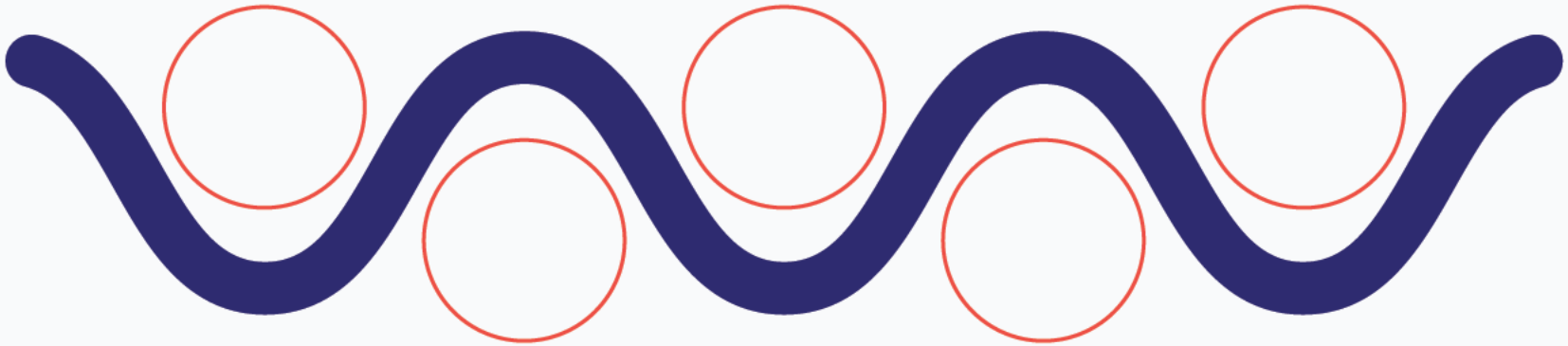
Johel Heraclio Batista Cárdenas
Cédula: 8-914-587
Grupo: 1IF-131

Los Patrones de Diseño en la Ingeniería de Software: Caso Prototype

Enlace de la Charla:
https://youtu.be/5I0DdMEeO9A

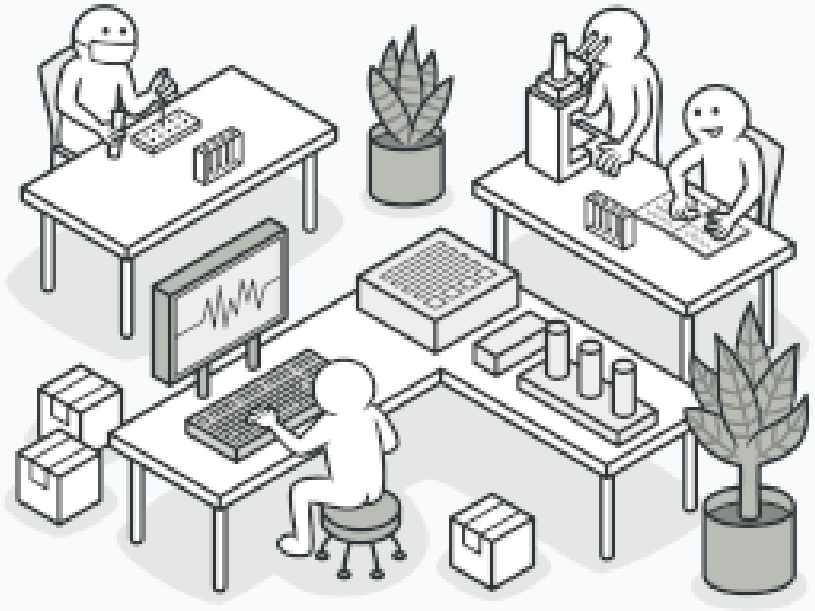
@ayudinga

Patrones de Diseño



¿Qué son los Patrones de Diseño en la Ing. De Software

Los patrones de diseño son esencialmente plantillas o guías de alto nivel que describen cómo abordar problemas comunes y recurrentes en la ingeniería de software. Estos patrones no representan una solución final o un código específico que se pueda reutilizar directamente, sino que sirven como un esquema o marco de referencia para solucionar problemas particulares.



