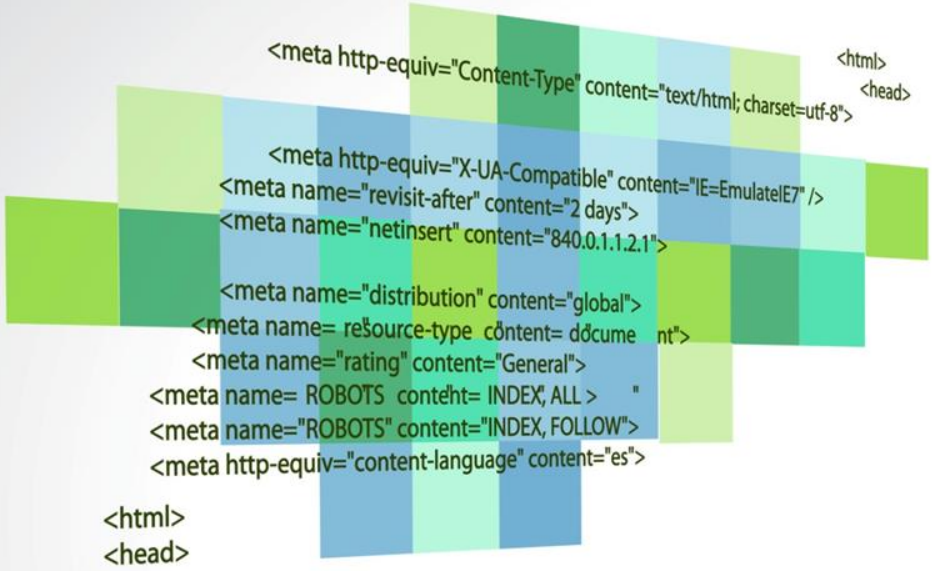


PROGRAMACIÓN ORIENTADA A OBJETOS

Patrones de Diseño

UNIDAD 4

Indicaciones

Para un estudio eficaz, te invitamos a que sigas las siguientes recomendaciones:

Lee con atención

Relaciona

Contrasta y complementa

Elabora

Realiza la tarea y participa en el foro

Logros de la sesión

Al finalizar esta unidad, estarás preparado para:

- ✓ Al finalizar la unidad el alumno aplica patrones de diseño en la solución de programas informáticos.

Temario

- ¿Qué es un patrón de Diseño?
- ¿Qué no es un patrón de Diseño?
- Tipos de Patrones de Diseño

¿Qúe es un patrón de Diseño?

Motivación

- En toda tarea humana, podemos encontrar una serie de **patrones que se repiten.** Incluso en las bellas artes, actividad representativa de la creatividad humana, podemos hallar carácterísticas comunes que nos permiten clasificar las obras en distintos movimientos artísticos. Y cómo no, en el arte de la programación no iba a ser menos.

Motivación

- Era de esperar, por tanto que alguien se animara tarde o temprano a estudiar los distintos patrones que pueden encontrarse en la inmensa mayoría del software, de forma que los problemas solucionados por estos quedasen perfectamente clasificados junto con su solución, para que así no fuese necesario reinventar la rueda cada vez que un programador se enfresentase a un obstáculo similar al descrito en uno de esos patrones. Nacieron así los **patrones de diseño de sistemas de software.**

Historia

- En 1994 apareció el libro “Design Patterns: Elements of Reusable Object Oriented Software” escrito por los ahora famosos Gang of Four (GoF).
- GoF = Erich Gamma,
  Richard Helm,
  Ralph Johnson,
  John Vlissides

¿Qúe no es un patrón de Diseño?

Un patrón de diseño no es un diseño final que puede ser transformado directamente a código fuente o código maquina.

Es una descripción o plantilla de cómo solucionar un problema que puede ser usado en muchas situaciones diferentes

Un patrón puede tener 1000 implementaciones

Tipos de Patrones

Tipos de Patrones

- De creación
  - Resuelven problemas relacionados con la creación de instancias de objetos
- De estructura
  - Se centran en problemas relacionados con la forma de estructurar las clases.
- De comportamiento
  - Permiten resolver problemas relacionados con el comportamiento de la aplicación, normalmente en tiempo de ejecución.

