Juan Esteban Diaz Castro – 202121646

Taller #5 – Patrones

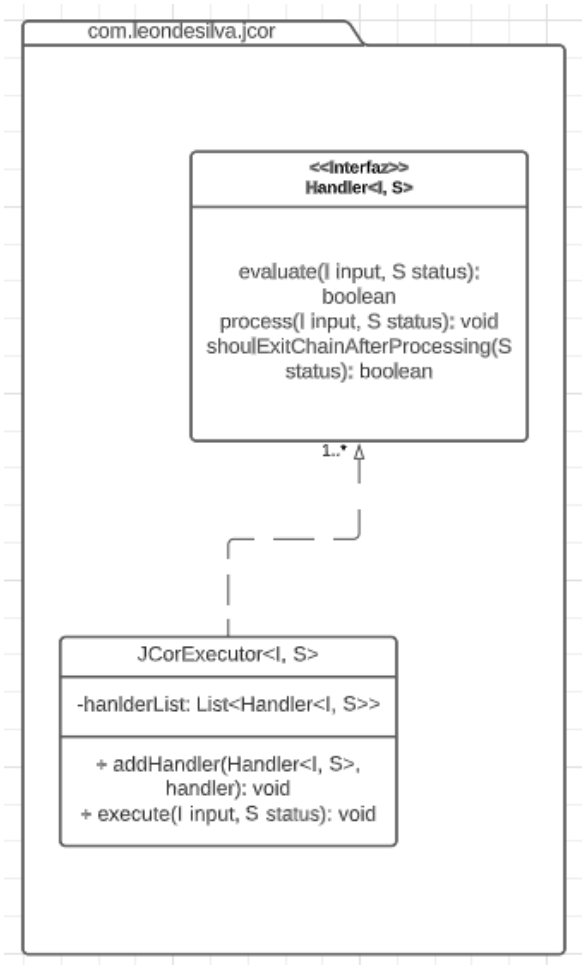
Patrón seleccionado: “Chain of Responsability”

Proyecto Seleccionado: https://github.com/LeonDeSilva/jcor.git

I. Información General

El proyecto seleccionado para el taller es una librería que implementa el patrón “Chain of Responsibility”. El autor de este proyecto asegura que esta librería es ideal para evitar escribir multiples “else if”, que hacen que el código no sea organizado del todo. Con este proyecto se busca hacer que cada instancia de un condicional (if, else if, else) sea uno de los “handlers” (Trabajadores) que existen en el “Chain of Responsibility” y que por medio de estos se pueda hacer recrear la cadena y a su vez realizar el condicional. Lo difícil del proyecto es la cantidad de “handlers” que puede llegar a manejar el proyecto, haciendo que el espacio de este sea muy alto.

II. Estructura



Luego para el archivo de prueba se debe implementar la interfaz de Handler para poder crear todos los necesarios (haciendo referencia a los condicionales). Una vez eso, se hace llamado al JCorExecutor dentro de la clase (se hace una asociación) y queda listo para el uso.

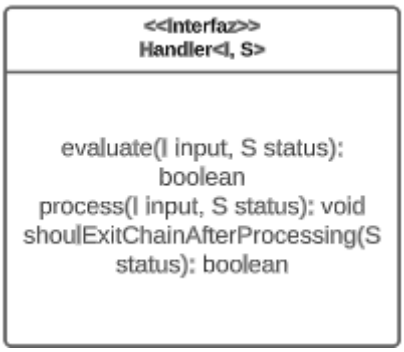
III. Patrón

El patrón seleccionado es “Chain of Responsibility” conocido también como “CoR” o “Chain of Command”. Este patrón es un patrón de diseño de comportamiento (de tipo Behavioral) que funciona por medio de la comunicación entre distintos “trabajadores” de distintas solicitudes en un proyecto. Una vez que llega a un “trabajador” este decide si realiza la solicitud o se la pasa al siguiente “trabajador” de la cadena.

Usualmente se puede usar en llamadas de soporte tecnológico donde uno llama, una grabadora lo conecta a uno con un asesor y en caso de no tener solución alguna esta solicitud pasa a alguien con mayor conocimiento y responsabilidad.

IV. Aplicación del Patrón

El patrón se está utilizando dentro del proyecto en los distintos “Handler”. Esto pues cada “Handler” está programado para evaluar si puede cumplir con el input que se le suministra, en caso de que si lo pueda realizar retorna “true” y en caso de que no pueda realizarlo retorna “false”. Si puede realizar el proceso hace uso del método “process”. También el último método se encarga de evaluar si el proceso debe ser completado por el “Handler” actual y en caso de no ser así se pasa al siguiente “Handler”.



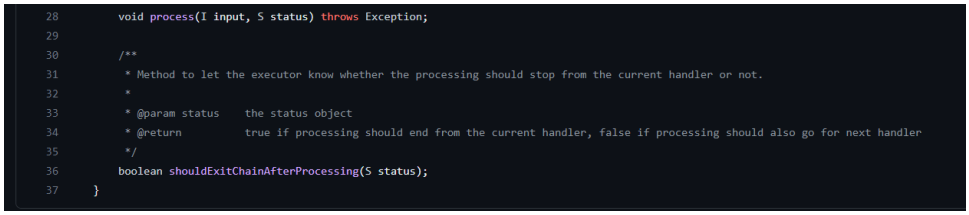
Code Blame 37 lines (33 loc) · 1.23 KB

```java
package com.leondesilva.jcor;

/**
 * Interface to represent a handler in JCor.
 * Handler is equal to a single node or processor in chain of responsibility design.
 *
 * @param <I>   the input object
 * @param <S>   the status object
 */
public interface Handler<I, S> {

    /**
     * Method to evaluate whether the current handler is capable for processing the given input object.
     *
     * @param input  the input object
     * @param status the status object
     * @return  true if processing should happen through the current handler, else returns false
     */
    boolean evaluate(I input, S status);

    /**
     * Method to do the processing.
     *
     * @param input     the input object
     * @param status    the status object keep track of the status of execution
     * @throws Exception if an error occurs
     */
```



```
28          void process(I input, S status) throws Exception;
29
30          /**
31           * Method to let the executor know whether the processing should stop from the current handler or not.
32           *
33           * @param status    the status object
34           * @return          true if processing should end from the current handler, false if processing should also go for next handler
35           */
36          boolean shouldExitChainAfterProcessing(S status);
37      }
```

V. Ventajas del Uso del Patrón

La principal ventaja es que por medio de este patrón se puede simplificar los condicionales haciéndolos aún más llamativos para que de ese modo el programa no sea tan desordenado.

VI. Desventajas del Uso del Patrón

Una de las principales desventajas de este proyecto es la cantidad de espacio que se necesita para los “handlers”. Además, cuando uno de estos pueda realizar la solicitud los que le siguen a este quedan inutilizados haciendo que se desperdicie el espacio para estos.

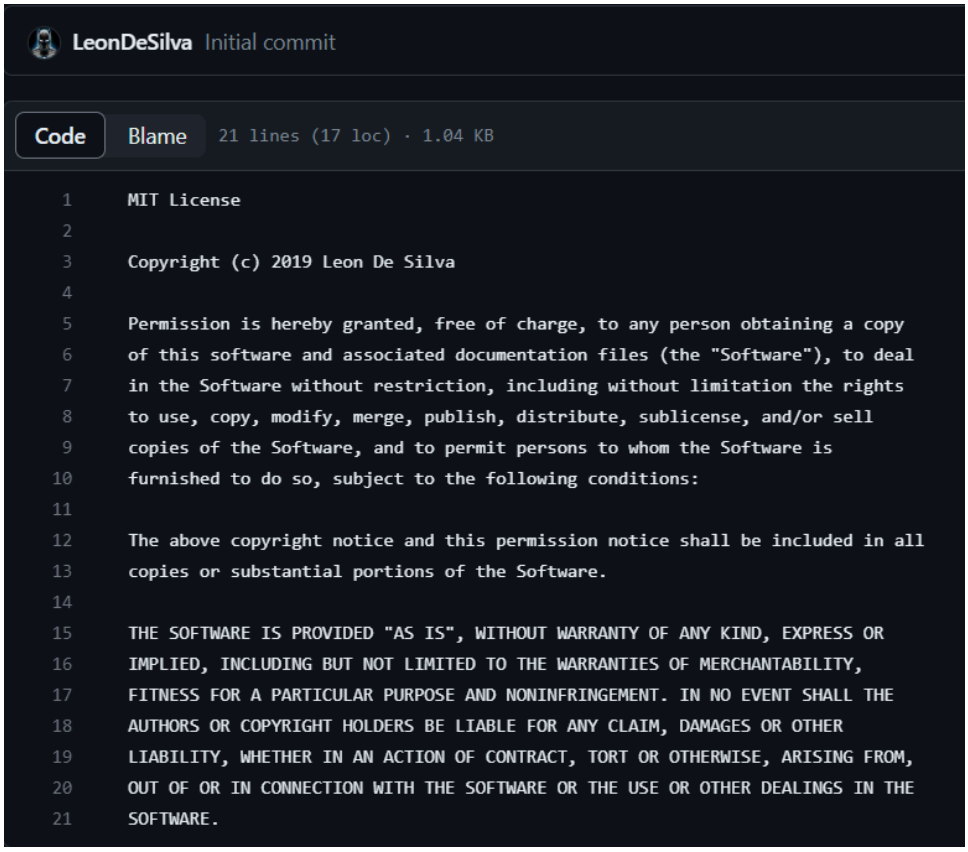
VII. Formas Alternativas

Como formas alternativas lo principal sería realizar un “switch” o también crear clases donde se maneje cada excepción del condicional. Sin embargo, estos procesos no evitan el uso de espacio y generación de desorden dentro de los métodos de una clase. De ahí que, esta resulta ser la manera más efectiva para resolver el problema de optimización de los condicionales.

VIII. Referencias

Refactoring.Guru. (2023). Chain of Responsibility. *Refactoring.Guru*. https://refactoring.guru/design-patterns/chain-of-responsibility#:~:text=Chain%20of%20Responsibility%20is%20a,next%20handler%20in%20the%20chain.

LeonDeSilva. (2019). *GitHub - LeonDeSilva/jcor: A Java chain of responsibility library*. GitHub. https://github.com/LeonDeSilva/jcor.git


LeonDeSilva Initial commit
Code
Blame
21 lines (17 loc) · 1.04 KB
MIT License
Copyright (c) 2019 Leon De Silva
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.