

# Java 教程

翻译作品

屈庆磊

[quqinglei@icloud.com](mailto:quqinglei@icloud.com)

2013 年 9 月 27 日

## 目录

1 说明	1
2 读者	1
3 先决条件	1
4 版权声明	1
5 译者声明	2
6 Java 观其大略	2
6.1 Java 的历史	2
6.2 你需要的工具	3
6.3 下一步，我们要干啥	3
7 Java 环境配置	3
8 Java 基本语法	3
8.1 第一个程序	3
8.2 基本句法	4
8.3 Java 标识符	4
8.4 Java 修饰符	5
8.5 Java 变量	5
8.6 Java 数组	5
8.7 Java 枚举变量	5

1 说明	2
8.8 保留字	6
8.9 注释	6
8.10 空白行	6
8.11 继承	6
8.12 接口	6
8.13 下一步我们要干啥呢	7
9 Java 对象和类	7
9.1 对象	7
9.2 类	7
9.3 构造函数	9
9.4 创建一个对象	9
9.5 访问实例的变量和方法	10
9.6 源代码文件声明规则	11

## 1 说明

Java 是一门面向对象的高级语言，最初由太阳微系统公司于 1995 年发布。Java 运行于 Windows, Mac OS, 各种 UNIX 以及其变种的操作系统之上。本教程将会完整讲述 Java 语言。

本参考手册将会从简单、可练习的方式循序渐进的教你如何去学习这门语言。

## 2 读者

本参考手册准备向初学者讲述 Java 的基本概念和与其相关的高级用法。

## 3 先决条件

在你开始决定练习本参考手册上面的各种例子之前，我们假设你已经了解什么是计算机程序以及什么是编程语言。本参考手册并不要求你有太多经验。

## 4 版权声明

本手册所有内容版权归属于 [tutorialspoint.com](http://tutorialspoint.com) 在没有得到作者允许的情况下本文档的任何内容不得经过任何形式的再发布，否则视为对侵犯版权法。

本手册可能不可避免的存在一些错误，我们不担保网站以及本手册内容的正确性。如果你发现网站以及本手册存在错误，请发邮件至 [webmaster@tutorialspoint.com](mailto:webmaster@tutorialspoint.com)

## 5 译者声明

本文档为译者个人翻译作品，不允许任何形式的复制、再发布，如果需要可能会联系本作品的最初撰写者，取得中文版的发布权利。

## 6 Java 观其大略

Java 程序设计语言最初是由太阳微系统公司开发，起初是由 James Gosling 创建，于 1995 年作为太阳微系统公司的核心模块发布，也就是所谓的 Java 1.0 [J2SE]

翻译这本手册的时候，Java 的最新版本已经是 Java7u21。归于 Java 语言的先进以及大众化，多种平台被配置从而适合 Java 运行。J2EE 是 Enterprise Applications 的缩写，J2ME 是 Mobile Applications 的缩写，即企业应用和移动应用。

太阳微系统公司已经把 J2 重命名为 Java SE 为和 Java EE 以及 Java ME 相互对照。Java 总冠以：编写一次，随处运行。

Java 是：

- **面向对象** Java 的所有都是面向对象的。由于 Java 基于对象模型，所以她很容易扩展。
- **平台独立** 和其他包括 C/C++ 在内的语言不通，Java 不会被编译为机器相关的二进制程序，她只是被编译为了字节码。字节码分布于网络中并为运行在平台上的虚拟机（JVM）所执行。
- **简单** Java 被设计为简单易学的语言，如果你懂得简单的面向对象模型，就很容易掌握她。
- **安全** 基于 Java 的安全特性可以开发出免受病毒侵扰、无法篡改的系统，她基于公共密钥加密。
- **架构中立** Java 编译器产生的是架构中立的对象文件，所以这种文件可以在许多处理器上运行。
- **可移植** 由架构中立、平台无关性的特征使 Java 非常容易移植。编译器和 Java 是采用 ANSI C 编写，并属于 POSIX 的子集。
- **可靠性** Java 努力在编译时和运行时做错误检查和运行时检查，当然错误无法避免。
- **多线程** Java 具有多线程编程的特点，可以同时运行多个任务。可以使开发者顺利运行交互性程序。
- **解释性** Java 边解释边执行，并不存储在任何地方。The development process is more rapid and analytical since the linking is an incremental and light weight process
- **高性能** 使用 Just-In-Time 编译器使 Java 拥有高性能。
- **分布式** Java 是为网络的分布式环境设计的。
- **动态性** Java 被认为比 C/C++ 更为动态化，她被设计为更加适应不断变化的环境。她携带大量的运行时信息，用于验证和解决运行时对象访问。

