第1章-0-Java语言概述

面向对象支持：封装、继承和多态。

编译和解释并存。由编译器将Java源程序（xxx.java)编译成字节码文件(xxx.class)，然后再由Java运行系统解释执行字节码文件（解释器将字节码再翻译成二进制码运行）。

字节码（bytecode)是JVM的指令组，最大的好处：跨平台运行。编写一次，到处运行。

第1章-1-Java语言开发工具

Java应用程序命名规则：源文件扩展名须为.java，如果源文件有多个类，则最多只能有一个public类，如果有的话，那么源文件的名字必须与这个public类名字相同(文件名的大小写可以与public类名的大小写不同)；（吐槽：可恶，这种大小写不规范真是为以后埋雷！）如果源文件没有public类，那么源文件名字可任意。

第1章-2-Java语言基础知识

面向对象基本特征：抽象、封装、继承、多态。

Java语言的复合语句与C复合语句不同的是：Java语言**不允许**在两个嵌套的复合语句内声明两个同名的变量。

常量：变量声明前面加上final修饰。常量一旦被初始化以后就不能被修改了。

类的成员变量有默认值。这些变量通过new在堆中分配。想象为new操作本身对分配的内存区域首先全部清零。

方法局部变量没有默认值。这些变量分配在栈中，随着方法的调用和返回而分配和释放。

(测试了一下，如果对方法局部变量不设初始值就使用，编译无法通过)。

Java基本数据类型同C基本一致，但char为2bytes，布尔型为boolean，字符串为String

字符串连接运算符+

<<左移，>>算术右移，>>>逻辑右移

移位运算的移位数，会根据变量字长做调整。

▫ 比如整数占用四个字节，长度为32, 所以整数移位 的最大有效位数为31(5个二进制位). 大于31的数， 会对32取模然后移位（对逻辑右移同样成立）

数据存储的字节顺序为大端存储。（比如int a = 1; 00为低位，01在高位）

Review

- 运行环境搭建和应用程序例子
- JVM和字节码
- 类，对象是类的实例(new关键字)；概念，实体。
- 没有指针；类不支持多继承
- 抽象、封装、继承、多态
- 基本数据类型：没有无符号数
- 默认值: 类的成员变量有默认值，方法局部变量没有默认值，必须初始化才能引用，否则编译错误。
- 作用域: class内部；方法/代码块内部
- 文字量：整数默认类型是int, 小数默认类型是double
- 常量: final
- 字符串连接: + ,可以连接字符串和任意类型，结果是字符串
- >>>: 逻辑移位。
- 数据存储的字节顺序总是 big endian

简洁运算（&&、||）

非简洁运算（&、|）

Review

- 简洁运算(&&、||)与非简洁运算(&、|)
- >>>
- 变量默认值
- 增强型for循环
- 运算符+
- String 常量

类型比较运算符 instanceof

类型转换

扩展转换。从整数类型向float或double转换，会损失精度。

窄化转换。窄化转换可能会丢失信息。

每个数组都有一个由public final修饰的成员变量length.

对比：字符串长度是一个方法：

```
String str = “abc”;
int len = str.length();
```

声明数组时无需指定数组元素个数，也不为其分配内存空间。不能直接使用，必须经过初始化分配内存后才能使用。

用关键字new构成数组的创建表达式，可以指定数组类型和数组元素个数。元素个数可以是常量也可以是**变量**。基本类型数组的每个元素都是一个基本类型的变量；引用类型数组的每个元素都是对象的引用。

创建数组时，如果没有指定初始值，数组便被富裕默认值初始值。

数组名是一个引用。

```
public class MyArray{
    public static void main(String[] args){
        int MyArray[];
        MyArray = new int[5];
        for(int i = 0;i < MyArray.length;i++)
            MyArray[i] = i;
        int Test[];
        Test = MyArray;
        for(int i : Test)
            System.out.print(i + " ");
    }
}
```

数组的复制：System类提供的方法

```
public static void arraycopy(Object source, int srcIndex, Object dest, int
destIndex, int length)
```

```
public class MyArray{
    public static void main(String[] args){
        char c[] = {'a', 'b', 'c', 'd', 'e', 'f'};
        char copyTo[] = new char[3];
        System.arraycopy(c, 1, copyTo, 0, 3);
        System.out.println(copyTo);
    }
}
```

```
public class MyArray{
    public static void main(String[] args){
        int [][] arr = {{1, 2, 3}, {1, 2, 3, 4, 5}, {1, 2, 3, 4}};
        System.out.println(arr.length);
        System.out.println(arr[1].length);
        System.out.println(arr[2].length);
    }
}
```

1.6.5 多维数组(续)

```
int[ ][ ] myArray;
myArray = new int[3][ ] ;
myArray[0] = new int[3];
```

```
int[ ] x = {0, 2};
int[ ] y = {0, 1, 2, 3, 4} ;
myArray[1] = x ;
myArray[2] = y ;
```