CECS 575 - Group 7

Food Ordering System

Assignment 3

Find **one application of Structural Pattern** and implement it in Java. Create a sequence and class diagram for it.

Contents

Structural Patterns

We have implemented 2 Structural Patterns:

1. Composite Pattern

The composite pattern helps to compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.

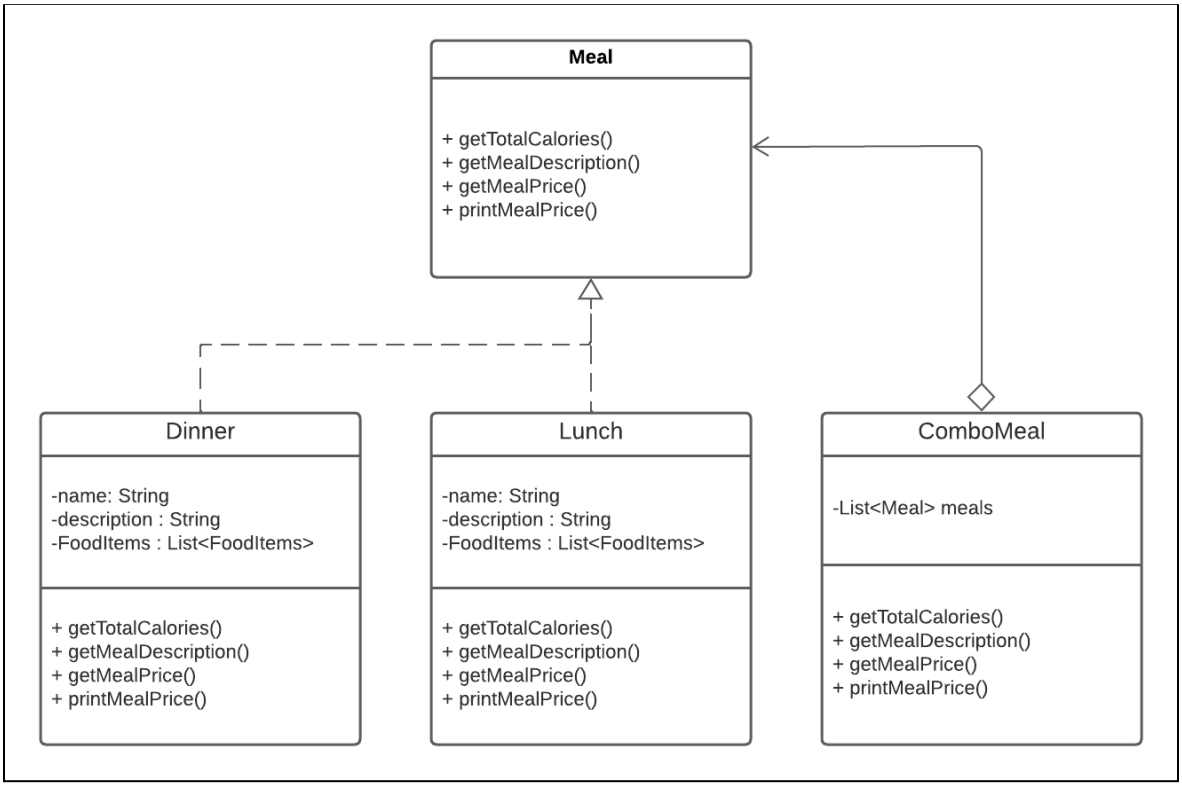
In the Food ordering System, we have used the Composite Pattern in the following way.

The Meal Interface is the Base Component which defines the set of operations that are to be carried out.

Lunch, Breakfast, Dinner are the Leaf components. These classes implement Meal, which includes the operations defined in the Meal interface.