### 6.1 Java 的历史

James Gosling 在为他的机顶盒项目中于 1991 年创建了 Java 语言。起初她叫 Oak，那是因为 Gosling 的办公室外面有一棵橡树，她也曾经被称之为 Green。后来被命名为 Java，这算是从一列单词中随机选择的吧。

太阳微电子公司于 1995 年发布了 Java 1.0 版本，她承诺编写一次，随处运行。在流行的平台上提供无成本运行时间。*It promised Write once, Run Anywhere(WORA), providing no-cost run-times on popular platforms*

2007 年 5 月 8 日，太阳微电子公司完成除了对一小部分不属于自己的源代码以外的 Java 所有核心代码的开源进程。

## 6.2 你需要的工具

你需要一个配置在 Pentium 200-MHz 64MB RAM 以上的计算机，来执行本教程中的例子，除此之外还需要一些软件：

- Linux 7.1 或者 Windows 95/98/2000/xp/vista/7 操作系统
- Java JDK
- 一个编辑器，比如：Microsoft Notepad 或者其他文本编辑器。

本文档会提供创建图形操作界面，网络，以及基于 Web 应用程序的必要技术。

## 6.3 下一步，我们要干啥

下一章节介绍开发环境的安装部署

# 7 Java 环境配置

在我们为走更远的路之前，最重要的是先配置好环境。本章会告诉你如何下载、安装 Java，请跟随下面的步骤。算了，我想这个章节还是不要翻译了，必经书本太老，JDK 在不断更新，而且网络上可以搜的到，比如你可以搜：Java 开发环境搭建 跟随网络上的教程就可以了。

# 8 Java 基本语法

我们可以把 Java 程序理解为定义好的一大堆对象，他们通过调用彼此的方法进行交互。我们现在简单的看一下类、对象、方法和实例变量的意义。

- **对象** 对象拥有状态和行为。比如一只狗拥有一下状态特征：颜色、名字、品种，同样拥有摇头摆尾、狗吠、吃饭等行为特征。对象为类的实例。
- **类** 类可以描述为是一种模板或者蓝图，可以把它理解为是一种定义。
- **实例变量** 每个对象都有其独立的实例变量。对象状态的建立其实就是实例变量的赋值。

## 8.1 第一个程序

我们看一下下面这段简单的代码，也就是我们在学习语言中遇到的第一个程序：Hello World。

---

```
1 // MyFirstJavaProgram.java
2 public class MyFirstJavaProgram {
3     /* This is my first java program.
4      * This will print 'Hello World' as the output
5      */
6     public static void main(String [] args) {
7         System.out.println("Hello World"); // prints Hello World
8     }
9 }
```

---

保存以上代码到MyFirstJavaProgram.java 按照以下步骤编译并运行

- 打开一个文本编辑器
- 保存以上代码为: MyFirstJavaProgram.java
- 打开一个终端, 切换到文件保存的路径, 执行以下命令
- javac MyFirstJavaProgram.java
- java MyFirstJavaProgram 结果打印为: Hello World, 则证明编译运行成功

---

```
1 # compile example LINUX
2 % javac MyFirstJavaProgram.java
3 % java MyFirstJavaProgram
4 Hello World
```

---

## 8.2 基本句法

关于 Java 编程, 下面的要点需要记住:

- **大小写敏感** Java 是大小写敏感的语言, "Hello" 和"hello" 代表不同的意思
- **类名** 类名的第一个字母要大些, 当然你小写也可以, 这只是一种约定
- **方法名** 方法名开头字母应该为小写, 这也属于编程的规范约定
- **文件名** 文件名要和类名完全一致, 除了后面的后缀以外
- **public static void main(String args[])** Java 程序从主函数开始执行

## 8.3 Java 标识符

Java 的组件包括各种名称, 类名、变量、方法都被称为标识符。关于标识符的命名规范如下, 其实和其他语言类似, 特别是类 C 语言。

- Java 标识符由数字, 字母和下划线, 美元符号组成。在 Java 中是区分大小写的, 而且还要求首位不能是数字

- Java 关键字不能当作标识符
- 合法标识符如: `age`, `$salary`, `_value`, `_1_value`
- 不合法标识符如: `123abc`, `-salary`

