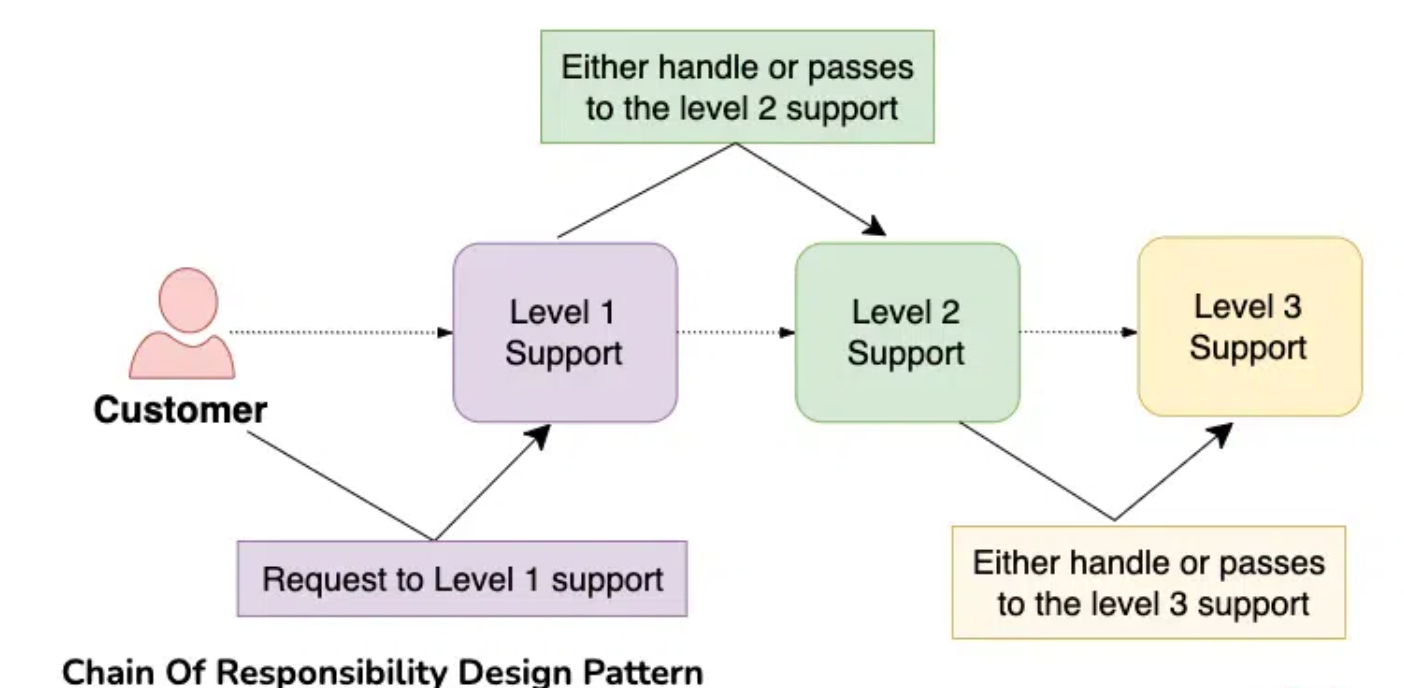
Chain of Responsibility Pattern or Chain of Responsibility Method is a [Behavioral Design Pattern](https://www.geeksforgeeks.org/behavioral-design-patterns/),

