Model Classes

1. MealCategory:

Attributes: None. This is an Enum class with values such as BREAKFAST, LUNCH, DINNER, and SNACKS.
Methods: Enums generally don't have methods unless they're utility functions, which we don't have here.
2. MenuItem:

Attributes: Includes name (String), description (String), calories (int), and category (MealCategory) to describe each food item.
Methods: addToMacroScreen() is a key method that adds the item to the nutritional tracking section of the application.
3. UserNutrition:

Attributes: Captures nutritional information such as UserEnteredCals (int) and TotalMealCals (double), along with detailed breakdowns for each meal category and nutrient type (breakfastTotalCals, lunchTotalFat, ect).
Methods: Functions for adding/removing meals, calculating total and category-specific nutrition, and updating macronutrient totals are essential here.

Controller Classes

1. MealController:
Responsibilities: Manages interactions related to meal management, such as adding, removing, and viewing meals.
Interacts With: MenuItem and UserNutrition to update nutritional data based on user actions.

2. NutritionController:
Responsibilities: Focuses on calculations and management of nutritional goals and data.
Interacts With: UserNutrition to perform calculations and provide nutritional insights.

3. RestaurantController:
Responsibilities: Oversees the menus and items from various restaurants, enabling users to browse and select meals.
Interacts With: MenuItem to manage and update restaurant-specific offerings.

Class Diagrams

Model Classes Diagram: This would show MealCategory connected to MenuItem, indicating the category of each item. MenuItem would have a unidirectional association with UserNutrition, signifying that items can be added to track nutritional intake.

Controller Classes Diagram: Illustrates MealController linked to MenuItem and UserNutrition for meal management. NutritionController is connected to UserNutrition for handling nutritional data. RestaurantController is associated with MenuItem to manage restaurant menus.

Overall Code Structure

The code follows the Model-View-Controller (MVC) architecture.

Model: Defines the data structure and business logic, shown in classes like MenuItem and UserNutrition.
View: Represents the UI components (HomeScreen, ChickfilaScreen, ect), which are not detailed here but crucial for user interaction.
Controller: Acts as the intermediary, handling user input and updating the model or view accordingly, with classes like MealController and NutritionController.

Design Patterns and Architectural Choices

Singleton Pattern: Applied to the AccessControlCenter to ensure a single access point, enhancing security.
Factory Method: Employed in the RestaurantController for creating MenuItem objects, simplifying the addition of new restaurants.