Patrones de Diseño: Patrones Estructurales. Tema 4-6: Facade

Descripción del patrón

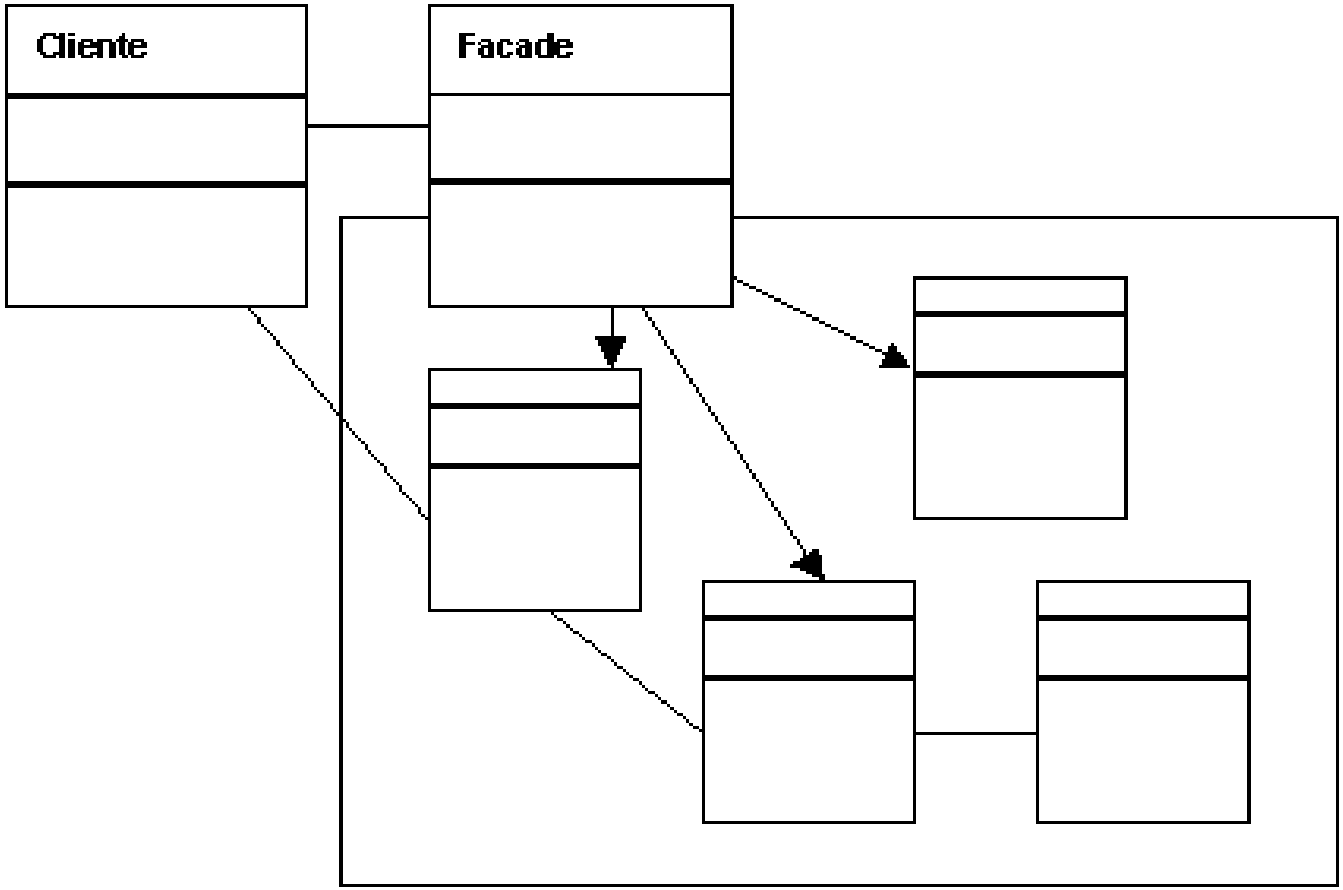
- **Nombre:**
  - Fachada
- **Propiedades:**
  - Tipo: estructural
  - Nivel: objeto, componente
- **Objetivo o Propósito:**
  - Simplifica el acceso a un conjunto de subsistemas o un sistema complejo. Representa una única interfaz unificada, que envuelve el subsistema, y es responsable de colaborar con los componentes del subsistema.

Aplicabilidad

- Use el patrón Facade cuando:
  - Tenga que reducir la dependencia entre clases, Facade ofrece un punto de acceso al resto de clases, si estas cambian o se sustituyen por otras, sólo hay que actualizar la clase Facade sin que el cambio afecte a las aplicaciones cliente.

    Facade no oculta las clases sino que ofrece una forma más sencilla de acceder a ellas, en los casos en que se requiere se puede acceder directamente a ellas.
  - Se ofrece un acceso sencillo a subsistemas complejos proporcionando una interfaz más sencilla sin eliminar las opciones avanzadas.
  - Tenga que descomponer un sistema en capas, puede utilizar Facade para ofrecer una interfaz de acceso entre las mismas.
  - Reducir el acoplamiento entre los clientes y los subsistemas.

Estructura



Estructura. Participantes

- **Facade:** Conoce las clases del subsistema y delega las peticiones de los clientes en los objetos del subsistema.
- **Clases del subsistema:** Implementan la funcionalidad del subsistema y llevan a cabo las peticiones que les envía la fachada, aunque no la conozcan.
- **Clientes:** Se comunican con el subsistema a través de la fachada, que reenvía las peticiones a los objetos del subsistema apropiados y puede realizar también algún trabajo de traducción. Los clientes que usan la fachada no necesitan acceder directamente a los objetos del sistema.

Estructura. Variaciones

- **Variaciones del patrón:**
- Se puede implementar la fachada como una interfaz o una clase abstracta. Con esto dejemos los detalles de implementación para un momento posterior y reducimos el acoplamiento.
- Varias fachadas pueden proporcionar distintas interfaces al mismo conjunto de subsistemas.

Consecuencias

- Oculta a los clientes parte de la complejidad de los subsistemas reduciendo así el número de objetos con los que tratan los clientes y haciendo que el subsistema sea más fácil de usar.
- Disminuye el acoplamiento entre los subsistemas y el cliente. Un acoplamiento débil nos permite modificar los componentes del sistema sin que sus clientes se vean afectados. Se facilita la estructuración en capas.
- No impide que los clientes usen las clases del subsistema directamente en caso de necesitarlo.

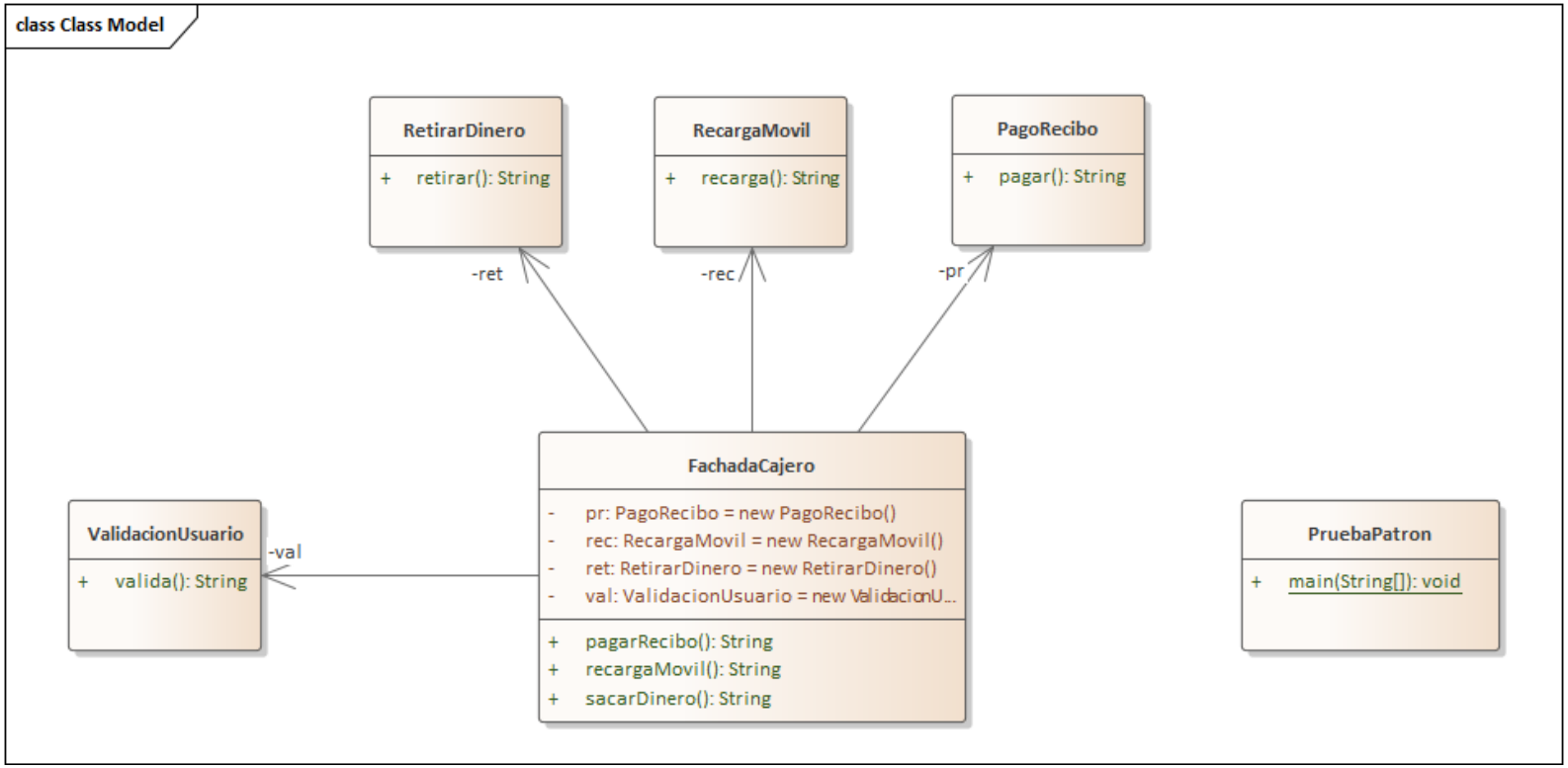
Patrones relacionados

- **Abstract Factory:** Abstract Factory crea familias de objetos relacionados, para simplificar su acceso se puede crear un objeto fachada para todos ellos.
- **Mediator:** Es muy parecido a Facade en el sentido de que abstrae funcionalidad a partir de clases existentes, pero se diferencian en que Mediator abstrae cualquier comunicación entre objetos similares pudiendo proporcionar comportamientos adicionales. Facade es solo una abstracción de la interfaz de varios subsistemas.
- **Singleton:** Normalmente se accede a las fachadas por medio de un patrón Singleton.
- **Adapter:** El patrón Adaptador puede utilizarse para envolver el acceso a sistemas que tienen interfaces diferentes. Esto es una clase de fachada pero el énfasis está en proporcionar adaptación a interfaces diferentes.

Código de ejemplo

CAJERO

Código de ejemplo



Código de ejemplo

- Identificamos a continuación los elementos del patrón:
  - Facade: FachadaCajero.
  - Clases del subsistema: ValidacionUsuario, RetirarDinero, PagoRecibo, RecargaMovil.
  - Clientes: PruebaPatron.