



LO NUEVO DE JAVA 10

Adolfo Sebastián Jara Gavilanes

CAMBIOS DE JAVA 10



INFERENCIA DE LAS
VARIABLES
LOCALES



VERSIÓN DE
LANZAMIENTOS
BASADOS EN LA
FECHA.



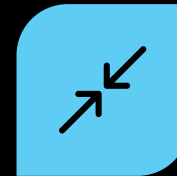
EXTENSIONES DE
ETIQUETA DE
IDIOMA UNICODE
ADICIONALES.



THREAD LOCAL-
HANDSHAKES.



GARBAGE
COLLECTION EN
PARALELO PARA
G1.



INTERFAZ
MEJORADA DE
GARBAGECOLLECT
OR.



CAMBIOS DE
API(PARA
COLECCIONES NO
MODIFICABLES).

INFERENCIA DE LAS VARIABLES LOCALES

Se utiliza la palabra reservada "var", y reemplaza las palabras para construir un nuevo tipo de objeto.

"var" puede ser nombre de una variable, método, paquete pero no de una clase

En las funciones lambda, no es necesario declarar los parámetros.

Facilita la lectura y escritura del código.

Se los puede utilizar solo en los siguientes casos:

Funciona con variables locales e inicializadas.

Se los utiliza en bucles "for".

```
public class Test {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        ControladorPersona controlador = new ControladorPersona();

        controlador.create(new Persona("Jose", "Jose", "01", 65, "Cuenca"));
        controlador.create(new Persona("Pepe", "Pepe", "02", 45, "Azogues"));
        controlador.create(new Persona("Andres", "Andres", "03", 25, "Quito"));
        controlador.create(new Persona("Santiago", "Santiago", "04", 32, "Guayaquil"));
        controlador.create(new Persona("Mateo", "Mateo", "05", 15, "Salinas"));

        //se recibe un List<Persona>
        var personas = controlador.findAll();
        System.out.println(personas);

        System.out.println("\n-----Imprimir una lista-----\n");

        //usamos ahora la palabra var para recorrer un array
        for (var var : personas) {
            System.out.println(var.toString());
        }

        //llamamos al metodo var del controlador
        controlador.var();
    }
}
```

[Persona:

Nombre: Jose apellido: Jose cedula: 01 edad: 65 ciudadVivir: Cuenca, Persona:

Nombre: Pepe apellido: Pepe cedula: 02 edad: 45 ciudadVivir: Azogues, Persona:

Nombre: Andres apellido: Andres cedula: 03 edad: 25 ciudadVivir: Quito, Persona:

Nombre: Santiago apellido: Santiago cedula: 04 edad: 32 ciudadVivir: Guayaquil, Persona:

Nombre: Mateo apellido: Mateo cedula: 05 edad: 15 ciudadVivir: Salinas]

-----Imprimir una lista-----

Persona:

Nombre: Jose apellido: Jose cedula: 01 edad: 65 ciudadVivir: Cuenca

Persona:

Nombre: Pepe apellido: Pepe cedula: 02 edad: 45 ciudadVivir: Azogues

Persona:

Nombre: Andres apellido: Andres cedula: 03 edad: 25 ciudadVivir: Quito

Persona:

Nombre: Santiago apellido: Santiago cedula: 04 edad: 32 ciudadVivir: Guayaquil

Persona:

Nombre: Mateo apellido: Mateo cedula: 05 edad: 15 ciudadVivir: Salinas

Imprimiendo desde el método llamado 'var'

VERSIÓN DE LANZAMIENTOS BASADO EN LA FECHA



- Revisa el esquema del string de versión de Java SE Platform y el JDK, y la información de versiones relacionada, para los modelos de versiones actuales y futuros basados en el tiempo.
- Métodos para saber la versión:
 - Feature().- Este incrementará cada 6 meses y basado en las actualizaciones de Java
 - Interim().- Este es un contador que incrementará cada vez que salga una actualización de mejora de "bugs". Normalmente es cero porque depende de las actualizaciones de las versiones.
 - Update().- Contador que incrementa cada vez que hay nuevos lanzamientos de seguridad, y cambios importantes. Normalmente la primera vez se lo hace después de 1 mes y luego cada 3 meses.
 - Patch().- Contador que incrementará cada vez que salga una actualización de urgencia para solucionar algún problema crítico.


```
Version version = Runtime.version();  
System.out.println("Feature: " + version.feature());  
System.out.println("Interim: " + version.interim());  
System.out.println("Updates: " + version.update());  
System.out.println("Patches: " + version.patch());
```

-----Ejemplo de la versión-----

Feature: 14

Interim: 0

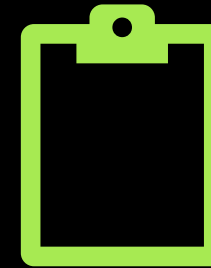
Updates: 1

Patches: 0

EXTENSIONES DE ETIQUETA DE IDIOMA UNICODE ADICIONALES.



Se mejoraron las APIs.



