NOVEDADES DE JAVA

Trabajo Realizado por: Álvaro Castilla Cano

Programación

NOVEDADES DE JAVA

---

1. NOVEDADES JAVA 13

Java 13 incorpora algunas características nuevas que facilitan la lectura del código, entre las más destacadas están los bloques de texto y las expresiones switch mejoradas.

1.1. BLOQUES DE TEXTO

Para definir una cadena de caracteres que tuviese varias líneas en Java había que emplear concatenación de cadenas. Cuando una cadena comenzaba con una comilla doble, cuando tenía un salto de línea, debíamos de emplear el carácter escape de salto de línea \n.

```
System.out.println("Hola me llamo Álvaro "
                                + "y estoy realizando un proyecto de programación");
```

En el ejemplo anterior, aunque el texto esté en diferentes líneas, a la hora de mostrarlo, aparecen ambas en la misma línea. Posteriormente a java 13, para solucionar este problema usábamos \n como ya comenté anteriormente.

```
System.out.println("Hola me llamo Álvaro \n"
                                + "y estoy realizando un proyecto de programación");
```

Sin embargo, en java 13 han implementado una solución más sencilla y eficaz. Esta nos permite poner una cadena de caracteres en diferentes líneas ausentando la necesidad del uso de \n. Está nueva forma se una poniendo tres comillas doble al comienzo de una cadena de caracteres, por ejemplo:

```
System.out.println("""
                                Hola me llamo Álvaro y estoy realizando
                                un proyecto de programación
                                """);
```

Cuando mostremos esta cadena de caracteres aparecerán en distintas líneas.

1.2. EXPRESIONES SWITCH MEJORADAS

En las novedades de Java 12 implementaron la posibilidad de que los switch fueran expresiones que retornan un valor en vez de una sentencia y se evitaba el uso de la palabra reservada break.
En Java 13 en vez de solo poder poner un único valor, se permite crear bloques de sentencias para cada *case* y retornar el valor con la palabra reservada *yield.*

```
double factura  = 30;
        int envio = 5;
        int descuento = 0;

        descuento=switch(envio) {
            case 1,2,3->{
                int op = envio*2;
                yield op;
            }
            case 4,5 ->{
                int op=envio*+1;
                yield op;
            }
            default -> 5;
        };
        System.out.println(factura*100/(100+descuento));
```

Como podemos ver en ejemplo anterior, cuando el envio sea 1, 2, o 3 devolvera un valor para descuento, y cuando sea 4 o 5 devolverá otro valor.
En conclusión, break devuelve sentencias, y yield devuelve valores.

2. NOVEDADES JAVA 14

2.1. EXCEPCIONES NullPointerException

Cuando usas referencia de objeto cuyo valor es null, Java emite un NullPointerException, este indica la línea de código donde se ha producido el error, la clase, y el método donde se ha intentado hacer referencia. Por ejemplo:

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

Sin embargo, hay casos en el que las trazas no son lo suficientemente precisas para determinar la causa sin el debugger. A partir de Java 14, las excepciones NullPointerException son más útiles e indican de forma más precisa que ha producido la excepción y dónde se ha producido. Por ejemplo:

```
Exception in thread "main" java.lang.NullPointerException:
        Cannot assign field "i" because "a" is null
    at Prog.main(Prog.java:5)
```

2.2. EXPRESIONES SWITCH

Las características de expresiones switch que fueron introducidas de forma previa en Java 12 y Java 13 se califican como estándar.

2.3. RECORDS

Esta característica destaca por permitirnos reducir significativamente el código necesario para algunas clases. Evitan mucho del código que es necesario en Java para definir los constructores, los métodos Getters and Setters, toString(), e implementar de forma correcta los métodos equals y hashCode.

NOVEDADES DE JAVA

Cuando una clase es record, adquiere automáticamente los siguientes miembros:

- Los atributos de la clase de forma privada.
- Constructores.
- Getters and Setters .
- toString().
- equals y hashCode.