Equipo de Desarrollo de Producto

@batistajohel

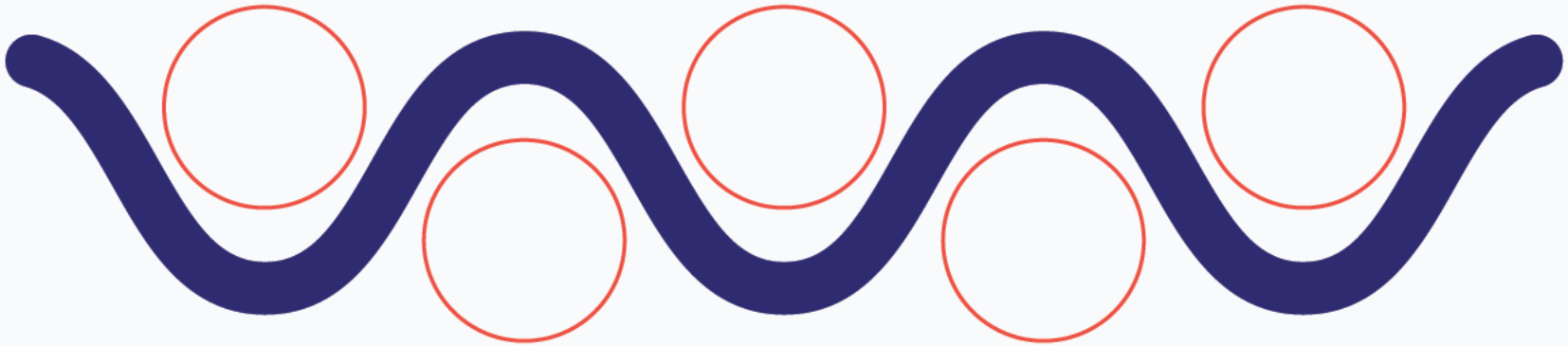
Claves de los Patrones de Diseño

Equipo de Desarrollo de Producto

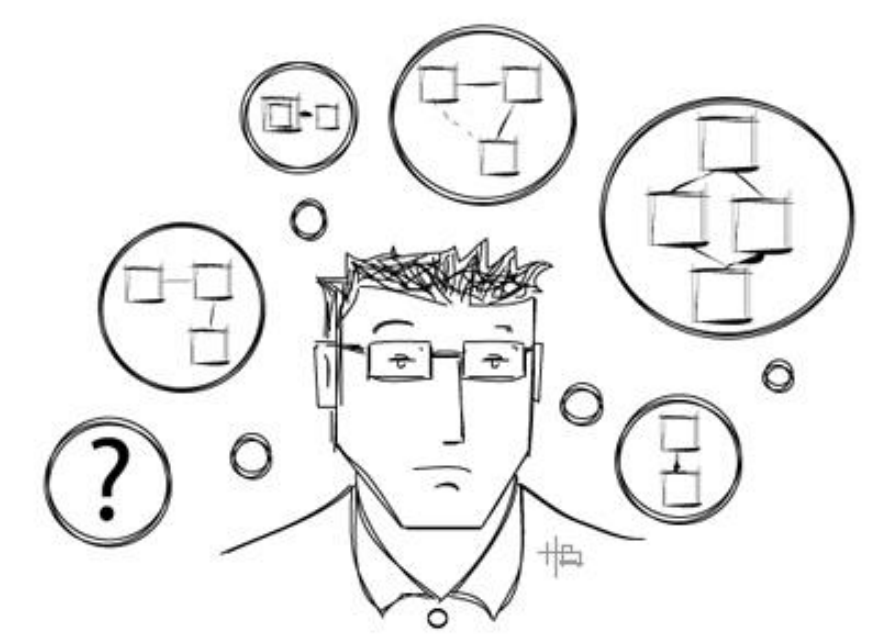
1. **Reutilización y Eficiencia**: Los patrones de diseño representan soluciones optimizadas y reutilizables a problemas comunes en el diseño de software, lo que mejora la eficiencia en el desarrollo y el mantenimiento del software.
2. **Comunicación y Documentación**: Los patrones de diseño proporcionan un lenguaje común que facilita la comunicación entre los desarrolladores y la documentación de los sistemas de software.
3. **Aplicación Cuidadosa**: Los patrones de diseño deben aplicarse con buen juicio y comprensión del problema a resolver, no todos los patrones son apropiados para todas las situaciones.

