**W05 Team Activity: Mindfulness Design**

**1. Class Structure**

The program will be designed using **inheritance**. We'll define a **base class** for shared functionality and **four derived classes** for each activity type.

**Base Class: Activity**

* Stores shared attributes and behaviors for all activities.
* Contains common methods for displaying messages and handling timing.
* Tracks the number of times each activity has been performed.

**Derived Classes:**

1. **BreathingActivity** – Guides the user through deep breathing exercises.
2. **ReflectionActivity** – Prompts the user to reflect on meaningful experiences.
3. **ListingActivity** – Encourages the user to list positive aspects in their life.
4. **GratitudeActivity** (New) – Guides the user in writing down things they are grateful for.

**2. Class Behaviors (Methods)**

**Base Class (Activity)**

* StartActivity(): Displays the starting message, description, and duration prompt.
* EndActivity(): Displays the completion message and summary.
* ShowSpinner(int seconds): Displays an animation (spinner or countdown) during pauses.
* GetDuration(): Asks the user for activity duration and stores it.
* IncrementActivityCount(): Increases the count for each activity performed.

**Derived Classes**

* **BreathingActivity**
  + RunBreathing(): Alternates between "Breathe in..." and "Breathe out..." messages with an enhanced animation.
* **ReflectionActivity**
  + GetRandomPrompt(): Selects a random reflection prompt.
  + RunReflection(): Displays the prompt, then asks follow-up reflection questions with pauses.
* **ListingActivity**
  + GetRandomPrompt(): Selects a random listing prompt.
  + RunListing(): Starts a timer, allowing the user to input as many responses as possible.
  + DisplayResults(): Shows the total number of responses entered.
* **GratitudeActivity** (New)
  + GetRandomPrompt(): Selects a random gratitude prompt.
  + RunGratitude(): Asks the user to list things they are grateful for and stores responses.

**3. Class Attributes (Member Variables)**

**Activity (Base Class)**

* protected string activityName; // Stores activity title
* protected string description; // Brief activity description
* protected int duration; // Stores user-selected duration
* private static Dictionary<string, int> activityLog; // Tracks activity counts

**BreathingActivity**

* No additional attributes needed (inherits duration from Activity)

**ReflectionActivity**

* private List<string> prompts; // Stores reflection prompts
* private List<string> questions; // Stores follow-up reflection questions

**ListingActivity**

* private List<string> prompts; // Stores listing prompts
* private List<string> userResponses; // Stores user input during the activity

**GratitudeActivity (New)**

* private List<string> prompts; // Stores gratitude prompts
* private List<string> userResponses; // Stores user responses

**4. Constructor Definitions**

Each class will have a constructor to initialize attributes:

**Activity (Base Class)**

public Activity(string name, string desc) {

activityName = name;

description = desc;

if (activityLog == null) {

activityLog = new Dictionary<string, int>();

}

}

**BreathingActivity**

public BreathingActivity() : base("Breathing Activity", "This activity will help you relax by guiding your breathing.") {}

**ReflectionActivity**

public ReflectionActivity() : base("Reflection Activity", "This activity will help you reflect on meaningful moments.") {

prompts = new List<string>() { "Think of a time when you stood up for someone else.", ... };

questions = new List<string>() { "Why was this experience meaningful to you?", ... };

}

**ListingActivity**

public ListingActivity() : base("Listing Activity", "This activity will help you reflect on the good things in life.") {

prompts = new List<string>() { "Who are people that you appreciate?", ... };

}

**GratitudeActivity (New)**

public GratitudeActivity() : base("Gratitude Activity", "This activity helps you appreciate things in your life.") {

prompts = new List<string>() { "What are three things you're grateful for today?", ... };

}

**5. Basic Code Stubs (C#)**

Each class should be placed in its own file:

**Activity.cs**

public class Activity {

protected string activityName;

protected string description;

protected int duration;

private static Dictionary<string, int> activityLog = new Dictionary<string, int>();

public Activity(string name, string desc) {

activityName = name;

description = desc;

}

public void StartActivity() {

Console.WriteLine($"Starting {activityName}...\n{description}");

Console.Write("Enter duration (seconds): ");

duration = int.Parse(Console.ReadLine());

IncrementActivityCount();

}

public void EndActivity() {

Console.WriteLine("Good job! You completed " + activityName + " for " + duration + " seconds.");

}

public void IncrementActivityCount() {

if (!activityLog.ContainsKey(activityName)) {

activityLog[activityName] = 0;

}

activityLog[activityName]++;

}

}

**BreathingActivity.cs**

public class BreathingActivity : Activity {

public BreathingActivity() : base("Breathing Activity", "This activity will help you relax by guiding your breathing.") {}

public void RunBreathing() {

for (int i = 0; i < duration / 2; i++) {

Console.Write("Breathe in... ");

Thread.Sleep(2000);

Console.Write("\b\b\b\b\b\b\b\b\bBreathe out... ");

Thread.Sleep(3000);

}

}

}