
	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: _____

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

Firma: _____

		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: Identificar los cambios importantes de Java Diseñar e Implementar las nuevas tecnicas de programación Entender los patrones de Java			
ACTIVIDADES DESARROLLADAS			
1. Revisar la teoría y conceptos de Patrones de Diseño de Java Son una forma estandarizada para representar soluciones generales de problemas que se encuentran comúnmente en el desarrollo de software orientado a objetos. Adaptan la solución general al contexto del problema actual.			
2. Probar y modificar el patrón de diseño a fin de generar cuales son las ventajas y desventajas. El patrón Builder resulta especialmente útil cuando debes crear un objeto con muchas opciones posibles de configuración. Este patrón de diseño separa la creación de un objeto de su representación, de modo que el mismo proceso de construcción puede crear diferentes representaciones. Estructura: <ul style="list-style-type: none"> - Interface Builder: Especificación de una interfaz abstracta para crear partes de un objeto. - Concrete Builder: Es para construir y ensamblar partes de un objeto para implementación de la interfaz Builder, Provee una interfaz para recuperar el objeto. - Director: Se encarga de construir un objeto es la clase donde se ensambla todo el objeto. - Producto: Objeto que se ha construido tras el proceso definido por el patrón. Ventajas: <ul style="list-style-type: none"> - Permite variar la representación interna de un producto. - Encapsula el código de construcción y de representación. Los clientes no necesitan saber nada sobre las clases que definen la estructura interna del producto. - Proporciona un control más explícito sobre el proceso de construcción. A diferencia de los demás patrones de creación, que construyen los productos una sola vez, el patrón Builder construye el producto paso a paso, bajo el control del director. 			

Desventajas:

- Hay que crear un BuilderCorrecto para cada representación de un producto, lo que puede acabar con multitud de clases.

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

Diagrama:

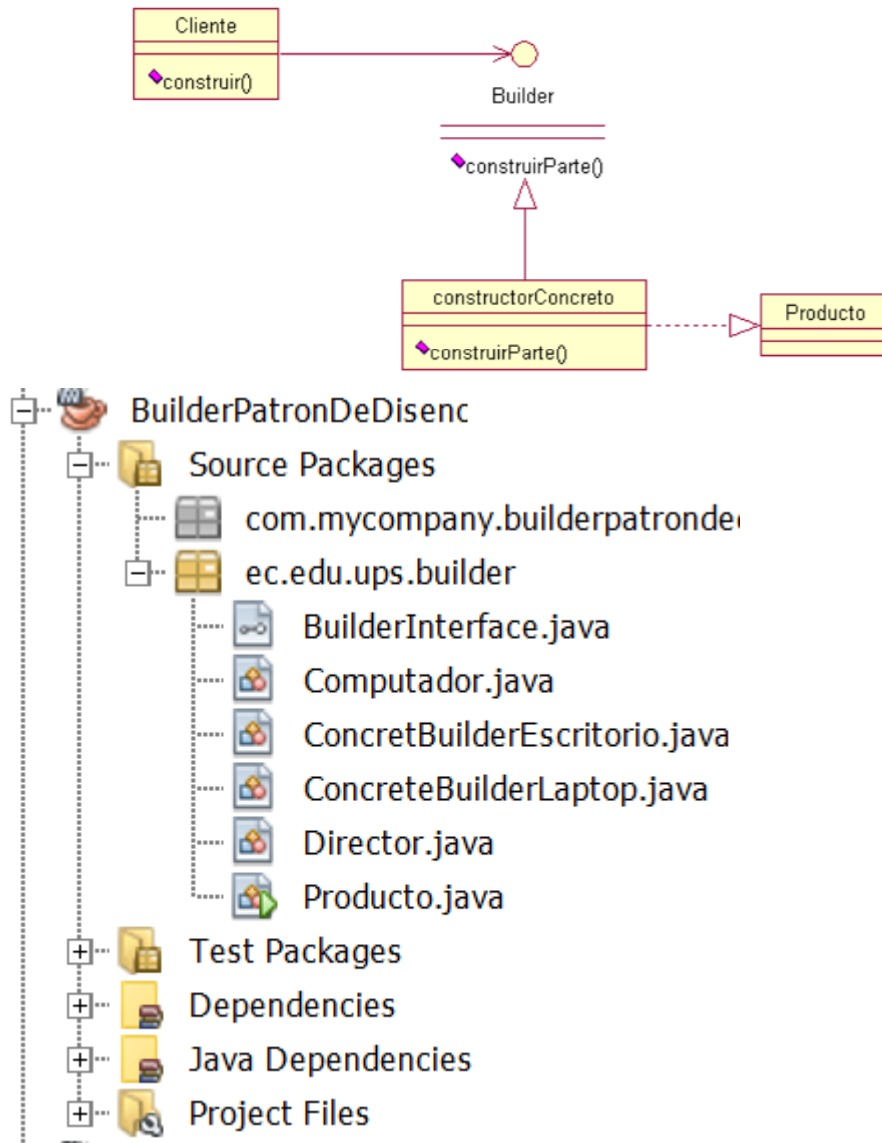


Imagen de la creación de las clases y la interface que componen la estructura del patrón de diseño builder


Interfaz Builder

```
public interface BuilderInterface {  
    public void buildBaterly ();  
    public void buildDisk ();  
    public void buildMemory ();  
    public void buildSO ();  
    Computador getObject ();  
}
```

Código de la clase Interface builder donde se especifican los métodos abstractos para la creación del producto

Concrete Builder Escritorio

```
package ec.edu.ups.builder;  
  
/**  
 *  
 * @author NANCY  
 */  
public class ConcretBuilderEscritorio implements BuilderInterface {  
    private Computador compu= new Computador("Computadora de Escritorio");  
  
    @Override  
    public void buildBaterly() {  
        compu.setBaterly(false);  
    }  
  
    @Override  
    public void buildDisk() {  
        compu.setDisk("Disco del ordenador 5 TB ");  
    }  
  
    @Override  
    public void buildMemory() {  
        compu.setMemory("Memoria del ordenador 8 GB");  
    }  
  
    @Override  
    public void buildSO() {  
        compu.setSo("Linux");  
    }  
  
    @Override  
    public Computador getObject() {  
        return compu;  
    }  
}
```

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

Código de una de las clases concrete builder “Escritorio” clase donde se ensambla uno de los productos psandoles sus respectivos datos

Concrete Builder Laptop

```
package ec.edu.ups.builder;

/**
 *
 * @author NANCY
 */
public class ConcreteBuilderLaptop implements BuilderInterface{

    private Computador compu= new Computador("Laptop");

    @Override
    public void buildBaterly() {
        compu.setBaterly(true);
    }

    @Override
    public void buildDisk() {
        compu.setDisk("Disco del ordenador 1 TB ");
    }

    @Override
    public void buildMemory() {
        compu.setMemory("Memoria del ordenador 8 GB");
    }

    @Override
    public void buildSO() {
        compu.setSo("Windous");
    }

    @Override
    public Computador getObject() {
        return compu;
    }


}
```

Código de una de las clases concrete builder “Laptop” clase donde se ensambla uno de los productos psandoles sus respectivos datos

Clase Director

```
package ec.edu.ups.builder;
```

```
/**
 *
```


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

```
* @author NANCY
*/
```

```
public class Director {

    private BuilderInterface builderComputador;

    public Director(BuilderInterface bulderComputador){
        this.builderComputador = bulderComputador;
    }
    public Computador builder(){
        builderComputador.buildBatory();
        builderComputador.buildDisk();
        builderComputador.buildMemory();
        builderComputador.buildSO();
        return builderComputador.getObject();
    }
}
```

Se encarga de construir un objeto es la clase donde se ensambla todo el objeto, Se encarga de construir un objeto utilizando el Constructor Builder.

RESULTADO(S) OBTENIDO(S):

Generar un problema para poder crear objetos o productos de una manera mas eficaz

CONCLUSIONES:

- Permite variar la representación interna del objeto, respetando la clase builder. Es decir, conseguimos independizar la construcción de la representación.

RECOMENDACIONES:

Denominar bien los métodos sen la clase interfaz

Estudiantes: Romel Ávila

Firma:

