L'exercice que je vous propose sera une implémentation simplifiée d'un système de commandes pour un restaurant.

**Diagramme UML avec le Design Pattern Commande**:

1. **Commande** (Interface)
   * Execute()
2. **CommandeConcret** (Classe) : Implémente **Commande**
   * Execute()
3. **Invocateur** (Classe)
   * SetCommande(Commande)
   * Invoke()
4. **Receveur** (Classe)
   * Action()

**Scénario**: L'Invocateur est le serveur du restaurant. La Commande représente une demande faite par le client, par exemple commander un plat. Le Receveur est la cuisine qui prépare le plat.

**Implémentation C#**:

csharp

// Commande Interface

public interface ICommande

{

void Execute();

}

// CommandeConcret

public class CommandePlat : ICommande

{

private readonly Receveur \_receveur;

private readonly string \_plat;

public CommandePlat(Receveur receveur, string plat)

{

\_receveur = receveur;

\_plat = plat;

}

public void Execute()

{

\_receveur.Prepare(\_plat);

}

}

// Receveur

public class Receveur

{

public void Prepare(string plat)

{

Console.WriteLine($"Préparation du {plat}");

}

}

// Invocateur

public class Serveur

{

private ICommande \_commande;

public void SetCommande(ICommande commande)

{

\_commande = commande;

}

public void PrendreCommande()

{

\_commande.Execute();

}

}

// Client

public class Client

{

public static void Main(string[] args)

{

// Initialisation

Receveur cuisine = new Receveur();

ICommande commande = new CommandePlat(cuisine, "Steak Frites");

Serveur serveur = new Serveur();

// Le serveur prend la commande

serveur.SetCommande(commande);

serveur.PrendreCommande();

}

}

Ce code représente une utilisation basique du pattern Commande. Quand vous exécutez ce code, il va afficher "Préparation du Steak Frites". Vous pouvez étendre cette implémentation pour gérer plusieurs types de commandes, comme commander une boisson, ajouter des garnitures, etc.

Notez que cette implémentation est une version simplifiée du Design Pattern Commande pour des raisons pédagogiques.