That's a very common and effective architectural change! A Menu class is usually responsible for handling user interactions and navigation, which fits well with the *Display* and *GetUserInput* ideas you had.

Here is the updated consolidated class diagram information, replacing the PromptBank with a Menu class and making adjustments to the methods:

Consolidated Class Diagram (with Menu)

Class: Journal (No Change)

Attribute	Type/Description
entries	A **List/Array of Entry** objects.

Method	Parameters	Description
Display()	None	Displays all entries in the journal.
Write()	None	Creates a new entry (calls Entry.WriteEntry()).
Save()	**string** (filename)	Saves the current journal to a file.
Load()	**string** (filename)	Loads a journal from a file.

Class: Entry (Minor Change)

Attribute	Type/Description
Date	A **Date/DateTime** type.
Prompt	**const string**
Response	**string**

Method	Parameters	Description
Display()	None	Displays the date, prompt, and response for this single entry.
GetPrompt()	None	Generates a random prompt based on a random number generator
WriteEntry()	None	A method to guide the user through creating the entry (getting the prompt, getting the response, setting the date). **It will need a way to access the prompts.**

Class: Menu (Replaced PromptBank)

Attribute	Type/Description
options	**List/Array of strings** (The main menu options: Write, Display, Save, Load, Quit).

Method	Parameters	Description
DisplayMenu()	None	Displays the main menu options to the user.
GetUserInput()	None	**Returns a string/integer** representing the user's menu choice.

Relationships

- **Journal has a list of Entry** objects (Composition/Aggregation).
- The **main program logic (e.g., in main)** will **use the Menu** to interact with the **Journal** (Dependency).
- The **Entry** class (specifically its WriteEntry method) now needs to **use the Menu** (or the Menu's GetRandomPrompt method) to retrieve a prompt (Dependency).