PRÁCTICA DE LABORATORIO / TALLER Nro. 3

Carrera Computación

A. DATOS INFORMATIVOS		
Asignatura: Software Engineering Models	**Ciclo / Semestre:** Séptimo	**Paralelo:** A
Docente: Roberth Gustavo Figueroa Díaz	**Período Académico:** Octubre – Marzo 2024	
Alumno: Melissa Maribel Tuza Jiménez.		

B. INFORMACIÓN GENERAL	
Unidad: 3. Patrones de ingeniería de software (análisis, diseño y de arquitectura).	
Tema: Patrones de análisis, diseño y de arquitectura	
Fecha: Loja, 2023	**Nro. horas:** 4

Objetivos:

- Resolver problemas de análisis de software a través de la aplicación de patrones de análisis.
- Identificar y modelar a través de UML los patrones de análisis de acuerdo con el contexto del problema.

Corresponde al resultado de aprendizaje:

Desarrolla software usando patrones, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad

Recursos y/o materiales:

- Computador: Computadora con acceso a internet.
- Navegador: Revisión de información por parte del estudiante.
- Documentación Universitaria: Documentación necesaria para validar estado del estudiante.
- Lenguaje de programación
- Herramientas de modelado para UML
- IDE con plugin para UML
- Material bibliográfico o recurso indicado en el EVA.

C. DESARROLLO

Instrucciones:

1. Revisar los diversos contenidos y modelos revisados en clases, use el libro base e información relacionada.
2. Instalar la herramienta para modelado de UML deseada.
3. En base al ejercicio planteado de *una Institución de Educación Superior* descrito en la siguiente sección, elaborar:

   a. Analizar el ejercicio planteado en la actividad.
   b. Obtener el modelo PIM de acuerdo con los patrones encontrados y aplicados

4. Desarrollar el informe demostrando el proceso de desarrollado en base al punto anterior

5. Responda a las preguntas planteadas en la sección de preguntas a responder.
6. Establecer sus respectivas conclusiones, tomar en cuenta los objetivos planteados.

Trabajar de forma Individual.

1. Realizar la revisión de información relacionada al tema.
2. Investigue cual patrón de diseño se puede aplicar al ejercicio planteado.
3. Analice cual o cuales patrones de diseño se puede aplicar al ejercicio planteado:
   Ejercicio(s):

   Una Institución de Educación Superior (IES) necesita un software que permita almacenar información relacionada a su esquema de organización académica para lo cual se indica alguna de ellas (Como es un dominio que ustedes los conocen pueden adjuntar los que ustedes crean necesario)

   - **Reglas de negocio**
     - La Institución es el ente principal y contiene entre ellos el nombre, código a nivel de Unidades de Educación Superior, autoridad (Rector y Vicerrector), dirección, teléfono entre otros.
     - La institución está conformada por facultades las cuales también posee un nombre, código interno, autoridad (Decano), dirección, teléfono entre otros.
     - Las facultades tienen un conjunto de carreras las cuales también posee un nombre, código interno de la facultad, autoridad (Director o Géstor), dirección, teléfono entre otros.
     - Cada unidad académica está conformada por Estudiantes, Docentes (Nombramiento y Temporal) y personal administrativo (Nombramiento y Temporal); para cada uno de ellos se necesita conocer sus nombres, apellidos, fecha de nacimiento, título y sus formas de contactarlos que puede ser a través de su correo electrónico, teléfono fijo, teléfono móvil, dirección física y dirección web (sitio web si posee).
     - Para los Docentes y personal administrativo se requiere tener un control del tipo de contrato (Ocasional Tiempo Completo, Ocasional Medio Tiempo), su tiempo de contrato (desde hasta cuándo), junto con sus responsabilidades o actividades para las cuales fueron contratados.

4. Realice su Informe indicando la identificación de cuatro patrones aplicados a través de los siguientes aspectos:
   - Nombre de patrón:
   - Problema que resuelve:
   - Solución planteada:
   - Estructura general del patrón (Diagrama UML)
   - Estructura específica del patrón (Diagrama UML de acuerdo con el contexto del problema planteado)
5. Realizar el Modelo Final o limpio aplicando todos los patrones (único diagrama UML completo).

DESARROLLO

- **Nombre del patrón:** Patrón Composite.
- **Problema que resuelve:** Maneja las jeraraquías entre institución, facultades, carrera, en donde cada uno de estos obejetos se manejan de forma individual y contienen diferentes niveles.

- **Solución planteada:** Plantear una estructura de árbol en donde desde un tronco, se desprende cada rama correspondiente a objetos a los objetos institución, facultades, carrera, de esta forma se disminuye la complejidad al momento de gestionar las entidades.
- **Estructura general del patrón:** Composite

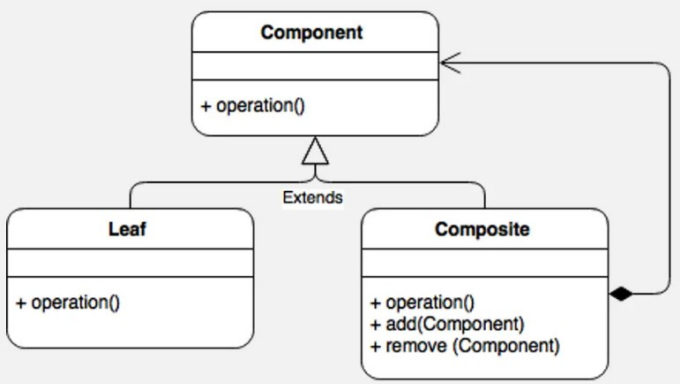

Fig 1. Diagrama general el patrón Composite

En el diagrama, el patrón Composite trabaja con los objetos de forma individual y uniformemente, es decir, la interacción se realiza entre el cliente y un objeto simple o compuesto de la misma forma, de esta forma, simplificando el código es simplificado ya que no es necesario especificar una clase concreta de los objetos con los que va a trabajar.

- Estructura aplicada con el patrón:

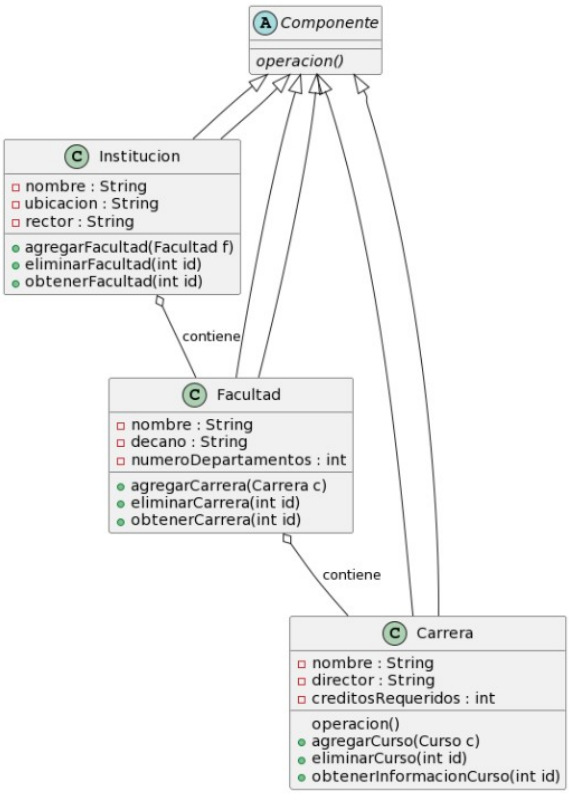

Fig 2. Diagrama aplicado el patrón Composite

En el diagrama, se trabaja con una clase abstracta llamada “Componente” la cual maneja una operación común, es implementada por las clases institución, facultad y carrera. En donde, una institución puede contener una carrera y una facultad, finalmente facultad está compuesta por una clase carrera.