@ayudinga

Historia de los Patrones de Diseño



Patrones de Diseño | Historia

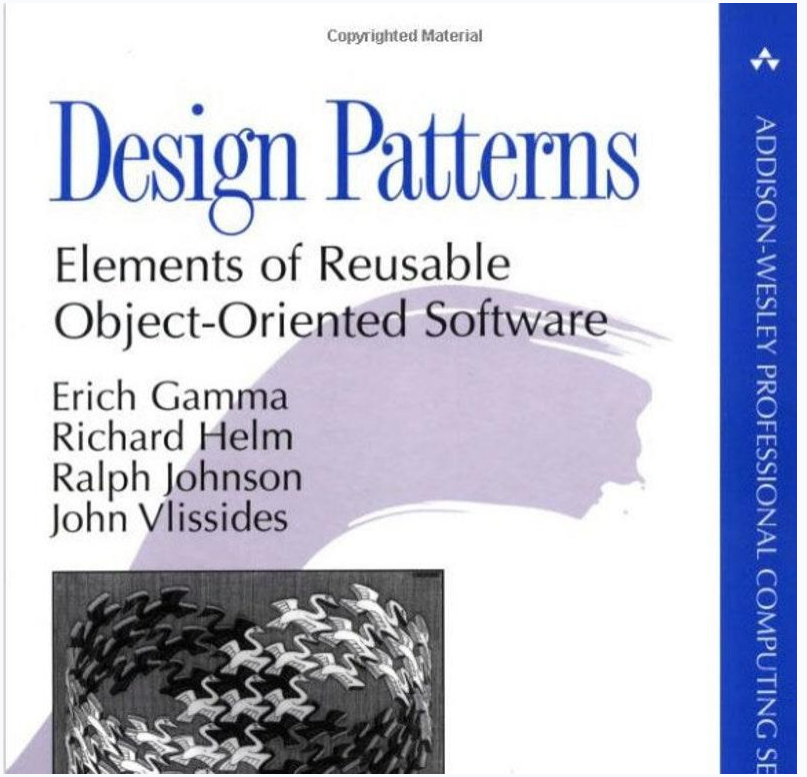


Creación de los Primeros Patrones de Diseño

- Los patrones de diseño, originados en la arquitectura y adaptados a la ingeniería de software en los años 80, ganaron relevancia en los 90 con el libro de la "Gang of Four", que introdujo 23 patrones esenciales.
- Estos patrones han influenciado el desarrollo y enseñanza del software, inspirando lenguajes y frameworks.
- Siguen siendo una herramienta valiosa en la actualidad para proporcionar soluciones comprobadas a problemas de diseño recurrentes y facilitar la comunicación entre desarrolladores.

Claves de la Historia de los Patrones de Diseño

1. **Influencia Arquitectónica**: El concepto de patrones de diseño se originó en la arquitectura y fue adaptado a la ingeniería de software, mostrando la interconexión entre diversas disciplinas.
2. **Contribución de la "Gang of Four"**: El libro "Design Patterns: Elements of Reusable Object-Oriented Software", escrito por la "Gang of Four", es un hito importante en la historia de los patrones de diseño, estableciendo 23 patrones
3. **Evolución Continua**: Los patrones de diseño han evolucionado y se han expandido a lo largo de la historia de la ingeniería de software.

