PDF-XChange Click to buy NOW! www.docu-track.com

一． 基础部份

java 语言是解释执行，java 源码是通过编译生成一种特殊的.class 的中间字解码文件，然后再有 JVM 进行解释执行。

java 语言对指针进行了上层的封装，它保证能够通过这个指针（引用），来访问有效的内存单元。

java 语言不允许多继承，使继承关系成树装图，每个类都只能由一个父类。

java 语言的开发效率高，但执行效率低。（相当于 c++的 55%）

java 的垃圾回收机制，在 java 中 new 的对象不需要向 c++一样进行 delete 操作，JVM 会根据情况回收垃圾对象。（懒汉机制，等待资源没有的时候才回收）我们只能够建议 JVM 进行垃圾回收，例如（System.gc() RunTime.gc()这两个方法就是建议 JVM 进行垃圾回收的方法）

JDK，java 开发工具包（类库和运行命令），JRE，java 运行环境，JVM，java 虚拟机（解释执行的核心，对字节码进行翻译成运行环境的机器码，它可以屏蔽平台差异。JVM 是不跨平台的。）
JAVA_HOME,指明 JDK 安装的位置，CLASSPATH，指明类文件的位置，PATH，指明命令的可执行文件的位置。

java 源文件的文件名必须和文件中定义 public class 的类名（大小写页要相同）相同。

java 源代码中的 main 方法的定义写法。main 方法是程序的入口。

```
public static void main(String[] args){
        System.out.println("Hello world");
    }
```

java 源文件也要先编译，使用 javac xxx.java 格式的命令得来编译，使用 java xxx 来运行。
定义包结构要放在有效代码的第一行，package xxx.xxx，包的定义在一个程序中只能由一个，在加上包定义之后编译可以使用 javac -d 路径 xxxx.java,这个-d 这个命令行的参数可以指定包结构的位置“ .”代表当前目录。在运行时要使用类的全名

java xxx.xxx.xxxx 用包名以点分隔。运行时要在包结构的上一层目录来运行。

java 中的注释，
单行注释 //.....
多行注释 /* .....*/
文档注释/** .....<p>(换行标签)*/,用 javadoc 命令可以根据原码中的文档注释生成注释文档（html 格式）。文档注释中可以使用 html 标签。
javadoc -d 路径 （指定注释文档的保存路径）
文档注释一般写在类定义之前，方法之前，属性之前。
在文档注释中可以用 @author 表示程序的作者，@version 表示程序的版本，前两个注释符


号要写在类定义之前，用于方法的注释@param 对参数进行注释,@return 对返回值进行注释 @throws 对抛出异常的注释。

jar 命令用于打一个 xxx.jar 文件

用法：jar {ctxu}[vfm0Mi] [jar-文件] [manifest-文件] [-C 目录] 文件名 ...

选项：

-c 创建新的存档
-t 列出存档内容的列表
-x 展开存档中的命名的（或所有的）文件
-u 更新已存在的存档
-v 生成详细输出到标准输出上
-f 指定存档文件名
-m 包含来自标明文件的标明信息
-0 只存储方式；未用 ZIP 压缩格式
-M 不产生所有项的清单（manifest）文件
-i 为指定的 jar 文件产生索引信息
-C 改变到指定的目录，并且包含下列文件：

如果一个文件名是一个目录，它将被递归处理。

清单（manifest）文件名和存档文件名都需要被指定，按'm' 和 'f'标志指定的相同顺序

示例 1：将两个 class 文件存档到一个名为 'classes.jar' 的存档文件中：

```
jar cvf classes.jar Foo.class Bar.class
```

示例 2：用一个存在的清单（manifest）文件 'mymanifest' 将 foo/ 目录下的所有文件存档到一个名为 'classes.jar' 的存档文件中：

```
jar cvfm classes.jar mymanifest -C foo/ .
```

