想了解全部内容请移步至

|  |  |
| --- | --- |
| Java SE 8 | <https://docs.oracle.com/javase/8/index.html>  <https://docs.oracle.com/javase/8/docs/> |
| Java SE 9 | <https://docs.oracle.com/javase/9/index.html> |

|  |  |
| --- | --- |
| Java SE 5 | <http://doc.d8jd.com/down/164.html> |
| Java SE 8 | <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-doc-downloads-2133158.html> |
| Java SE 9 | <http://www.oracle.com/technetwork/java/javase/downloads/jdk9-doc-downloads-3850606.html> |

<https://javadoc.allimant.org/>

更新地址：<https://github.com/jhcarl0814/notes>

若发现文档内容与实际不符，请向GitHub上的作者反馈。

红色的是正则表达式算子<https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F>

蓝色的单元格是未完成（目前无法完成）的内容。

源代码文件结构

|  |
| --- |
| 所属包的声明  package (**包名**.)\***包名**;  1、若文件没有所属包的声明，则文件属于default package。 |
| 引入类的声明  import (**包名**.)\***包名**.(**类名**|\*);  1、不能引入default package中的类。  2、编译器自动引入java.lang包。  引入类的静态成员的声明  import static (**包名**.)\***包名**.**类名**.(**成员名**|\*);  3、不能引入default package中的类的静态成员。  4、导入的标识符相当于全局变量，与类成员重名时（即使类成员不是static的也）被类成员名覆盖。  5、若有多个单一导入重名且标识符被使用，错误；若有多个全体导入重名且没有单一导入且标识符被使用，错误。 |
| 类和接口的声明  1、1个文件至多含有1个public类。若含有public类，文件名与public类的类名相同。 |

# 运算符

同级运算符执行顺序：从左向右。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 优先级 |  | 用法 | 计算结果 | 副作用 |  |
| 1 | . |  |  |  |  |
| [] |  |  |  |  |
| () |  |  |  |  |
| 2 | ++ |  |  |  |  |
| -- |  |  |  |  |
| ! |  |  |  |  |
| ~ |  |  |  |  |
| instanceof |  |  |  |  |
| 3 | new |  |  |  |  |
| 4 | \* |  |  |  |  |
| / |  |  |  |  |
| % |  |  |  |  |
| 5 | + |  |  |  |  |
| - |  |  |  |  |
| 6 | >> |  |  |  |  |
| >>> |  |  |  |  |
| << |  |  |  |  |
| 7 | < |  |  |  |  |
| > |  |  |  |  |
| <= |  |  |  |  |
| >= |  |  |  |  |
| 8 | == |  |  |  |  |
| != |  |  |  |  |
| 9 | & |  |  |  |  |
| 10 | ^ |  |  |  |  |
| 11 | | |  |  |  |  |
| 12 | && |  |  |  |  |
| 13 | || |  |  |  |  |
| 14 | ?: |  |  |  |  |
| 15 | = |  |  |  |  |
| += |  |  |  | 1、相当于**表达式1**=**表达式1** **运算符** **表达式2**。 |
| -= |  |  |  |
| \*= |  |  |  |
| /= |  |  |  |
| %= |  |  |  |
| ^= |  |  |  |
| 16 | &= |  |  |  |
| |= |  |  |  |
| <<= |  |  |  |
| >>= |  |  |  |
| >>>= |  |  |  |

# 标识符、关键字

变量作用域

|  |  |  |
| --- | --- | --- |
| 类成员变量 | 在类的声明体中声明。 | 整个类。 |
| 局部变量 | 在方法体或方法的代码块中声明。 | 从声明到当前代码块结束 |

标识符声明

1、标识符由大写字母、小写字母、下划线、$、数字组成，第1个字符是大写字母、小写字母、下划线、或$。

2、标识符区分大小写。

变量访问限制

|  |  |
| --- | --- |
| final | 声明的不再是变量而是常量。 |

保留关键字：const、goto等。

关键字

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| import  package | static | final | |  | | --- | | void | | boolean | | byte  short  int  long | | float  double | | char | | |  | | --- | | class  public  protected  private  this  super  extends  abstract  native  synchronized  transient  volatile  strictfp | | interface  implements |   enum | |  | | --- | | if  else  switch  case  default | | while  for  do | | break  continue  return | | try  catch  finally  throw  throws | | instanceof  new | assert | true  false  null |

