

## 加餐一 | 用一篇文章带你了解专栏中用到的所有Java语法

2019-11-17 王争

设计模式之美

[进入课程 >](#)



讲述：冯永吉

时长 03:16 大小 2.99M



尽管说设计模式跟编程语言没有直接关系，但是，我们也无法完全脱离代码来讲设计模式。我本人熟悉的是 Java 语言，所以专栏中的代码示例我都是用 Java 语言来写的。考虑到有些同学并不熟悉 Java 语言，我今天用一篇文章介绍一下专栏中用到的 Java 语法。

如果你有一定的编程基础，熟悉一门编程语言，结合我今天讲的 Java 语法知识，那看懂专栏中的代码基本不成问题。

如果你熟悉的是 C/C++、C#、PHP，那几乎不用费多大力气，就能看懂 Java 代码。我当时从 C++ 转到 Java，也只看了一天的书，基本语法就全部掌握了。

如果你熟悉的是 Python、Go、Ruby、JavaScript，这些语言的语法可能跟 Java 的区别稍微有些大，但是，通过这篇文章，做到能看懂也不是难事儿。

好了，现在，就让我们一块儿看下，专栏中用到的所有 Java 语言的语法。

## Hello World

我们先来看一下，Java 语言的 Hello World 代码如何编写。

在 Java 中，所有的代码都必须写在类里面，所以，我们定义一个 HelloWorld 类。main() 函数是程序执行的入口。main() 函数中调用了 Java 开发包 JDK 提供的打印函数 System.out.println() 来打印 hello world 字符串。除此之外，Java 中有两种代码注释方式，第一种是 “// 注释...” 双斜杠，表示后面的字符串都是注释，第二种是 “/注释.../” ，表示中间的内容都是注释。

 复制代码

```
1  /*hello world 程序 */
2  public class HelloWorld {
3      public static void main(String []args) {
4          System.out.println("Hello World"); // 打印 Hello World
5      }
6  }
```

## 基本数据类型

Java 语言中的基本数据类型跟其他语言类似，主要有下面几种：

整型类型：byte（字节）、short（短整型）、int（整型）、long（长整型）

浮点类型：float（单精度浮点）、double（双精度浮点）

字符型：char

布尔型：boolean

如下，我们来定义一个基本类型变量：

 复制代码

```
1  int a = 6;
```

除此之外，为了方便我们使用，Java 还提供了一些封装这些基本数据类型的类，这些类实现了一些常用的功能函数，可以直接拿来使用。常用的有下面几个类：

Integer：对应封装了基本类型 int；

Long：对应封装了基本类型 long；

Float：对应封装了基本类型 float；


Double：对应封装了基本类型 double；

Boolean：对应封装了基本类型 boolean；

String：对应封装了字符串类型 char[]。

如下，我们来定义一个 Integer 对象：

```
1 Integer oa = new Integer(6);
```

 复制代码

## 数组


Java 中，我们使用 [] 来定义一个数组，如下所示：

```
1 int a[] = new int[10]; // 定义了一个长度是 10 的 int 类型数组
```

 复制代码

在 Java 中，我们通过如下方式访问数组中的元素：


```
1 a[1] = 3; // 将下标是 1 的数组元素赋值为 3
2 System.out.println(a[2]); // 打印下标是 2 的数组元素值
```

 复制代码

## 流程控制


流程控制语句跟其他语言类似，主要有下面几种。

if-else 语句，代码示例如下所示：

 复制代码

```
1 // 用法一
2 int a;
3 if (a > 1) {
4     // 执行代码块
5 } else {
6     // 执行代码块
7 }
8
9 // 用法二
10 int a;
11 if (a > 1) {
12     // 执行代码块
13 } else if (a == 1) {
14     // 执行代码块
15 } else {
16     // 执行代码块
17 }
```

switch-case 语句，代码示例如下所示：

 复制代码

```
1 int a;
2 switch (a) {
3     case 1:
4         // 执行代码块
5         break;
6     case 2:
7         // 执行代码块
8         break;
9     default:
10        // 默认执行代码
11 }
```

