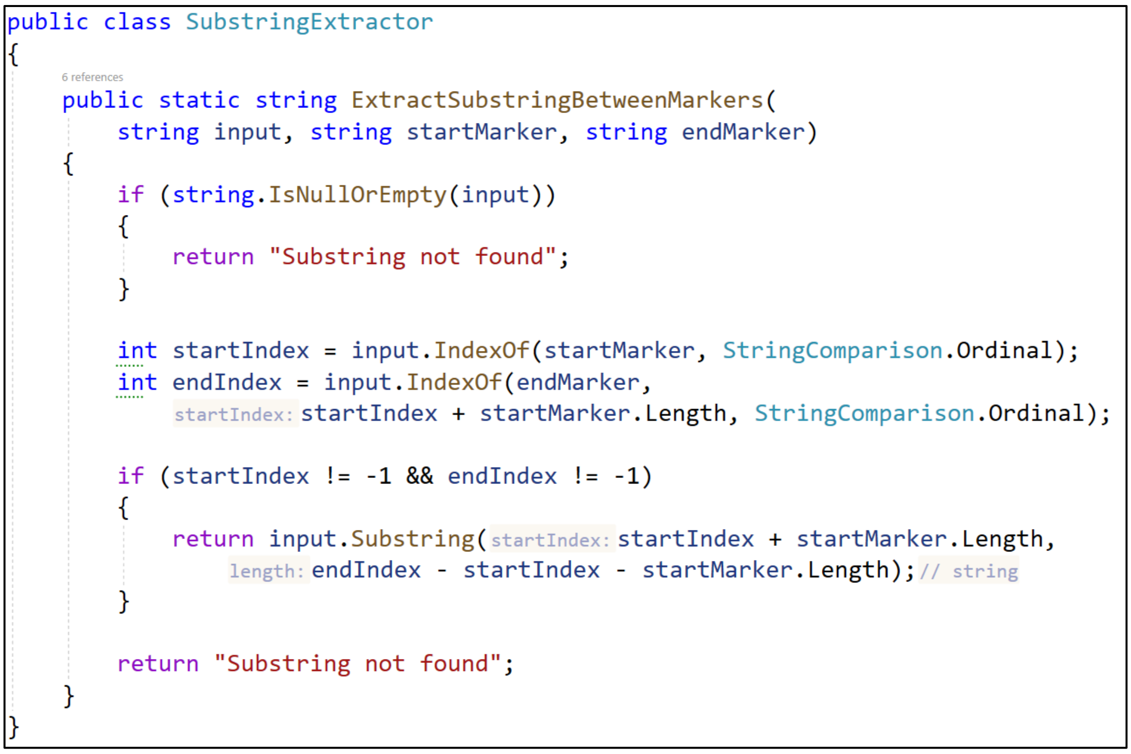
Exam Preparation III

Submit your solutions here: https://judge.softuni.org/Contests/4514/Exam-Preparation-III

