using System;

namespace DecoratorPattern

{

public interface Pizza

{

string MakePizza();

}

public class PlainPizza : Pizza

{

public string MakePizza()

{

return "Plain Pizza";

}

}

public abstract class PizzaDecorator : Pizza

{

protected Pizza pizza;

public PizzaDecorator(Pizza pizza)

{

this.pizza = pizza;

}

public virtual string MakePizza()

{

return pizza.MakePizza();

}

}

public class ChickenPizzaDecorator : PizzaDecorator

{

public ChickenPizzaDecorator(Pizza pizza) : base(pizza)

{

}

public override string MakePizza()

{

return pizza.MakePizza() + AddChicken();

}

private string AddChicken()

{

return ", Chicken added";

}

}

public class VegPizzaDecorator : PizzaDecorator

{

public VegPizzaDecorator(Pizza pizza) : base(pizza)

{

}

public override string MakePizza()

{

return pizza.MakePizza() + AddVegetables();

}

private string AddVegetables()

{

return ", Vegetables added";

}

}

class Program

{

static void Main(string[] args)

{

PlainPizza plainPizzaObj = new PlainPizza();

string plainPizza = plainPizzaObj.MakePizza();

Console.WriteLine(plainPizza);

PizzaDecorator chickenPizzaDecorator = new ChickenPizzaDecorator(plainPizzaObj);

string chickenPizza = chickenPizzaDecorator.MakePizza();

Console.WriteLine("\n'" + chickenPizza + "' using ChickenPizzaDecorator");

VegPizzaDecorator vegPizzaDecorator = new VegPizzaDecorator(plainPizzaObj);

string vegPizza = vegPizzaDecorator.MakePizza();

Console.WriteLine("\n'" + vegPizza + "' using VegPizzaDecorator");

Console.Read();

}

}

}