Se agregaron etiquetas de información adicional para los idiomas:


cu (tipo de divisa currency).
fw (primer día de la semana).
rg (sobreescribir región).
tz (zona horaria).

```
//escogemos el primer dia de la semana
Calendar cal = Calendar.getInstance();
cal.setFirstDayOfWeek(Calendar.MONDAY);

//escogemos la region horaria
cal.setTimeZone(TimeZone.getTimeZone("UTC"));

InformacionTiempoMoneda informacion = new InformacionTiempoMoneda(Locale.FRANCE, cal);

System.out.println(informacion.toString());
```



```
-----Informacion de la localizacion-Moneda-Tiempo-----  
Moneda:  
localizacion: fr_FR, currency: EUR  
Primer día de la semana: 2 Zona horaria: UTC
```



THREAD LOCAL- HANDSHAKES

- Es una mejora interna de JVM para ser más eficaz.
- “handshake” = es una operación de llamada de vuelta a un hilo que se encuentra en un punto seguro, ejecutada por `JavaThread`.
- La llamada puede ser realizada por el mismo hilo o por VM.
- Esto nos permite llamar devuelta a un hilo individual sin importar de la VM, lo cual antes o se paraban todos los hilos o ninguno.

GARBAGE COLLECTION EN PARALELO PARA G1, INTERFAZ MEJORADA DE GARBAGECOLLECTOR.

- HotSpot = es implementada en Just-in-Time compilador. Partes de una app, y partes de su código son leídas como nativas y son guardadas en memoria caché para una mayor velocidad de ejecución.
- Esto hace más eficaz el “HotSpot” JVM.
- Se ha mejorado el recolector de basura G1 añadiendo soporte para paralelismo y mejorado las pausas en los peores escenarios.
- Antes de java 10, GC era fragmentado lo cual derivaba en algunos problemas:
 - Conocer estos lugares para saber en que lugar implementar, eliminar un GC.
 - Difícil de encontrarlos en un lugar de código.


```
System.out.println("\n-----Ejemplo Garbage Collection-----");

//creo objeto de esta clase
Persona t1 = new Persona("Lautaro", "Lautaro", "55", 15, "Loja");

//aviso al GC que va a tener que recoger basura
System.gc();

//pongo un valor nulo
t1 = null;

//se pide al JVM que ejecute el GC
Runtime.getRuntime().gc();
```



```
-----Ejemplo Garbage Collection-----
```

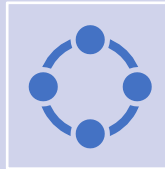
```
Garbage collector llamado
```

```
Garbage collector elimino: Persona:
```


CAMBIOS DE API(PARA COLECCIONES NO MODIFICABLES).



Se creo un nuevo método para crear una copia de colecciones que no pueden ser modificadas.



Método:
`copyOf(Collection).`

```
System.out.println("\n-----Ejemplo copias de colecciones inmodificables-----");
var listaPer = controlador.findAll();

var copiaListaPer = List.copyOf(listaPer);

System.out.println("Tamaño lista original: " + listaPer.size()
|      |      + " Tamaño lista copia: " + copiaListaPer.size());

//se intenta añadir una persona
try {
|      copiaListaPer.add(new Persona("Sebastian", "Sebastian", "10", 30, "Puyo"));
} catch (UnsupportedOperationException e) {
|      System.out.println("\nNO HAY COMO AÑADIR OBJETOS A LA LISTA DUPLICADA\n");
}

listaPer.add(new Persona("Sebastian", "Sebastian", "10", 30, "Puyo"));

System.out.println("Tamaño lista original: " + listaPer.size()
|      |      + " Tamaño lista copia: " + copiaListaPer.size());
```

-----Ejemplo copias de colecciones inmodificables-----

Nombre: Lautaro apellido: Lautaro cedula: 55 edad: 15 ciudadVivir: Loja
Tamaño lista original: 5 Tamaño lista copia: 5

NO HAY COMO AÑADIR OBJETOS A LA LISTA DUPLICADA

Tamaño lista original: 6 Tamaño lista copia: 5

BIBLIOGRAFÍA

Bibliography

- Dalorzo, E. (2018, Agosto 2). *stackoverflow*. Retrieved from Difference between JVM and HotSpot:
<https://stackoverflow.com/questions/16568253/difference-between-jvm-and-hotspot>
- Fernandez, R. (2018, Marzo 16). *RicardoGeek*. Retrieved from Qué hay de nuevo en Java 10:
<https://ricardogeek.com/que-hay-de-nuevo-en-java-10/>
- Foley, R. (2018, Abril 10). *IDR Solutions*. Retrieved from Collections explained in 5 minutes:
<https://blog.idrsolutions.com/2018/04/java-10-improvements-to-garbage-collection-explained-in-5-minutes/>
- Kumar, R. (2018, Abril 6). *JournalDev*. Retrieved from Java 10 Features:
<https://www.journaldev.com/20395/java-10-features#time-based-release-versioning-jep-322>
- Oracle. (2018, Julio 17). *Oracle*. Retrieved from JDK 10 Release Notes:
<https://www.oracle.com/java/technologies/javase/10-relnote-issues.html#NewFeature>
- Vaidya, A. (2019, Agosto 9). *GeeksforGeeks*. Retrieved from Garbage Collection in Java:
<https://www.geeksforgeeks.org/garbage-collection-java/>
- Villatoro, G. (2018, Marzo 21). *LoMasNuevo*. Retrieved from Lo nuevo de Java 10:
<https://www.lomasnuevo.net/noticias/lo-nuevo-de-java-10/>