

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**“Patrones de Diseño”**

Curso: Calidad y Pruebas de Software

Docente: Mag. Ing. Patrick Cuadros Quiroga

**Integrantes:**

* **Valdez Apaza, Rafael Jesús (2019063317)**
* **Poma Chura, Jhon Romario** **(2019064022)**
* **Sagua Ramos, Gustavo Alonso (2018062254)**
* **Yucra Mamani, Vanessa (2019063635)**
* **Romero Roque, Angelica Beatriz (2019063327)**

**Tacna – Perú**

**2021**

**Índice**

[Resumen 3](#_Toc81331491)

[Abstract 3](#_Toc81331492)

[Introducción 3](#_Toc81331493)

[Desarrollo 3](#_Toc81331494)

[Patrones de diseño estructural 3](#_Toc81331495)

[Patrones de diseño de comportamiento 4](#_Toc81331496)

[Conclusiones 5](#_Toc81331497)

[Recomendaciones 5](#_Toc81331498)

[Bibliografía 5](#_Toc81331499)

## Resumen

## En este documento se hablará sobre los patrones de diseño y sus tipos. ¿Porque usar patrones de diseño? El gran crecimiento del sector de las tecnologías de la información ha hecho que las prácticas de desarrollo de software evolucionen. Antes se requería completar todo el software antes de realizar pruebas, lo que suponía encontrarse con problemas. Para ahorrar tiempo y evitar volver a la etapa de desarrollo una vez que este ha finalizado, se introdujo una práctica de prueba durante la fase de desarrollo. Esta práctica se usa para identificar condiciones de error y problemas en el código que pueden no ser evidentes en ese momento. En definitiva, los patrones de diseño nos ayudan a estar seguro de la validez de tu código, ya que son soluciones que funcionan y han sido probados por muchísimos desarrolladores siendo menos propensos a errores.

## Abstract

## This document will discuss design patterns and their types. Why use design patterns? The tremendous growth of the information technology industry has caused software development practices to evolve. Previously, all software was required to be completed before testing, which led to problems. To save time and avoid going back to the development stage after the development has finished, a test practice was introduced during the development stage. This practice is used to identify error conditions and problems in your code that may not be apparent at the time. In short, design patterns help us to be sure of the validity of your code, since they are solutions that work and have been tested by many developers, being less prone to errors.

## Introducción

Los patrones de diseño (design patterns) son soluciones habituales a problemas comunes en el diseño de software. Cada patrón es como un plano que se puede personalizar para resolver un problema de diseño particular de tu código.

También acelerar el proceso de desarrollo al proporcionar paradigmas de desarrollo probados y comprobados. El diseño de software eficaz requiere considerar problemas que pueden no ser visibles hasta más adelante en la implementación. La reutilización de patrones de diseño ayuda a prevenir problemas sutiles que pueden causar problemas importantes y mejora la legibilidad del código para codificadores y arquitectos familiarizados con los patrones.

## Desarrollo

## 

## Patrones de diseño creacional

Estos patrones de diseño tienen que ver con la instanciación de clases. Este patrón se puede dividir en patrones de creación de clases y patrones de creación de objetos. Mientras que los patrones de creación de clases usan la herencia de manera efectiva en el proceso de creación de instancias, los patrones de creación de objetos usan la delegación de manera efectiva para hacer el trabajo.

* Abstract Factory: Crea una instancia de varias familias de clases
* Generador: Separa la construcción de objetos de su representación
* Método de fábrica: Crea una instancia de varias clases derivadas
* Grupo de objetos: Evite la adquisición costosa y la liberación de recursos mediante el reciclaje de objetos que ya no están en uso.
* Prototipo: Una instancia completamente inicializada para ser copiada o clonada
* Singleton: Una clase de la que solo puede existir una instancia

## Patrones de diseño estructural

Estos patrones de diseño tienen que ver con la composición de clases y objetos. Los patrones de creación de clases estructurales utilizan la herencia para componer interfaces. Los patrones de objetos estructurales definen formas de componer objetos para obtener una nueva funcionalidad.

* Adaptador: Match interfaces de diferentes clases
* Puente: Separa la interfaz de un objeto de su implementación.
* Compuesto: Una estructura de árbol de objetos simples y compuestos.
* Decorador: Agregar responsabilidades a los objetos de forma dinámica
* Fachada: Una sola clase que representa un subsistema completo
* Flyweight: Una instancia detallada que se utiliza para compartir de forma eficiente
* Los datos de la clase privada: restringen el acceso de accesor / mutador
* Proxy: Un objeto que representa a otro objeto

## Patrones de diseño de comportamiento

Estos patrones de diseño tienen que ver con la comunicación de objetos de Class. Los patrones de comportamiento son aquellos patrones que están más específicamente relacionados con la comunicación entre objetos.

* Cadena de responsabilidad: Una forma de pasar una solicitud entre una cadena de objetos.
* Comando: Encapsular una solicitud de comando como un objeto
* Intérprete: Una forma de incluir elementos del lenguaje en un programa.
* Iterador Accede: secuencialmente a los elementos de una colección
* Mediador: Define la comunicación simplificada entre clases
* Memento: Capture y restaure el estado interno de un objeto
* Objeto nulo: Diseñado para actuar como valor predeterminado de un objeto
* Observador: Una forma de notificar cambios a varias clases.
* Estado: Altera el comportamiento de un objeto cuando cambia su estado
* Estrategia: Encapsula un algoritmo dentro de una clase
* Método de plantilla: Diferir los pasos exactos de un algoritmo a una subclase
* Visitante: Define una nueva operación para una clase sin cambios

## Conclusiones

* La realidad es que podrías trabajar durante años como programador sin conocer un solo patrón. Mucha gente lo hace. Incluso en ese caso, podrías estar implementando patrones sin saberlo.
* Los patrones de diseño son un juego de herramientas de soluciones comprobadas a problemas habituales en el diseño de software. Incluso aunque nunca te encuentres con estos problemas, conocer los patrones sigue siendo de utilidad, porque te enseña a resolver todo tipo de problemas utilizando principios del diseño orientado a objetos.
* Los patrones de diseño definen un lenguaje común que puedes utilizar con tus compañeros de equipo para comunicaros de forma más eficiente. Podrías decir: “Oh, utiliza un singleton para eso”, y todos entenderían la idea de tu sugerencia. No habría necesidad de explicar qué es un singleton si conocen el patrón y su nombre.

## Recomendaciones

* Amplía tu conocimiento de programación. Los patrones te permiten personalizar soluciones ya conocidas en lugar de tener que reinventar la rueda. Tu código tiene menos errores porque utilizas una solución probada y estandarizada que cubre todos los problemas ocultos.
* Para una buena comunicación con los colegas Programadores. Indica el nombre del patrón en lugar de perder una hora explicando a otros programadores los detalles de tu estupendo diseño y sus clases. Consigue la gloria sin el sudor.

## Bibliografía

**LIBROS SOBRE PATRONES DE DISEÑO:**

* **Design Patterns, de Gamma y amigos:** según Christopher Alexander, “cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo 2 veces”.
* **UML and Patterns, de Larman:**

Implementation Patterns, de Kent Beck

C++ Coding Standards, de Sutter & Alexandrescu

Applying Domain-Driven Design and Patterns, de Jimmy Nilsson

Patrones de Diseño: elementos de software orientado a Objetos reutilizables

Patrones de diseño en Java por Steven Metsker

Patrones de programación de juegos