Explain Abstraction

Abstract is the concept that allows you to hide the type of implementation (which isn't necessary) and show only what'sneeded to use a class or system. This allows developers to concentrate on what a component does, as opposed to how it internally processes things. This is important because it simplifies matters, makes code easier to understand and avoidsthe mixing of different parts in a whole program.

A significant advantage of abstracting is maintainability. Since you hide details behind well defined methods, you can change things inside of a class without changining the program elsewhere. This results in less smelly code, fewer bugsand ease of updates or extensions.

Rather, the abstraction is provided by the Journal class in my program. The Program class doesn't deal directly with lists of entries or file operations. Instead, it simply hold references to basic, meaningful operations like AddEntry and SaveToFile without caring about how they are carried out.

That code is shown to be abstract because the logic required to keep and save entries is kept inside the Journalclass. This makes the program less cluttered and easier to read, edit, and expand, for example making it possible to later change the file format without having to rewrite that logic in main_menu().

Example:

```
Journal journal = new Journal();

Entry entry = new Entry

{

   Date = DateTime.Now.ToShortDateString(),

  Prompt = prompt,

Response = response

};

journal.AddEntry(entry);

journal.SaveToFile(filename);
```