一般在使用使用 jar cvf 文件名.jar 文件所在路径（xxx/xxx/xxx.class）也可以压缩一个目录,只要在制定路径是指定为文件夹，jar 命令的命令行参数在使用时可以以“ -”开头，也可以不用。

java 程序的运行过程，首先是启动 java 虚拟机，然后就是去找.class 文件，先是从系统的类库中找（系统之会在跟目录下查找，所以需要完整类名），如果找不到的话会去 CLASSPATH 所设置的目录去找。然后加载到 java 虚拟机中。

系统会在每个 java 程序中隐含导入了 java.lang 这个包，import 包名，导入包中的类文件。

java.lang 包，这是一个基础包。

java.util 包,这个包是工具类的包。

java.io 包,这个包是用于输入输出操作的

java.net 包,这个包是用于网络编程。

java.awt，java.swing,javax.swing java.event 包，这些包用于 java 的图形编程用的包。

applaction java 的应用程序，java 应用程序中必须有一个 main()方法。

标识符和关键字

Java 代码中的“；”、“ {}”、“ ”


Java 语句以分号分隔，Java 代码块包含在大括号内，忽略空格。标识符

1) 用以命名类、方法和变量、以及包遵守 JAVA 的命名规范类以每个单词都以大写字母开头。方法和变量第一个字母不大写，其他照旧。

2) 只能以字符、“ _”或“ $”开头；

3) 无长度限制。

java 中的关键字

goto 和 const 在 java 中虽然不再使用但是还作为关键字存在

java 中没有 sizeof 这个关键字了，java 中的 boolean 类型的值只能用 true 和 false，且这两值也是关键字。

java 语言中没有无符号这个关键字（unsigned）

java 中的数据类型

1) 整型

byte	1 字节	8 位	-128 到 127
short	2 字节	16 位	-2^15 到 2^15-1
int	4 字节	32 位	-2^31 到 2^31-1
long	8 字节	64 位	-2^63 到 2^63-1

2) 浮点类型

float	4 字节	32 位
double	8 字节	64 位

3) 字符类型

char	2 字节	16 位

4) 布尔型

boolean	false/true

注：1) char 是无符号的 16 位整数,字面值必须用单引号括起来； ‘ a ’

2) String 是类，非原始数据类型；

3) 长整型数字有一个后缀为“ L”或“ l”，八进制前缀为“ 0”，十六进制前缀为“ 0x”；

4) 默认浮点类型为 double；

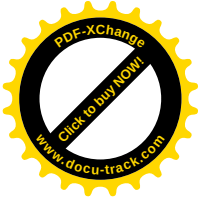
5) float 数据类型有一个后缀为“ f”或“ F ”,Double 数据类型后可跟后缀“ D”或“ d “

6)char 类型也可以用通用转译字符，但是不能用 ASCII 码。可以用“ \u0000”这种格式，因为 char 型中使用的是 unicode 编码方式。

注：整型值存放，正数存放原码（二进制码），负数则存放补码（原码按位取反末位加一）。

注：实型值在存储时会损失精度，所以不要直接比较两个实型值。系统默认的实型都是 double 型，要使用时要在数据后加个 f，或者强行转换。强转（占字节数大的类型转到占字节数小的类型）时会放弃高位值只取低位值。

java 中的数字数据类型减灾由占字节数小的类型到占字节数大的类型的可以有自动转换，反之则需要强行转换，char 型和 int 型之间可以相互转换。char 和 short 不能像户转换。

注意：隐式类型转换；
a 运算符 b ，如果 a，b 中有任意一个是 double 型，前面运算的结果就是 double 型，如果 a，b 中有任意一个是 float 型，前面运算的结果就是 float 型，如果 a，b 中有任意一个是 long 型，前面运算的结果就是 long 型，如果 a，b 中没有 double、float、long 型，那么其结果就为 int 型。