1. Unit Test: Substring Extractor

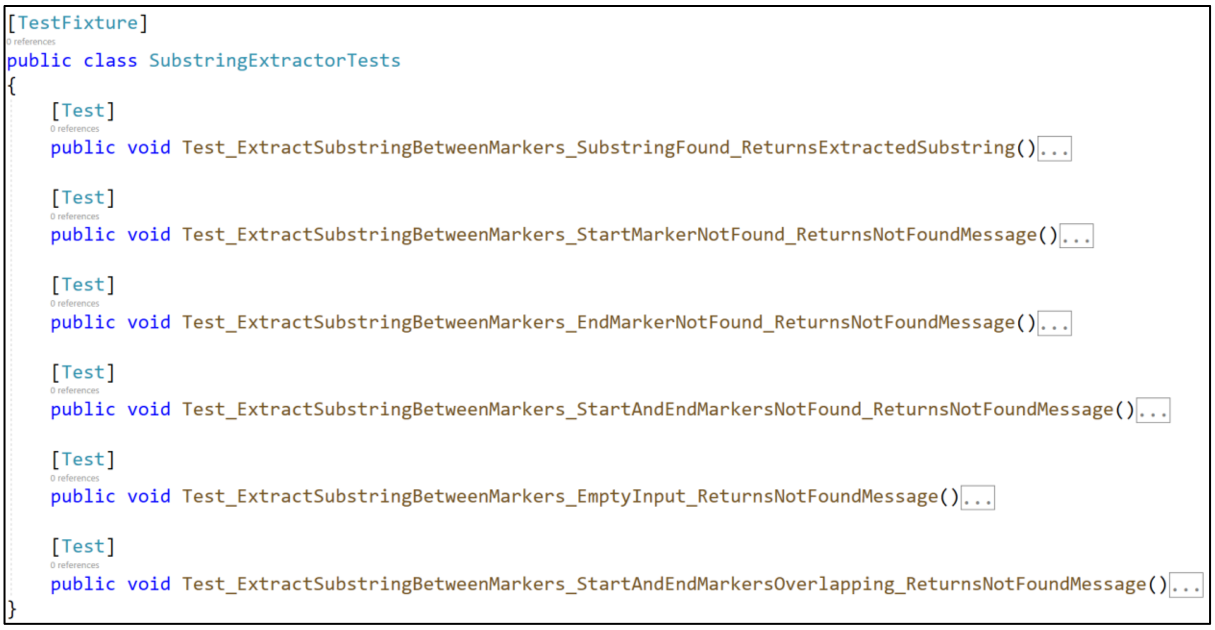
Test a given method which takes in **a string, and 2 markers marking beginning and end** and **extracts the text between the markers.**

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

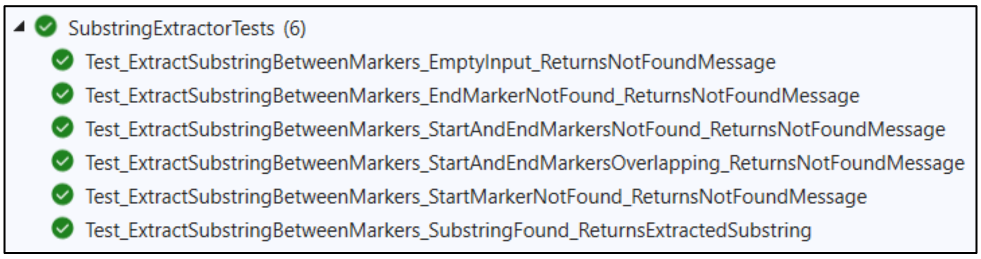


```
public class SubstringExtractor
{
    6 references
    public static string ExtractSubstringBetweenMarkers(
        string input, string startMarker, string endMarker)
    {
        if (string.IsNullOrEmpty(input))
        {
            return "Substring not found";
        }

        int startIndex = input.IndexOf(startMarker, StringComparison.Ordinal);
        int endIndex = input.IndexOf(endMarker,
            startIndex: startIndex + startMarker.Length, StringComparison.Ordinal);

        if (startIndex != -1 && endIndex != -1)
        {
            return input.Substring(startIndex: startIndex + startMarker.Length,
                length: endIndex - startIndex - startMarker.Length); // string
        }

        return "Substring not found";
    }
}
```

You are given a **test file `SubstringExtractorTests.cs`** containing **6 empty tests**. Implement all tests:


```
[TestFixture]
0 references
public class SubstringExtractorTests
{
    [Test]
    0 references
    public void Test_ExtractSubstringBetweenMarkers_SubstringFound_ReturnsExtractedSubstring()...

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

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

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

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

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

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


