## 18.4 Java12

美国当地时间 2019 年 3 月 19 日，也就是北京时间 20 号 Java 12 正式发布了，Java12也不是 LTS版本，总共有8个新的JEP(JDK Enhancement Proposals)

#### 1.switch表达式（预览）

Switch 表达式也是作为预览语言功能的第一个语言改动被引入新版 Java 中来的，预览语言功能的想法是在 2018 年初被引入 Java 中的，本质上讲，这是一种引入新特性的测试版的方法。通过这种方式，能够根据用户反馈进行升级、更改，在极端情况下，如果没有被很好的接纳，则可以完全删除该功能。预览功能的关键在于它们没有被包含在 Java SE 规范中。

传统的switch声明语句(switch statement)在使用中有一些问题：

* 匹配是自上而下的，如果忘记写break, 后面的case语句不论匹配与否都会执行；
* 所有的case语句共用一个块范围，在不同的case语句定义的变量名不能重复；
* 不能在一个case里写多个执行结果一致的条件；
* 整个switch不能作为表达式返回值；

Java 12将会对switch声明语句进行扩展，可将其作为增强版的 switch 语句或称为 "switch 表达式"来写出更加简化的代码。

扩展的 switch 语句，不仅可以作为语句（statement），还可以作为表达式（expression），并且两种写法都可以使用传统的 switch 语法，或者使用简化的“case L ->”模式匹配语法作用于不同范围并控制执行流。这些更改将简化日常编码工作，并为 switch 中的模式匹配（JEP 305）做好准备。

* 使用 Java 12 中 Switch 表达式的写法，省去了 break 语句，避免了因少写 break 而出错。
* 同时将多个 case 合并到一行，显得简洁、清晰也更加优雅的表达逻辑分支，其具体写法就是将之前的 case 语句表成了：case L ->，即如果条件匹配 case L，则执行标签右侧的代码 ，同时标签右侧的代码段只能是表达式、代码块或 throw 语句。
* 为了保持兼容性，case 条件语句中依然可以使用字符 : ，这时 fall-through 规则依然有效的，即不能省略原有的 break 语句，但是同一个 Switch 结构里不能混用 -> 和 : ，否则会有编译错误。并且简化后的 Switch 代码块中定义的局部变量，其作用域就限制在代码块中，而不是蔓延到整个 Switch 结构，也不用根据不同的判断条件来给变量赋值。

代码示例：

Java12之前：

@Test

**public** **void** testBefore12() {

**int** month = 3;

**switch**(month) {

**case** 3:

**case** 4:

**case** 5:

System.***out***.println("春季");

**break**;

**case** 6:

**case** 7:

**case** 8:

System.***out***.println("夏季");

**break**;

**case** 9:

**case** 10:

**case** 11:

System.***out***.println("秋季");

**break**;

**case** 12:

**case** 1:

**case** 2:

System.***out***.println("冬季");

**break**;

**default**:

System.***out***.println("月份输入有误！");

}

}

Java12之后：

@Test

**public** **void** testAfter12() {

**int** month = 3;

**switch**(month) {

**case** 3,4,5 -> System.***out***.println("春季");

**case** 6,7,8 -> System.***out***.println("夏季");

**case** 9,10,11 -> System.***out***.println("秋季");

**case** 12,1,2 -> System.***out***.println("冬季");

**default**->System.***out***.println("月份输入有误！");

};

}

Java12之后的switch还可以作为表达式使用：

@Test

**public** **void** testAfter12Exp() {

**int** month = 3;

String season = **switch**(month) {

**case** 3,4,5 ->"春季";

**case** 6,7,8 -> "夏季";

**case** 9,10,11 -> "秋季";

**case** 12,1,2 -> "冬季";

**default**->**throw** **new** IllegalArgumentException("月份输入有误！");

};

}

这个语法如果做过Android开发的不会陌生，因为Kotlin 家的 when 表达式就是这么干的！Switch Expressions 或者说起相关的 Pattern Matching 特性，为我们勾勒出了 Java 语法进化的一个趋势，将开发者从复杂繁琐的低层次抽象中逐渐解放出来，以更高层次更优雅的抽象，既降低代码量，又避免意外编程错误的出现，进而提高代码质量和开发效率。

## 18.5 Java13

美国当地时间2019年9月17日，即北京时间18日国际知名的OpenJDK开源社区发布了Java编程语言环境的最新版本OpenJDK13。Java13也不是 LTS版本，总共有5个新的JEP(JDK Enhancement Proposals)。

* 语法层面，改进了Switch Expressions，新增了Text Blocks，二者皆处于Preview状态；
* API层面主要使用NioSocketImpl来替换JDK1.0的PlainSocketImpl
* GC层面则改进了ZGC，以支持Uncommit Unused Memory

#### switch表达式（预览）

在JDK 12中引入了Switch表达式作为预览特性。JDK 13提出了第二个switch表达式预览。JEP 354修改了这个特性，它引入了yield语句，用于返回值。这意味着，switch表达式(返回值)应该使用yield, switch语句(不返回值)应该使用break

在以前，我们想要在switch中返回内容，还是比较麻烦的，一般语法如下：

@Test  
**public void** testBeforeSwitch12(){  
 **int** week = 2;  
 String weekName = **""**;  
 **switch**(week){  
 **case** 1:  
 weekName = **"Monday"**;  
 **break**;  
 **case** 2:  
 weekName = **"Tuesday"**;  
 **break**;  
 **case** 3:  
 weekName = **"Wednesday"**;  
 **break**;  
 **case** 4:  
 weekName = **"Thursday"**;  
 **break**;  
 **case** 5:  
 weekName = **"Friday"**;  
 **break**;  
 **case** 6:  
 weekName = **"Saturday"**;  
 **break**;  
 **case** 7:  
 weekName = **"Suday"**;  
 **break**;  
 **default**:  
 weekName = **"Error"**;  
 }  
 System.***out***.println(weekName);  
}

现在我们可以使用yield返回值：

@Test  
**public void** testAfterSwitch13(){  
 **int** week = 12;  
 String weekName =**switch**(week){  
 **case** 1:  
 **yield "Monday"**;  
 **case** 2:  
 **yield "Tuesday"**;  
 **case** 3:  
 **yield "Wednesday"**;  
 **case** 4:  
 **yield "Thursday"**;  
 **case** 5:  
 **yield "Friday"**;  
 **case** 6:  
 **yield "Saturday"**;  
 **case** 7:  
 **yield "Suday"**;  
 **default**:  
 **yield "Error"**;  
 };  
 System.***out***.println(weekName);  
}

在这之后，switch中就多了一个关键字用于跳出switch块了，那就是yield，他用于返回一个值。和return的区别在于：return会直接跳出当前循环或者方法，而yield只会跳出当前switch块。