Patrón de diseño DAO y Factory

López Velázquez Mauricio Damman
López Vidal Luis Armando

¿Qué es un patrón de diseño?

Los patrones de diseño son soluciones habituales a problemas que ocurren con frecuencia en el diseño de software

Componentes:

- Propósito
- Motivación
- Estructura

Clasificación

Patrones creacionales

- Creación de objetos
- Flexibilidad
- Reutilización de código

Patrones estructurales

- Objetos y clases
- Estructura
- Flexibilidad y eficiencia

Patrones de comportamiento

- Comunicación
- Responsabilidad entre objetos

Patrón de Diseño DAO

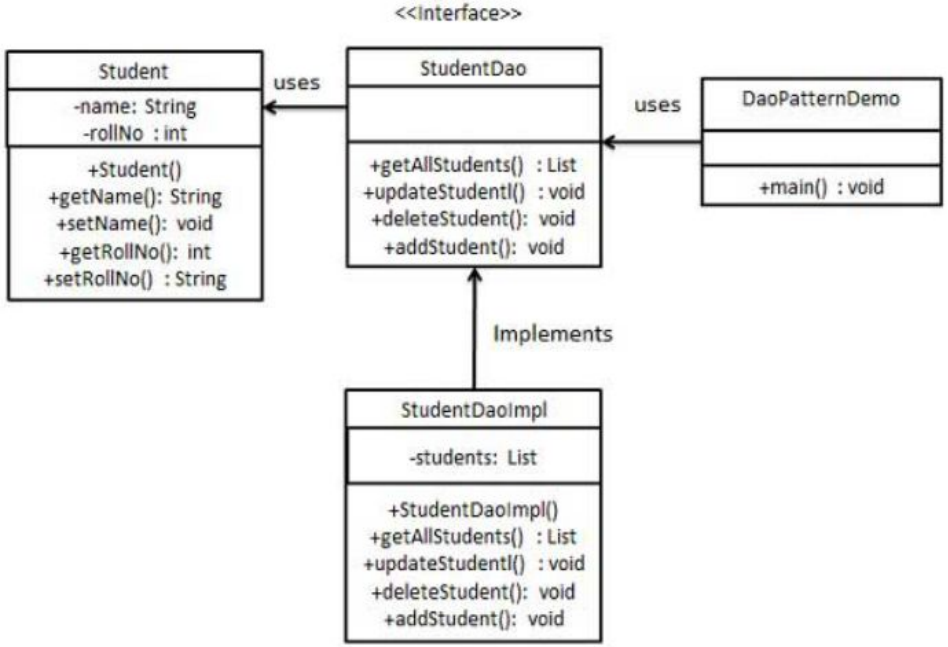
Problemática

- La implementación y formato de la información puede variar según la fuente de los datos
- Implementar la lógica de acceso a datos en la capa de lógica de negocio puede ser un gran problema, pues tendríamos que lidiar con la lógica de negocio en sí, más la implementación para acceder a los datos.

Solución

- El patrón Arquitectónico Data Access Object (DAO), permite separar la lógica de acceso a datos de los Objetos de negocios (Bussines Objects), de tal forma que el DAO encapsula toda la lógica de acceso de datos al resto de la aplicación.
- Es común utilizarlo para acceder a operaciones de bases de datos
- Se puede abstraer la lógica de acceso de un conjunto de operaciones

Implementación



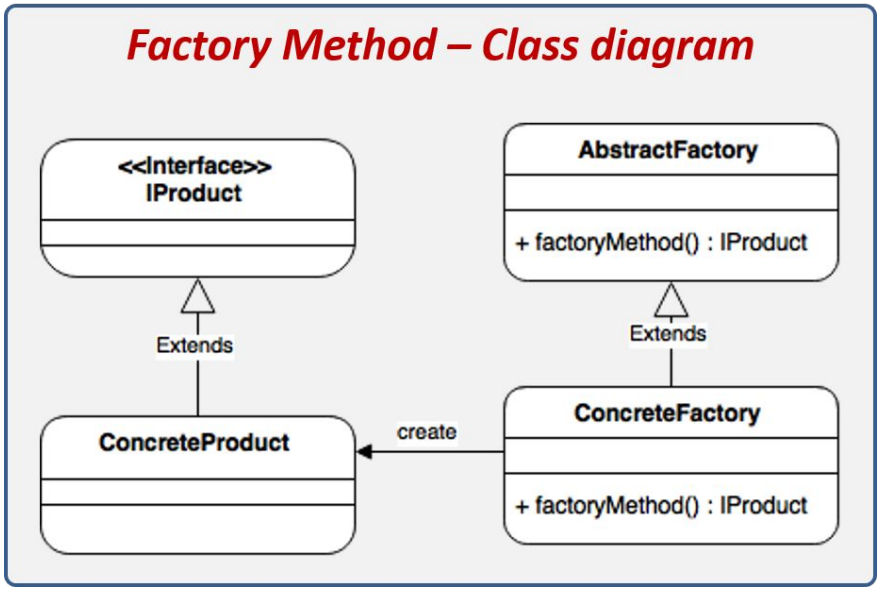
Patrón de Diseño Factory

El patrón de diseño Factory Method nos permite la creación de un subtipo determinado por medio de una clase de Fábrica, sin tener que especificar su clase exacta, esto resulta en que el objeto creado bajo determinada situacion puede ser intercambiado con flexibilidad..

- El objeto real creados es enmascarado detrás de una interfaz común entre todos los objetos que pueden ser creado, con la finalidad de que estos pueden variar sin afectar la forma en que el cliente interactúa con ellos.
- Es normal que un Factory pueda crear varios subtipos de una determinada interfaz y que todos los objetos concretos fabricados hagan una tarea similar pero con detalles de implementación diferentes.
- La intención del Factory Method es tener una clase a la cual delegar la responsabilidad de la creación de los objetos, para que no sea el mismo programador el que decida qué clase instanciará, si no que delegará esta responsabilidad al Factory confiando en que este le regresará la clase adecuada para trabajar.

COMPONENTES DEL PATRÓN DE DISEÑO FACTORY

- **AbstractFactory:** Este componente puede ser opcional, sin embargo, se recomienda la creación de un *AbstractFactory* que define el comportamiento por default de los *ConcreteFactory*.
- **Concrete Factory:** Representa una fábrica concreta la cual es utilizada para la creación de los *ConcreteProduct*, esta clase hereda el comportamiento básico del *AbstractFactory*.
- **ConcreteProduct:** Representa una implementación concreta de la interface *IProduct*, la cual es creada a través del *ConcreteFactory*.
- **IProduct:** Representa de forma abstracta el objeto que queremos crear, mediante esta interface se definen la estructura que tendrá el objeto creado.



SECUENCIA DE EJECUCIÓN

1. El cliente le solicita al *ConcreteFactory* la creación del *ProductA* .
2. El *ConcreteFactory* localiza la implementación concreta de *ProductA* y crea una nueva instancia.
3. El *ConcreteFactory* regresa el *ConcreteProductA* creado.
4. El cliente le solicita al *ConcreteFactory* la creación del *ProductB*.
5. El *ConcreteFactory* localiza la implementación concreta del *ProductB* y crea una nueva instancia.
6. El *ConcreteFactory* regresa el *ConcreteProductB* creado.