如果一个对象不能处理该请求，那么它会把相同的请求传给下一个接收者，依此类推。

**典型：双亲委派模式（）**

public abstract class AbstractLogger {

public static int INFO = 1;

public static int DEBUG = 2;

public static int ERROR = 3;

protected int level;

//责任链中的下一个元素

protected AbstractLogger nextLogger;

public void setNextLogger(AbstractLogger nextLogger){

this.nextLogger = nextLogger;

}

public void logMessage(int level, String message){

if(this.level <= level){

write(message);

}

if(nextLogger !=null){

nextLogger.logMessage(level, message);

}

}

abstract protected void write(String message);

}

*ConsoleLogger.java*

public class ConsoleLogger extends AbstractLogger {

public ConsoleLogger(int level){

this.level = level;

}

@Override

protected void write(String message) {

System.out.println("Standard Console::Logger: " + message);

}

}

*ErrorLogger.java*

public class ErrorLogger extends AbstractLogger {

public ErrorLogger(int level){

this.level = level;

}

@Override

protected void write(String message) {

System.out.println("Error Console::Logger: " + message);

}

}

*FileLogger.java*

public class FileLogger extends AbstractLogger {

public FileLogger(int level){

this.level = level;

}

@Override

protected void write(String message) {

System.out.println("File::Logger: " + message);

}

}。

*ChainPatternDemo.java*

public class ChainPatternDemo {

private static AbstractLogger getChainOfLoggers(){

AbstractLogger errorLogger = new ErrorLogger(AbstractLogger.ERROR);

AbstractLogger fileLogger = new FileLogger(AbstractLogger.DEBUG);

AbstractLogger consoleLogger = new ConsoleLogger(AbstractLogger.INFO);

errorLogger.setNextLogger(fileLogger);

fileLogger.setNextLogger(consoleLogger);

return errorLogger;

}

public static void main(String[] args) {

AbstractLogger loggerChain = getChainOfLoggers();

loggerChain.logMessage(AbstractLogger.INFO,

"This is an information.");

loggerChain.logMessage(AbstractLogger.DEBUG,

"This is an debug level information.");

loggerChain.logMessage(AbstractLogger.ERROR,

"This is an error information.");

}

}