Tipos de Patrones

- De creación
  - Abstract Factory
  - Factory Method
  - Singleton
  - Prototype
  - Builder

Tipos de Patrones

- De estructura
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Facade
  - Flyweight
  - Proxy

Tipos de Patrones

- De comportamiento (1/2)
  - Chain of Responsability
  - Command
  - Interpreter
  - Iterator
  - Mediator
  - Memento
  - Observer

Tipos de Patrones

- De comportamiento (2/2)
  - State
  - Strategy
  - Template method
  - Visitor

Patrones de Creación

Patrones de Diseño e Interfaces

- Uno de los principios del diseño orientado a objetos es:

  "Programar con interfaces, no implementaciones"

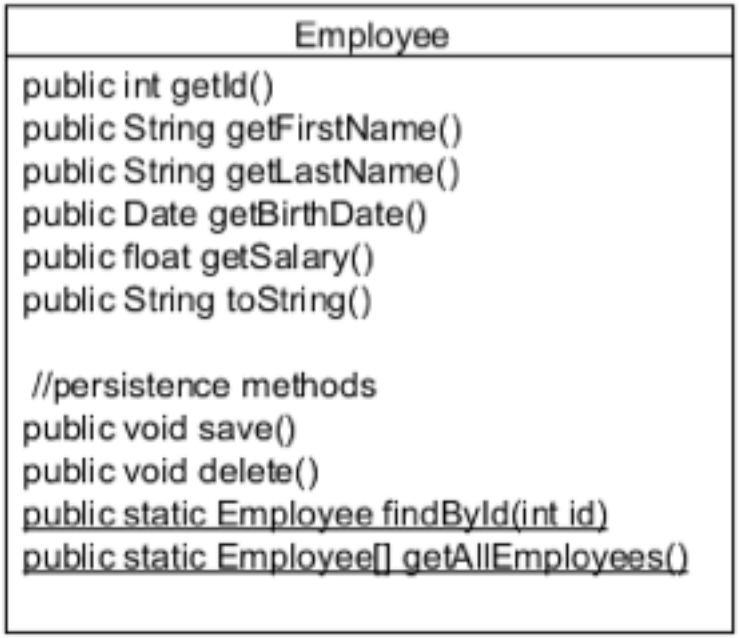
- Este es un tema común en muchos patrones de diseño. Este juega un rol importante en:
  - El patrón de diseño DAO
  - El patrón de diseño Factory

Patrón DAO

- El patrón de Data Access Object (DAO) es usado cuando creas una aplicación que debe persistir información. El patrón DAO:
  - Separa el dominio del problema de los mecanismos de persistencia
  - Usa una interface para definir los metodos que se usaran para la persistencia. Una interface permite que la implementación pueda ser reemplazada por:
    - ✓ Implementación basada en memoria como solución temporal
    - ✓ Implementación basada en archivos como una solución inicial
    - ✓ Implementación basada en JDBC para soportar persistencia en base de datos.
    - ✓ Implementación basada en JPA (Java Persistence API) para trabajar con persistencia en base de datos.