所有基本数据类型在使用时会事先分配空间，只本身就存在空间中，在传递时，就是值传递，不是引用传递。

在类中定义的方法在返回值前加上 static 修饰符就可以在 main 方法中调用了。如果不用 static 那就需要在 main 方法中创建对象，使用对象来调用对象的方法。

```
public class Test{
    public static void main(String[] args){
        Test t=new Test();
        int b=1;
        int c=2;
        int[] a=new int[10];
        t.sqort(a);
        add(b,c)

    }
    public int[] sqort(int[] a){
        .......
    }
    static int add(b,c){
        .......
    }
}
```

java 中的运算符（java 的运算符的优先级和结合性和 c++相同）

System.out.println(3/2)　按整型计算 得 1
1) >>=　前面是零补零，前面是一补一；
2) >>>=　无符号右移（强制右移都会移进一），
　　>>=和>>>=对于负数不一样
　　正数：右移 n 位等于除以 2 的 n 次方
　　负数：变成正数。
3) &&　短路与，前面为假，表达式为假，后面的操作不会进行，& 会对所有条件进行判断。


4) || 短路或，前面为真，表达式为真，后面的操作不会进行，| 会对所有条件进行判断。
例：

```
if(a<3&(b=a)==0)     b 赋值
if(a<3&&(b=a)==0)    b 不赋值
```

5)instanceof，是用于判断一个对象是否属于某个类型
6)java 中的求余运算符“ %”可以对两个实型变量求余

注：按位与是为了让某些位置一，按位或是令某些位置零，按位异或是令某些位取反。

注：使用左右位移和无符号右移运算符的使用方法是 变量名<<=位移位数 ，变量名>>=位移位数 (前两个运算符是不会忽略整形符号位，也称逻辑位移),变量名>>>=位移位数

注意：左右位移和无符号右移运算符只能用于整形及其兼容类型（byte，int，short，long）

注意：java 程序的运行过程，首先是启动 java 虚拟机，然后就是去找.class 文件，先是从系统的类库中找（系统之会在跟目录下查找，所以需要完整类名），如果找不到的话会去 CLASSPATH 所设置的目录去找。然后加载到 java 虚拟机中。如果要使用到其他的在 JAVA_HOME 中没有的类或者是其他公司提供的第三方的.jar（jar 包）文件时，要把它的路径及文件名加到 CLASSPATH 中。

java 的流程控制
控制流
if()
if()….else
if()…..else if()….else

注意：else 只是和其上面的同层的最近的 if()来配对。

```
switch(){
case 'a':……..
case 1:……break;
default:
……
}
```

注解：switch()内数据类型为 byte short char int 类型，只有以上四种类型的才可以在 switch()中使用。case 块中不加 break 时顺序执行下面的语句。

循环语句

```
for(int i=0;i<n;i++){}
while(){}
do{} while();----------注意加分号
```

例子：


```
loop:for(int i=0;i<n;i++){
    for(int j=0;j<m;j++){
        if(3==j){
            break loop;//----------------loop 为标签  只能用在循环语句中，循环//嵌套中用于跳到外层循环
}
    }
}
```

辨析：

```
int x,a=6,b=7;
x=a++ + b++; //----------a=7,b=8,x=13
int  x=6;x=~x;//----------------  6 的二进制 0110 取反得 11001 再转成补码（取反加一）10111 ＝ －7
```

break，跳出本层循环，执行后面的代码，continue，提前终止本次循环，再一次进行循环或循环条件满足或不满足后退出循环。break 标签名; continue 标签名；这两条语句知识表示跳出有标签的循环和提前终止本次有标签的循环，只能用在循环语句（多层循环嵌套）中，循环嵌套中用于跳到外层循环。

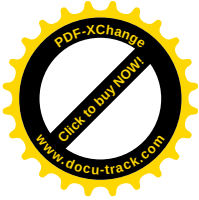
注意：for 循环在使用时一定要注意不要忘记()中的两个"；"，死循环的写法 for(;;){}或者是用
while(true){}

