**Chain of Responsibility Pattern: Customer Support System**

**Description:** The **Chain of Responsibility Pattern** is a behavioral design pattern that enables a request to be passed along a chain of handlers until one of them processes it. This pattern is used to decouple the sender of a request from its receivers and allows multiple objects to handle the request, passing it along the chain if the current handler is unable to process it.

**Benefits:**

1. **Decoupling:**  
   The sender of a request does not need to know which handler will process it. This decouples the sender from the specific handler, making the system more flexible.
2. **Flexibility:**  
   Handlers can be dynamically added or removed from the chain without modifying the client code, allowing easy changes and extensions.
3. **Single Responsibility Principle:**  
   Each handler has a specific responsibility, aligning with the Single Responsibility Principle. This makes the code easier to manage and understand.
4. **Improved Maintainability:**  
   Changes to the handling logic can be made in individual handlers, simplifying maintenance and updates.
5. **Separation of Concerns:**  
   Different levels of support are managed by separate handler classes, each focusing on a specific type of issue.

**Real-Life Example: Customer Support System**

In a customer support system, issues are often categorized into different levels: Basic, Technical, and Managerial support. Each support level is represented by a different handler:

* **BasicSupportHandler:** Handles simple, routine issues that can be resolved at the basic support level.
* **TechnicalSupportHandler:** Deals with more complex technical problems that require specialized knowledge.
* **ManagerSupportHandler:** Manages issues that need escalation to managerial levels for resolution.

**How It Works:**

1. **Issue Handling:** When a customer reports an issue, it is passed to the BasicSupportHandler.
2. **Chain Processing:** If the BasicSupportHandler cannot resolve the issue, it forwards the request to the TechnicalSupportHandler.
3. **Escalation:** If the TechnicalSupportHandler also cannot resolve the issue, it is passed to the ManagerSupportHandler.
4. **Resolution:** Each handler in the chain either resolves the issue or passes it to the next handler, ensuring that the request is managed by the appropriate level of support.