for、while 循环，代码示例如下所示：

 复制代码

```
1 for (int i = 0; i < 10; ++i) {
2     // 循环执行 10 次此代码块
```

```
3 }
4
5 int i = 0;
6 while (i < 10) {
7     // 循环执行 10 次此代码块
8 }
```

continue、break、return，代码示例如下所示：

 复制代码

```
1 for (int i = 0; i < 10; ++i) {
2     if (i == 4) {
3         continue; // 跳过本次循环，不会打印出 4 这个值
4     }
5     System.out.println(i);
6 }
7
8 for (int i = 0; i < 10; ++i) {
9     if (i == 4) {
10        break; // 提前终止循环，只会打印 0、1、2、3
11    }
12    System.out.println(i);
13 }
14
15 public void func(int a) {
16     if (a == 1) {
17         return; // 结束一个函数，从此处返回
18     }
19     System.out.println(a);
20 }
```

## 类、对象

Java 语言使用关键词 class 来定义一个类，类中包含成员变量（也叫作属性）和方法（也叫作函数），其中有一种特殊的函数叫作构造函数，其命名比较固定，跟类名相同。除此之外，Java 语言通过 new 关键词来创建一个类的对象，并且可以通过构造函数，初始化一些成员变量的值。代码示例如下所示：

 复制代码

```
1 public class Dog { // 定义了一个 Dog 类
2     private int age; // 属性或者成员变量
3     private int weight;
4 }
```

```
5     public Dog(int age, int weight) { // 构造函数
6         this.age = age;
7         this.weight = weight;
8     }
9
10    public int getAge() { // 函数或者方法
11        return age;
12    }
13
14    public int getWeight() {
15        return weight;
16    }
17
18    public void run() {
19        // ...
20    }
21 }
22
23 Dog dog1 = new Dog(2, 10); // 通过 new 关键词创建了一个 Dog 对象 dog1
24 int age = dog1.getAge(); // 调用 dog1 的 getAge() 方法
25 dog1.run(); // 调用 dog1 的 run() 方法
```

## 权限修饰符

在前面的代码示例中，我们多次用到 `private`、`public`，它们跟 `protected` 一起，构成了 Java 语言的三个权限修饰符。权限修饰符可以修饰函数、成员变量。

`private` 修饰的函数或者成员变量，只能在类内部使用。


`protected` 修饰的函数或者成员变量，可以在类及其子类内使用。

`public` 修饰的函数或者成员变量，可以被任意访问。

除此之外，权限修饰符还可以修饰类，不过，专栏中所有的类定义都是 `public` 访问权限的，所以，我们可以不用去了解三个修饰符修饰类的区别。

对于权限修饰符的理解，我们可以参看下面的代码示例：

```
1 public class Dog { // public 修饰类
2     private int age; // private 修饰属性，只能在类内部使用
3     private int weight;
4
5     public Dog(int age, int weight) {
```

 复制代码

```
6     this.age = age;
7     this.weight = weight;
8 }
9
10 public int getAge() { //public 修饰的方法，任意代码都是可以调用
11     return age;
12 }
13
14 public void run() {
15     // ...
16 }
17
18 }
```

## 继承

Java 语言使用 `extends` 关键字来实现继承。被继承的类叫作父类，继承类叫作子类。子类继承父类的所有非 `private` 属性和方法。具体的代码示例如下所示：

 复制代码

```
1 public class Animal { // 父类
2     protected int age;
3     protected int weight;
4
5     public Animal(int age, int weight) {
6         this.age = age;
7         this.weight = weight;
8     }
9
10    public int getAge() { // 函数或者方法
11        return age;
12    }
13
14    public int getWeight() {
15        return weight;
16    }
17
18    public void run() {
19        // ...
20    }
21 }
22
23 public class Dog extends Animal { // 子类
24     public Dog(int age, int weight) { // 构造函数
25         super(age, weight); // 调用父类的构造函数
26     }
27
28     public void wangwang() {
```

```
29     //...
30 }
31 }
32
33 public class Cat extends Animal { // 子类
34     public Cat(int age, int weight) { // 构造函数
35         super(age, weight); // 调用父类的构造函数
36     }
37
38     public void miaomiao() {
39         //...
40     }
41 }
42
43 // 使用举例
44 Dog dog = new Dog(2, 8);
45 dog.run();
46 dog.wangwang();
47 Cat cat = new Cat(1, 3);
48 cat.run();
49 cat.miaomiao();
```

