Atitit java 新特性10 11 12 13 14 15

目录

[1.1. java9--11 1](#_Toc7029)

[1.2. java11 2](#_Toc22818)

[2. Java13 3](#_Toc10347)

[2.1. Switch 表达式标签简化形式 3](#_Toc28308)

[2.2. 文本块（预览功能） 3](#_Toc22844)

[3. JDK/Java 14 可能带来什么新特性？ - OSCHINA - 中文开源技术交流社区 4](#_Toc5709)

[4. Jdk15 4](#_Toc2735)

[5. Ref Java 5，6，7，8，9，10，11新特性吐血总结 - 简书 4](#_Toc697)

# Java17LTS

果Java 17确实成为下一个LTS版本，社区可能会决定“半面包总比没有好”-它会具有一些不错的功能（类数据共享，JFR流，记录，切换表达式等）

从Java 16：

* [JEP 394](https://openjdk.java.net/jeps/394)：instanceof的模式匹配
* [JEP 395](https://openjdk.java.net/jeps/395)：记录

这几年的Java版本中，就缺乏这种重大功能的升级了，我把我有点印象的功能升级列一下：

|  |  |
| --- | --- |
| **版本** | **重要特性** |
| Java 9 | ****模块化系统****  Java Shell  :交互式命令行控制台 |
| Java 10 | ****局部变量类型推导**** |
| Java 11 | 各种加强：Stream , String, Collection  标准化的HTTP Client  ****ZGC**** |
| Java 12 | Switch表达式扩展 |
| Java 13 | 重新实现Socket API  文本块(多行字符串) |

## java9--11

1. java模块系统 （Java Platform Module System）。
2. 新的版本号格式。$MAJOR.$MINOR.$SECURITY.$PATCH
3. java shell，交互式命令行控制台。
4. 在private instance methods方法上可以使用@SafeVarargs注解。
5. diamond语法与匿名内部类结合使用。
6. 下划线\_不能单独作为变量名使用。
7. 支持私有接口方法(您可以使用diamond语法与匿名内部类结合使用)。

#### java10

局部变量类型推断（Local-Variable Type Inference）

var是一个保留类型名称，而不是关键字。所以之前使用var作为变量、方法名、包名的都没问题，但是如果作为类或接口名，那么这个类和接口就必须重命名了。

var的使用场景主要有以下四种：

* 1. 本地变量初始化。
  2. 增强for循环中。
  3. 传统for循环中声明的索引变量。
  4. Try-with-resources 变量。​

Optional类添加了新的方法orElseThrow(无参数版)。相比于已经存在的get方法，这个方法更推荐使用。

## java11 LTs

1. 。
2. Lambda参数的局部变量语法。java10中引入的var字段得到了增强，现在可以用在lambda表达式的声明中。如果lambda表达式的其中一个形式参数使用了var，那所有的参数都必须使用var。

直接使用var定义，而不用写具体的类型

Java 11 增加了一系列的字符串处理方法

InputStream 加强

InputStream 终于有了一个非常有用的方法：transferTo，可以用来将数据直接传输到 OutputStream，这是在处理原始数据流时非常常见的一种用法，如下示例。

化繁为简，一个命令编译运行源代码

看下面的代码。

// 编译

javac Javastack.java

// 运行

java Javastack

在我们的认知里面，要运行一个 Java 源代码必须先编译，再运行，两步执行动作。而在未来的 Java 11 版本中，通过一个 java 命令就直接搞定了，如以下所示。

java Javastack.java

Java REPL (JShell)

# Java13

在 Java 12 之后，关于 Switch 表达式的写法改进为如下：

## Switch 表达式****标签简化形式****

private static String getText(int number) {

String result = switch (number) {

case 1, 2 -> "one or two";

case 3 -> "three";

case 4, 5, 6 -> "four or five or six";

default -> "unknown";

};

return result;

}

## 文本块（预览功能）

一直以来，Java 语言在定义字符串的方式是有限的，字符串需要以双引号开头，以双引号结尾，这导致字符串不能够多行使用，而是需要通过换行转义或者换行连接符等方式来变通支持多行，但这样会增加编辑工作量，同时也会导致所在代码段难以阅读、难以维护。

Java 13 引入了文本块来解决多行文本的问题，文本块以三重双引号开头，并以同样的以三重双引号结尾终止，它们之间的任何内容都被解释为字符串的一部分，包括换行符，避免了对大多数转义序列的需要，并且它仍然是普通的 java.lang.String 对象，文本块可以在 Java 中可以使用字符串文字的任何地方使用，而与编译后的代码没有区别，还增强了 Java 程序中的字符串可读性。并且通过这种方式，可以更直观地表示字符串，可以支持跨越多行，而且不会出现转义的视觉混乱，将可以广泛提高 Java 类程序的可读性和可写性。

# JDK/Java 14 可能带来什么新特性？ - OSCHINA - 中文开源技术交流社区

# Jdk15

#### **378:Text Blocks**

文本块，是一个多行字符串，它可以避免使用大多数转义符号，自动以可预测的方式格式化字符串，并让开发人员在需要时可以控制格式。

文本块最早准备在 JDK 12 添加的，但最终撤消了，然后在 JDK 13 中作为预览特性进行了添加，然后又在 JDK 14 中再次预览，在 JDK 15 中，文本块终于转正，暂不再做进一步的更改。

来看下这个示例你就懂了：

# Ref Java 5，6，7，8，9，10，11新特性吐血总结 - 简书

Java 13 新特性概述 – IBM Developer

Java 15 正式发布， 14 个新特性，刷新你的认知！！ - Java技术栈 - 博客园