Sin embargo, esto también contiene restricciones:

- No se puede extender a ninguna otra clase, y no se pueden declarar campos que no sean privados.
- Los registros son implícitamente final y no pueden ser abstract.
- Son inmutables.

```
public record Persona(String nombre, String email, int telefono) {

    }
```

En el caso anterior, Persona tiene la característica record, por lo que todas las cosas que nombré anteriormente se generaron automáticamente. Si queréis comprobarlo podéis usar el ejemplo anterior y probar añadiendo una Persona y posteriormente mostrando la persona con un Syso para ver que el toString se autogenera solo o realizar más pruebas para comprobar que esto es real.

2.4. *Pattern Matching* PARA EL OPERADOR *instanceof*

Al usar el operador instanceOf para comprobar si un objeto es una instancia de una clase si se realiza en un if, posteriormente es necesario hacer un cast del objeto a la clase.
Sin embargo, en Java 14 se introdujo la novedad que nos permitía que el instanceOf permitiera renombrar la variable sin necesidad de realizar el cast. Esto simplifica el código y evita posibles errores.

Posterior a Java 14 →
```
if (obj instanceof String) {
    String s = (String) obj;
    // use s
}
```

A partir de Java 14 →
```
if (obj instanceof String s) {
    // use s
}
```

El uso del término "pattern" en este contexto proviene del hecho de que estás buscando un patrón específico en el objeto que estás evaluando. Este patrón puede ser una clase, una interfaz o una subclase específica, y cuando se encuentra, se realiza una acción asociada (como asignar la variable en la misma línea).

En resumen, "pattern" se refiere al patrón que estás buscando en el objeto (el tipo específico) y la acción que realizas cuando encuentras ese patrón (asignación de la variable).

3. NOVEDADES JAVA 15

3.1. BLOQUES DE TEXTO

Se añade de forma definitiva la novedad de bloque de texto de Java 13.

3.2. CLASES OCULTAS

Se añaden las clases ocultas o hidden classes que son clases que no pueden usarse directamente por otras clases. Es decir, esta es una clase privada (private).

3.3. SEALED CLASSES

En Java las clases permiten la reutilización de código mediante la herencia, los métodos de una clase son heredados por las subclases que la extiendan.
En Jav toda clase puede ser extendida por defecto la única forma de no permitir extender una clase es utilizando la palabra reservada final. Sin embargo, esto impide la extensión de la clase completamente. Las clases sealed especifican de forma explícita que clases tiene permitido la extensión y herencia. Las clases sealed son más restrictivas que el comportamiento por defecto de permitir a cualquier clase la extensión pero más permisivo que si se utiliza la palabra clave final que impide a cualquier clase la extensión.

```
public abstract sealed class Figura permits Circulo, Rectangulo, Triangulo {

   ...

}
public Circuloextends Figura{

   ...

}


public Rectangulo extends Figura{

   ...
```

```
}

public Triangulo extends Figura{
    ...
}
```

Una cosa importante a tener en cuenta al crear las clases hijas de la clase sealed es que en la línea donde se declara dicha clase, debemos poner alguna de las palabras reservadas final, sealed o non-sealed, dependiendo de lo que queramos hacer con esa clase.
En conclusión, la diferencia es que una clase final evita que se creen subclases, una sealed class permite subclases, pero controla explícitamente cuáles son permitidas, ofreciendo mayor nivel de control sobre la extensión de la jerarquía de clases.

3.4. OTRAS NOVEDADES

Pattern Matching para instanceof y records se mantienen en la categoría de funcionalidad preliminar y no son implementadas aún de forma definitiva.

4. NOVEDADES JAVA 16

Hay varias novedades pero no son muy importantes. Se implantan de foma definitiva los records, esto se publicaron como primera vista previa en Java 14 y una segunda versión en Java 15.

5. NOVEDADES JAVA 17

5.1. SEALED CLASSES

Las clase sealed fueron propuestas en Java 15 en modo de vista previa, en Java 16 fueron propuestas de nuevo con algunos cambios. Las clases sealed son incorporadas de forma final en Java 16.

