SF-Assessment 2

Classes & Structure

Meal Planner App

Meal Planner App

Each day consists of 4 specific meal times: breakfast, lunch, dinner, and snack.

The meals themselves are flexible, and users can choose any dish for each meal time.

The system must handle different types of meal plans, with specific rules for activation and deactivation.

Your task is to implement this app by deciding when to use **classes** or **structs**, using **inheritance**, **method overriding**, and **computed properties** to manage meals and their calorie counts.

Requirements

Meal

A **Meal** should have the following properties:

- **Name**: representing the name of the meal (e.g., “Omelet").
- **Calories**: automatically sets the calorie count based on the meal name:
    - Omelet: 250 calories
    - Salad: 150 calories
    - Pasta: 500 calories
    - Fruit: 100 calories
    - Sandwich: 300 calories

The ***calorie value should be determined based on the dish name, and users cannot directly modify the calorie count.***

Meals should ***behave independently*** when copied or modified, so that changing one meal should not affect other meal instances.

Requirements

Day

A **Day** represents a single day in the meal planner and must have the following properties:

- **Day Of The Week**: representing the day (e.g., “Monday").
- **Meal Type:** A **Meal** object **(**breakfast, lunch, dinner, snack)

Days should ***behave independently*** when copied or modified,

For example, if a user modifies Monday’s breakfast, it should not change Tuesday’s breakfast.

Requirements

MealPlan

A **MealPlan** represents a plan for a user's meals for a specific day, with the following properties:

- **Name:** representing the name of the meal plan (e.g., "Weekly Plan").
- **Day:** A **Day** object representing the day for which the meal plan applies.
- **Is Active:** indicating whether the meal plan is currently active.

The meal plan must be ***shared across users, so any changes made to one instance of the meal plan should be visible to all users sharing the plan.***

The MealPlan class must have a method called **`activate()`** that sets the plan to active and prints **`"Meal plan activated!".`**

Requirements

Standard MealPlan

StandardMealPlan is-a **MealPlan** and adds the rule that the plan can only be activated once. Once deactivated, it cannot be reactivated.

Override the **`activate()`** method to ensure that if a user tries to reactivate a deactivated plan, the message **`"This meal plan can only be activated once."`** is printed.

Requirements

Premium MealPlan

PremiumMealPlan is-a MealPlan and allows for multiple activations and deactivations.

Override the **`activate()`** method to allow the plan to be activated multiple times. Each time it is activated, print **`"Premium meal plan activated!".`**

Demonstration

- Create **`Meal`** instances for breakfast, lunch, dinner, and snack.
- Create a **`Day`** instance that uses these meals for a specific day (e.g., "Monday").
- Create instances of both **`StandardMealPlan`** and **`PremiumMealPlan`**, and demonstrate how the **`activate()`** method behaves for each.
- Modify a **`MealPlan`** and show how changes are visible to all references to the same plan.

Rubrics

Total - 15 marks

Component	Novice (1 point)	Intermediate (2 points)	Proficient (3 points)	Mastered (4 points)	Exemplary (5 points)
Class vs. Struct Usage	Attempted implementation of **Meal, Day, or MealPlan**, but did not follow instructions regarding class/struct usage.	Correctly implemented either **Meal, Day, or MealPlan** with appropriate class/ struct usage based on requirements.	Correctly implemented **two out of three types** (Meal, Day, or MealPlan) with proper class/ struct choices.	Correctly implemented **all three types (Meal, Day, and MealPlan)** with appropriate decisions on class/ struct usage.	Correctly implemented both **StandardMealPlan** and **PremiumMealPlan**, along with all previous types, demonstrating mastery of class vs. struct usage.
Component 2: Type members and Type Initialization	Properly implemented properties for all types, **except calories in Meal.**	**All properties** are implemented, including **the calories property in Meal.**	**All properties and methods** are implemented for all types, demonstrating a good understanding of their functionality.	**All properties and methods** are implemented correctly, **with appropriate naming conventions.**	**All properties and methods** are implemented correctly, with proper naming conventions and **initialized appropriately.**
Component 3: Inheritance & Method OverRiding	Partially implemented inheritance with **only one class inheriting.**	Successfully implemented inheritance with **both StandardMealPlan and PremiumMealPlan inheriting.**	Partially implemented **overridden methods with errors or incomplete functionality.**	Correct syntax in overridden methods, but they **do not behave as expected or lead to logical errors in functionality.**	Fully implemented method overriding, with all overridden methods functioning as required and **behaving correctly.**