# 数据

## 基本数据类型

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | 字节数 | | byte | 1 | | short | 2 | | int | 4 | | long | 8 | | float | 4 | | double | 8 | | boolean |  | | char | 2 |   void用于表示函数不接收参数、函数不返回值。 | |
| 变量的声明  (final )?**基本数据类型名** **变量名**(=**表达式**)?(,  **变量名**(=**表达式**)?)\*; | 1、不能声明void数据类型的变量。  2、变量的数据类型是**基本数据类型**。 |
| 布尔  可能取值：true、false。 | |
| 字符  转义字符   |  |  |  | | --- | --- | --- | | \' | 单引号 | 在字符串常量中可以直接写' | | \" | 双引号 | 在字符常量中可以直接写" | | \\ | 反斜杠 |  | | \r | 回车 |  | | \n | 回车/换行 |  | | \f | 进纸 |  | | \t | 水平制表符 |  | | \b | 退格 |  | | \u**hhhh** |  | **h**表示1位十六进制数字 | | |
| 整数   |  |  | | --- | --- | |  | 前缀 | | 二进制常量 | 0b或0B | | 八进制常量 | 0 | | 十六进制常量 | 0x或0X |  |  |  | | --- | --- | |  | 后缀 | | long常量 | l或L | | |
| 浮点数   |  |  | | --- | --- | |  | 后缀 | | float常量 | f或F | | double常量 | 无或d或D | | |
| 数据类型转换  所有数据类型都可以向字符串数据类型转换。  扩展转换、窄化转换  赋值转换、方法调用转换、强制数据类型转换、字符串转换   |  |  | | --- | --- | | char→int | int→long→float→double | | byte→short→int |   不可达的需要强制数据类型转换。  boolean无法和其它基本数据类型转换。   |  | | --- | | public static void main(String[] args) {  boolean v\_boolean=false;  byte v\_byte=0;  char v\_char=0;  short v\_short=0;  int v\_int=0;  long v\_long=0;  float v\_float=0;  double v\_double=0;    v\_boolean=(boolean)v\_char;错误  v\_boolean=(boolean)v\_byte;错误  v\_boolean=(boolean)v\_short;错误  v\_boolean=(boolean)v\_int;错误  v\_boolean=(boolean)v\_long;错误  v\_boolean=(boolean)v\_float;错误  v\_boolean=(boolean)v\_double;错误    v\_byte=(byte)v\_boolean;错误  v\_byte=(byte)v\_char;  v\_byte=(byte)v\_short;  v\_byte=(byte)v\_int;  v\_byte=(byte)v\_long;  v\_byte=(byte)v\_float;  v\_byte=(byte)v\_double;    v\_char=(char)v\_boolean;错误  v\_char=(char)v\_byte;！！！  v\_char=(char)v\_short;  v\_char=(char)v\_int;  v\_char=(char)v\_long;  v\_char=(char)v\_float;  v\_char=(char)v\_double;    v\_short=(short)v\_boolean;错误  v\_short=(short)v\_char;！！！  v\_short=v\_byte;  v\_short=(short)v\_int;  v\_short=(short)v\_long;  v\_short=(short)v\_float;  v\_short=(short)v\_double;    v\_int=(int)v\_boolean;错误  v\_int=v\_char;  v\_int=v\_byte;  v\_int=v\_short;  v\_int=(int)v\_long;  v\_int=(int)v\_float;  v\_int=(int)v\_double;  v\_long=(long)v\_boolean;错误  v\_long=v\_char;  v\_long=v\_byte;  v\_long=v\_short;  v\_long=v\_int;  v\_long=(long)v\_float;  v\_long=(long)v\_double;    v\_float=(float)v\_boolean;错误  v\_float=v\_char;  v\_float=v\_byte;  v\_float=v\_short;  v\_float=v\_int;  v\_float=v\_long;  v\_float=(int)v\_double;  v\_double=(float)v\_boolean;错误  v\_double=v\_char;  v\_double=v\_byte;  v\_double=v\_short;  v\_double=v\_int;  v\_double=v\_long;  v\_double=v\_float;  } | | |
| 装箱拆箱   |  |  |  |  | | --- | --- | --- | --- | |  |  | 构造函数 | 方法 | | byte | Byte |  |  | | short | Short |  |  | | int | Integer | Integer(int) | int intvalue() | | long | Long |  |  | | float | Float |  |  | | double | Double |  |  | | boolean | Boolean |  |  | | char | Character |  |  |   ！！！**表达式1** **运算符**=**表达式2**相当于**表达式1**=**表达式1** **运算符** **表达式2**，**表达式1** **运算符** **表达式2**生成1个新对象。 | |

