**UNIVERSIDAD TECNOLÓGICA DEL PERÚ**  
**Facultad de Ingeniería**  
**Escuela Profesional de Ingeniería de Sistemas e Informática**

**PRÁCTICA CALIFICADA N° 1**

Curso : Diseño de Patrones  
Docente : Roy Saavedra Jiménez  
Alumno(s) : Huamani Champe Alexander  
Código(s) : U23320859

**Pregunta 1: (12 puntos)**

**Caso: Sistema de notificaciones**

Una empresa desea implementar un sistema de **notificaciones** para comunicarse con sus clientes. El sistema debe permitir enviar mensajes por diferentes **canales** (Email, SMS, Push). En el futuro, la empresa quiere agregar nuevos canales (por ejemplo, WhatsApp) sin tener que modificar el código existente.

Asimismo, las notificaciones pueden ser de diferentes tipos de **mensajes**:

* Alerta (urgente).
* Recordatorio (normal).
* Promoción (con descuentos).

El sistema debe cumplir con buenas prácticas de diseño orientado a objetos, en particular los **principios SOLID**, de modo que sea modular, extensible y fácil de mantener.

**Requisitos del sistema**

**1. Gestión de mensajes**

* Cada mensaje debe almacenar: contenido y tipo (Alerta, Recordatorio o Promoción).
* Los mensajes deben poder extenderse sin modificar el código existente.

**2. Canales de notificación**

* Todos los mensajes pueden enviarse por diferentes canales: Email, SMS y Push.
* El sistema debe permitir agregar nuevos canales de manera flexible (ejemplo: WhatsApp).

**3. Envío de notificaciones**

* El sistema debe tener un **Gestor de Notificaciones** que reciba un canal de notificación y un mensaje, y procese el envío.
* El gestor no debe depender de clases concretas, sino de **abstracciones**.

**4. Principios de diseño (SOLID)**

* **SRP (Responsabilidad Única):** Cada clase debe cumplir un único propósito (ejemplo: Mensaje solo almacena datos, CanalNotificacion solo envía).
* **OCP (Abierto/Cerrado):** El sistema debe permitir agregar nuevos canales o tipos de mensajes sin modificar las clases existentes.
* **LSP (Sustitución de Liskov):** Subclases como Promocion o Alerta deben poder usarse en lugar de la clase abstracta Mensaje sin alterar la lógica.
* **ISP (Segregación de Interfaces):** Las interfaces deben ser pequeñas y específicas, como CanalNotificacion.
* **DIP (Inversión de Dependencias):** El GestorNotificaciones debe depender de la abstracción CanalNotificacion, no de implementaciones concretas.

**Objetivo**

Implementar un sistema de notificaciones modular y extensible que:

* Permita agregar fácilmente nuevos canales de comunicación y tipos de mensajes.
* Implemente los principios SOLID para mantener un diseño limpio y flexible.
* Genere un flujo de envío claro y reutilizable.

**Pregunta 2: (6 puntos)**

Completar la tabla siguiente en función de su proyecto final, colocar las clases, clases derivadas, interfaces, implementación y otros identificados en su proyecto final. Explique y sustente para un mejor entendimiento de la aplicación de los principios SOLID.

|  |  |
| --- | --- |
| **SOLID** | **PROYECTO FINAL** |
| **SRP (Responsabilidad Única)** | AnalisisClinico: Clase que representar los datos del análisis clínico  EnvioInforme: Clase que se encarga de solo enviar el informe al dueño (Solo tiene una responsabilidad)  MuestraServicio: Clase que solo se encarga de tomar muestra para el análisis. (Solo tiene una responsabilidad)  RegistroResultado: Clase que solo se encarga de registrar el resultado del análisis. (Solo tiene una responsabilidad)  Main: Clase que se encarga de la ejecución del codigo |
| **OCP (Abierto/Cerrado):** | Examen: Interfaz que define el contrato para cualquier tipo de examen  ExamenOrina, ExamenSangre: Clases que implementan el comportamiento específico  ProcesadorExamenes: Clase que procesa examenes sin depender del tipo concreto  Main: Clase que se encarga de la ejecución del codigo |
| **LSP (Sustitución de Liskov)** | ProfesionalVeterinario: Clase que define la actividad comun (registrar actividad)  Veterinario: Clase que registra diagnosticos  TecnicoVeterinario: Clase que registra procedimientos técnicos  Main: Clase que se encarga de la ejecución del codigo |
| **ISP (Segregación de Interfaces)** | IRecepcionista: Interfaz de funciones de recepcionista  ITecnicoLaboratorio: Interfaz de procesos de laboratorio  IVeterinario: Interfaz de atención medica a mascotas  Recepcionista: Clase que implementa solo lo que necesita su rol  Tecnico: Clase que implementa solo funciones de laboratorio  Veterinario: Clase que implementa solo funciones de veterinario  Main: Clase que se encarga de la ejecución del codigo |
| **DIP (Inversión de Dependencias)** | EmpleadoVeterinaria: Interfaz de abstracción que define tareas comunes  Veterinario, TecnicoVeterinario: Clases que implementan tareas específicas según el rol  EmpresaVeterinaria: Clase que usa la abstracción para ejecutar tareas  Main: Clase que se encarga de la ejecución del codigo |

**Pregunta 3: (2 puntos)**

* Los grupos que subieron su entregable de avances de proyecto final dentro del plazo establecido obtienen 2 puntos.
* Los grupos que no realizaron la entrega obtienen 0 puntos