华为 java 培训讲义 第一天

配置 java 环境变量：

JAVA_HOME：配置 JDK 的目录

CLASSPATH：指定到哪里去找运行时需要用到的类代码（字节码）

PATH：指定可执行程序的位置

LINUX 系统（在" .bash_profile "下的环境变量设置）

```
JAVA_HOME=/opt/jdk1.5.0_06
CLASSPATH=.:$JAVA_HOME/lib/tools.jar:$JAVA_HOME/lib/dt.jar
PATH=$PATH:$JAVA_HOME/bin:.
export JAVA_HOME CLASSPATH PATH        (将指定的环境变量声明为全局的)
```

windows 系统：

右击我的电脑-->属性-->高级-->环境变量

Java 的运行过程：

编译：生成可执行文件，如 C++中利用 g++生成 a.out，效率高，但不跨平台

解释：解释器把源文件逐行解释，跨平台但效率不高

在 java 中：先编译后解释，把.java 文件编译成.class 字节码文件

Java 源代码文件(.java 文件)--->

Java 编译器(javac)--->

Java 字节码文件(.class 文件，平台无关的)--->

Java 解释器(java)，执行 Java 字节码

Java 的垃圾回收：

由一个后台线程 gc 进行垃圾回收

虚拟机判定内存不够的时候会中断代码的运行，这时候 gc 才进行垃圾回收

缺点：不能够精确的去回收内存

java.lang.System.gc(); 建议回收内存，但系统不一定回应，他会先去看内存是否够用，够用则不予理睬，不够用才会去进行垃圾回收

内存中什么算是垃圾：

不再被引用的对象(局部变量，没有指针指向的)

