**Decorator:**

public interface IBeverage

{

double GetCost();

string GetDescription();

}

public class Coffee : IBeverage

{

public double GetCost()

{

return 50.0;

}

public string GetDescription()

{

return "Coffee";

}

}

public abstract class BeverageDecorator : IBeverage

{

protected IBeverage \_beverage;

public BeverageDecorator(IBeverage beverage)

{

\_beverage = beverage;

}

public virtual double GetCost()

{

return \_beverage.GetCost();

}

public virtual string GetDescription()

{

return \_beverage.GetDescription();

}

}

public class MilkDecorator : BeverageDecorator

{

public MilkDecorator(IBeverage beverage) : base(beverage) { }

public override double GetCost()

{

return base.GetCost() + 10.0;

}

public override string GetDescription()

{

return base.GetDescription() + ", Milk";

}

}

public class SugarDecorator : BeverageDecorator

{

public SugarDecorator(IBeverage beverage) : base(beverage) { }

public override double GetCost()

{

return base.GetCost() + 5.0;

}

public override string GetDescription()

{

return base.GetDescription() + ", Sugar";

}

}

public class ChocolateDecorator : BeverageDecorator

{

public ChocolateDecorator(IBeverage beverage) : base(beverage) { }

public override double GetCost()

{

return base.GetCost() + 15.0;

}

public override string GetDescription()

{

return base.GetDescription() + ", Chocolate";

}

}

**Adapter:**

public interface IPaymentProcessor

{

void ProcessPayment(double amount);

void RefundPayment(double amount);

}

public class InternalPaymentProcessor : IPaymentProcessor

{

public void ProcessPayment(double amount)

{

Console.WriteLine($"Processing payment of {amount} via internal system.");

}

public void RefundPayment(double amount)

{

Console.WriteLine($"Refunding payment of {amount} via internal system.");

}

}

public class ExternalPaymentSystemA

{

public void MakePayment(double amount)

{

Console.WriteLine($"Making payment of {amount} via External Payment System A.");

}

public void MakeRefund(double amount)

{

Console.WriteLine($"Making refund of {amount} via External Payment System A.");

}

}

public class ExternalPaymentSystemB

{

public void SendPayment(double amount)

{

Console.WriteLine($"Sending payment of {amount} via External Payment System B.");

}

public void ProcessRefund(double amount)

{

Console.WriteLine($"Processing refund of {amount} via External Payment System B.");

}

}

public class PaymentAdapterA : IPaymentProcessor

{

private ExternalPaymentSystemA \_externalSystemA;

public PaymentAdapterA(ExternalPaymentSystemA externalSystemA)

{

\_externalSystemA = externalSystemA;

}

public void ProcessPayment(double amount)

{

\_externalSystemA.MakePayment(amount);

}

public void RefundPayment(double amount)

{

\_externalSystemA.MakeRefund(amount);

}

}

public class PaymentAdapterB : IPaymentProcessor

{

private ExternalPaymentSystemB \_externalSystemB;

public PaymentAdapterB(ExternalPaymentSystemB externalSystemB)

{

\_externalSystemB = externalSystemB;

}

public void ProcessPayment(double amount)

{

\_externalSystemB.SendPayment(amount);

}

public void RefundPayment(double amount)

{

\_externalSystemB.ProcessRefund(amount);

}

}

**Main:**

static void Main(string[] args)

{

//Decorator

IBeverage beverage = new Coffee();

Console.WriteLine($"{beverage.GetDescription()} : {beverage.GetCost()}");

beverage = new MilkDecorator(beverage);

Console.WriteLine($"{beverage.GetDescription()} : {beverage.GetCost()}");

beverage = new SugarDecorator(beverage);

Console.WriteLine($"{beverage.GetDescription()} : {beverage.GetCost()}");

beverage = new ChocolateDecorator(beverage);

Console.WriteLine($"{beverage.GetDescription()} : {beverage.GetCost()}");

//Adapter

IPaymentProcessor internalProcessor = new InternalPaymentProcessor();

internalProcessor.ProcessPayment(100.0);

internalProcessor.RefundPayment(50.0);

ExternalPaymentSystemA externalSystemA = new ExternalPaymentSystemA();

IPaymentProcessor adapterA = new PaymentAdapterA(externalSystemA);

adapterA.ProcessPayment(200.0);

adapterA.RefundPayment(100.0);

ExternalPaymentSystemB externalSystemB = new ExternalPaymentSystemB();

IPaymentProcessor adapterB = new PaymentAdapterB(externalSystemB);

adapterB.ProcessPayment(300.0);

adapterB.RefundPayment(150.0);

}

