Answer on Question#62454 – Programming – C#

Student.cs:

```
namespace FreelanceStudents
{
   using System;
   using System.Collections.Generic;

   public struct Student
   {
      public Guid Id { get; set; }

      public string Name { get; set; }

      public IEnumerable<DisciplineGrade> DisciplineGrades { get; set; }

      public override bool Equals(object obj)
      {
         return base.Equals(obj);
      }

      public bool Equals(Student other)
      {
         return Id.Equals(other.Id) && string.Equals(Name, other.Name) && Equals(DisciplineGrades,
other.DisciplineGrades);
      }

      public override int GetHashCode()
      {
         unchecked
         {
            var hashCode = Id.GetHashCode();
            hashCode = (hashCode*397) ^ (Name != null ? Name.GetHashCode() : 0);
            hashCode = (hashCode*397) ^ (DisciplineGrades != null ? DisciplineGrades.GetHashCode() : 0);
            return hashCode;
         }
      }
   }
}
```

DisciplineGrade.cs:

```
namespace FreelanceStudents
{
   using System;

   public struct DisciplineGrade
   {
      public Guid Id { get; set; }

      public string DisciplineName { get; set; }
```

```
        public int Grade { get; set; }

        public Guid StudentId { get; set; }

        public Student Student { get; set; }
    }
}
```

Program.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FreelanceStudents
{
    class Program
    {
        private const string MathName = "Math";
        private const string EnglishName = "English";
        private const string ScienceName = "Science";

        static void Main(string[] args)
        {
            Console.WriteLine("Enter number of students");
            var numberOfStudents = int.Parse(Console.ReadLine());
            var students = new Student[numberOfStudents];

            for (var i = 0; i < numberOfStudents; i++)
            {
                Console.WriteLine("Enter name of student");
                var studentName = Console.ReadLine();
                Console.WriteLine($"Enter grade in {MathName}");
                var mathGrade = int.Parse(Console.ReadLine());
                Console.WriteLine($"Enter grade in {EnglishName}");
                var englishGrade = int.Parse(Console.ReadLine());
                Console.WriteLine($"Enter grade in {ScienceName}");
                var scienceGrade = int.Parse(Console.ReadLine());

                var studentId = Guid.NewGuid();
                var student = new Student
                {
                    Id = studentId,
                    Name = studentName,
                    DisciplineGrades = new List<DisciplineGrade>
                    {
                        new DisciplineGrade
                        {
                            Id = Guid.NewGuid(),
```

```
            DisciplineName = MathName,
            Grade = mathGrade,
            StudentId = studentId
        },
        new DisciplineGrade
        {
            Id = Guid.NewGuid(),
            DisciplineName = EnglishName,
            Grade = englishGrade,
            StudentId = studentId
        },
        new DisciplineGrade
        {
            Id = Guid.NewGuid(),
            DisciplineName = ScienceName,
            Grade = scienceGrade,
            StudentId = studentId
        }
    }
  };

  students[i] = student;
}

foreach (var student in students)
{
  var averageGrade = student.DisciplineGrades.Average(dg => dg.Grade);

  var studentGrades = GetGradesForStudent(student);

  Console.WriteLine($"{studentGrades}");
  Console.WriteLine($"{student.Name} average grade is: {averageGrade}");

  if (averageGrade > 60)
  {
    Console.WriteLine($"{student.Name} has gained over 60%! Congratulations!");
  }
}

Console.WriteLine("Search for grades of student: (enter name of student)");
var studentSearchName = Console.ReadLine();

var searchStudentItem = students.FirstOrDefault(st => st.Name == studentSearchName);

Console.WriteLine(searchStudentItem.Equals(default(Student))
  ? "Student with provided name has not found"
  : GetGradesForStudent(searchStudentItem));

Console.WriteLine("Enter subject to find greatest grade");
var subjectToFindGrade = Console.ReadLine();
```

```
        var searchSubjectItems = students.SelectMany(st => st.DisciplineGrades.Where(dg => dg.DisciplineName
== subjectToFindGrade));
        var subjectItemWithMaxGrade =
            searchSubjectItems.First(si => si.Grade == searchSubjectItems.Max(i => i.Grade));
        subjectItemWithMaxGrade.Student = students.Single(st => st.Id ==
subjectItemWithMaxGrade.StudentId);

        Console.WriteLine(!searchSubjectItems.Any()
            ? "Subject with provided name has not found"
            : $"{subjectItemWithMaxGrade.Student.Name} has gained max score on subject
{subjectItemWithMaxGrade.DisciplineName}: {subjectItemWithMaxGrade.Grade}");


    }

    private static string GetGradesForStudent(Student student)
    {
        return $"{student.Name} " + string.Join(" | ", student.DisciplineGrades
            .Select(dg => string.Join(" ", dg.DisciplineName, ":", dg.Grade)));
    }
  }
}
```