Clean Code:

A Handbook of Agile Software Craftsmanship

Chapter 08: Boundaries

Third Party Code

Many times we use codes and instructions that are known as third-party code. The codes that we did not create ourselves; They might be libraries or code from other people.

Issue 1: We must learn to use API instructions, and this can be quite tough.

Issue 2: We might depend too heavily on APIs in our program. If the API's main parts change, we'll get many errors in our code and need to modify it.

Issue 3: This API might offer more abilities than we actually require.

To avoid our code becoming reliant on the API, we should set some boundaries. This will help keep our code independent.

Solution To Issue 1: Learning Tests

When dealing with third-party code, it's important to understand it. However, this can be a challenging task. There are a few ways to go about it. One approach involves reading the API's documentation, which can be time-consuming. Another option is to search online, especially if we're using a well-known library or framework in our code. The most effective method, though, is to create learning tests. This approach is quick, cost-effective, and secure.

In learning tests, we interact with the methods and properties we want to understand, based on our expectations of how they should function. These tests aren't meant to verify if the API is functioning correctly overall; they focus on our specific needs from it.

The great thing about learning tests is that they don't require a significant investment. We need to grasp the API's functionality one way or another. By creating tests, we're making a smart choice. When a new version of the API is released, we can rerun all the tests to check if anything has changed that we need to be aware of.

Solution To Problems 2 And 3: Wrapper Module

Once we've gained an understanding of the third-party code, we can start incorporating it into our program.

There are two approaches to consider:

The less effective approach: Using the third-party code directly all throughout our program. This can lead to problems when we update to a new version of the API. We'd be required to modify all parts of our code connected to it, likely due to encountering numerous errors.

The better approach: Creating a wrapper class that interacts with the API and constructing our own methods using it. This approach separates the API from the rest of the software. Any issues or conflicts resulting from different API versions will be confined to the wrapper class.

In essence, the second method is more advantageous as it provides better organization and control over the third-party code's integration.

Benefits Of Using A Wrapper Class

- **Controlled API Usage:** We can restrict the API's functions to only those needed for our program.
- **Reduced Dependency:** By relying on the wrapper class, we decrease how much our program relies on the API directly.
- **Easy Adaptation:** Changing the API or library becomes simpler since modifications are only confined to the wrapper class.
- **Enhanced Testing:** While we can't directly test the API itself, we can test the wrapper class effectively.
- **Forward Planning:** Picture a scenario where the API isn't available yet. We could create other modules of our program, test and use them without problems. This is because our dependency on the API is limited to just one class. This grants us and the API development team more flexibility and time for development.