注意：System.out.println("..."+a)在使用这个语句时，它会将其中非字符串（String）的值转换成字符串（不是所有数据类型都可以的）。

java 中的数组 Array，其包含两个部分，分别是数组的引用和数组的空间两部分。

声明数组

1) 一组相同类型(可以是类)数据的集合;
2) 一个数组是一个对象;
3) 声明一个数组没有创建一个对象;
4) 数组能以下列形式声明：
   int[] i 或 int i[]
   Car[] c 或 Car c[]
   *C++中只能 Car c[]
   *JAVA 中推荐用 Car[] c;
   5)数组的定义 如：
   int[] a（数组引用声明）=new int[10]（数组空间的声明，并把空间首地址赋值给数组的引用）
   int[] a；

a=new int[20];

创建数组

1) 创建基本数据类型数组　　int[] i = new int[2];
2) 创建引用数据类型数组　Car[] c = new Car[100];
3) 数组创建后有初始值。
   数字类型为 0 布尔类型为 false 引用类型为 null

注意：访问没有初始化的数组中的值，是会抛出异常的（NULLpointerException）,java 中只保证一位数组的地址是连续的，二维数组实际上是一维数组中有存储了一维数组的引用。

初始化数组

1) 初始化、创建、和声明分开
   int[] i;
   i = new int[2];
   i[0] = 0;
   i[1] = 1;
2) 初始化、创建、和声明在同一时间
   int[] i = {0,1};
   Car[] c = {new Car(),new Car()};

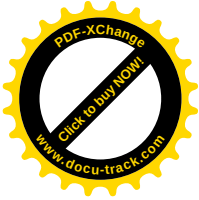
多维数组

1) 有效定义
   int[][] i1 = new int[2][3]; （同时给定一维，二维的空间）
   int[][] i2 = new int[2][]; （给定一维的空间，二维空间待定）
   i2[0] = new int[2],i2[1] = new int[3];
   *C++中 int[][] =new int[][3];有效
2) 无效定义
   int[][] i1 = new int[][3];
3) 数组长度 ----------数组的属性 length（在二维数组中这个属性只代表第一维的长度）
   int[] i = new int[5];
   int len = i.length;//len = 5;
   Student[][] st = new Student[4][6];
   len = st.length;//len = 4;
   len = st[0].length;//len = 6;

数组拷贝

System.arrayCopy(Object src, int srcPos, Object dest, int destPos, int length);
src 源数组，srcPos 从第几位开始拷贝，dest 目标数组，destPos 目标数组放置的起始位置，length，表示要拷贝的长度。
拷贝一个数组到另一个数组。

类的对象的创建和对象数组

一个 xxx.Java 文件中可以定义多个类但是只能由一个 public 修饰的类，也只能以这个类的类名作为.java 的文件名。

java 中的类的对象的创建，要先创建这个对象的引用， 例如：Car c；然后用 new 这个关键字创建一个对象的实例（对象的空间） 例如：c=new Car();，然后对象的实例的空间首地址赋值给对象的引用。多个对象的引用可以同时引用自同一个对象的实例，但是对象的引用只能引用一个对象的实例。

对象的引用和对象的实例间就像是牵着气球的线和气球一样。

注意：只有一个没有被任何对象的引用所引用的对象的实例才会边城垃圾等待被垃圾回收。

对象数组

例：Car[] c=new Car[3];
　　c[0]=new Car();

注意：存放基本类型的数组的数据是直接存放在数组的空间中，而对象的数组在数组空间中存放的则是对象的引用。

定义在类中类的属性是实例变量，定义在类的方法中的变量是局部变量。实例变量是保存在对象空间中的，而局部变量则是在方法调用的分配空间，调用结束后就释放空间。

注意：在类的定义中属性的定义和方法的定义 必须写在类里。

注意：系统会自动初始化实例变量，数字类型为 0 ，布尔类型为 false ，引用类型为 null。局部变量需要初始化，必须赋初值。如果不赋初值无法通过编译。

Java 中的方法调用中参数传递有两种，一个是对于参数是基本类型的使用的是值传递（直接传参数的值），另一个是引用传递，它是用于参数是类的对象，它传递的是这个对象的引用。

作者：叶加飞　(steven ye)

mailto: leton.ye@gmail.com