
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Universidad Politécnica Salesiana

Vicerrectorado Docente

Código del Formato:	GUIA-PRL-001
Versión:	VF1.0
Elaborado por:	Directores de Área del Conocimiento Integrantes Consejo Académico
Fecha de elaboración:	2016/04/01
Revisado por:	Consejo Académico
Fecha de revisión:	2016/04/06
Aprobado por:	Lauro Fernando Pesántez Avilés Vicerrector Docente
Fecha de aprobación:	2016/14/06
Nivel de confidencialidad:	Interno

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Descripción General

Propósito


El propósito del presente documento es definir un estándar para elaborar documentación de guías de práctica de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana, con la finalidad de lograr una homogenización en la presentación de la información por parte del personal académico y técnico docente.


Alcance

El presente estándar será aplicado a toda la documentación referente a informes de prácticas de laboratorio, talleres o centros de simulación de las Carreras de la Universidad Politécnica Salesiana.

Formatos


- Formato de Guía de Práctica de Laboratorio / Talleres / Centros de Simulación – para Docentes
- Formato de Informe de Práctica de Laboratorio / Talleres / Centros de Simulación – para Estudiantes


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		


		FORMATO DE GUÍA DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA DOCENTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Investigación de las versiones de Java	
OBJETIVO: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación Entender la cada uno de las características nuevas en Java			
INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):		1. Revisar los conceptos fundamentales de las nuevas versiones de Java	
		2. Establecer las características de Java	
		3. Implementar y diseñar los nuevos componentes de programación	
		4. Realizar el informe respectivo según los datos solicitados.	
ACTIVIDADES POR DESARROLLAR (Anotar las actividades que deberá seguir el estudiante para el cumplimiento de la práctica)			
1. Revisar la teoría y conceptos de Java 9 ,10, 11, 12			
2. Diseñar e implementar las características de Java,			
3. Probar su funcionamiento y rendimiento dentro de los equipos de computo			
4. Realizar práctica codificando los códigos de las nuevas características de Java.			
RESULTADO(S) OBTENIDO(S): Realizar procesos de investigación sobre los cambios importantes de Java Entender las aplicaciones de codificación de las nuevas características Entender las funcionalidades adicionales de Java.			
CONCLUSIONES: Aprenden a trabajar en grupo dentro de plazos de tiempo establecidos, manejando el lenguaje de programación de Java.			
RECOMENDACIONES: Realizar el trabajo dentro del tiempo establecido.			

Docente / Técnico Docente: _____

Firma: _____

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES	
CARRERA: COMPUTACIÓN		ASIGNATURA: Programación Aplicada	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Investigación de las versiones de Java	
OBJETIVO: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación Entender la cada uno de las características nuevas en Java			
ACTIVIDADES DESARROLLADAS			
<p>1. Revisar la teoría y conceptos de Java 10</p> <p>JDK 10, una implementación de Java Standard Edition 10, se lanzó el 20 de marzo de 2018. Las mejoras clave incluyen tipos de variables locales y mejoras para la recolección y compilación de basura. El JDK 10 está programado para ser solo un lanzamiento a corto plazo, y las actualizaciones públicas para JDK 10 están programadas para finalizar en seis meses.</p> <p>Inferencia en el tipo de variables locales</p> <p>Local-Variable Type Inference. Esta funcionalidad mejora la experiencia de desarrollo al eliminar el código adicional, necesario hasta ahora, para declarar variables locales expresando el tipado completo de la variable.</p> <p>La nueva inferencia de tipos introduce la palabra reservada <code>var</code> para declarar variables locales e inferir su tipo.</p> <p>La inferencia de tipos es una característica que tienen casi todos los lenguajes de programación que usan llaves y tipos estáticos, como por ejemplo C# (<code>var</code>), Scala (<code>var/val</code>) o incluso Go (<code>:=</code>); y no es más que la opción que los lenguajes le dan a los desarrolladores de inferir el tipo de la variable que se está trabajando. Vamos en otras palabras, el lenguaje adivina que estas queriendo decir.</p> <p>En resumen, esto solo funciona en un ámbito local y con variables inicializadas.</p> <p>Entonces la inferencia de tipos funciona a muchos niveles, pero con un gran poder viene una gran responsabilidad, ya que entre más libertades da un lenguaje más fácil es que los programadores adopten malas prácticas.</p> <p>Métodos de copia</p> <p>Métodos estáticos <code>copyOf</code> en interfaces de colección para crear copias no modificables</p> <p>Nuevas API para habilitar mejor la creación de colecciones no modificables. Los métodos <code>copyOf</code>, <code>Set.copyOf</code> y <code>Map.copyOf</code> crean nuevas instancias de recopilación a partir de instancias existentes.</p> <p>Se puede modificar la colección original pero no se puede modificar la copia.</p>			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Métodos de reducción

Se han añadido a la clase Collectors para facilitar la creación de colecciones inmodificables.