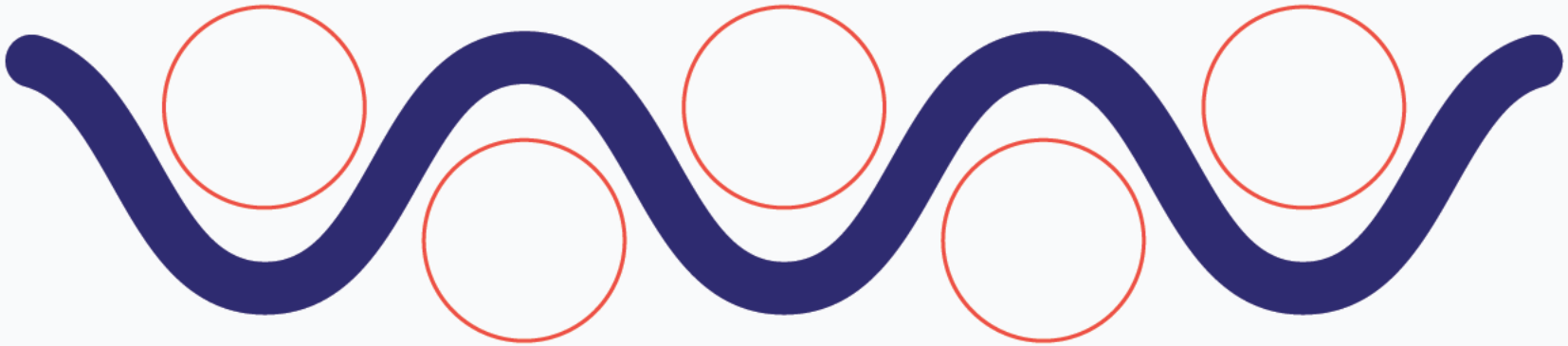


Design Patters por “Gang of Four”

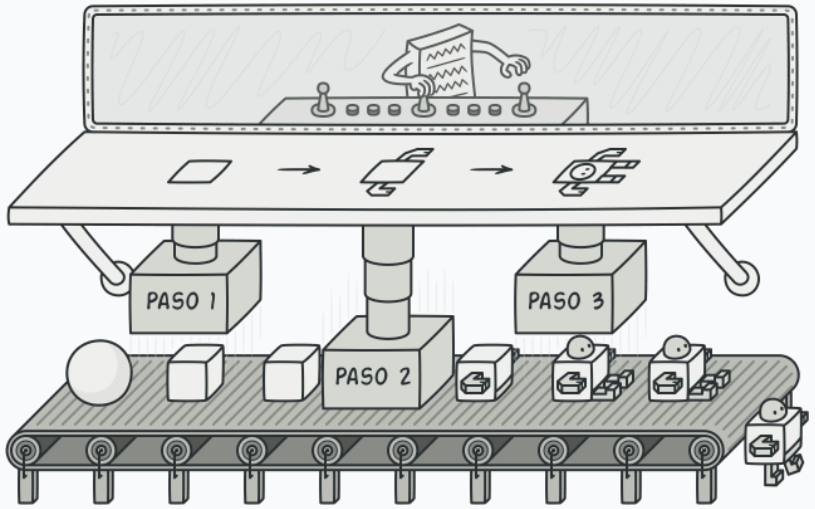
@batistajohel

@ayudinga

Tipos de Patrones de Diseño



Patrones de Diseño Creacionales



Ejemplo de Patrón de Diseño Builder

Patrones Creacionales: Los patrones de diseño creacionales se ocupan de los mecanismos de creación de objetos.

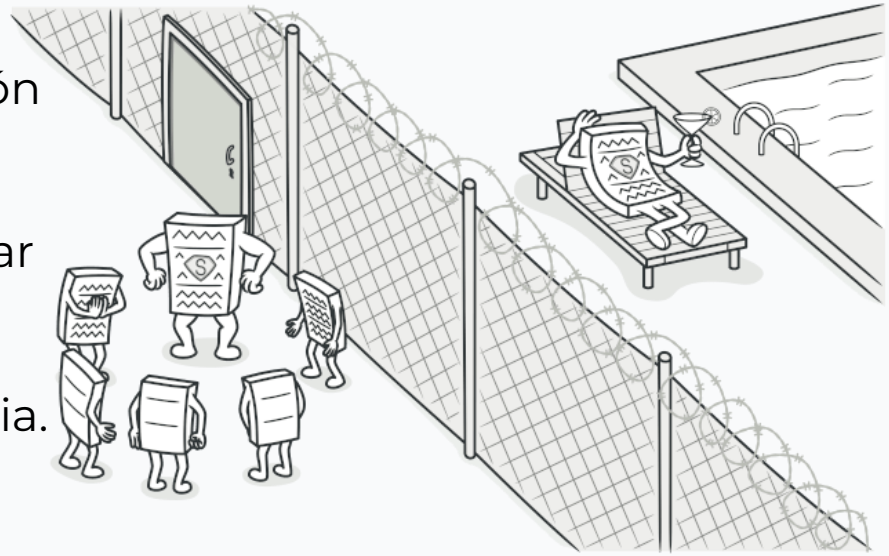
- Proporcionan maneras de crear objetos mientras ocultan la lógica de creación, en lugar de instanciar objetos directamente.
- Esto permite a los programadores hacer el sistema independiente de cómo sus objetos son creados, compuestos y representados.
  - Algunos ejemplos notables de patrones creacionales incluyen el Singleton, Factory Method, Abstract Factory, Prototype y Builder.

@batistajohel

Patrones de Diseño Estructurales

Los patrones de diseño estructurales abordan la composición de objetos y clases.

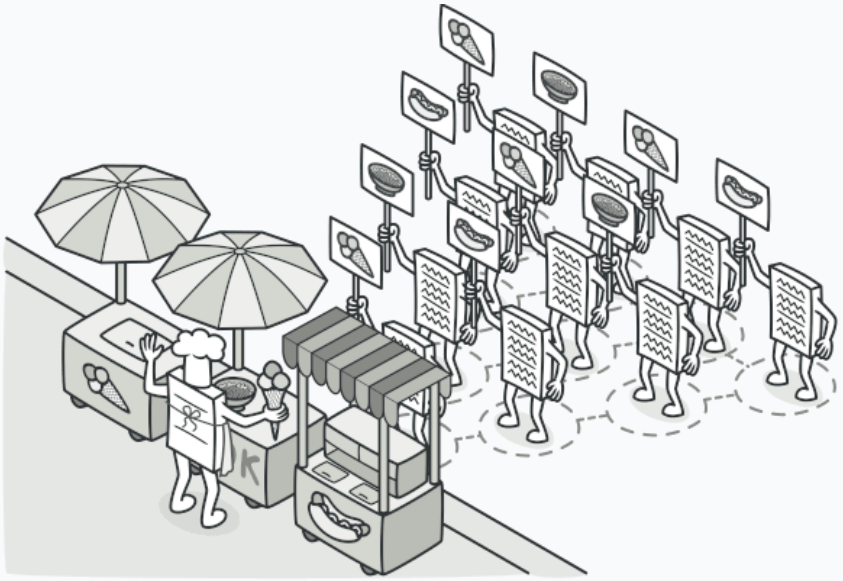
1. **Adapter** hace compatibles interfaces incompatibles.
2. **Bridge** separa la abstracción de la implementación permitiendo su variación independiente.
3. **Composite** trata los objetos individuales y las composiciones de manera similar para representar jerarquías.
4. **Decorator** agrega responsabilidades dinámicamente, siendo flexible frente a la herencia.
5. **Facade** ofrece una interfaz simplificada para subsistemas complejos.
6. **Flyweight** reduce el uso de memoria compartiendo datos.
7. **Proxy** controla el acceso a otro objeto proporcionando.



Patrón de Diseño Proxy

@batistajohel

Patrones de Diseño de Comportamiento



Ejemplo de Patrón de Diseño Visitor

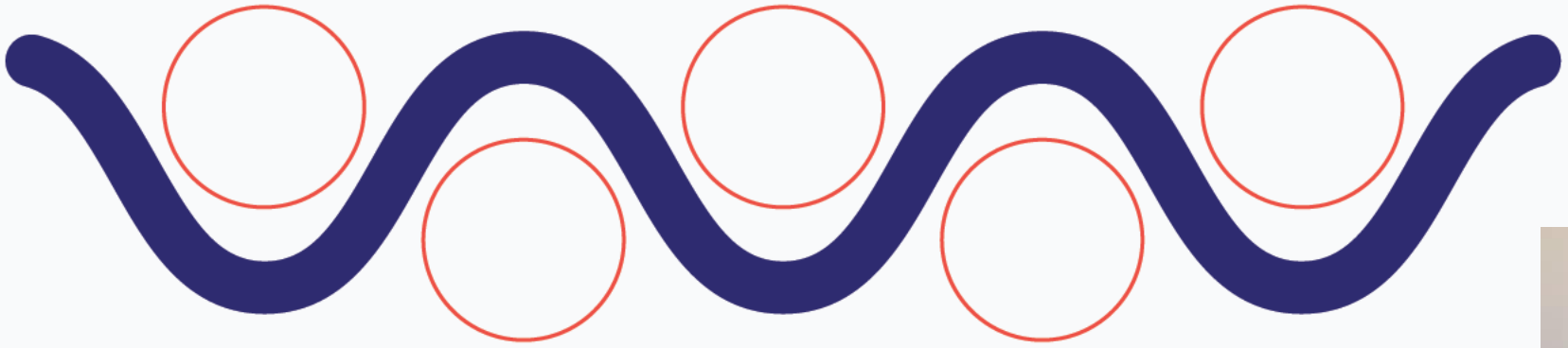
Los patrones de diseño de comportamiento manejan la comunicación y responsabilidades entre objetos.

- **Chain of Responsibility** pasa solicitudes entre manejadores en una cadena.
- **Command** convierte solicitudes en objetos independientes.
- Interpreter representa e interpreta una gramática dada como un lenguaje.
- **Iterator** permite recorrer objetos agregados sin revelar su representación.
- **Mediator** encapsula la interacción entre objetos para reducir dependencias.
- **Memento** guarda y restaura estados de objetos.
- **Observer** actualiza automáticamente objetos dependientes cuando un objeto cambia de estado.
- **State** altera el comportamiento del objeto según su estado.
- **Strategy** encapsula algoritmos intercambiables.
- **Template Method** define esqueletos de algoritmos.
- **Visitor** permite realizar operaciones en elementos sin cambiar sus clases.

@batistajohel

@ayudinga

Patrón de Diseño Prototype



Caso de Estudio
Aplicación de Dibujo Vectorial

En una aplicación de dibujo vectorial, los usuarios pueden crear formas complejas como casas, compuestas de subformas.

Para crear casas similares pero únicas, el patrón de diseño Prototype permite clonar una "casa prototipo" y personalizarla, en lugar de crear cada casa desde cero.

Esto ahorra tiempo y esfuerzo, promueve la eficiencia, y permite una gran personalización, demostrando la utilidad del patrón Prototype en la creación de objetos similares pero no idénticos.

