22053575-Anand-Ayushman-Das

Q1) Code

// Logger class with Singleton pattern

public class Logger

{

private static Logger? \_instance;

private static readonly object \_lock = new object();

private Logger()

{

Console.WriteLine("Logger instance created!");

}

public static Logger GetInstance()

{

if (\_instance == null)

{

lock (\_lock)

{

if (\_instance == null)

{

\_instance = new Logger();

}

}

}

return \_instance;

}

public void Log(string message)

{

Console.WriteLine($"[LOG {DateTime.Now:yyyy-MM-dd HH:mm:ss}]: {message}");

}

public string GetInstanceInfo()

{

return $"Logger Instance Hash Code: {this.GetHashCode()}";

}

}

// Main Program class

public class Program

{

public static void Main(string[] args)

{

Console.WriteLine("=== Singleton Pattern Test ===\n");

Console.WriteLine("Test 1: Getting Logger instances...");

Logger logger1 = Logger.GetInstance();

Logger logger2 = Logger.GetInstance();

Logger logger3 = Logger.GetInstance();

Console.WriteLine("\nTest 2: Checking instance equality...");

Console.WriteLine($"logger1 == logger2: {ReferenceEquals(logger1, logger2)}");

Console.WriteLine($"logger2 == logger3: {ReferenceEquals(logger2, logger3)}");

Console.WriteLine("\nTest 3: Instance information...");

Console.WriteLine(logger1.GetInstanceInfo());

Console.WriteLine(logger2.GetInstanceInfo());

Console.WriteLine("\nTest 4: Testing logging functionality...");

logger1.Log("Application started");

logger2.Log("User logged in");

logger3.Log("Data processing completed");

Console.WriteLine("\n=== All tests completed ===");

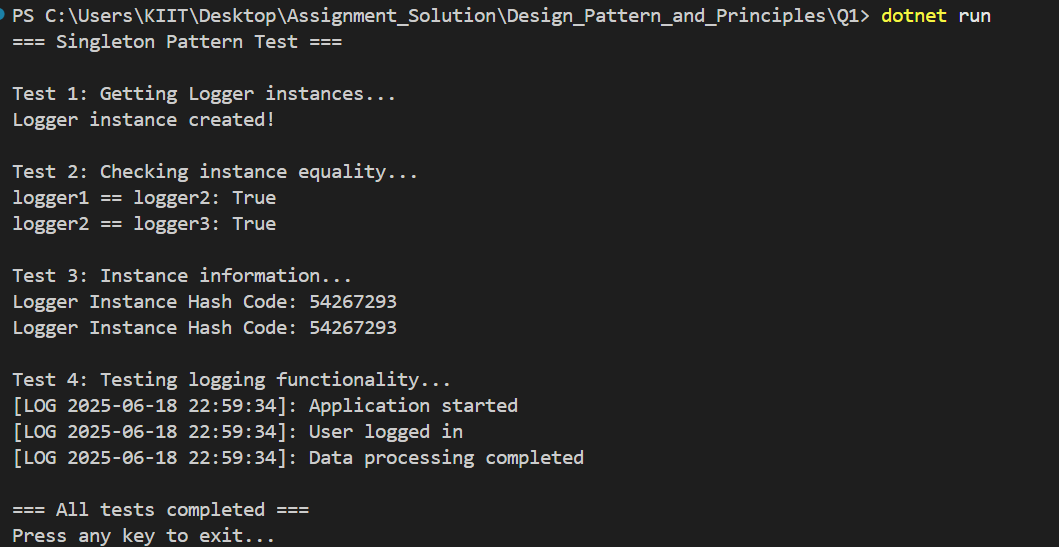
Console.WriteLine("Press any key to exit...");

Console.ReadKey();

}

}

Q1) Output



Q2) Code

using System;

// Step 2: Abstract Document Interface

public abstract class Document

{

public abstract void Open();

public abstract void Save();

public abstract void Close();

public abstract string GetDocumentType();

}

// Step 3: Concrete Document Classes

public class WordDocument : Document

{

public override void Open()

{

Console.WriteLine("Opening Word Document (.docx)");

}

public override void Save()

{

Console.WriteLine("Saving Word Document with formatting and text styles");

}

public override void Close()

{

Console.WriteLine("Closing Word Document");

}

public override string GetDocumentType()

{

return "Microsoft Word Document";

}

}

public class PdfDocument : Document

{

public override void Open()

{

Console.WriteLine("Opening PDF Document (.pdf)");

}

public override void Save()

{

Console.WriteLine("Saving PDF Document with fixed layout and security options");

}

public override void Close()

{

Console.WriteLine("Closing PDF Document");

}

public override string GetDocumentType()

{

return "Portable Document Format";

}

}

public class ExcelDocument : Document

{

public override void Open()

{

Console.WriteLine("Opening Excel Document (.xlsx)");

}

public override void Save()

{

Console.WriteLine("Saving Excel Document with spreadsheet data and formulas");

}

public override void Close()

{

Console.WriteLine("Closing Excel Document");

}

public override string GetDocumentType()

{

return "Microsoft Excel Spreadsheet";

}

}

// Step 4: Abstract Factory Class

public abstract class DocumentFactory

{

// Factory Method - to be implemented by concrete factories

public abstract Document CreateDocument();

// Template method that uses the factory method

public void ProcessDocument()

{

Document doc = CreateDocument();

Console.WriteLine($"\n--- Processing {doc.GetDocumentType()} ---");

doc.Open();

doc.Save();

doc.Close();

Console.WriteLine("--- Process Complete ---\n");

}

}

// Step 4: Concrete Factory Classes

public class WordDocumentFactory : DocumentFactory

{

public override Document CreateDocument()

{

Console.WriteLine("WordDocumentFactory: Creating Word Document");

return new WordDocument();

}

}

public class PdfDocumentFactory : DocumentFactory

{

public override Document CreateDocument()

{

Console.WriteLine("PdfDocumentFactory: Creating PDF Document");

return new PdfDocument();

}

}

public class ExcelDocumentFactory : DocumentFactory

{

public override Document CreateDocument()

{

Console.WriteLine("ExcelDocumentFactory: Creating Excel Document");

return new ExcelDocument();

}

}

// Alternative: Simple Factory (Bonus Implementation)

public class SimpleDocumentFactory

{

public static Document CreateDocument(string documentType)

{

return documentType.ToUpper() switch

{

"WORD" => new WordDocument(),

"PDF" => new PdfDocument(),

"EXCEL" => new ExcelDocument(),

\_ => throw new ArgumentException($"Unknown document type: {documentType}")

};

}

}

// Step 5: Test Class (Main Program)

public class Program

{

public static void Main(string[] args)

{

Console.WriteLine("=== Factory Method Pattern Demo ===");

Console.WriteLine("Document Management System\n");

// Test 1: Using Factory Method Pattern

Console.WriteLine("TEST 1: Factory Method Pattern");

Console.WriteLine("================================");

// Create different document factories

DocumentFactory wordFactory = new WordDocumentFactory();

DocumentFactory pdfFactory = new PdfDocumentFactory();

DocumentFactory excelFactory = new ExcelDocumentFactory();

// Use factories to create and process documents

wordFactory.ProcessDocument();

pdfFactory.ProcessDocument();

excelFactory.ProcessDocument();

// Test 2: Direct document creation using factories

Console.WriteLine("TEST 2: Direct Document Creation");

Console.WriteLine("================================");

Document[] documents = {

wordFactory.CreateDocument(),

pdfFactory.CreateDocument(),

excelFactory.CreateDocument()

};

foreach (Document doc in documents)

{

Console.WriteLine($"Created: {doc.GetDocumentType()}");

}

// Test 3: Simple Factory Pattern (Alternative approach)

Console.WriteLine("\nTEST 3: Simple Factory Pattern (Alternative)");

Console.WriteLine("============================================");

try

{

Document wordDoc = SimpleDocumentFactory.CreateDocument("WORD");

Document pdfDoc = SimpleDocumentFactory.CreateDocument("PDF");

Document excelDoc = SimpleDocumentFactory.CreateDocument("EXCEL");

Console.WriteLine($"Simple Factory created: {wordDoc.GetDocumentType()}");

Console.WriteLine($"Simple Factory created: {pdfDoc.GetDocumentType()}");

Console.WriteLine($"Simple Factory created: {excelDoc.GetDocumentType()}");

// Test error handling

Console.WriteLine("\nTesting error handling:");

Document unknownDoc = SimpleDocumentFactory.CreateDocument("UNKNOWN");

}

catch (ArgumentException ex)

{

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

}

Console.WriteLine("\n=== Demo Complete ===");

Console.WriteLine("Press any key to exit...");

Console.ReadKey();

}

}

Q2) Output