java 的安全性：

沙箱机制：只能做沙箱允许的操作

通过下面环节，实现安全

加载有用的类文件，不需要的不加载

校验字节码，查看允许的操作

查看代码和虚拟机的特性是否相符

查看代码是否有破坏性
查看是否有违规操作，如越界
查看类型是否匹配，类型转换是否能正确执行

源程序：

```
package mypack;          //相当于一个目录

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

注：

1、文件名必须和 public 修饰的类名一致，以.java 作为文件后缀，如果定义的类不是 public 的，则文件名与类名可以不同。

2、一个.java 文件中可以有多个 class，但是只有一个 public 修饰的类。

3、java 源代码文件编译后，一个类对应生成一个.class 文件

4、一个 java 应用程序应该包含一个 main()方法，而且其签名是固定的，它是应用程序的入口方法，可以定义在任意一个类中，不一定是 public 修饰的类

编译：javac -d . HelloWorld.java

含有包的类，在编译的时候最好用上面的格式，-d 指的是让该类生成的时候按照包结构去生成，" . "指的是在当前路径下生成

如果不用上面的格式，也可以用 javac HelloWorld.java，但是需要注意的是包结构就要由自己去建立，然后将生成的.class 文件放到该目录下

执行：java mypack.HelloWorld

将字节码文件交给 Java 虚拟机去解释执行

需要注意的事，必须使用包名.类名去解释执行

包(package)：把源文件放在目录下

由于工程的需要，将不同的源文件放在不同的目录下，从而引入了包。

包可以看作就是一个存放 java 源文件的目录。

在源码中声明一个包名：package p;(只能放在第一行，且最多只能是一行)

如果指定多层包，那么在包名之间我们可以用.作为分隔符：package p1.p2.p3.p4;

用“javac HelloWorld.java –d 绝对路径”，编译后生成的字节码文件就会放在指定的包结构下

执行该程序需要用" java 包名.类名 "

引进包中的某个类：import 包名.类名;

引进包中的所有类：import 包名.*;

注释：

// 单行注释， 到本行结束的所有字符会被编译器忽略

/* */多行注释， 在/* */之间的所有字符会被编译器忽略

/** */ 文档注释， java 特有的，在/** */之间的所有字符会被编译器忽略

可以用 javadoc 把 java 源程序中这种注释抽取出来形成 html 页面(只有写在包，类，属性，方法，构造器，引入之前的注释才可以进行抽取)

标识符：

命名规则：

(1) 由字母、数字、下划线、$组成，不能以数字开头

(2) 大小写敏感

(3) 不得使用 java 中的关键字和保留字

关键字：都是小写的，jdk1.2 多了 strictfp(精准浮点型)，关键字 jdk1.4 多了 assert(断言)关键字， jdk1.5 多了 enum(枚举) 关键字

随着学习进度，会慢慢接触到的

true、false、null 严格说不应该算关键字，应称其为保留字更合适

习惯：

(1) 标识符要符合语义信息

(2) 包名所有字母小写

(3) 类名每个单词首字母大写，其它小写 //TarenaStudent

(4) 变量和方法：第一个单词小写，从第二个单词开始首字母大写 //tarenaStudent

(5) 常量：所有字母大写，每个单词之间用" _ "连接

基本数据类型：8 种

1) 整型

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

2) 浮点类型

float	4B	32 位
double	8B	64 位

3) 字符类型

char	2B	16 位

4) 布尔型 1B

boolean	false/true

注：

1、Java 中的自动类型提升问题。

1)、正向过程：从低字节到高字节可以自动转换。

byte->short->int->long->float->double

2)、逆向过程：从高字节到低字节用强制类型转换。

例：int a = (int)4.562；

注：逆向转换将丢失精度。

2、boolean：只有 true 和 false。

3、char：Java 中用" \u 四位十六进制的数字 (即使在注释中出现\u，后面如果跟的不是 4 个数字，也会报错)"表示将字符转换成对应的 unicode 编码，字符类型要用单引号括起来。

4、默认浮点类型为 double，float 数据类型有一个后缀为" f "或" F "。

5、long 类型有一个后缀，为" l " 或者" L "

引用数据类型：

类、接口、数组

引用类型 变量名 = new 引用类型名(参数); //new 后面一般跟的都是类的构造器

成员：写在类体括号里面的

内存空间的分配：

内存分为：

栈：存放简单数据类型变量(值和变量名都存在栈中)，存放引用数据类型的变量名以及它所指向的实例的首地址

堆：存放引用数据类型的实例

华为培训讲义第二天

局部变量：不是声明在类体括号里面的变量

(1)必须要先赋值，后使用，否则通不过编译，局部变量没有默认初始化值

(2)作用范围：定义开始到定义它的代码块结束

(3)同一范围内，不允许 2 个局部变量命名冲突

参数传递时，简单类型进行值转递 (参数进行传递时都会先去栈中生成一个副本的，使用结束后释放)

自动类型提升：

```
byte a = 1;
byte b = 2;
a = a+b;          //编译出错自动类型提升成 int
a += b;           //自加没有自动类型提升问题
```

类型自动提升规则：

a 和 b 作某种运算

a 和 b 中有 double，结果就是 double

a 和 b 中有 float，结果就是 float

a 和 b 中有 long，结果就是 long

除此之外，结果都是 int

把高字节转成低字节,需要作强制类型转换. byte c=(byte)a+b;

移位运算符：效率最高

>> 有符号右移，补符号位

移负数位，则将该数值加 32 后再进行移位

数值的 2 进制是按照补码保存的

>>> 右移后高位都补 0

逻辑运算符：

&/|也可以作为逻辑运算符

&& 先判断前面一个条件，如果为假，则不用计算后一个条件

|| 先判断前面一个条件，如果为真，则不用计算后一个条件

" + "运算符：

两个操作的对象是数值时，是加法

如果有一个是字符串时，则是字符串的连接

流程控制语句：

同 Core C++

switch 中的变量类型只能是 byte、 short、int、char 四种类型以及 enum 类型

switch(exp) exp 可以是整形表达式或者 enum 类型数据

数组：

声明数组：

数组能以下列形式声明：

类型[] array;

类型 array[];

注：

JAVA 中推荐用：类型[] array;

一个数组是一个对象

声明一个数组没有创建一个对象

声明时不用指定长度

创建数组：

创建基本数据类型数组：int[] i = new int[2];

创建引用数据类型数组：Student[] s = new Student[100];

数组创建后其中的元素有初始值

类型	默认值
byte	0
short	0
int	0
long	0l
float	0.0f
double	0.0d
char	\u0000
boolean	false
reference types	null

注：