## 接口

Java 语言通过 `interface` 关键字来定义接口。接口中只能声明方法，不能包含实现，也不能定义属性。类通过 `implements` 关键字来实现接口中定义的方法。在专栏的第 8 讲中，我们会详细讲解接口，所以，这里我只简单介绍一下语法。具体的代码示例如下所示：

 复制代码

```
1 public interface Runnable {
2     void run();
3 }
4
5 public class Dog implements Runnable {
6     private int age; // 属性或者成员变量
7     private int weight;
8
9     public Dog(int age, int weight) { // 构造函数
10         this.age = age;
11         this.weight = weight;
12     }
13
14     public int getAge() { // 函数或者方法
15         return age;
16     }
17
18     public int getWeigt() {
```



```
19     return weight;
20 }
21
22 @Override
23 public void run() { // 实现接口中定义的 run() 方法
24     // ...
25 }
26 }
```

## 容器


Java 提供了一些现成的容器。容器可以理解为一些工具类，底层封装了各种数据结构。比如 ArrayList 底层就是数组，LinkedList 底层就是链表，HashMap 底层就是散列表等。这些容器我们可以拿来直接使用，不用从零开始开发，大大提高了编码的效率。具体的代码示例如下所示：

 复制代码

```
1 public class DemoA {
2     private ArrayList<User> users;
3
4     public void addUser(User user) {
5         users.add(user);
6     }
7 }
```

## 异常处理

Java 提供了异常这种出错处理机制。我们可以直接使用 JDK 提供的现成的异常类，也可以自定义异常。在 Java 中，我们通过关键字 throw 来抛出一个异常，通过 throws 声明函数抛出异常，通过 try-catch-finally 语句来捕获异常。代码示例如下所示：

 复制代码

```
1 public class UserNotFoundException extends Exception { // 自定义一个异常
2     public UserNotFoundException() {
3         super();
4     }
5
6     public UserNotFoundException(String message) {
7         super(message);
8     }
9
10    public UserNotFoundException(String message, Throwable e) {
```

```

11     super(message, e);
12 }
13 }
14
15 public class UserService {
16     private UserRepository userRepo;
17     public UserService(UserRepository userRepo) {
18         this.userRepo = userRepo;
19     }
20
21     public User getUserById(long userId) throws UserNotFoundException {
22         User user = userRepo.findUserById(userId);
23         if (user == null) { // throw 用来抛出异常
24             throw new UserNotFoundException();// 代码从此处返回
25         }
26         return user;
27     }
28 }
29
30 public class UserController {
31     private UserService userService;
32     public UserController(UserService userService) {
33         this.userService = userService;
34     }
35
36     public User getUserById(long userId) {
37         User user = null;
38         try { // 捕获异常
39             user = userService.getUserById(userId);
40         } catch (UserNotFoundException e) {
41             System.out.println("User not found: " + userId);
42         } finally { // 不管异常会不会发生, finally 包裹的语句块总会被执行
43             System.out.println("I am always printed.");
44         }
45         return user;
46     }
47 }

```

## package 包

Java 通过 package 关键字来分门别类地组织类，通过 import 关键字来引入类或者 package。具体的代码示例如下所示：

```

1  /*class DemoA*/
2  package com.xzg.cd; // 包名 com.xzg.cd
3
4  public class DemoA {

```

 复制代码

```
5    //...
6 }
7
8 /*class DemoB*/
9 package com.xzg.alg;
10
11 import java.util.HashMap; // Java 工具包 JDK 中的类
12 import java.util.Map;
13 import com.xzg.cd.DemoA;
14
15 public class DemoB {
16     //...
17 }
```

## 总结

今天，我带你一块学习了专栏中用到的所有的 Java 基本语法。不过，我希望你不要纠结于专栏或者某某书籍到底是用什么编程语言来写的。语言层面的东西完全不会限制我的讲解和你的理解。这就像我们读小说一样，不管它是用英语写的，还是中文写的，故事都可以同样精彩。而且，多了解一些 Java 语法，对于你今后阅读 Java 语言编写的书籍或者文档，也很有帮助。

