Clean Code:

A Handbook of Agile Software Craftsmanship

Chapter 08: Boundaries

Third Party Code

So many times we use codes and instructions that are known as third party code. The codes that we did not create; Like a library or someone else's code.

Problem 1: We need to learn to work with the API's instructions and it is really difficult.

Problem 2: We may rely too much on APIs in our program. If some major features of the API change, then we face too many errors in our code and will have to change it.

Problem 3: This API may present more capabilities than we need.

We should decline some boundaries to keep our code from getting dependent to the API.

Solution To Problem 1: Learning Tests

To work with third party code, we should learn it; which is difficult task to do. One way is to read the documentations of the API and consume a large amount of time. We could google it and search about its methods, and that is only if we used a famous library or framework in our code. But the best way is to write learning tests. It is fast, not expensive and safe.

We call the methods and properties that we want to learn about as we expect from their functionality to check the API. This kind of test does not test if the API works properly. It concentrates on what we want from it.

Learning tests have no expense. We had to learn the API in one way or another. By writing tests we did the best thing, because by releasing the new version of the API we could run all the tests again to see if something is changed that we should care about.

Solution To Problems 2 And 3: Wrapper Module

After learning the third party code, we use it in our program.

The bad way: We use the third party code directly in all over our code. After installing the new version of the API we are forced to change all the codes related to it. Because we face a lot of errors most probably.

The good way: We write a wrapper class that uses the API and write our own method using them. This isolates the API from other parts of the software and any kind of error or conflict between different versions of the API will occur only in the wrapper class.

The Advantages Of Wrapper Class

- We can limit the capabilities of the API to be used in our program.
- Minimize the dependency to the API.
- We can change the API or library easily.
- We can test the wrapper class, but not the API itself.
- Let's imagine the API is not provided yet. We can develop other modules of our program test them, use them in the software and encounter no problems. Because there is no dependency to the API except for one class. This gives us or the team that is responsible for developing the API time.