## 8.4 Java 修饰符

和其他语言一样 Java 同样拥有修饰符, 通过修饰符我们可以更改类、方法等元素。基本上有两类修饰符:

- 访问修饰符: `default`, `public`, `protect`, `private`
- 不可访问修饰符: `final`, `abstract`, `strictfp` [tbd]

未来的章节我们会对修饰符进行更深刻的讨论

## 8.5 Java 变量

- 局部变量
- 类变量, (即静态变量) 是全局变量
- 常量, 常量不能被更改, 使用关键字 `final` 修饰, 必须在常量声明时进行初始化

## 8.6 Java 数组

数组是同样类型的不同变量。Java 的数组是存放在堆上的, 这不同于 C, C 默认是在栈上的。我们会在后面的章节中讲述如何声明、创建、初始化数组。

## 8.7 Java 枚举变量

枚举变量从 Java 5.0 被引进。枚举限定一个变量有一个或者多个预定义的值, 在此里列表中的值称之为枚举变量。

使用枚举可以尽量避免使用数字带来的错误。

下面是一个枚举类型的使用例子: [tbd]

---

```
1 class FreshJuice {
2     enum FreshJuiceSize {SMALL, MEDUIM, LARGE}
3     FreshJuiceSize size;
4 }
5
6 public class FreshJuiceTest {
7     public static void main(String args[]) {
8         FreshJuice juice = new FreshJuice();
9         juice.size = FreshJuice.FreshJuiceSize.MEDUIM;
10    }
11 }
```

---

8.8 保留字

下面的表格中是 Java 的保留字，这些保留字不能被当做变量或者标识符名称。

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

8.9 注释

Java 支持多行和单行注释，这个和 C/C++ 一致，且注释不能嵌套。

```
1 public class MyFristJavaProgram {
2     public static void main(String args[]) {
3         /* 这里面包含的就是注释，编译器会自动忽略，
4            而且是多行注释 */
5         System.out.println("Hello World"); // 单行注释，编译器自动忽略
6     }
7 }
```

8.10 空白行

空行和注释行都被称为空白行，Java 会忽略他们

8.11 继承

如果你需要创建一个新的类型，而已经存在一个其他的类，你仅仅需要的是再添加一些其他的代码，你可以继承这个类。从而演化成一个新的类。

这属于代码重用，避免你重复劳动。

8.12 接口

在 Java 语言中，我们可以定义接口，接口是两个对象之间用来通讯的方式。方法在继承中扮演很重要的角色。

接口定义方法，派生类可以使用它，但是至于方法实现完全取决于派生类。

### 8.13 下一步我们要干啥呢

下一章节将要讲述对象和类。在下一章节的结尾你应该能对对象和类有一个清晰的认识。

## 9 Java 对象和类

Java 是一种面向对象的编程语言。她支持以下的概念：

- **Polymorphism** 多态性
- **Inheritance** 继承
- **Encapsulation** 封装
- **Abstraction** 抽象
- **Classes** 类
- **Objects** 对象
- **Instance** 实例
- **Method** 方法
- **Message Parsing** 消息解析

本章中我们将会讲述类和对象的概念。

- **对象** 对象拥有状态和行为。比如狗有颜色、名字、种类的状态，以及摇头摆尾、大声吠叫、吃饭等行为
- **类** 类可以被理解成是模板或者蓝图，而对象是类的实例。或者说类是规范。

### 9.1 对象

让我们深入理解一下什么是对象，其实我们周围有很多很多的对象，车、狗、人等等，这些对象都拥有状态以及行为。

软件对象也拥有状态和行为，一个软件的对象状态存储在一定的字段中而行为在方法定义中。

所以在软件开发中，方法是在对象内部操作，对象间的通讯是通过方法之间的调用进行的。

### 9.2 类

我们可以把类看做是蓝图，下面是一个类的例子：



---

```
1 public class Dog {
2     String breed; // 种类
3     int age;
4     String Color;
5
6     void barking() {
7     }
8     void hungry() {
9     }
10    void sleeping() {
11    }
12 }
```

---

一个类可以包含如下变量类型:

- **局部变量** 定义在方法、构造函数、代码块中的变量被称作局部变量，变量在方法里定义、初始化，并随方法执行完毕而释放。
- **常量** 常量是定义在类中，方法外的变量，这些变量在类加载的时候被初始化。常量可以被类中的方法、构造函数、代码块访问。常量不允许修改，使用关键字 `final` 定义。
- **类变量** 类变量在类中声明，在方法之外，使用 `static` 关键字定义，特点是如果一个类的对象更改了类变量的值，那么在其他对象中，类变量的值随之改变。下面是个例子：

---

```
1 // Cat.java
2 class Cat {
3     static String name = "Kitty";
4     void sayName() {
5         System.out.println(name);
6     }
7 }
8
9 // CatTest.java
10 public class CatTest {
11     public static void main(String args[]) {
12         Cat cat01 = new Cat();
13         Cat cat02 = new Cat();
14         System.out.println(" 名字没被修改之前 : ");
15         cat01.sayName();
16         cat02.sayName();
17
18         cat01.name = "Tom"; // 修改名字为 Tom
19         System.out.println(" 名字被修改之后 : ");
20         cat01.sayName();
21         cat02.sayName();
22     }
23 }
24 // 运行后的结果 :
```

```
25 名字没被修改之前：  
26  Kitty  
27  Kitty  
28 名字被修改之后：  
29  Tom  
30  Tom
```

---

### 9.3 构造函数

我们在谈论类的时候，构造函数是一个必然不会拉掉的话题，它是类的一个很重要的特性。每个类都有一个构造函数。如果我们没有写构造函数，编译器会自动构造一个默认的构造函数。

构造函数必须和类名一致，每创建一个新的对象时，至少有一个构造函数将被调用。一个类可以有多个构造函数。

---

```
1  public class Puppy {  
2      public Puppy() {  
3      }  
4      public Puppy(String name) {  
5          //This constructor has one parameter, name.  
6      }  
7  }  
8  }
```

---

Java 支持单例模式，你可以创建一个类的唯一实例，[tbr]

### 9.4 创建一个对象

我们在前面提及了类，类为对象提供蓝图。所以基本上一个对象是通过类创建的，在 Java 中 `new` 关键字被用来创建一个新的对象。

下面是一个对象的创建步骤：

- **声明** 一个被声明为对象的变量
- **实例化** 使用关键字 `new`
- **初始化** `new` 关键字后面跟着一个构造器。这会初始化一个新的对象

下面是一个例子：

---

```
1  public class Puppy {  
2      public Puppy(String name) {  
3          System.out.println("Passed Name is: " + name);  
4      }  
5      public static void main(String args[]) {  
6          Puppy myPuppy = new Puppy("tommy");  
7      }  
8  }
```

---

```

7         }
8     }

```

---

如果我们编译并执行，结果会是：

```

1 Passed Name is: tommy

```

---

## 9.5 访问实例的变量和方法

实例的变量和方法可以通过创建的对象访问。如下所示：

```

1  /* First create an object */
2  ObjectReference = new Constructor();
3  /* Now call a variable as follows */
4  ObjectReference.variableName;
5  /* Now you can call a class method as follows */
6  ObjectReference.MethodName();

```

---

看下面的例子：

```

1  public class Puppy {
2      int puppyAge;
3      public Puppy(String name) {
4          System.out.println("Passed Name is: " + name);
5      }
6      public void setAge(int age) {
7          puppyAge = age;
8      }
9      public int getAge() {
10         System.out.println("Puppy's age is: " + puppyAge);
11         return puppyAge;
12     }
13     public static void main(String args[]) {
14         Puppy myPuppy = new Puppy("tommy");
15         myPuppy.setAge(2);
16         myPuppy.getAge();
17         System.out.println("Variable value: " + myPuppy.puppyAge);
18     }
19 }
20
21 // 以下是运行结果：
22 Passed Name is: tommy
23 Puppy's age is: 2
24 Variable value: 2

```

---

## 9.6 源代码文件声明规则

本章节的最后我们看一下源文件的一些声明规则。这些规则很有必要，特别是当你声明一个类、执行导入、和使用包语句。

- 一个源文件里面最多只能有一个 `public` 类
- `public` 类名应该就是源文件的名称，如果类名为 `public Employee` 那么源文件名就应该为：`Employee.java`
- 如果类定义在一个包内，那么包声明语句应该在文件开头
-