Se agregaron nuevos métodos aUnDynamifiableList, aUnmodifiableSet y aUnDatosmodificables a la clase Recopiladores en el paquete Stream, lo que permite que los elementos de un Stream se recopilen en una colección no modificable.

Consolidar todos los repositorios del JDK en uno solo

Hasta Java 9 se tenían 8 repositorios en mercurial: root, corba, hotspot, jaxp, jaxws, jdk, langtools, and nashorn. Estos han sido consolidados en uno mismo para efectos de hacer cambios de manera más rápida y agilizar la gestión de la configuración.

La base de código hasta ahora se ha dividido en varios repositorios, lo que puede causar problemas con la administración del código fuente.

Extensiones de etiquetas de idioma Unicode adicionales

Se mejoró java.util.Locale y sus API relacionadas para implementar extensiones Unicode adicionales de etiquetas de idioma. A partir de Java 10 se soportará las siguientes etiquetas de lenguaje:

cu (tipo de divisa currency)
fw (primer día de la semana)
rg (Sobre-escribir región)
tz (zona horaria)

Una interfaz limpia y unificada para el Garbage-Collector

Mejore el aislamiento del código fuente de diferentes recolectores de basura mediante la introducción de una interfaz limpia del recolector de basura

Esta interfaz está definida por la clase CollectedHeap que todos los GarbageCollectors implementan. La clase CollectedHeap dirigirá la mayoría de los aspectos de interacción entre el colector y el resto del HotSpot. Más específicamente una implementación de un GarbageCollector tendrá que proveer lo siguiente:

- Un heap, que sea una subclase de CollectedHeap
- Un barrier set, que sea una subclase de BarrierSet, que implemente algún barrier en tiempo de ejecución.
- Una implementación de CollectorPolicy
- Una implementación de GCInterpreterSupport, Que implementa las barreras que GC provee para el intérprete usando instrucciones de assembler.
- Una implementación de GCC1Support, Que implemente las barreras de un GC para el compilador C1
- Una implementación de GCC2Support, Que implemente las barreras de un GC para el compilador C2
- La inicialización de argumentos específicos eventuales del GC
- Configurar un MemoryService, los memory pools relacionados, gestores de memoria, etc.


2. Diseñar e implementar las características de Java,

Inferencia en el tipo de variables locales

```
13  /**
14   *
15   * @author Romel
16   */
17  public class Principal {
18      //var c=8;//no valido en atributos
19      public static void main (String [] arg){
20
21          int numero= 9;
22          //numero="hola";//no se puede asignar una variable de tipo diferente a la declarada primero
23          ArrayList datos=new ArrayList<Integer>();
24
25
26          datos.add(20);
27          datos.add(30);
28          datos.add(40);
29          datos.add(numero);
30
31          System.out.println("Entero-"+numero);
32          System.out.println(datos);
```

```
13  /**
14   *
15   * @author Romel
16   */
17  public class Principal {
18      var c=8;//no valido en atributos
19      public static void main (String [] arg){
20
21          var numero= 9;
22          numero="hola";//no se puede asignar una variable de tipo diferente a la declarada primero
23          var datos=new ArrayList<Integer>();
24
25
26          datos.add(20);
27          datos.add(30);
28          datos.add(40);
29          datos.add(numero);
30
31          System.out.println("Entero-"+numero);
32          System.out.println(datos);
```

En las imágenes podemos comprobar que se puede remplazar la var por cualquier tipo de variable

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Métodos de copia

```

23      var datos=new ArrayList<Integer>();
24
25      datos.add(20);
26      datos.add(30);
27      datos.add(40);
28      datos.add(numero);
29
30      var copia= List.copyOf(datos);
31
32      System.out.println("Copia-"+copia);
33
34      datos.add(50);
35
36      System.out.println(datos);
37
38      //copia.add(80);//No se puede modificar la copia

```

En la imagen se demuestra como se hace una copia de una lista.