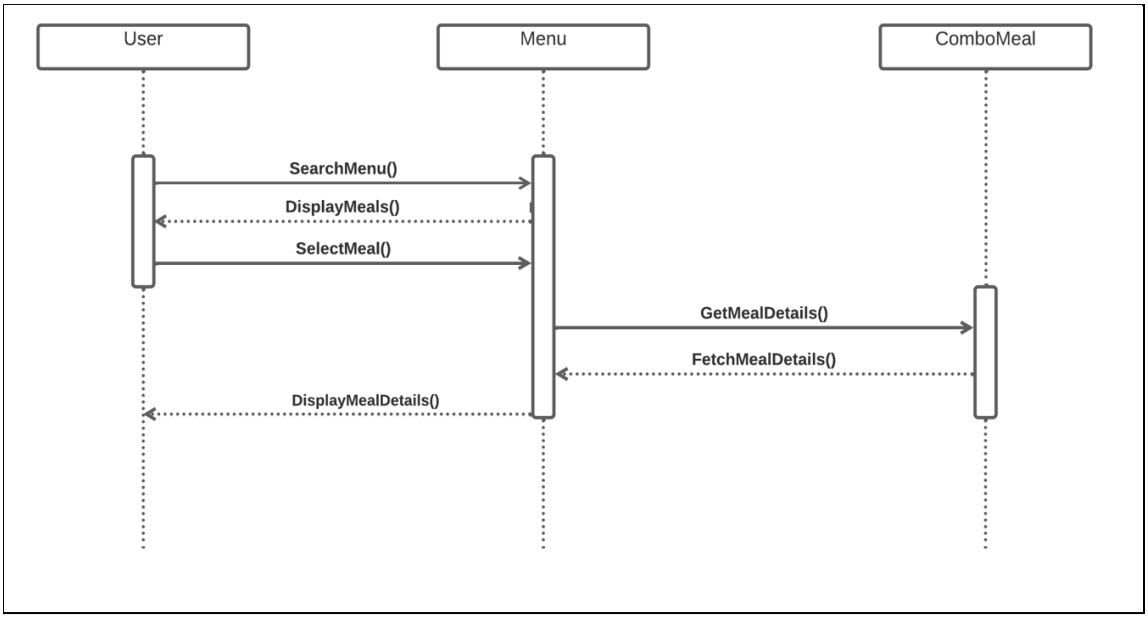
ComboMeal is the Composite Class which implements Meal interface and also includes the collection of Meal objects (Lunch / Dinner / Breakfast) including the operations like adding a combo meal.

1.1: Class Diagram



1.1: Class diagram for Composite Pattern implementation

1.2: Sequence Diagram



1.2: Sequence diagram for Composite Pattern implementation

2. Decorator Pattern

A decorator pattern allows a user to add new functionality to an existing object without altering its structure. This type of design pattern comes under a structural pattern as this pattern acts as a wrapper to the existing class. This pattern creates a decorator class that wraps the original class and provides additional functionality keeping the class methods signature intact.