实际上，我之前在 Google 工作的时候，大家都不太在意自己熟悉的是哪种编程语言，很多同事都是“现学现卖”，什么项目适合用什么语言就现学什么语言。除此之外，Google 在招聘的时候，也不限定候选人一定要熟悉哪种编程语言，也很少问跟语言特性相关的问题。因为他们觉得，编程语言只是一个工具，对于一个有一定学习能力的人，学习一门编程语言并不是件难事。

除此之外，对于专栏中的代码示例，你也可以用你熟悉语言重新实现一遍，我相信这也是件很有意义的事情，也更能加深你对内容的理解。

## 课堂讨论

不同的公司开发使用的编程语言可能不一样，比如阿里一般都是用 Java，今日头条用 Go、C++ 比较多。在招聘上，这些公司都倾向于招聘熟悉相应编程语言的同学，毕竟熟练掌握一门语言也是要花不少时间的，而且用熟悉的编程语言来开发，肯定会更得心应手，更不容易出错。今天课堂讨论的话题有两个：

1. 分享一下你学习一门编程语言的经历，从入门到熟练掌握，大约花了多少的时间？有什么好的学习编程语言的方法？
2. 在一个程序员的技术能力评价体系中，你觉得“熟练使用某种编程语言”所占的比重有多大？

欢迎在留言区写下你的想法，和同学一起交流和分享。如果有收获，也欢迎你把这篇文章分享给你的朋友。

小程序学习打卡邀请

## 8个月，攻克设计模式



扫一扫参与小程序打卡



新版升级：点击「👤请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 06 | 理论三：面向对象相比面向过程有哪些优势？面向过程真的过时了吗？

下一篇 07 | 理论四：哪些代码设计看似是面向对象，实际是面向过程的？

### 精选留言 (57)

写留言



辣么大

2019-11-17

Java用的时间最长，大概4-5年，不敢说自己“熟练”掌握了。最近反而觉得不懂的更多了。我没有抓入Java8不放，而是跟着Java的发展，开始学习Java11和Java13的新特性，紧跟语言的变化，并学习虚拟机和并发编程的相关知识。

我觉得熟练使用某种编程语言，在技术能力评价中占比起码50%。因为“熟练”是衡量...

展开 ▾

💬 4

👍 26



**编程界的小学生**

2019-11-17

原谅我这篇文章三十秒就看完了，因为我是JAVA

1.用了多久我也不确定，但是学习方法是有的，首先看视频，资料，动手敲，晚上睡觉前在脑海里回顾一下学了什么，明天在动手敲一遍昨天学的内容，最后用自己的语言将其组织成一篇属于自己的文章。

2.熟练需要看成都，就比如很多人都说看源码感觉没用，看了就忘，也不知道能干嘛。我...

展开 ▾

💬 1

👍 7



**Y024**

2019-11-17

Day013 加餐一

勘误：第二种是 `"/注释.../"`

应为：第二种是 `"/* 注释...*/"`

展开 ▾

作者回复: 嗯嗯 多谢指出



💬

👍 6



**李小四**

2019-11-17

从第一次接触Java，到得心应手，大概花了两年时间。这个周期让我理解了学习的非线性。

大一一开始学习C语言，学的似懂非懂，做了课程设计就放下了，发大二开始学Java，同样似懂非懂。大三开始接触Android开发，用到了Java，才发现自己Java知识不足，于是花时间重学了Java，过程中发现有些东西不理解，又穿插着把C需要的指针内存啃了几遍，大...

展开 ▾

💬

👍 4



**小马哥**

2019-11-17

大学课程中学习了C，工作中自学并使用JAVA，主要用于web和大数据开发，JAVA不仅仅是一门语言，还是一个技术体系，包括了后来的很多技术框架，学习JAVA语言如果有其他

语言基础是很快的，精通后面的一些常用框架就需要一些设计模式的积累。所以还是学习能力最重要：学习，操练，总结，分享，这个路线是我认为很快捷的学习方法。最后学习的东西越多，越容易融会贯通，后来使用Python做推荐系统，我们几个JAVA开发人员， ...  
展开 ▾



2



**斜杠ing...**

2019-11-17

语言虽然只是个工具，但对于大多数人来说，几乎就用一种语言，还是精通后才能说语言是相通的。



2



**William**

2019-11-17

居然有加餐，666.

