


NOMBRE DE LA ASIGNATURA	Patrones De Diseño de Software							
NOMBRE DE LA ACTIVIDAD	Introducción a los patrones de diseño							
TIPO DE ACTIVIDAD	Sincrónica		Asincrónica	x	Individual	x	Grupal	
TEMÁTICA REQUERIDA PARA LA ACTIVIDAD			OBJETIVOS					
<ul style="list-style-type: none"> ○ Principios universales de diseño de software ○ Principios SOLID ○ Antipatrones ○ Patrón Factory Method 			<p>Desarrollar refactorización de aplicación para resolver o mejorar aspectos de diseño, mediante la implementación de patrón de diseño Factory Method.</p>					
COMPETENCIAS			INSUMOS PARA EL DESARROLLO DE LA ACTIVIDAD / REFERENCIAS BIBLIOGRÁFICAS					
<ul style="list-style-type: none"> ● Aplicación de principios de diseño SOLID ● Factory Method 			<ul style="list-style-type: none"> ● Material educativo y material complementario de la asignatura “Unidad 3.” ● Fuentes bibliográficas del módulo. 					
CONOCIMIENTOS PREVIOS REQUERIDOS								
Conceptos básicos sobre principios de diseño, patrones y antipatrones de diseños, refactorización								
ESPECIFICACIONES DE LA ACTIVIDAD								
<p>Refactorización aplicación de procesamiento de pagos:</p> <p>Se desea realizar una refactorización al código de una aplicación de procesamiento de pagos, debido a que el equipo de diseño identifico en el código desarrollado algunos problemas que puede afectar la reutilización, la extensibilidad y acoplamiento. Dentro de los problemas encontrados en el código existe:</p> <ul style="list-style-type: none"> ● Violación del principio Open/Closed ● Alto acoplamiento ● Difícil de extender ● Lógica de procesamiento mezclada ● Múltiples responsabilidades en un solo método <p>Este ejercicio solicita una refactorización completa utilizando el patrón Factory Method, transformando el código con múltiples condicionales a un diseño flexible y extensible. Beneficios perseguidos con la refactorización:</p> <ul style="list-style-type: none"> ● Cumplimiento del principio Open/Closed ● Alta cohesión y bajo acoplamiento ● Fácil extensión para nuevos métodos de pago ● Separación clara de responsabilidades ● Código más mantenible y legible 								

- Implementar una API de pagos

A continuación, se presenta el componente clave de la aplicación que presenta los problemas de diseño, anteriormente mencionados:

java

 Copiar

```
public class PaymentProcessor {
    public double processPayment(String paymentType, double amount) {
        double finalAmount = 0.0;

        if (paymentType.equals("CREDIT_CARD")) {
            // Lógica de procesamiento para tarjeta de crédito
            double commissionRate = 0.03;
            finalAmount = amount + (amount * commissionRate);
            System.out.println("Procesando pago con tarjeta de crédito");

            // Validaciones específicas
            if (amount > 1000) {
                finalAmount += 10; // Cargo adicional
            }
        }
        else if (paymentType.equals("DEBIT_CARD")) {
            // Lógica de procesamiento para tarjeta de débito
            double commissionRate = 0.01;
            finalAmount = amount + (amount * commissionRate);
            System.out.println("Procesando pago con tarjeta de débito");

            // Validaciones específicas
            if (amount > 500) {
                finalAmount += 5; // Cargo adicional
            }
        }
        else if (paymentType.equals("PAYPAL")) {
            // Lógica de procesamiento para PayPal
            double commissionRate = 0.02;
            finalAmount = amount + (amount * commissionRate);
            System.out.println("Procesando pago con PayPal");

            // Validaciones específicas
            if (amount > 750) {
                finalAmount += 7; // Cargo adicional
            }
        }
        else {
            throw new IllegalArgumentException("Método de pago no soportado");
        }

        return finalAmount;
    }
}
```

**RECOMENDACIONES /
OBSERVACIONES**

La actividad se desarrollará en grupos de dos estudiantes, se deberá sustentar los resultados de su trabajo.

Puede utilizar el lenguaje o framework de su preferencia.

En todo caso, se deberá incluir un enlace GitHub a repositorio con el código de muestra y también la bibliografía consultada.

La actividad será sustentada en el aula.

Elaboro: Ing. Jairo Seoanes, Msc Ingeniería de Sistemas y Computación