Las características más destacables de Java 8 en adelante

Java es uno de los lenguajes de programación más potentes. El lenguaje se volvió más fuerte cuando Oracle lanzó una nueva versión de Java, **Java SE 8**, el 18 de marzo de 2014 con novedades importantes.

1. Expresión Lambda

La expresión Lambda es una función anónima (una función sin nombre) que ayuda a escribir código en un cierto estilo funcional. Dado que los datos se pueden iterar, filtrar y extraer, es muy útil, especialmente en la biblioteca de recopilación. La expresión Lambda también ayuda a reducir la complejidad del código.

Sintaxis

```
(Parametros) -> Expresión
```

Código:

```
(a,b) -> a + b // Esto nos devuelve la suma de a + b
```

2. Stream API

Stream API ofrece una técnica para el procesamiento de datos de diferentes maneras, como filtrado, extracción, transformación, con la ayuda del paquete `java.util.stream`. Una de las principales ventajas de **Stream API** es que no altera su fuente, es decir, si se filtra un conjunto de datos, se crea un nuevo conjunto de datos con los datos filtrados en lugar de modificar la fuente original. **Stream API** evalúa el código sólo cuando es necesario y no repite el código más de una vez.

3. Método ForEach ()

Java SE8 ofrece un nuevo método llamado `forEach` que se define en la interfaz `Iterable`. El bucle `ForEach ()` se puede usar en una clase de colección que extiende la interfaz Iterable. Como este método ingresa un solo parámetro, una expresión lambda también se puede pasar como parámetro.

Código: Recurre cada edad y la imprime

```
edad.forEach( edad -> (System.out.println(edad);));
```

4. Método predeterminado

Normalmente, los métodos no abstractos no se pueden agregar a las interfaces. Pero en el caso de **Java 8**, podemos agregar esos métodos en las interfaces. Estos métodos se escriben con una palabra clave predeterminada y se conocen como métodos predeterminados. Como son métodos no abstractos, también se puede incluir el cuerpo del método por ej:

Código

```
public interface Movible(
    default void Sonido()(
        System.out.println("Hola Ana, ¿cómo está mi sonido?");
 ))
```

5. Fecha Hora API

En Java 8, se ha introducido una nueva API de tiempo y API de fecha donde las fechas de manejo están en un método diferente en comparación con otras versiones de Java. Algunas de las siguientes son clases de hora y fecha que están disponibles en Java. paquete de tiempo:

- Clase Jtime.LocalDate
- Clase LocalTime
- Clase LocalDateTime
- Clase de MonthDay

Código:

```
Clock cl = Clock.systemDefaultZone();
System.out.println(cl.getZone());
```

7. Referencias de métodos

Las referencias de métodos son otra característica que se introduce en **Java 8**, que se puede usar en métodos de interfaz funcional. De otra manera, se puede decir que son un subconjunto de otra característica de Java, las expresiones lambda. Esto se debe a que también se puede usar una referencia de método si se puede usar una expresión lambda.

Los métodos pueden ser:

- Referencia al constructor
- Referencia al método estático
- Referencia a un método de instancia

Diferencias entre java 8 y Java 9

Java SE8 es la nueva versión de Java desarrollada por Oracle que ofrece varias características nuevas. La expresión **lambda** se considera como la característica más significativa entre ellas.

Ahora es momento de hablar sobre **Java 9**, el principal cambio que tuvo el JDK 9 fue la introducción de un componente llamada "el módulo" el proyecto Jigsaw.

Es un proyecto que nos ayuda a reducir el tamaño y complejidad tanto de las aplicaciones java como del propio runtime de java JRE, de este modo podemos utilizar solamente lo que necesitamos en nuestra aplicación y no la plataforma java completa.

También tuvimos la implementación del Método **Factory** que proviene del patrón de proyecto llamado **Factory**. La idea es que, pensando en las relaciones entre objetos, los métodos de interfaz sean creados sin especificar sus clases concretas. Hablar así puede parecer extraño pero esta implementación, cuando pensamos en el método .of() se usa frecuentemente al pensar en `Enums` en Java.

Al finalizar **Java 9**, no podemos dejar de mencionar la incorporación del flujo reactivo por medio del Flow Api, una clase llamada `Flow` en el paquete `Util` de Java que busca mejorar el flujo de procesamiento asincrónico de datos. Tenemos cuatro clases que trabajan este flujo: Suscripción, Procesador, Suscriptor y Publicador.

- Clase Suscriptor trabaja como origen de los mensajes y demandas.
- Clase Publicador trabaja con el procesamiento de las demandas de forma asincrónica.
- Clase Suscripción se encarga de la conexión entre las clases Suscriptor y Publicador al manejar los mensajes de control.
- Clase Procesador puede trabajar tanto como la clase Suscriptor como la clase Publicador.

Una vez mencionadas las características más destacables de esta versión, pasemos a **java 10**.

Características de Java 10

1. Inferencia de tipo variable local

Esta nueva característica busca simplificar y mejorar la experiencia del desarrollador. La nueva sintaxis reducirá la longitud del código relacionado con Java, al mismo tiempo que mantendrá el compromiso de la seguridad del tipo estático.

La inferencia de tipo variable local introducirá la palabra clave `var`, es decir, puede definir libremente las variables sin tener que especificar el tipo de variable, como por ej:

Pasamos de esto:

```
list <String> list = new ArrayList <String> ();
Stream <String> stream = getStream();
```

A esto:

```
var list = new ArrayList <String> ();
var stream = getStream();
```

2- Interfaz del Recolector de Basura

En la estructura anterior de JDK, los componentes que formaban una implementación del Recolector de Basura (GC) estaban dispersos en varias partes del código base. Esto ha cambiado en **Java 10**. Ahora, es una interfaz limpia dentro del código fuente de **JVM** para permitir que los colectores alternativos se integren rápida y fácilmente. Mejorará el aislamiento del código fuente de los diferentes recolectores de basura.

Características de Java 11

1- Cliente HTTP

La meta con la nueva característica `API HttpClient` es proporcionar una manera estándar de consumir ApiRest, usando patrones modernos y mejores prácticas ya aplicadas en las librerías de terceros mencionadas anteriormente.

Tiene también, soporte para conexiones seguras, `HTTP2, websockets y request` asíncronos. Hay 3 conceptos básicos en este API:

- `HttpClient`: Se encarga de crear la conexión y su configuración. Proxies, Authenticators, executors pools, cookie manager, etc.
- `HttpRequest`: Es básicamente el request a un URI. Podemos indicar el timeout, headers, si el request es asíncrono o síncrono.
- `HttpResponse`: Objeto para manejar la respuesta del request. En él se puede ver el status code, body response, headers y más.

2- Ejecución desde archivo de código fuente único

Para ejecutar un programa Java es necesario primero compilarlo a *bytecode* para posteriormente ejecutarlo. Se necesitan dos pasos para facilitar la ejecución de los programas que se componen de un único archivo de código fuente, se suma la posibilidad de lanzar un programa desde el código fuente. Esto es útil para programas pequeños o para los casos de estar aprendiendo el lenguaje.

Y por último vamos a repasar algunas características de **Java 17** en este artículo.

Características de Java 17

- Se definió un estándar para el `switch` ya muy usado en Java y otros lenguajes con un foco en una mejor legibilidad y modelado del código. Se evita así el uso de repetidos `IF/ELSE` y el operador `instanceOf`. Este estilo es algo más declarativo, podemos visualizarlo mejor con un ejemplo:

```
static String formateo (Object o){
String formato = "desconocido";
  if(o instanceof Integer i){
      formato = String.format("int %d", i);
  }else if (o instanceof Long l){
      formato = String.format("long %d", l);
  }else if(o instanceof Double d){
      formato = String.format("double %f" , d);
  }else if(o instanceof String s){
      if(s.length > 3){
          formato = String.format("Cadena Corta %s" + s);
      }else{
          formato = String.format("Cadena Larga %s" + s);
      }
  }
  return formato;
}
```

  Ahora podemos hacerlo de esta otra forma:

```
static String formatoSwitch (Object o){
  return switch (o) {
          case Integer i -> String.format("int %d" , i);
          case Long l -> String.format("long %d", l);
          case Double d -> String.format("double %f" , d);
          case String s && (s.length > 3) -> String.format("Cadena Corta %
          case String s && (s.lenght > 10) -> String.format("Cadena Larga
          default -> o.toString();
  };
}
```

  - Se elimina Applet API de Java, que generaba un archivo binario que podía ser utilizado para tener pantallas web. Ya había sido definido como obsoleto en **Java 9** pero solo ahora en **Java 11** fue oficialmente eliminado
  - Además, se mejoró la parte referente a generación de números aleatorios, introduciendo una nueva interfaz llamada `RandomGenerator`, una API uniforme que puede generar números para diferentes tipos numéricos.

Finalmente vamos a hablar un poquito sobre **Java 18**sin ser una versión LTS.

Características de Java 18

Java 18 fue lanzado a principios de 2022 y cuenta con actualizaciones no enfocadas directamente en el código Java, sino en la plataforma de desarrollo. Fue estandarizado el conjunto de caracteres con el UTF-8 y se agregó una etiqueta @Snippet para agregar código fuente en documentaciones creadas para códigos/proyectos en el lenguaje. Creando una utilización con pocas líneas de código para generar un simple servidor web para el programador.

Conclusión

Finalmente podemos sacar una conclusión de toda esa nueva información que tenemos sobre las diferentes versiones de nuestro querido Java. Es que nos ahorramos de escribir tanto código usando la sentencia `Switch` en vez de escribir tantos IF/ELSE anidados, podemos escribir código funcional a través de las Lambda. Tenemos el bucle `forEach`, manejamos de una formas más eficiente las fechas y las horas con las nuevas APIS, ya no es obligatorio preceder el nombre de una variable con el tipo de dato, sino que usamos la palabra "var" y consumir de una mejor manera las ApiRest a través del cliente HTTP.