```
◢ ✓ SubstringExtractorTests (6)
    ✓ Test_ExtractSubstringBetweenMarkers_EmptyInput_ReturnsNotFoundMessage
    ✓ Test_ExtractSubstringBetweenMarkers_EndMarkerNotFound_ReturnsNotFoundMessage
    ✓ Test_ExtractSubstringBetweenMarkers_StartAndEndMarkersNotFound_ReturnsNotFoundMessage
    ✓ Test_ExtractSubstringBetweenMarkers_StartAndEndMarkersOverlapping_ReturnsNotFoundMessage
    ✓ Test_ExtractSubstringBetweenMarkers_StartMarkerNotFound_ReturnsNotFoundMessage
    ✓ Test_ExtractSubstringBetweenMarkers_SubstringFound_ReturnsExtractedSubstring
```

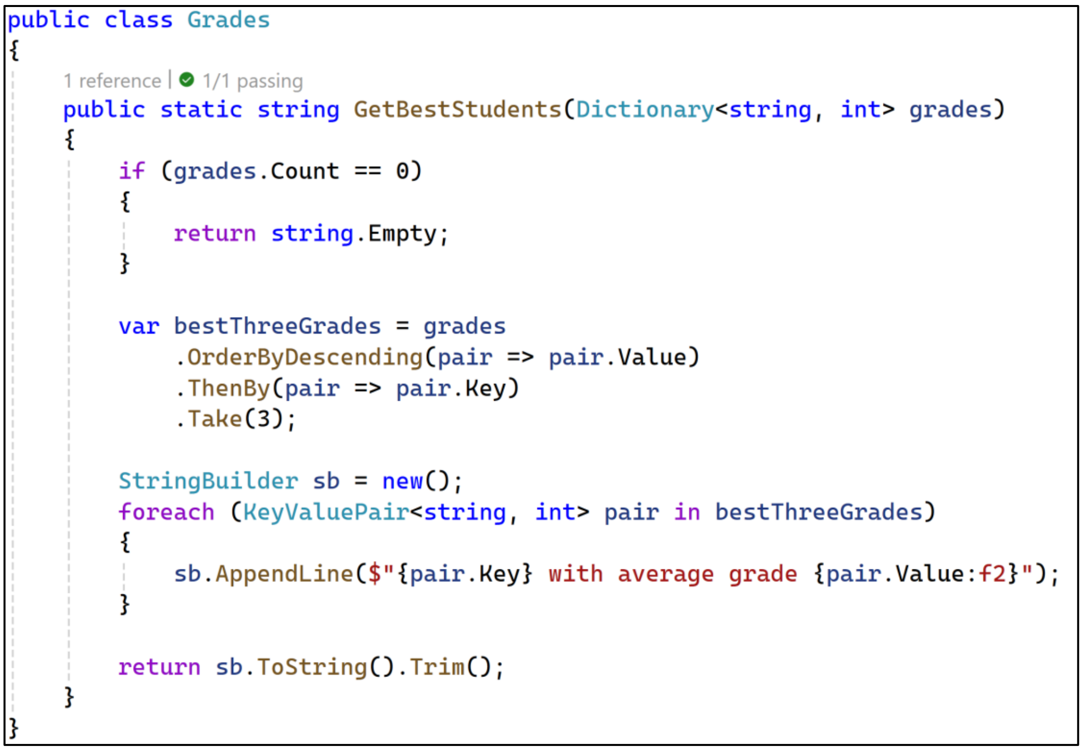
IMPORTANT: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.

2. Unit Test: Grades

Test a given method which takes in **a dictionary of strings representing names** and an **integer representing grade** and **prints the 3 best grades in the form of**:

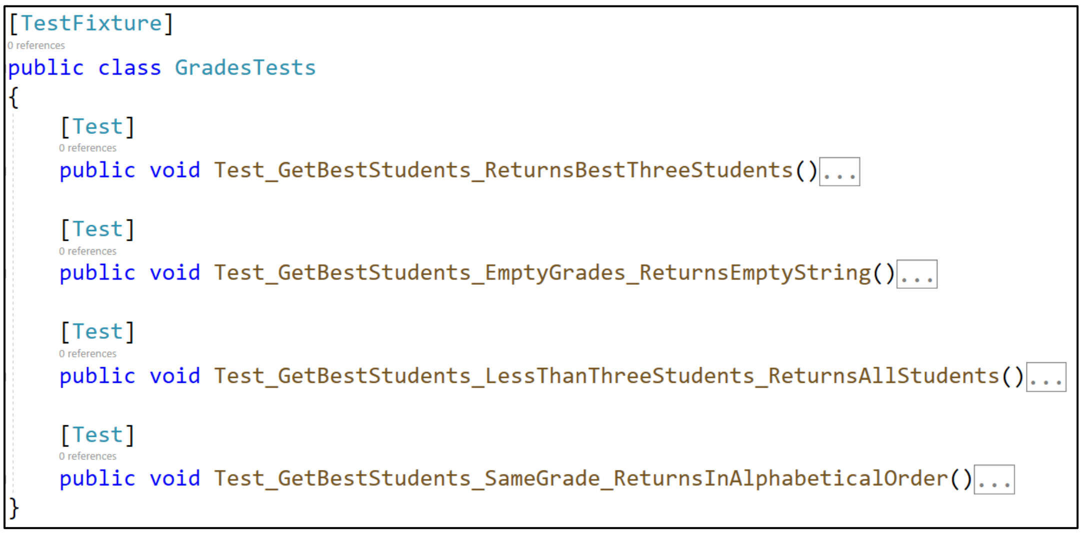
"{name} with average grade {grade}"

The method is found in the **Grades.cs** file:



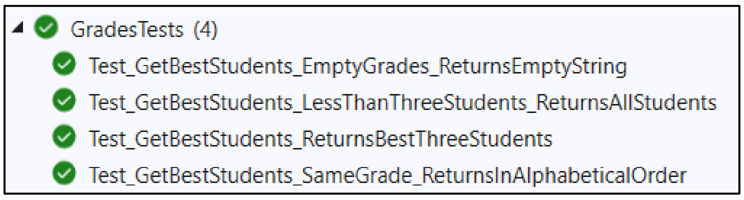
```
public class Grades
{
    1 reference | ✓ 1/1 passing
    public static string GetBestStudents(Dictionary<string, int> grades)
    {
        if (grades.Count == 0)
        {
            return string.Empty;
        }

        var bestThreeGrades = grades
            .OrderByDescending(pair => pair.Value)
            .ThenBy(pair => pair.Key)
            .Take(3);

        StringBuilder sb = new();
        foreach (KeyValuePair<string, int> pair in bestThreeGrades)
        {
            sb.AppendLine($"{pair.Key} with average grade {pair.Value:f2}");
        }

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

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



```
[TestFixture]
0 references
public class GradesTests
{
    [Test]
    0 references
    public void Test_GetBestStudents_ReturnsBestThreeStudents()...

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

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

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

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

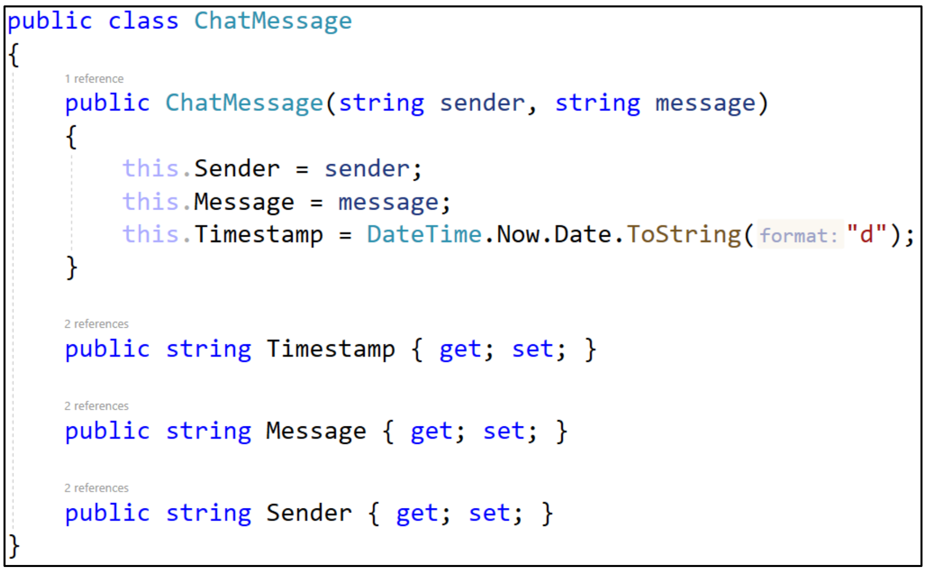