一篇文章涵盖java 语法，赞.

另外补充一下，关于权限限定符， ...

展开 ▾



2



**Flynn**

2019-11-18

“熟练使用某种编程语言” 所占的比重：入门程序员80%，之后的程序员20%



1



**Chandler**

2019-11-18

感觉学编程还是得注重语言特性的学习，编程语言有很多种，但语言特性就那么些东西。推荐一篇关于如何学习，选择编程语言的文章。

<http://www.yinwang.org/blog-cn/2017/07/06/master-pl>

展开 ▾



1



**稳**

2019-11-17

1、拿个人Python的经历，入门3天，熟练的话需要至少写一个项目，工作写了2年也不敢吹精通。我觉得学习一门语言，入门时一定要打牢基础，把基础语法耐心看仔细了，以后才会少踩坑，书籍专栏视频都可以。熟练的话一定要多练，主要时熟悉常用库和生态，精

通就需要研究源码和不断的思考了。

2、虽然语言是工具，我觉得必须要有一门足够熟练的编程语言，这样整个人思维深度都...

展开 ▾



1



**相逢是缘**

2019-11-17

非计算机专业，在学校就学习了C语言，在工作中也一直使用C。后来自己看了C++，JAV A。学习一门语言最开始就是一些基本语法和数据结构，了解了这两个就可以自己调试一些简单的demo。接下来就是你的程序不能只在控制台打印信息，要能对外输入和输出。这块一般是两部分:1、还是在自己的计算机内部，能从磁盘读取和保存数据。2、能够和别的计算机通信，也就是socket http等网络编程。再接下来就是一些每个编程语言自己特有的...

展开 ▾



1



**黄林晴**

2019-11-17

打卡

学习编程语言是持续的过程，可能两年前我觉得我对某某熟悉了，后来在工作不断的积累总结，越来越发现自己所掌握的是多么的皮毛，都不敢说自己熟练掌握，希望跟着课程一起进步！

展开 ▾



1



**helloworld**

2019-11-18

当项目中需要新的编程语言时现学现卖，说是这么说，但是万一遇到了需要使用新语言开发的项目，一是能不能过了自己的心里的关（每个人在潜意识中会根据自己所经常使用的编程语言而将自己归类），二是目前大多数领导看的是结果和效率吧，如果没有提前学习过这门语言，效率会低很多吧，几乎得面向google编程了

展开 ▾



**被水淹没**

2019-11-18

高中学了vb，以为大多编程语言都差不多，基本的变量、控制语句都差不多  
然后去学C，看到后面... 我擦这指针啥玩意？看来得补一下指针，然后搜了下.. 厚厚的一本指针教程... 然后就被打击到了  
直到大学学了java，没指针，舒坦... 大学完就只会ssh，ssm，还好实习找得到工作，现在努力提升自己，先把争哥的这2套课程学了先 ^-^

展开 ▾



**Dimple**

2019-11-18

楼上的留言都写的挺好的，我都不知道怎么补充。我一开始学的是C语言，后来接触上了Android，就一直把Java当成了主力语言，其实我工作之后的半年，才真正理解什么是面向对象，什么是面向对象编程，今天学习 最新的07课程，才知道，我的面相对象也还是片面，尴尬。

...

展开 ▾



**WIZ**

2019-11-18

打卡

展开 ▾



**EidLeung**

2019-11-18

木有讲泛型 😊

展开 ▾

作者回复: 因为专栏中没有用到 毕竟泛型不是所有的语言都支持 而且稍微还有点复杂



**迁橘**

2019-11-18

之前我还在想 把所有的参数之类的常量写到一个类中，觉得写很多个配置类不好，感觉好麻烦，最后没写是因为量有点多，哈哈，现在看来。还好没写。听了好几遍，开始有点费解，慢慢的理解了老师所讲的。



**逆流的鱼**

2019-11-18

强烈建议加餐其他语言

展开 ▾





**Lyre**

2019-11-18

这篇有点混😏

展开▼

作者回复: 😏 这是加餐啊 大哥

