Exam Preparation II

Submit your solutions here: https://judge.softuni.org/Contests/4513/Exam-Preparation-II

1. Unit Test: CSV Parser

Test a given method which takes in **a string representing comma separated value data** and **returns an array with each value**.

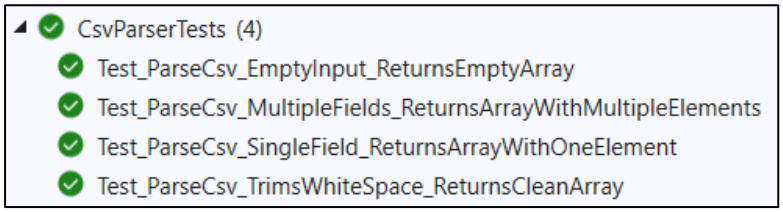
The method is found in the **`CsvParser.cs`** file:

```
public class CsvParser
{
    1 reference | 1/1 passing
    public static string[] ParseCsv(string csvData)
    {
        if (string.IsNullOrEmpty(csvData))
        {
            return Array.Empty<string>();
        }

        return csvData.Trim().Split(separator: ',', StringSplitOptions.TrimEntries);
    }
}
```

You are given a **test file `CsvParserTests.cs`** containing **4 empty tests**. Implement all tests:

```
public class CsvParserTests
{
    [Test]
    0 references
    public void Test_ParseCsv_EmptyInput_ReturnsEmptyArray()...

    [Test]
    0 references
    public void Test_ParseCsv_SingleField_ReturnsArrayWithOneElement()...

    [Test]
    0 references
    public void Test_ParseCsv_MultipleFields_ReturnsArrayWithMultipleElements()...

    [Test]
    0 references
    public void Test_ParseCsv_TrimsWhiteSpace_ReturnsCleanArray()...
}
```

When you are ready make sure your **tests run:**



IMPORTANT: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.

2. Unit Test: Fruits

Test a given method which takes in **a `<string, int>` dictionary representing fruits** and **a string fruit name**. It returns **the quantity of the fruit**.

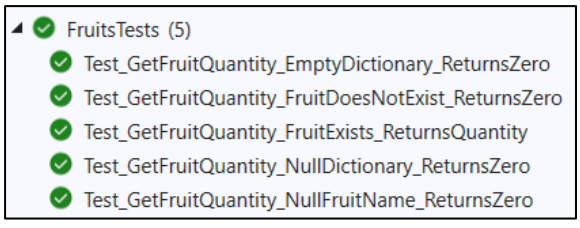
The method is found in the **`Fruits.cs`** file:

```
public class Fruits
{
    5 references
    public static int GetFruitQuantity(
        Dictionary<string, int>? fruitDictionary,
        string? fruitName)
    {
        if (fruitDictionary is null || fruitName is null)
        {
            return 0;
        }

        if (fruitDictionary.TryGetValue(fruitName, out int quantity))
        {
            return quantity;
        }

        return 0;
    }
}
```

You are given a **test file `FruitsTests.cs`** containing **5 empty tests**. Implement all tests:

```
public class FruitsTests
{
    [Test]
    0 references
    public void Test_GetFruitQuantity_FruitExists_ReturnsQuantity()...

    [Test]
    0 references
    public void Test_GetFruitQuantity_FruitDoesNotExist_ReturnsZero()...

    [Test]
    0 references
    public void Test_GetFruitQuantity_EmptyDictionary_ReturnsZero()...

    [Test]
    0 references
    public void Test_GetFruitQuantity_NullDictionary_ReturnsZero()...

    [Test]
    0 references
    public void Test_GetFruitQuantity_NullFruitName_ReturnsZero()...
}
```

When you are ready make sure your **tests run:**



IMPORTANT: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.

3. Unit Test: ToDo

You are given a **folder of 2 classes - `TaskItem`** and **`ToDoList`**. The **`TaskItem` class** is just a helper class:

```
public class TaskItem
{
    1 reference
    public TaskItem(string title, DateTime dueDate)
    {
        this.Title = title;
        this.DueDate = dueDate;
        this.IsCompleted = false;
    }

    3 references
    public string Title { get; }

    2 references
    public DateTime DueDate { get; }

    3 references
    public bool IsCompleted { get; set; }
}
```

The **`ToDoList`** class holds a **list of tasks** and **methods** for **using the list** that you will **test**:

```
public class ToDoList
{
    private readonly List<TaskItem> _tasks = new();

    4 references
    public void AddTask(string title, DateTime dueDate)
    {
        TaskItem newTask = new(title, dueDate);
        this._tasks.Add(newTask);
    }
```

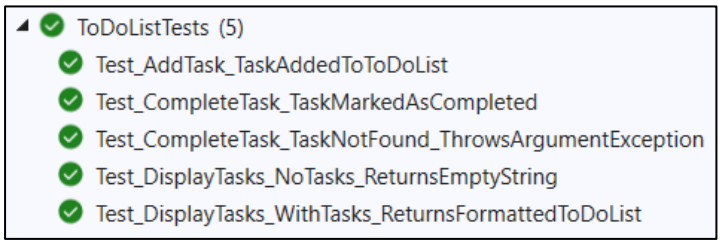
```
    public void CompleteTask(string title)
    {
        TaskItem? taskToComplete = this._tasks.FirstOrDefault(task =>
            task.Title.Equals(title, StringComparison.OrdinalIgnoreCase));
        if (taskToComplete is null)
        {
            throw new ArgumentException(message: "Task with given title does not exist.");
        }

        taskToComplete.IsCompleted = true;
    }
```

```
    public string DisplayTasks()
    {
        StringBuilder sb = new();

        sb.AppendLine("To-Do List:");
        foreach (TaskItem task in this._tasks)
        {
            sb.AppendLine($"{(task.IsCompleted ? "[✓]" : "[ ]")} {task.Title} " +
                          $"- Due: {task.DueDate.ToString(format: "MM/dd/yyyy")}");
        }

        return sb.ToString().Trim();
    }
}
```

You will need to use the test file **`ToDoListTests.cs`**, inside they are **5 empty tests with a setup method:**

```
public class ToDoListTests
{
    private ToDoList _toDoList = null!;

    [SetUp]
    0 references
    public void SetUp()...

    [Test]
    0 references
    public void Test_AddTask_TaskAddedToToDoList()...

    [Test]
    0 references
    public void Test_CompleteTask_TaskMarkedAsCompleted()...

    [Test]
    0 references
    public void Test_CompleteTask_TaskNotFound_ThrowsArgumentException()...

    [Test]
    0 references
    public void Test_DisplayTasks_NoTasks_ReturnsEmptyString()...

    [Test]
    0 references
    public void Test_DisplayTasks_WithTasks_ReturnsFormattedToDoList()...
}
```

When you are ready make sure your **tests run:**



IMPORTANT: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.