- **Nombre de patrón:** Strategy
- **Problema que resuelve: P**ermite definir una familia de algoritmos, encapsular cada uno de ellos y hacerlos intercambiables. Este patrón es útil para gestionar diferentes tipos de contratos y políticas de contratación de personal y docentes.
- **Solución planteada:** Implementar diferentes estrategias de contratación (por ejemplo, para nombramiento, ocasional tiempo completo, ocasional medio tiempo) que pueden ser aplicadas dinámicamente a los docentes y al personal administrativo según sea necesario.
- **Estructura general del patrón (Diagrama UML)**

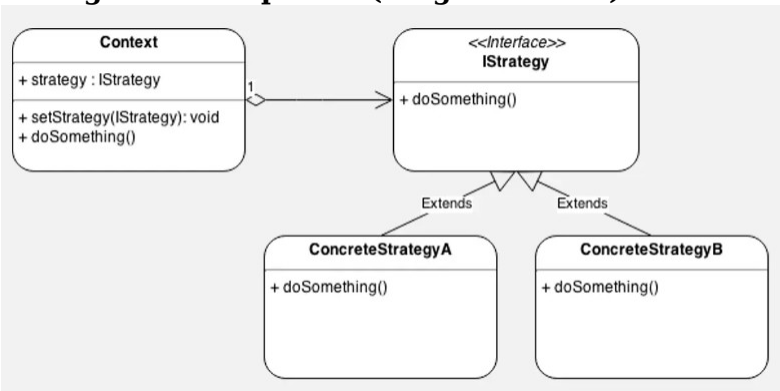

Fig 3. Diagrama general el patrón Strategy

En el diagrama, muestra el patrón de diseño Strategy. En este patrón, 'Context' mantiene una referencia a una 'IStrategy', permitiendo cambiar su comportamiento dinámicamente al establecer diferentes subclases de 'IStrategy' ('ConcreteStrategyA' o 'ConcreteStrategyB'). 'Context' delega la ejecución de una operación, a través del método 'doSomething()', a la estrategia actual, permitiendo variar el comportamiento del 'Context' en tiempo de ejecución sin modificar su código.

- **Estructura específica del patrón (Diagrama UML de acuerdo con el contexto del problema planteado)**

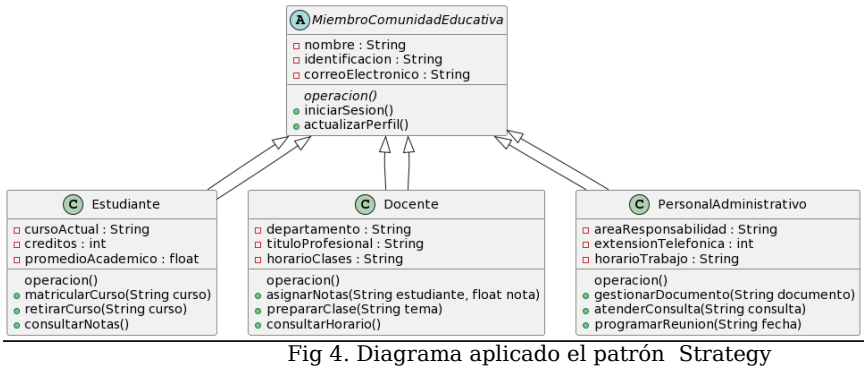

Fig 4. Diagrama aplicado el patrón Strategy

En el diagrama, la clase abstracta MiembroComunidadEducativa actúa como superclase con atributos comunes como nombre, identificacion y correoElectronico, y operaciones como operacion(), iniciarSesion() y actualizarPerfil(). Las clases Estudiante, Docente y PersonalAdministrativo heredan de esta superclase y agregan sus atributos y operaciones específicos, como matricularCurso() para

estudiantes, asignarNotas() para docentes y gestionarDocumento() para personal administrativo, permitiendo así un manejo diferenciado y especializado según el tipo de usuario.

- **Nombre de patrón:** Observer
- **Problema que resuelve:** Este patrón establece una relación uno-a-muchos entre objetos, donde un objeto (conocido como el "sujeto" o "publicador") mantiene una lista de sus dependientes (los "observadores" o "suscriptores") y los notifica automáticamente de cualquier cambio de estado. Por ejemplo, dentro del sistema de gestión educativa donde el horario de clases es susceptible a cambios debido a imprevistos como ausencias de docentes o problemas logísticos.
- **Solución planteada:** Cada tipo de miembro (Estudiante, Docente, PersonalAdministrativo) es una Componente en el patrón y puede contener operaciones y atributos específicos, pero también comparte operaciones y atributos comunes definidos en la clase abstracta MiembroComunidadEducativa. Esto facilita la gestión de miembros de la comunidad educativa de manera uniforme a través de la interfaz común.
- **Estructura general del patrón (Diagrama UML)**

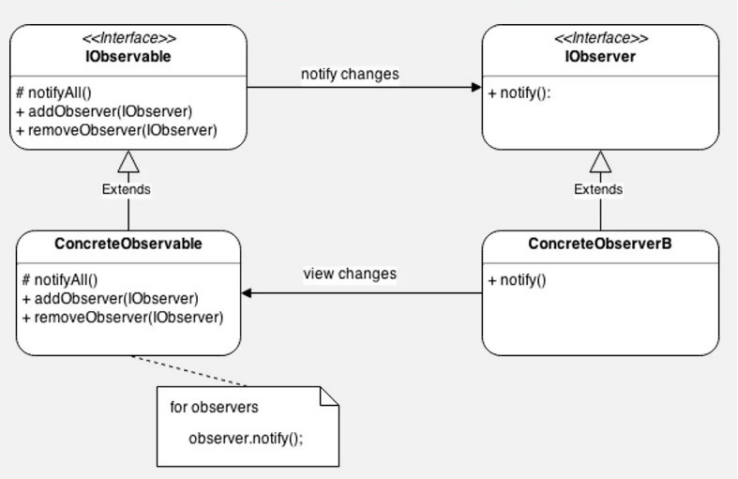

Fig 5. Diagrama general el patrón Observer

En el diagrama, opera con el patrón Observer, donde IObservable es la interfaz para el sujeto que notifica los cambios, y ConcreteObservable es una implementación concreta que notifica a todos sus observadores suscritos. IObserver es la interfaz para los observadores que deben ser notificados, y ConcreteObserverB es una implementación concreta de un observador que reacciona a los cambios. Cuando ConcreteObservable tiene un cambio de estado, utiliza notifyAll() para llamar al método notify() en cada ConcreteObserverB, el cual realiza las acciones apropiadas en respuesta a la notificación

- **Estructura específica del patrón (Diagrama UML de acuerdo con el contexto del problema planteado)**

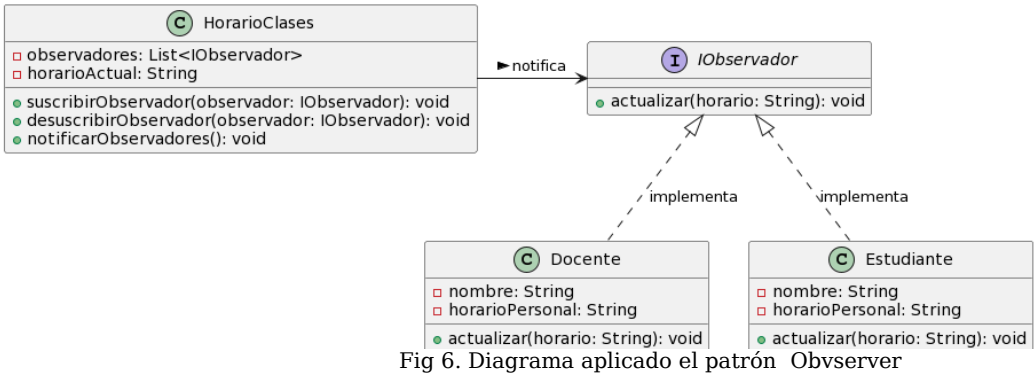

Fig 6. Diagrama aplicado el patrón Obvserver

En el diagrama, muestra HorarioClases actúa como sujeto que mantiene una lista de observadores y notifica cambios de horario. La interfaz IObservador define el método actualizar, que es implementado por las clases concretas Docente y Estudiante, permitiendo la actualización de sus horarios personales cuando se notifica un cambio en el horario de clases.

- **Nombre de patrón:** Factory Method
- **Problema que resuelve:** Define una interfaz para crear un objeto, pero deja que las subclases decidan qué clase instanciar. Permite flexibilidad en la creación de diferentes tipos de entidades (Estudiantes, Docentes, Personal Administrativo) sin especificar sus clases concretas.
- **Solución planteada:** Implementar una clase creadora que defina métodos para la creación de objetos de Estudiantes, Docentes y Personal Administrativo, permitiendo que las subclases personalicen la creación de estas entidades.
- **Estructura general del patrón (Diagrama UML)**

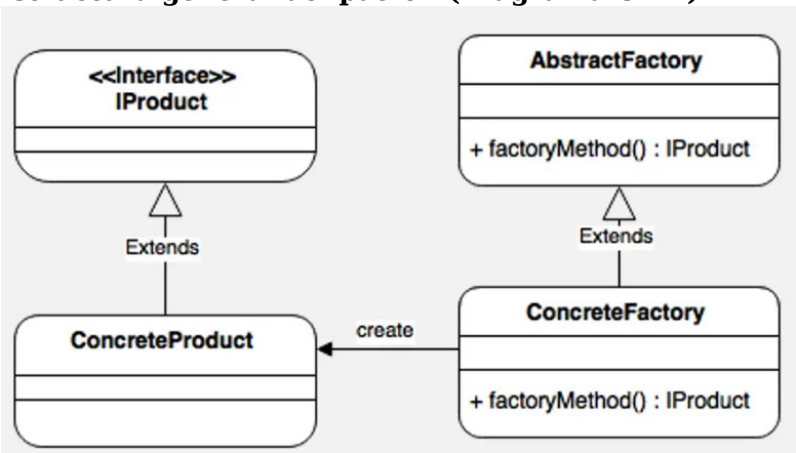

Fig 7. Diagrama general el patrón Factory Method

El diagrama UML representa el patrón Factory Method. La clase AbstractFactory tiene un método factoryMethod que devuelve una referencia a un objeto de tipo IProduct, una interfaz para los productos. La ConcreteFactory hereda de AbstractFactory y sobrescribe factoryMethod para crear y retornar una instancia de ConcreteProduct, que es una implementación específica de IProduct. Este patrón encapsula la creación de objetos y permite que las subclases determinen el tipo de objeto a crear.

- **Estructura específica del patrón (Diagrama UML de acuerdo con el contexto del problema planteado)**

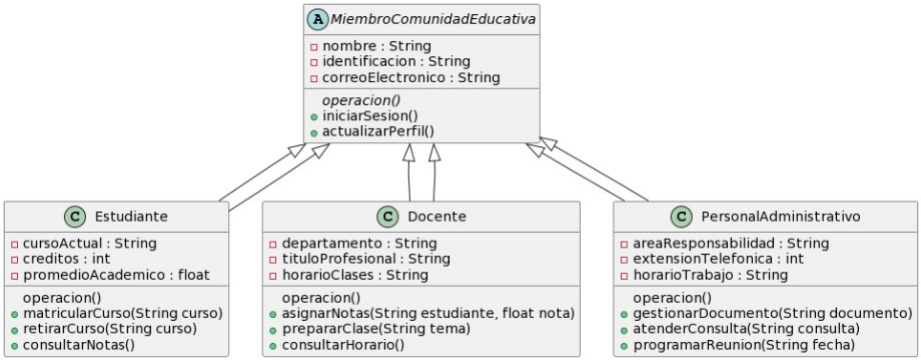

Fig 8. Diagrama aplicado el patrón Factory Method

En el diagrama, la clase abstracta 'MiembroComunidadEducativa' define operaciones comunes como 'operacion()', 'iniciarSesion()' y 'actualizarPerfil()'. Las clases 'Estudiante', 'Docente' y 'PersonalAdministrativo' heredan de ella, cada una con atributos y operaciones específicas que reflejan sus roles únicos dentro de la comunidad, como 'matricularCurso()' para 'Estudiante', 'prepararClase()' para 'Docente', y 'gestionarDocumento()' para 'PersonalAdministrativo'.

Preguntas a responder:

Examine el procedimiento y responda las siguientes preguntas:

1. ¿Indicar cuál patrón de diseño sería el más adecuado aplicar al ejercicio planteado?
   Para el ejercicio planteado, el patrón Composite sería el más adecuado. Este patrón permite componer objetos en estructuras de árbol para representar jerarquías parte-todo. En este contexto, el patrón facilita la representación de la estructura jerárquica de una Institución de Educación Superior (IES), desde la institución misma hasta sus facultades y carreras, y permite tratar de manera uniforme a todos los miembros (estudiantes, docentes, personal administrativo) que pueden tener diferentes atributos y métodos.
2. ¿Cuál patrón no se aplicaría al ejercicio planteado, indique la razón?
   El patrón Singleton no sería adecuado para este ejercicio, ya que su propósito es garantizar que una clase tenga una única instancia en toda la aplicación. Dado que la IES necesita gestionar múltiples instancias de facultades, carreras, estudiantes, docentes y personal administrativo, el patrón Singleton restringiría este requisito al limitar la instancia a una sola entidad, lo cual no es deseable en un sistema que requiere manejar numerosos objetos con diferentes identidades y atributos.

Conclusiones:

- La aplicación del patrón Composite al sistema de gestión de una Institución de Educación Superior demostró ser especialmente valiosa para modelar su jerarquía organizativa compleja. Este patrón no solo simplificó la representación de las relaciones entre las distintas unidades académicas, sino que también proporcionó un mecanismo coherente y escalable para tratar con operaciones que afectan a múltiples niveles de la jerarquía, desde la institución hasta el nivel individual de estudiantes y personal.
- El análisis de los patrones de diseño reveló la importancia de elegir el patrón adecuado para las necesidades específicas del sistema. Mientras que el patrón Composite se alineó bien con la estructura jerárquica de la IES, el patrón Singleton fue identificado como inapropiado debido a su naturaleza restrictiva,

lo que resalta la necesidad de una consideración cuidadosa de los patrones en relación con los objetivos del sistema y su dominio de aplicación. • El ejercicio subrayó la importancia de la flexibilidad y la mantenibilidad en el diseño de sistemas de software. Utilizando patrones de diseño como el Factory Method y el Observer, el sistema propuesto para la IES no solo puede adaptarse a cambios futuros en la estructura organizativa o en los requisitos funcionales, sino que también facilita la expansión y la integración con otros sistemas, asegurando que la solución sea a largo plazo sostenible y robusta.

D. RÚBRICA DE EVALUACIÓN	
Informe de trabajo: • Contenido: pertinente y concreto; • Estructura y organización: Elementos vinculados y estructurados coherentemente. • Originalidad y creatividad: trabajo inédito, presentación de nuevas ideas.	1 puntos
Resolución de Preguntas: • Modelo Independiente de la Plataforma (PIM) para cada patrón: 2 puntos o Modelo de clases para cada patrón con los aspectos solicitados. • Modelo Final o limpio aplicando todos los patrones: 4 puntos • Redacción de respuestas a sección de preguntas a responder: 2 puntos	8 puntos
Conclusiones: • Originalidad y creatividad: conclusiones inéditas en base a su experiencia y objetivos planteados.	1 puntos
Total **(Ponderado en Aprendizaje Práctico Experimental)**	10 puntos

E. FIRMAS DE RESPONSABILIDAD:	
Estudiante(s):	**Firma**
Melissa Tuza.