# Java 14 & 15 New Features Overview

Guide Java New Java Features About 1599 words About 5 minutes

## Java 14

### Null pointer exception accurate prompt

By adding it to the JVM parameters `-XX:+ShowCodeDetailsInExceptionMessages`, you can get more detailed call information in the null pointer exception, and locate and solve the problem more quickly.

```java
a.b.c.i = 99; // 假设这段代码会发生空指针
```

Before Java 14:

```java
Exception in thread "main" java.lang.NullPointerException
    at NullPointerExample.main(NullPointerExample.java:5)
```

After Java 14:

```java
 // 增加参数后提示的异常中很明确的告知了哪里为空导致
Exception in thread "main" java.lang.NullPointerException:
        Cannot read field 'c' because 'a.b' is null.
    at Prog.main(Prog.java:5)
```

### Switch enhancement (forwarding)

The switch feature (preview feature) introduced in Java 12 became a formal version in Java 14. It does not require adding parameters to enable it and can be used directly in JDK 14.

Java 12 introduced a lambda-like syntax for switch expressions, which allows executing blocks after a condition is successfully matched. This eliminates the need to write break statements. Java 13 provides `yield` a return value within a block.

```java
String result = switch (day) {
            case "M", "W", "F" -> "MWF";
            case "T", "TH", "S" -> "TTS";
            default -> {
                if(day.isEmpty())
                    yield "Please insert a valid day.";
                else
                    yield "Looks like a Sunday.";
            }

        };
System.out.println(result);
```

# Preview new features

## record keyword

`record` The keyword can simplify the definition of **data classes** (a Java class cannot be modified once it is instantiated). By using `record` instead of `class` defining a class, you only need to declare the attributes and you can get the attribute access methods, as well as `toString()` , `hashCode()` , `equals()` methods.

It is similar to `class` defining a class with , but with the lombok plugin and annotations `@Getter,@ToString,@EqualsAndHashCode` .

```java
/**
 * 这个类具有两个特征
 * 1. 所有成员属性都是final
 * 2. 全部方法由构造方法，和两个成员属性访问器组成（共三个）
 * 那么这种类就很适合使用record来声明
 */
final class Rectangle implements Shape {
    final double length;
    final double width;

    public Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double length() { return length; }
```

```
    double width() { return width; }
}
/**
 * 1. 使用record声明的类会自动拥有上面类中的三个方法
 * 2. 在这基础上还附赠了equals(), hashCode()方法以及toString()方法
 * 3. toString方法中包括所有成员属性的字符串表示形式及其名称
 */
record Rectangle(float length, float width) { }
```

### Text Block

In Java 14, text blocks are still a preview feature, but they introduce two new escape characters:

- `\` : indicates the end of the line, without introducing a newline character
- `\s` : represents a single space

```java
String str = "凡心所向，素履所往，生如逆旅，一苇以航。";

String str2 = """
        凡心所向，素履所往， \
        生如逆旅，一苇以航。""";
System.out.println(str2);// 凡心所向，素履所往， 生如逆旅，一苇以航。
String text = """
        java
        c++\sphp
        """;
System.out.println(text);
//输出:
java
c++ php
```

### instanceof enhancements

It is still **a preview feature** and was introduced in Java 12 New Features .

## other

- ZGC, introduced in Java 11, is the next-generation GC algorithm after G1. It supports Linux platforms and has been supported on macOS and Windows since Java 14.

(Personally, I feel that I can finally experience the effects of ZGC in my daily development tools, although G1 is actually sufficient.)
- Removed the CMS (Concurrent Mark Sweep) garbage collector (successfully retired)
- Added jpackage tool, which not only packages the application into jar package as standard, but also supports feature packages of different platforms, such as linux `deb` and `rpm`, `msi` and `.exe`

# Java 15

## CharSequence

`CharSequence` The interface adds a default method `isEmpty()` to determine whether the character sequence is empty and returns true if it is.

```java
public interface CharSequence {
  default boolean isEmpty() {
      return this.length() == 0;
  }
}
```

## TreeMap

`TreeMap` The following new methods have been introduced:

- `putIfAbsent()`
- `computeIfAbsent()`
- `computeIfPresent()`
- `compute()`
- `merge()`

## ZGC (regularization)

When Java 11 was released, ZGC was still in the experimental stage.

At that time, the emergence of ZGC allowed many Java developers to see another possibility for garbage collectors, and therefore attracted much attention.

After multiple versions of iteration, continuous improvement and problem fixing, ZGC is now officially available in Java 15!

