Object-Oriented Programming with C++
ENSIA 2024–2025
Tutorial 10: Polymorphism (Part 2)

Exercise 1: Ordering System for a Restaurant

Part 1: Menu Item Hierarchy

You are tasked with creating a simple ordering system for a restaurant in C++. The system should handle different types of menu items using polymorphism.

To begin, define an abstract base class named **MenuItem**. This class should encapsulate private attributes such as `name` and `price`, and provide virtual functions such as `input()`, `display()`, and `calculatePrice()`.

Next, create the following derived classes, each of which extends `MenuItem` and introduces attributes specific to its category:

- **MainCourse**: Includes attributes like `protein`, `sideDish`, and `cookingStyle`.
- **Appetizer**: Includes attributes such as `description` and a boolean `vegetarian` flag.
- **Dessert**: Includes attributes like `flavor` and `glutenFree`.
- **Drink**: Includes `type` (e.g., soda, juice), `servingSize`, and `temperature` (hot or cold).

Each subclass should override the `input()`, `display()`, and `calculatePrice()` functions.

Pricing Rules:

- **MainCourse**: Price depends on the base price, extra protein weight, and protein price per kilogram.
- **Appetizer**: Price varies based on size:
  - `S` (Small): base price
  - `L` (Large): base price + 50% of base price
  - `M` (Mega): base price + 80% of base price
- **Dessert** and **Drink**: No additional costs; the final price equals the base price.

Task 1: Implement the base class `MenuItem` and the derived classes `MainCourse`, `Appetizer`, `Dessert`, and `Drink`.

Part 2: Order Processing

To manage customer orders, define a class named **Order**. This class should include:

- A vector of pointers to `MenuItem` objects.
- A method for adding new items to the order, with interactive customization for each type.
- A method to display an invoice showing item details and the total cost.

Task 2: Implement the `Order` class, then write a `main()` function to test your implementation.

Part 3: Discount The restaurant manager has decided to apply a **2% discount** on the `protein price per kilogram` in all main courses.

Task 3: Add a member function named `applyDiscount()` to the `Order` class to implement this policy. This function should adjust the cost of applicable items and display both the discount and the updated total in the final invoice.

Hint: Use `dynamic_cast` or `typeid()` for safe downcasting when identifying objects of type `MainCourse`.