UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

PRÁCTICA DE LABORATORIO

CARRERA: COMPUTACION		**ASIGNATURA**: PROGRAMACION APLICADA
NRO. PRÁCTICA:	1	**TÍTULO PRÁCTICA**: Cambios en Java 10

OBJETIVO ALCANZADO:

Reconocer y aplicar los cambios más representativos de Java 10.

ACTIVIDADES DESARROLLADAS

1. Realizar una presentación de los cambios en Java 10.

La presentación de powerpoint será presentada en clases. Dicha presentación consta de 19 diapositivas en la cual se incluyen las descripciones de cada cambio y están acompañadas por capturas de pantalla de ejemplos de cada cambio.

2. Investigación realizada.

Debido a que son varios cambios, las divide en 8 cambios y cada una con sus distintas diapositivas. A continuación, se describirán los cambios más importantes de java.

- **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, y Se los utiliza en bucles “for”.
- **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 incrementara 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.**

- **Extensiones de etiqueta de idioma UNICODE adicionales:**
  - Se mejoraron y aclararon muchas APIs.
  - Se agregaron etiquetas adicionales para los idiomas:
  - cu (tipo de divisa currency).
  - fw (primer dia de la semana).
  - rg (sobreescribir región).
  - tz (zona horaria).
- **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 qué lugar implementar, eliminar un GC.
  - Difícil de encontrarlos en un lugar de código.
- **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).

1. Código elaborado.

El proyecto creado para este deber se llama: EjemploCambiosJava10. En esta se encuentran 3 paquetes: ups.edu.ec.controlador, ups.edu.ec.modelo, ups.edu.ec.test. En esta última se encuentra la clase principal. En el paquete del controlador se encuentra la clase encargada de guardar datos, y en el paquete modelo se encuentran dos clases: Persona y InformacionMoneda. La última clase se encuentran la mayoría de los cambios que hay en el Unicode language.

Primero mostraremos el código de las clases del paquete modelo:

Clase Persona:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package ups.edu.ec.modelo;


import java.util.Objects;


/**
 *
 * @author Adolfo
 */
public class Persona {


    private String nombre;
    private String apellido;
    private String cedula;
    private int edad;
    private String ciudadVivir;

    public Persona(String nombre, String apellido, String cedula, int edad, String ciudadVivir) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.cedula = cedula;
        this.edad = edad;
        this.ciudadVivir = ciudadVivir;
    }

    public Persona() {
    }

    public String getNombre() {
```

```
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public String getApellido() {
    return apellido;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public String getCedula() {
    return cedula;
}

public void setCedula(String cedula) {
    this.cedula = cedula;
}

public int getEdad() {
    return edad;
}

public void setEdad(int edad) {
    this.edad = edad;
```

```
}

public String getCiudadVivir() {
    return ciudadVivir;
}

public void setCiudadVivir(String ciudadVivir) {
    this.ciudadVivir = ciudadVivir;
}

@Override
public int hashCode() {
    int hash = 3;
    hash = 79 * hash + Objects.hashCode(this.cedula);
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass() != obj.getClass()) {
        return false;
    }
    final Persona other = (Persona) obj;
```

```
        if (!Objects.equals(this.cedula, other.cedula)) {
            return false;
        }
        return true;
    }



    /**
     * Metodo que se ejecuta con el ejemplo del GC justo antes de borrar un objeto
     * @throws Throwable
     */
    @Override
    protected void finalize() throws Throwable{
        try {
            System.out.println("Garbage collector llamado");
            System.out.println("Garbage collector elimino: " + this);
        } finally {
            super.finalize();
        }
    }

    @Override
    public String toString() {
        return "Persona:\n" + "Nombre: " + nombre + " apellido: " + apellido +
                " cedula: " + cedula + " edad: " + edad + " ciudadVivir: " + ciudadVivir;
    }


