**UNIVERSIDAD PRIVADA DE TACNA**



**FACULTAD DE INGENIERIA**

**Escuela Profesional de Ingeniería de Sistemas**

**Trabajo Encargado Nº 01**

**“Patrones de Diseño”**

Curso: Calidad y pruebas de software

Docente: Ing. Patrick José cuadros

INTEGRANTES:

**-Joel Robert Ccalli Chata (2017057528)**

**-Tell Ivan Casilla Maquera (2017057888)**

**-Jesus leonel Castro Gutierrez(2018062481)**

**-Arturo José Bedoya Hume (2015052678)**

**-Luis Fernando Bedoya Escudero(2019063853)**

**Tacna – Perú**

**2021**

***1.-Resumen***

El concepto de patrón de diseño lleva existiendo desde finales de los 70, pero su verdadera popularización surgió en los 90 con el lanzamiento del libro de [Design Pattern](http://devexperto.com/design-patterns) de la [Banda de los Cuatro](https://es.wikipedia.org/wiki/Gang_of_Four_(dise%C3%B1o)) (Gang of Four), que aunque parezca que estamos hablando de los Trotamúsicos, es el nombre con el que se conoce a los creadores de este libro: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. En él explican 23 patrones de diseño, que desde entonces han sido un referente.



***2.-Abstract***

The concept of design pattern has been around since the late 70s, but its true popularization emerged in the 90s with the launch of the Design Pattern book by the Gang of Four, which although it seems that we are talking about the Trotamúsicos, is the name by which the creators of this book are known: Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides. In it they explain 23 design patterns, which have been a benchmark ever since.

***3.-Introduccion***

Los patrones de diseño representan una mejor practica , a menudo es adoptado por los desarrolladores de software orientado a objetos experimentados.

Los patrones de diseño son la soluciones a los problemas comunes de los desarrolladores de software en el proceso del desarrollo en la cara. Estas soluciones son muchos desarrolladores de software a través de un largo periodo de prueba y error.

Los patrones de diseño se fijan para ser utilízalo en varias ocasiones , la mayoría de la gente sabe , después de catalogación , sumarias código de diseño de la experiencia . Utilizando los patrones de diseño para la reutilización de código , que el código sea mas fácil de comprender a los demás , para garantizar la facilidad del código. Sin lugar a duda , los patrones de diseño tiene sobre los demás en el sistema son de ganar

, patrones de diseño hace que el código compilado real de ingeniería , patrones de diseño es la piedra angular de la ingeniería de software, como la construcción de una piedra de ladrillo misma. Proyecto de uso racional de los patrones de diseño puede ser la solución perfecta para muchos problemas, cada modo correspondiente al principio de la realidad tienen que corresponde, cada modo describe un problema que se repite constantemente a nuestro alrededor, así como el problema solución central, que es ampliamente utilizado patrones de diseño que puede ser la razón.

***4.-Desarrollo:***

Los patrones de diseño más utilizados se clasifican en tres categorías principales, cada patrón de diseño individual conforma un total de 23 patrones de diseño. Las tres categorías principales son:

**-Patrones creacionales:**

Los patrones de creación proporcionan diversos mecanismos de creación de objetos, que aumentan la flexibilidad y la reutilización del código existente de una manera adecuada a la situación. Esto le da al programa más flexibilidad para decidir qué objetos deben crearse para un caso de uso dado.

**Estos son los patrones creacionales:**

**Abstract Factory**

En este patrón, una interfaz crea conjuntos o familias de objetos relacionados sin especificar el nombre de la clase.

**Builder Patterns**

Permite producir diferentes tipos y representaciones de un objeto utilizando el mismo código de construcción. Se utiliza para la creación etapa por etapa de un objeto complejo combinando objetos simples. La creación final de objetos depende de las etapas del proceso creativo, pero es independiente de otros objetos.

**Factory Method**

Proporciona una interfaz para crear objetos en una superclase, pero permite que las subclases alteren el tipo de objetos que se crearán. Proporciona instanciación de objetos implícita a través de interfaces comunes

**Prototype**

Permite copiar objetos existentes sin hacer que su código dependa de sus clases. Se utiliza para restringir las operaciones de memoria / base de datos manteniendo la modificación al mínimo utilizando copias de objetos.

**Singleton**

Este patrón de diseño restringe la creación de instancias de una clase a un único objeto.

**-Patrones estructurales:**

Facilitan soluciones y estándares eficientes con respecto a las composiciones de clase y las estructuras de objetos. El concepto de herencia se utiliza para componer interfaces y definir formas de componer objetos para obtener nuevas funcionalidades.

**Adapter**

Se utiliza para vincular dos interfaces que no son compatibles y utilizan sus funcionalidades. El adaptador permite que las clases trabajen juntas de otra manera que no podrían al ser interfaces incompatibles.

**Bridge**

En este patrón hay una alteración estructural en las clases principales y de implementador de interfaz sin tener ningún efecto entre ellas. Estas dos clases pueden desarrollarse de manera independiente y solo se conectan utilizando una interfaz como puente.

**Composite**

Se usa para agrupar objetos como un solo objeto. Permite componer objetos en estructuras de árbol y luego trabajar con estas estructuras como si fueran objetos individuales.

**Decorator**

Este patrón restringe la alteración de la estructura del objeto mientras se le agrega una nueva funcionalidad. La clase inicial permanece inalterada mientras que una clase decorator proporciona capacidades adicionales.

**Facade**

Proporciona una interfaz simplificada para una biblioteca, un marco o cualquier otro conjunto complejo de clases.

**Flyweight**

El patrón Flyweight se usa para reducir el uso de memoria y mejorar el rendimiento al reducir la creación de objetos. El patrón busca objetos similares que ya existen para reutilizarlo en lugar de crear otros nuevos que sean similares.

**Proxy**

Se utiliza para crear objetos que pueden representar funciones de otras clases u objetos y la interfaz se utiliza para acceder a estas funcionalidades

**-Patrones de comportamiento:**

El patrón de comportamiento se ocupa de la comunicación entre objetos de clase. Se utilizan para detectar la presencia de patrones de comunicación ya presentes y pueden manipular estos patrones.

Estos patrones de diseño están específicamente relacionados con la comunicación entre objetos.

**Chain of responsibility**

El patrón de diseño Chain of Responsibility es un patrón de comportamiento que evita acoplar el emisor de una petición a su receptor dando a más de un objeto la posibilidad de responder a una petición.

**Command**

Convierte una solicitud en un objeto independiente que contiene toda la información sobre la solicitud. Esta transformación permite parametrizar métodos con diferentes solicitudes, retrasar o poner en cola la ejecución de una solicitud y respaldar operaciones que no se pueden deshacer.

**Interpreter**

Se utiliza para evaluar el lenguaje o la expresión al crear una interfaz que indique el contexto para la interpretación.

**Iterator**

Su utilidad es proporcionar acceso secuencial a un número de elementos presentes dentro de un objeto de colección sin realizar ningún intercambio de información relevante.

**Mediator**

Este patrón proporciona una comunicación fácil a través de su clase que permite la comunicación para varias clases.

**Memento**

El patrón Memento permite recorrer elementos de una colección sin exponer su representación subyacente.

**Observer**

Permite definir un mecanismo de suscripción para notificar a varios objetos sobre cualquier evento que le suceda al objeto que está siendo observado.

**State**

En el patrón state, el comportamiento de una clase varía con su estado y, por lo tanto, está representado por el objeto de contexto.

**Strategy**

Permite definir una familia de algoritmos, poner cada uno de ellos en una clase separada y hacer que sus objetos sean intercambiables.

**Template method**

Se usa con componentes que tienen similitud donde se puede implementar una plantilla del código para probar ambos componentes. El código se puede cambiar con pequeñas modificaciones.

**Visitor**

El propósito de un patrón Visitor es definir una nueva operación sin introducir las modificaciones a una estructura de objeto existente.

***5.-Conclusiones:***

En conclusión, los patrones de diseño son las soluciones a los problemas comunes de los desarrolladores de software en el proceso del desarrollo en la cara así mismo estas soluciones son muchos desarrolladores de software a través de un largo periodo de prueba y error.

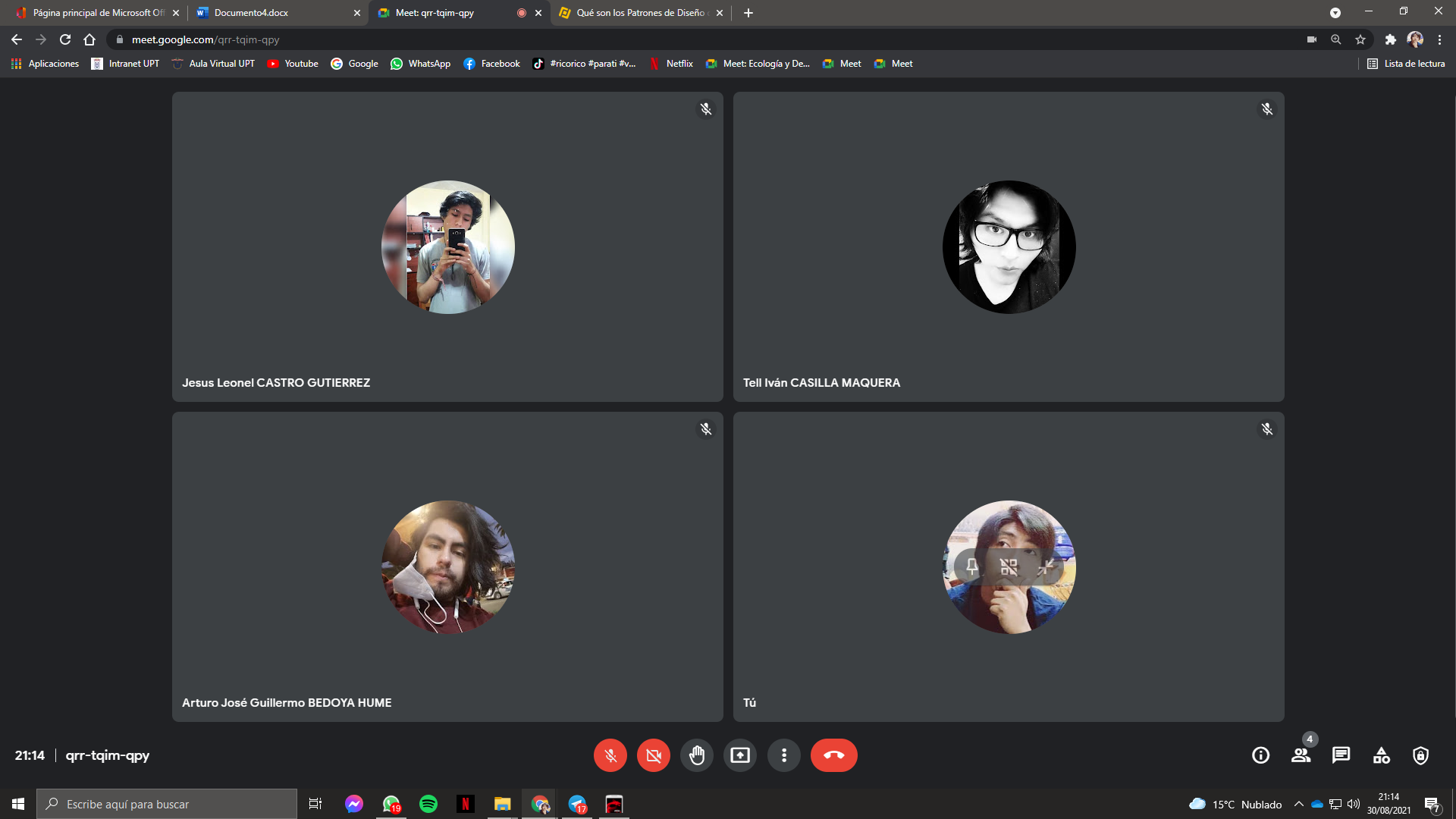
En conclusión, los patrones de diseño más utilizados se clasifican en tres categorías principales y son: **Patrones creacionales Patrones estructurales y Patrones de comportamiento.**

En conclusión los patrones de creación proporcionan diversos mecanismos de creación de objetos, que aumentan la flexibilidad y la reutilización del código existente de una manera adecuada a la situación.

***6.-Recomendaciones:***

* No abusar de los patrones de diseño , no intentes de usar un patrón de diseño solo por que lo conoces , te puede complicar mas que ayudar . Los patrones de diseño son una platilla de soluciones , los puedes usar adaptándolos a tu problema particular , pero respetando siempre el concepto sobre el que descansa si tienes que modificar demasiadas cosas de un patrón para que se adapte a tu problema , quizás sea mejor no utilizarlo

*x.-Evidencias de Trabajo Grupal:*



***7.-Bibliografía:***

*Fuente de información:*

[***https://es.wikipedia.org/wiki/Patr%C3%B3n\_de\_dise%C3%B1o***](https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o)

*Fuente de información:*

[***https://devexperto.com/patrones-de-diseno-software/***](https://devexperto.com/patrones-de-diseno-software/)

*Fuente de información:*

[***http://www.w3big.com/es/design-pattern/design-pattern-intro.html***](http://www.w3big.com/es/design-pattern/design-pattern-intro.html)

*Fuente de información:*

[***http://www3.uji.es/~belfern/Docencia/Presentaciones/ProgramacionAvanzada/Tema2/conceptoPatronDiseno.html#23***](http://www3.uji.es/~belfern/Docencia/Presentaciones/ProgramacionAvanzada/Tema2/conceptoPatronDiseno.html#23)

