菜鸟教程：<https://www.runoob.com/java/java-basic-datatypes.html>

基础类型【6种数字类型+**boolean**+char】：核心是数字类型根据范围来使用，因为byte占用内存空间小，如果确定数字不大，就可以用byte或者short；一般用int；较小的用short或byte，特别大的用long【可以用8进制，10进制，16进制的方式来表示，一把用十进制数字】

前缀 **0** 表示 8 进制，而前缀 **0x** 代表 16 进制

Byte:8位 ：【**-128--127**】 100级别

Short：16位 【**-32768--32767**】3万级别

Int：32位【**-2,147,483,648--** **2,147,483,647**】：20亿级别

Long：64位【64位】：922万兆

**Float：**float 数据类型是单精度、32位、符合IEEE 754标准的；浮点数浮点数不能用来表示精确的值，如货币；

**Double：**是双精度、64 位、符合 IEEE 754 标准的浮点数；也不能用来表示精确的值，如货币

Char：单个字符；

String:字符串

引用类型:对象，数组，默认是null

数据转换：byte,short,char—> int —> long—> float —> double

高精度转低精度，必须强制转换: int a = 123; byte b = (byte)a;//强制类型转换为byte

浮点数到整数的转换是通过舍弃小数得到，而不是四舍五入，

不能对boolean类型进行类型转换。

整数的默认类型是 int; 小数默认是 double 类型浮点型!!!!!!!!!!!!!!!!!!!!

* 局部变量：局部变量是在栈上分配的。

堆和栈区别：

申请方式的不同。栈由系统自动分配，而堆是人为申请开辟;

申请大小的不同。栈获得的空间较小，而堆获得的空间较大

申请效率的不同。栈由系统自动分配，速度较快，而堆一般速度比较慢

存储内容的不同。栈在函数调用时，函数调用语句的下一条可执行语句的地址第一个进栈，然后函数的各个参数进栈，其中静态变量是不入栈的。而堆一般是在头部用一个字节存放堆的大小，堆中的具体内容是人为安排;

底层不同。栈是连续的空间（！！！！！），而堆是不连续的空间（！！！！！）

**Java的堆是一个运行时数据区，类的对象从堆中分配空间**

**栈中主要存放一些基本数据类型的变量（byte，short，int，long，float，double，boolean，char）和对象的引用**

栈的优势是，存取速度比堆快，栈数据可以共享。但缺点是，存放在栈中的数据占用多少内存空间需要在编译时确定下来

说白了：堆用来放引用类型数据【包括引用类型的变量】；栈内存用来存储局部变量和方法调用

局部变量没有默认值，所以局部变量被声明后，必须经过初始化

变量的销毁：java垃圾是自动回收

实例变量是对象创建的时候产生，对象销毁的时候销毁

局部变量是函数执行的时候内存自动分配，函数执行完就销毁

实例变量：默认可以被内部的方法访问，构造函数，语句块等访问

final修饰类 当final关键字修饰一个类,则该类会成为最终类,即该类不能被继承

被final修饰的成员变量，一定要被赋值且只能被赋值一次，且必须是在这个成员变量所在的类对象创建之前被赋值

**Final的局部变量，该变量必须在使用之前赋值，且只能被赋值一次！**

修饰符：对于“包，子类，类的访问权限”public全部都可以访问，default是包内可访问；protected是子类和当前类可访问；private是当前类可访问

**public** : 对所有类可见。使用对象：类、接口、变量、方法

**default** (即默认，什么也不写）: 在同一包【package】内可见，不使用任何修饰符。使用对象：类、接口、变量、方法。

**protected** : 当前类和基础当前类的子类可访问。使用对象：变量、方法。 **注意：不能修饰类（外部类）**

**private** : 在同一类内可见。使用对象：变量、方法。 **注意：不能修饰类（外部类）**

final 表示"最后的、最终的"含义，变量一旦赋值后，不能被重新赋值。被 final 修饰的实例变量必须显式指定初始值。

final 修饰符通常和 static 修饰符一起使用来创建类常量。

synchronized 关键字声明的方法同一时间只能被一个线程访问。synchronized 修饰符可以应用于四个访问修饰符

公共方法：

其他计算语法：和js相同

Math：和js类似

字符串创建：String str=“ssdfds”或者String str=new String（“ssdfds”）【用构造函数创建的是一个String对象，返回的是一个堆里面的引用（创建两次地址不同）；非构造函数创建的是栈内的变量，是公共池子内的变量，都是同一个地址，所以非构造函数创建两个相同的字符串，是相同的地址，是相等的】

str.length();//要添加括号，和js不同，其他js中有的字符串函数，java中基本都有

java数组 创建：

double[] myList = new double[10];//10表示数组的长度，可以用变量代替10；java数组内所有数据必须相同类型

double[] myList = {1.9, 2.9, 3.4, 3.5};这样也可以直接定义数组，这里要特别注意，是用括号来初始化数组；不是[]

数组遍历：for (double element: myList) {element是数组内的每一个元素，}；或者for循环遍历

多维数组：int[][] a = new int[2][3];

Java的数组方法较少，主要是concat，sort，binarySearch

Date：基本不用，前端做

正则：基本不用，前端做

Java的常用方法：

**System.out.println()**

**//java的重写和重载是java的多态的表现**

**方法的重载：就是函数名称相同，参数格式不同**

**方法的重写：就是子类里面重写父类的方法，参数格式必须和父类相同，而且有很多其他限制，详见**<https://www.runoob.com/java/java-override-overload.html>

当使用多态方式调用方法时，首先检查父类中是否有该方法，如果没有，则编译错误；如果有，再去调用子类的同名方法。！！！！！！！！！！！！！！！！！！

**作用域：java有块级作用域，也就是for ，switch，if等语句，里面是块级作用域，外面是无法访问里面的变量的**

**命令行参数的使用：https://www.runoob.com/java/java-methods.html**

Java手动内存清理【基本用不上，内存泄漏也是自己java写得有问题】：System.gc(); //调用Java垃圾收集器；同时叠加finalize()方法，详见https://www.runoob.com/java/java-methods.html

控制台的输入输出和java的数据流【分字符流和字节流】：https://www.runoob.com/java/java-files-io.html

IO处理：Java.io 包几乎包含了所有操作输入、输出需要的类

一个流可以理解为一个数据的序列。输入流表示从一个源读取数据，输出流表示向一个目标写数据

FileInputStream（从文件读取数据）；FileOutputStream（创建一个文件并向文件中写数据）

* [File Class(类)](https://www.runoob.com/java/java-file.html)
* [FileReader Class(类)](https://www.runoob.com/java/java-filereader.html)
* [FileWriter Class(类)](https://www.runoob.com/java/java-filewriter.html)

Java异常：

* 用户输入了非法数据。
* 要打开的文件不存在。
* 网络通信时连接中断，或者JVM内存溢出。
* try
* {
* // 程序代码
* }catch(ExceptionName e1)
* {
* //Catch 块
* }

Java继承：不支持多继承，就是有a和b两个父类，c不能同时继承a和继承b

**引用数据类型：数组，类，接口**

抽象类+抽象方法：<https://www.runoob.com/java/java-abstraction.html> 一个不完整的类，不能直接创建实例，只能拖过子类的继承，然后子类区创建实例

接口：**在Java中，接口可以看成是：多个类的公共规范，是一种引用数据类型，接口只是定义一个规范，用到哪些方法，至于方法的实现，都是各自的类有各自的实现；** 接口只是一个行为的规范或规定

接口使用案例 <https://www.cnblogs.com/yfc0818/p/11072600.html> 【这个描述最清晰，最容易理解接口】

接口有利于实现数据相互隔离：<https://blog.csdn.net/curry_for_3/article/details/114699992>

A类的对象a的实例方法funcA内想要调用B类的实例b的方法funcB；一般是在funcA内传入b对象，执行b. funcB；

这样会导致funcA内可以访问b对象的其他属性【不安全，业务也会耦合】；如果给funcA传入b对象实现的接口C；在funcA中通过接口C一样可以调用b对象的funcB方法，实现了数据安全以及解耦

<https://www.runoob.com/java/java-interfaces.html> ：其实就是处理特定的业务逻辑的函数集合，和实例无关，只是处理输入输出的逻辑；

接口是抽象方法的集合

接口不是被类继承了，而是被类实现。！！！！

接口支持多继承。

接口中的方法是不能在接口中实现的，只能由实现接口的类来实现接口中的方法

接口中的成员变量只能是 **public static final** 类型的

一个类只能继承一个其他类，但是却可以实现多个接口【也就是这个类可以集成各种各样的接口，从而使用这些接口里面的功能函数】

**接口不能创建对象，但是可以被实现（ implements ，类似于被继承）。【如果没有修饰词默认是abstract 抽象类型】**

**类与接口的关系为实现关系，即类实现接口，该类可以称为接口的实现类，也可以称为接口的子类，直接就理解成，接口继承了类里面的方法，也就是各个把类里面的方法用接口拿出来让其他模块使用【不需要之歌各个类对应相同的接口函数是如何实现的，各自有各自的实现，只要知道它们做了这件事情】！！！！**

**接口，是 Java 语言中一种引用类型，是方法的集合，如果说类的内部封装了成员变量、构造方法和成员方法，那么接口的内部主要就是封装了方法，包含抽象方法( JDK 7及以前），默认方法和静态方法（ JDK 8)，私有方法（ JDK 9)。**

接口变量必须引用实现了接口的类对象 ！！！！！

有一条实际经验：在合理的范围内尽可能地抽象。显然，接口比抽象类还要抽象。因此，一般更倾向使用接口而不是抽象类

Java的泛型！！！！！！！！：<inter,double,…>表示泛型，就是以一个变量的类型不确定，可能是几中类型里面的一种，就可以用泛型；而js默认就支持泛型【不用声明类型】

# 枚举：Java 枚举(enum)

Java 枚举是一个特殊的类，一般表示一组常量

enum Color

{

RED, GREEN, BLUE;

}

**public** **class** Test  
{  
    *// 执行输出结果*  
    **public** **static** **void** main(String[] args)  
    {  
        Color c1 = Color.RED;//这样使用  
        System.out.println(c1);  
    }  
}

# Java 包(package)

为了更好地组织类，Java 提供了包机制，用于区别类名的命名空间：把不同的 java 程序分类保存，类似功能放到一个包里面，更方便的被其他 java 程序调用。一个包（package）可以定义为一组相互联系的类型（类、接口、枚举和注释）

* 1、把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用。
* 2、如同文件夹一样，包也采用了树形目录的存储方式。同一个包中的类名字是不同的，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。因此，包可以避免名字冲突。
* 3、包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类。

包声明应该在源文件的第一行，每个源文件只能有一个包声明，这个文件中的每个类型都应用于它。如果一个源文件中没有使用包声明，那么其中的类，函数，枚举，注释等将被放在一个无名的包（unnamed package）中。

这个例子创建了一个叫做animals的包。通常使用小写的字母来命名避免与类、接口名字的冲突。

## Animal.java 文件代码如下：第一行说明这个文件属于哪个包

/\* 文件名: Animal.java \*/

package animals;

interface Animal { public void eat(); public void travel(); }

**MammalInt.java 文件代码如下：**

package animals;

/\* 文件名 : MammalInt.java \*/

public class MammalInt implements Animal{ public void eat(){ System.out.println("Mammal eats"); } public void travel(){ System.out.println("Mammal travels"); } public int noOfLegs(){ return 0; } public static void main(String args[]){ MammalInt m = new MammalInt(); m.eat(); m.travel(); } }

在其他包里面想要用animals包里面的Animal接口，就需要通过import animals.Animal 来导入或者import payroll.\*;导入所有 ；import 声明必须在包声明之后，类声明之前

通常，一个公司使用它互联网域名的颠倒形式来作为它的包名.例如：互联网域名是 runoob.com，所有的包名都以 com.runoob 开头。包名中的每一个部分对应一个子目录。

有一个 **com.runoob.test** 的包，这个包包含一个叫做 Runoob.java 的源文件，那么相应的，应该有如下面的一连串子目录：

....\com\runoob\test\Runoob.java

编译的时候，编译器为包中定义的每个类、接口等类型各创建一个不同的输出文件，输出文件的名字就是这个类型的名字，并加上 .class 作为扩展后缀。 例如：

// 文件名: Runoob.java

package com.runoob.test;

public class Runoob { }

class Google { }

类目录的绝对路径叫做 **class path**。设置在系统变量 **CLASSPATH** 中

编译器和 java 虚拟机通过将 package 名字加到 class path 后来构造 .class 文件的路径。

<path- two>\classes 是 class path，package 名字是 com.runoob.test,而编译器和 JVM 会在 <path-two>\classes\com\runoob\test 中找 .class 文件。

一个 class path 可能会包含好几个路径，多路径应该用分隔符分开。默认情况下，编译器和 JVM 查找当前目录。JAR 文件按包含 Java 平台相关的类，所以他们的目录默认放在了 class path 中。

菜鸟教程里面的java高级教程需要看下

//最最核心：https://www.runoob.com/java/java-collections.html

就是java.util包，里面有各种各样的类和接口，就类似于js的各种工具库，实现各种功能

Java Object 类是所有类的父类，也就是说 Java 的所有类都继承了 Object，**子类可以使用 Object 的所有方法**。