```
◢ ✓ GradesTests (4)
    ✓ Test_GetBestStudents_EmptyGrades_ReturnsEmptyString
    ✓ Test_GetBestStudents_LessThanThreeStudents_ReturnsAllStudents
    ✓ Test_GetBestStudents_ReturnsBestThreeStudents
    ✓ Test_GetBestStudents_SameGrade_ReturnsInAlphabeticalOrder
```

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

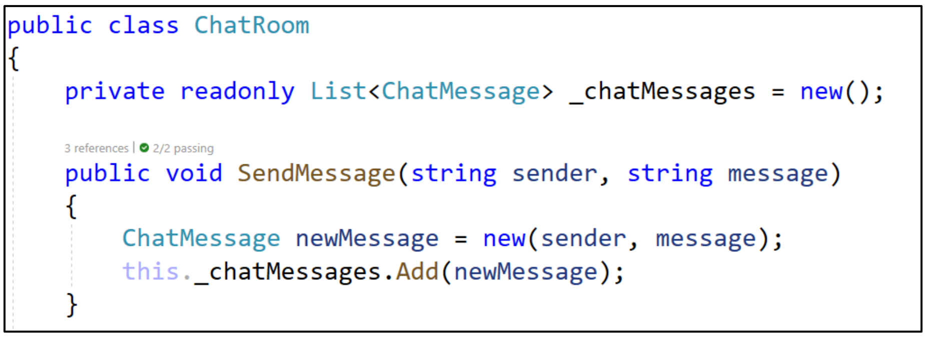
3. Unit Test: Chat

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

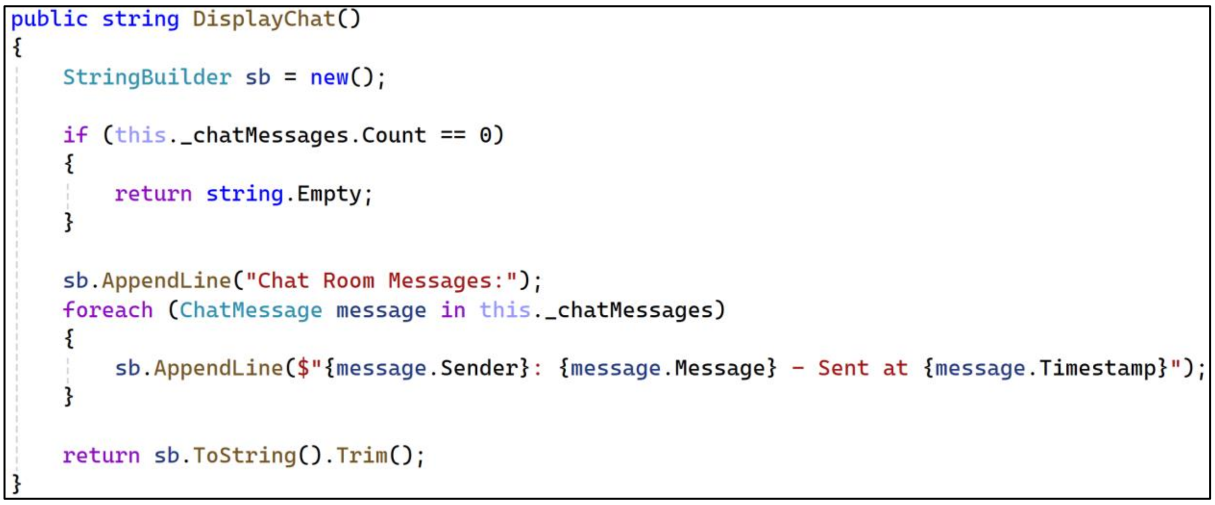