## 派生数据类型

（**某数据类型**的）数组

|  |  |
| --- | --- |
| 声明  在**某数据类型**的变量声明语句中，在**某数据类型**后插入[]或在**变量名**后插入[]。 | 1、新数据类型是**某数据类型**的第一个[]前（如果没有[]就在**某数据类型**后）插入一个[]。 |
| 创建  new **某数据类型**的第一个[]前（如果没有[]就在**某数据类型**后）插入一个[**元素个数**]  1、boolean数据类型的元素初始化为false，其它基本数据类型的元素初始化为0，引用数据类型的元素初始化为null。  {**表达式**(,**表达式**)\*}  2、必须在声明语句中使用。 | 1、例：   |  | | --- | | **某数据类型**[]**变量名**;  **变量名**=new **某数据类型**的第一个[]前（如果没有[]就在**某数据类型**后）插入一个[**元素个数**]; | | **某数据类型**[]**变量名**=new **某数据类型**的第一个[]前（如果没有[]就在**某数据类型**后）插入一个[**元素个数**]; | | **某数据类型**[]**变量名**={**表达式**(,**表达式**)\*}; | |
| 数据成员  length |  |
| 方法成员 |  |
| 操作   |  |  | | --- | --- | | []引用数组元素 | ArrayOutOfBoundsException | | 1、数组下标的数据类型可以是：byte、short、int、char。 |

## 自定义数据类型

类

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 声明  (|public )(|abstract |final )class **类名**( extends **基类名**)?( implements **接口名**(,**接口名**)\*)?  {  ( (**数据成员声明**|**方法成员声明**)  )\*} | 1、存取控制符   |  |  | | --- | --- | | 没有 | 默认类，可以被当前包中的类访问。 | | public | 公有类，可以被所有类访问（需要引入）。 |   2、   |  |  | | --- | --- | | 没有 | 都不是 | | abstract | 抽象类 | | final | 终结类 | |
| 类成员的访问控制符（访问权限控制符）   |  |  | | --- | --- | | 无 | 可以被当前包中的类访问。 | | public | 可以被所有类访问。 | | protected | 可以被当前类和当前类的子类访问。 | | private | 可以被当前类访问。 | | |
| 数据成员   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 声明  (|public |protected |private )(static )?(final )?(transient )?(volatile )?**数据类型名** **变量名**(=**表达式**)?(,  **变量名**(=**表达式**)?)\*; | 1、   |  |  |  | | --- | --- | --- | |  | 没有 | 有 | | static | 实例变量 | 类变量（静态变量） | | final |  | 只读。 | | transient |  | 变量是临时常量。 | | volatile |  | 变量是共享变量，由多个并发线程共享。确保多个线程对变量的存取的正确性。 |   2、若不初始化，则boolean数据类型的元素初始化为false，其它基本数据类型的元素初始化为0，引用数据类型的元素初始化为null。 | | |
| 方法成员  1、构造方法   |  |  | | --- | --- | | 声明  (|public |protected |private )**类名**((**变量声明语句**(,**变量声明语句**)\*)?)  {  ( **语句**  )\*} | 1、第一条语句可以是this(**实参列表**);，调用1个构造方法。 | | 调用 | 1、不能被直接调用。 |   2、其它方法成员   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 声明  (|public |protected |private )(static )?(final )?(abstract )?(native )?(synchronized )? **返回值数据类型名** **方法名**((**变量声明语句**(,**变量声明语句**)\*)?)  {  ( **语句**  )\*}  可变长参数：变量的数据类型是**某数据类型**的数组，**变量声明语句**换成**某数据类型名**...**变量名**。 | 1、   |  |  |  | | --- | --- | --- | |  | 没有 | 有 | | static | 实例方法 | 类方法 | | final |  | 终结方法 | | abstract |  | 抽象方法 | | native |  |  | | synchronized |  | 确保多个线程对方法的访问的正确性。 |   2、可变长参数至多1个，如果有可变长参数，可变长参数是最后1个参数。  ！！！主调函数最少可以给可变长参数传0个实际参数。 | | finalize方法  protected void finalize()throws throwable | 1、在java.lang.Object中声明。  2、垃圾收集器释放对象之前自动调用。 | | toString方法  public String toString() | 1、默认return getClass().getName()+'@ '+Integer.toHexString(hashCode());。  2、对象转成字符串时自动调用。 |   3、传参   |  |  | | --- | --- | | 基本数据类型 | 值传递 | | 对象类型、数组 | 引用传递 |   4、this：非static方法成员的1个隐含形参。 | |
| 类和类的关系：组合  （略） | |
| 类和类的关系：继承  1、不指定基类则默认java.lang.Object。 | |

