**Abstraction**

Abstraction is where complicated ideas are turned into simple ones. Non-Relevant characteristics are removed until only necessary information remains. It allows software developers to focus on the higher level, core functionality required in projects, without having to be bogged down in the intricacies of lower-level components.

One benefit of Abstraction is that it allows code to be reused and maintained better. Code is broken down into simple, reusable components. Because of this streamlined and simple process developers can make changes to key parts of the project without unintentionally affecting the rest of it. This allows developers to be more flexible and reduces complexity.

Abstraction is applied and used in Object Oriented Programming, through classes. In my Journal Entry Program this is seen and applied in classes like ‘Journal’, ‘Entry’, and ‘PromptGenerator’. Each of these classes have simple methods like AddEntry, DisplayAll and SavetoFile which is abstracted because when someone calls the addentry or savetofile, they don’t need to understand how the list works or how the file saving is handled, they just use the provided methods. This makes the program easer to use and modify in the future, because you can change the other aspects of the Journal Program without affecting other parts of the code.

Example:

public class Journal

{

public List<Entry> \_entries = new List<Entry>();

public void AddEntry(Entry newEntry)

{

\_entries.Add(newEntry);

}

public void SaveToFile(string file)

{

using (StreamWriter outputFile = new StreamWriter(file))

{

foreach (Entry entry in \_entries)

{

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

}

}

}

}