```
public class ChatMessage
{
    1 reference
    public ChatMessage(string sender, string message)
    {
        this.Sender = sender;
        this.Message = message;
        this.Timestamp = DateTime.Now.Date.ToString(format: "d");
    }

    2 references
    public string Timestamp { get; set; }

    2 references
    public string Message { get; set; }

    2 references
    public string Sender { get; set; }
}
```

The **`ChatRoom`** class holds a **list** and **methods** for **using the collection** that you will **test**:

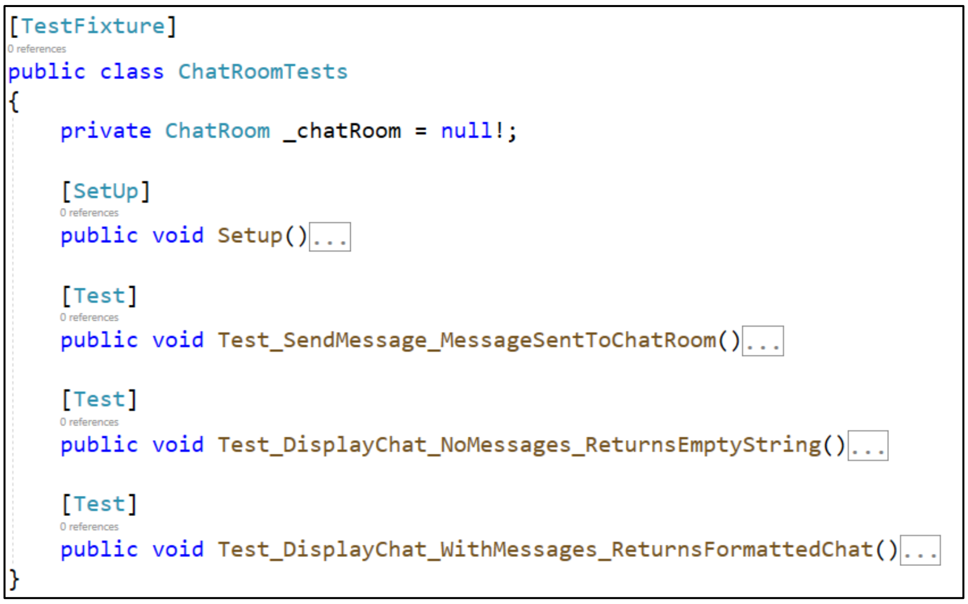


```
public class ChatRoom
{
    private readonly List<ChatMessage> _chatMessages = new();

    3 references | 2/2 passing
    public void SendMessage(string sender, string message)
    {
        ChatMessage newMessage = new(sender, message);
        this._chatMessages.Add(newMessage);
    }
```



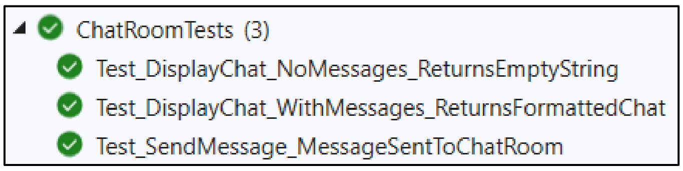
```
public string DisplayChat()
{
    StringBuilder sb = new();

    if (this._chatMessages.Count == 0)
    {
        return string.Empty;
    }

    sb.AppendLine("Chat Room Messages:");
    foreach (ChatMessage message in this._chatMessages)
    {
        sb.AppendLine($"{message.Sender}: {message.Message} - Sent at {message.Timestamp}");
    }

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

You will need to use the test file **`ChatRoomTests.cs`**, inside they are **3 empty tests with a setup method:**



```
[TestFixture]
0 references
public class ChatRoomTests
{
    private ChatRoom _chatRoom = null!;

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

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

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

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

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



```
◢ ✓ ChatRoomTests (3)
    ✓ Test_DisplayChat_NoMessages_ReturnsEmptyString
    ✓ Test_DisplayChat_WithMessages_ReturnsFormattedChat
    ✓ Test_SendMessage_MessageSentToChatRoom
```

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