Week 1 .NET FSE Hands-On Exercises

# **Exercise 1: Implementing the Singleton Pattern**

**Question: Implement the Singleton design pattern in C#. Ensure that only one instance of the class is created and provide a global access point to it.**

**Answer:**

public sealed class Singleton  
{  
 private static Singleton \_instance = null;  
 private static readonly object \_lock = new object();  
  
 private Singleton() { }  
  
 public static Singleton Instance  
 {  
 get  
 {  
 lock (\_lock)  
 {  
 if (\_instance == null)  
 \_instance = new Singleton();  
 return \_instance;  
 }  
 }  
 }  
  
 public void ShowMessage()  
 {  
 Console.WriteLine("Singleton instance method called.");  
 }  
}

# **Exercise 2: Implementing the Factory Method Pattern**

**Question: Implement the Factory Method design pattern in C#. Create a base Product class and derived classes with different implementations, and use a factory method to instantiate them.**

**Answer:**

public abstract class Product  
{  
 public abstract string GetDetails();  
}  
  
public class ConcreteProductA : Product  
{  
 public override string GetDetails() => "Product A created";  
}  
  
public class ConcreteProductB : Product  
{  
 public override string GetDetails() => "Product B created";  
}  
  
public abstract class Creator  
{  
 public abstract Product FactoryMethod();  
}  
  
public class CreatorA : Creator  
{  
 public override Product FactoryMethod() => new ConcreteProductA();  
}  
  
public class CreatorB : Creator  
{  
 public override Product FactoryMethod() => new ConcreteProductB();  
}  
  
public class Program  
{  
 public static void Main()  
 {  
 Creator creator = new CreatorA();  
 Product product = creator.FactoryMethod();  
 Console.WriteLine(product.GetDetails());  
  
 creator = new CreatorB();  
 product = creator.FactoryMethod();  
 Console.WriteLine(product.GetDetails());  
 }  
}

# **Exercise 3: E-commerce Platform Search Function**

**Question: Create a search function for an e-commerce platform using appropriate data structures to store and search product information.**

**Answer:**

using System;  
using System.Collections.Generic;  
  
public class Product  
{  
 public int Id { get; set; }  
 public string Name { get; set; }  
}  
  
public class Program  
{  
 static Dictionary<int, Product> productCatalog = new Dictionary<int, Product>  
 {  
 { 1, new Product { Id = 1, Name = "Laptop" } },  
 { 2, new Product { Id = 2, Name = "Mouse" } },  
 { 3, new Product { Id = 3, Name = "Keyboard" } }  
 };  
  
 public static Product SearchProduct(int id)  
 {  
 if (productCatalog.ContainsKey(id))  
 return productCatalog[id];  
 return null;  
 }  
  
 public static void Main()  
 {  
 var product = SearchProduct(2);  
 Console.WriteLine(product != null ? product.Name : "Product not found");  
 }  
}

# **Exercise 4: Financial Forecasting**

**Question: Use arrays or other data structures to implement a financial forecasting solution using moving average.**

**Answer:**

using System;  
  
public class FinancialForecast  
{  
 public static double MovingAverage(int[] sales, int windowSize)  
 {  
 double sum = 0;  
 for (int i = sales.Length - windowSize; i < sales.Length; i++)  
 sum += sales[i];  
 return sum / windowSize;  
 }  
  
 public static void Main()  
 {  
 int[] monthlySales = { 100, 150, 200, 250, 300, 350 };  
 int windowSize = 3;  
 double forecast = MovingAverage(monthlySales, windowSize);  
 Console.WriteLine($"Next month's forecast: {forecast}");  
 }  
}