5.1. PATTERN MATCHING PARA LOS SWITCHS

La novedad de pattern matching para switch en Java 17 es una forma más fácil y concisa de escribir código cuando necesitas hacer diferentes acciones dependiendo del tipo o valor de una variable. Antes, tenías que usar instanceof y luego convertir o "castear" la variable a un tipo específico para trabajar con ella. Con esta novedad, puedes hacer todo eso directamente dentro del switch.

Por ejemplo, supongamos que tienes una variable llamada obj y quieres hacer algo diferente dependiendo de si es una cadena o un número:

```
Object obj = "Hello";

switch (obj) {
    case String s -> System.out.println("Es una cadena: " + s);
    case Integer i -> System.out.println("Es un entero: " + i);
    default -> System.out.println("No es ni una cadena ni un entero.");
}
```

Aquí, el switch verifica el tipo de obj y, si es una cadena, la asigna a la variable s directamente dentro del case String s. De manera similar, si es un entero, lo asigna a la variable i dentro del case Integer i.
OJO!! Esta mejora fue implementada en Java 17, sin embargo, no se puede utilizar hasta Java 21. En el ejercicio de ejemplo aparece hecho y explicado en forma de comentario, para que veáis como se usa, pero si tenéis Java 17 os dará error.

En conclusión, Java 17 no introduce muchas novedades en el lenguaje, lo realmente importante son las mejoras de rendimiento y además integra diferentes mejoras ya comentadas en anteriores versiones (sealed classes, textos multilíneas…)

6. NOVEDADES JAVA 18

En regla general es una versión con muy pocas novedades con respecto al lenguaje. Tan solo añade mejoras a los *Pattern Matching* para las sentencias switch. En esta nueva revisión el compilador es más preciso al comprobar la completitud de todos los casos del switch en las clases sealed.

7. NOVEDADES JAVA 19

7.1. RECORD PATTERNS

Ahora el pattern matching se aplica a los records que permite una forma de desestructurar este tipo de clases para obtener los elementos de record en variables disponibles en el ámbito de uso de la expresión.

ANTES DEL CAMBIO:

```
record Point(int x, int y) {}

static void printSum(Object o) {

   if (o instanceof Point p) {
```

```
    int x = p.x();
    int y = p.y();
    System.out.println(x + y);
  }
}
```

Al escribir la expresión pattern matching para un record permite al mismo tiempo desestructurar los elementos y extraerlos a variables. La expresión resultante para desestructurar los elementos es muy verbosa pero hace más simple el acceso posterior a las variables record.

```
void printSum(Object o) {
  if (o instanceof Point(int x, int y)) {
    System.out.println(x + y);
  }
}
```

7.2. PATTERN MATCHING FOR SWITCH

El pattern matching en las expresiones switch han sido mejorados en esta nueva versión.

- La semántica del switch en los casos en que es null la expresión se ajusta a la semántica existente que ha existido en los switchs lanzando un NullPointerException.

  Por otro lado todo, ahora en los casos de un switch es posible usar el label null para seleccionar la rama cuando la expresión del switch toma un valor nulo, el comportamiento anterior era que si la expresión switch era null producía un NullPointerException lo que requería utilizar un guard clause antes del switch para evitar el NPE.

Soportar la etiqueta *null* permite ahora escribir un switch de la siguiente forma.

```java
Object o = ...;

switch (o) {
    case null:
    case String s: {
        System.out.println("String, including null");
    } break;
}
```

8. Actividad De Ejemplo.

Crea una tienda de figuras, todas las figuras van a tener un precio base, y su precio final se calculará multiplicando este por su área. Crea la tienda con record para así implementar una de las novedades de Java.
Crea una clase Figura, que tan solo permita heredar a sus dos hijas y tenga un método para calcular el área. Crea dos hijas de Figura (Cuadrado y Círculo). La clase Cuadrado debe de tener como atributo lado, y Círculo debe de tener como atributo radio. El círculo tendrá un método que muestre el mensaje que desees.
Debemos de crear un menú en el que tengamos las distinta opciones:

- Mostrar lista de figuras con aviso
- Un case donde se realice un sorteo y dependiendo del número obtenido obtenga una recompensa u otra (esta recompensa será de elección libre).
- Mostrar área de Círculo con un mensaje.