VectorAPK: Aplicación de Dibujo Vectorial para Android

@batistajohel

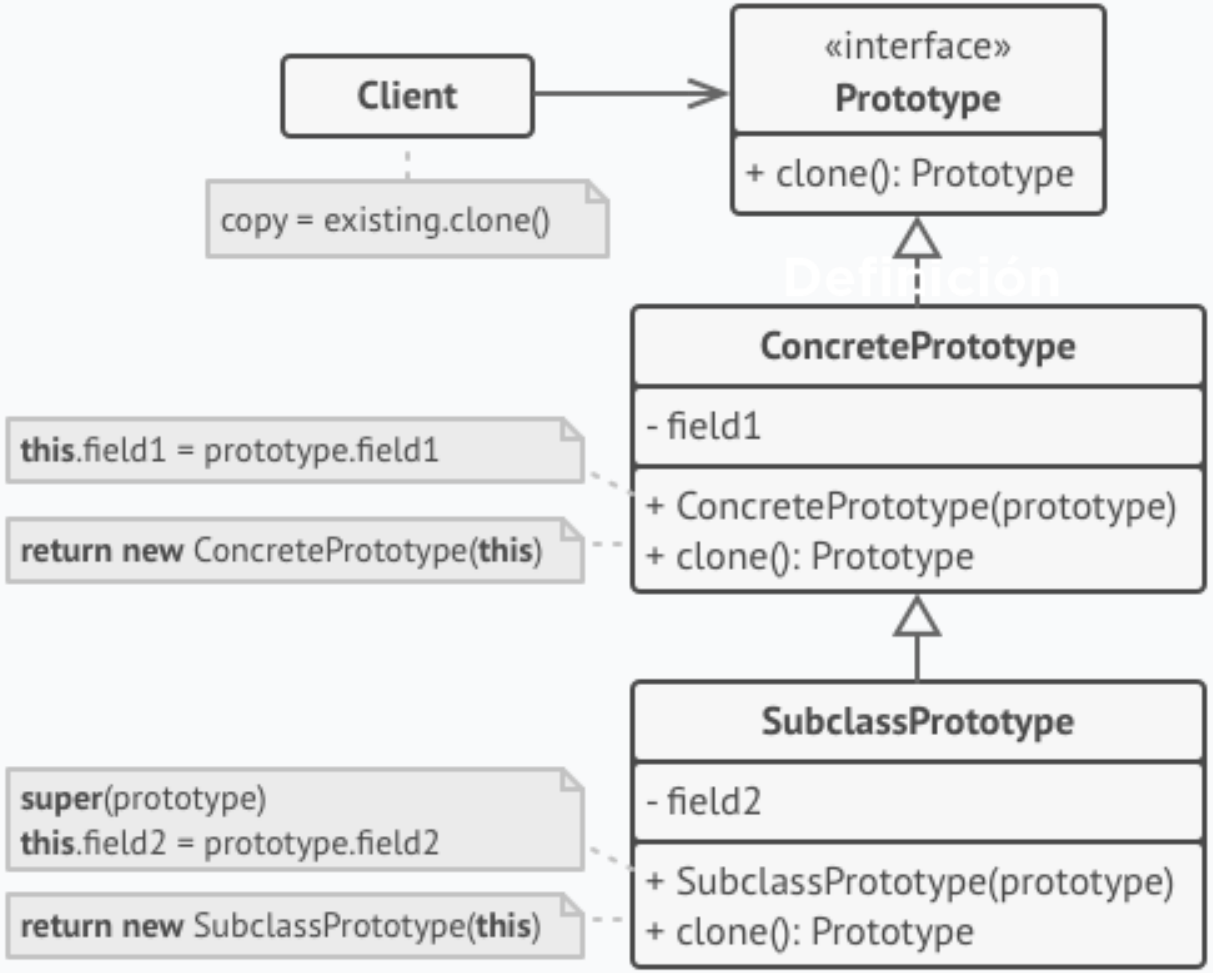


Diagrama de Clases
Patrón de Diseño Prototype

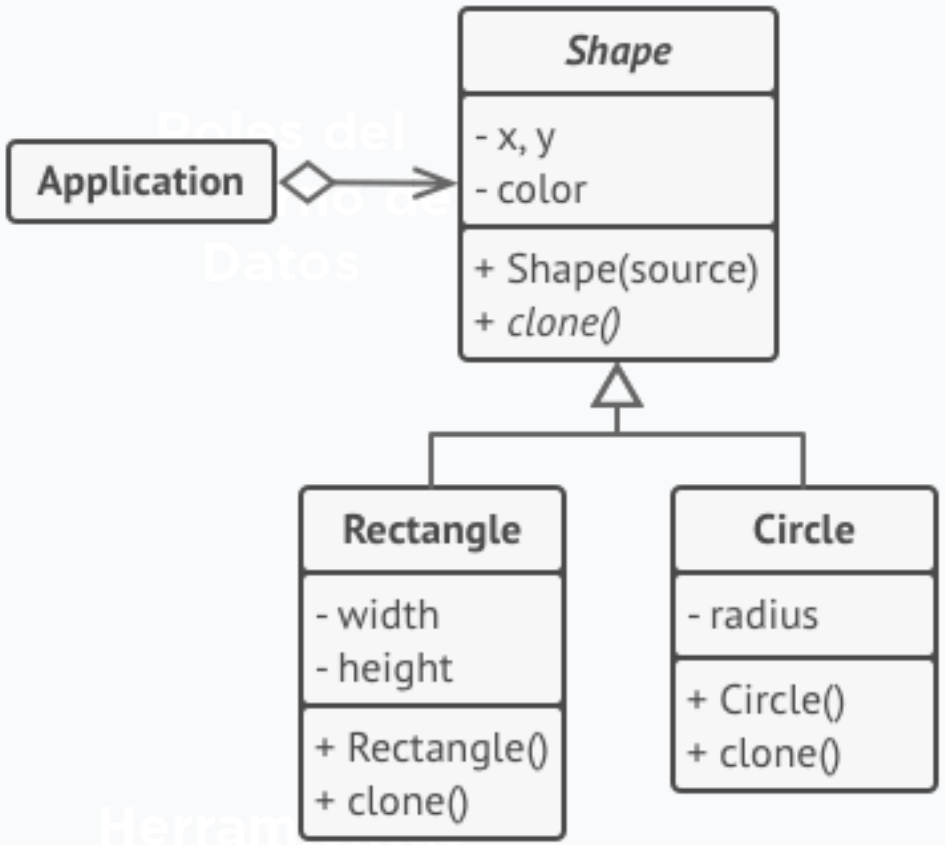


Diagrama de Clases
Objetos Geométricos

@batistajohel

Comentarios Finales

Este trabajo proporciona un análisis detallado de los Patrones de Diseño en Ingeniería de Software, con énfasis en el patrón Prototype.

El patrón Prototype facilita la creación eficiente de objetos clonando instancias existentes, resultando en ahorros significativos de tiempo y recursos.
Este patrón promueve la reutilización y flexibilidad del software al generar copias de objetos sin suponer su tipo, mejorando así la modularidad y la adaptabilidad del código.

Una implementación práctica en Java destacó sus ventajas y su impacto en la eficiencia del código.

Los Patrones de Diseño, especialmente Prototype, son herramientas valiosas para los ingenieros de software, mejorando la eficiencia, calidad y mantenibilidad del software.

@batistajohel

Referencias Bibliográficas

1. *Tutorialspoint. (s.f.). Design Pattern Tutorial. Recuperado de https://www.tutorialspoint.com/design_pattern/index.htm*
