Exceptional Exception Handling

Scenario

- Your company is in charge of creating different software products. Some products are rather safety critical, some can crash without consequence, and others are somewhere in between. Your company is tired of having to rewrite exception handling for different pieces of software and changing this during the different phases of development. Ideally, a system can be put in place which allows developers to create a specific object to handle exceptions, and allows this object to be constructed dynamically to handle different exception levels dynamically.

Problem

Your task is to develop a set of classes to handle BetterException objects. The classes can be structured in any way, but must implement the method

void handle(BetterException e)

For this problem, there are three ways to handle exceptions. Exceptions may simply be printed on the screen, exceptions may be printed to a file, or exceptions may be logged to an exception database. Different programs should assign the handling of exceptions differently, based on the safety-critical nature of the particular program.

A TestCode class is provided to demonstrate your exception handler. You should only need to modify the constructor, creating an instance of BetterExceptionHandler (which you will define). The better exception handler can just print out the exception to the screen, indicating how the exception will be handled. For example:

“Handling Exception by printing to screen: LOW ERROR: Low-level error”
“Handling Exception by writing to file: MODERATE ERROR: Moderate-level error”
“Handling Exception by logging to database: LOW ERROR: Low-level error”

Make sure to use the toString() method of the exception to get the exception level and error message.

For this example the TestCode should handle exceptions in the following way:

- WARNINGS and LOW level errors should be printed to the screen
- MODERATE level errors should be written to a file
- HIGH and CRITICAL errors should be written to the database

Your solution should be dynamic enough the different means of exception handling can be easily added. You do not need to be concerned with creation of the overall exception handling system--for this task, you can simply use constructors and add methods as you see fit.

Deliverables

1. Identify the design pattern you used to solve this problem, and the participants (i.e., the roles each class takes).
2. An implementation in a language of your choice.
3. A class diagram of your solution (including existing classes), so future developers can easily see how to work with your solution.