**Name:** Diego Páez

**NRC:** 2963

**Summary Clean Code**

Chapter 7: Error Handling

The captain tells us about the advantages and disadvantages of processing errors in the code, there is a problem that the approaches confuse the caller. Unfortunately, it is easy to forget. For this reason, it is better to throw an exception when it encounters an error. The calling code is cleaner. Its logic is not obscured by error handling.

The use of try-catch-finally, it is important since we can execute the try and cancel it at any time and resume the catch, this helps define what the user expects from the code and show the error when the catch is being executed

Since the purpose of the exceptions is to allow you to handle errors remotely, it is a pity that the verified exceptions break the encapsulation in this way. Marked exceptions can sometimes be useful if you are writing a critical library: you must detect them. But, in general, application development dependency costs outweigh the benefits.

There are many ways to classify errors. We can classify them by their source: Did they come from one component or another? Or their type: Are they device failures, network failures, or programming errors? However, when we deﬁne exception classes in an application, our most important concern should be how they are caught.

It should be borne in mind that any exception thrown should provide sufficient context to determine the origin and location of an error, for which informational error messages must be created and pass them along with their exceptions, it is also necessary to mention the operation that failed and the type of failure.