1 - Meaning of Abstraction

Abstraction in the context of object-oriented software development means highlighting the fundamental aspects and hiding implementation details that are not relevant to the end user.

2- Benefit of Abstraction

Abstraction has several benefits, one of which is that it makes maintenance easier.

Because internal details are hidden, it is possible to modify and update internal behavior without having to change other components of the system that use the component being modified.

In other words, internal changes in one component will not require changes in components external to it.

3- Provide an application of Abstraction.

Suppose a class that calculates the taxes to be paid by the user on a retail purchase.

Suppose that the tax rates have changed due to revisions in government rules.

Abstraction, by isolating within the class the real way in which the tax amount is calculated, allows this calculation to be modified internally, without the external components feeling this change.

Another example, suppose that a certain component saves the data of a certain program in a csv file. Abstraction allows you to change this component so that it saves in a database, without modifying anything in the components external to it.

In other words, the user does not need to worry about the internal details that make the component work, just understand how it works in general.

For example, the Console.WriteLine command is used to send text to the console (Terminal). The user does not need to know anything about the internal code that makes this method work in order to be able to use it and write a message to the console, these details are hidden from the user. The user just needs to pass his message to this method as in the following example Console.WriteLine("Sample text to the console"), which in the terminal will be printed "Sample text to the console".

4- Code example of Abstraction

In the Program.cs file, if the user presses option 2 in the main menu,

The code will be taken to the following section:

else if (option == 2)

{

// Option equal to 2 means that the user wants to display his diary records on the screen.

j.display(qlist);

}

In this code, j is a variable of type Entry, which is responsible for the user's diary records.

When the user chooses this feature, the program executes the following more complex code:

public Dictionary<string,string> \_journalRecordList = new Dictionary<string, string>();

public void display(PromptGenerator *qlist*)

{

//This method takes all the Entries(Records) from the Journal and displays

//them on the screen separated one by one by empty lines between them.

string[] part;

string recordKeyData;

string question;

if (\_journalRecordList.Count !=0)

{

foreach (*KeyValuePair*<string, string> input in \_journalRecordList)

{

//The keys will be formatted as follows: data-questionNumber.

recordKeyData = input.Key;

part = recordKeyData.Split("-");

question = *qlist*.getQuestion(int.Parse(part[1]));

//Prints the records found in the Journal

Console.WriteLine("");

Console.WriteLine($"Date: {part[0]} - Prompt: {question}");

Console.WriteLine($"{input.Value}");

Console.WriteLine("");

}

}

else

{

Console.WriteLine("The Journal is empty.");

}

}

Note that the abstraction hides the complex code needed to display the user's Journal data on the screen. All you need to do is instantiate the class and use the display method, passing the correct parameter.

The developer who uses this class can abstract this code and simply instantiate the class and call the display method.