Andrew Halisky

CSE 210: Programming with Classes

Brother Poulson

June 14, 2023

For our project this week our team came up with different class attributes for the Mindfulness Project. This project will contain a main class to call each of the three activity classes, and an activity class to call each method in those classes. The breathing activity will allow the user to enter a custom range of time in seconds. Then it alternates the breathing method, allowing the user to breathe in and out for 5 seconds each until the entire duration has stopped. The reflection activity will be relatively similar to the breathing activity. Instead of using the breathing method, it will give the user a random question about a time in their life, then allow them to write as many answers as they can from the range of time that they entered in.

For the listing activity, the user will enter a specific time range again. This time, the program will prompt a user to list as many responses as they can to different questions within the range of time entered within the program. It will then display each of those entries. It will then ask the user if they want to play again or go back to the main menu. When the user types “quit”, it will then completely exit the program.

The program will be written with the following class attributes:

* In Program.cs
  + In the main program, there will be a loop written as a switch method that contains four different cases. Three of those cases will contain a method that will call each constructor from each activity class. The fourth case will exit the program. This is where the EndLoop() will occur, ending the entire program.
* In Activity.cs  
  + Will add the public class Activity:
    - Will contain two strings: \_startMessage and \_endMessage to call and return different start and end messages for each activity class.
    - Will contain the integer \_duration. This will get the total duration for the loading process of the program.
    - Will contain the \_description string for each activity class. This will call and return a regular description about each activity.
    - Will add the SetDuration() constructor which will set the entire duration for each activity.
    - Will add the GetDuration() constructor which will get the duration that was set.
    - Will add the Pause() constructor which will contain a code that will pause the program.
* In BreathingActivity.cs
  + Will add the public class Breathe Activity:
    - Will add the Breathe() constructor. This will contain a Console.Write() getting the custom time entered by the user.
    - Will add a Console.WriteLine() for breathe in and breathe out.
    - Will add the GetCountdown() to add a countdown for 5 seconds in between each breathe in and out.
* In ListingActivity.cs
  + Will add the public class Listing Activity
    - Will add the \_prompt list <List> which will contain the list of prompts for the Listing activity.
    - Will add the \_count string which will count the number of answers listed for each prompt?
    - Will add the CountInput() constructor which will count the number of times the user answered the generate question.
    - Will add the SetPromptList which will allow the user to enter as many prompts as possible during the custom duration of the program.
    - Will add the GetPrompt() which will get a random prompt from the list of prompts.
* In ReflectionActivity.cs
  + Will add the public class Reflection Activity
    - Will add the \_prompt list <List> which will contain the list of events on experiences and questions with regards to that experience for the Reflection activity.
    - Will add the SetPromptList() constructor which will allow the user to enter as many prompts necessary relating to reflections on those events.
    - Will add the NewPrompt() constructor which will get a new event from the list and generate a random one at a certain time. It will then generate a random question afterwards.

Here is a diagram of the project with each class attached below: A picture containing text, diagram, screenshot, parallel

Description automatically generated