**Java 基础语法**

一个Java程序可以认为是一系列对象的集合，而这些对象通过调用彼此的方法来协同工作。下面简要介绍下类、对象、方法和实例变量的概念。

* **对象**：对象是类的一个实例，有状态和行为。例如，一条狗是一个对象，它的状态有：颜色、名字、品种；行为有：摇尾巴、叫、吃等。
* **类**：类是一个模板，它描述一类对象的行为和状态。
* **方法**：方法就是行为，一个类可以有很多方法。逻辑运算、数据修改以及所有动作都是在方法中完成的。
* **实例变量**：每个对象都有独特的实例变量，对象的状态由这些实例变量的值决定。

**第一个Java程序**

下面看一个简单的Java程序，它将打印字符串*Hello World*

**实例**

public class HelloWorld {  
    /\* 第一个Java程序  
     \* 它将打印字符串 Hello World  
     \*/  
    public static void main(String []args) {  
        System.out.println("Hello World"); // 打印 Hello World  
    }  
}

[**运行实例 »**](http://www.runoob.com/try/showjava.php?filename=HelloWorld2)

下面将逐步介绍如何保存、编译以及运行这个程序：

* 打开Notepad，把上面的代码添加进去；
* 把文件名保存为：HelloWorld.java；
* 打开cmd命令窗口，进入目标文件所在的位置，假设是C:\
* 在命令行窗口键入 javac HelloWorld.java  按下enter键编译代码。如果代码没有错误，cmd命令提示符会进入下一行。（假设环境变量都设置好了）。
* 再键入java HelloWorld 按下Enter键就可以运行程序了

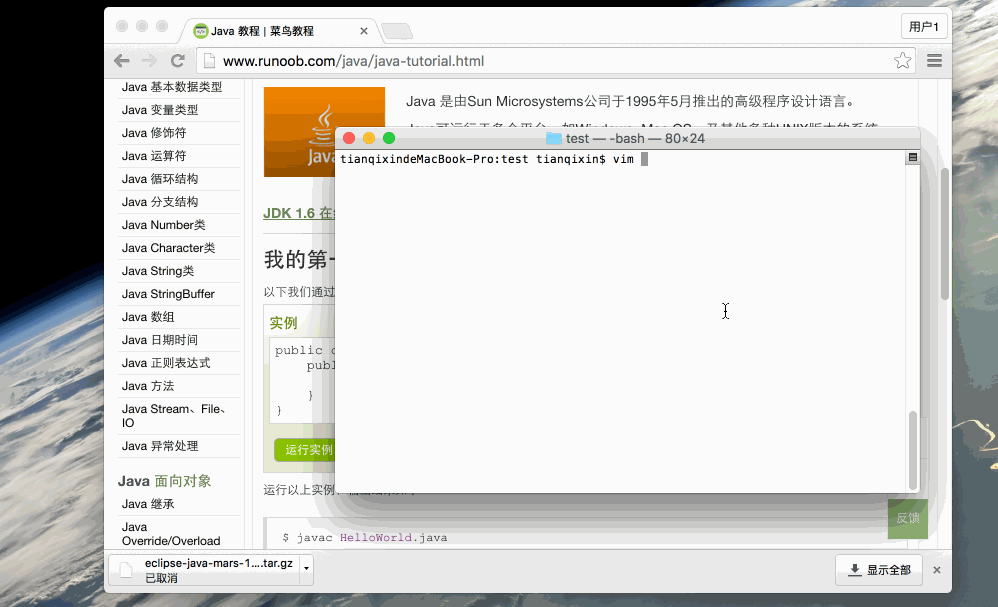
你将会在窗口看到 Hello World

C : > javac HelloWorld.java

C : > java HelloWorld

Hello World

Gif 图演示：



**基本语法**

编写Java程序时，应注意以下几点：

* **大小写敏感**：Java是大小写敏感的，这就意味着标识符Hello与hello是不同的。
* **类名**：对于所有的类来说，类名的首字母应该大写。如果类名由若干单词组成，那么每个单词的首字母应该大写，例如 MyFirstJavaClass 。
* **方法名**：所有的方法名都应该以小写字母开头。如果方法名含有若干单词，则后面的每个单词首字母大写。
* **源文件名**：源文件名必须和类名相同。当保存文件的时候，你应该使用类名作为文件名保存（切记Java是大小写敏感的），文件名的后缀为.java。（如果文件名和类名不相同则会导致编译错误）。
* **主方法入口**：所有的Java 程序由**public static void main(String []args)**方法开始执行。

**Java标识符**

Java所有的组成部分都需要名字。类名、变量名以及方法名都被称为标识符。

关于Java标识符，有以下几点需要注意：

* 所有的标识符都应该以字母（A-Z或者a-z）,美元符（$）、或者下划线（\_）开始
* 首字符之后可以是任何字符的组合
* 关键字不能用作标识符
* 标识符是大小写敏感的
* 合法标识符举例：age、$salary、\_value、\_\_1\_value
* 非法标识符举例：123abc、-salary