|  |  |
| --- | --- |
| 声明对象  **类名** **变量名**=**表达式**(,  **变量名**=**表达式**)\*; | 1、变量的数据类型是**类名**。 |
| 创建对象  new **类名**(**实参列表**) | 1、例：**类名** **变量名**=new **类名**(**实参列表**);。 |

枚举类型

|  |
| --- |
| 声明  (|public )enum **枚举类型名**( implements **接口名**(,**接口名**)\*)?  {  ( **枚举对象名**(,**枚举对象名**)\*;  )?( (**数据成员声明**|**方法成员声明**)  )\*}  1、自动继承java.lang.Enum类。  2、构造函数的访问权限默认是private且只能是private。  3、每个枚举对象是public、static、final的，数据类型是**枚举类型名**。若有枚举对象被使用，第1次被使用时，所有枚举对象各自动调用1次构造函数。 |
| 方法   |  |  | | --- | --- | | static **类名**[]values()  得到枚举值的数组。 | int ordinal()  得到枚举对象的下标，从0开始。 | | static **类名** valueOf(String)  得到枚举对象名对应的枚举对象。 | String toString()  得到声明时写的枚举对象名。 | |

DecimalFormat类

java.text包

|  |  |
| --- | --- |
| DecimalFormat(String) | String format(double) |

System.out

|  |
| --- |
| printf |

# 语句

控制转移

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 选择结构   |  |  |  | | --- | --- | --- | | if(**表达式**)  **语句** |  |  | | if(**表达式**)  **语句**  else  **语句** | 相当于   |  | | --- | | if(**表达式**)  **语句**  if(!**表达式**)  **语句** |   才怪，因为执行语句可能会影响表达式的值。 |  | | switch(**表达式**)  {  ((case **表达式**:|default:)  ( **语句**  )\*)\*} | 1、switch语句的表达式的数据类型是整型或字符串。  2、case的表达式是常量表达式，数据类型是整型或字符串。1个switch语句直接包含的每个case标签的表达式的值互不相等。  3、1个switch语句直接包含至多1个default标签。  4、（无论default在什么位置，）若无case标签被匹配，控制跳到default标签。 | 1、1个case的语句的最后1句为break;可以防止执行后面所有并列的case。 | |
| 循环结构   |  |  | | --- | --- | | while(**表达式**)  **语句** |  | | for(**语句1**(**表达式1**)?;(**表达式2**)?)  **语句2** | 相当于   |  | | --- | | **语句1**  while(**表达式1**)  {  **语句2**  **表达式2**;  } |   才怪，因为**语句2**可能直接包含continue;。  1、for后的括号内语句是空语句、声明语句或表达式语句。  2、**表达式1**不写相当于写1。 | | do  **语句**  while(**表达式**); |  | |
| 跳转  1、**标签名**:定义1个标签。  2、标签与直接包含它的代码块的}之间至少有1条语句。   |  |  |  | | --- | --- | --- | | break; | 在switch中，跳过剩余语句；在while、for、do中，退出循环。 | 1、只能在switch、while、for、do中使用。  2、只能退出直接包含该语句的1层switch/while/for/do。 | | break **标签名**; | 跳过标签指定的代码块、switch的剩余语句或退出标签指定的while、for、do循环。 | 1、只能在代码块、switch、while、for、do中使用。 | | continue; | 略过循环体剩余语句1次。 | 1、只能在while、for、do中使用。  2、只能略直接包含该语句的1层while/for/do。 | | continue **标签名**; | 略过标签指定的循环的循环体剩余语句1次。 | 1、只能在while、for、do中使用。 | | return( **表达式**)?; | 将控制返回给主调函数。  在返回值的数据类型不为void的函数中，返回表达式的值。 | 1、返回值的数据类型不是void的函数必须包含至少1个return语句。 | |

# 函数

函数重载

1、任意两个版本的函数的形参类型列表不相同。