Technical Design Document

Introduction

This document is the technical design for the solution developed to accomplish the requirements for MinDrivesTest.

On the development of this project we have used the platform .Net Core 3.1 under Microsoft Windows 10 architecture. The IDE we used was Visual Studio Community 2019 and the programming language selected was C#. We have also implemented the Unit Tests to validate the requirements using the XUnit framework within .Net Core and Visual Studio ecosystem.

The development contains two implementations of the same requirement, one that uses the modern standards and techniques like Linq and helper classes to better accomplish the solution.

System Overview

As of late, your usually high-performance computer has been acting rather sluggish. You come to realize that while you have plenty of free disk space on your machine, it is split up over many hard drives. The secret to improving performance is to consolidate all the data on your computer onto as few hard drives as possible.

Given a int[] used, representing the amount of disk space used on each drive, and a corresponding int[] total , representing the total capacity of each drive mentioned in used, you should return the minimum number of hard drives needed to store the data after the consolidation is complete. You may assume that the data consists of very small files, such that splitting it up and moving parts of it onto different hard drives never presents a problem.

System Characteristics

The proposal contains a .Net Core Console Application with the classic Program.cs class and its Main method that will invoke the two implementations we have developed to cover these requirements.

This is a multiplatform application that can be compiled and executed on Windows/Linux/Mac stacks.

The prerequisites are:

- Latest Visual Studio 2019 Community.
- .Net Core 3.1 (Included with latest Visual Studio Community 2019 installation)
- xUnit (Included with latest Visual Studio Community 2019 installation)

System Architecture

No complex architecture has been used. Just the benefits of using static methods to easily invoke the two methods implemented. Also this helps on the unit tests since the methodos becomes accessible from the external test project and easy to invoke.

System Design

The implementation consists of two main elements or methods, one called “MinDrivesLinq” and the other one “MinDrivesConventional”.

The LINQ version is built in using the latest techniques and the power of lambda expressions, this version also uses a collection (List) of class “HardDrive” as a helper to easily store the Used and Total properties within the same container for further uses instead of having to handle two separated arrays.

The “conventional” version is just a raw implementation using the basic arrays and loops available on any languages.

Both were implemented within a class called “DiskSpace” to help with the Unit Testing.

Documentation Standards

Classes and methods has been properly documented using XML comments on headers.

Naming conventions

We have used camelCase for all variables in the program plus PascalCase for all methods and classes.

Programming Standards

C# was the language selected and .Net Core 3.1 the framework behind. XUnit was the selection for Unit Testing.

Both methods implemented are statics since there is not a specific behavior as object to have an instance of the class “” every time we needed to use them.

Software development tools

Visual Studio Community 2019

Unit Tests

Unit testing is an important part of software development life cycle and this solution contains a project to accomplish this important item.

The selected framework was XUnit and we have performed 6 test cases.