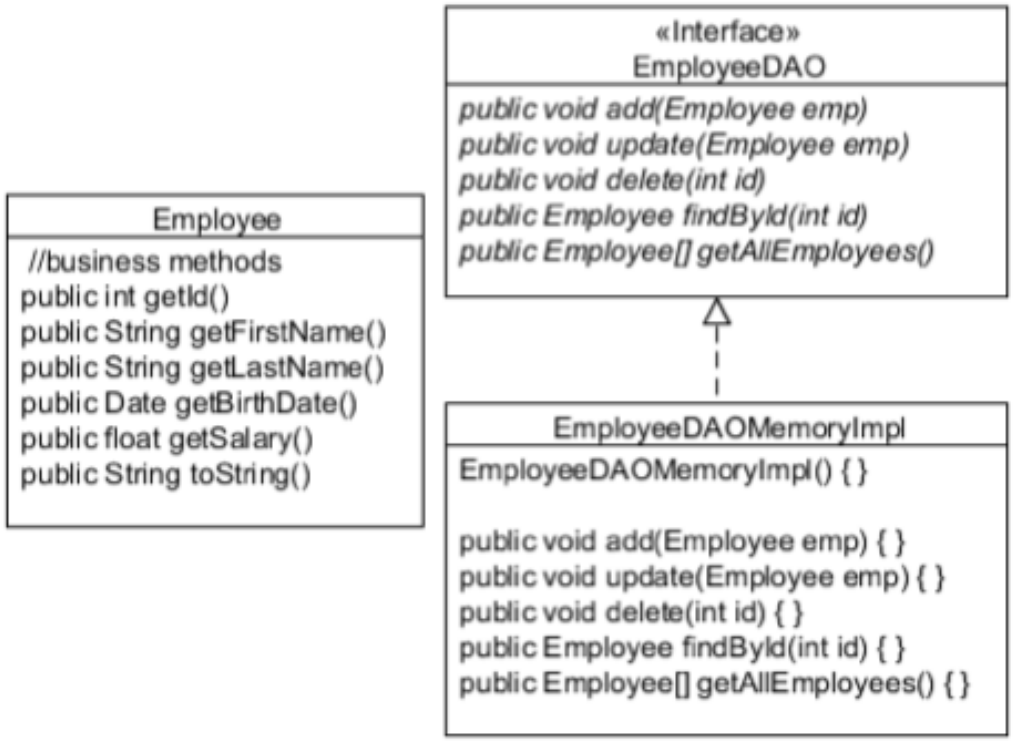
Antes del Patrón DAO

Observe como los métodos de persistencia estan combinados con los métodos de negocio.



Despues del Patrón DAO

El patrón DAO mueve la lógica de persistencia fuera de las clases del dominio en clases separadas.



La necesidad por el patrón Factory

El patrón DAO depende del uso de interfaces para definir una abstracción. Pero, igual se necesita la implementación de un DAO para una especifica implementación.

```
EmployeeDAO dao = new EmployeeDAOMemoryImpl();
```

Con el uso de la interface cualquier implementacion puede ser asignada. Se usa esta referencia en el código.

La implementación es seleccionada y ejecutada, pero, aparecera en muchos lugares de la aplicación

Usando el patrón Factory

Usando un Factory nos permite prevenir que nuestra aplicación este fuertemente acoplada a una implementación específica del DAO.

```
EmployeeDAOFactory factory = new EmployeeDAOFactory();
EmployeeDAO dao = factory.createEmployeeDAO();
```

La implementación de EmployeeDAO esta oculta

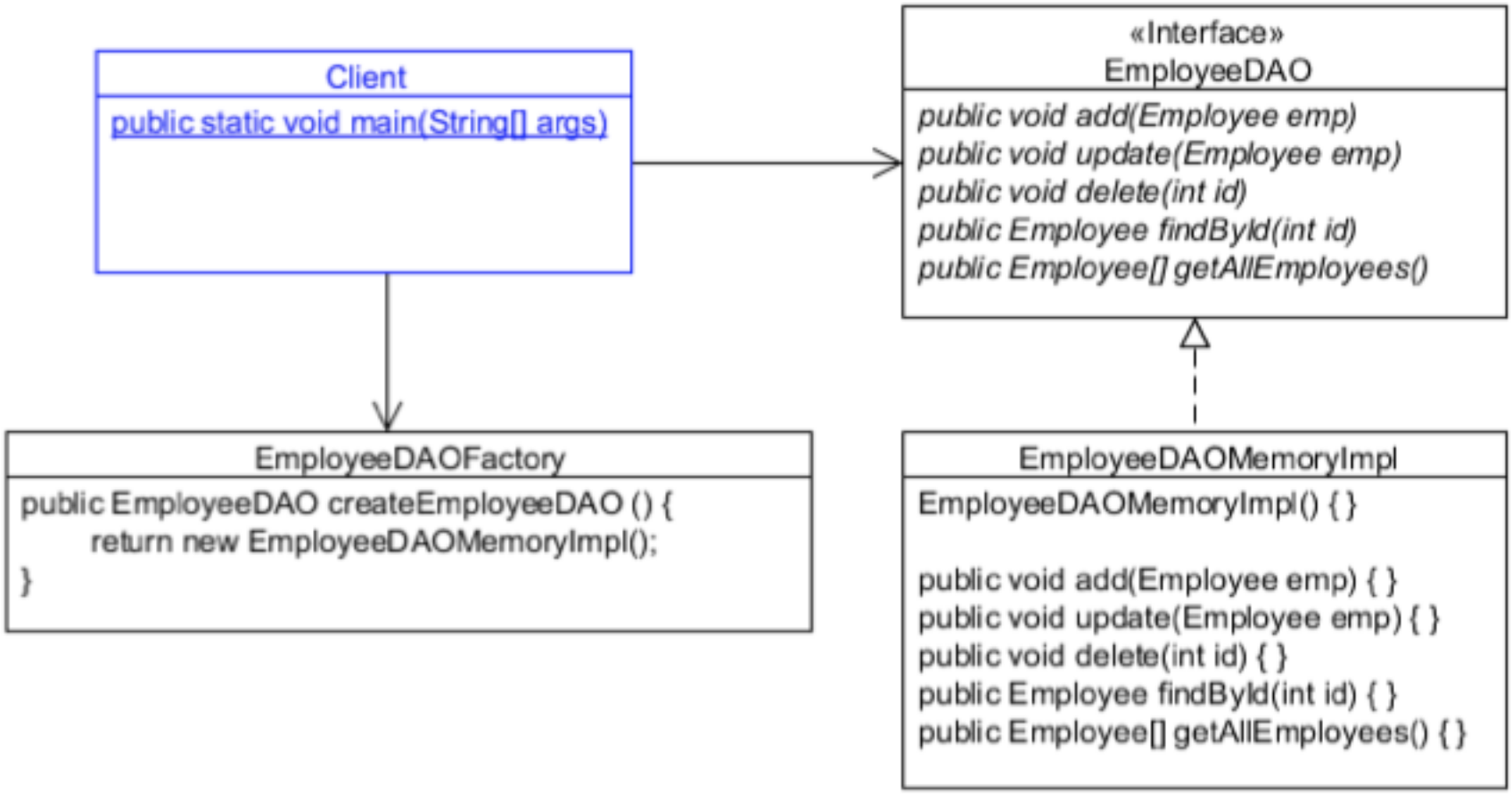
El Factory

La implementación del Factory es el único punto en la aplicación que debería depender de una clase concreta DAO.

```
public class EmployeeDAOFactory {

    public EmployeeDAO createEmployeeDAO() {
        return new EmployeeDAOMemoryImpl();
    }
}
```

Returns an interface typed reference

El Patrón DAO y Factory Juntos



El Patrón Singleton

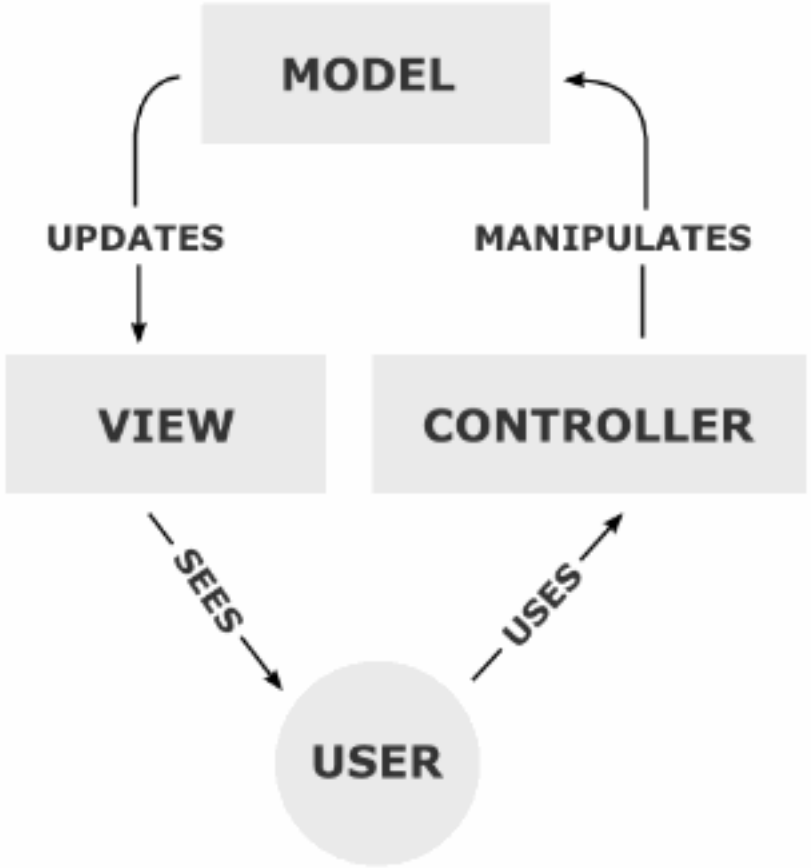
El patrón de diseño singleton detalla una implementación de clase que puede ser instanciada solo una vez.

```
public class SingletonClass {
1   private static final SingletonClass instance =
            new SingletonClass();

2   private SingletonClass() {}

    public static SingletonClass getInstance() {
3       return instance;
    }
}
```

El Model View Controller

El **modelo-vista-controlador (MVC)** es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos: que son el **modelo, la vista y la controladora.**
Es decir por un lado define componentes para la representación de la información y por otro lado para la interacción del usuario.

El Model View Controller



Descripción del Patrón

- El **Modelo**: es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.

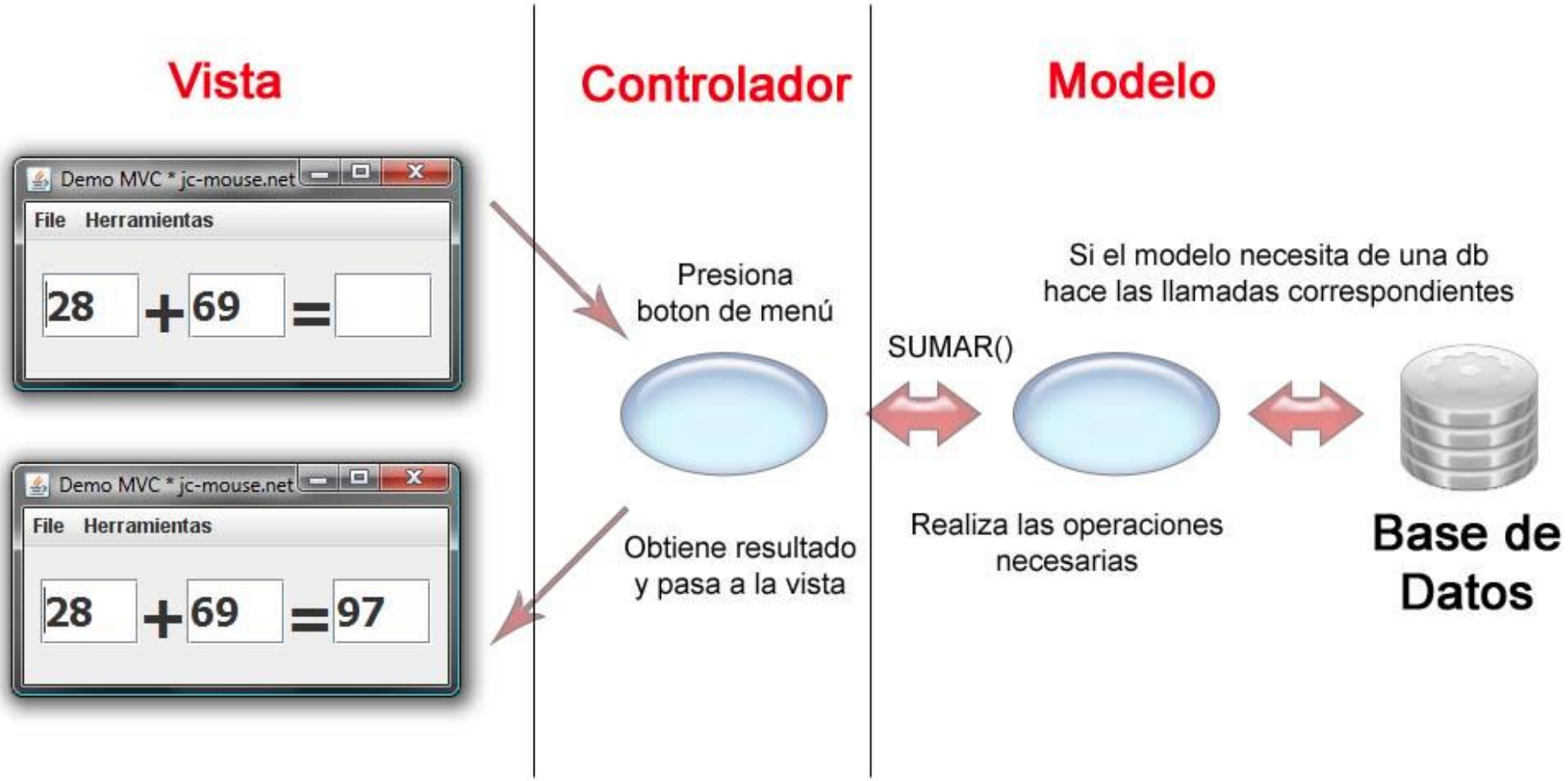
Descripción del Patrón

- El **Controlador**: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en la base de datos). El controlador es el intermediario entre la 'vista' y el 'modelo'.

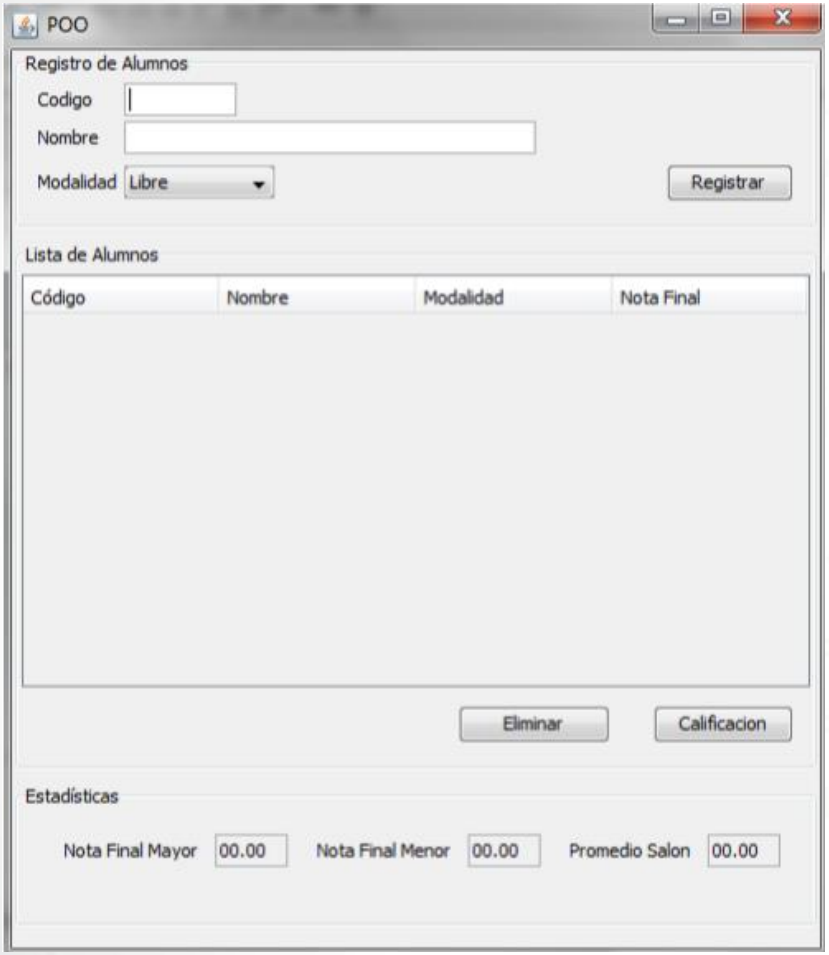
Descripción del Patrón

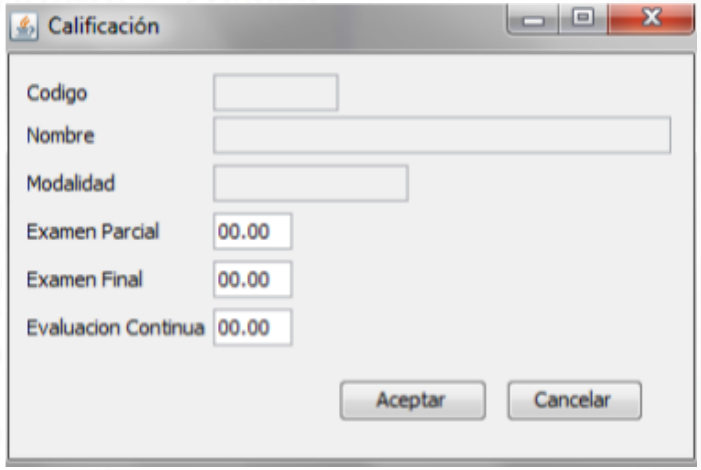
- La **vista**: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Ejemplo Simple

