arr 传入一个 int 类型的数组
63     * @param num 数组下标
64     * @return 查找下标的值
65     */
66     public static int getIndex(int[] arr,int num) {
67         int value = -1;
68         for (int i = 0; i < arr.length; i++) {
69             if (arr[i]==arr[num]) {
70                 return value = arr[i];
71             }
72         }
73         return value;
74     }
75
76     /**
77     * 获取数组的降序比较方法
78     * @param arr 传入一个 int 类型的数组
79     */
80     private static void compare(int[] arr, int i, int j) {
81         if(arr[i]>arr[j]){
82             int temp = arr[i];

```

```
83         arr[i] = arr[j];
84         arr[j] = temp;
85     }
86 }
87
88 }
```

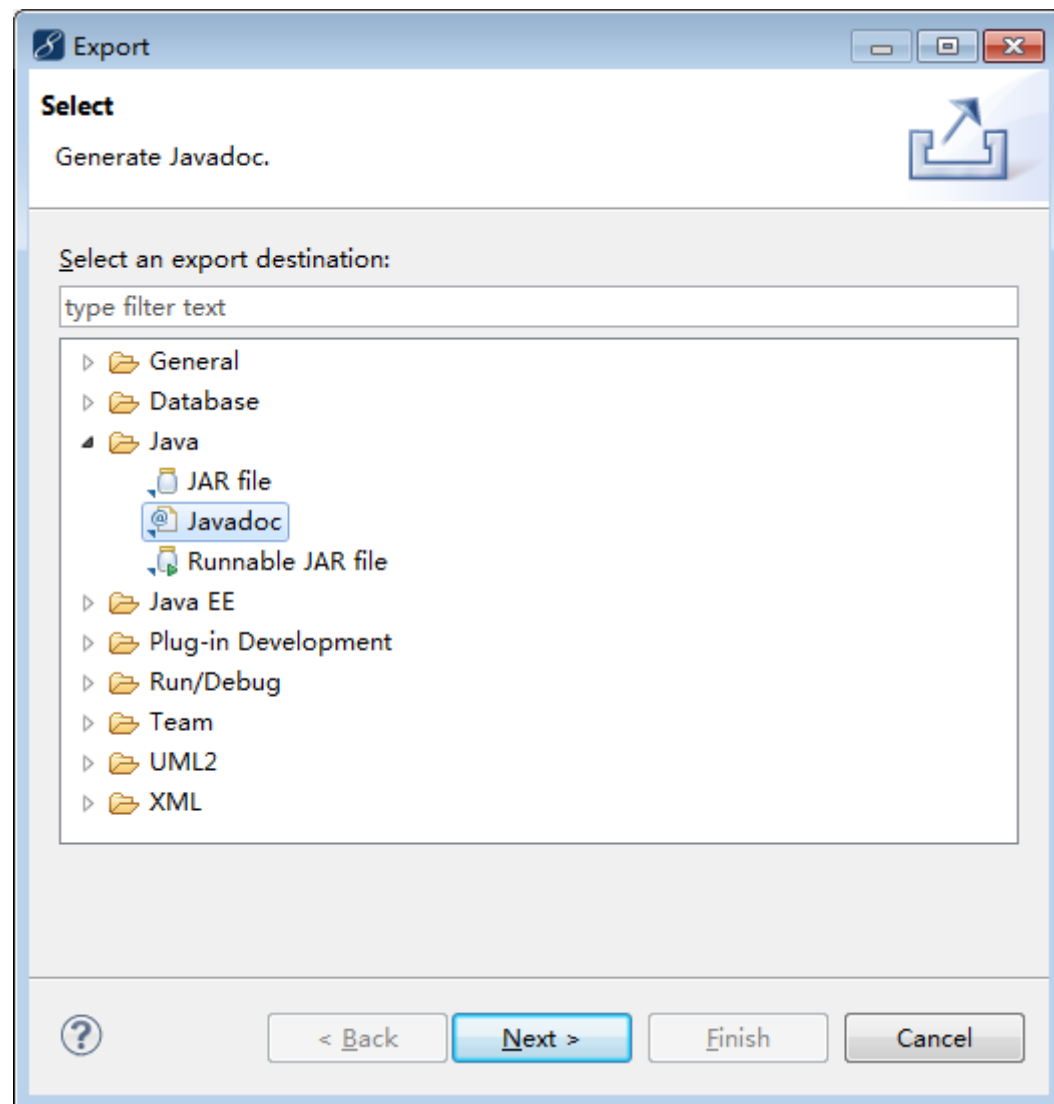


在不想让人家看见源码而又能使用这个类的时候我们需要对此类进行封装并生成帮助文档方便使用

当别人使用时只需要编译好的.class 文件和帮助文档就能够使用其中的功能。

使用 myeclipse 生成帮助文档：

- 1.在项目或类上右键选择 Export...
- 2.选择 java 中的 javadoc 点击 next

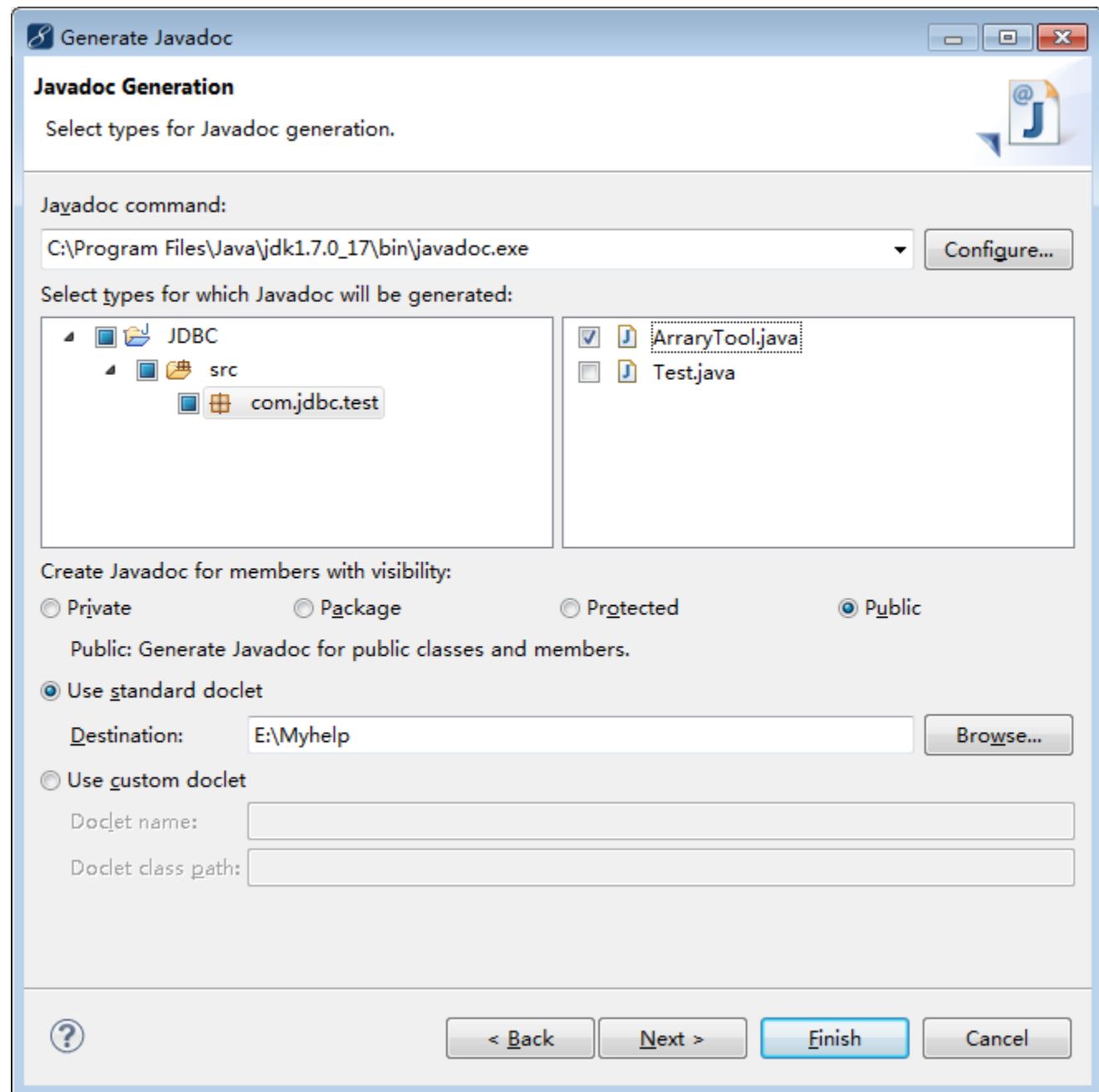


3.Javadoc command: 选择已安装 jdk 路径下的 bin\javadoc.exe

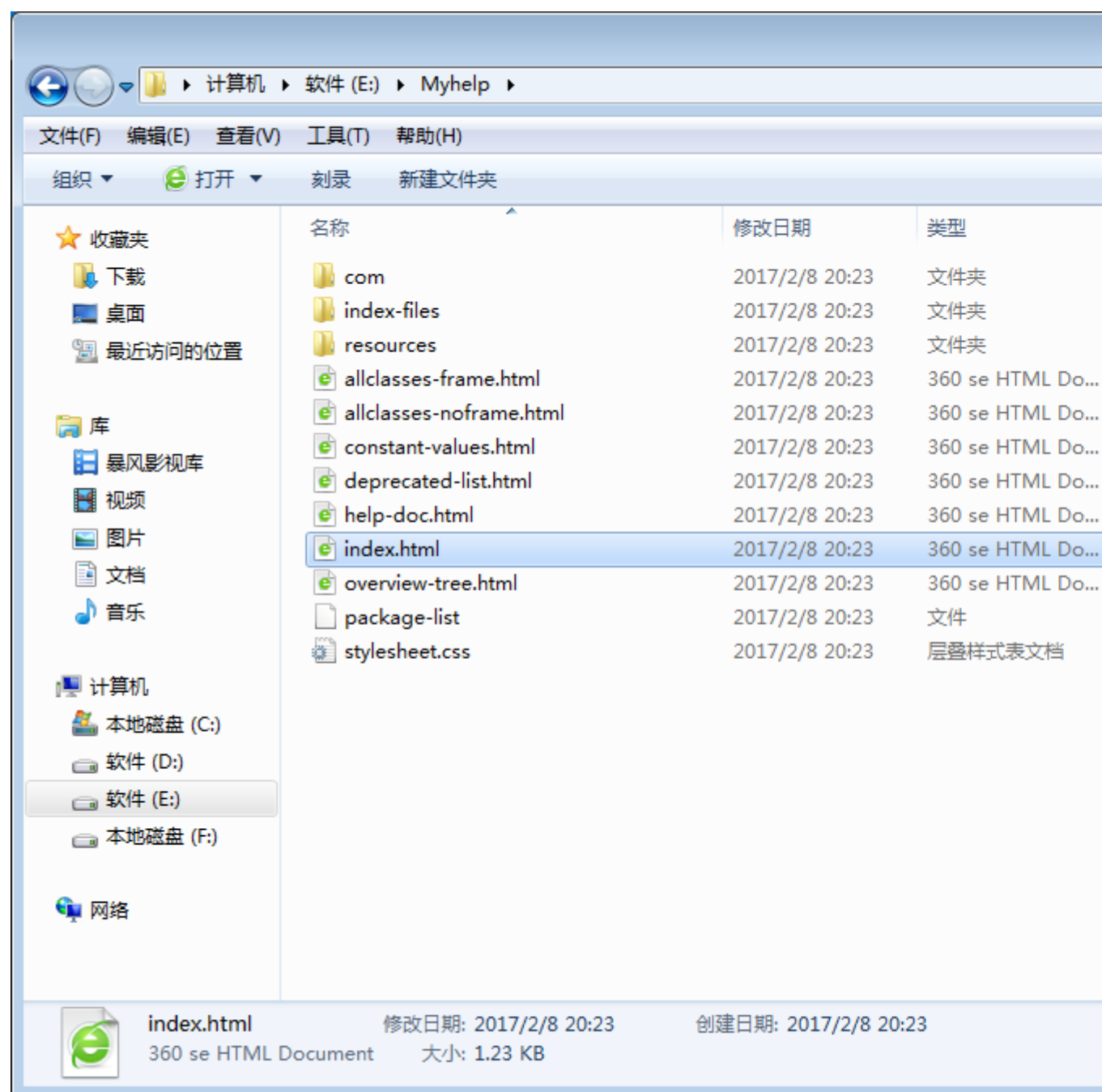
Select types fro which javadoc will be generated: 选择要生成帮助文档的类

Destination: 生成帮助文档的路径

点击 finish



4.到帮助文档路径下找到 index.html 打开



5.打开后的首页



6. 点击 ArrayTool，就能查看到这个类中的方法及用法

360安全浏览器 7.1

[←](#) [↻](#) [🏠](#) [🛡️ file:///E:/Myhelp/index.html](#)

★ 收藏 ▾

[🐶 百度](#) [🐶 淘宝](#) [📄 京东商城](#) [🐶 天猫精选](#) [🐶 9.9包邮](#) [🎮 手机游戏](#) [👉 美女图片](#) [👉 女性网](#) [🎮 小游戏](#)

I>

[📄 博客后台管理 - 博客园](#) ×

[📄 生成的文档 \(无标题\)](#) ×

+

所有类

ArrayTool

[上一个类](#) [下一个类](#) [框架](#) [无框架](#)

[概要: 嵌套](#) | [字段](#) | [构造器](#) | [方法](#) [详细资料: 字段](#) | [构造器](#) | [方法](#)

com.jdbc.test

类 **ArrayTool**

java.lang.Object
com.jdbc.test.ArrayTool

```
public class ArrayTool
extends java.lang.Object
```

数组工具类,用于数组的一些常用方法

版本:

v1.0

作者:

张三

方法概要

方法

限定符和类型	方
static void	an 数
static int	ge 根
static int	ge 获
static int	ge 获
static java.lang.String	se 查

7.JAVA 基础复习——JAVA 中的设计模式单例模式

设计模式：是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。

总体来说设计模式分为 23 种三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

单例模式：类在内存中只有一个对象实例。

单例模式有两种实现方式：

饿汉式：类加载时创建对象

懒汉式：当需要实例对象时才创建对象(延迟加载)

代码：

```
1 public class Single {
2     /*单例模式：类在内存中只有一个对象实例。有两种方式实现
3     *实现细节：
4     *    1. 不让创建对象由我们来创建对象（私有化构造：不可创建对
象。私有化实例：不可访问此对象。）
5     *    2. 定义一个方法用来获取实例对象。
6     *    4. 静态变量会在类加载时执行所以堆内存中存在 Single() 对
象。
7     *    5. 程序会先从栈中找如果没有去堆中寻找对象。
8     *    6. 堆内存中有此对象所以会返回此对象。
9     *    7. 第二次调用时同理会返回此对象所以指向的是同一个对象
10    */
11    //单例模式饿汉式：类加载时就创建实例对象
12    private static Single s = new Single();
13    public static Single getSingle() {
14        return s;
15    }
16    //私有化构造不让创建对象
17    private Single() {}
18
19    /*单例模式：类在内存中只有一个对象实例。
20    *实现细节：
```

```

21      *    1. 不让别创建对象由我们来创建对象（私有化构造：不可创建对
    象。私有化实例：不可访问此对象。）
22      *    2. 定义一个方法用来获取实例对象。
23      *    4. 静态变量会在类加载时执行此处为空没由创建实例对象
24      *    5. 程序会先从栈中找如果没有去堆中寻找对象。
25      *    6. 堆内存中没有此对象所以会先创建实例对象然后返回
26      *    7. 当第二次调用时因为堆内存中有了此对象所以会直接返回所以
    都是指向同一个实例对象。
27      */
28      //单例模式懒汉式：类加载时不创建对象，用到实例时才加载。
29      private static Single s = null;
30      public static Single getSingle() {
31          if(s==null) {
32              s = new Single();
33          }
34          return s;
35      }
36 }

```

java 面试题——HashMap 和 Hashtable 的区别

一.HashMap 和 Hashtable 的区别

我们先看 2 个类的定义

