`SourceDataManager` Class

Brief Overview

The `SourceDataManager` class is the static repository for the Heatington application's core data management. It gets data from and saves data to a specific data source, which is an form of `IDataSource` interface.

Important Bits

- `_dataSource`: This is a private copy of the the `IDataSource` interface. It has all the methods needed to work with

- a specific data source.

- `_filePath`: This is a private copy of the path to the file where the data operations happen.

- `TimeSeriesData`: This public `List<DataPoint>` works like a holding area for the data. Each item represents a data point at a specific time.

Constructor

`SourceDataManager(IDataSource dataSource, string filePath)`

Constructing a new `SourceDataManager` needs an `IDataSource` instance and a file path string. This gives it the flexibility to work with any type of data source and file location. We use constructor injection to make sure that `SourceDataManager` is not tightly coupled to any specific data source.

Core Methods

- `ConvertApiToCsv(List<DataPoint> dataFromApi)`: This takes a list of `DataPoint` items and saves them using the `SaveData` method from `IDataSource`. **This method is for future use when we implement the API-driven iteration.**

- `FetchTimeSeriesData()`: This method fetches the data from the `_dataSource` using the `GetData` method and stores it in the `TimeSeriesData` property.

- `LogTimeSeriesData()`: This method logs the data in `TimeSeriesData`. Each log message contains the index, formatted start and end times, heat demand, and electricity price for each data point. **This method will be removed once we move to the GUI-driven iteration.**

Quick Note

The `SourceDataManager` implementation simplifies testing because `IDataSource` can be easily mocked. It also adds flexibility, as different implementations of `IDataSource` can be used without major code changes.

An Example Of How To Use It

```
using Heatington.Data;

namespace Heatington
{
    internal static class Program
    {

        public static async Task Main(string[] args) // async Task -> if we want to
implement async operation
                                                     // especially for IO
        {
            // Define the file path
            const string filePath = "../Assets/winter_period.csv";

            // Create a new CsvDataSource
            IDataSource dataSource = new CsvDataSource();

            SourceDataManager.SourceDataManager sdm = new(dataSource, filePath);

            // Fetch data from dataSource
            await sdm.FetchTimeSeriesDataAsync().ConfigureAwait(true);

            // Log the loaded data to the console
            sdm.LogTimeSeriesData();

            Console.WriteLine("Data loaded successfully.");

        }
    }
}
```