C# Assignments & Solutions: Part 5

Assignment 1. Identify the need for single and multidimensional arrays as properties of a class

Question: Create a Library class to store and manage books in a single-dimensional array where each element represents a book title. Also, use a two-dimensional array in a Classroom class where each element represents the seat of a student based on rows and columns.

Assignment 2. Identify the need for an array of objects

Question: Define a Movie class with properties like Title and Year. Create a MovieLibrary class that stores an array of Movie objects and displays each movie's information.

Assignment 3. Identify the need for enumerations

Question: Create an enumeration OrderStatus with values Pending, Shipped, and Delivered. Define an Order class with an OrderStatus property to manage the status of each order.

Assignment 4. Identify the need for value types and reference types

Question: Create a Circle struct to represent a circle as a value type, and a Shape class to represent a reference type. Show how changing values affects each.

Assignment 5. Identify the need for pass by reference using ref keyword

Question: Create a method IncreaseScore that takes a ref integer parameter representing a player's score and increases it by 10.

Assignment 6. Identify the need for out parameters

Question: Create a method Divide that takes two integers, calculates their quotient and remainder, and returns both values using the out keyword.

Assignment 7. What is Exception and its Types

Question: Write a C# program that demonstrates a FileNotFoundException and an IndexOutOfRangeException. Explain what causes these exceptions.

Assignment 8. How to Handle Exception

Question: Create a method GetUserInput that prompts the user to enter an integer. If the input is invalid, catch the exception and prompt the user again until they enter a valid integer.

Assignment 9. Multiple Catch Blocks

Question: Create a method that calculates the division of two integers. Use multiple catch blocks to handle DivideByZeroException and FormatException.

Assignment 10. Finally Block

Question: Write a method ReadFile that opens a file, reads its contents, and closes it. Use a finally block to ensure the file is closed, even if an exception occurs.

Assignment 11. Difference between throw and throws

Question: Explain the purpose of the throw keyword in C#. Create a method ValidateAge that throws an exception if the age is below 18, and handle it in the calling method.

Assignment 12. Custom Exception

Question: Create a custom exception class NegativeNumberException to handle negative inputs. Write a method CheckNumber that throws this exception if the number is negative.

Assignment 13. Logging Exceptions

a. Install log4net via NuGet

Install log4net by opening the NuGet Package Manager Console and running:

Install-Package log4net

b. Configure log4net in App.config or Web.config

Add the following configuration in App.config or Web.config:

```
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,
log4net"/>
  </configSections>

  <log4net>
    <appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">
      <file value="log-file.log" />
      <appendToFile value="true" />
      <rollingStyle value="Size" />
      <maxSizeRollBackups value="5" />
      <maximumFileSize value="1MB" />
```

```
      <staticLogFileName value="true" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date [%thread] %-5level %logger -
%message%newline" />
      </layout>
    </appender>
    <root>
      <level value="DEBUG" />
      <appender-ref ref="RollingFileAppender" />
    </root>
  </log4net>
</configuration>
```

c. Initialize log4net by Creating a Logger Class

```
using log4net;
using log4net.Config;

public class Logger
{
    private static readonly ILog log = LogManager.GetLogger(typeof(Logger));

    static Logger()
    {
        XmlConfigurator.Configure();
    }

    public static void LogError(string message, Exception ex)
    {
        log.Error(message, ex);
    }
}
```

d. Use the Logger to Log Exceptions in the Application

Question: Write a method Divide that performs division and logs any DivideByZeroException using the Logger class.

Answer:

```
using System;

public class LoggingExample
{
    public static void Divide(int a, int b)
    {
        try
        {
            int result = a / b;
            Console.WriteLine("Result: " + result);
        }
        catch (DivideByZeroException ex)
        {
            Logger.LogError("Attempted to divide by zero.", ex);
            Console.WriteLine("DivideByZeroException caught and logged.");
```

```
        }
    }

    public static void Main()
    {
        Divide(10, 0);
    }
}
```