
	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: Patrones en Java																					
OBJETIVO: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas tecnicas de programación Entender los patrones de Java																							
INSTRUCCIONES (Detallar las instrucciones que se dará al estudiante):		1. Revisar los conceptos fundamentales de Java																					
		2. Establecer las características de Java basados en patrones de diseño																					
		3. Implementar y diseñar los nuevos patrones de Java																					
		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 Patrones de Diseño de Java																							
2. Diseñar e implementa cada estudiante un patron de diseño y verificar su funcionamiento. A continuación se detalla el patron a implementar:																							
<table border="1"> <thead> <tr> <th>Nombre</th> <th>Patron</th> </tr> </thead> <tbody> <tr> <td><u>NIXON ANDRES ALVARADO CALLE</u></td> <td>Factory Method</td> </tr> <tr> <td><u>ROMEL ANGEL AVILA FAICAN</u></td> <td>Builder</td> </tr> <tr> <td><u>JORGE SANTIAGO CABRERA ARIAS</u></td> <td>Abstract Factory</td> </tr> <tr> <td><u>EDITH ANAHI CABRERA BERMEO</u></td> <td>Prototype</td> </tr> <tr> <td><u>JUAN JOSE CORDOVA CALLE</u></td> <td>Chain of Responsibility</td> </tr> <tr> <td><u>DENYS ADRIAN DUTAN SANCHEZ</u></td> <td>Command</td> </tr> <tr> <td><u>JOHN XAVIER FAREZ VILLA</u></td> <td>Interpreter</td> </tr> <tr> <td><u>PAUL ALEXANDER GUAPUCAL CARDENAS</u></td> <td>Iterator</td> </tr> <tr> <td><u>PAUL SEBASTIAN IDROVO BERREZUETA</u></td> <td>Mediator</td> </tr> </tbody> </table>				Nombre	Patron	<u>NIXON ANDRES ALVARADO CALLE</u>	Factory Method	<u>ROMEL ANGEL AVILA FAICAN</u>	Builder	<u>JORGE SANTIAGO CABRERA ARIAS</u>	Abstract Factory	<u>EDITH ANAHI CABRERA BERMEO</u>	Prototype	<u>JUAN JOSE CORDOVA CALLE</u>	Chain of Responsibility	<u>DENYS ADRIAN DUTAN SANCHEZ</u>	Command	<u>JOHN XAVIER FAREZ VILLA</u>	Interpreter	<u>PAUL ALEXANDER GUAPUCAL CARDENAS</u>	Iterator	<u>PAUL SEBASTIAN IDROVO BERREZUETA</u>	Mediator
Nombre	Patron																						
<u>NIXON ANDRES ALVARADO CALLE</u>	Factory Method																						
<u>ROMEL ANGEL AVILA FAICAN</u>	Builder																						
<u>JORGE SANTIAGO CABRERA ARIAS</u>	Abstract Factory																						
<u>EDITH ANAHI CABRERA BERMEO</u>	Prototype																						
<u>JUAN JOSE CORDOVA CALLE</u>	Chain of Responsibility																						
<u>DENYS ADRIAN DUTAN SANCHEZ</u>	Command																						
<u>JOHN XAVIER FAREZ VILLA</u>	Interpreter																						
<u>PAUL ALEXANDER GUAPUCAL CARDENAS</u>	Iterator																						
<u>PAUL SEBASTIAN IDROVO BERREZUETA</u>	Mediator																						


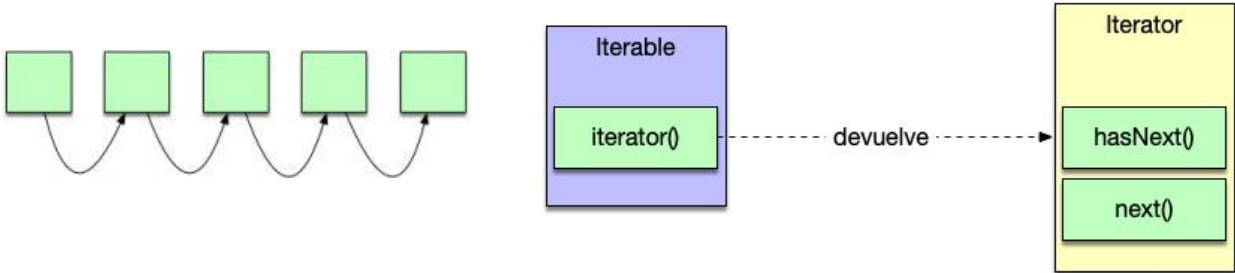
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

	<u>ADOLFO SEBASTIAN JARA GAVILANES</u>	Observer
	<u>ADRIAN BERNARDO LOPEZ ARIZAGA</u>	State
	<u>ESTEBAN DANIEL LOPEZ GOMEZ</u>	Strategy
	<u>GEOVANNY NICOLAS ORELLANA JARAMILLO</u>	Visitor
	<u>NELSON PAUL ORTEGA SEGARRA</u>	Adapter
	<u>BRYAM EDUARDO PARRA ZAMBRANO</u>	Bridge
	<u>LISSETH CAROLINA REINOSO BAJAÑA</u>	Composite
	<u>MARTIN SEBASTIAN TOLEDO TORRES</u>	Decorator
	<u>SEBASTIAN ROBERTO UYAGUARI RAMON</u>	Flyweight
	<u>ARIEL RENATO VAZQUEZ CALLE</u>	Proxy
	CHRISTIAN ABEL JAPON CHAVEZ	Facade
3. Probar y modificar el patron de diseño a fin de generar cuales son las ventajas y desventajas.		
4. Realizar práctica codificando los codigos de los patrones y su estructura.		
RESULTADO(S) OBTENIDO(S): Realizar procesos de investigación sobre los patrones de diseño de Java Entender los patrones y su utilización dentro de aplicaciones Java. Entender las funcionalidades basadas en patrones.		
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. Revisar el siguiente link: https://refactoring.guru/es/design-patterns/java		

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:	03	TÍTULO PRÁCTICA: Patrones Java	
OBJETIVO ALCANZADO: <ul style="list-style-type: none"> Identificar los cambios importantes de Java Diseñar e Implementar las nuevas técnicas de programación Entender los patrones de Java 			
ACTIVIDADES DESARROLLADAS			
1. Revisar la teoría y conceptos de Patrones de Diseño de Java			
2. Diseñar e implementar cada estudiante un patrón de diseño y verificar su funcionamiento. Patrón de comportamiento “Iterator” <ul style="list-style-type: none"> ¿Qué es un Iterator? <ul style="list-style-type: none"> ➤ Lo primero es tener claro que hay que distinguir el método iterator (en minúsculas) y el tipo definido en el api de Java Iterator (con mayúsculas). Iterator con mayúsculas es un tipo definido por la Interface Iterator, igual que List es un tipo definido por la interface List. Por el contrario, iterator () con minúsculas es un método igual que puede ser toString () o cualquier otro. Esto hay que tenerlo claro desde el principio para no llegar a falsas conclusiones. ➤ En programación de computadores, un iterator se refiere al objeto que permite al programador recorrer un contenedor, (una colección de elementos) particularmente listas. ➤ Varios tipos de iterators se suministran frecuentemente a través de una interfaz del contenedor. La interfaz y la semántica de un determinado iterator suelen ser fijas. Un iterator sigue una ruta y da acceso a elementos de datos del contenedor, pero no realiza iteración (es decir, no tiene total libertad). Un iterator se comporta como el cursor de una base de datos. Los iterators se empezaron a utilizar en el lenguaje de programación CLU en 1974. ➤ La interface Iterator también pertenece como las anteriores al framework Collections de Java. Esta interface nos permite iterar sobre una colección de elementos. Para ello hemos de implementar sus métodos, que son los siguientes: boolean hasNext(), E next(), void remove(). <ul style="list-style-type: none"> Interfaz Iterable <ul style="list-style-type: none"> ➤ El concepto de Java Iterable es un concepto clásico en el mundo Java y existe desde la versión de Java 1.5. Un Iterable es una interface que hace referencia a una colección de elementos que se puede recorrer, ni más ni menos. 			
			
<ul style="list-style-type: none"> ➤ Así pues, la interface solo necesita que implementemos un método para poder funcionar de forma correcta, este método es iterator (). 			

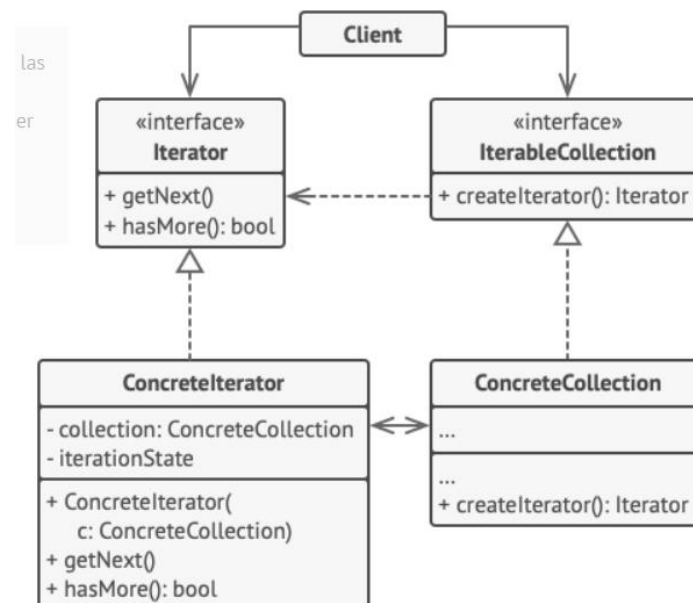
• Propósitos y Problemas

- Iterator es un patrón de diseño de comportamiento que te permite recorrer elementos de una colección sin exponer su representación subyacente (lista, pila, árbol, etc.).
- Las colecciones son de los tipos de datos más utilizados en programación. Sin embargo, una colección tan solo es un contenedor para un grupo de objetos.
- La mayoría de las colecciones almacena sus elementos en simples listas, pero algunas de ellas se basan en pilas, árboles, grafos y otras estructuras complejas de datos.
- Independientemente de cómo se estructure una colección, debe aportar una forma de acceder a sus elementos de modo que otro código pueda utilizar dichos elementos. Debe haber una forma de recorrer cada elemento de la colección sin acceder a los mismos elementos una y otra vez.
- Añadir más y más algoritmos de recorrido a la colección nubla gradualmente su responsabilidad principal, que es el almacenamiento eficiente de la información. Además, puede que algunos algoritmos estén personalizados para una aplicación específica, por lo que incluirlos en una clase genérica de colección puede resultar extraño.

• Soluciones para los problemas


- La idea central del patrón Iterator es extraer el comportamiento de recorrido de una colección y colocarlo en un objeto independiente llamado iterator.
- Un objeto iterator encapsula todos los detalles del recorrido, como la posición actual y cuántos elementos quedan hasta el final. Debido a esto, varios iterators pueden recorrer la misma colección al mismo tiempo, independientemente los unos de los otros.
- los iterators aportan un método principal para extraer elementos de la colección. El cliente puede continuar ejecutando este método hasta que no devuelva nada, lo que significa que el iterator ha recorrido todos los elementos.
- Todos los iterators deben implementar la misma interfaz. Esto hace que el código sea compatible con cualquier tipo de colección o cualquier algoritmo de recorrido, siempre y cuando exista un iterator adecuado.

• Estructura del patrón "Iterator"



3. Probar y modificar el patrón de diseño a fin de generar cuales son las ventajas y desventajas.

• Ventajas e Inconvenientes

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

- Permite variaciones en el recorrido del agregado encapsulando los algoritmos de recorrido en distintas subclases de iteradores.
- Simplifica la interfaz del agregado al no incluir las operaciones de iteración.
- Permite recorridos simultáneos con varios iteradores a la vez.
- Simplifica el acceso secuencial a los elementos de cualquier estructura de datos.
- En estructuras de datos no ordenadas, el orden de los elementos al ser recorridos puede ser diferente en diferentes recorridos, lo cual puede generar problemas si no se tiene en cuenta.
- Existen iteradores que crean una copia de la estructura de datos al ser creados, por si se modifica durante el recorrido.
- Se pueden implementar varios iteradores para que se pueda recorrer una colección compleja.

4. Realizar práctica codificando los códigos de los patrones y su estructura.

Ejemplo del funcionamiento del patrón "Iterator"

- Ec.edu.ups.modelo

- Clase Persona

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

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

    private int idPersona;
    private String nombre;
    private int altura;

    public Persona() {
    }

    public Persona(int idPersona, String nombre, int altura) {
        this.idPersona = idPersona;
        this.nombre = nombre;
        this.altura = altura;
    }

    public int getIdPersona() {
        return idPersona;
    }

    public void setIdPersona(int idPersona) {
        this.idPersona = idPersona;
    }

    public String getNombre() {
        return nombre;
    }

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

    public int getAltura() {
```

```
        return altura;
    }

    public void setAltura(int altura) {
        this.altura = altura;
    }

    @Override
    public String toString() {
        return "Persona{" + "idPersona=" + idPersona + ", nombre=" + nombre + ",
altura=" + altura + '}';
    }
}
```

- Clase Lista Personas

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

import java.util.ArrayList;
import java.util.Iterator;

/**
 *
 * @author paul_
 */
public class ListaPersonas {

    private ArrayList<Persona> listapersona = null; //Campo de la clase

    public ListaPersonas() { // Constructor de la clase

        listapersona = new ArrayList<Persona>();
    }

    public ArrayList<Persona> getListaPersonas() { //Omitimos otros métodos

        return listapersona;
    }

    protected class MiIteradorListaPersonas implements Iterator<Persona>{ // Clase
interna

        public int posicion = 0;
        public boolean sepuedeeliminar = false; // Campos

        @Override

        public boolean hasNext() {
            return posicion < listapersona.size();
        } // Método

        @Override
```



```
public Persona next() { // Método

    Persona res = listapersona.get(posicion); // Creamos un objeto Persona
    igual al que recorremos
    posicion++;
    sepuedeeliminar = true;
    return res;
}

@Override

public void remove() {

    if (sepuedeeliminar) {
        listapersona.remove(posicion - 1);
        posicion--;
        sepuedeeliminar = false;
    }

}

}

public Iterator<Persona> iterator() { // Método de la clase

    return new MiIteradorListaPersonas();

}

@Override
public String toString() { // Método de la clase

    return listapersona.toString();

}

}
```

- **Ec.edu.ups.test**


- **Clase Programa**

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

import ec.edu.ups.modelo.ListaPersonas;
import ec.edu.ups.modelo.Persona;
import java.util.Iterator;

/**
 *
 * @author paul_
 */
public class Programa {

    public static void main(String[] args) {
```

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

```

ListaPersonas lp = new ListaPersonas();
Iterator<Persona> it;
Persona e; // Este objeto lo usaremos para almacenar temporalmente objetos
Persona
lp.getListasPersonas().add(new Persona(1, "Maria", 175));
lp.getListasPersonas().add(new Persona(2, "Carla", 160));
lp.getListasPersonas().add(new Persona(3, "Enriqueta", 190));
System.out.println("\nLista antes de recorrer/eliminar: " + lp.toString());
it = lp.iterator();
while (it.hasNext()) {
    e = it.next();
    if (e.getAltura() < 170) {
        it.remove();
    } // it.remove() elimina la persona de la colección
}
System.out.println("\nLista después de recorrer/eliminar: " +
lp.toString());
}
}

```

RESULTADO(S) OBTENIDO(S):

- Realizar procesos de investigación sobre los patrones de diseño de Java
- Entender los patrones y su utilización dentro de aplicaciones Java.
- Entender las funcionalidades basadas en patrones.

CONCLUSIONES:

- Una manera más eficaz de manejar este tipo de aplicaciones dentro de NetBeans es con el uso de un controlador dentro del proyecto en sí, porque dentro de este se crearían los métodos que luego van a ser instanciados a las ventanas donde serán usados y a la vez requeridos para su ayuda dentro del código.
- La aplicación de patrones de diseño y arquitecturas dentro de la aplicación creada son fundamentales y facilitan el trabajo a la vez que simplifican el código para que no venga a ser muy pesado y tenga una mayor aceptabilidad dentro de los ámbitos en los que se deseen proyectar su respectivo uso.

RECOMENDACIONES:

- Utilizar la ID NetBeans para este tipo de aplicaciones es muy recomendable, puesto que su variedad de diccionarios, métodos, patrones, nos ahorran mucho tiempo en cuanto a el desarrollo de la interfaz del mismo, o más bien el entorno que será visualizado por el usuario.
- Realizar el trabajo dentro del tiempo establecido.

Nombre de estudiante:

PAUL ALEXANDER GUAPUCAL CARDENAS

Firma de estudiante:

