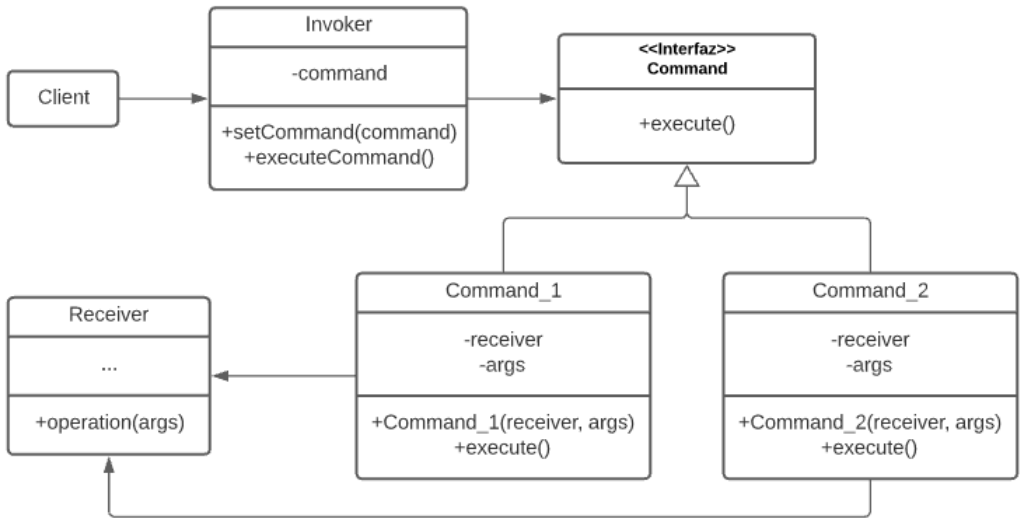
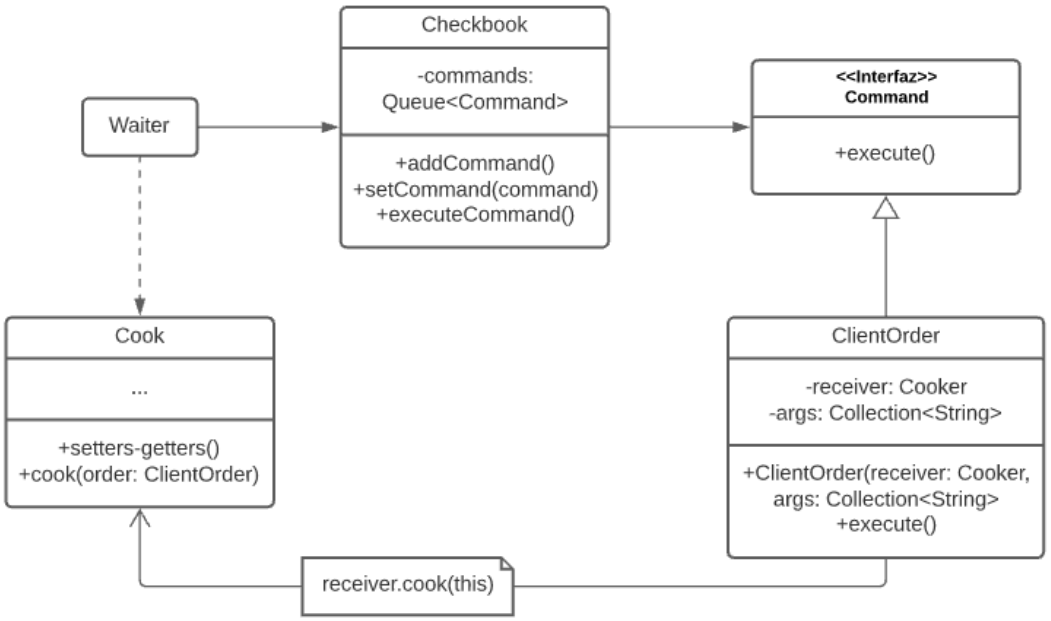
Primera parte. Identificación de casos propuestos

Trabajo asincrónico durante la semana 17 (lunes 14 a viernes 18 de junio).

Se presentan 6 casos que cubren los segundos patrones de comportamiento que hemos visto durante la semana 14 a 16. Los equipos deberán indicar el patrón con que se resuelve TODOS y cada uno de los casos, colocando los detalles de solución en la siguiente ficha:

| Problema No. 1 |
|---|
| Contexto:<br>Cuando se visita un restaurante, el camarero (a) toma el pedido de lo que van a desear los comensales, y la escriben en algo parecido a una orden. Esta orden, en el ámbito de los restaurantes se llama “check”, y se “encola” en lo que debe atender el cocinero pues vienen las indicaciones de lo que éste debe preparar para una mesa en particular. De tal modo que el talonario de “checks” puede ser utilizado por cada mesero sin depender del menú que se tenga diseñado y se pueden expresar en cada uno de ellos indicaciones para que el cocinero elabore platillos diferentes incluso con anotaciones particulares. |
| Patrón Propuesto: ( ) Comportamiento<br>**Command** |
| Diagrama original del patrón:<br>  <br> |
| Diagrama del patrón en el contexto del problema: |




Rol de cada elemento del patrón en la solución propuesta:

- Waiter: Se encarga de crear las órdenes para que el Cook las ejecute. Es el Client del patrón.
- Checkbook: Contiene una Queue de órdenes, la cual representa el talonario de órdenes. Es el Invoker del patrón.
- Command: Es el interface que se especializa en los tipos de órdenes de los clientes.
- Cook: Es el cocinero del restaurante, funciona como el Receiver del patrón
- ClientOrder: Es la implementación de Command, tiene una lista de parámetros que representan lo que anota el Waiter.

Problema No. 2

Contexto:

Los constructores de casas o edificios toman una base como una división sobre la cual pueden crear construcciones más grandes. Si lo piensa una casa de dos pisos es casi como dos veces una casa de una planta en cuanto a la fabricación, no al uso ni decoración que se le aportará.

Una subdivisión consiste en un conjunto limitado de pisos con variaciones para cada uno de ellos, estas variaciones se incluyen en cierto momento del proceso de construcción para producir una amplia gama de modelos.

Patrón Propuesto: ( ) Comportamiento

**Template**

Diagrama original del patrón:

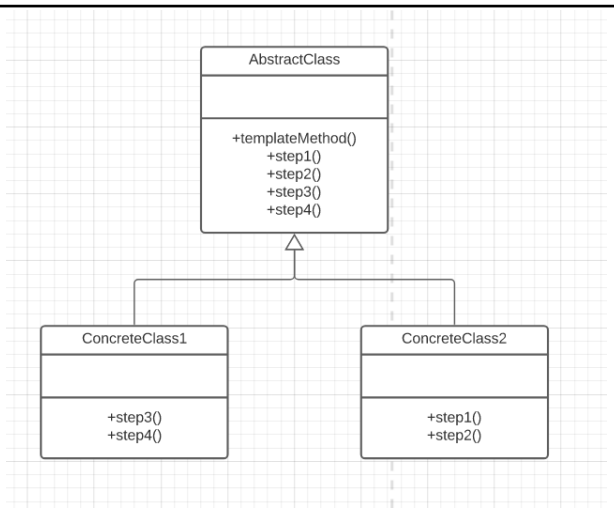
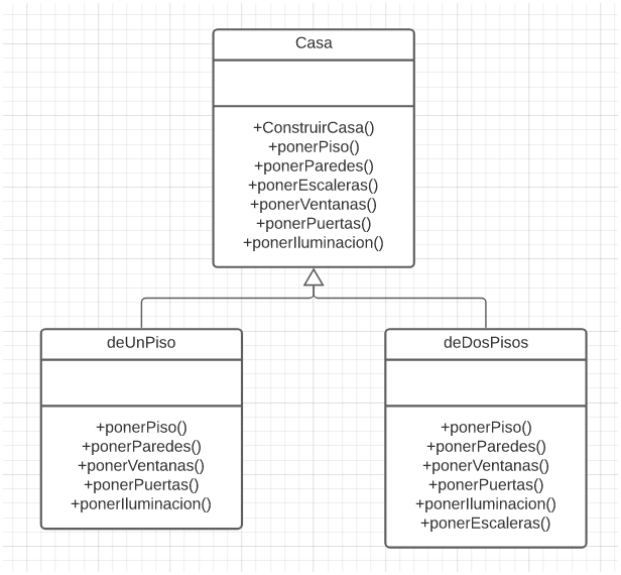



Diagrama del patrón en el contexto del problema:




Rol de cada elemento del patrón en la solución propuesta:

- Casa: Es una clase abstracta que declara los métodos como pasos para la construcción del algoritmo, en este caso una casa.
- deUnPiso y deDosPisos: Clases concretas que pueden sobreescribir todos los pasos, pero no el propio método de la plantilla.

Problema No. 3

Contexto:

Se requiere desarrollar una calculadora que pueda trabajar tres variables de tipo doble y poder realizar operaciones aritméticas sobre dichas variables, las cuales tienen un identificador y un posible valor, por ejemplo, x es el nombre de una variable y almacena un valor de tipo doble en un momento del tiempo.

Sobre las variables se pueden desarrollar varias acciones: asignar un valor específico, realizar una operación aritmética con otra variable o un valor literal (sumar, restar, multiplicar, dividir, elevar a la potencia, sacar raíz) y el resultado de esa operación quedar almacenado en la misma variable alterando su valor original. Ejemplo a la variable x le puedo sumar 3 y dejar el resultado en x.

La calculadora tiene una opción para guardar el estado de la calculadora en el momento en el que el usuario así lo desee y lo almacena en un historial de "savepoints" que se van incrementando secuencialmente en su identificación conforme el usuario indique que desea guardar el estado.

Patrón Propuesto: ( ) Comportamiento

**Memento**

Diagrama original del patrón:

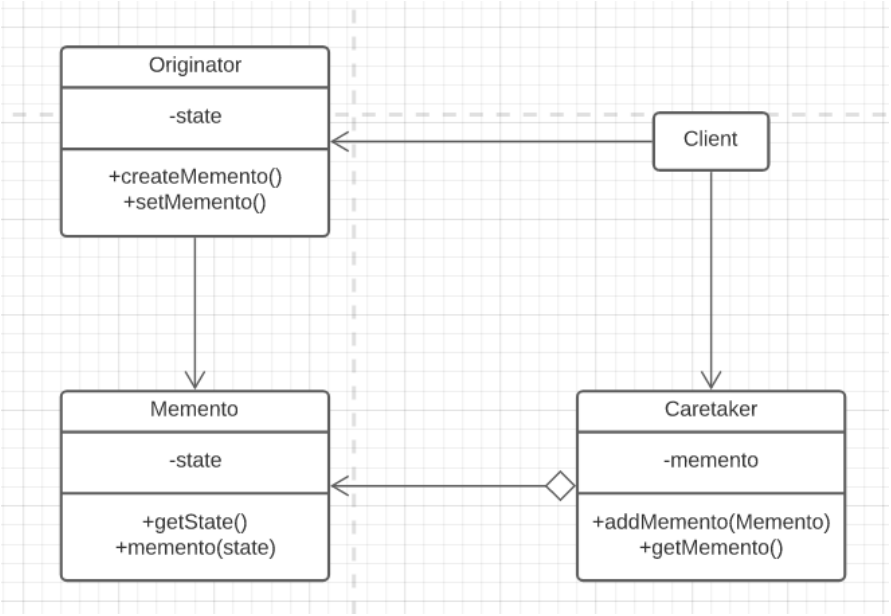
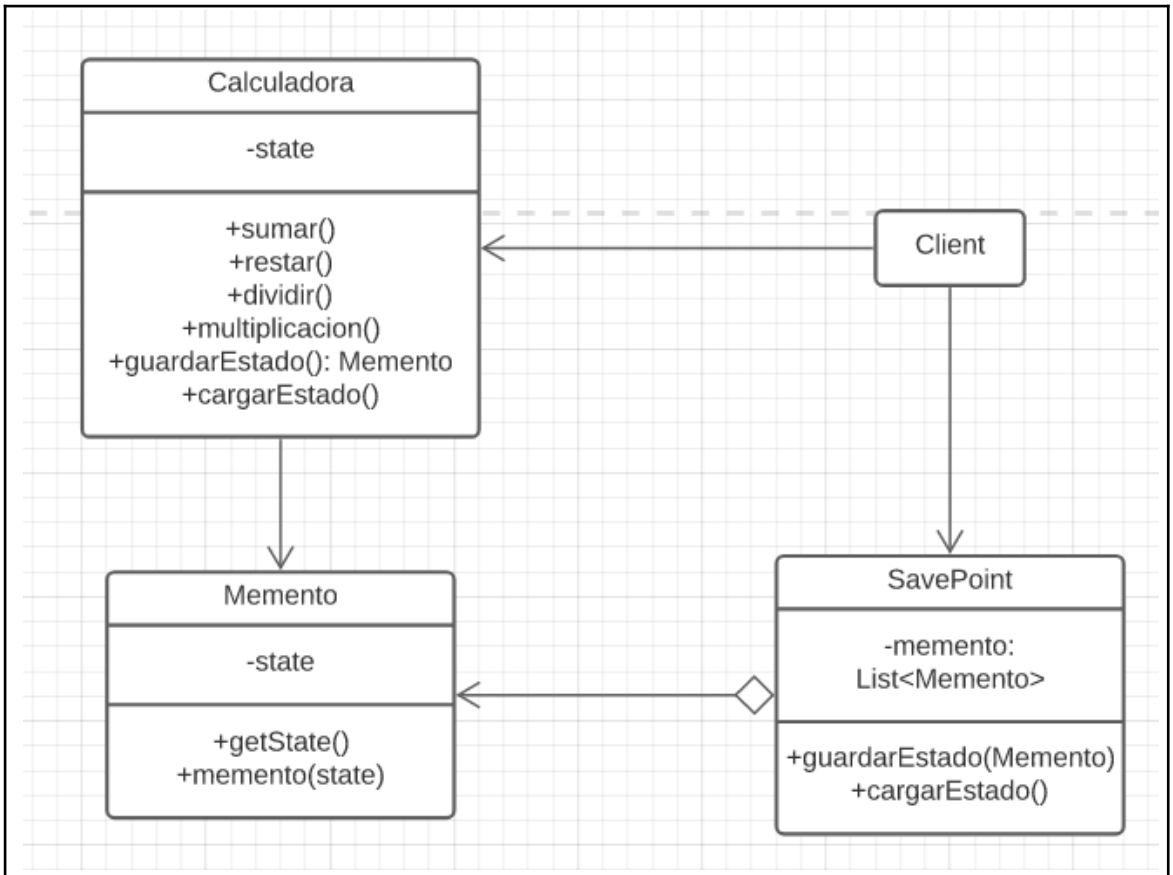



Diagrama del patrón en el contexto del problema:




Rol de cada elemento del patrón en la solución propuesta:

- Calculadora: Aquí es donde se puede cambiar el estado
- Memento: Este componente almacena el estado de la Calculadora en un cierto momento
- SavePoint: En este componente se registran los cambios de la Calculadora y poder viajar a los diferentes estados almacenados
- Client: Interactúa con la calculadora y crea nuevos estados usando el SavePoint.

Problema No. 4

Contexto: Su compañía ha adquirido un contrato para desarrollar una componente de software para una aplicación analítica a una compañía de salud, la cual deberá brindar información al usuario detalles sobre sus problemas particulares de salud, su historial, tratamientos, medicamentos, etc.
Para esto la compañía ha recibido una enorme cantidad de datos en una variedad de formatos. Se le ha encomendado a su equipo, desarrollar un componente para procesar la información y guardar estos datos en la base de datos de la compañía.

Los usuarios proporcionarán los datos en cualquier formato (archivos de texto, documentos, imágenes, audios, videos, por lo menos estos son los que se han identificado pero podrían aparecer nuevos) y usted debe proporcionar una única

interfaz para cargar los datos en la base de datos y al usuario que ha contratado el servicio realmente no le interesa la forma en la que esta información sea almacenada (si es estructurada o no) y al parecer su equipo necesitará desarrollar diferentes manejadores para guardar los distintos formatos de datos.

Patrón Propuesto: ( ) Comportamiento

**Visitor**

Diagrama original del patrón:

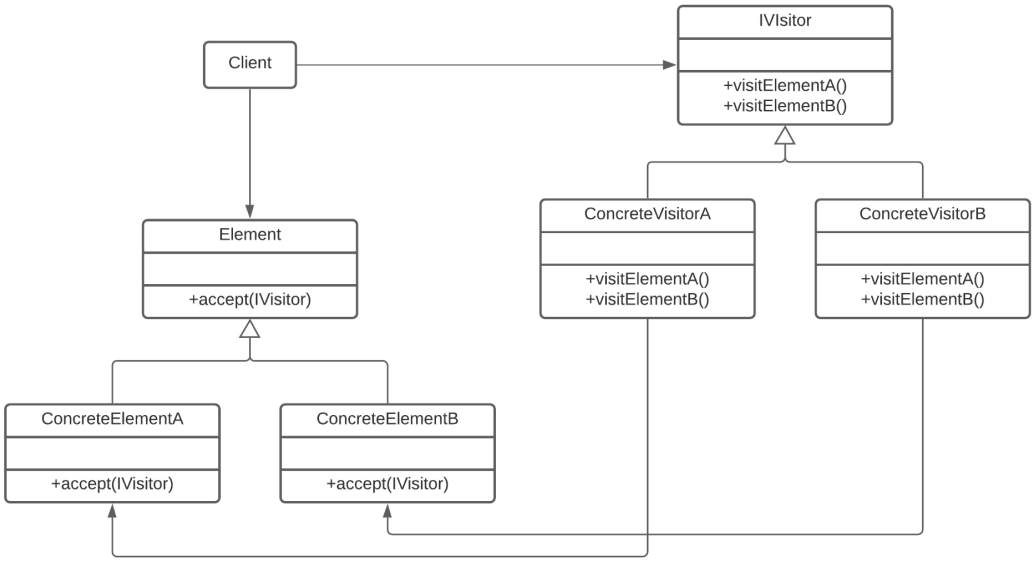
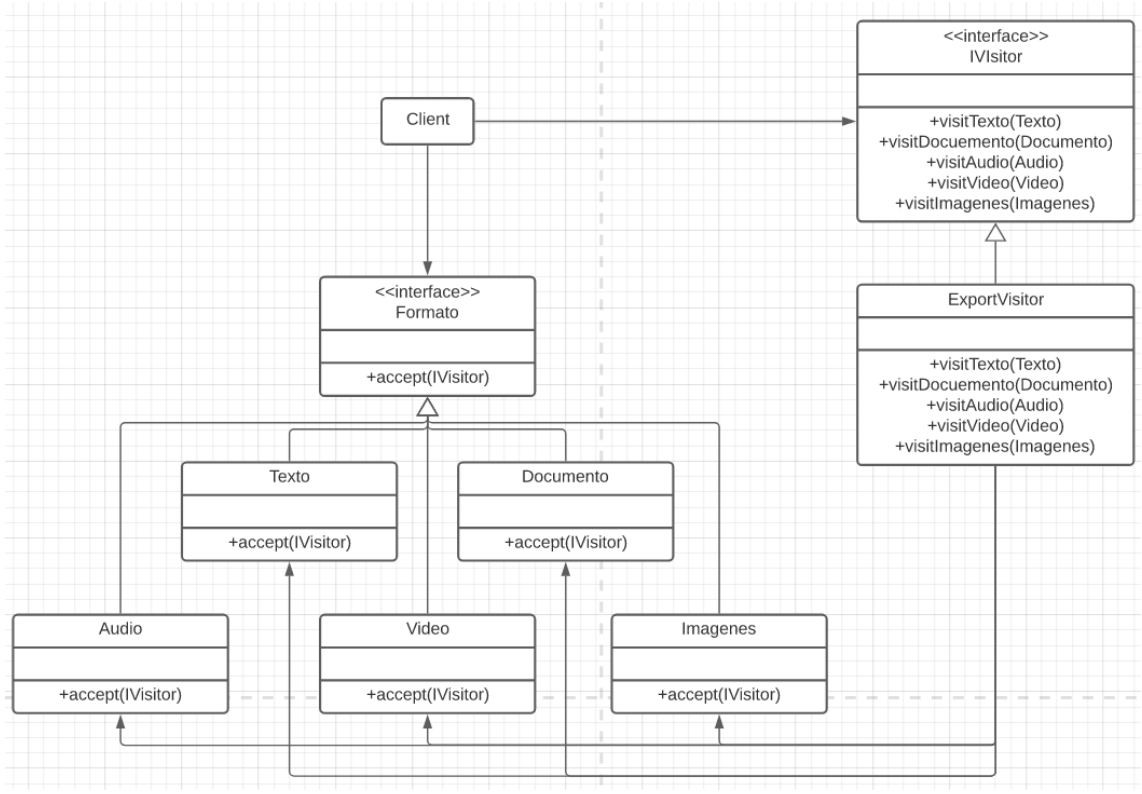



Diagrama del patrón en el contexto del problema:




Rol de cada elemento del patrón en la solución propuesta:

- Formato: Representa la raíz de la estructura, debe implementar todos los objetos de la estructura.
- Elementos(Documento, Video, etc.): Representa un hijo de la estructura compuesta, implementan el método de aceptación.
- IVIsitor: Interface que define la estructura del visitante.
- ExportVisitor: Representa una implementación del visitante y puede realizar una operación sobre el elemento
- Client: Interactúa con la estructura (Formato) y el visitante, crea los visitantes y los envía al elemento para procesarlos.

Problema No. 5

Contexto:

Suponga que se está desarrollando una aplicación financiera, para la cual se ha incorporado el patrón ***Composite*** para el manejo de su estructura dado que existen diferentes valores básicos o simples como acciones, bonos, certificados, fondos, pero además hay valores compuestos como carteras de valores (un cliente puede tener más de un tipo de valor simple bajo un mismo producto), cuentas.

Se ha decidido además utilizar un patrón adicional para realizar diferentes operaciones sobre un valor compuesto, tales como calcular su precio u obtener información fiscal anual. Resuelva este caso concentrado en la definición de este patrón adicional.

Patrón Propuesto: ( ) Comportamiento

**Chain of Responsibility**

Diagrama original del patrón:

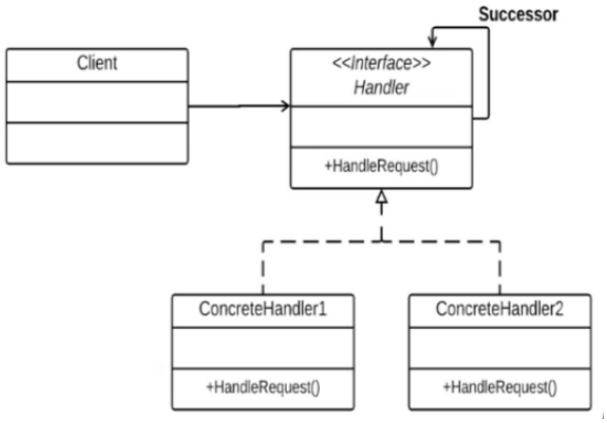
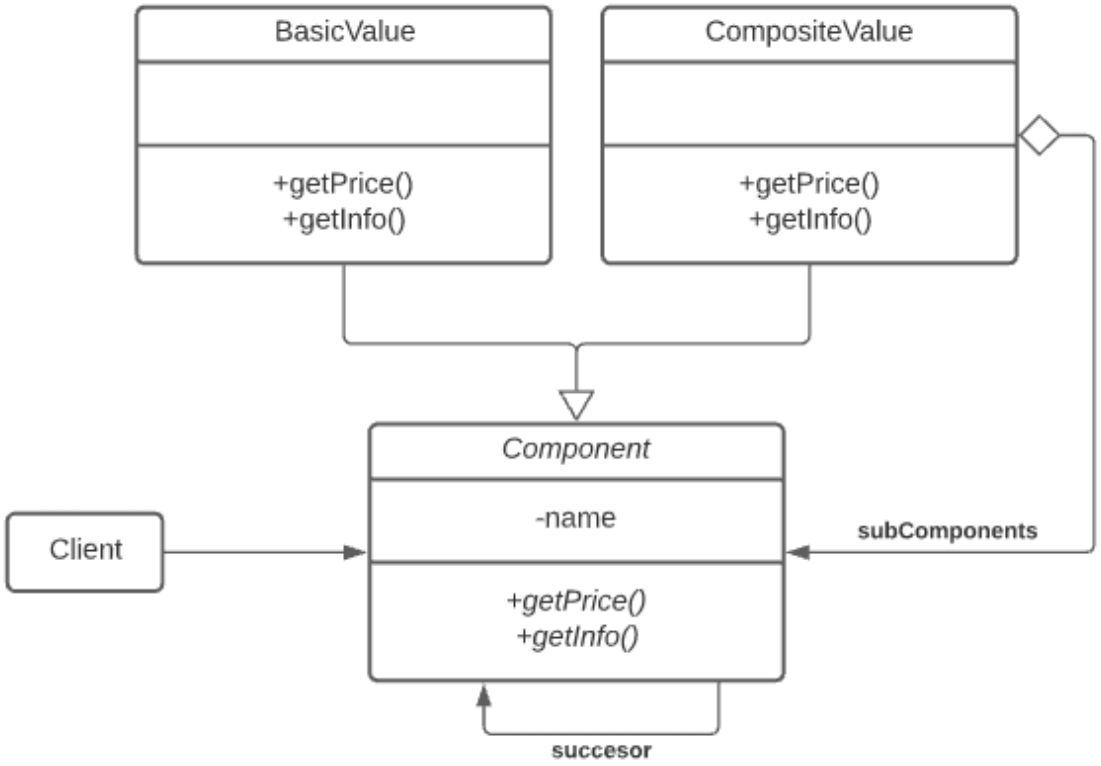



Diagrama del patrón en el contexto del problema:




Rol de cada elemento del patrón en la solución propuesta:

- Client: Es el usuario del sistema, el posee valores, los cuales son implementados con un Composite pattern y hace operaciones mediante el Chain of Responsibility.
- Component: Es la clase abstracta del Chain of Responsibility y a su vez, del Composite pattern.
- BasicValue: Clase básica del composite pattern. Representa los valores atómicos que un usuario puede tener, como lo son acciones y bonos. A su vez, es uno de los ConcreteHandlers del Chain of Responsability.
- CompositeValue: Clase compuesta del composite pattern. Representa los valores compuestos que un usuario puede tener, como lo son las carteras de valores. A su vez, es uno de los ConcreteHandlers del Chain of Responsability.

Problema No. 6

Contexto:
Una gran compañía electrónica le ha pedido que desarrolle una pieza de software para operar su nueva lavadora totalmente automática.

La compañía le ha proporcionado la especificación de hardware y el conocimiento práctico de la máquina. En la especificación, le han proporcionado los diferentes programas de lavado que soporta la máquina. Quieren producir un lavado totalmente automático, esto es que la máquina requerirá casi el 0% de interacción humana, por lo que el usuario sólo debe conectar la máquina con un toque para suministre agua, cargue la ropa para lavar, establezca el tipo de ropa en la máquina, como algodón, seda o jeans, etc., proporcione detergente y suavizante a sus bandejas respectivas, y presione el botón de inicio.

La máquina debe ser lo suficientemente inteligente como para llenar el agua en el tambor, tanto como sea necesario. Debe ajustar la temperatura de lavado sí mismo encendiendo el calentador, según el tipo de ropa en él. Debe arrancar el motor y girar el tambor tanto como se requiera, enjuague de acuerdo con las necesidades de la ropa, elimine la suciedad si es necesario y también con suavizante.

Como es sabido, esperemos, una lavadora tiene un tambor que es donde se deposita la ropa a lavar, además tiene un calentador, un sensor para comprobar la temperatura y un motor. Además, la máquina también tiene una válvula para controlar el suministro de agua, detergente y suavizante y la eliminación de residuos.

Todos estos componentes deben funcionar organizadamente entre ellos pues cada uno de ellos realiza de manera adecuada lo que su función, pero ésta sólo puede realizarse en cierto momento del tiempo cuando otro ya haya concluido su tarea.

Patrón Propuesto: ( ) Comportamiento

**Mediator**

Diagrama original del patrón:

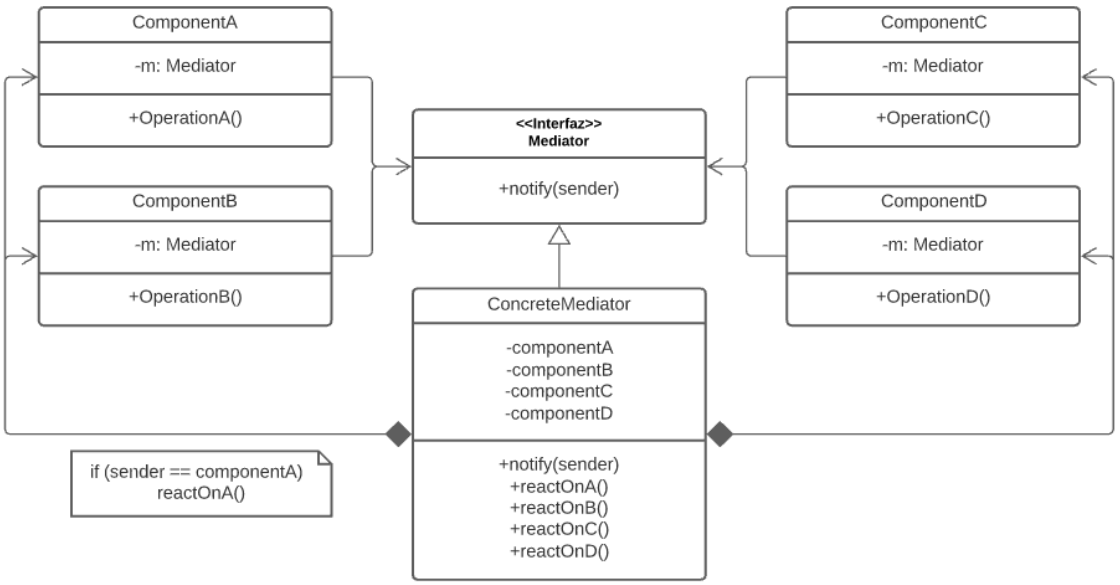
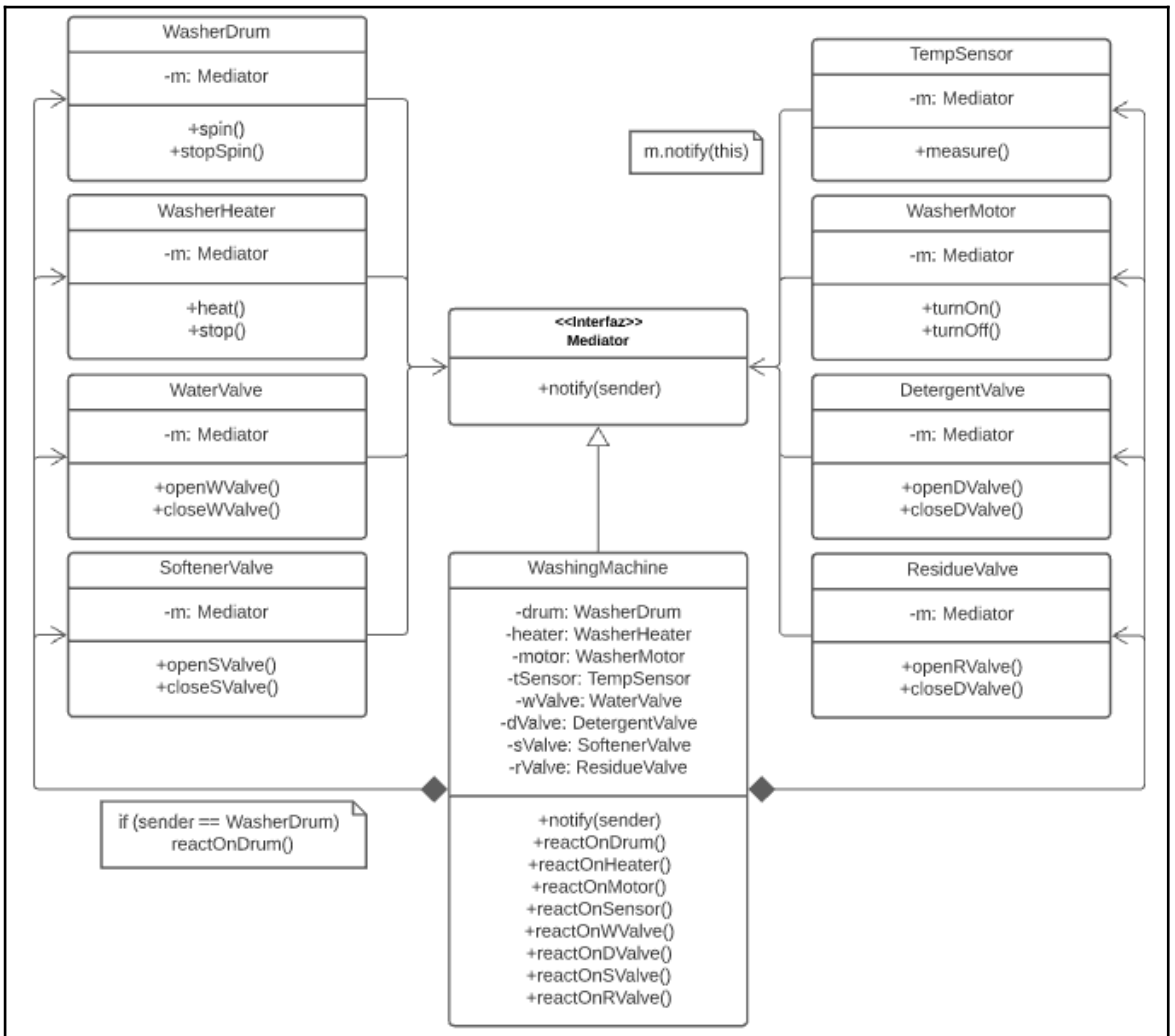



Diagrama del patrón en el contexto del problema:




Rol de cada elemento del patrón en la solución propuesta:

- Mediator: Es el interface que notifica que una operación ha ocurrido.
- WashingMachine: Representa la lavadora automatizada. Implementa el Mediator, por lo que sabe cuando actúa uno de los componentes.
- WasherDrum: Representa el tambor de la ropa. Puede girar o no girar.
- WasherHeater: Representa el calentador. Puede calentar o no calentar.
- WasherMotor: Representa el motor. Puede estar encendido o apagado.
- TempSensor: Representa el sensor de temperatura. Puede medir la temperatura.
- WaterValve: Representa la válvula de agua. Se puede abrir o cerrar.
- SoftenerValve: Representa la válvula de suavizante. Se puede abrir o cerrar.
- DetergentValve: Representa la válvula de detergente. Se puede abrir o cerrar.
- ResidueValve: Representa la válvula de residuos. Se puede abrir o cerrar.

**Estas fichas deben quedar subidas en el muro de publicaciones del canal privado al final de la clase asincrónica como evidencia de su trabajo de clase.**

## Segunda parte. Propuesta de un contexto para un patrón asignado

Adicionalmente, cada equipo deberá redactar un contexto de problema que pueda resolverse con un patrón asignado de acuerdo a la siguiente tabla:

| Equipo | Patrón Asignado | | Equipo | Patrón Asignado |
|---|---|---|---|---|
| 01 | Memento | | 06 | Command |
| 02 | Chain of Responsibility | | 07 | Template Method |
| 03 | Visitor | | 08 | Mediator |
| 04 | Mediator | | 09 | Visitor |
| 05 | Template Method | | 10 | Memento |

Coloque su contexto y su solución propuesta en UML en el siguiente cuadro:

| Patrón de Comportamiento para redacción de un contexto [Command] |
|---|
| Contexto Propuesto:<br>Se quiere hacer un programa que lea macros para controles en un reproductor de música para realizar diferentes acciones. Entre las acciones que se quieren tener están:<br>● Poner x2 de velocidad<br>● Reiniciar canción<br>● Adelantar 5 segundos<br>● Retroceder 5 segundos |
| Diagrama original del patrón: |

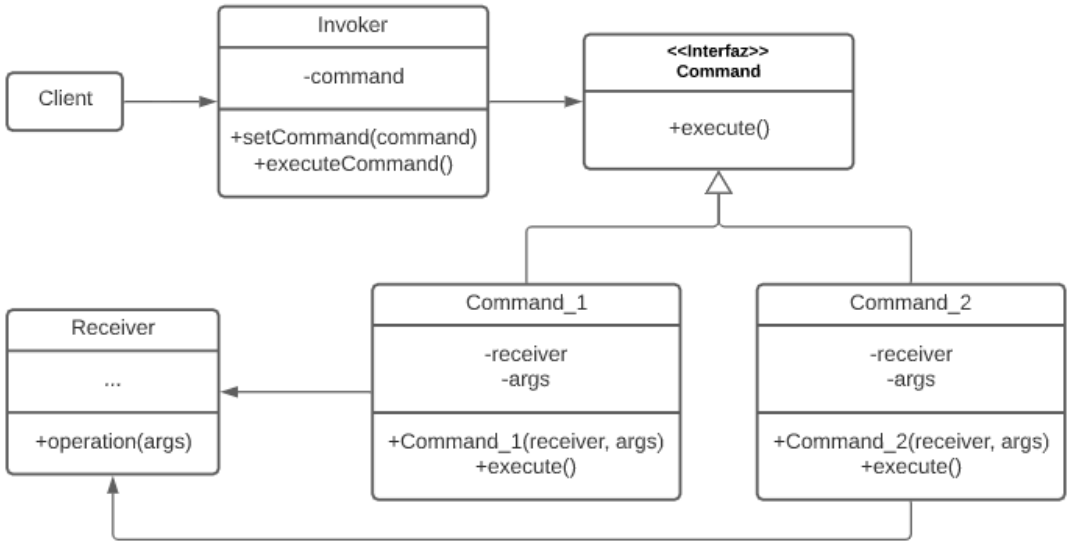
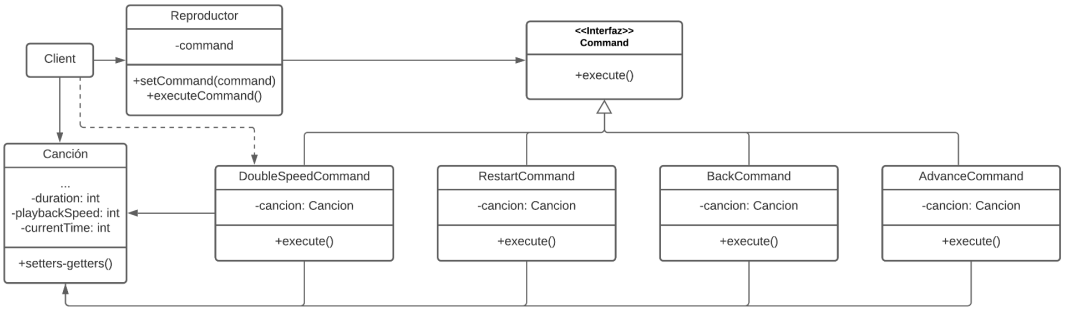



Diagrama del patrón en el contexto del problema:




Rol de cada elemento del patrón en la solución propuesta:

- Client: Clase principal del programa. Llama al Reproductor para ejecutar el comando que se le asignó con la canción que se va a trabajar
- Reproductor: Clase Invoker del patrón. Se le asigna un comando concreto y esta lo ejecuta sin importar el tipo de comando que sea.
- Cancion: Clase Receiver del patrón. Es manipulada por los comandos que se le pasen al Reproductor. Sus datos que son manipulados son el tiempo actual (currentTime) y velocidad de reproducción (playbackSpeed).
- Command: Interface con el método execute() para que los comandos concretos lo implementen.
- DoubleSpeedCommand: Comando concreto que implementa método execute() cambiando el plabackSpeed de la Cancion asignada a 2.0.
- RestartCommand: Comando concreto que implementa método execute() cambiando el currentTime de la Cancion asignada a 0.

- BackCommand: Comando concreto que implementa método execute() cambiando el currentTime de la Cancion asignada al original - 5.
- AdvanceCommand: Comando concreto que implementa método execute() cambiando el currentTime de la Cancion asignada al original + 5.

## Tercera parte. Programación de casos

De los casos de la primera parte se asigna un caso para que el equipo programe su propuesta de acuerdo a lo definido en la primera parte.

| Casos | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Equipo a cargo | 01-07 | 03-02 | 09 | 04-08 | 05-06 | 02 |

Para efectos de comprender entonces la instrucción se pone el siguiente ejemplo:

El equipo 01 completa todas las fichas de los 6 casos. Posteriormente propone un contexto de acuerdo al patrón MEMENTO asignado en la segunda parte y programa en la tercera parte tanto el contexto propuesto de la segunda parte como la solución del caso 1.

**Entregables:**

1. Imágenes de las tablas de resolución de casos en muro de publicaciones del canal privado de cada equipo.

2. Se entrega este documento completo con el proyecto con los dos casos programados para el jueves 03 de junio al final del día

## Caso 1


Cuando se visita un restaurante, el camarero (a) toma el pedido de lo que van a desear los comensales, y la escriben en algo parecido a una orden. Esta orden, en el ámbito de los restaurantes se llama “check”, y se “encola” en lo que debe atender el cocinero pues vienen las indicaciones de lo que éste debe preparar para una mesa en particular. De tal modo que el talonario de “checks” puede ser utilizado por cada mesero sin depender del menú que se tenga diseñado y se pueden expresar en cada uno de ellos indicaciones para que el cocinero elabore platillos diferentes incluso con anotaciones particulares.

## Caso 2

Los constructores de casas o edificios toman una base como una división sobre la cual pueden crear construcciones más grandes. Si lo piensa una casa de dos pisos es casi como dos veces una casa de una planta en cuanto a la fabricación, no al uso ni decoración que se le aportará.

Una subdivisión consiste en un conjunto limitado de pisos con variaciones para cada uno de ellos, estas variaciones se incluyen en cierto momento del proceso de construcción para producir una amplia gama de modelos.

Este es el caso de la tendencia de construcción de locales comerciales y habitacionales con contenedores.

Este link muestra un artículo que destaca el uso de contenedores de furgón como subdivisión para construir lugares muy cómodos y sustentables, y posiblemente en su proceso se utilice el mismo principio de diseño que usamos para reutilizar estructuras.

https://www.infobae.com/economia/real-estate/2019/03/06/sustentables-y-practicos-los-containers-vivienda-la-tendencia-que-llego-para-quedarse/

**Caso 3**

---

Se requiere desarrollar una calculadora que pueda trabajar tres variables de tipo doble y poder realizar operaciones aritméticas sobre dichas variables, las cuales tienen un identificador y un posible valor, por ejemplo, x es el nombre de una variable y almacena un valor de tipo doble en un momento del tiempo.

Sobre las variables se pueden desarrollar varias acciones: asignar un valor específico, realizar una operación aritmética con otra variable o un valor literal (sumar, restar, multiplicar, dividir, elevar a la potencia, sacar raíz) y el resultado de esa operación quedar almacenado en la misma variable alterando su valor original. Ejemplo a la variable x le puedo sumar 3 y dejar el resultado en x.

La calculadora tiene una opción para guardar el estado de la calculadora en el momento en el que el usuario así lo desee y lo almacena en un historial de "savepoints" que se van incrementando secuencialmente en su identificación conforme el usuario indique que desea guardar el estado.

El historial de las operaciones registradas siempre indica la operación que se realizó, los valores de los operandos antes y después de la operación.

Cuando se solicita la opción de guardar el estado se despliega el indicador de SAVEPOINT i, donde i es es i-ésimo savepoint solicitado.

La calculadora provee además las opciones de deshacer en varias modalidades:

- Un simple deshacer restablecería el estado de la calculadora al estado antes de la última operación realizada.
- Un deshacer a un punto específico del historial de operaciones que restaurará la calculadora y los valores de las variables a ese punto. Este punto particular se referencia por parte del usuario indicando el número de savepoint al que desea retornar.
- Un deshacer todo que borrará todo el estado de la calculadora y restaura las

variables al punto de inicio.

**Caso 4**

Su compañía ha adquirido un contrato para desarrollar una componente de software para una aplicación analítica a una compañía de salud, la cual deberá brindar información al usuario detalles sobre sus problemas particulares de salud, su historial, tratamientos, medicamentos, etc.

Para esto la compañía ha recibido una enorme cantidad de datos en una variedad de formatos. Se le ha encomendado a su equipo, desarrollar un componente para procesar la información y guardar estos datos en la base de datos de la compañía.

Los usuarios proporcionarán los datos en cualquier formato (archivos de texto, documentos, imágenes, audios, videos, por lo menos estos son los que se han identificado pero podrían aparecer nuevos) y usted debe proporcionar una única interfaz para cargar los datos en la base de datos y al usuario que ha contratado el servicio realmente no le interesa la forma en la que esta información sea almacenada (si es estructurada o no) y al parecer su equipo necesitará desarrollar diferentes manejadores para guardar los distintos formatos de datos.

**Caso 5**

---

Suponga que se está desarrollando una aplicación financiera, para la cual se ha incorporado el patrón ***Composite*** para el manejo de su estructura dado que existen diferentes valores básicos o simples como acciones, bonos, certificados, fondos, pero además hay valores compuestos como carteras de valores (un cliente puede tener más de un tipo de valor simple bajo un mismo producto), cuentas.

Se ha decidido además utilizar un patrón adicional para realizar diferentes operaciones sobre un valor compuesto, tales como calcular su precio u obtener información fiscal anual. Resuelva este caso concentrado en la definición de este patrón adicional.

**Caso 6**

---

Una gran compañía electrónica le ha pedido que desarrolle una pieza de software para operar su nueva lavadora totalmente automática.

La compañía le ha proporcionado la especificación de hardware y el conocimiento práctico de la máquina. En la especificación, le han proporcionado los diferentes programas de lavado que soporta la máquina. Quieren producir un lavado totalmente automático, esto es que la máquina requerirá casi el 0% de interacción humana, por lo que el usuario sólo debe conectar la máquina con un toque para suministre agua, cargue la ropa para lavar, establezca el tipo de ropa en la máquina, como algodón, seda o jeans, etc., proporcione detergente y suavizante a sus bandejas respectivas, y presione el botón de inicio.

La máquina debe ser lo suficientemente inteligente como para llenar el agua en el tambor, tanto como sea necesario. Debe ajustar la temperatura de lavado sí mismo encendiendo el calentador, según el tipo de ropa en él. Debe arrancar el motor y girar el tambor tanto como se requiera, enjuague de acuerdo con las necesidades de la ropa, elimine la suciedad si es necesario y también con suavizante.

Como es sabido, esperemos, una lavadora tiene un tambor que es donde se deposita la ropa a lavar, además tiene un calentador, un sensor para comprobar la temperatura y un motor. Además, la máquina también tiene una válvula para controlar el suministro de agua, detergente y suavizante y la eliminación de residuos.

Todos estos componentes deben funcionar organizadamente entre ellos pues cada uno de ellos realiza de manera adecuada lo que su función, pero ésta sólo puede realizarse en cierto momento del tiempo cuando otro ya haya concluido su tarea.