2. *SourceMaking. (s.f.). Design Patterns. Recuperado de https://sourcemaking.com/design_patterns*
3. *DZone. (s.f.). Refcardz: Design Patterns. Recuperado de https://dzone.com/refcardz/design-patterns*
4. *Refactoring Guru. (s.f.). Design Patterns. Recuperado de https://refactoring.guru/design-patterns*
5. *Gang of Four (GoF) Design Patterns. (s.f.). Recuperado de https://www.gofpatterns.com/*
6. *Software Design Patterns by Example. (s.f.). Recuperado de https://www.softwaretestinghelp.com/design-patterns-tutorial-1/*
7. *Design Patterns in Java. (s.f.). Recuperado de https://www.javatpoint.com/design-patterns-in-java*
8. *Journal of Object Technology (JOT). (s.f.). Recuperado de https://www.jot.fm/*
9. *Software Engineering Institute (SEI) - Design Patterns. (1994). Recuperado de https://www.sei.cmu.edu/reports/94tr007.pdf*
10. *IBM Developer - Design Patterns. (s.f.). Recuperado de https://developer.ibm.com/tutorials/j-patterns/*

¡Gracias por su Atención!

¿Alguna pregunta, duda, comentario o sugerencia?

@batistajohel

Johel.batista@utp.ac.pa

+507 6920-4843

Enlace de la Charla:
https://youtu.be/5I0DdMEeO9A