Challenge: Building a Custom Meal

Imagine you're creating a software system for a restaurant, and you want to implement a builder pattern for creating custom meals. Each meal can have multiple components like a main course, side dishes, and drinks.

Your goal is to create a flexible builder pattern that allows customers to build their meals step by step. Here are the requirements:

1. Create a `MealBuilder` struct with methods for adding components to a meal.
2. Implement builder faucets for each component type (e.g., `MainCourse`, `SideDish`, `Drink`) that allow chaining.
3. Implement functional builder methods for setting properties of each component (e.g., `WithName`, `WithSize`, `WithSauce`).
4. Ensure that a meal can have one main course, one or more side dishes, and one drink.
5. Provide a method to calculate the total cost of the meal.

Here's a simplified example of how the code might look:

```
// Implement builder methods for MealBuilder, MainCourse, SideDish, and Drink

func main() {
    // Create a custom meal using the builder pattern
    customMeal := MealBuilder().
        MainCourse().
            WithName("Steak").
            WithSize("Medium").
            WithSauce("Pepper").
        SideDish().
            WithName("Salad").
            WithSize("Small").
        Drink().
            WithName("Soda").
            WithSize("Large").
        Build()

    // Calculate and print the total cost of the meal
    totalCost := customMeal.CalculateTotalCost()
    fmt.Printf("Total cost of the meal: $%.2f\n", totalCost)
}
```