**Java修饰符**

像其他语言一样，Java可以使用修饰符来修饰类中方法和属性。主要有两类修饰符：

* 可访问修饰符 : default, public , protected, private
* 不可访问修饰符 : final, abstract, strictfp

在后面的章节中我们会深入讨论Java修饰符。

**Java变量**

Java中主要有如下几种类型的变量

* 局部变量
* 类变量（静态变量）
* 成员变量（非静态变量）

**Java数组**

数组是储存在堆上的对象，可以保存多个同类型变量。在后面的章节中，我们将会学到如何声明、构造以及初始化一个数组。

**Java枚举**

Java 5.0引入了枚举，枚举限制变量只能是预先设定好的值。使用枚举可以减少代码中的bug。

例如，我们为果汁店设计一个程序，它将限制果汁为小杯、中杯、大杯。这就意味着它不允许顾客点除了这三种尺寸外的果汁。

**实例**

class FreshJuice {

enum FreshJuiceSize{ SMALL, MEDUIM, LARGE }

FreshJuiceSize size;

}

public class FreshJuiceTest {

public static void main(String []args){

FreshJuice juice = new FreshJuice();

juice.size = FreshJuice. FreshJuiceSize.MEDUIM ;

}

}

**注意：**枚举可以单独声明或者声明在类里面。方法、变量、构造函数也可以在枚举中定义。

**Java关键字**

下面列出了Java保留字。这些保留字不能用于常量、变量、和任何标识符的名称。

|  |  |
| --- | --- |
| **关键字** | **描述** |
| abstract | 抽象方法，抽象类的修饰符 |
| assert | 断言条件是否满足 |
| boolean | 布尔数据类型 |
| break | 跳出循环或者label代码段 |
| byte | 8-bit 有符号数据类型 |
| case | switch语句的一个条件 |
| catch | 和try搭配扑捉异常信息 |
| char | 16-bit Unicode字符数据类型 |
| class | 定义类 |
| const | 未使用 |
| continue | 不执行循环体剩余部分 |
| default | switch语句中的默认分支 |
| do | 循环语句，循环体至少会执行一次 |
| double | 64-bit双精度浮点数 |
| else | if条件不成立时执行的分支 |
| enum | 枚举类型 |
| extends | 表示一个类是另一个类的子类 |
| final | 表示一个值在初始化之后就不能再改变了 表示方法不能被重写，或者一个类不能有子类 |
| finally | 为了完成执行的代码而设计的，主要是为了程序的健壮性和完整性，无论有没有异常发生都执行代码。 |
| float | 32-bit单精度浮点数 |
| for | for循环语句 |
| goto | 未使用 |
| if | 条件语句 |
| implements | 表示一个类实现了接口 |
| import | 导入类 |
| instanceof | 测试一个对象是否是某个类的实例 |
| int | 32位整型数 |
| interface | 接口，一种抽象的类型，仅有方法和常量的定义 |
| long | 64位整型数 |
| native | 表示方法用非java代码实现 |
| new | 分配新的类实例 |
| package | 一系列相关类组成一个包 |
| private | 表示私有字段，或者方法等，只能从类内部访问 |
| protected | 表示字段只能通过类或者其子类访问 子类或者在同一个包内的其他类 |
| public | 表示共有属性或者方法 |
| return | 方法返回值 |
| short | 16位数字 |
| static | 表示在类级别定义，所有实例共享的 |
| strictfp | 浮点数比较使用严格的规则 |
| super | 表示基类 |
| switch | 选择语句 |
| synchronized | 表示同一时间只能由一个线程访问的代码块 |
| this | 表示调用当前实例 或者调用另一个构造函数 |
| throw | 抛出异常 |
| throws | 定义方法可能抛出的异常 |
| transient | 修饰不要序列化的字段 |
| try | 表示代码块要做异常处理或者和finally配合表示是否抛出异常都执行finally中的代码 |
| void | 标记方法不返回任何值 |
| volatile | 标记字段可能会被多个线程同时访问，而不做同步 |
| while | while循环 |

**Java注释**

类似于C/C++，Java也支持单行以及多行注释。注释中的字符将被Java编译器忽略。

public class HelloWorld {

/\* 这是第一个Java程序

\*它将打印Hello World

\* 这是一个多行注释的示例

\*/

public static void main(String []args){

// 这是单行注释的示例

/\* 这个也是单行注释的示例 \*/

System.out.println("Hello World");

}

}

**Java 空行**

空白行，或者有注释的的行，Java编译器都会忽略掉。

**继承**

在Java中，一个类可以由其他类派生。如果你要创建一个类，而且已经存在一个类具有你所需要的属性或方法，那么你可以将新创建的类继承该类。

利用继承的方法，可以重用已存在类的方法和属性，而不用重写这些代码。被继承的类称为超类（super class），派生类称为子类（subclass）。

**接口**

在Java中，接口可理解为对象间相互通信的协议。接口在继承中扮演着很重要的角色。

接口只定义派生要用到的方法，但是方法的具体实现完全取决于派生类。

下一节介绍Java编程中的类和对象。之后你将会对Java中的类和对象有更清楚的认识。