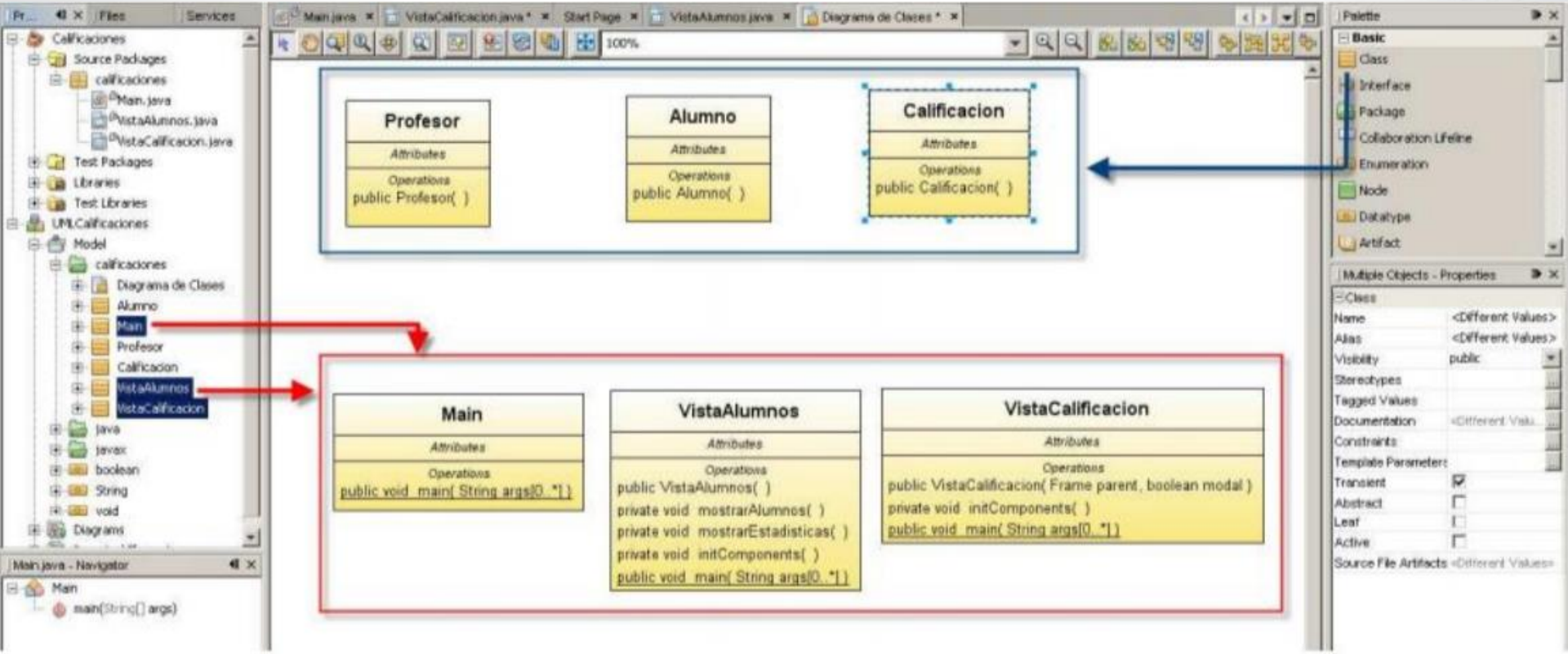


Ejemplo Registro Notas





Ejemplo Registro Notas



Alumno, Calificación: **Modelo** Profesor: **Controladora** , VistaXXXX: **Vistas**

Preguntas...

Si, luego del estudio de este tema, tienes dudas, ingresa al Aula Virtual y participa en el **foro de dudas académicas** de la unidad

Si quieres conocer más…

- Te invito a leer más sobre estos temas de los siguientes libros