Ejercicio 1:

Builder

public class Sandwich {

private String bread;

private String cheese;

private String meat;

private String vegetables;

private String condiments;

private Sandwich(Builder builder) {

this.bread = builder.bread;

this.cheese = builder.cheese;

this.meat = builder.meat;

this.vegetables = builder.vegetables;

this.condiments = builder.condiments;

}

public static class Builder {

private String bread;

private String cheese;

private String meat;

private String vegetables;

private String condiments;

public Builder setBread(String bread) {

this.bread = bread;

return this;

}

public Builder setCheese(String cheese) {

this.cheese = cheese;

return this;

}

public Builder setMeat(String meat) {

this.meat = meat;

return this;

}

public Builder setVegetables(String vegetables) {

this.vegetables = vegetables;

return this;

}

public Builder setCondiments(String condiments) {

this.condiments = condiments;

return this;

}

public Sandwich build() {

return new Sandwich(this);

}

}

@Override

public String toString() {

return "Sandwich with " + bread + " bread, " + cheese + " cheese, " + meat + " meat, " + vegetables + " vegetables, and " + condiments + " condiments.";

}

}

Ejercicio 2:  
En este caso se modificaría el main, se agregaría una interfaz que le permita a las clases aplicar el patrón

public interface ICloneable<T>

{

T Clone();

}

public abstract class GameUnit : ICloneable<GameUnit>

{

public int Health { get; set; }

public int Attack { get; set; }

public int Defense { get; set; }

// Simula la carga de recursos costosos como modelos 3D, texturas, etc.

public virtual void LoadResources()

{

Console.WriteLine("Loading resources...");

}

public abstract GameUnit Clone();

}

public class Archer : GameUnit, ICloneable<Archer>

{

public Archer()

{

LoadResources();

Health = 100;

Attack = 15;

Defense = 5;

}

private Archer(Archer archer)

{

Health = archer.Health;

Attack = archer.Attack;

Defense = archer.Defense;

}

public override GameUnit Clone()

{

return CloneSpecific();

}

public Archer CloneSpecific()

{

return new Archer(this);

}

}

public class Knight : GameUnit, ICloneable<Knight>

{

public Knight()

{

LoadResources();

Health = 200;

Attack = 20;

Defense = 10;

}

private Knight(Knight knight)

{

Health = knight.Health;

Attack = knight.Attack;

Defense = knight.Defense;

}

public override GameUnit Clone()

{

return CloneSpecific();

}

public Knight CloneSpecific()

{

return new Knight(this);

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Creating original Archer...");

Archer originalArcher = new Archer();

Archer copiedArcher1 = originalArcher.Clone();

Archer copiedArcher2 = originalArcher.Clone();

Console.WriteLine("Creating original Knight...");

Knight originalKnight = new Knight();

Knight copiedKnight1 = originalKnight.Clone();

Knight copiedKnight2 = originalKnight.Clone();

}

}

3 factory

public interface IMessage{

void sendMessage(String msg);

}

public class EmailMessage implements IMessage{

@Override

public void sendMessage(String msg) {

System.*out*.println("Email: " + msg);

}

}

public class SmsMessage implements IMessage{

@Override

public void sendMessage(String msg) {

System.*out*.println("SMS: " + msg);

}

}

public class FacebookMessage implements IMessage{

@Override

public void sendMessage(String msg) {

System.*out*.println("Facebook: " + msg);

}

}

public class MessageApp {

public IMessage sendMessage(String type) {

if(type.equals("email")) {

return new EmailMessage();

} else if(type.equals("sms")) {

return new SmsMessage();

} else if(type.equals("facebook")) {

return new FacebookMessage();

}else{

return null;

}

}

}

public class App {

private MessageApp messageApp;

public App(MessageApp messageApp){

this.messageApp = messageApp;

}

public void sendMessage(String type, String msg) {

IMessage message = this.messageApp.sendMessage(type);

message.sendMessage(msg);

}

}

4 prototype

public interface IClonable<T>

{

T Clone();

}

public class Book : IClonable<Book>

{

public string Title { get; set; }

public string Author { get; set; }

public List<string> BorrowedStudents { get; set; }

public Book()

{

// Simular la carga de recursos.

Console.WriteLine("Acquiring a new book...");

BorrowedStudents = new List<string>();

}

public Book Clone()

{

Book copy = (Book)this.MemberwiseClone();

copy.BorrowedStudents = new List<string>();

return copy;

}

public void BorrowBook(string studentName)

{

BorrowedStudents.Add(studentName);

}

public void PrintBorrowedStudents()

{

Console.WriteLine($"Book: {Title}, Borrowed by: {string.Join(", ", BorrowedStudents)}");

}

}

class Program

{

static void Main(string[] args)

{

// Adquirir el libro original.

Book originalBook = new Book

{

Title = "Harry Potter",

Author = "J.K. Rowling"

};

// Prestar el libro original a un estudiante.

originalBook.BorrowBook("Alice");

// Clonar el libro original para crear una copia adicional.

Book additionalCopy = originalBook.Clone();

// Prestar la copia adicional a otro estudiante.

additionalCopy.BorrowBook("Bob");

// Imprimir los estudiantes a los que se les prestó cada copia del libro.

originalBook.PrintBorrowedStudents();

additionalCopy.PrintBorrowedStudents();

}

}

5 builder

public static TravelPlanBuilder Builder(string flight, string hotel)

{

return new TravelPlanBuilder(flight, hotel);

}

public class TravelPlanBuilder

{

private string \_flight;

private string \_hotel;

private string \_carRental;

private string[] \_activities;

private string[] \_restaurantReservations;

public TravelPlanBuilder(string flight, string hotel)

{

\_flight = flight;

\_hotel = hotel;

}

public TravelPlanBuilder SetCarRental(string carRental)

{

\_carRental = carRental;

return this;

}

public TravelPlanBuilder SetActivities(string[] activities)

{

\_activities = activities;

return this;

}

public TravelPlanBuilder SetRestaurantReservations(string[] restaurantReservations)

{

\_restaurantReservations = restaurantReservations;

return this;

}

public TravelPlan Build()

{

return new TravelPlan(\_flight, \_hotel, \_carRental, \_activities, \_restaurantReservations);

}

}

}

class Program

{

static void Main(string[] args)

{

TravelPlan plan = TravelPlan.*Builder*("Flight1", "Hotel1")

.SetCarRental(null)

.SetActivities(new string[] { "Tour1", "Tour2" })

.SetRestaurantReservations(null)

.Build();

}

}

6 singleton

public class Service implements SomeService

{

private static Service *instance*;

private string apiEndpoint = "https://example.com/api";

private Service() { }

public static ConfigurationManager Instance

{

if (*instance* == null)

{

*instance* = new ConfigurationManager();

}

return *instance*;

}

public string GetConfiguration(string a)

{

if (a == "apiEndpoint")

{

return apiEndpoint;

}

else

{

return null;

}

}

}