**Chain Of Responsibility Design Pattern**

The Chain of Responsibility design pattern is a [behavioral design pattern](https://www.geeksforgeeks.org/behavioral-design-patterns/) that allows an object to pass a request along a chain of handlers. Each handler in the chain decides either to process the request or to pass it along the chain to the next handler.

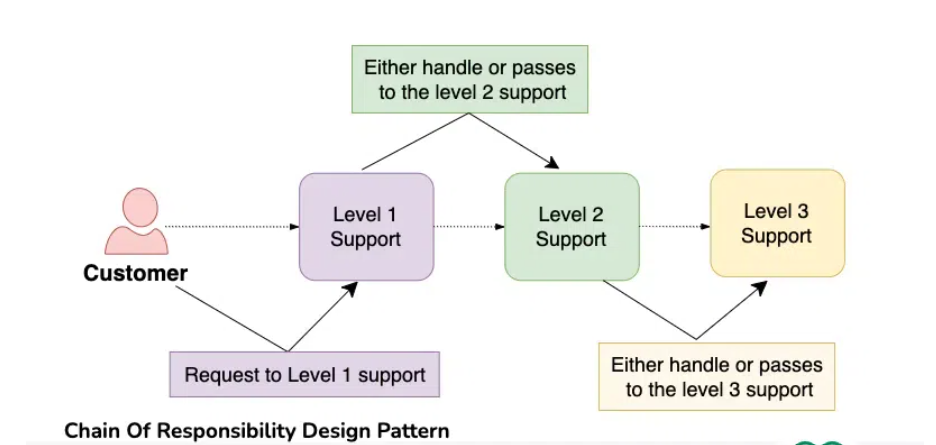
**Characteristics of the Chain of Responsibility Design Pattern**

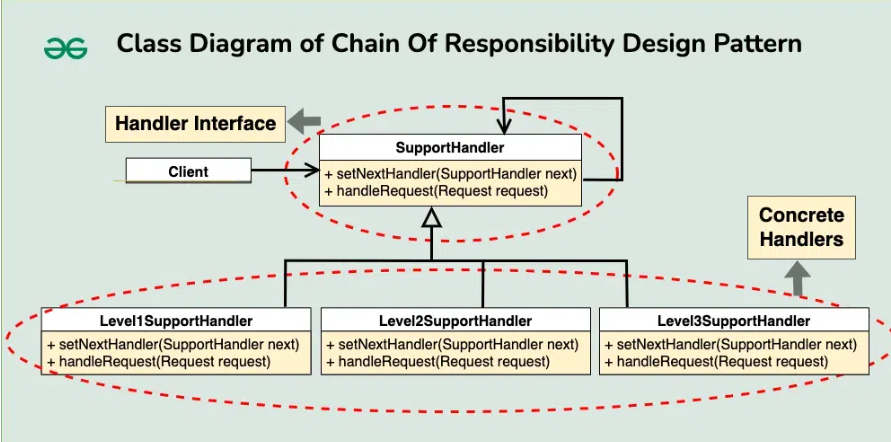
* Loose Coupling: The pattern promotes loose coupling between the sender and receiver of a request, as the sender doesn’t need to know which object will handle the request and the receiver doesn’t need to know the structure of the chain.
* Dynamic Chain: The chain can be modified dynamically at runtime, allowing for flexibility in adding or removing handlers without affecting the client code.
* Single Responsibility Principle: Each handler in the chain has a single responsibility, either handling the request or passing it to the next handler, which helps in maintaining a clean and modular design.
* Sequential Order: Requests are processed sequentially along the chain, ensuring that each request is handled in a predefined order.
* Fallback Mechanism: The chain can include a mechanism to handle requests that are not handled by any handler in the chain, providing a fallback or default behavior.
* Variants: The pattern has variants like a linear chain, where each handler has a single successor, or a tree-like structure, where a handler can have multiple successors, allowing for more complex processing logic.
* Enhanced Flexibility: The pattern allows for enhanced flexibility in handling requests, as the chain can be configured or modified to suit different requirements without changing the client code.

**Real-World Analogy of the Chain Of Responsibility Design Pattern**

*Imagine a customer service department with multiple levels of support staff, each responsible for handling different types of customer inquiries based on their complexity. The chain of responsibility can be illustrated as follows:*

* **Level 1 Support**:
  + This represents the first point of contact for customer inquiries. Level 1 support staff handle basic inquiries and provide general assistance. If they cannot resolve the issue, they escalate it to Level 2 support.
* **Level 2 Support**:
  + This level consists of more experienced support staff who can handle more complex issues that Level 1 support cannot resolve. If Level 2 support cannot resolve the issue, they escalate it to Level 3 support.
* **Level 3 Support**:
  + This is the highest level of support, consisting of senior or specialized staff who can handle critical or highly technical issues. If Level 3 support cannot resolve the issue, they may involve other departments or experts within the organization.





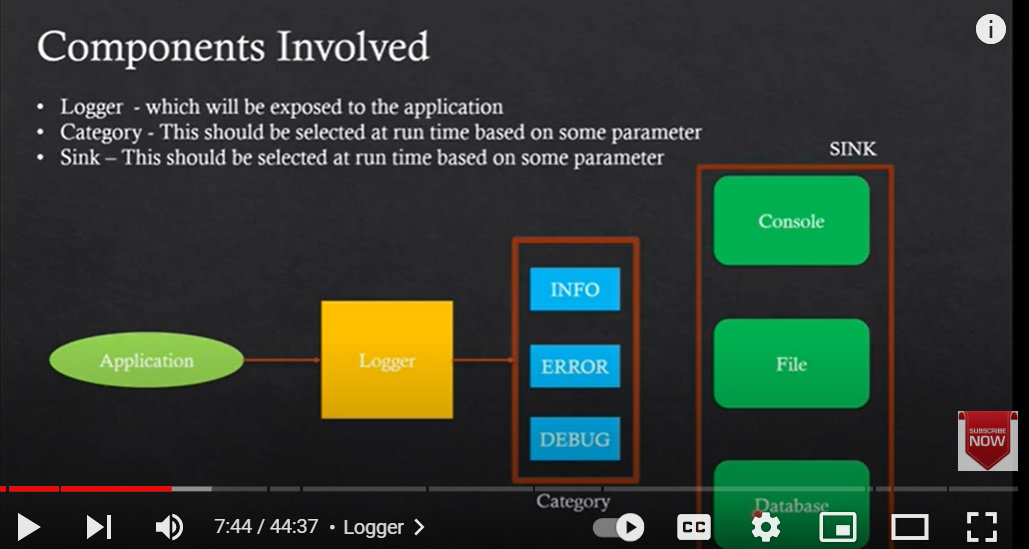
**Logging Design Pattern**

**Requirement**

1. should be able to login more than one place console, log file and database  
2. Should be able to log multiple category of messages like INFO, DEBUG, ERROR etc  
3. Category and place of logging should be configurable

**Component**

1. **Logger—Which will exposed to the application**
2. **Category- This should be selected based at run time based on the category**
3. **Sink – This should be selected at run time based on some parameter**

****

**Design Pattern will be used**

1. **Singleton Design pattern**

**The Logger file used as Singleton design pattern because we want it will create a single instance and all the logs will be add in one text file on sink component. If we will not make it as singleton, then Logger will have multiple instances and multiples files will be creating.**

1. **Chain of Responsibility Design Pattern**

**Category component have the multiple categories. It will work as if INFO is not able to do, then go to ERROR and if ERROR is not able to do then go to DEBUG.**

1. **Observe Design Pattern**

**Since Sink will be dependent on the Category like which category we have picked to write log. It means it observing the Category component.**