Abstraction is the process of turning complex ideas into simple ideas, so you focus on the what instead of the how. You do this by hiding long unnecessary code and only show the important parts in a simple interface by calling a method or function. For example in the journal program I called the journalEntry() method in the Program.cs and that came from a method in the journal class called journalEntry.Display() and the Display() came from the Entry.cs class where the code originates from (See picture bellow). Instead of focusing on the how the display program works in one long complex program we just called a simple method that just showed us the what it does.

Abstraction is important because it simplifies complex codes so the programmer can just focus on methods, classes, and APIs instead of understanding the background stuff. This principle improves productivity and reusability, which is one of the most important aspects of programming with classes. In a programmer’s world they are faced with long clunky code in large systems. Abstraction allows programmers to break it down into layers in clean interfaces, which makes developers work more efficiently.

I’m a supply chain management major and my whole career is doing abstraction but in a business world. No one cares or wants to know how a product gets to the store, all they care about is that it is there. As supply chain managers we make sure all of the complexity of a product gets to the customer in the right place, at the right price, and at the right time, so that they don’t have to worry about the complexity of it they can just worry about what to get at the store.

Here is the original complex code in the Entry.cs:   
A screen shot of a computer code

AI-generated content may be incorrect.