2.1: Class Diagram

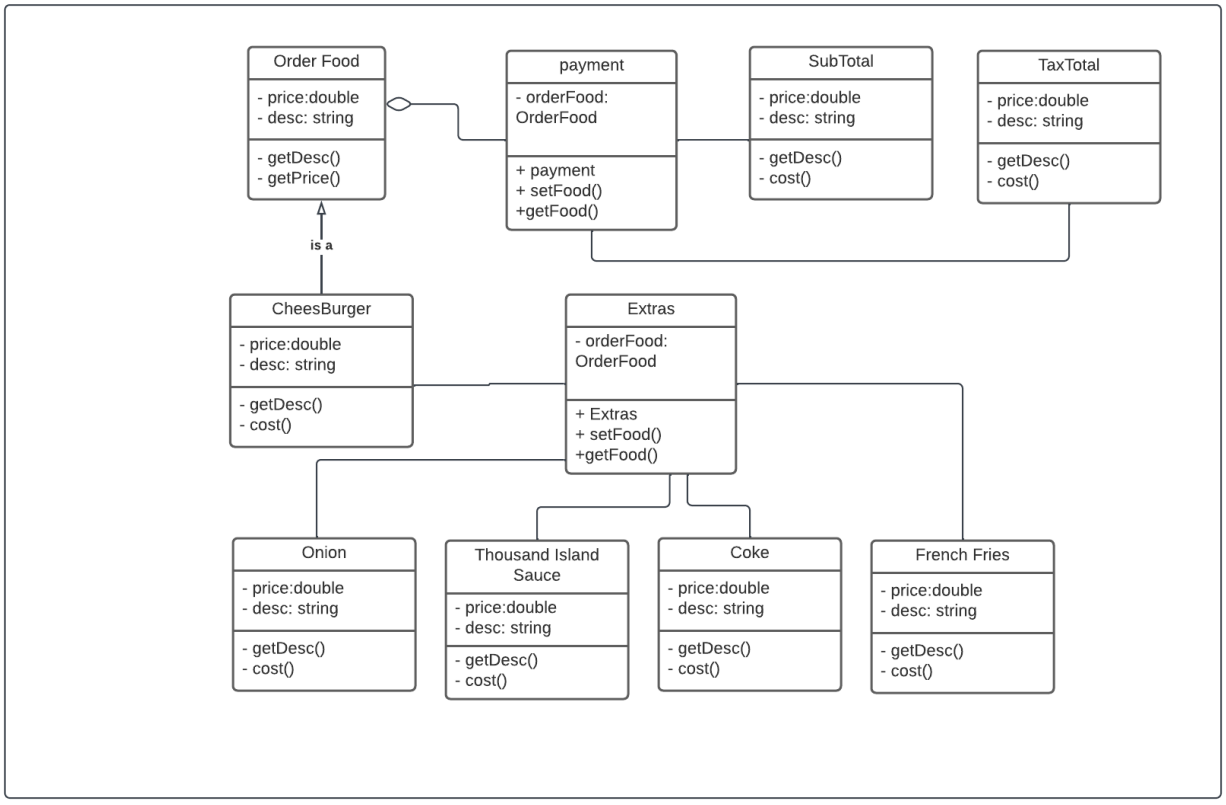


Fig 2.1: Class Diagram for decorator pattern implementation

2.2: Sequence Diagram

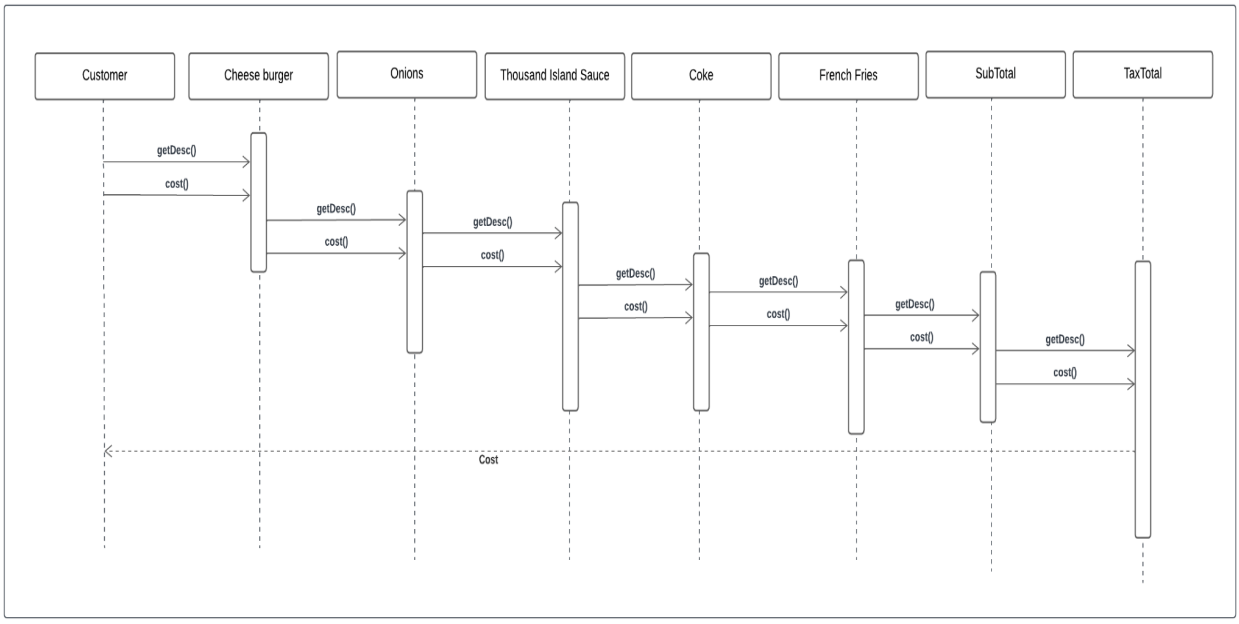


Fig 2.2: Sequence Diagram for Decorator pattern

Console Output

```
***  Hungry? Order the food you love! ***


 ---- COMPOSITE PATTERN ----

DISPLAYING THE MEALS MENU OF PARADISE RESTAURANT


Meal name: Handcrafted Breakfast for a bright day
Meal includes following items:
  - Ham Sandwhich
  - Bacon Burger
  - Scrambled Eggs
Total Calories in this meal: 300.0
Total Price: 32.64


Meal name: Handcrafted Lunch for filling meal
Meal includes following items:
  - Noodles
  - Chicken Rice
  - Tiramisu
Total Calories in this meal: 300.0
Total Price: 49.12


Meal name: Handcrafted Dinner for a delighted evening
Meal includes following items:
  - Onion Fritters
  - Soya Garlic Rice
  - Yogurt Milk
  - Mint ice cream
Total Calories in this meal: 400.0
Total Price: 22.94
```

```
 ---- DECORATOR PATTERN ----


 Begin Food Order
===================================
• Cheese Burger, with no onions : 3.0$
=========================================================================
• Onion +  Cheese Burger, with no onions : 3.0$
=========================================================================
• Thousand Island Sauce +  Onion +  Cheese Burger, with no onions : 3.0$
=========================================================================
• Coca Cola +  Thousand Island Sauce +  Onion +  Cheese Burger, with no onions : 5.94$
=========================================================================
• French Fries +  Coca Cola +  Thousand Island Sauce +  Onion +  Cheese Burger, with no onions : 7.89$
=========================================================================
 Sub Total =  7.89$
===================================
 Tax = 0.81$  (7.89 % 10.25)
===================================
 Grand Total =  8.7$
===================================
 End of Order, Check Closed.
```

Main.java

```
public class Main {

	  public static void main(String[] args) {

				Logger log = Logger.getLoggerInstance();
				log.logMessage("Inside Log");
		System.out.println("\n***  Hungry? Order the food you love! ***\n");


		System.out.println("\n\n ---- COMPOSITE PATTERN ----");

		//
		// COMPOSITE PATTERN IMPLEMENTATION
		//

		// Meal is the Base Component ~ Interface
		// Lunch / Dinner / Breakfast is the Leaf Component
		// ComboMeal is the Composite Element

		List<Meal> paradise_meals = new ArrayList<Meal>();

		ComboMeal paradiseComboMeal = new ComboMeal(paradise_meals);
		paradiseComboMeal.addMeal(MealFactory.getMealType(MealType.breakfast));
