UNIVERSIDAD DE LIMA

INGENIERÍA DE SOFTWARE II

Laboratorio N°2: SOLID

Estimado alumno,

¡Felicidades! Ha sido preseleccionado como Ingenieros de Software en la empresa “Tech Startup”. El gerente le está proponiendo un desafío interesante: ser contratado como mid-senior, pero, para lograrlo, deberá demostrar su capacidad para aplicar los principios SOLID y mejorar un proyecto de la empresa.

La startup ha sido diseñado sin considerar las buenas prácticas, lo que lo ha vuelto difícil de mantener y poco escalable. Su tarea es refactorizar el código que se les ha adjuntado, aplicando cada uno de los principios SOLID, de modo que el software sea más modular, escalable y fácil de mantener.

Los principios **SOLID** son un conjunto de buenas prácticas que ayudan a escribir código más robusto y fácil de modificar sin romper el comportamiento de la aplicación.

1. **SRP** (Single Responsibility Principle - Principio de Responsabilidad Única): Cada clase debe tener una sola responsabilidad o razón de cambio. Ayuda a mantener el código limpio y fácil de modificar.

2. **OCP** (Open/Closed Principle - Principio de Abierto/Cerrado): Las clases deben estar abiertas a la extensión, pero cerradas a la modificación. Facilita agregar nuevas funcionalidades sin tocar el código existente.

3. **LSP** (Liskov Substitution Principle - Principio de Sustitución de Liskov): Las subclases deben poder reemplazar a sus clases base sin alterar el comportamiento del programa. Evita comportamientos inesperados cuando se usan herencias.

4. **ISP** (Interface Segregation Principle - Principio de Segregación de Interfaces): Los clientes no deberían estar forzados a depender de interfaces grandes. Es mejor tener interfaces pequeñas y específicas para evitar que una clase implemente métodos que no necesita.

5. **DIP** (Dependency Inversion Principle - Principio de Inversión de Dependencias): Las clases de alto nivel no deben depender de clases de bajo nivel. Ambas deben depender de abstracciones (interfaces o clases abstractas). Esto desacopla el código y lo hace más flexible.

El gerente te ha adjuntado 5 partes del código que necesitan mejorar. Debes aplicar un principio SOLID en cada uno de ellos:

1. SRP (Principio de Responsabilidad Única):
- Archivos: Informe.java y Main.java
- Mejora: Refactoriza el código para que cada clase tenga una única responsabilidad.

2. OCP (Principio de Abierto/Cerrado):
- Archivos: Empleado.java y Main.java
- Mejora: Asegúrate de que las clases estén abiertas a la extensión pero cerradas a la modificación.

3. LSP (Principio de Sustitución de Liskov):
- Archivos: Pato.java, PatoDeGoma.java y Main.java
- Mejora: Las subclases deben poder reemplazar a sus clases base sin romper el comportamiento.

4. ISP (Principio de Segregación de Interfaces):
- Archivos: Trabajador.java, Gerente.java, Limpiador.java y Main.java
- Mejora: Divide las interfaces grandes en interfaces más pequeñas y específicas.

5. DIP (Principio de Inversión de Dependencias):
- Archivos: ProveedorConcreto.java, Tienda.java y Main.java
- Mejora: Desacopla el código usando abstracciones, haciendo que las clases de alto nivel dependan de interfaces en lugar de implementaciones concretas.

Objetivo: Refactorizar cada parte del proyecto de manera que se respeten los principios SOLID. Recuerden que esto hará que el código sea más flexible, fácil de mantener y escalable a futuro. Al completar este reto, demostrarán que están listos para asumir roles de mayor responsabilidad en su carrera profesional.

¡Buena suerte! Si tienen preguntas o necesitan alguna aclaración, no duden en consultar.