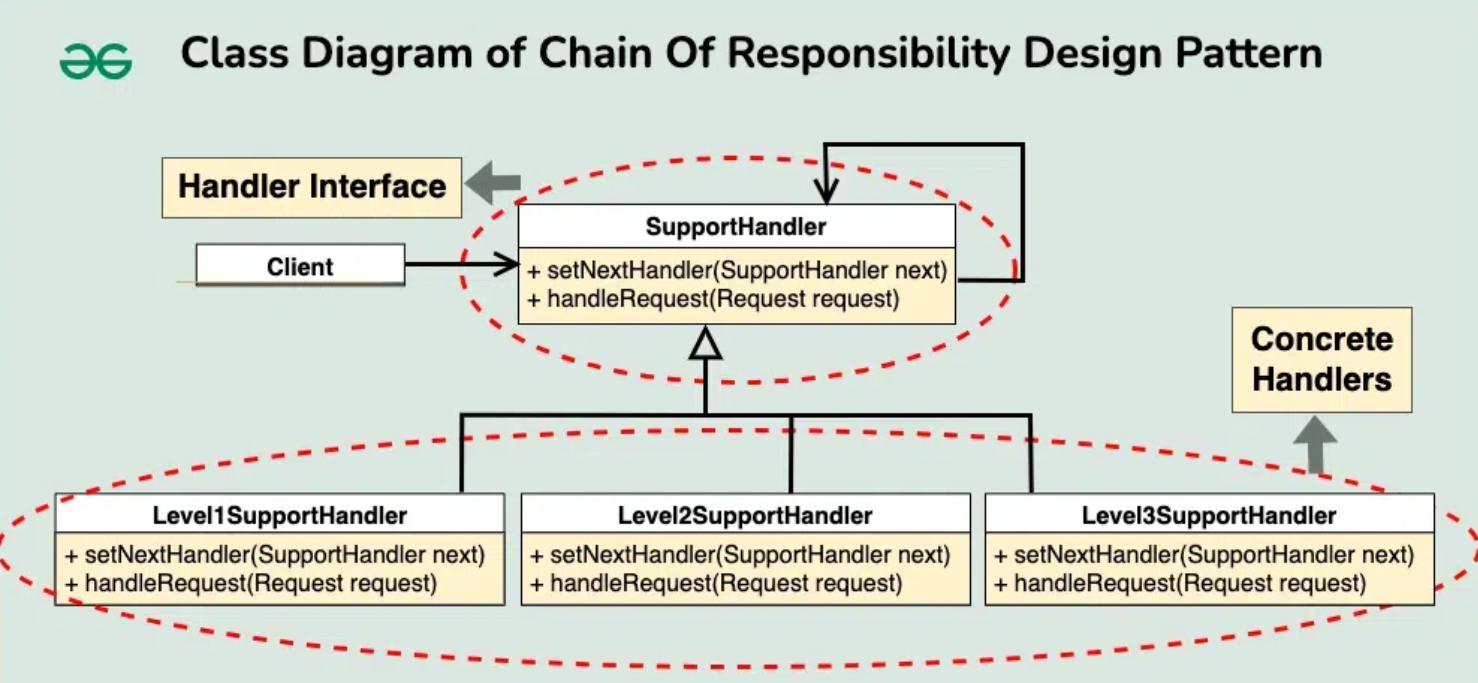
which allows an object to send a request to other objects without knowing who is going to handle it.

* This pattern is frequently used in the chain of multiple objects,

where each object either handles the request or passes it on to the next object in the chain if it is unable to handle that request.

* This pattern encourages loose coupling between sender and receiver, providing freedom in handling the request.





*// Handler Interface  
interface SupportHandler* {  
 *void* handleRequest(Request request);  
 *void* setNextHandler(*SupportHandler* nextHandler);  
}  
  
*// Concrete Handlers  
class* Level1SupportHandler *implements SupportHandler* {  
 *private SupportHandler* nextHandler;  
  
 *public void* setNextHandler(*SupportHandler* nextHandler) {  
 *this*.nextHandler = nextHandler;  
 }  
  
 *public void* handleRequest(Request request) {  
 *if* (request.getPriority() == Priority.*BASIC*) {  
 System.*out*.println("Level 1 Support handled the request.");  
 } *else if* (nextHandler != *null*) {  
 nextHandler.handleRequest(request);  
 }  
 }  
}  
  
*class* Level2SupportHandler *implements SupportHandler* {  
 *private SupportHandler* nextHandler;  
  
 *public void* setNextHandler(*SupportHandler* nextHandler) {  
 *this*.nextHandler = nextHandler;  
 }  
  
 *public void* handleRequest(Request request) {  
 *if* (request.getPriority() == Priority.*INTERMEDIATE*) {  
 System.*out*.println("Level 2 Support handled the request.");  
 } *else if* (nextHandler != *null*) {  
 nextHandler.handleRequest(request);  
 }  
 }  
}  
  
*class* Level3SupportHandler *implements SupportHandler* {  
 *public void* handleRequest(Request request) {  
 *if* (request.getPriority() == Priority.*CRITICAL*) {  
 System.*out*.println("Level 3 Support handled the request.");  
 } *else* {  
 System.*out*.println("Request cannot be handled.");  
 }  
 }  
  
 *public void* setNextHandler(*SupportHandler* nextHandler) {  
 *// No next handler for Level 3* }  
}  
  
*// Request Class  
class* Request {  
 *private* Priority priority;  
  
 *public* Request(Priority priority) {  
 *this*.priority = priority;  
 }  
  
 *public* Priority getPriority() {  
 *return* priority;  
 }  
}  
  
*// Priority Enum  
enum* Priority {  
 *BASIC*, *INTERMEDIATE*, *CRITICAL*}  
  
*// Main Class  
public class* ChainOfResponsibility {  
 *public static void* main(String[] args) {  
 *SupportHandler* level1Handler = *new* Level1SupportHandler();  
 *SupportHandler* level2Handler = *new* Level2SupportHandler();  
 *SupportHandler* level3Handler = *new* Level3SupportHandler();  
  
 level1Handler.setNextHandler(level2Handler);  
 level2Handler.setNextHandler(level3Handler);  
  
 Request request1 = *new* Request(Priority.*BASIC*);  
 Request request2 = *new* Request(Priority.*INTERMEDIATE*);  
 Request request3 = *new* Request(Priority.*CRITICAL*);  
  
 level1Handler.handleRequest(request1);  
 level1Handler.handleRequest(request2);  
 level1Handler.handleRequest(request3);  
 }  
}

Level 1 Support handled the request.

Level 2 Support handled the request.

Level 3 Support handled the request.