		paradiseComboMeal.addMeal(MealFactory.getMealType(MealType.lunch));
		paradiseComboMeal.addMeal(MealFactory.getMealType(MealType.dinner));

		Menu paradise_menu = new Menu(paradiseComboMeal);


		Restaurant paradise = new Restaurant(1, "Paradise Restaurant" , paradise_menu);

		//
		// Displaying the Meals menu
		//
		System.out.println("\nDISPLAYING THE MEALS MENU OF PARADISE RESTAURANT");
		paradise.getMenu().printMeals();


		System.out.println("\n\n ---- DECORATOR PATTERN ----\n\n");



		/*
		 *   DECORATOR PATTERN
		 */
		DecimalFormat df = new DecimalFormat("#.00");

		System.out.println(" Begin Food Order");
		System.out.println("======================================");

		// Order Cheese Burger
		OrderFood food = new CheeseBurger();
		float price = Float.valueOf(df.format(food.cost()));
		System.out.println("•" + food.getDesc() + " : " + price + "$");

System.out.println("================================================================================
===");
```

```
        // Add Onions
        food = new Onion(food);
        price = Float.valueOf(df.format(food.cost()));
        System.out.println("•" + food.getDesc() + " : " + price + "$");

System.out.println("=============================================================================
===");

        // Add Extra Thousand Island Sauce
        food = new ThousandIslandSauce(food);
        System.out.println("•" + food.getDesc() + " : " + price + "$");

System.out.println("=============================================================================
===");

        // Add Coke
        food = new Coke(food);
        price = Float.valueOf(df.format(food.cost()));
        System.out.println("•" + food.getDesc() + " : " + price + "$");

System.out.println("=============================================================================
===");

        // Add French Fries
        food = new Fries(food);
        price = Float.valueOf(df.format(food.cost()));
        System.out.println("•" + food.getDesc() + " : " + price + "$");

System.out.println("=============================================================================
===");

        // Return SubTotal
        food = new SubTotal(food);
        price = Float.valueOf(df.format(food.cost()));
        System.out.println(food.getDesc() + " " + price + "$");
        System.out.println("=====================================");

        // Add tax and show Total
        food = new TaxTotal(food);
        price = Float.valueOf(df.format(food.cost()));
        System.out.println(food.getDesc() + " " + price + "$");
        System.out.println("=====================================");
        System.out.println(" End of Order, Check Closed.");


}
}
```