**Informe del Proyecto: *SOLID-Javadoc***

**Principio Open/Closed (OCP)**

**1. Introducción**

* Objetivo del informe.
* Importancia de los principios SOLID.
* Enfoque específico en el principio OCP.
* Breve resumen del contenido y estructura del trabajo.

**2. Concepto Principal del OCP** Definición del principio Open/Closed.

* Significado de “abierto para extensión, cerrado para modificación”.
* Origen (Bertrand Meyer y SOLID).
* Problemas comunes que busca evitar.
* Relevancia en el desarrollo de software moderno.

**3. Analogías del Mundo Real**

* Analogías prácticas (plugins, puertos USB, apps modulares).
* Tabla comparativa: analogía vs aplicación técnica.
* Breve reflexión sobre la utilidad de pensar en analogías.

**4. Ejemplos con UML**

* Diagrama de una mala implementación que viola OCP.
* Diagrama corregido que respeta OCP.
* Comparación visual y estructural entre ambas versiones.
* Comentarios sobre mantenibilidad y escalabilidad.

**5. Ejemplos con Código en Java**

* Ejemplo de código que **viola** OCP (explicado).
* Reescritura que **respeta** OCP utilizando abstracciones.
* Separación del main y cumplimiento del principio SRP.
* Comentarios detallados en el código.

**6. Opciones de Implementación**

* OCP en Java: interfaces, clases abstractas y composición.
* Patrones de diseño que aplican OCP:
  + Strategy
  + Template Method
  + Decorator
* Ejemplos breves o diagramas de cada patrón.
* Alternativas o enfoques complementarios.
* Cuándo **no** aplicar OCP.

**7. Usos en la Práctica**

* Ejemplos reales: Spring, Eclipse, patrones de plugins.
* Casos donde el OCP está integrado en frameworks.
* Enlaces a documentación (pueden ir en anexos o pie de página).
* Recomendaciones y buenas prácticas para su uso en proyectos reales.

**8. Ventajas y Desventajas**

**Ventajas:**

* Escalabilidad
* Mantenibilidad
* Flexibilidad

**Desventajas:**

* Complejidad innecesaria
* Sobreuso de abstracción

**Ejemplos:**

* Caso de éxito (agregar nuevas funcionalidades).
* Caso problemático (exceso de interfaces).

**9. Relación con Otros Principios SOLID**

* OCP + SRP: Una clase con responsabilidad única es más fácil de extender.
* OCP + LSP: Las extensiones deben respetar el contrato del padre.
* OCP + DIP: Las dependencias deben ser abstractas para facilitar la extensión.

**10. Conclusión**

* Reflexión grupal sobre lo aprendido.
* Aplicabilidad en futuros proyectos.
* Ventajas de aplicar correctamente OCP en conjunto con otros principios.

**12. Bibliografía**

* Citas de artículos, documentación, libros y fuentes utilizadas.
* Formato IEEE (ya recopilaste algunas; puedo ayudarte a darles forma si necesitas).

**13. Anexos *(opcional)***

* Código completo
* Diagramas UML en PlantUML
* Capturas de pruebas o ejemplos visuales