**Strategy**

public interface IShippingStrategy

{

decimal CalculateShippingCost(decimal weight, decimal distance);

}

public class StandardShippingStrategy : IShippingStrategy

{

public decimal CalculateShippingCost(decimal weight, decimal distance)

{

return weight \* 0.5m + distance \* 0.1m;

}

}

public class ExpressShippingStrategy : IShippingStrategy

{

public decimal CalculateShippingCost(decimal weight, decimal distance)

{

return (weight \* 0.75m + distance \* 0.2m) + 10;

}

}

public class InternationalShippingStrategy : IShippingStrategy

{

public decimal CalculateShippingCost(decimal weight, decimal distance)

{

return weight \* 1.0m + distance \* 0.5m + 15;

}

}

public class NightShippingStrategy : IShippingStrategy

{

public decimal CalculateShippingCost(decimal weight, decimal distance)

{

return weight \* 0.6m + distance \* 0.15m + 20;

}

}

public class DeliveryContext

{

private IShippingStrategy \_shippingStrategy;

public void SetShippingStrategy(IShippingStrategy strategy)

{

\_shippingStrategy = strategy;

}

public decimal CalculateCost(decimal weight, decimal distance)

{

if (\_shippingStrategy == null)

{

throw new InvalidOperationException("Стратегия доставки не установлена.");

}

if (weight <= 0 || distance <= 0)

{

throw new ArgumentException("Вес и расстояние должны быть положительными числами.");

}

return \_shippingStrategy.CalculateShippingCost(weight, distance);

}

}

class Program

{

static void Main(string[] args)

{

DeliveryContext deliveryContext = new DeliveryContext();

Console.WriteLine("Выберите тип доставки: 1 - Стандартная, 2 - Экспресс, 3 - Международная, 4 - Ночная");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

deliveryContext.SetShippingStrategy(new StandardShippingStrategy());

break;

case "2":

deliveryContext.SetShippingStrategy(new ExpressShippingStrategy());

break;

case "3":

deliveryContext.SetShippingStrategy(new InternationalShippingStrategy());

break;

case "4":

deliveryContext.SetShippingStrategy(new NightShippingStrategy());

break;

default:

Console.WriteLine("Неверный выбор.");

return;

}

try

{

Console.WriteLine("Введите вес посылки (кг):");

decimal weight = Convert.ToDecimal(Console.ReadLine());

Console.WriteLine("Введите расстояние доставки (км):");

decimal distance = Convert.ToDecimal(Console.ReadLine());

decimal cost = deliveryContext.CalculateCost(weight, distance);

Console.WriteLine($"Стоимость доставки: {cost}$");

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

}

}