}
```

Clase InformacionTiempoMoneda.

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ups.edu.ec.modelo;

import java.util.Calendar;
import java.util.Currency;
import java.util.Locale;

/**
 *
 * @author Adolfo
 */
public class InformacionTiempoMoneda {

    private Locale localizacion;
    private Currency currency;
    private Calendar calendario;

    public InformacionTiempoMoneda(Locale localizacion, Calendar calendario) {
        this.localizacion = localizacion;
        currency = Currency.getInstance(localizacion);
        this.calendario = calendario;
    }

    @Override
```

```
    public String toString() {
        return "Moneda: \n" + "localizacion: " + localizacion + ", currency: " + currency +
                "\nPrimer día de la semana: " + calendario.getFirstDayOfWeek() +
                " Zona horaria: " + calendario.getTimeZone().getID();
    }



}
```

Ahora procedemos con la clase ControladorPersona del paquete controlador:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ups.edu.ec.controlador;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import ups.edu.ec.modelo.Persona;

/**
 *
 * @author Adolfo
 */
public class ControladorPersona {

    private List<Persona> listadoPersonas;
```

```
    public ControladorPersona() {
        listadoPersonas = new ArrayList<>();
    }

    public void create(Persona persona){
        listadoPersonas.add(persona);
    }

    public Optional<Persona> read(String cedula){
        return listadoPersonas.stream().filter(per->per.getCedula().equals(cedula)).findFirst();
    }

    public List<Persona> findAll(){
        return listadoPersonas;
    }

    public void var(){
        System.out.println("\nImprimiendo desde el método llamado 'var'\n");
    }
}
```

Ahora procedemos con la clase Test del paquete Test:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package ups.edu.ec.test;


import java.lang.Runtime.Version;
import java.time.ZonedDateTime;
import java.util.*;
import java.util.stream.Collector;
import ups.edu.ec.controlador.ControladorPersona;
import ups.edu.ec.modelo.InformacionTiempoMoneda;
import ups.edu.ec.modelo.Persona;


/**
 *
 * @author Adolfo
 */
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();




/*
Para el segundo ejemplo: Version
 */


System.out.println("\n----------Ejemplo de la versión----------");
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());
```

```
/*
Para el tercer ejemplo: Unicode Language
 */
System.out.println("\n---------------Informacion de la localizacion-Moneda-Tiempo----------------");

//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());




/*
Para el cuarto ejemplo: GC
 */
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();



/*
Para el quinto ejemplo: Copias de colecciones inmodificables
 */

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());

    }

}
```

3. **Resultados obtenidos.**
   - Para la inferencia de variables locales tenemos el siguiente ejemplo:

```java
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'
```

- Ahora para la versión de lanzamientos basados en la fecha:

```
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
```

- Para las extensiones de etiqueta del lenguaje Unicode:

```
//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
```

- Para el GarbageCollector:

```
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:
```

- Para las colecciones no modificables:

```
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
```

4. **Generar el informe de la práctica.**

RESULTADO(S) OBTENIDO(S):

- Ampliar el conocimiento de nuevos conceptos introducidos en las ultimas versiones de Java 10.
- Reconocer los cambios más importantes en Java para poder aplicarlos en nuestros programas.
- Aplicar mediante ejemplos cortos y sencillos de explicar los cambios producidos en Java 10.
- Generar un resumen de los cambios realizados en Java 10.
- Mejorar el código producido mediante la comprensión de los nuevos conceptos realizados en Java 10.

CONCLUSIONES:

- Fue un trabajo de investigación de mucha ayuda para poder comprender los nuevos cambios realizados en la versión de Java10. Esto nos ayuda para futuras prácticas de programación, ya que así podremos escribir códigos de Java de mayor facilidad de lectura y escritura. Además, esto nos ayudara para comprender códigos ya hechos en internet y que muchas veces no sabíamos cómo eran.
- Esto también nos servirá para acostumbrarnos a leer la documentación de las actualizaciones de java y así poder estar bien informados sobre ellas y también así poder aplicar los nuevos cambios introducidos en Java.

RECOMENDACIONES:

La única recomendación que tengo, es que si nos pudiera aclarar lo que debería estar en el informe fuera algo que nos ayudaría a los estudiantes ya que así supiéramos que información considera esencial el profesor.

***Nombre de estudiante*: _____**

***Firma de estudiante*: _____**