Jordan Peterson

Explain Abstraction

The principle of abstraction is the hiding of complex or simple pieces of code behind much simpler calls. The key benefits to this separation though abstraction is that it allows you to keep the core of you program smaller and easier to read and understand by separating what the program does from how it does it. It also keeps groups of logic separate and more independent which helps greatly with troubleshooting, fixing, or improving that logic without needing to change the main program logic.

Abstraction is widely used for another reason, it greatly enhances code reusability. By creating generalized and modular components, abstraction promotes reusability. For instance, different file-saving methods (e.g., JSON, CSV, XML) can be implemented without altering existing code.

As an example, in my Journal project I was able to hide all of the logic for loading saved data from a file behind one line of code. This keeps the main logic short an readable.

*workingJournal.loadFromFile();*

Behind that one line was the following.

*string loadedEntries;*

*Console.Clear();*

*Console.Write("Enter the file name you want to load from: ");*

*string loadFilePath = Console.ReadLine();*

*if (File.Exists(loadFilePath))*

*{*

*loadedEntries = File.ReadAllText(loadFilePath);*

*\_entries = JsonSerializer.Deserialize<List<Entry>>(loadedEntries);*

*}*

*else*

*{*

*Console.WriteLine("File not found. No entries were loaded.");*

*}*