Clemence K Matsika

W02 Assignment: Explaining Abstraction

Abstraction means hiding complex details while showing only what's necessary. Think of it like using a car - you just need to know how to use the steering wheel, pedals, and gear shift, not how the engine works inside.

In programming, abstraction lets us simplify complicated systems by focusing on what things do rather than how they do it. This makes our code easier to understand and maintain.

A major benefit is that it reduces complexity. When code gets large, abstraction helps us manage it by breaking it into understandable chunks with clear purposes.

In the Journal program assignment I did, the Journal class demonstrates abstraction perfectly:

public void AddEntry(Entry newEntry)

{

\_entries.Add(newEntry);

}

public void SaveToFile(string filename)

{

using (StreamWriter outputFile = new StreamWriter(filename))

{

foreach (Entry entry in \_entries)

{

outputFile.WriteLine($"{entry.\_date}~|~{entry.\_promptText}~|~{entry.\_entryText}");

}

}

}

Here, AddEntry() hides the details of how entries are stored. The user doesn't need to know we're using a List - they just call journal.AddEntry(myEntry). Similarly, SaveToFile() handles all the complex file writing operations. Users don't need to understand how entries are formatted or written to disk.

This abstraction makes the program more adaptable. If we later change how entries are stored (maybe to a database), the code that uses these methods won't need to change at all.