链滴

JDK17 你必须知道的一些新特性

作者：luofeng0603

原文链接：https://ld246.com/article/1683253173777

来源网站：链滴

前言

之前一直用的是jdk8，不过新项目上了Spring Boot3.0，必须要用jdk17了，所以了解下jdk8之后的些新特性吧~

本地变量var

jdk10提供的:

```
#原先我们需要这么定义
Test t = new Test();
#现在这样定义
var t1 = new Test();
```

其实就是一个语法糖，虚拟机可不认识var，编译成字节码的时候，还是会找到变量的真正类型的。跟java的泛型机制差不多。

Switch表达式

jdk13引入了yield，先来看下我们原先怎么写的：

```
int i;
switch(x) {
    case "1":
        i = 1;
    break;
    case "2":
    i = 2;
    break;
    case "3":
```

```
    i = 3;
    break;
    default:
    i = x.length();
}
```

jdk13我们可以写成：

```
int i = switch(x) {
    case "1" -> 1;
    case "2" -> 2;
    case "3" -> 3;
    default -> {
        int len = x.length();
     yield len;
    }
}
```

或者：

```
int i = switch(x) {
    case "1": yield 1;
    case "2": yield 2;
    case "3": yield 3;
    default -> {
        int len = x.length();
     yield len;
    }
}
```

这里的yield只是用来返回一个值，跟return不太一样，yield只会跳出当前的switch块。

Text Blocks

jdk13提供，现在写多行字符串方便多了，看下面的例子：

```
"""
    <html>
    <head>Title</head>
    <body>
        <p>Body</p>
      </body>
    </html>
"""
```

Records

jdk14新特性，用来申明一个简单类的，这个类里面只有字段，编译器会自动创建所有方法并让所有段参与hashCode等方法。

```
record Person(String name, Integer age){}
```

封闭类

很多场景中，我们写的接口或者父类，只允许我们指定范围的类去实现或者继承，这个时候就可以用封闭类。关键字是 sealed，permits

```
public sealed interface Service permits Car,Truck { }
```

定义了一个封闭接口，只允许Car、Truck来实现。

继承也是一样的

```
public abstract sealed class Vechile permits Car,Truck { }
```

定义了一个抽象类 Verchile只允许 Car和Truck来继承

instanceof模式匹配

我们以前一般在强转之前用来做判断的，比如：

```
if(animal instanceof Cat) {
    Cat cat = (Cat) animal;
    cat.eat();
} else if (animal instanceof Dog) {
    Dog dog = (Dog) animal;
    dog.eat();
```

jdk14中可以简写了：

```
if(animal instanceof Cat) {
    cat.eat();
} else if (animal instanceof Dog) {
    dog.eat();
}
```

强转那部分代码可以不用写了，更加的直观。

Switch模式匹配

基于instanceof模式匹配这个特性，我们可以使用如下方式来处理对象o

```
public static String formatter(Object o) {
    String formatted = "";
    if(o instanceof Integer i) {
    formatted = String.format("int %d", i);
    } else if (o instanceof Long l) {
    formatted = String.format("long %d", l);
    }
    ......
}
```

jdk17中switch中的case还可以模式匹配，我们可以通过switch改写如下：

```
public static formatterPatternSwitch(Object o){
    return switch(o) {
    case Integer i -> String.format("int %d",i);
    case Long l -> String.format("long %d",l);
```

原文链接：JDK17 你必须知道的一些新特性

```
    ......
    default -> o.toString();
    }
}
```

switch处理的是Object，而且也不是精确匹配，而是模式匹配了。

差不多就这些~