However, the default garbage collector is still G1. You can start ZGC with the following parameters:

```bash
java -XX:+UseZGC className
```

## EdDSA (Digital Signature Algorithm)

A new digital signature algorithm based on the Edwards-Curve Digital Signature Algorithm (EdDSA) has been added with enhanced security and performance.

Although its performance is better than the existing ECDSA implementation, it will not completely replace the existing Elliptic Curve Digital Signature Algorithm (ECDSA) in the JDK.

```java
KeyPairGenerator kpg = KeyPairGenerator.getInstance("Ed25519");
KeyPair kp = kpg.generateKeyPair();

byte[] msg = "test_string".getBytes(StandardCharsets.UTF_8);

Signature sig = Signature.getInstance("Ed25519");
sig.initSign(kp.getPrivate());
sig.update(msg);
byte[] s = sig.sign();

String encodedString = Base64.getEncoder().encodeToString(s);
System.out.println(encodedString);
```

Output:

```
0Hc0lxxASZNvS52WsvnncJOH/mlFhnA8Tc6D/k5DtAX5BSsNVjtPF4R4+yMWXVjrvB2m
```

## Text block (regularization)

In Java 15, text blocks are an official feature.

## Hidden Classes

Hidden classes are designed for frameworks. Hidden classes cannot be used directly by the bytecode of other classes. They can only be generated at runtime and used indirectly through reflection.

## Preview new features

### Sealing

**Sealed Classes** is a preview feature in Java 15.

Before sealed classes, in Java, if you wanted to prevent a class from being inherited or modified, you could `final` modify it with the `` keyword. However, this approach was not very flexible and directly blocked the inheritance and modification channels of a class.

Sealed classes can restrict the classes that inherit or implement them, so that the class can only be inherited by the specified class.

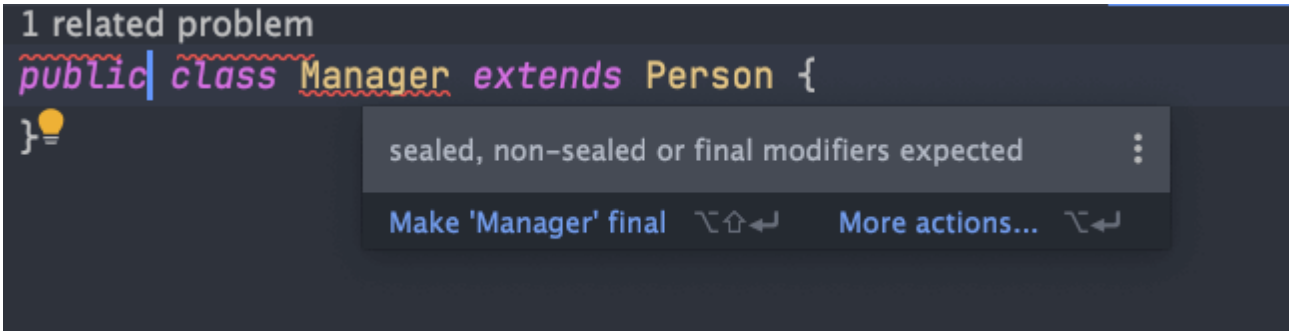
```java
// 抽象类 Person 只允许 Employee 和 Manager 继承。
public abstract sealed class Person
    permits Employee, Manager {

    //...
}
```

Additionally, any class that extends a sealed class must itself be declared as `sealed`, `non-sealed` or `final`.

```
public final class Employee extends Person {
}

public non-sealed class Manager extends Person {
}
```




If the subclasses that are allowed to be extended and the enclosing class are in the same source code file, the enclosing class does not need to use the permits statement. The Java compiler will retrieve the source file and add the permitted subclasses to the enclosing class during compilation.

### instanceof pattern matching

Java 15 has not adjusted this feature and continues to preview it, mainly to receive more usage feedback.

In future Java versions, Java aims to continue to improve `instanceof` pattern matching new features.

## other

- **The Nashorn JavaScript engine has been completely removed** : Nashorn was introduced in Java 8. Java 9 made some enhancements to Nashorn, implementing some new ES6 features. It was deprecated in Java 11 and completely removed in Java 15.
- **DatagramSocket API refactoring**
- **Disabling and deprecating biased locking** : The introduction of biased locking increases the complexity of the JVM by more than the performance gains it brings. However, you can still use `-XX:+UseBiasedLocking` Enable biased locking, but it will indicate that it is a deprecated API.
- ...

Recently Updated2023/12/30 16:14
Contributors: guide , Guide , Mr.Hope , paigeman