3. Probar su funcionamiento y rendimiento dentro de los equipos de computo

Inferencia en el tipo de variables locales



```

13  /**
14   *
15   * @author Romel
16   */
17  public class Principal {
18      //var c=8;//no valido en atributos
19      public static void main (String [] arg){
20
21          int numero= 9;
22          //numero="hola";//no se puede asignar una variable de tipo diferente a la declarada primero
23          ArrayList datos=new ArrayList<Integer>();
24
25          datos.add(20);
26          datos.add(30);
27          datos.add(40);
28          datos.add(numero);
29
30          System.out.println("Entero-"+numero);


```

Output - Run (Java10)

```

-----
>>> Building Java10 1.0-SNAPSHOT
-----
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Java10 ---
Entero-9
[20, 30, 40, 9]
-----
BUILD SUCCESS

```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

```

13  /**
14   *
15   * @author Romel
16   */
17  public class Principal {
18      //var c=8;//no vaildo en atributos
19      public static void main (String [] arg){
20
21          var numero= 9;
22          //numero="hola";//no se puede asignar una variable de tipo diferente a la declarada primero
23          var datos=new ArrayList<Integer>();
24
25          datos.add(20);
26          datos.add(30);
27          datos.add(40);
28          datos.add(numero);
29
30          System.out.println("Entero-"+numero);

```

Output - Run (Java10) x

```

--- exec-maven-plugin:1.5.0:exec (default-cli) @ Java10 ---
Entero-9
[20, 30, 40, 9]
BUILD SUCCESS

```

En las imágenes vemos que el resultado es el mismo

Métodos de copia

```

25      datos.add(20);
26      datos.add(30);
27      datos.add(40);
28      datos.add(numero);
29
30      System.out.println("Entero-"+numero);
31      System.out.println(datos);
32
33      var copia= List.copyOf(datos);
34
35      System.out.println("Copia-"+copia);
36
37      datos.add(50);
38
39      System.out.println(datos);
40
41      copia.add(80);//No se puede modificar la copia
42  }

```


Output - Run (Java10) x

```

--- exec-maven-plugin:1.5.0:exec (default-cli) @ Java10 ---
Entero-9
[20, 30, 40, 9]
Copia-[20, 30, 40, 9]
[20, 30, 40, 9, 50]
Exception in thread "main" java.lang.UnsupportedOperationException
at java.base/java.util.ImmutableCollections.uoe(ImmutableCollections.java:73)
at java.base/java.util.ImmutableCollections$AbstractImmutableCollection.add(ImmutableCollections.java:77)
at ec.edu.ups.test.Principal.main(Principal.java:41)

```

En la imagen podemos ver como funciona la copia y como no podemos modificarla porque nos dará un error.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

4. Realizar práctica codificando los codigos de las nuevas características de Java.

```

package ec.edu.ups.test;

import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Rome1
 */
public class Principal {
    //var c=8;//no vaildo en atributos
    public static void main (String [] arg){

        int numero= 9;
        //numero="hola";//no se puede asignar una variable de tipo diferente a la declarada
        primero
        ArrayList datos=new ArrayList<Integer>();

        datos.add(20);
        datos.add(30);
        datos.add(40);
        datos.add(numero);

        System.out.println("Entero-"+numero);
        System.out.println(datos);

        var copia= List.copyOf(datos);

        System.out.println("Copia-"+copia);


        datos.add(50);

        System.out.println(datos);

        //copia.add(80);//No se puede modificar la copia
    }
}

```

Código escrito en java de la implementación de los métodos

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

5. Bibliografía:

A. Martín, 27 feb. 2019 extraído de: https://www.youtube.com/watch?v=SMKpLo_68IE&t=419s

Mar 26, 2018 Extraído de: <https://ricardogeek.com/que-hay-de-nuevo-en-java-10/>

Roman Kennke, 2018/04/09 extraído de: <http://openjdk.java.net/jeps/304>

Marc Nuri, 2018-03-31 Extraído de: <https://blog.marcnuri.com/java-10-probando-las-nuevas-funcionalidades/>

RESULTADO(S) OBTENIDO(S):

Pusimos en práctica métodos que vienen incluidos en el java 10 aprendiendo más sobre estas herramientas que nos traen las nuevas versiones de java actualizándonos más cada día.

CONCLUSIONES:

En conclusión, desde el punto de vista del programador, de este java 10 es la inferencia de tipos de variables locales y lo otro es la optimización que hace que el JDK sea aún más rápido y eficiente.

RECOMENDACIONES:

Investigar más sobre los métodos que traen las nuevas versiones para así poder ponerlas en practica en la programación

Estudiantes: Romel Ávila

Firma:

