
	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:	4	TÍTULO PRÁCTICA: Patrones en 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			
<p>1. Revisar la teoría y conceptos de Patrones de Diseño de Java</p> <p>Que es abstract Factory</p> <p>Abstract Factory es un patrón de diseño creacional que resuelve el problema de crear familias enteras de productos sin especificar sus clases concretas.</p> <p>El patrón Abstract Factory define una interfaz para crear todos los productos, pero deja la propia creación de productos para las clases de fábrica concretas. Cada tipo de fábrica se corresponde con cierta variedad de producto.</p> <p>Para que nos sirve el patrón abstract factory</p> <p>El patrón de diseño Abstract Factory busca agrupar un conjunto de clases que tiene un funcionamiento en común llamadas familias, las cuales son creadas mediante un Factory, este patrón es especialmente útil cuando queremos tener ciertas familias de clases para resolver un problema.</p> <p>Aplicabilidad</p> <ul style="list-style-type: none"> • Utiliza el patrón Abstract Factory cuando tu código deba funcionar con varias familias de productos relacionados, pero no desees que dependa de las clases concretas de esos productos, ya que puede ser que no los conozcas de antemano o sencillamente quieras permitir una futura extensibilidad. • El patrón Abstract Factory nos ofrece una interfaz para crear objetos a partir de cada clase de la familia de productos. Mientras tu código cree objetos a través de esta interfaz, no tendrás que preocuparte por crear la variante errónea de un producto que no combine con los productos que ya ha creado tu aplicación. • Considera la implementación del patrón Abstract Factory cuando tengas una clase con un grupo de métodos de fábrica que nublen su responsabilidad principal. 			

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

Pros de utilizar el patrón

- Puedes tener la certeza de que los productos que obtienes de una fábrica son compatibles entre sí.
- Evitas un acoplamiento fuerte entre productos concretos y el código cliente.
- Principio de responsabilidad única. Puedes mover el código de creación de productos a un solo lugar, haciendo que el código sea más fácil de mantener.
- Aísla las clases de implementación: ayuda a controlar los objetos que se creen y encapsula la responsabilidad y el proceso de creación de objetos producto.
- Hace fácil el intercambio de familias de productos. Solo necesitaremos cambiar de Factory.
- Fomenta la consistencia entre productos.

Contras del patrón

- Puede ser que el código se complique más de lo que debería, ya que se introducen muchas nuevas interfaces y clases junto al patrón.
- Para añadir un nuevo producto, se requiere la implementación del interfaz y todos sus métodos.

2. Diseñar e implementa cada estudiante un patrón de diseño y verificar su funcionamiento. A continuación, se detalla el patrón a implementar:

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

Tenemos un menú donde le pida que ingrese una opción.

```

Seleccione una opcion
1.Servicio de diseno
2.Servicio de paginas web
3.Servicio Estudiantil
4.Salir
-----


```

Al seleccionar una opción nos dará la información que hemos puesto en los productos concretos en la clase Servicio de diseño.

```

Hemos asignado el trabajo a realizar.
La fecha limite de la entrega es el 25/11/2020.
El pago se realizara en efectivo.

```

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

Al seleccionar la opción 2 nos dará la información que hayamos puesto en el producto abstracto en la clase Servicio Páginas Web.

```
2
Hemos asignado el trabajo a realizar.
La fecha limite de la entrega es el 30/12/2020.
El pago se realizara en efectivo.
```

Al seleccionar la opción 3 nos dará la información que hayamos puesto en el producto abstracto en la clase Servicio Estudiantil.

```
3
El trabajo ha sido asignado
Fecha limite: 30/11/20
El pago sera dispuesto por la cantidad de estudiantes.
```

Las desventajas serian si queremos añadir un nuevo producto, se requiere la implementación del interfaz y todos sus métodos.

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

- Factoría Abstracta

```
package ec.ups.edu.diseño;

/**
 *
 * @author santi
 */
public interface ServicioCreador {


    public ServicioInformatico crearServicio();

}
```

- Producto Abstracto

```
package ec.ups.edu.diseño;

/**
 *
 * @author santi
```

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

```

*/
public interface ServicioInformatico {

    public void asignarTrabajo();
    public void IndicarFechaEntrega();
    public void informarSobrePago();

}

```

- **Factorías Concretas.**

- **Clase Diseño Factoría.**

```

package ec.ups.edu.productos;

import ec.ups.edu.diseño.ServicioCreador;
import ec.ups.edu.diseño.ServicioInformatico;

/**
 *
 * @author santi
 */
public class DisenoFactoria implements ServicioCreador{

    @Override
    public ServicioInformatico crearServicio() {

        return new ServicioDeDiseno();

    }

}

```

- **Clase estudiantil Factoria.**

```

package ec.ups.edu.productos;

import ec.ups.edu.diseño.ServicioCreador;
import ec.ups.edu.diseño.ServicioInformatico;

/**
 *
 * @author santi
 */
public class EstudiantilFactoria implements ServicioCreador{


    @Override
    public ServicioInformatico crearServicio() {

        return new ServicioEstudiatil();

    }

}

```


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

```
}
```

- **Clase Páginas web Factoría.**

```
package ec.ups.edu.productos;

import ec.ups.edu.diseño.ServicioCreador;
import ec.ups.edu.diseño.ServicioInformatico;

/**
 *
 * @author santi
 */
public class PaginasWebFactoria implements ServicioCreador{

    @Override
    public ServicioInformatico crearServicio() {

        return new ServicioDeDiseno();

    }

}
```

- **Producto abstracto**

- **Servicio de Diseño**

```
package ec.ups.edu.productos;

import ec.ups.edu.diseño.ServicioInformatico;


/**
 *
 * @author santi
 */
public class ServicioDeDiseno implements ServicioInformatico{

    @Override
    public void asignarTrabajo() {

        System.out.println("Hemos asignado el trabajo a realizar.");

    }

    @Override
    public void IndicarFechaEntrega() {
```

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

```

        System.out.println("La fecha limite de la entrega es el 25/11/2020.");
    }

    @Override
    public void informarSobrePago() {

        System.out.println("El pago se realizara en efectivo.");

    }

}

```

- **Servicio estudiantil**

```

package ec.ups.edu.productos;

import ec.ups.edu.diseño.ServicioInformatico;

/**
 *
 * @author santi
 */
public class ServicioEstudiantil implements ServicioInformatico{

    @Override
    public void asignarTrabajo() {

        System.out.println("El trabajo ha sido asignado");

    }

    @Override
    public void IndicarFechaEntrega() {

        System.out.println("Fecha limite: 30/11/20");

    }


    @Override
    public void informarSobrePago() {

        System.out.println("El pago sera dispuesto opr la cantidad de
estudiantes.");

    }

}

```

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

- **Servicio páginas web**

```
package ec.ups.edu.productos;

import ec.ups.edu.diseño.ServicioInformatico;

/**
 *
 * @author santi
 */
public class ServicioPaginasWeb implements ServicioInformatico{

    @Override
    public void asignarTrabajo() {

        System.out.println("El trabajo ha sido aceptado.");

    }

    @Override
    public void IndicarFechaEntrega() {

        System.out.println("La fecha a entregar el trabajo es el 30/11/2020");

    }

    @Override
    public void informarSobrePago() {

        System.out.println("El pago a realizar incluirea dominio y hosting.");

    }


}
```

- **Clase Test**

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

import ec.ups.edu.diseño.ServicioCreador;
import ec.ups.edu.diseño.ServicioInformatico;
import ec.ups.edu.productos.DisenoFactoria;
import ec.ups.edu.productos.EstudiantilFactoria;
import ec.ups.edu.productos.PaginasWebFactoria;
import java.util.Scanner;

/**
 *
 * @author santi
 */
public class Test {
```

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

```

private static Scanner sc = new Scanner(System.in);

public static void usarServicio(ServicioCreador factory) {
    ServicioInformatico servicio = factory.crearServicio();
    servicio.asignarTrabajo();
    servicio.IndicarFechaEntrega();
    servicio.informarSobrePago();
}

public static void main(String[] args) {

    int opcion = 0;

    do {

        System.out.println("\nSeleccione una opcion");
        System.out.println("1.Servicio de diseno");
        System.out.println("2.Servicio de paginas web");
        System.out.println("3.Servicio Estudiantil");
        System.out.println("4.Salir");
        System.out.println("-----");

        opcion = sc.nextInt();

        switch (opcion) {

            case 1:

                usarServicio(new DisenoFactoria());

                break;

            case 2:

                usarServicio(new PaginasWebFactoria());

                break;

            case 3:

                usarServicio(new EstudiantilFactoria());

                break;

            case 4:


                System.exit(0);

                break;

            default:

                System.out.println("Ingrese una opcion dentro del rango");

```

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

```

    }
    } while (opcion != 4);

}

}

```

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:

Como conclusión puedo decir que utilizar este patrón de diseño nos ayuda siempre y cuando tengamos clases que tengan un funcionamiento en común, es decir que utilizamos el patrón Abstract Factory cuando tu código deba funcionar con varias familias de productos relacionados.

RECOMENDACIONES:

Recomiendo utilizar este patrón de diseño siempre y cuando estemos yendo a crear un programa en el cual tengamos clases que tengan un funcionamiento en común.

Nombre de estudiante: Jorge Santiago Cabrera Arias.

Firma de estudiante:

