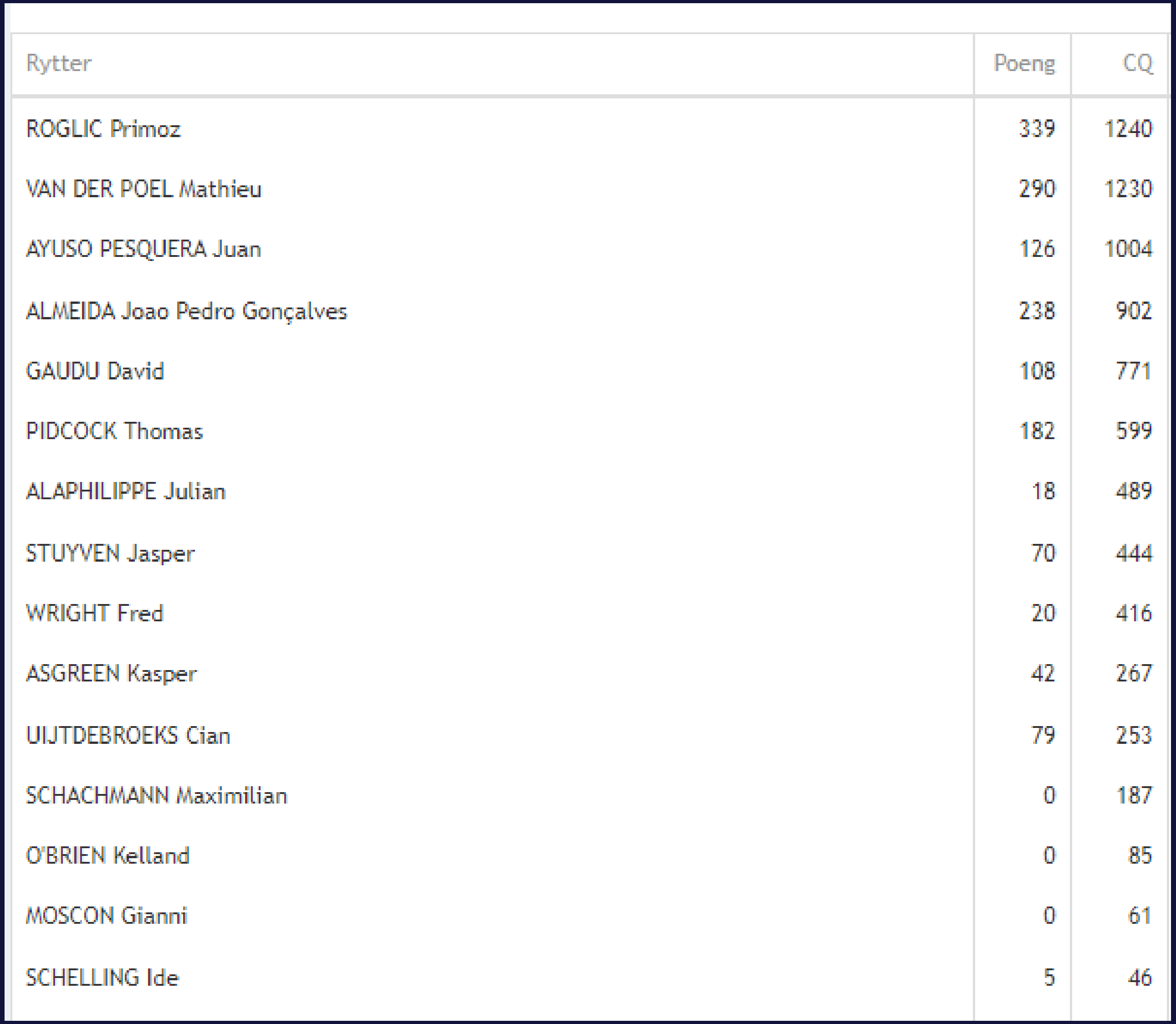
RUNNING MODULE APPLIED IN A PERSONAL PROJECT

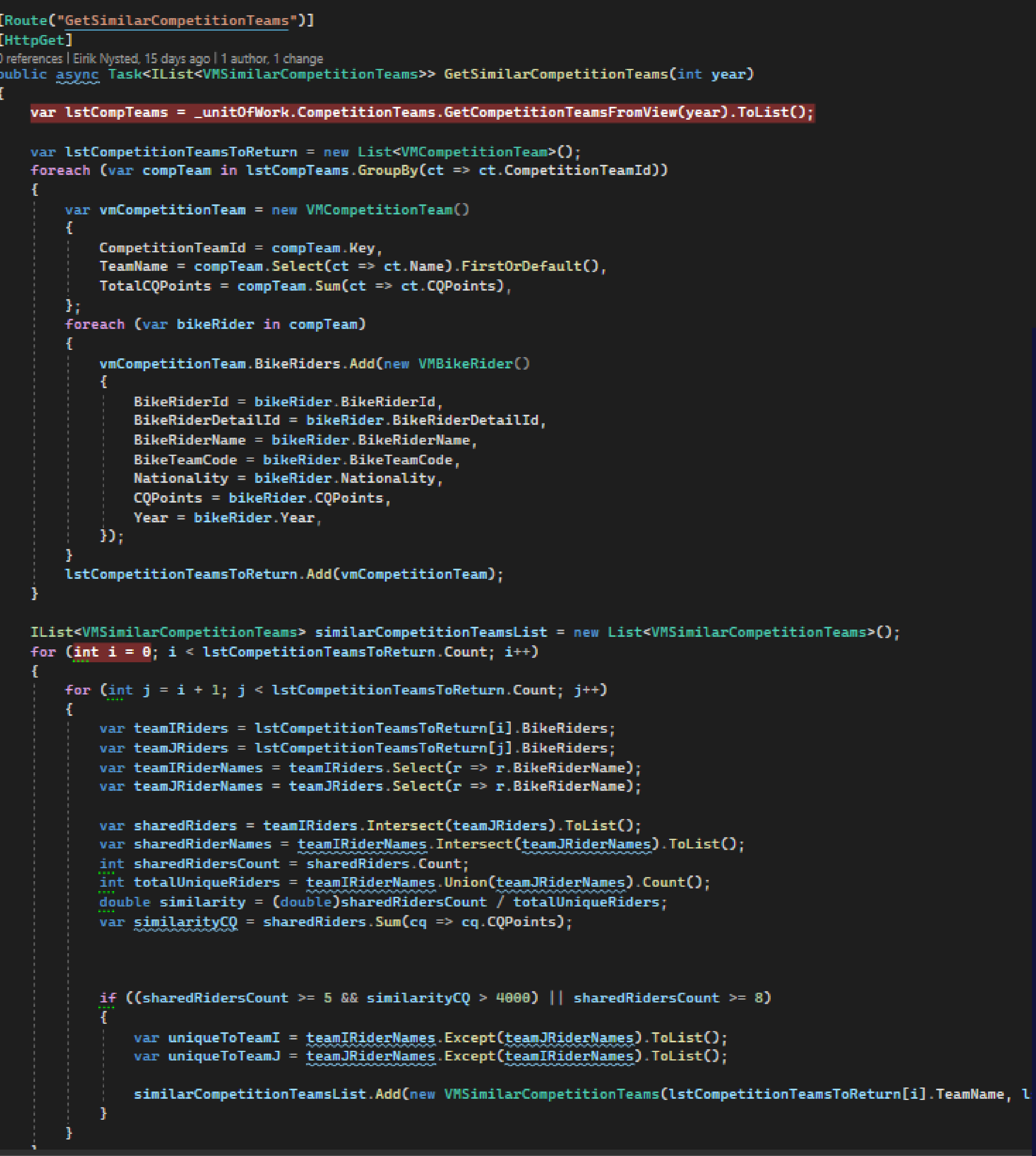
Refactoring, SOLID



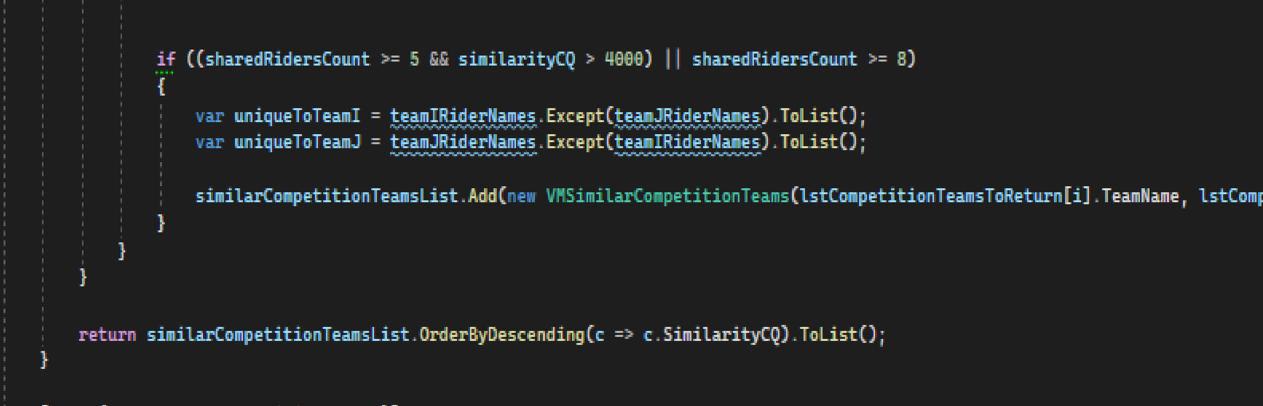
Rytter	Poeng	CQ
ROGLIC Primoz	339	1240
VAN DER POEL Mathieu	290	1230
AYUSO PESQUERA Juan	126	1004
ALMEIDA Joao Pedro Gonçalves	238	902
GAUDU David	108	771
PIDCOCK Thomas	182	599
ALAPHILIPPE Julian	18	489
STUYVEN Jasper	70	444
WRIGHT Fred	20	416
ASGREEN Kasper	42	267
UIJTDEBROEKS Cian	79	253
SCHACHMANN Maximilian	0	187
O'BRIEN Kelland	0	85
MOSCON Gianni	0	61
SCHELLING Ide	5	46

«SYKKELKONKEN»

- Fantasy cycling
- Pick a team based on a budget
- Compare teams to find the most similar ones

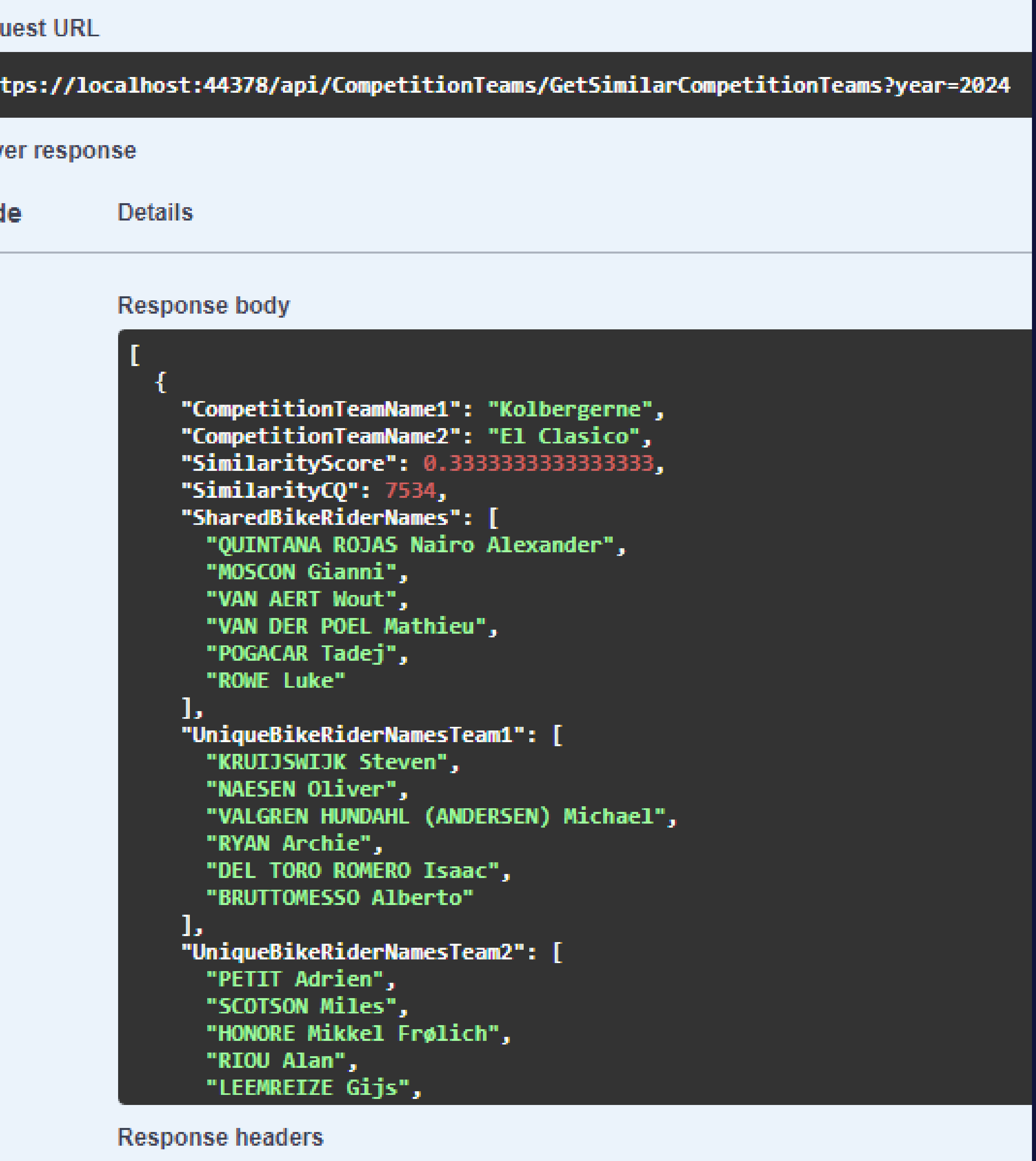


```
[Route("GetSimilarCompetitionTeams")]
[HttpGet]
) references | Eirik Nysted, 15 days ago | 1 author, 1 change
ublic async Task<IList<VMSimilarCompetitionTeams>> GetSimilarCompetitionTeams(int year)
{
    var lstCompTeams = _unitOfWork.CompetitionTeams.GetCompetitionTeamsFromView(year).ToList();

    var lstCompetitionTeamsToReturn = new List<VMCompetitionTeam>();
    foreach (var compTeam in lstCompTeams.GroupBy(ct => ct.CompetitionTeamId))
    {
        var vmCompetitionTeam = new VMCompetitionTeam()
        {
            CompetitionTeamId = compTeam.Key,
            TeamName = compTeam.Select(ct => ct.Name).FirstOrDefault(),
            TotalCQPoints = compTeam.Sum(ct => ct.CQPoints),
        };
        foreach (var bikeRider in compTeam)
        {
            vmCompetitionTeam.BikeRiders.Add(new VMBikeRider()
            {
                BikeRiderId = bikeRider.BikeRiderId,
                BikeRiderDetailId = bikeRider.BikeRiderDetailId,
                BikeRiderName = bikeRider.BikeRiderName,
                BikeTeamCode = bikeRider.BikeTeamCode,
                Nationality = bikeRider.Nationality,
                CQPoints = bikeRider.CQPoints,
                Year = bikeRider.Year,
            });
        }
        lstCompetitionTeamsToReturn.Add(vmCompetitionTeam);
    }

    IList<VMSimilarCompetitionTeams> similarCompetitionTeamsList = new List<VMSimilarCompetitionTeams>();
    for (int i = 0; i < lstCompetitionTeamsToReturn.Count; i++)
    {
        for (int j = i + 1; j < lstCompetitionTeamsToReturn.Count; j++)
        {
            var teamIRiders = lstCompetitionTeamsToReturn[i].BikeRiders;
            var teamJRiders = lstCompetitionTeamsToReturn[j].BikeRiders;
            var teamIRiderNames = teamIRiders.Select(r => r.BikeRiderName);
            var teamJRiderNames = teamJRiders.Select(r => r.BikeRiderName);

            var sharedRiders = teamIRiders.Intersect(teamJRiders).ToList();
            var sharedRiderNames = teamIRiderNames.Intersect(teamJRiderNames).ToList();
            int sharedRidersCount = sharedRiders.Count;
            int totalUniqueRiders = teamIRiderNames.Union(teamJRiderNames).Count();
            double similarity = (double)sharedRidersCount / totalUniqueRiders;
            var similarityCQ = sharedRiders.Sum(cq => cq.CQPoints);



            if ((sharedRidersCount >= 5 && similarityCQ > 4000) || sharedRidersCount >= 8)
            {
                var uniqueToTeamI = teamIRiderNames.Except(teamJRiderNames).ToList();
                var uniqueToTeamJ = teamJRiderNames.Except(teamIRiderNames).ToList();

                similarCompetitionTeamsList.Add(new VMSimilarCompetitionTeams(lstCompetitionTeamsToReturn[i].TeamName, l
            }
        }
```



```
            if ((sharedRidersCount >= 5 && similarityCQ > 4000) || sharedRidersCount >= 8)
            {
                var uniqueToTeamI = teamIRiderNames.Except(teamJRiderNames).ToList();
                var uniqueToTeamJ = teamJRiderNames.Except(teamIRiderNames).ToList();

                similarCompetitionTeamsList.Add(new VMSimilarCompetitionTeams(lstCompetitionTeamsToReturn[i].TeamName, lstComp
            }
        }
    }

    return similarCompetitionTeamsList.OrderByDescending(c => c.SimilarityCQ).ToList();
}
```

Not exactly SOLID



uest URL

```
tps://localhost:44378/api/CompetitionTeams/GetSimilarCompetitionTeams?year=2024
```

ver response

de Details

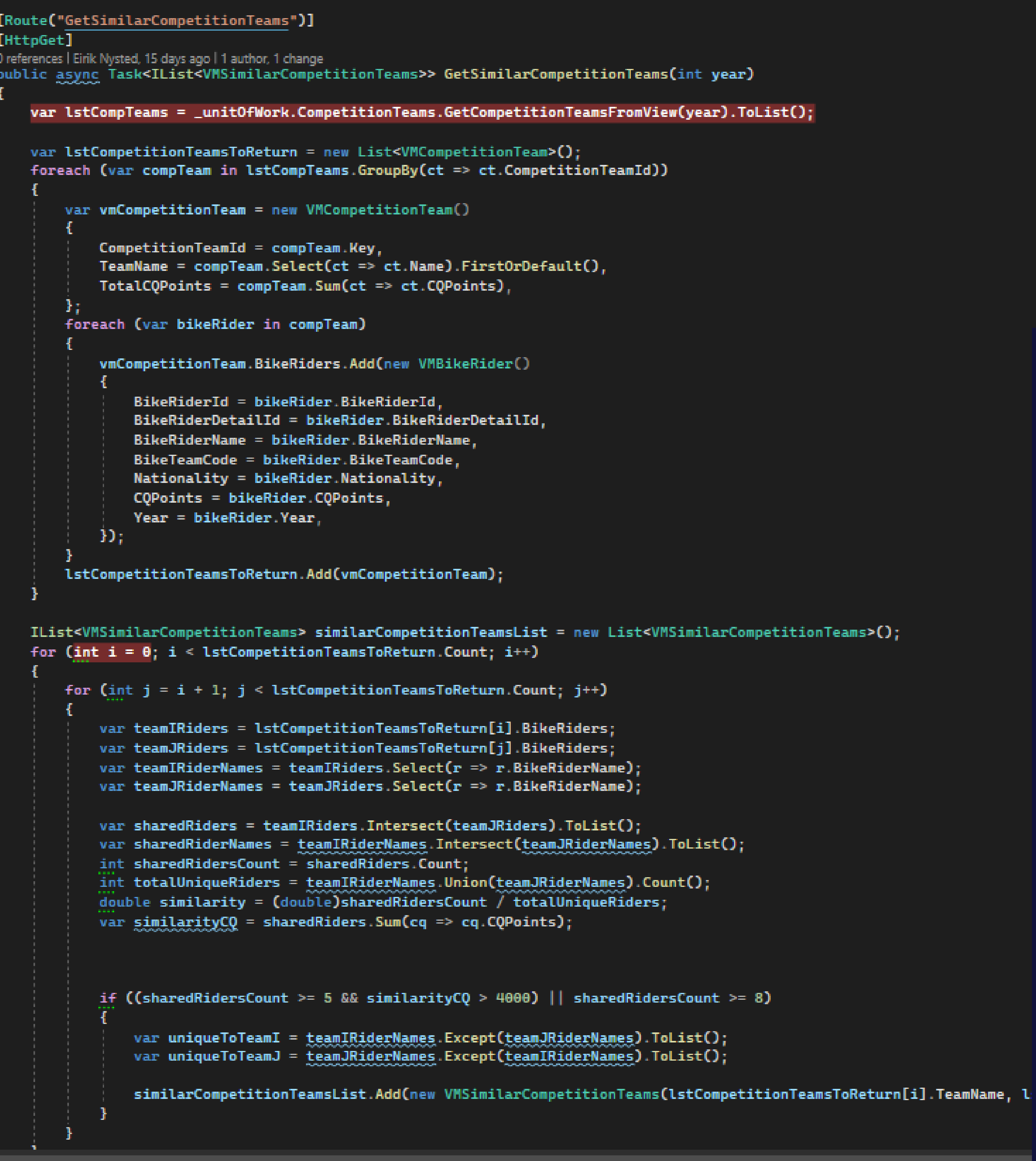
Response body

```
[
  {
    "CompetitionTeamName1": "Kolbergerne",
    "CompetitionTeamName2": "El Clasico",
    "SimilarityScore": 0.3333333333333333,
    "SimilarityCQ": 7534,
    "SharedBikeRiderNames": [
      "QUINTANA ROJAS Nairo Alexander",
      "MOSCON Gianni",
      "VAN AERT Wout",
      "VAN DER POEL Mathieu",
      "POGACAR Tadej",
      "ROWE Luke"
    ],
    "UniqueBikeRiderNamesTeam1": [
      "KRUIJSWIJK Steven",
      "NAESEN Oliver",
      "VALGREN HUNDAHL (ANDERSEN) Michael",
      "RYAN Archie",
      "DEL TORO ROMERO Isaac",
      "BRUTTOMESSO Alberto"
    ],
    "UniqueBikeRiderNamesTeam2": [
      "PETIT Adrien",
      "SCOTSON Miles",
      "HONORE Mikkel Frølich",
      "RIOU Alan",
      "LEEMREIZE Gijs",
```

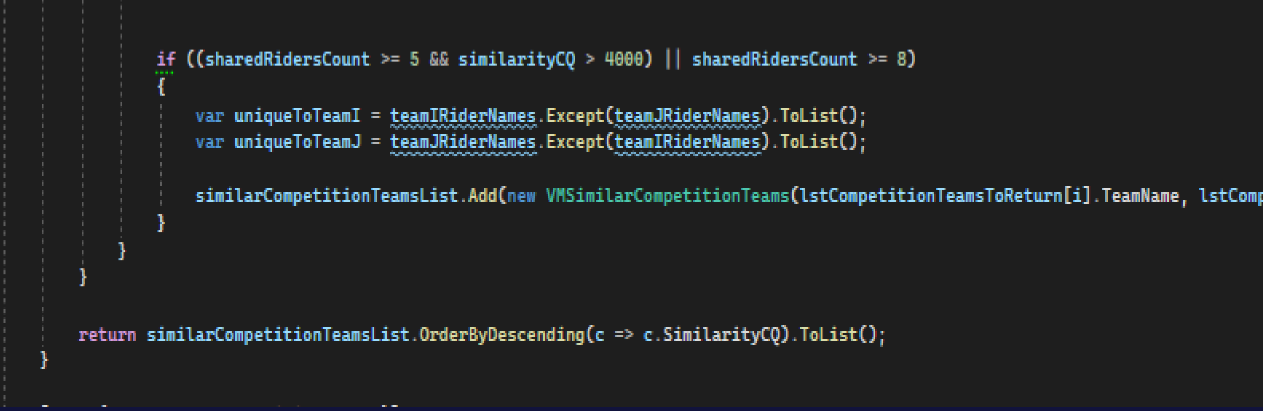
Response headers

Swagger output

- Similar teams above a score threshold
- Sorted by SimilarityCQ



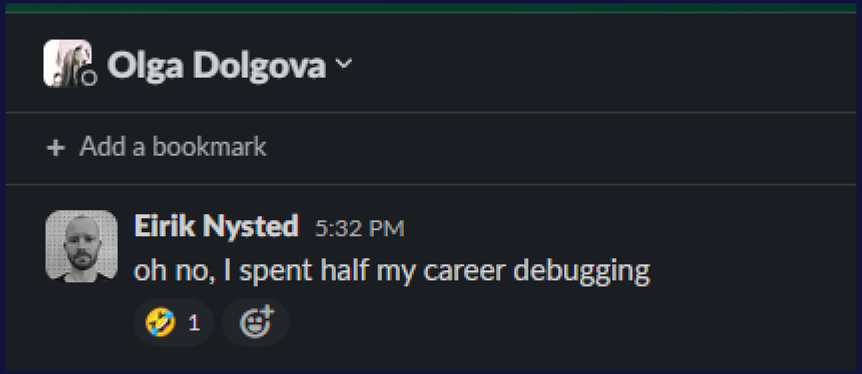
```
[Route("GetSimilarCompetitionTeams")]
[HttpGet]
) references | Eirik Nysted, 15 days ago | 1 author, 1 change
ublic async Task<IList<VMSimilarCompetitionTeams>> GetSimilarCompetitionTeams(int year)

    var lstCompTeams = _unitOfWork.CompetitionTeams.GetCompetitionTeamsFromView(year).ToList();

    var lstCompetitionTeamsToReturn = new List<VMCompetitionTeam>();
    foreach (var compTeam in lstCompTeams.GroupBy(ct => ct.CompetitionTeamId))
    {
        var vmCompetitionTeam = new VMCompetitionTeam()
        {
            CompetitionTeamId = compTeam.Key,
            TeamName = compTeam.Select(ct => ct.Name).FirstOrDefault(),
            TotalCQPoints = compTeam.Sum(ct => ct.CQPoints),
        };
        foreach (var bikeRider in compTeam)
        {
            vmCompetitionTeam.BikeRiders.Add(new VMBikeRider()
            {
                BikeRiderId = bikeRider.BikeRiderId,
                BikeRiderDetailId = bikeRider.BikeRiderDetailId,
                BikeRiderName = bikeRider.BikeRiderName,
                BikeTeamCode = bikeRider.BikeTeamCode,
                Nationality = bikeRider.Nationality,
                CQPoints = bikeRider.CQPoints,
                Year = bikeRider.Year,
            });
        }
        lstCompetitionTeamsToReturn.Add(vmCompetitionTeam);
    }

    IList<VMSimilarCompetitionTeams> similarCompetitionTeamsList = new List<VMSimilarCompetitionTeams>();
    for (int i = 0; i < lstCompetitionTeamsToReturn.Count; i++)
    {
        for (int j = i + 1; j < lstCompetitionTeamsToReturn.Count; j++)
        {
            var teamIRiders = lstCompetitionTeamsToReturn[i].BikeRiders;
            var teamJRiders = lstCompetitionTeamsToReturn[j].BikeRiders;
            var teamIRiderNames = teamIRiders.Select(r => r.BikeRiderName);
            var teamJRiderNames = teamJRiders.Select(r => r.BikeRiderName);

            var sharedRiders = teamIRiders.Intersect(teamJRiders).ToList();
            var sharedRiderNames = teamIRiderNames.Intersect(teamJRiderNames).ToList();
            int sharedRidersCount = sharedRiders.Count;
            int totalUniqueRiders = teamIRiderNames.Union(teamJRiderNames).Count();
            double similarity = (double)sharedRidersCount / totalUniqueRiders;
            var similarityCQ = sharedRiders.Sum(cq => cq.CQPoints);



            if ((sharedRidersCount >= 5 && similarityCQ > 4000) || sharedRidersCount >= 8)
            {
                var uniqueToTeamI = teamIRiderNames.Except(teamJRiderNames).ToList();
                var uniqueToTeamJ = teamJRiderNames.Except(teamIRiderNames).ToList();

                similarCompetitionTeamsList.Add(new VMSimilarCompetitionTeams(lstCompetitionTeamsToReturn[i].TeamName, l
            }
        }
```



```
            if ((sharedRidersCount >= 5 && similarityCQ > 4000) || sharedRidersCount >= 8)
            {
                var uniqueToTeamI = teamIRiderNames.Except(teamJRiderNames).ToList();
                var uniqueToTeamJ = teamJRiderNames.Except(teamIRiderNames).ToList();

                similarCompetitionTeamsList.Add(new VMSimilarCompetitionTeams(lstCompetitionTeamsToReturn[i].TeamName, lstComp
            }
        }
    }

    return similarCompetitionTeamsList.OrderByDescending(c => c.SimilarityCQ).ToList();
}
```

A lot of WTFs per minute

A lot of debugging

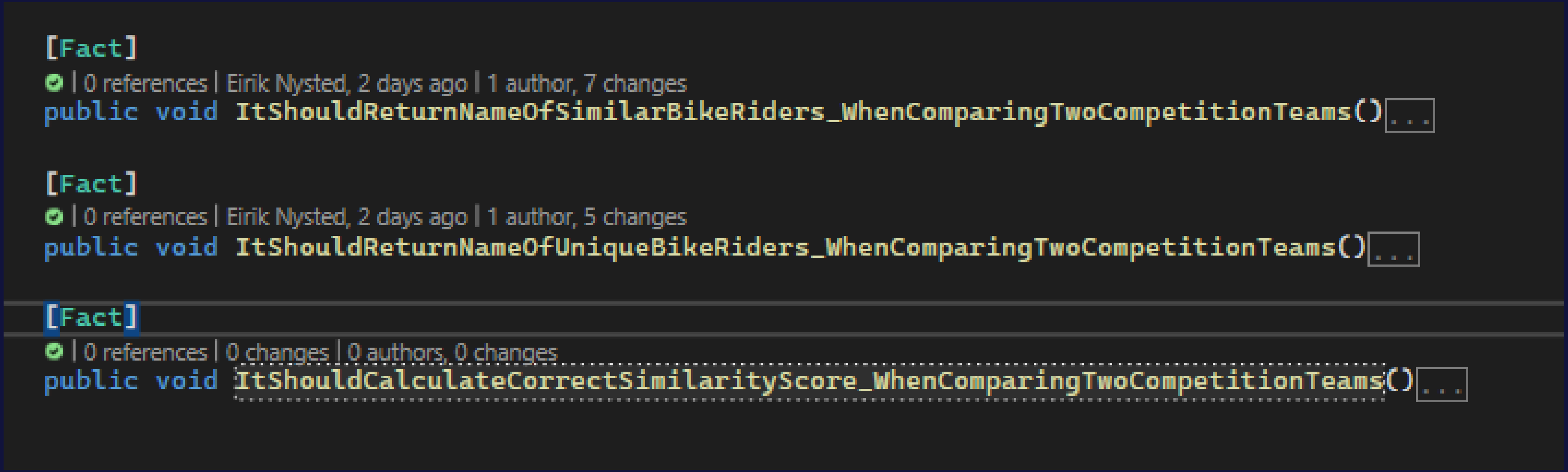


Olga Dolgova

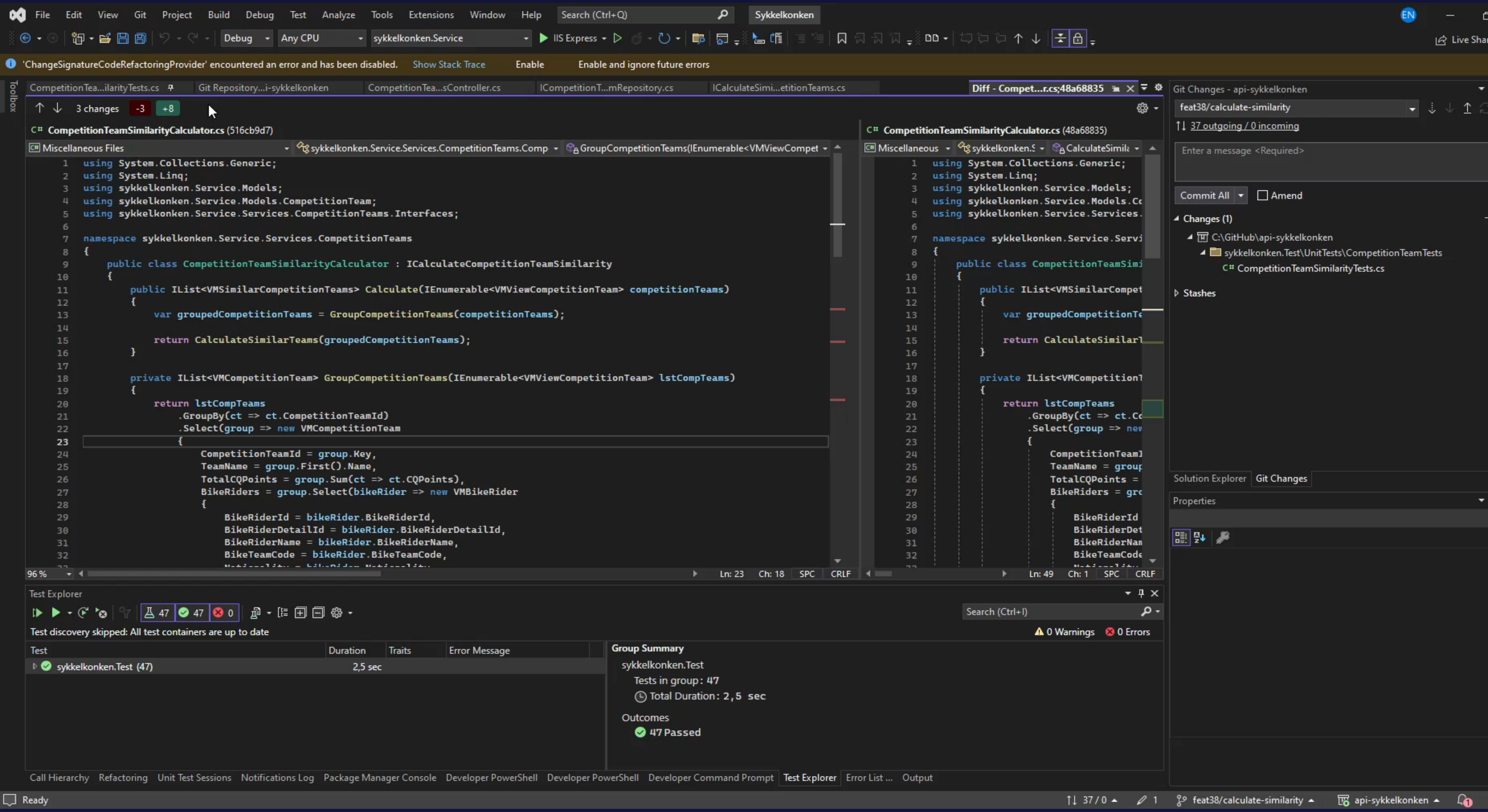
+ Add a bookmark

Eirik Nysted 5:32 PM
oh no, I spent half my career debugging
🤣 1

Comparing two teams



```
[Fact]
✓ | 0 references | Eirik Nysted, 2 days ago | 1 author, 7 changes
public void ItShouldReturnNameOfSimilarBikeRiders_WhenComparingTwoCompetitionTeams()...

[Fact]
✓ | 0 references | Eirik Nysted, 2 days ago | 1 author, 5 changes
public void ItShouldReturnNameOfUniqueBikeRiders_WhenComparingTwoCompetitionTeams()...

[Fact]
✓ | 0 references | 0 changes | 0 authors, 0 changes
public void ItShouldCalculateCorrectSimilarityScore_WhenComparingTwoCompetitionTeams()...
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Sykkelkonken
EN
Debug
Any CPU
sykkelkonken.Service
IIS Express
Live Share

'ChangeSignatureCodeRefactoringProvider' encountered an error and has been disabled. Show Stack Trace Enable Enable and ignore future errors

CompetitionTea...ilarityTests.cs
Git Repository...i-sykkelkonken
CompetitionTea...sController.cs
ICompetitionT...mRepository.cs
ICalculateSimi...etitionTeams.cs
Diff - Compet...r.cs;48a68835
3 changes -3 +8

CompetitionTeamSimilarityCalculator.cs (516cb9d7)

Miscellaneous Files | sykkelkonken.Service.Services.CompetitionTeams.Comp | GroupCompetitionTeams(IEnumerable<VMViewCompet

```
using System.Collections.Generic;
using System.Linq;
using sykkelkonken.Service.Models;
using sykkelkonken.Service.Models.CompetitionTeam;
using sykkelkonken.Service.Services.CompetitionTeams.Interfaces;

namespace sykkelkonken.Service.Services.CompetitionTeams
{
    public class CompetitionTeamSimilarityCalculator : ICalculateCompetitionTeamSimilarity
    {
        public IList<VMSimilarCompetitionTeams> Calculate(IEnumerable<VMViewCompetitionTeam> competitionTeams)
        {
            var groupedCompetitionTeams = GroupCompetitionTeams(competitionTeams);

            return CalculateSimilarTeams(groupedCompetitionTeams);
        }

        private IList<VMCompetitionTeam> GroupCompetitionTeams(IEnumerable<VMViewCompetitionTeam> lstCompTeams)
        {
            return lstCompTeams
                .GroupBy(ct => ct.CompetitionTeamId)
                .Select(group => new VMCompetitionTeam
                {
                    CompetitionTeamId = group.Key,
                    TeamName = group.First().Name,
                    TotalCQPoints = group.Sum(ct => ct.CQPoints),
                    BikeRiders = group.Select(bikeRider => new VMBikeRider
                    {
                        BikeRiderId = bikeRider.BikeRiderId,
                        BikeRiderDetailId = bikeRider.BikeRiderDetailId,
                        BikeRiderName = bikeRider.BikeRiderName,
                        BikeTeamCode = bikeRider.BikeTeamCode,
```

96 % Ln: 23 Ch: 18 SPC CRLF

CompetitionTeamSimilarityCalculator.cs (48a68835)

Miscellaneous | sykkelkonken.S | CalculateSimil

```
using System.Collections.Generic;
using System.Linq;
using sykkelkonken.Service.Models;
using sykkelkonken.Service.Models.Cc
using sykkelkonken.Service.Services.

namespace sykkelkonken.Service.Servi
{
    public class CompetitionTeamSimi
    {
        public IList<VMSimilarCompet
        {
            var groupedCompetitionTe

            return CalculateSimilarT
        }

        private IList<VMCompetitionT
        {
            return lstCompTeams
                .GroupBy(ct => ct.Cc
                .Select(group => new
                {
                    CompetitionTeamI
                    TeamName = group
                    TotalCQPoints =
                    BikeRiders = grc
                    {
                        BikeRiderId
                        BikeRiderDet
                        BikeRiderNam
                        BikeTeamCode
```

Ln: 49 Ch: 1 SPC CRLF

Git Changes - api-sykkelkonken

feat38/calculate-similarity

37 outgoing / 0 incoming

Enter a message <Required>

Commit All   Amend

Changes (1)
- C:\GitHub\api-sykkelkonken
  - sykkelkonken.Test\UnitTests\CompetitionTeamTests
    - CompetitionTeamSimilarityTests.cs

Stashes

Test Explorer

Test discovery skipped: All test containers are up to date

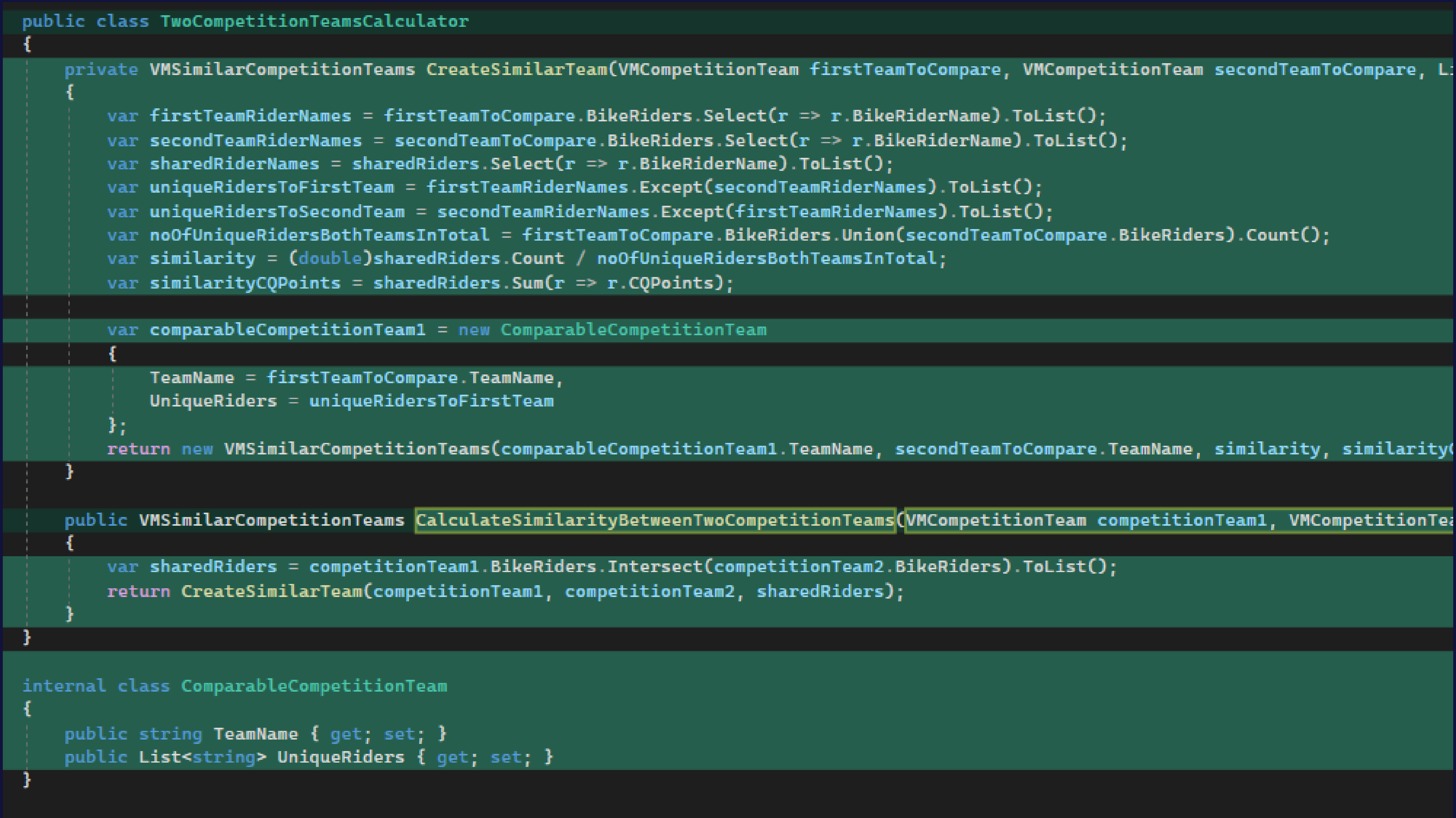
0 Warnings 0 Errors

Test	Duration	Traits	Error Message
sykkelkonken.Test (47)	2,5 sec		

Group Summary
sykkelkonken.Test
Tests in group: 47
Total Duration: 2,5 sec
Outcomes
47 Passed

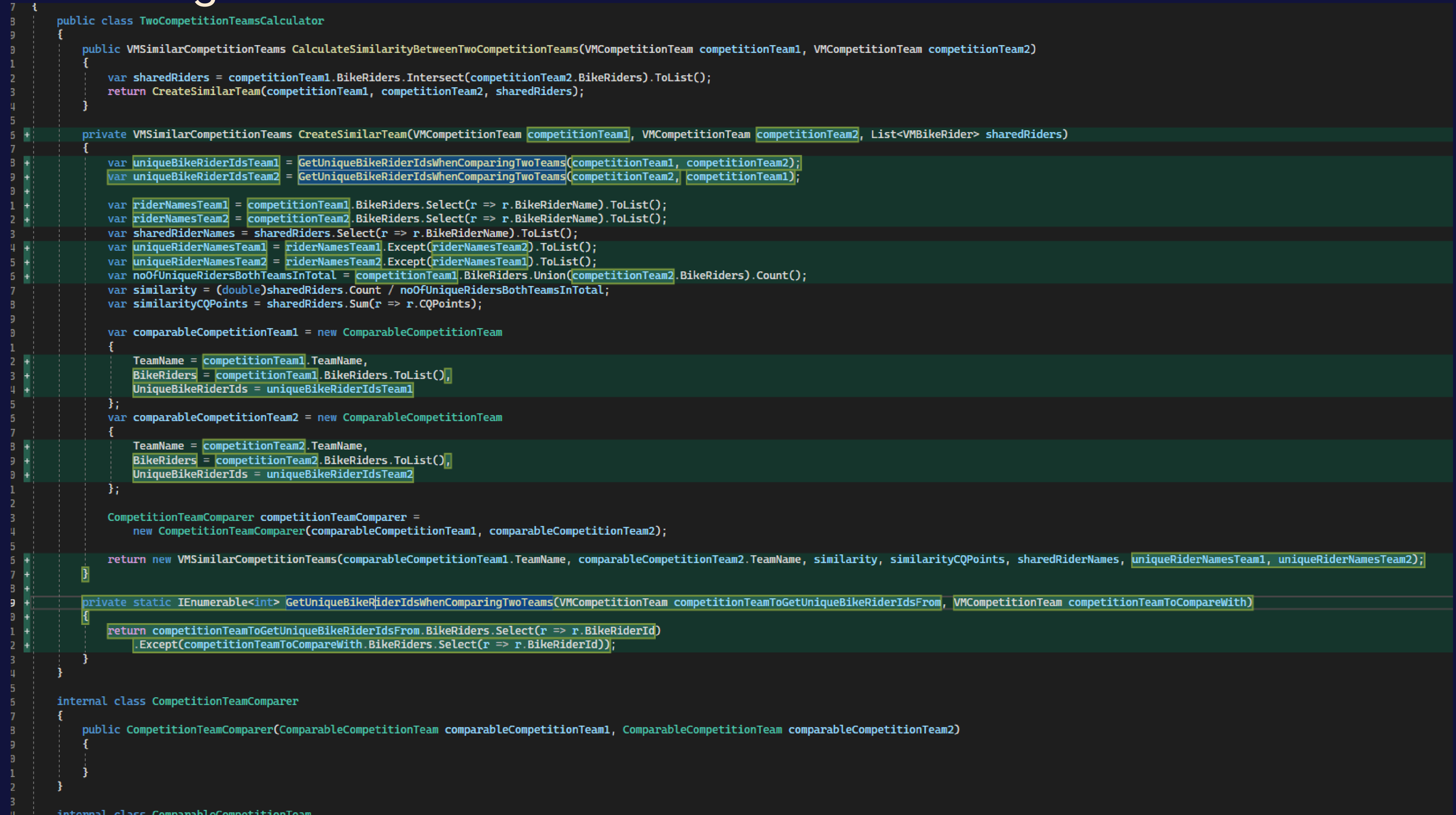
Call Hierarchy Refactoring Unit Test Sessions Notifications Log Package Manager Console Developer PowerShell Developer PowerShell Developer Command Prompt Test Explorer Error List ... Output
Ready
37 / 0
1
feat38/calculate-similarity
api-sykkelkonken

Moving to its own class

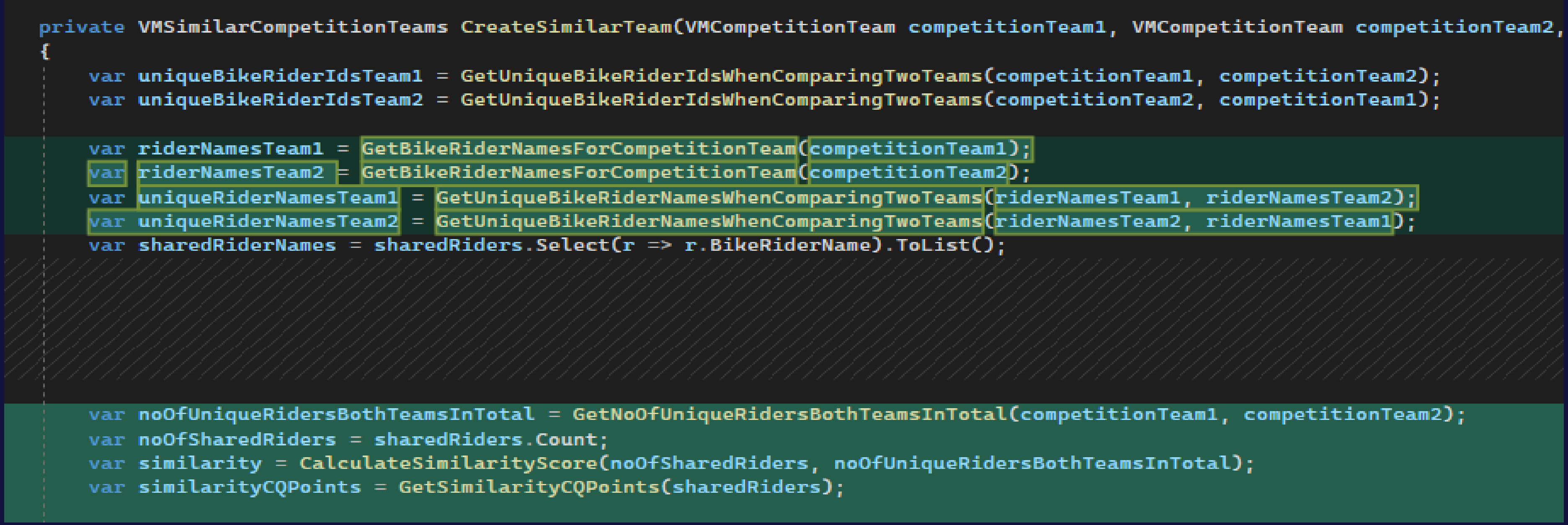


```
public class TwoCompetitionTeamsCalculator
{
    private VMSimilarCompetitionTeams CreateSimilarTeam(VMCompetitionTeam firstTeamToCompare, VMCompetitionTeam secondTeamToCompare, Li
    {
        var firstTeamRiderNames = firstTeamToCompare.BikeRiders.Select(r => r.BikeRiderName).ToList();
        var secondTeamRiderNames = secondTeamToCompare.BikeRiders.Select(r => r.BikeRiderName).ToList();
        var sharedRiderNames = sharedRiders.Select(r => r.BikeRiderName).ToList();
        var uniqueRidersToFirstTeam = firstTeamRiderNames.Except(secondTeamRiderNames).ToList();
        var uniqueRidersToSecondTeam = secondTeamRiderNames.Except(firstTeamRiderNames).ToList();
        var noOfUniqueRidersBothTeamsInTotal = firstTeamToCompare.BikeRiders.Union(secondTeamToCompare.BikeRiders).Count();
        var similarity = (double)sharedRiders.Count / noOfUniqueRidersBothTeamsInTotal;
        var similarityCQPoints = sharedRiders.Sum(r => r.CQPoints);

        var comparableCompetitionTeam1 = new ComparableCompetitionTeam
        {
            TeamName = firstTeamToCompare.TeamName,
            UniqueRiders = uniqueRidersToFirstTeam
        };
        return new VMSimilarCompetitionTeams(comparableCompetitionTeam1.TeamName, secondTeamToCompare.TeamName, similarity, similarity
    }

    public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam competitionTeam1, VMCompetitionTea
    {
        var sharedRiders = competitionTeam1.BikeRiders.Intersect(competitionTeam2.BikeRiders).ToList();
        return CreateSimilarTeam(competitionTeam1, competitionTeam2, sharedRiders);
    }
}

internal class ComparableCompetitionTeam
{
    public string TeamName { get; set; }
    public List<string> UniqueRiders { get; set; }
}
```

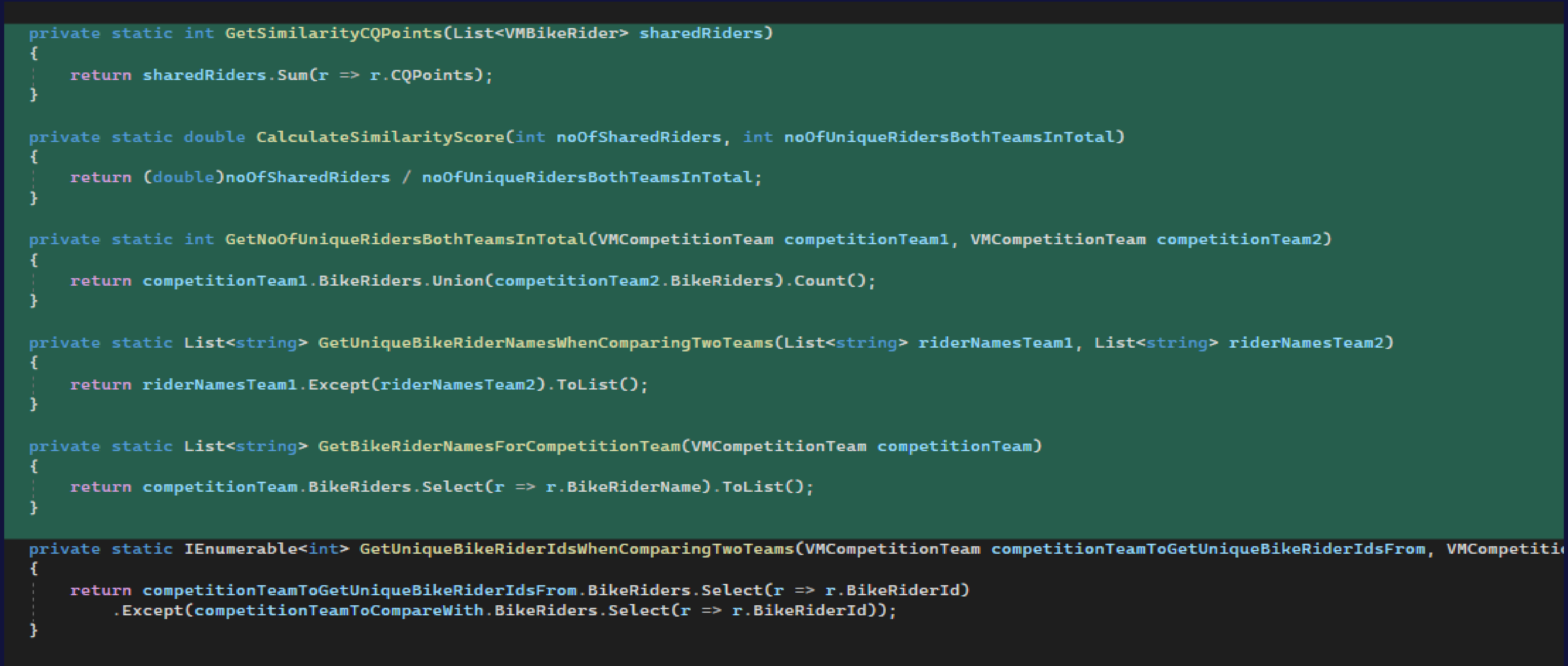
Renaming



```
{
    public class TwoCompetitionTeamsCalculator
    {
        public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2)
        {
            var sharedRiders = competitionTeam1.BikeRiders.Intersect(competitionTeam2.BikeRiders).ToList();
            return CreateSimilarTeam(competitionTeam1, competitionTeam2, sharedRiders);
        }

        private VMSimilarCompetitionTeams CreateSimilarTeam(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2, List<VMBikeRider> sharedRiders)
        {
            var uniqueBikeRiderIdsTeam1 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(competitionTeam1, competitionTeam2);
            var uniqueBikeRiderIdsTeam2 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(competitionTeam2, competitionTeam1);

            var riderNamesTeam1 = competitionTeam1.BikeRiders.Select(r => r.BikeRiderName).ToList();
            var riderNamesTeam2 = competitionTeam2.BikeRiders.Select(r => r.BikeRiderName).ToList();
            var sharedRiderNames = sharedRiders.Select(r => r.BikeRiderName).ToList();
            var uniqueRiderNamesTeam1 = riderNamesTeam1.Except(riderNamesTeam2).ToList();
            var uniqueRiderNamesTeam2 = riderNamesTeam2.Except(riderNamesTeam1).ToList();
            var noOfUniqueRidersBothTeamsInTotal = competitionTeam1.BikeRiders.Union(competitionTeam2.BikeRiders).Count();
            var similarity = (double)sharedRiders.Count / noOfUniqueRidersBothTeamsInTotal;
            var similarityCQPoints = sharedRiders.Sum(r => r.CQPoints);

            var comparableCompetitionTeam1 = new ComparableCompetitionTeam
            {
                TeamName = competitionTeam1.TeamName,
                BikeRiders = competitionTeam1.BikeRiders.ToList(),
                UniqueBikeRiderIds = uniqueBikeRiderIdsTeam1
            };
            var comparableCompetitionTeam2 = new ComparableCompetitionTeam
            {
                TeamName = competitionTeam2.TeamName,
                BikeRiders = competitionTeam2.BikeRiders.ToList(),
                UniqueBikeRiderIds = uniqueBikeRiderIdsTeam2
            };

            CompetitionTeamComparer competitionTeamComparer =
                new CompetitionTeamComparer(comparableCompetitionTeam1, comparableCompetitionTeam2);

            return new VMSimilarCompetitionTeams(comparableCompetitionTeam1.TeamName, comparableCompetitionTeam2.TeamName, similarity, similarityCQPoints, sharedRiderNames, uniqueRiderNamesTeam1, uniqueRiderNamesTeam2);
        }

        private static IEnumerable<int> GetUniqueBikeRiderIdsWhenComparingTwoTeams(VMCompetitionTeam competitionTeamToGetUniqueBikeRiderIdsFrom, VMCompetitionTeam competitionTeamToCompareWith)
        {
            return competitionTeamToGetUniqueBikeRiderIdsFrom.BikeRiders.Select(r => r.BikeRiderId)
                .Except(competitionTeamToCompareWith.BikeRiders.Select(r => r.BikeRiderId));
        }
    }

    internal class CompetitionTeamComparer
    {
        public CompetitionTeamComparer(ComparableCompetitionTeam comparableCompetitionTeam1, ComparableCompetitionTeam comparableCompetitionTeam2)
        {

        }
    }

    internal class ComparableCompetitionTeam
```

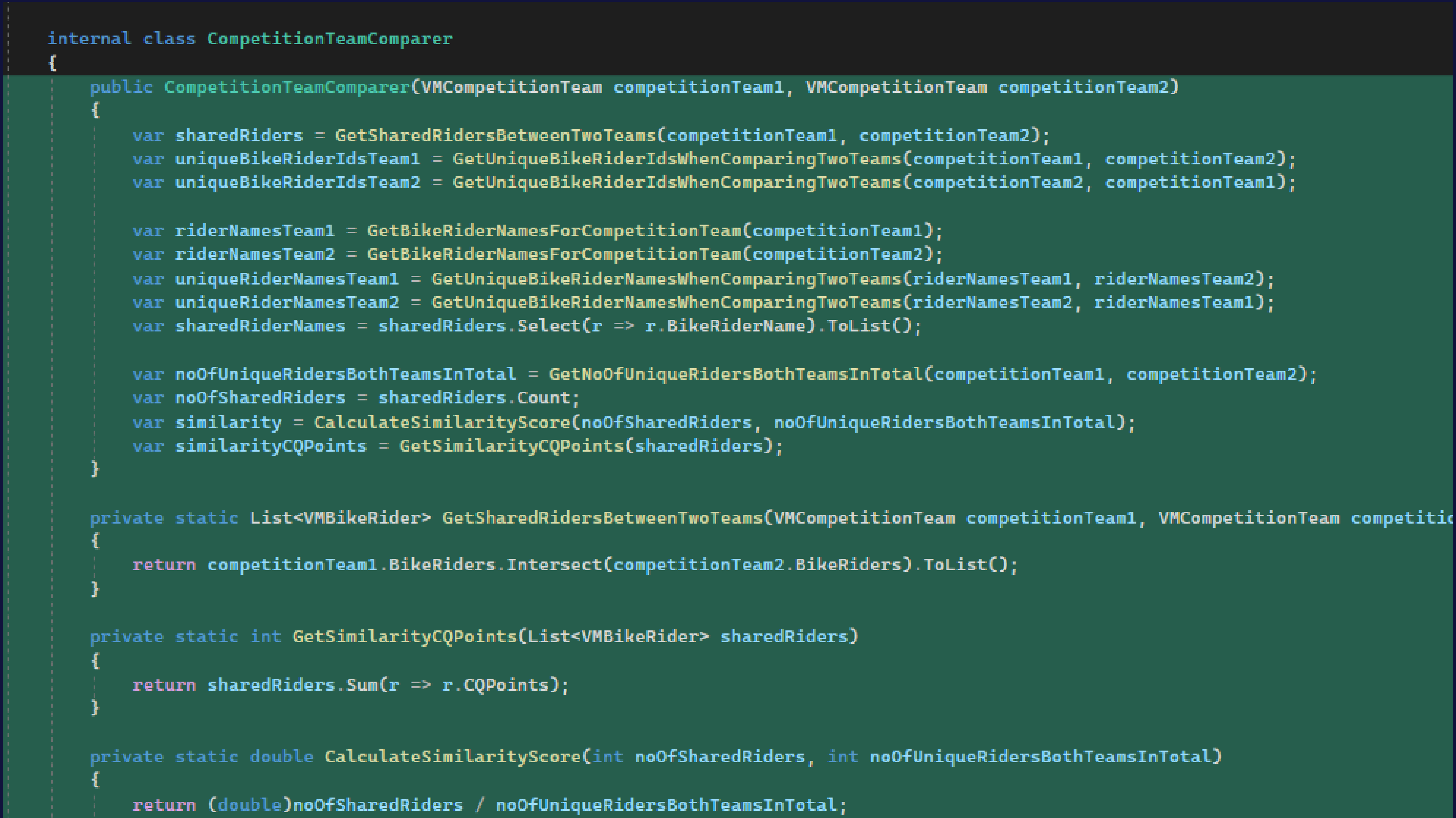


```
private VMSimilarCompetitionTeams CreateSimilarTeam(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2,
{
    var uniqueBikeRiderIdsTeam1 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(competitionTeam1, competitionTeam2);
    var uniqueBikeRiderIdsTeam2 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(competitionTeam2, competitionTeam1);

    var riderNamesTeam1 = GetBikeRiderNamesForCompetitionTeam(competitionTeam1);
    var riderNamesTeam2 = GetBikeRiderNamesForCompetitionTeam(competitionTeam2);
    var uniqueRiderNamesTeam1 = GetUniqueBikeRiderNamesWhenComparingTwoTeams(riderNamesTeam1, riderNamesTeam2);
    var uniqueRiderNamesTeam2 = GetUniqueBikeRiderNamesWhenComparingTwoTeams(riderNamesTeam2, riderNamesTeam1);
    var sharedRiderNames = sharedRiders.Select(r => r.BikeRiderName).ToList();



    var noOfUniqueRidersBothTeamsInTotal = GetNoOfUniqueRidersBothTeamsInTotal(competitionTeam1, competitionTeam2);
    var noOfSharedRiders = sharedRiders.Count;
    var similarity = CalculateSimilarityScore(noOfSharedRiders, noOfUniqueRidersBothTeamsInTotal);
    var similarityCQPoints = GetSimilarityCQPoints(sharedRiders);
```



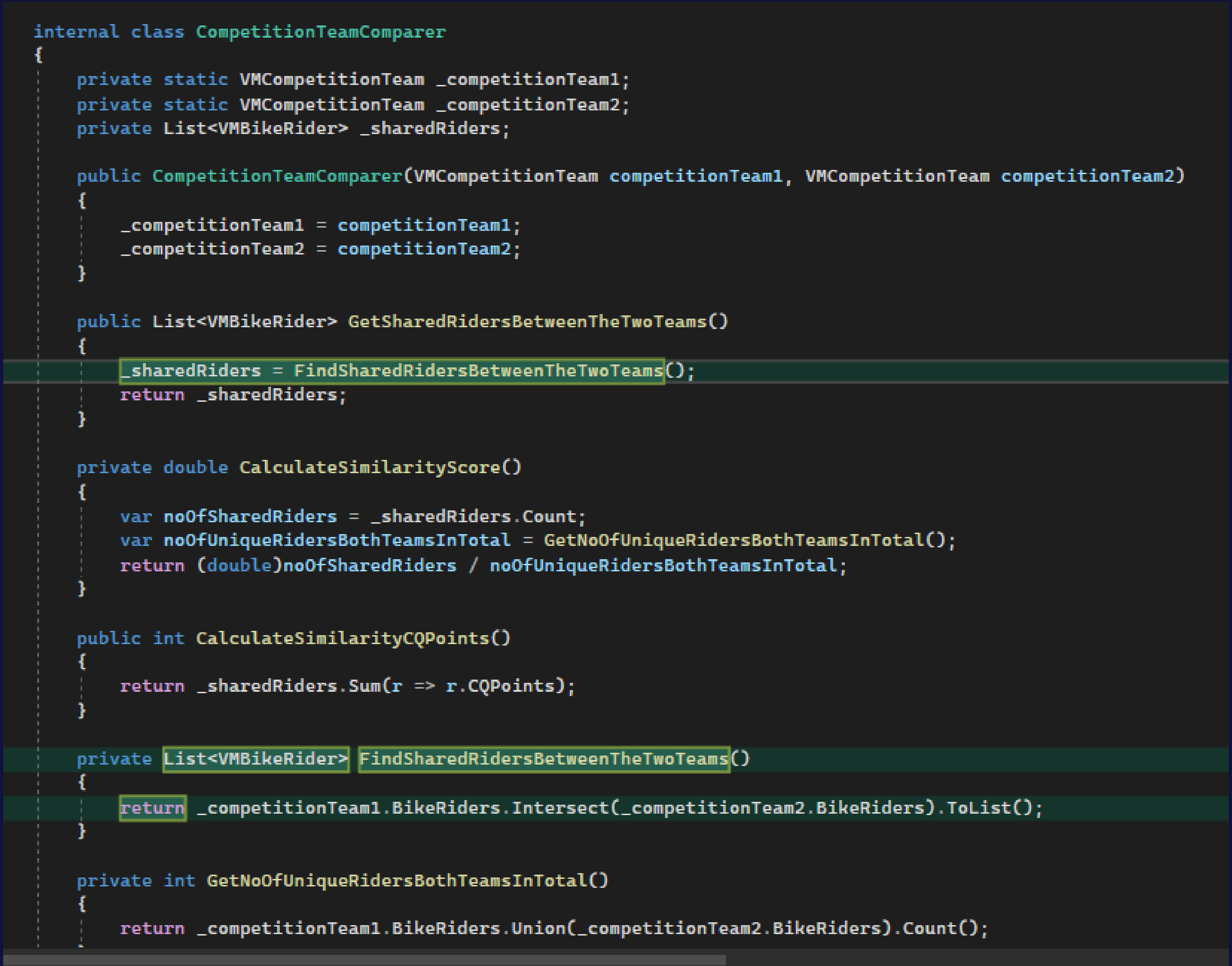
```
private static int GetSimilarityCQPoints(List<VMBikeRider> sharedRiders)
{
    return sharedRiders.Sum(r => r.CQPoints);
}

private static double CalculateSimilarityScore(int noOfSharedRiders, int noOfUniqueRidersBothTeamsInTotal)
{
    return (double)noOfSharedRiders / noOfUniqueRidersBothTeamsInTotal;
}

private static int GetNoOfUniqueRidersBothTeamsInTotal(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2)
{
    return competitionTeam1.BikeRiders.Union(competitionTeam2.BikeRiders).Count();
}

private static List<string> GetUniqueBikeRiderNamesWhenComparingTwoTeams(List<string> riderNamesTeam1, List<string> riderNamesTeam2)
{
    return riderNamesTeam1.Except(riderNamesTeam2).ToList();
}

private static List<string> GetBikeRiderNamesForCompetitionTeam(VMCompetitionTeam competitionTeam)
{
    return competitionTeam.BikeRiders.Select(r => r.BikeRiderName).ToList();
}

private static IEnumerable<int> GetUniqueBikeRiderIdsWhenComparingTwoTeams(VMCompetitionTeam competitionTeamToGetUniqueBikeRiderIdsFrom, VMCompetiti
{
    return competitionTeamToGetUniqueBikeRiderIdsFrom.BikeRiders.Select(r => r.BikeRiderId)
        .Except(competitionTeamToCompareWith.BikeRiders.Select(r => r.BikeRiderId));
}
```

Moved methods to a comparer class



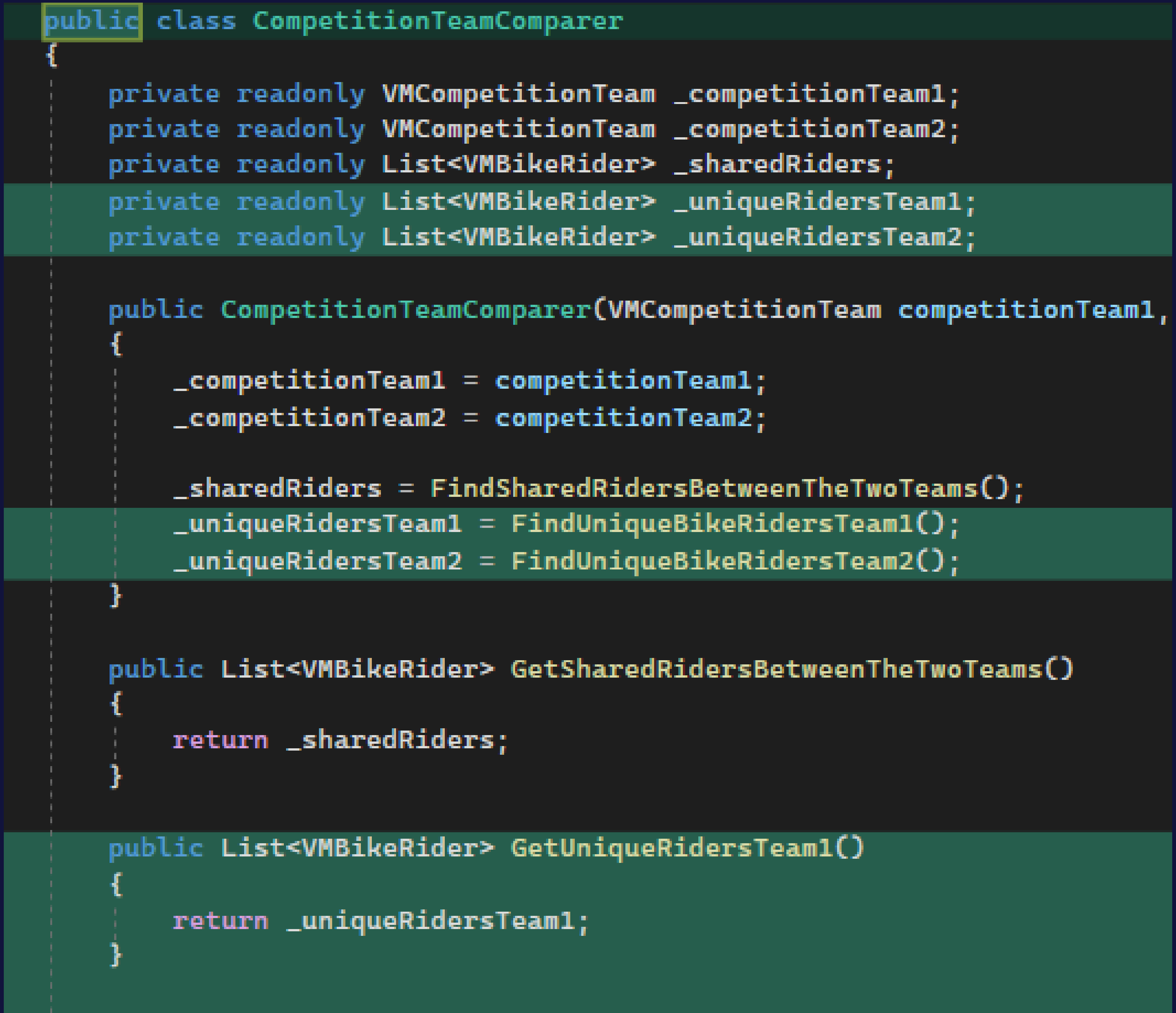
```
internal class CompetitionTeamComparer
{
    public CompetitionTeamComparer(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2)
    {
        var sharedRiders = GetSharedRidersBetweenTwoTeams(competitionTeam1, competitionTeam2);
        var uniqueBikeRiderIdsTeam1 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(competitionTeam1, competitionTeam2);
        var uniqueBikeRiderIdsTeam2 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(competitionTeam2, competitionTeam1);

        var riderNamesTeam1 = GetBikeRiderNamesForCompetitionTeam(competitionTeam1);
        var riderNamesTeam2 = GetBikeRiderNamesForCompetitionTeam(competitionTeam2);
        var uniqueRiderNamesTeam1 = GetUniqueBikeRiderNamesWhenComparingTwoTeams(riderNamesTeam1, riderNamesTeam2);
        var uniqueRiderNamesTeam2 = GetUniqueBikeRiderNamesWhenComparingTwoTeams(riderNamesTeam2, riderNamesTeam1);
        var sharedRiderNames = sharedRiders.Select(r => r.BikeRiderName).ToList();

        var noOfUniqueRidersBothTeamsInTotal = GetNoOfUniqueRidersBothTeamsInTotal(competitionTeam1, competitionTeam2);
        var noOfSharedRiders = sharedRiders.Count;
        var similarity = CalculateSimilarityScore(noOfSharedRiders, noOfUniqueRidersBothTeamsInTotal);
        var similarityCQPoints = GetSimilarityCQPoints(sharedRiders);
    }

    private static List<VMBikeRider> GetSharedRidersBetweenTwoTeams(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitic
    {
        return competitionTeam1.BikeRiders.Intersect(competitionTeam2.BikeRiders).ToList();
    }

    private static int GetSimilarityCQPoints(List<VMBikeRider> sharedRiders)
    {
        return sharedRiders.Sum(r => r.CQPoints);
    }

    private static double CalculateSimilarityScore(int noOfSharedRiders, int noOfUniqueRidersBothTeamsInTotal)
    {
        return (double)noOfSharedRiders / noOfUniqueRidersBothTeamsInTotal;
```

Smaller constructor



```csharp
internal class CompetitionTeamComparer
{
    private static VMCompetitionTeam _competitionTeam1;
    private static VMCompetitionTeam _competitionTeam2;
    private List<VMBikeRider> _sharedRiders;

    public CompetitionTeamComparer(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2)
    {
        _competitionTeam1 = competitionTeam1;
        _competitionTeam2 = competitionTeam2;
    }

    public List<VMBikeRider> GetSharedRidersBetweenTheTwoTeams()
    {
        _sharedRiders = FindSharedRidersBetweenTheTwoTeams();
        return _sharedRiders;
    }

    private double CalculateSimilarityScore()
    {
        var noOfSharedRiders = _sharedRiders.Count;
        var noOfUniqueRidersBothTeamsInTotal = GetNoOfUniqueRidersBothTeamsInTotal();
        return (double)noOfSharedRiders / noOfUniqueRidersBothTeamsInTotal;
    }

    public int CalculateSimilarityCQPoints()
    {
        return _sharedRiders.Sum(r => r.CQPoints);
    }

    private List<VMBikeRider> FindSharedRidersBetweenTheTwoTeams()
    {
        return _competitionTeam1.BikeRiders.Intersect(_competitionTeam2.BikeRiders).ToList();
    }

    private int GetNoOfUniqueRidersBothTeamsInTotal()
    {
        return _competitionTeam1.BikeRiders.Union(_competitionTeam2.BikeRiders).Count();
```

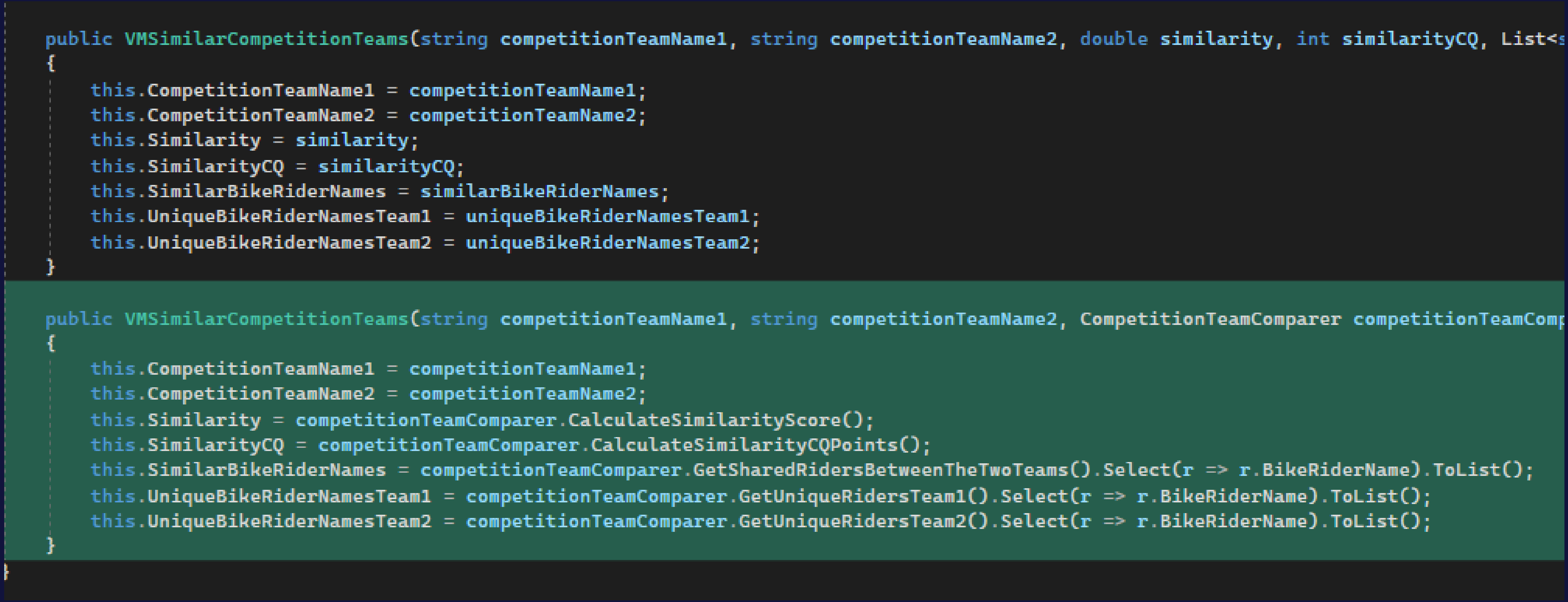
Lists for shared and unique BikeRiders



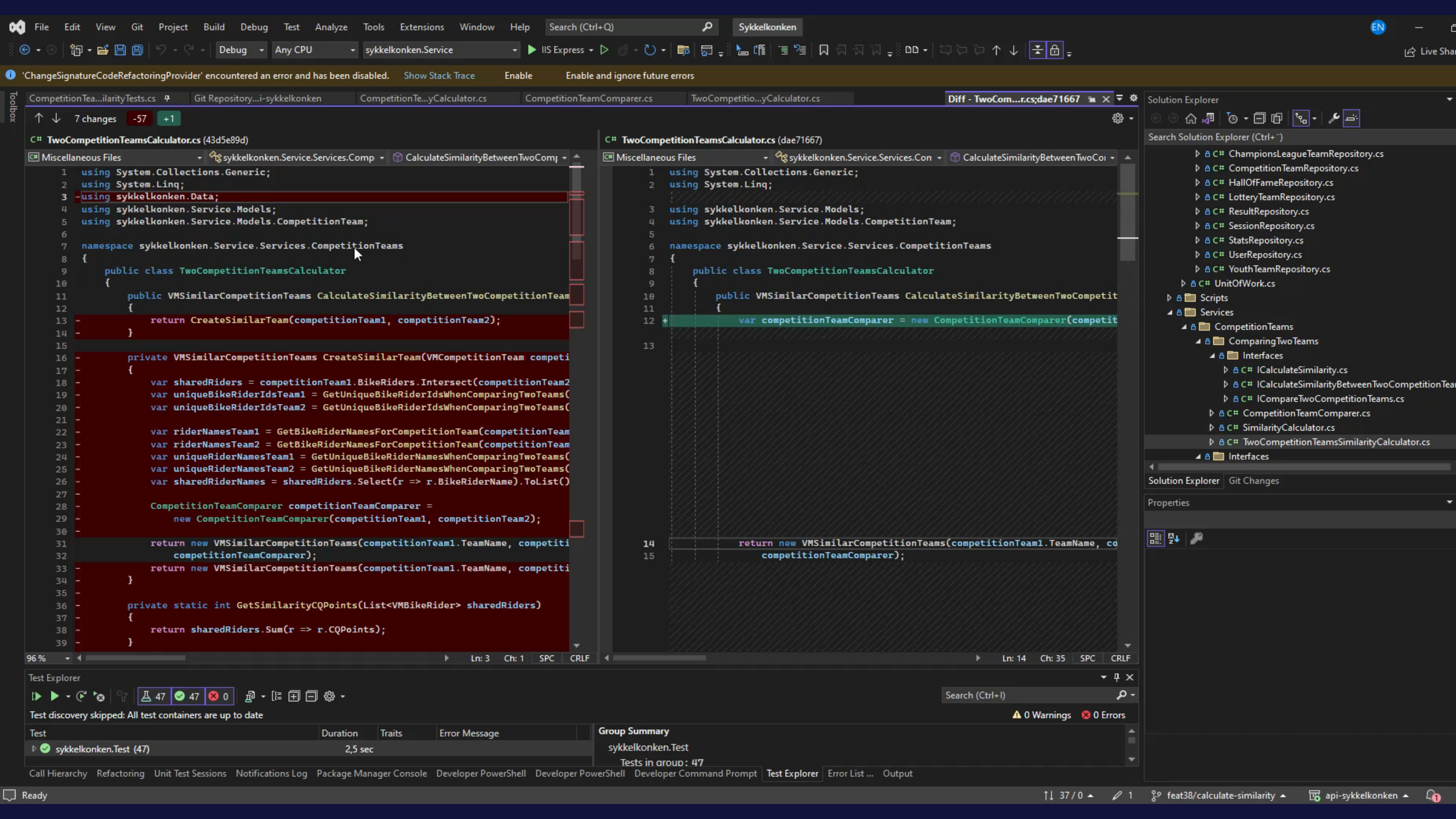
```
public class CompetitionTeamComparer
{
    private readonly VMCompetitionTeam _competitionTeam1;
    private readonly VMCompetitionTeam _competitionTeam2;
    private readonly List<VMBikeRider> _sharedRiders;
    private readonly List<VMBikeRider> _uniqueRidersTeam1;
    private readonly List<VMBikeRider> _uniqueRidersTeam2;

    public CompetitionTeamComparer(VMCompetitionTeam competitionTeam1,
    {
        _competitionTeam1 = competitionTeam1;
        _competitionTeam2 = competitionTeam2;

        _sharedRiders = FindSharedRidersBetweenTheTwoTeams();
        _uniqueRidersTeam1 = FindUniqueBikeRidersTeam1();
        _uniqueRidersTeam2 = FindUniqueBikeRidersTeam2();
    }

    public List<VMBikeRider> GetSharedRidersBetweenTheTwoTeams()
    {
        return _sharedRiders;
    }

    public List<VMBikeRider> GetUniqueRidersTeam1()
    {
        return _uniqueRidersTeam1;
    }
```

VMSimilarCompetitionTeams

Parallel change



```
    public VMSimilarCompetitionTeams(string competitionTeamName1, string competitionTeamName2, double similarity, int similarityCQ, List<s
    {
        this.CompetitionTeamName1 = competitionTeamName1;
        this.CompetitionTeamName2 = competitionTeamName2;
        this.Similarity = similarity;
        this.SimilarityCQ = similarityCQ;
        this.SimilarBikeRiderNames = similarBikeRiderNames;
        this.UniqueBikeRiderNamesTeam1 = uniqueBikeRiderNamesTeam1;
        this.UniqueBikeRiderNamesTeam2 = uniqueBikeRiderNamesTeam2;
    }

    public VMSimilarCompetitionTeams(string competitionTeamName1, string competitionTeamName2, CompetitionTeamComparer competitionTeamComp
    {
        this.CompetitionTeamName1 = competitionTeamName1;
        this.CompetitionTeamName2 = competitionTeamName2;
        this.Similarity = competitionTeamComparer.CalculateSimilarityScore();
        this.SimilarityCQ = competitionTeamComparer.CalculateSimilarityCQPoints();
        this.SimilarBikeRiderNames = competitionTeamComparer.GetSharedRidersBetweenTheTwoTeams().Select(r => r.BikeRiderName).ToList();
        this.UniqueBikeRiderNamesTeam1 = competitionTeamComparer.GetUniqueRidersTeam1().Select(r => r.BikeRiderName).ToList();
        this.UniqueBikeRiderNamesTeam2 = competitionTeamComparer.GetUniqueRidersTeam2().Select(r => r.BikeRiderName).ToList();
    }
}
```


File
Edit
View
Git
Project
Build
Debug
Test
Analyze
Tools
Extensions
Window
Help
Search (Ctrl+Q)
Sykkelkonken
Debug
Any CPU
sykkelkonken.Service
IIS Express
Live Share

'ChangeSignatureCodeRefactoringProvider' encountered an error and has been disabled. Show Stack Trace Enable Enable and ignore future errors

CompetitionTea...ilarityTests.cs
Git Repository...i-sykkelkonken
CompetitionTe...yCalculator.cs
CompetitionTeamComparer.cs
TwoCompetitio...yCalculator.cs
Diff - TwoCom...r.cs;dae71667
7 changes -57 +1

TwoCompetitionTeamsCalculator.cs (43d5e89d)

Miscellaneous Files | sykkelkonken.Service.Services.Comp | CalculateSimilarityBetweenTwoComp

```
using System.Collections.Generic;
using System.Linq;
using sykkelkonken.Data;
using sykkelkonken.Service.Models;
using sykkelkonken.Service.Models.CompetitionTeam;

namespace sykkelkonken.Service.Services.CompetitionTeams
{
    public class TwoCompetitionTeamsCalculator
    {
        public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetitionTeam
        {
            return CreateSimilarTeam(competitionTeam1, competitionTeam2);
        }

        private VMSimilarCompetitionTeams CreateSimilarTeam(VMCompetitionTeam competi
        {
            var sharedRiders = competitionTeam1.BikeRiders.Intersect(competitionTeam2
            var uniqueBikeRiderIdsTeam1 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(
            var uniqueBikeRiderIdsTeam2 = GetUniqueBikeRiderIdsWhenComparingTwoTeams(

            var riderNamesTeam1 = GetBikeRiderNamesForCompetitionTeam(competitionTeam
            var riderNamesTeam2 = GetBikeRiderNamesForCompetitionTeam(competitionTeam
            var uniqueRiderNamesTeam1 = GetUniqueBikeRiderNamesWhenComparingTwoTeams(
            var uniqueRiderNamesTeam2 = GetUniqueBikeRiderNamesWhenComparingTwoTeams(
            var sharedRiderNames = sharedRiders.Select(r => r.BikeRiderName).ToList()

            CompetitionTeamComparer competitionTeamComparer =
                new CompetitionTeamComparer(competitionTeam1, competitionTeam2);

            return new VMSimilarCompetitionTeams(competitionTeam1.TeamName, competiti
                competitionTeamComparer);
            return new VMSimilarCompetitionTeams(competitionTeam1.TeamName, competiti
        }

        private static int GetSimilarityCQPoints(List<VMBikeRider> sharedRiders)
        {
            return sharedRiders.Sum(r => r.CQPoints);
        }
```

96 % Ln: 3 Ch: 1 SPC CRLF

TwoCompetitionTeamsCalculator.cs (dae71667)

Miscellaneous Files | sykkelkonken.Service.Services.Con | CalculateSimilarityBetweenTwoCo

```
using System.Collections.Generic;
using System.Linq;

using sykkelkonken.Service.Models;
using sykkelkonken.Service.Models.CompetitionTeam;

namespace sykkelkonken.Service.Services.CompetitionTeams
{
    public class TwoCompetitionTeamsCalculator
    {
        public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetit
        {
            var competitionTeamComparer = new CompetitionTeamComparer(competit

            return new VMSimilarCompetitionTeams(competitionTeam1.TeamName, co
                competitionTeamComparer);
```

Ln: 14 Ch: 35 SPC CRLF

Solution Explorer

Search Solution Explorer (Ctrl+")

- ChampionsLeagueTeamRepository.cs
- CompetitionTeamRepository.cs
- HallOfFameRepository.cs
- LotteryTeamRepository.cs
- ResultRepository.cs
- SessionRepository.cs
- StatsRepository.cs
- UserRepository.cs
- YouthTeamRepository.cs
- UnitOfWork.cs
- Scripts
- Services
  - CompetitionTeams
    - ComparingTwoTeams
      - Interfaces
        - ICalculateSimilarity.cs
        - ICalculateSimilarityBetweenTwoCompetitionTea
        - ICompareTwoCompetitionTeams.cs
      - CompetitionTeamComparer.cs
      - SimilarityCalculator.cs
      - TwoCompetitionTeamsSimilarityCalculator.cs
    - Interfaces

Test Explorer

47 47 0

Test discovery skipped: All test containers are up to date

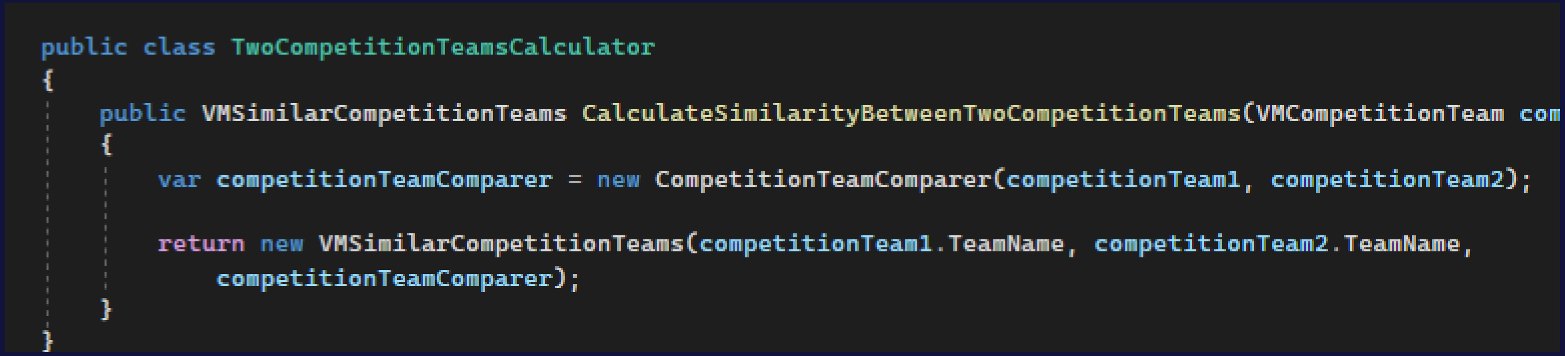
Search (Ctrl+I)

0 Warnings 0 Errors

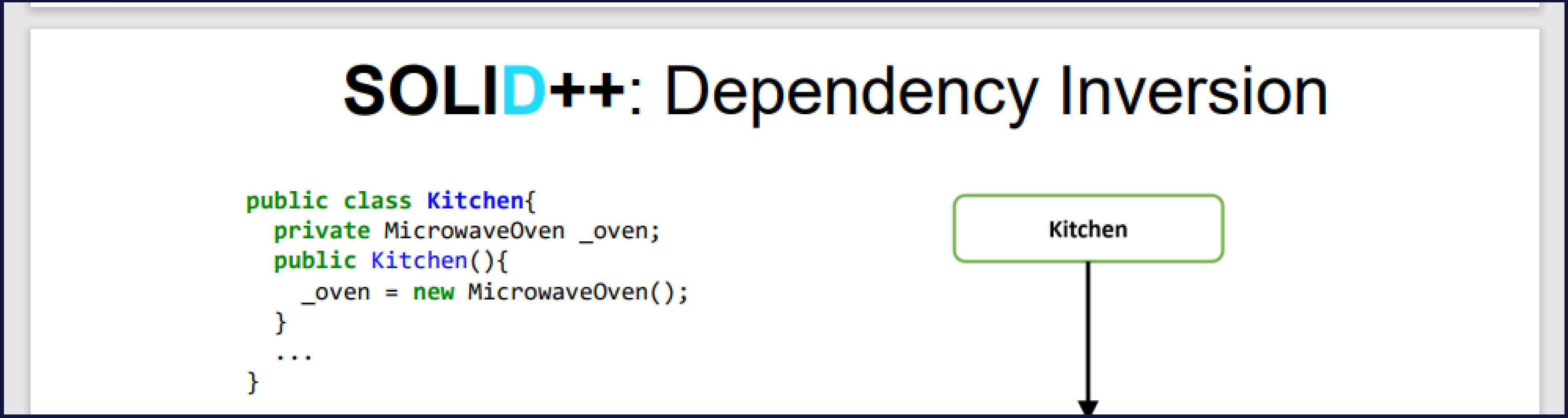
Test	Duration	Traits	Error Message
sykkelkonken.Test (47)	2,5 sec		

Group Summary
sykkelkonken.Test
Tests in group: 47

Dependency inversion violation?



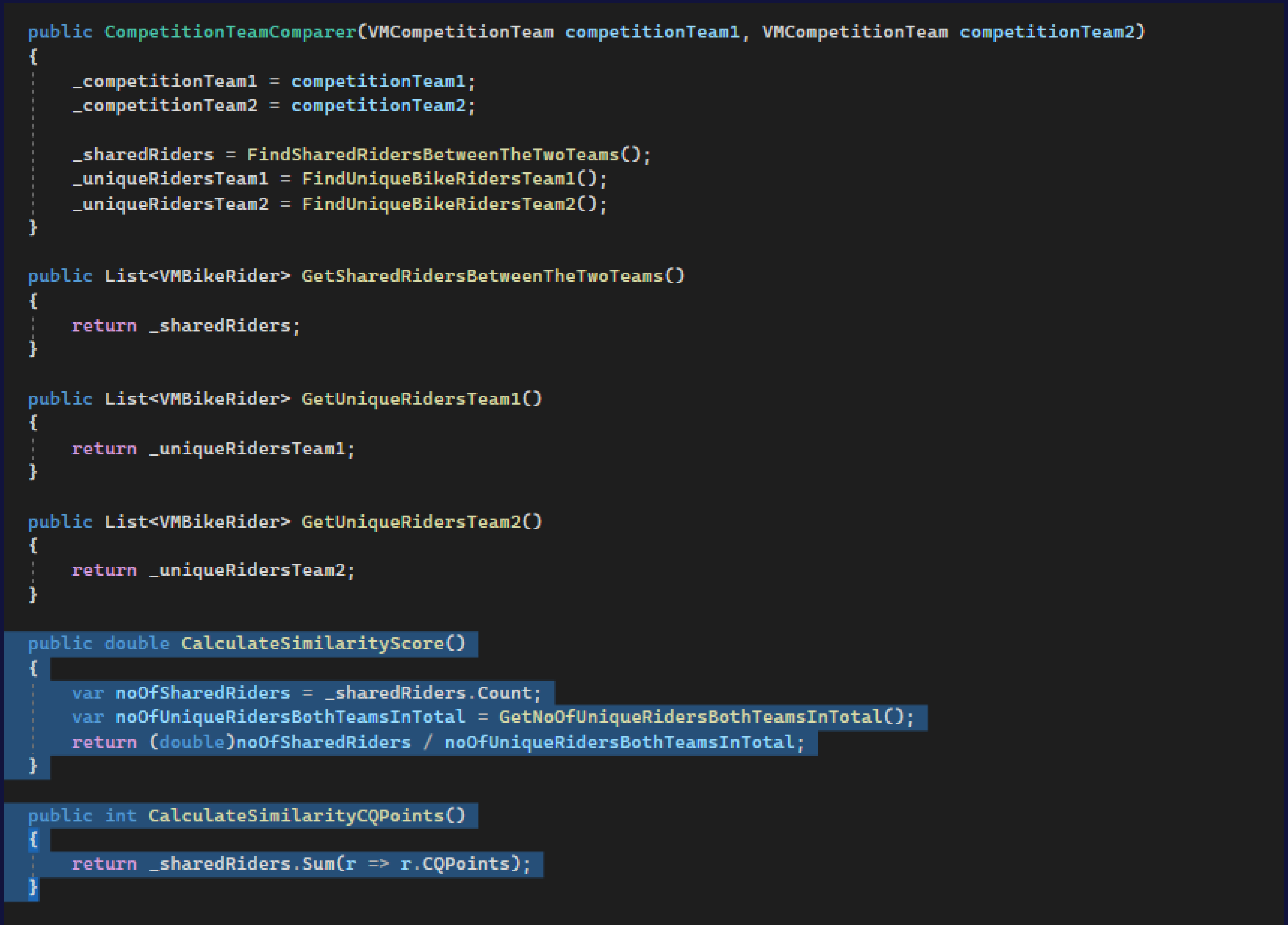
```
public class TwoCompetitionTeamsCalculator
{
    public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam com
    {
        var competitionTeamComparer = new CompetitionTeamComparer(competitionTeam1, competitionTeam2);

        return new VMSimilarCompetitionTeams(competitionTeam1.TeamName, competitionTeam2.TeamName,
            competitionTeamComparer);
    }
}
```



SOLID++: Dependency Inversion

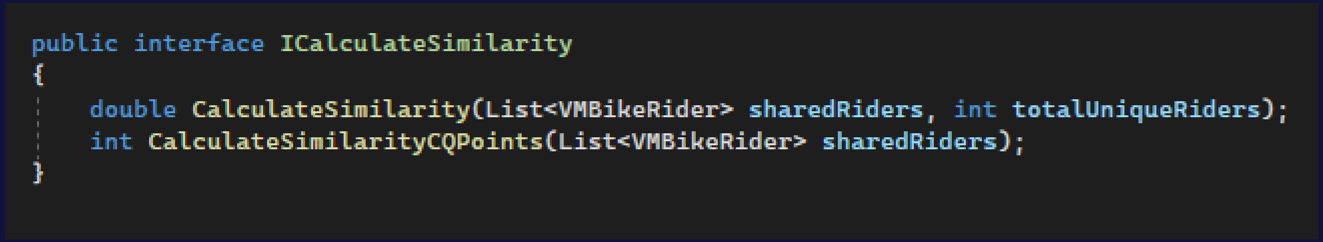
```
public class Kitchen{
  private MicrowaveOven _oven;
  public Kitchen(){
    _oven = new MicrowaveOven();
  }
  ...
}
```

Too much responsibility in this class?

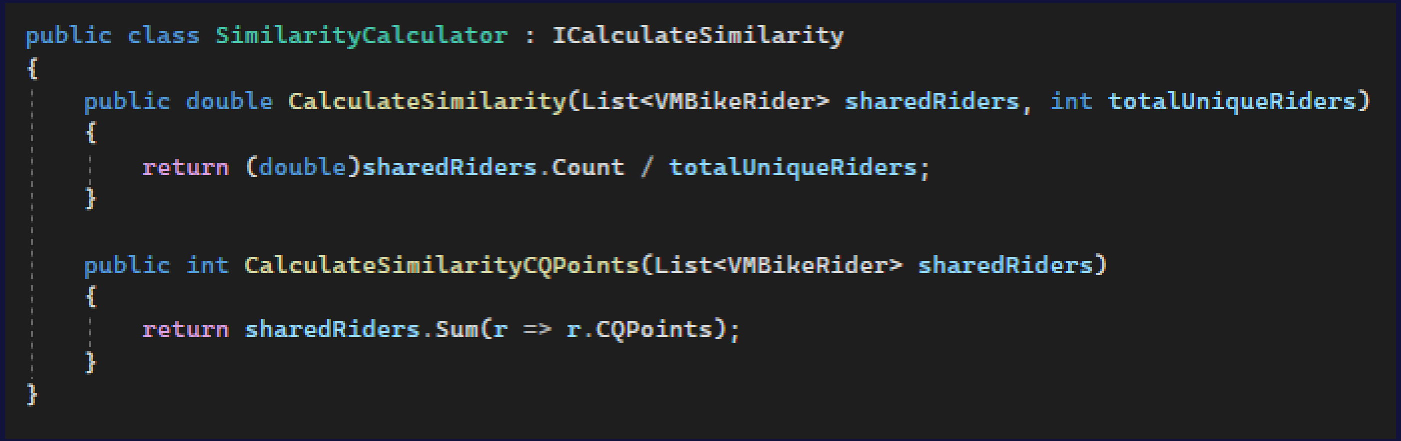


```
public CompetitionTeamComparer(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2)
{
    _competitionTeam1 = competitionTeam1;
    _competitionTeam2 = competitionTeam2;

    _sharedRiders = FindSharedRidersBetweenTheTwoTeams();
    _uniqueRidersTeam1 = FindUniqueBikeRidersTeam1();
    _uniqueRidersTeam2 = FindUniqueBikeRidersTeam2();
}

public List<VMBikeRider> GetSharedRidersBetweenTheTwoTeams()
{
    return _sharedRiders;
}

public List<VMBikeRider> GetUniqueRidersTeam1()
{
    return _uniqueRidersTeam1;
}

public List<VMBikeRider> GetUniqueRidersTeam2()
{
    return _uniqueRidersTeam2;
}

public double CalculateSimilarityScore()
{
    var noOfSharedRiders = _sharedRiders.Count;
    var noOfUniqueRidersBothTeamsInTotal = GetNoOfUniqueRidersBothTeamsInTotal();
    return (double)noOfSharedRiders / noOfUniqueRidersBothTeamsInTotal;
}

public int CalculateSimilarityCQPoints()
{
    return _sharedRiders.Sum(r => r.CQPoints);
}
```

Moving calculation responsibility

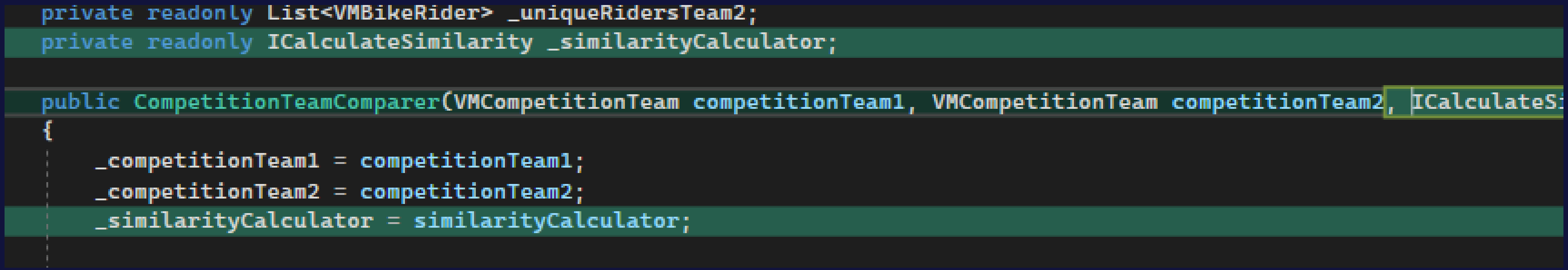


```
public interface ICalculateSimilarity
{
    double CalculateSimilarity(List<VMBikeRider> sharedRiders, int totalUniqueRiders);
    int CalculateSimilarityCQPoints(List<VMBikeRider> sharedRiders);
}
```



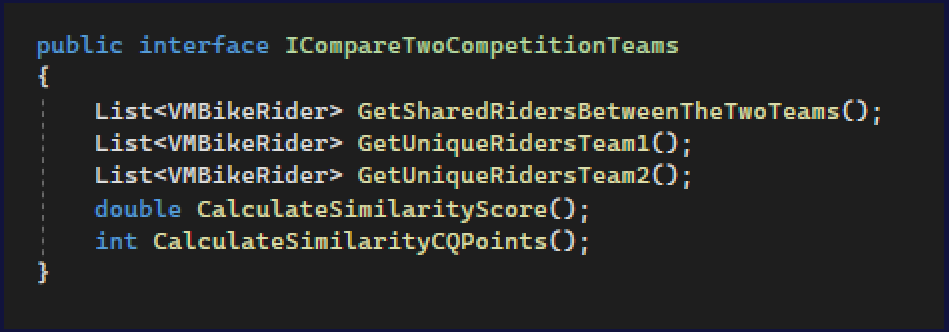
```
public class SimilarityCalculator : ICalculateSimilarity
{
    public double CalculateSimilarity(List<VMBikeRider> sharedRiders, int totalUniqueRiders)
    {
        return (double)sharedRiders.Count / totalUniqueRiders;
    }

    public int CalculateSimilarityCQPoints(List<VMBikeRider> sharedRiders)
    {
        return sharedRiders.Sum(r => r.CQPoints);
    }
}
```

Injected here

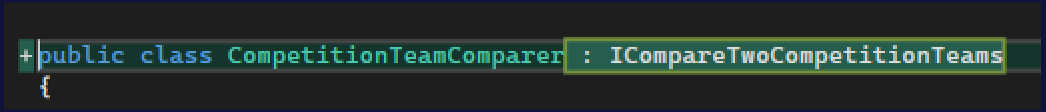


```
    private readonly List<VMBikeRider> _uniqueRidersTeam2;
    private readonly ICalculateSimilarity _similarityCalculator;

    public CompetitionTeamComparer(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2, ICalculateS
    {
        _competitionTeam1 = competitionTeam1;
        _competitionTeam2 = competitionTeam2;
        _similarityCalculator = similarityCalculator;
```

ICompareTwoCompetitionTeams

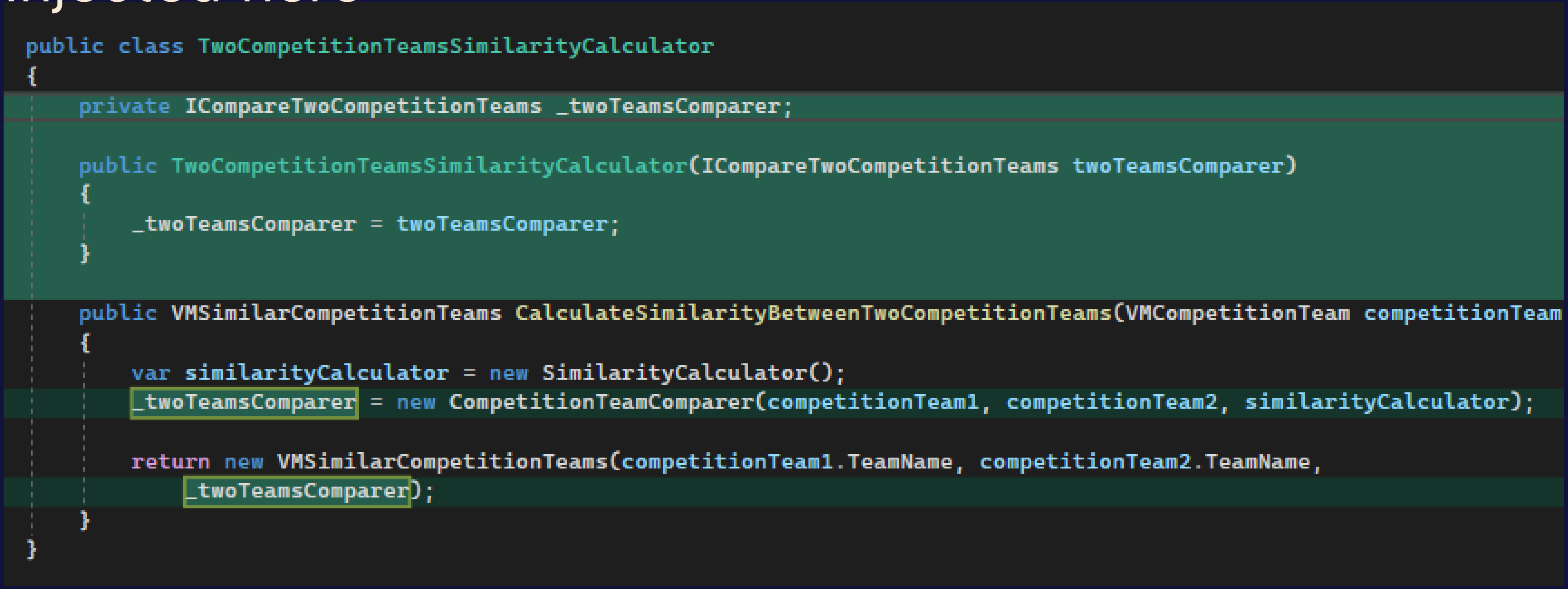


```
public interface ICompareTwoCompetitionTeams
{
    List<VMBikeRider> GetSharedRidersBetweenTheTwoTeams();
    List<VMBikeRider> GetUniqueRidersTeam1();
    List<VMBikeRider> GetUniqueRidersTeam2();
    double CalculateSimilarityScore();
    int CalculateSimilarityCQPoints();
}
```

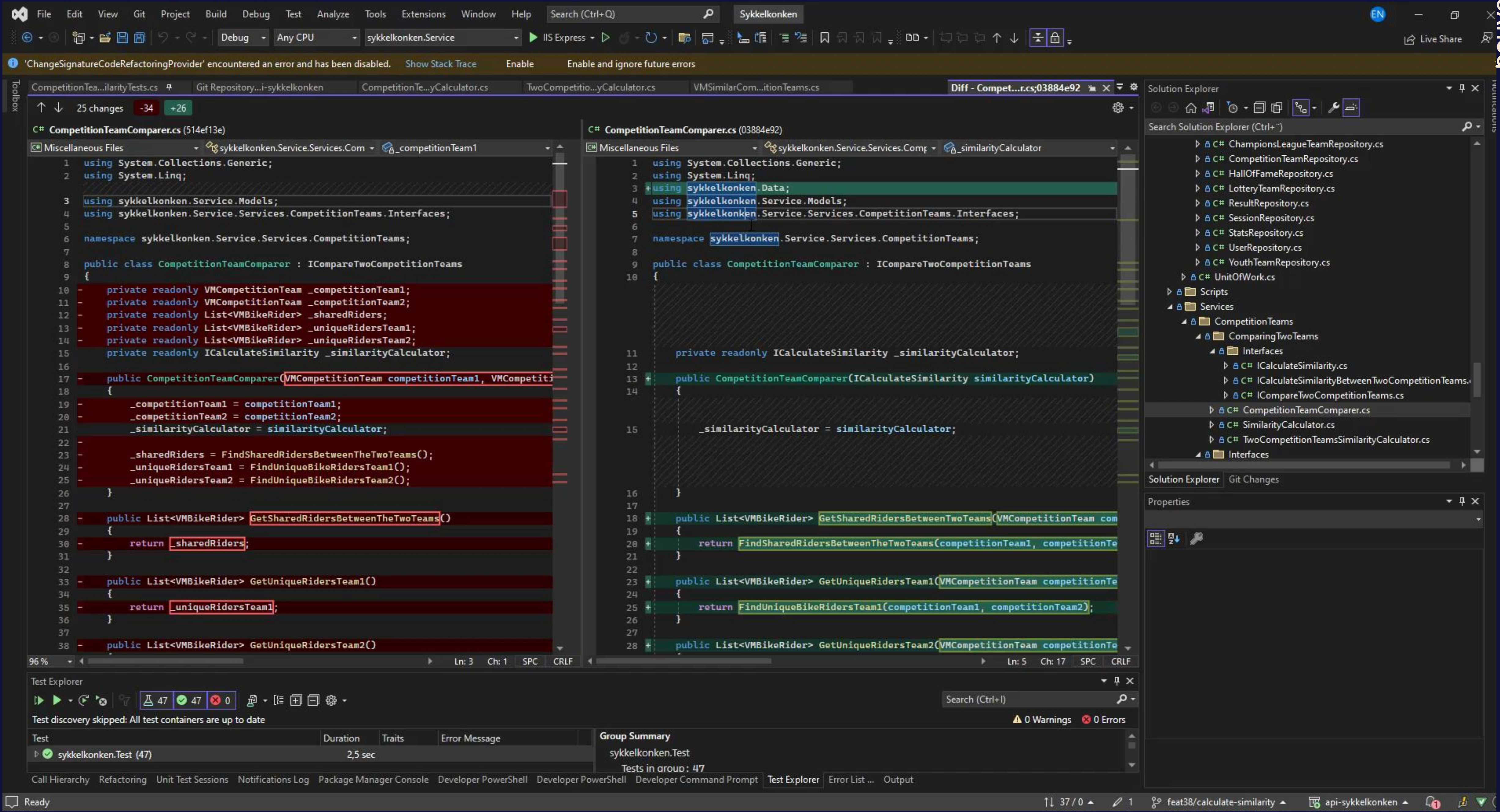


```
+public class CompetitionTeamComparer : ICompareTwoCompetitionTeams
 {
```

Injected here



```
public class TwoCompetitionTeamsSimilarityCalculator
{
    private ICompareTwoCompetitionTeams _twoTeamsComparer;

    public TwoCompetitionTeamsSimilarityCalculator(ICompareTwoCompetitionTeams twoTeamsComparer)
    {
        _twoTeamsComparer = twoTeamsComparer;
    }

    public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam competitionTeam
    {
        var similarityCalculator = new SimilarityCalculator();
        _twoTeamsComparer = new CompetitionTeamComparer(competitionTeam1, competitionTeam2, similarityCalculator);

        return new VMSimilarCompetitionTeams(competitionTeam1.TeamName, competitionTeam2.TeamName,
            _twoTeamsComparer);
    }
}
```


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
Sykkelkonken
Debug
Any CPU
sykkelkonken.Service
IIS Express
Live Share

'ChangeSignatureCodeRefactoringProvider' encountered an error and has been disabled. Show Stack Trace Enable Enable and ignore future errors

CompetitionTea...ilarityTests.cs
Git Repository...i-sykkelkonken
CompetitionTe...yCalculator.cs
TwoCompetitio...yCalculator.cs
VMSimilarCom...itionTeams.cs
Diff - Compet...r.cs;03884e92
25 changes -34 +26

CompetitionTeamComparer.cs (514ef13e)

Miscellaneous Files · sykkelkonken.Service.Services.Com · _competitionTeam1

```
using System.Collections.Generic;
using System.Linq;

using sykkelkonken.Service.Models;
using sykkelkonken.Service.Services.CompetitionTeams.Interfaces;

namespace sykkelkonken.Service.Services.CompetitionTeams;

public class CompetitionTeamComparer : ICompareTwoCompetitionTeams
{
-    private readonly VMCompetitionTeam _competitionTeam1;
-    private readonly VMCompetitionTeam _competitionTeam2;
-    private readonly List<VMBikeRider> _sharedRiders;
-    private readonly List<VMBikeRider> _uniqueRidersTeam1;
-    private readonly List<VMBikeRider> _uniqueRidersTeam2;
     private readonly ICalculateSimilarity _similarityCalculator;

-    public CompetitionTeamComparer(VMCompetitionTeam competitionTeam1, VMCompetiti
     {
-        _competitionTeam1 = competitionTeam1;
-        _competitionTeam2 = competitionTeam2;
         _similarityCalculator = similarityCalculator;
-
-        _sharedRiders = FindSharedRidersBetweenTheTwoTeams();
-        _uniqueRidersTeam1 = FindUniqueBikeRidersTeam1();
-        _uniqueRidersTeam2 = FindUniqueBikeRidersTeam2();
     }

-    public List<VMBikeRider> GetSharedRidersBetweenTheTwoTeams()
     {
-        return _sharedRiders;
     }

-    public List<VMBikeRider> GetUniqueRidersTeam1()
     {
-        return _uniqueRidersTeam1;
     }

-    public List<VMBikeRider> GetUniqueRidersTeam2()
```

96 % · Ln: 3 Ch: 1 SPC CRLF

CompetitionTeamComparer.cs (03884e92)

Miscellaneous Files · sykkelkonken.Service.Services.Comp · _similarityCalculator

```
 using System.Collections.Generic;
 using System.Linq;
+using sykkelkonken.Data;
 using sykkelkonken.Service.Models;
 using sykkelkonken.Service.Services.CompetitionTeams.Interfaces;

 namespace sykkelkonken.Service.Services.CompetitionTeams;

 public class CompetitionTeamComparer : ICompareTwoCompetitionTeams
 {

     private readonly ICalculateSimilarity _similarityCalculator;

+    public CompetitionTeamComparer(ICalculateSimilarity similarityCalculator)
     {

         _similarityCalculator = similarityCalculator;

     }

+    public List<VMBikeRider> GetSharedRidersBetweenTwoTeams(VMCompetitionTeam com
     {
+        return FindSharedRidersBetweenTheTwoTeams(competitionTeam1, competitionTe
     }

+    public List<VMBikeRider> GetUniqueRidersTeam1(VMCompetitionTeam competitionTe
     {
+        return FindUniqueBikeRidersTeam1(competitionTeam1, competitionTeam2);
     }

+    public List<VMBikeRider> GetUniqueRidersTeam2(VMCompetitionTeam competitionTe
```

Ln: 5 Ch: 17 SPC CRLF

Solution Explorer

Search Solution Explorer (Ctrl+¨)

```
ChampionsLeagueTeamRepository.cs
CompetitionTeamRepository.cs
HallOfFameRepository.cs
LotteryTeamRepository.cs
ResultRepository.cs
SessionRepository.cs
StatsRepository.cs
UserRepository.cs
YouthTeamRepository.cs
UnitOfWork.cs
Scripts
Services
  CompetitionTeams
    ComparingTwoTeams
      Interfaces
        ICalculateSimilarity.cs
        ICalculateSimilarityBetweenTwoCompetitionTeams.
        ICompareTwoCompetitionTeams.cs
      CompetitionTeamComparer.cs
      SimilarityCalculator.cs
      TwoCompetitionTeamsSimilarityCalculator.cs
    Interfaces
```

Solution Explorer · Git Changes

Properties

Test Explorer

47 · 47 · 0

Test discovery skipped: All test containers are up to date

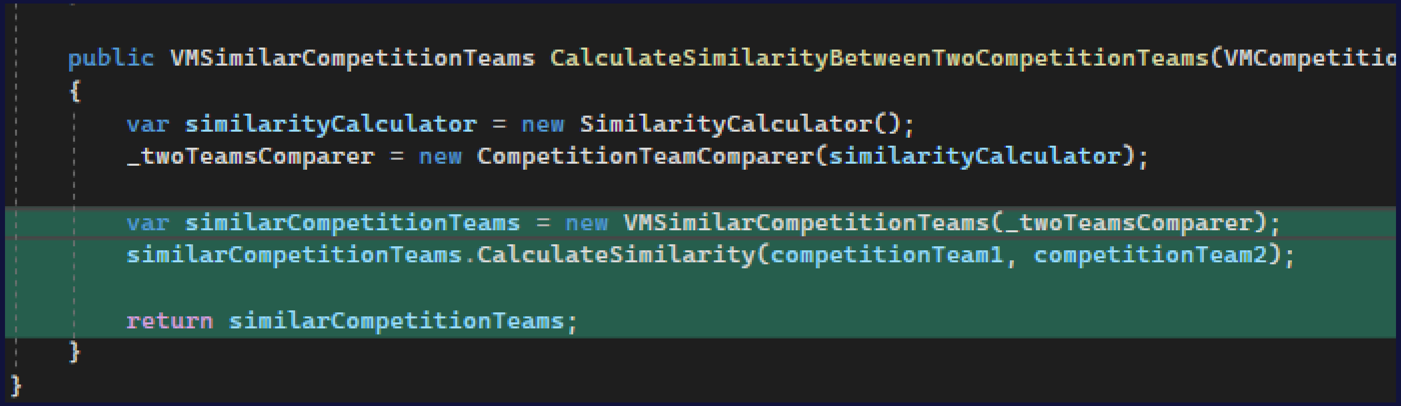
Search (Ctrl+I) · 0 Warnings · 0 Errors

Test	Duration	Traits	Error Message
sykkelkonken.Test (47)	2,5 sec		

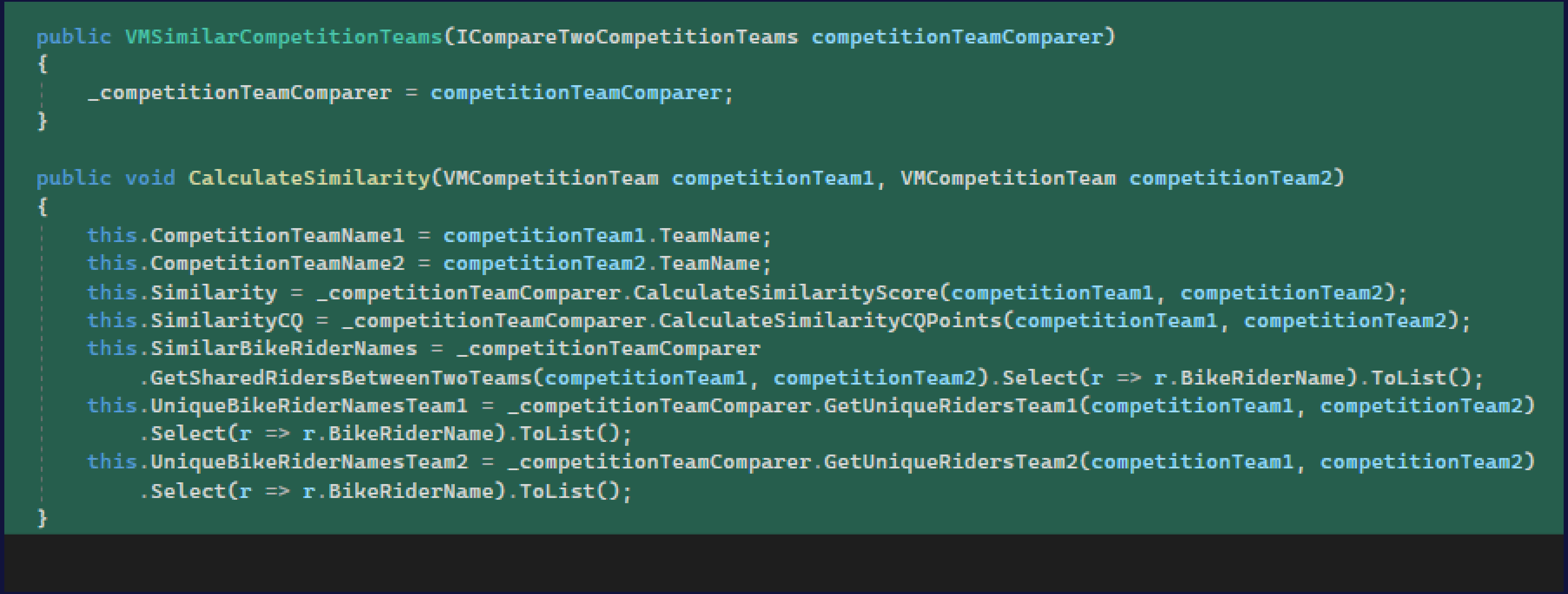
Group Summary
sykkelkonken.Test
Tests in group: 47

Call Hierarchy Refactoring Unit Test Sessions Notifications Log Package Manager Console Developer PowerShell Developer PowerShell Developer Command Prompt Test Explorer Error List ... Output
Ready
37 / 0
1
feat38/calculate-similarity
api-sykkelkonken

Side-effects in DTO

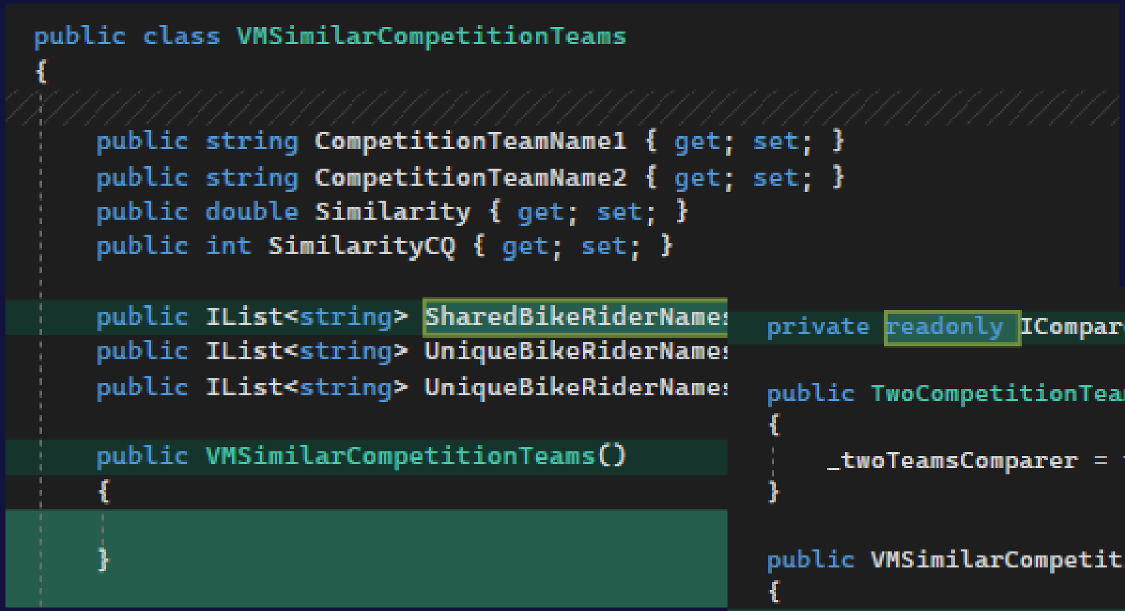


```
    public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitio
    {
        var similarityCalculator = new SimilarityCalculator();
        _twoTeamsComparer = new CompetitionTeamComparer(similarityCalculator);

        var similarCompetitionTeams = new VMSimilarCompetitionTeams(_twoTeamsComparer);
        similarCompetitionTeams.CalculateSimilarity(competitionTeam1, competitionTeam2);

        return similarCompetitionTeams;
    }
}
```

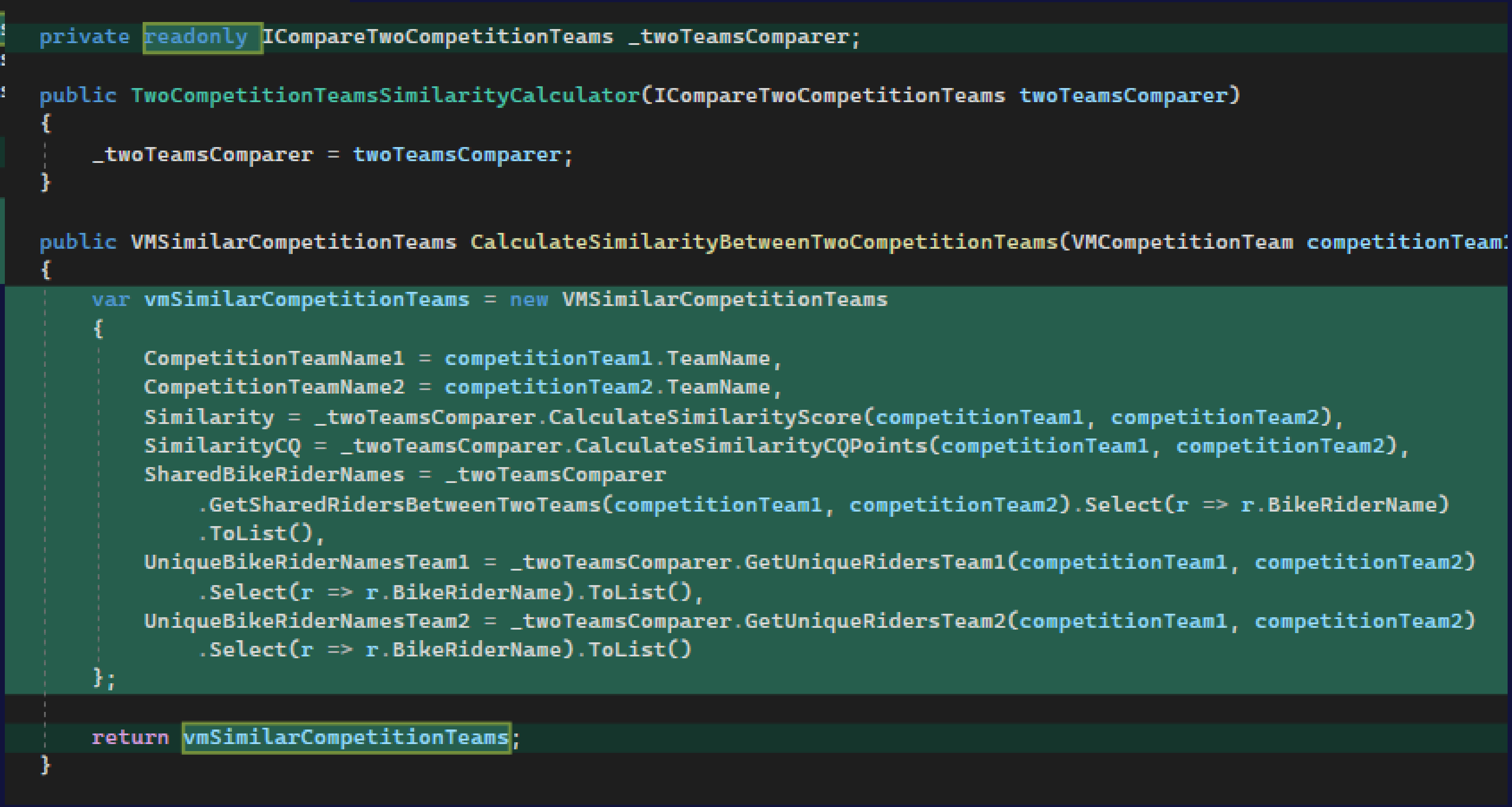


```
public VMSimilarCompetitionTeams(ICompareTwoCompetitionTeams competitionTeamComparer)
{
    _competitionTeamComparer = competitionTeamComparer;
}

public void CalculateSimilarity(VMCompetitionTeam competitionTeam1, VMCompetitionTeam competitionTeam2)
{
    this.CompetitionTeamName1 = competitionTeam1.TeamName;
    this.CompetitionTeamName2 = competitionTeam2.TeamName;
    this.Similarity = _competitionTeamComparer.CalculateSimilarityScore(competitionTeam1, competitionTeam2);
    this.SimilarityCQ = _competitionTeamComparer.CalculateSimilarityCQPoints(competitionTeam1, competitionTeam2);
    this.SimilarBikeRiderNames = _competitionTeamComparer
        .GetSharedRidersBetweenTwoTeams(competitionTeam1, competitionTeam2).Select(r => r.BikeRiderName).ToList();
    this.UniqueBikeRiderNamesTeam1 = _competitionTeamComparer.GetUniqueRidersTeam1(competitionTeam1, competitionTeam2)
        .Select(r => r.BikeRiderName).ToList();
    this.UniqueBikeRiderNamesTeam2 = _competitionTeamComparer.GetUniqueRidersTeam2(competitionTeam1, competitionTeam2)
        .Select(r => r.BikeRiderName).ToList();
}
```

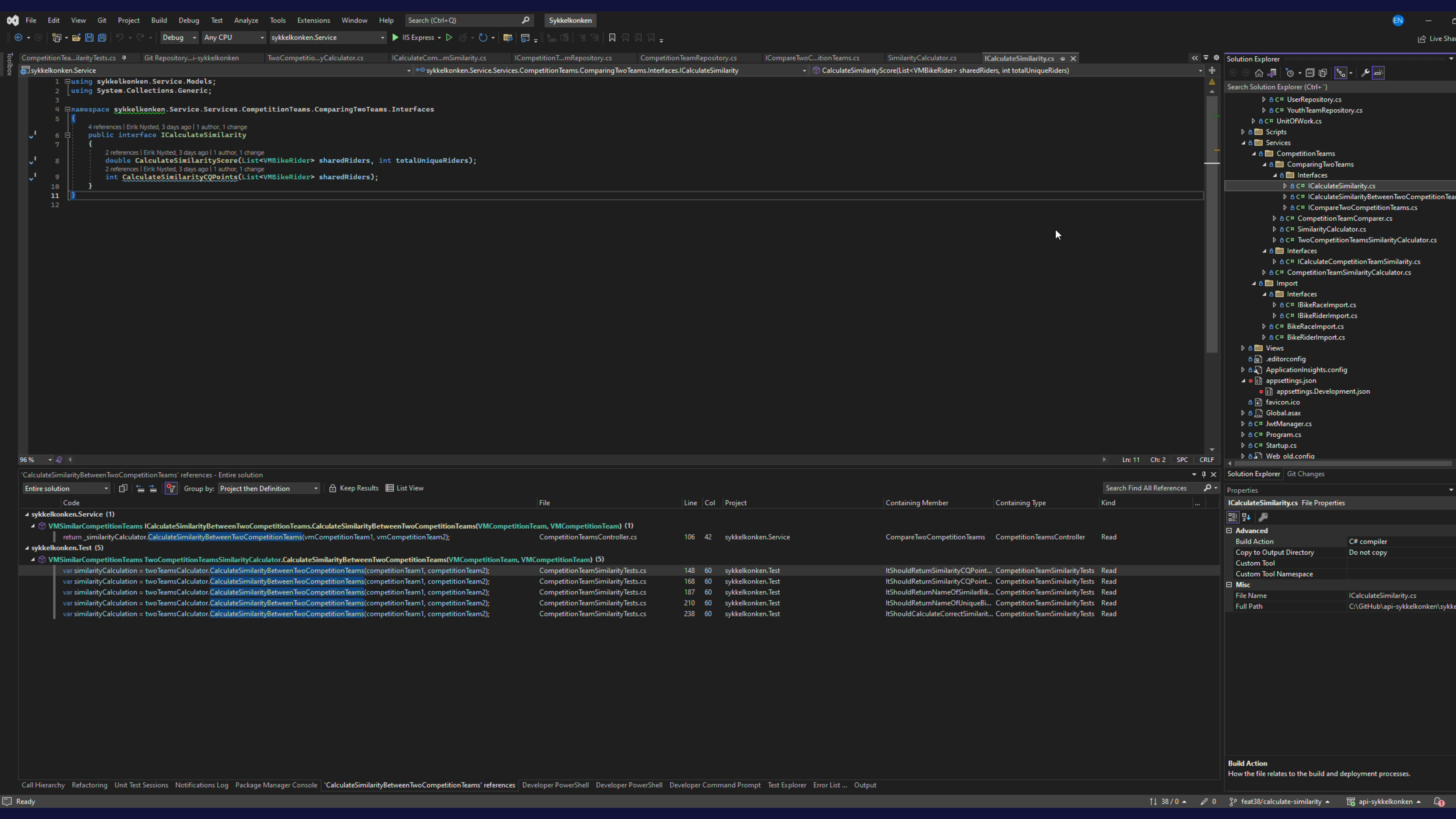
Make VMSimilarCompetitionTeams a DTO again



```
public class VMSimilarCompetitionTeams
{

    public string CompetitionTeamName1 { get; set; }
    public string CompetitionTeamName2 { get; set; }
    public double Similarity { get; set; }
    public int SimilarityCQ { get; set; }

    public IList<string> SharedBikeRiderNames
    public IList<string> UniqueBikeRiderNames
    public IList<string> UniqueBikeRiderNames

    public VMSimilarCompetitionTeams()
    {

    }
```



```
private readonly ICompareTwoCompetitionTeams _twoTeamsComparer;

public TwoCompetitionTeamsSimilarityCalculator(ICompareTwoCompetitionTeams twoTeamsComparer)
{
    _twoTeamsComparer = twoTeamsComparer;
}

public VMSimilarCompetitionTeams CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam competitionTeam
{
    var vmSimilarCompetitionTeams = new VMSimilarCompetitionTeams
    {
        CompetitionTeamName1 = competitionTeam1.TeamName,
        CompetitionTeamName2 = competitionTeam2.TeamName,
        Similarity = _twoTeamsComparer.CalculateSimilarityScore(competitionTeam1, competitionTeam2),
        SimilarityCQ = _twoTeamsComparer.CalculateSimilarityCQPoints(competitionTeam1, competitionTeam2),
        SharedBikeRiderNames = _twoTeamsComparer
            .GetSharedRidersBetweenTwoTeams(competitionTeam1, competitionTeam2).Select(r => r.BikeRiderName)
            .ToList(),
        UniqueBikeRiderNamesTeam1 = _twoTeamsComparer.GetUniqueRidersTeam1(competitionTeam1, competitionTeam2)
            .Select(r => r.BikeRiderName).ToList(),
        UniqueBikeRiderNamesTeam2 = _twoTeamsComparer.GetUniqueRidersTeam2(competitionTeam1, competitionTeam2)
            .Select(r => r.BikeRiderName).ToList()
    };

    return vmSimilarCompetitionTeams;
}
```



```
using sykkelkonken.Service.Models;
using System.Collections.Generic;

namespace sykkelkonken.Service.Services.CompetitionTeams.ComparingTwoTeams.Interfaces
{
    4 references | Eirik Nysted, 3 days ago | 1 author, 1 change
    public interface ICalculateSimilarity
    {
        2 references | Eirik Nysted, 3 days ago | 1 author, 1 change
        double CalculateSimilarityScore(List<VMBikeRider> sharedRiders, int totalUniqueRiders);
        2 references | Eirik Nysted, 3 days ago | 1 author, 1 change
        int CalculateSimilarityCQPoints(List<VMBikeRider> sharedRiders);
    }
}
```

'CalculateSimilarityBetweenTwoCompetitionTeams' references - Entire solution

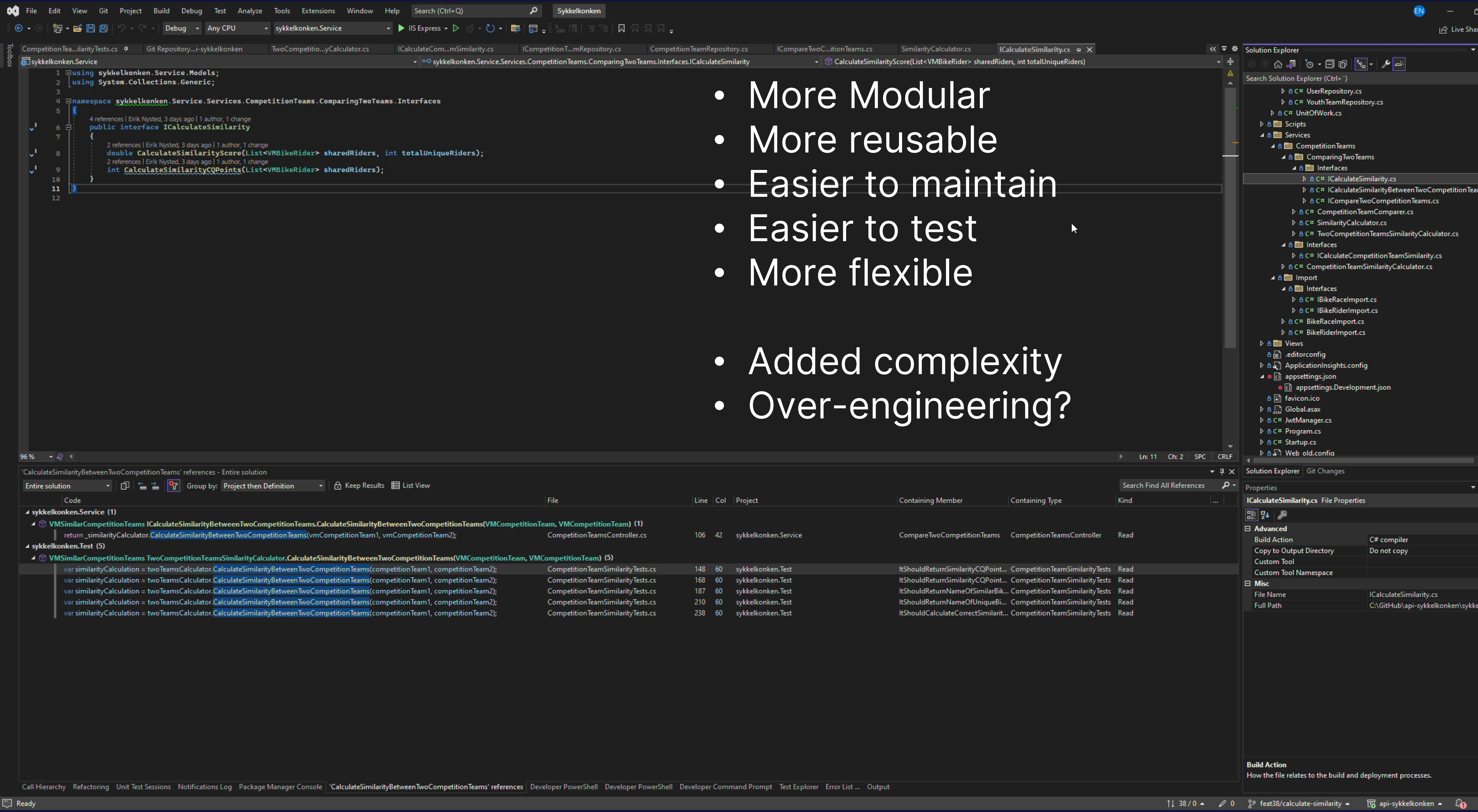
```
Code                                                                                                                              File                              Line  Col  Project              Containing Member                Containing Type                 Kind
◢ sykkelkonken.Service (1)
  ◢ VMSimilarCompetitionTeams ICalculateSimilarityBetweenTwoCompetitionTeams.CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam, VMCompetitionTeam) (1)
      return _similarityCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(vmCompetitionTeam1, vmCompetitionTeam2);   CompetitionTeamsController.cs      106   42   sykkelkonken.Service CompareTwoCompetitionTeams       CompetitionTeamsController      Read
◢ sykkelkonken.Test (5)
  ◢ VMSimilarCompetitionTeams TwoCompetitionTeamsSimilarityCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam, VMCompetitionTeam) (5)
      var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);   CompetitionTeamSimilarityTests.cs   148   60   sykkelkonken.Test   ItShouldReturnSimilarityCQPoint...   CompetitionTeamSimilarityTests   Read
      var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);   CompetitionTeamSimilarityTests.cs   168   60   sykkelkonken.Test   ItShouldReturnSimilarityCQPoint...   CompetitionTeamSimilarityTests   Read
      var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);   CompetitionTeamSimilarityTests.cs   187   60   sykkelkonken.Test   ItShouldReturnNameOfSimilarBik...   CompetitionTeamSimilarityTests   Read
      var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);   CompetitionTeamSimilarityTests.cs   210   60   sykkelkonken.Test   ItShouldReturnNameOfUniqueBi...   CompetitionTeamSimilarityTests   Read
      var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);   CompetitionTeamSimilarityTests.cs   238   60   sykkelkonken.Test   ItShouldCalculateCorrectSimilarit...   CompetitionTeamSimilarityTests   Read
```

Solution Explorer

```
UserRepository.cs
YouthTeamRepository.cs
UnitOfWork.cs
Scripts
Services
  CompetitionTeams
    ComparingTwoTeams
      Interfaces
        ICalculateSimilarity.cs
        ICalculateSimilarityBetweenTwoCompetitionTea
        ICompareTwoCompetitionTeams.cs
      CompetitionTeamComparer.cs
      SimilarityCalculator.cs
      TwoCompetitionTeamsSimilarityCalculator.cs
    Interfaces
      ICalculateCompetitionTeamSimilarity.cs
    CompetitionTeamSimilarityCalculator.cs
  Import
    Interfaces
      IBikeRaceImport.cs
      IBikeRiderImport.cs
    BikeRaceImport.cs
    BikeRiderImport.cs
Views
.editorconfig
ApplicationInsights.config
appsettings.json
  appsettings.Development.json
favicon.ico
Global.asax
JwtManager.cs
Program.cs
Startup.cs
Web_old.config
```

Properties — ICalculateSimilarity.cs File Properties

```
Advanced
  Build Action                 C# compiler
  Copy to Output Directory     Do not copy
  Custom Tool
  Custom Tool Namespace
Misc
  File Name                    ICalculateSimilarity.cs
  Full Path                    C:\GitHub\api-sykkelkonken\sykk
```



- More Modular
- More reusable
- Easier to maintain
- Easier to test
- More flexible

- Added complexity
- Over-engineering?

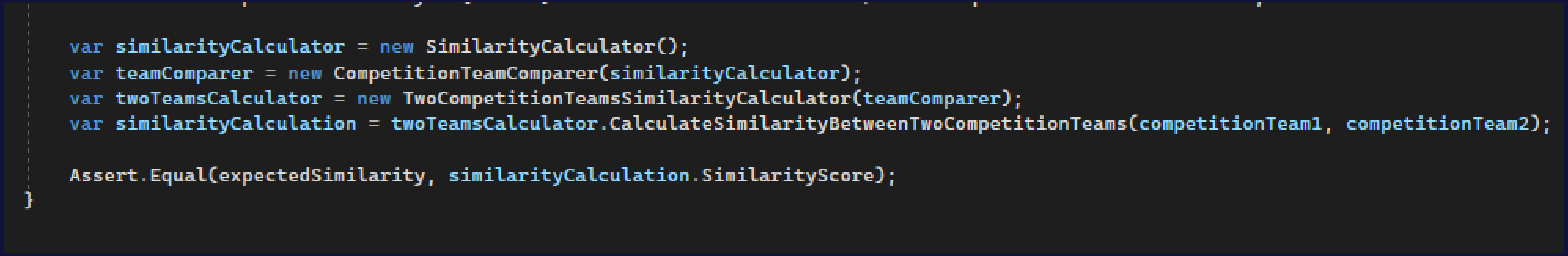
```
using sykkelkonken.Service.Models;
using System.Collections.Generic;

namespace sykkelkonken.Service.Services.CompetitionTeams.ComparingTwoTeams.Interfaces
{
    4 references | Eirik Nysted, 3 days ago | 1 author, 1 change
    public interface ICalculateSimilarity
    {
        2 references | Eirik Nysted, 3 days ago | 1 author, 1 change
        double CalculateSimilarityScore(List<VMBikeRider> sharedRiders, int totalUniqueRiders);
        2 references | Eirik Nysted, 3 days ago | 1 author, 1 change
        int CalculateSimilarityCQPoints(List<VMBikeRider> sharedRiders);
    }
}
```

'CalculateSimilarityBetweenTwoCompetitionTeams' references - Entire solution

Code	File	Line	Col	Project	Containing Member	Containing Type	Kind
sykkelkonken.Service (1)							
VMSimilarCompetitionTeams ICalculateSimilarityBetweenTwoCompetitionTeams.CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam, VMCompetitionTeam) (1)							
return _similarityCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(vmCompetitionTeam1, vmCompetitionTeam2);	CompetitionTeamsController.cs	106	42	sykkelkonken.Service	CompareTwoCompetitionTeams	CompetitionTeamsController	Read
sykkelkonken.Test (5)							
VMSimilarCompetitionTeams TwoCompetitionTeamsSimilarityCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(VMCompetitionTeam, VMCompetitionTeam) (5)							
var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);	CompetitionTeamSimilarityTests.cs	148	60	sykkelkonken.Test	ItShouldReturnSimilarityCQPoint...	CompetitionTeamSimilarityTests	Read
var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);	CompetitionTeamSimilarityTests.cs	168	60	sykkelkonken.Test	ItShouldReturnSimilarityCQPoint...	CompetitionTeamSimilarityTests	Read
var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);	CompetitionTeamSimilarityTests.cs	187	60	sykkelkonken.Test	ItShouldReturnNameOfSimilarBik...	CompetitionTeamSimilarityTests	Read
var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);	CompetitionTeamSimilarityTests.cs	210	60	sykkelkonken.Test	ItShouldReturnNameOfUniqueBi...	CompetitionTeamSimilarityTests	Read
var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);	CompetitionTeamSimilarityTests.cs	238	60	sykkelkonken.Test	ItShouldCalculateCorrectSimilarit...	CompetitionTeamSimilarityTests	Read

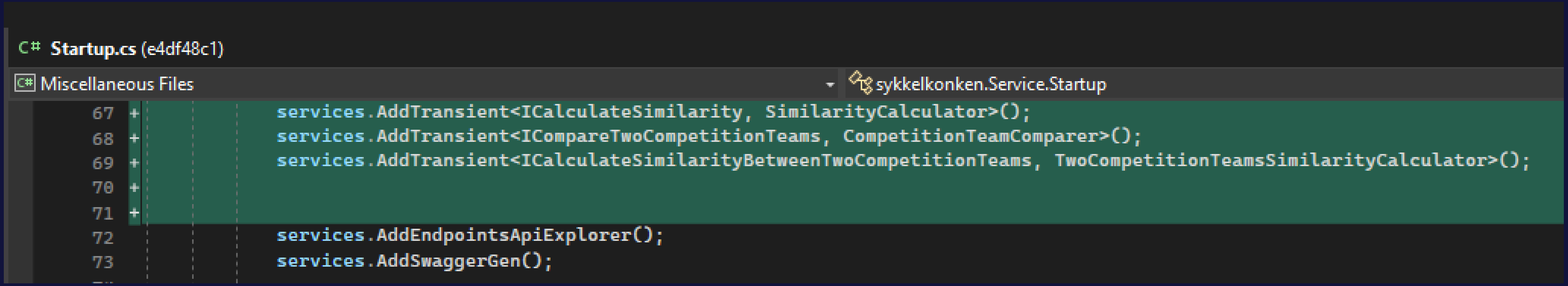
But wait there is more

Creating the dependencies in a test



```
    var similarityCalculator = new SimilarityCalculator();
    var teamComparer = new CompetitionTeamComparer(similarityCalculator);
    var twoTeamsCalculator = new TwoCompetitionTeamsSimilarityCalculator(teamComparer);
    var similarityCalculation = twoTeamsCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(competitionTeam1, competitionTeam2);

    Assert.Equal(expectedSimilarity, similarityCalculation.SimilarityScore);
}
```

Dependencies on startup

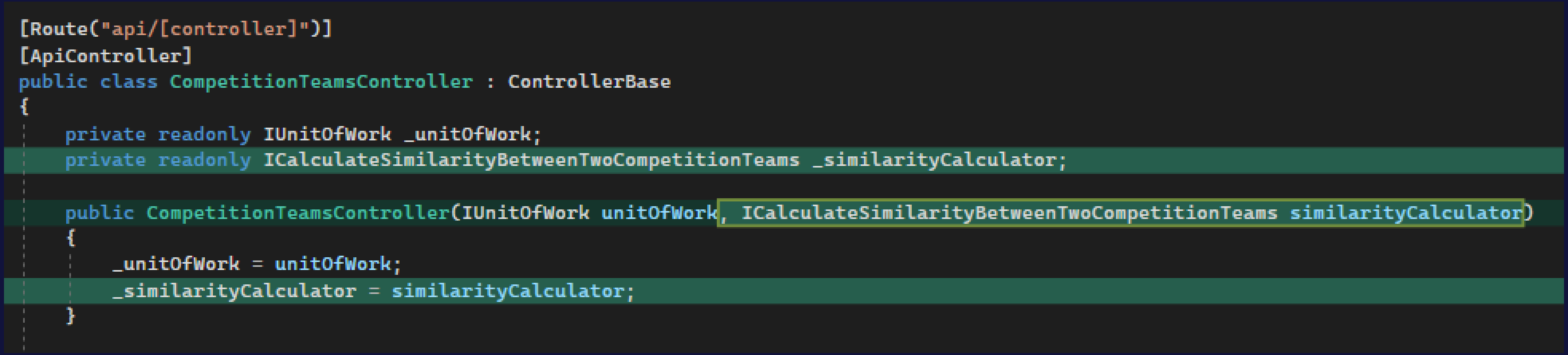


Startup.cs (e4df48c1)

Miscellaneous Files | sykkelkonken.Service.Startup

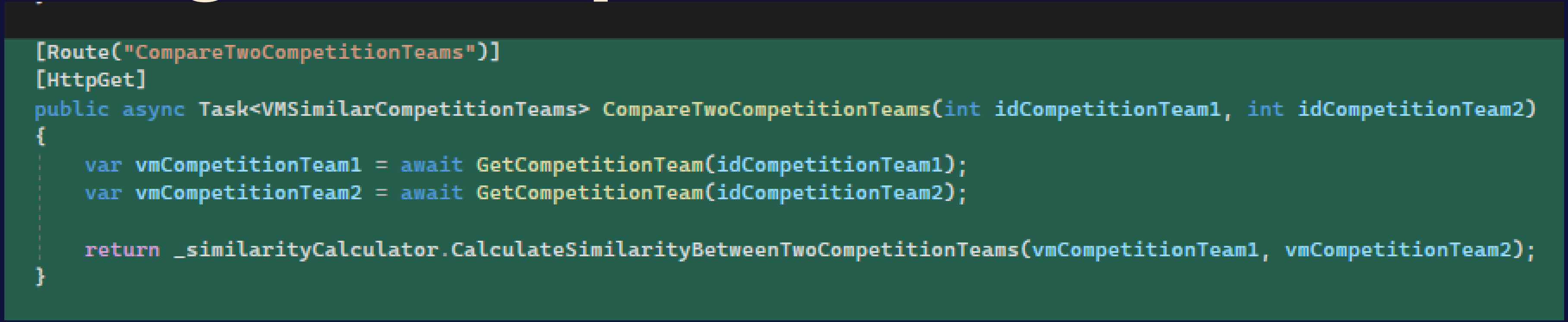
```
+           services.AddTransient<ICalculateSimilarity, SimilarityCalculator>();
+           services.AddTransient<ICompareTwoCompetitionTeams, CompetitionTeamComparer>();
+           services.AddTransient<ICalculateSimilarityBetweenTwoCompetitionTeams, TwoCompetitionTeamsSimilarityCalculator>();
+
+
            services.AddEndpointsApiExplorer();
            services.AddSwaggerGen();
```

Usage In Controller



```
[Route("api/[controller]")]
[ApiController]
public class CompetitionTeamsController : ControllerBase
{
    private readonly IUnitOfWork _unitOfWork;
    private readonly ICalculateSimilarityBetweenTwoCompetitionTeams _similarityCalculator;

    public CompetitionTeamsController(IUnitOfWork unitOfWork, ICalculateSimilarityBetweenTwoCompetitionTeams similarityCalculator)
    {
        _unitOfWork = unitOfWork;
        _similarityCalculator = similarityCalculator;
    }
```

Usage In Endpoint



```
[Route("CompareTwoCompetitionTeams")]
[HttpGet]
public async Task<VMSimilarCompetitionTeams> CompareTwoCompetitionTeams(int idCompetitionTeam1, int idCompetitionTeam2)
{
    var vmCompetitionTeam1 = await GetCompetitionTeam(idCompetitionTeam1);
    var vmCompetitionTeam2 = await GetCompetitionTeam(idCompetitionTeam2);

    return _similarityCalculator.CalculateSimilarityBetweenTwoCompetitionTeams(vmCompetitionTeam1, vmCompetitionTeam2);
}
```

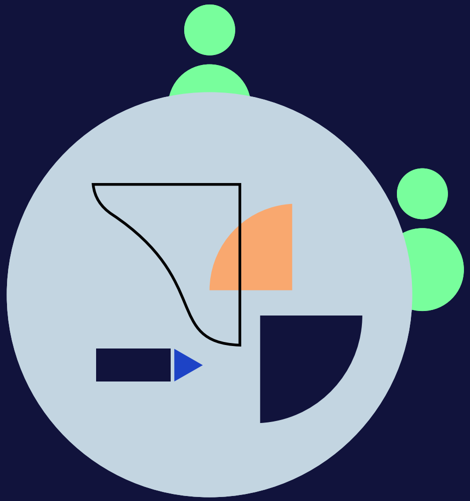
KEY REFLECTIONS AND TAKEAWAYS

- Testing makes you debug less
- Second guess you implementation
- Fun and challenging
- Practice, practice, practice

References

- Pedro Moreira Santos, Marco Consolaro and Alessandro Di Gioia. 2018-2019. Agile Technical Practices Distilled
- Lesson 4-SOLID++. Alcor Academy
- CodeAesthetic on youtube.
  Depencency Injection, the best pattern
  https://www.youtube.com/watch?v=J1f5b4vcxCQ

QUESTIONS?



THANK YOU!

Eirik Nysted
eirik.nysted@bouvet.no