BROCKHAUS GmbH
Member of BROCKHAUS Group

Lab Description

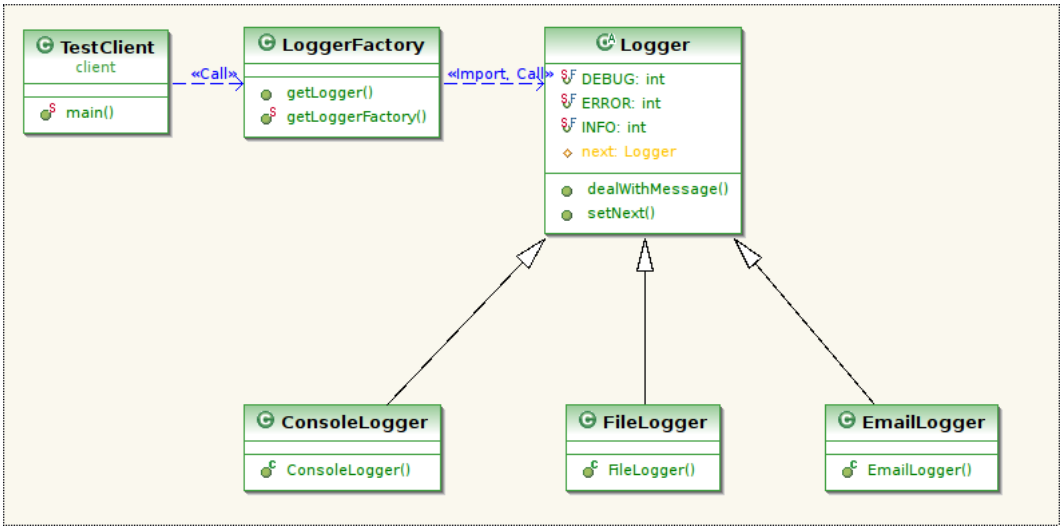
Chain of Responsibility

Overview	
Decorator was a nice one but it was - so to say – all or nothing. Once you are in the chain , you can't get out. This one is different, wherever something defined occurs the chain will be left.	
Objectives	Learn how to implement chain
Deployments / Builds	---
What's new	Chaining?
Aproximate time	10 min.

Tasklist

You know Log4J? Ever wondered how this works, you provide messages of defined severity and magically all the things with lower severity is printed to an appender.

This example provides an idea (and is no guide to write your own framework).



✗ **The super class: Logger**

The idea to provide an absract superclass sounds familiar, we used it in the decorator lab.

So we will do much the same now, check Logger.java and put an eye on:

a) The constructor which (funny the class is abstract, isn't it)

- defines the actual level
- forces subclasses to have a contructor of the same kind

b) the final method which deals with message handling

BROCKHAUS GmbH
Member of BROCKHAUS Group

Lab Description

A concrete Logger: ConsoleLogger

Do what you have been taught, follow the TODOs for all Loggers

How to get a Logger: LoggerFactory

Well, now we will provide a Logger to the requestor. Follow the TODOs