```


1 public class Hashtable
2     extends Dictionary
3     implements Map, Cloneable, java.io.Serializable
public class HashMap
    extends AbstractMap
    implements Map, Cloneable, Serializable

```

可见 Hashtable 继承自 Dictionary 而 HashMap 继承自 AbstractMap

Hashtable 的 put 方法如下

```


public synchronized V put(K key, V value) { //##### 注意这里 1
    // Make sure the value is not null
    if (value == null) { //##### 注意这里 2
        throw new NullPointerException();
    }
    // Makes sure the key is not already in the hashtable.
    Entry tab[] = table;
    int hash = key.hashCode(); //##### 注意这里 3
    int index = (hash & 0x7FFFFFFF) % tab.length;
    for (Entry e = tab[index]; e != null; e = e.next) {

```

```

        if ((e.hash == hash) && e.key.equals(key)) {
            V old = e.value;
            e.value = value;
            return old;
        }
    }
    modCount++;
    if (count >= threshold) {
        // Rehash the table if the threshold is exceeded
        rehash();
        tab = table;
        index = (hash & 0x7FFFFFFF) % tab.length;
    }
    // Creates the new entry.
    Entry e = tab[index];
    tab[index] = new Entry(hash, key, value, e);
    count++;
    return null;
}

```

注意 1 方法是同步的

注意 2 方法不允许 value==null

注意 3 方法调用了 key 的 hashCode 方法, 如果 key==null, 会抛出空指针异常 HashMap 的 put 方法如下

```

public V put(K key, V value) { //##### 注意这里 1
    if (key == null) //##### 注意这里 2
        return putForNullKey(value);
    int hash = hash(key.hashCode());
    int i = indexFor(hash, table.length);
    for (Entry e = table[i]; e != null; e = e.next) {
        Object k;
        if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {
            V oldValue = e.value;
            e.value = value;
            e.recordAccess(this);
            return oldValue;
        }
    }
    modCount++;
    addEntry(hash, key, value, i); //##### 注意这里
    return null;
}

```



注意 1 方法是非同步的

注意 2 方法允许 `key==null`

注意 3 方法并没有对 `value` 进行任何调用，所以允许为 `null`

补充：

`Hashtable` 有一个 `contains` 方法，容易引起误会，所以在 `HashMap` 里面已经去掉了
当然，2 个类都用 `containsKey` 和 `containsValue` 方法。

	HashMap	Hashtable
父类	AbstractMap	Dictionary
是否同步	否	是
k, v 可否 null	是	否

`HashMap` 是 `Hashtable` 的轻量级实现（非线程安全的实现），他们都完成了 `Map` 接口，

主要区别在于 `HashMap` 允许空（`null`）键值（`key`），由于非线程安全，效率上可能高于 `Hashtable`。

`HashMap` 允许将 `null` 作为一个 `entry` 的 `key` 或者 `value`，而 `Hashtable` 不允许。

`HashMap` 把 `Hashtable` 的 `contains` 方法去掉了，改成 `containsvalue` 和 `containsKey`。因为 `contains` 方法容易让人引起误解。

`Hashtable` 继承自 `Dictionary` 类，而 `HashMap` 是 `Java1.2` 引进的 `Map interface` 的一个实现。

最大的不同是，Hashtable 的方法是 Synchronize 的，而 HashMap 不是，在多个线程访问 Hashtable 时，不需要自己为它的方法实现同步，而 HashMap 就必须为之提供外同步 (Collections.synchronizedMap)。

Hashtable 和 HashMap 采用的 hash/rehash 算法都大概一样，所以性能不会有很大的差异。

总结：

HashMap 中键值 允许为空 并且是非同步的

Hashtable 中键值 不允许为空 是同步的

继承不同，但都实现了 **Map** 接口