

A detailed botanical illustration of various plants, including a tall stalk with a seed head, several green leafy stems, and a wheat-like grain, set against a light beige background.

CO453 Application Programming

Test Driven Development



App03: Student Grades - Features



1

Enter a mark
for each
student

2

Convert the
marks to
grades

3

Display all the
marks and
grades

4

Calculate and
display basic
marks statistics

5

Calculate and
display a grade
profile

Visual Paradigm: Community Edition

The screenshot displays the Visual Paradigm Enterprise software interface. On the left, a sidebar lists various modeling tools: DoDAF, NAF and MODAF (marked 'New'), Tabular (marked 'New'), LeSS and Nexus Canvas (marked 'New'), Form Builder, TOGAF ADM Tools, UML Diagrams, **Code Engineering** (highlighted in orange), Agile User Story Mapping, ERD & Database Engineering, UX Toolset, and BPMN. The main workspace shows a UML Class Diagram for a 'Sales Order System'. The diagram includes classes: Customer, Order, OrderStatus, OrderDetail, Item, and Payment. OrderStatus is an enumeration with states: CREATE, SHIPPING, DELIVERED, and PAID. Order is associated with Customer (1 to 0..*), OrderStatus (1 to 1), OrderDetail (1 to 1..*), and Payment (1 to 1..*). OrderDetail is associated with Item (0..* to 0..*). Payment is a generalization of Cash, Check, and WireTransfer. A 'Code Engineering' overlay is present at the bottom, featuring a 'Read More' button and a code icon (<code />). The overlay text reads: 'Code Engineering Online tools for collaborative diagramming, user story mapping, customer journey mapping and task management.'

Stay **Competitive** and **Responsive** to Change **Faster** & **Better** in the Digital World

Community Edition

[What's New](#) [Features](#) [Tutorials](#) [Support](#) [Pricing](#) [Try Now](#)

Download Visual Paradigm

No risk. No obligation. No registration. 30-day FREE Trial

Version: 16.1
Build number: 20200609

**Download Visual Paradigm
FREE Trial**

SSL Secure Connection

For Windows 64bit

[More Options](#) | [System Requirements](#) | [End User License Agreement](#)

Download Features List

[Product Leaflet](#) | [Brief Feature List](#) | [Full Feature List](#)

Windows

[Installer](#) [654MB]
[InstallFree](#) [648MB]

SSL Secure Connection

Download Visual Paradigm for other platform

[Linux](#) [Mac OS X](#)

Verify Download with [Checksum](#)

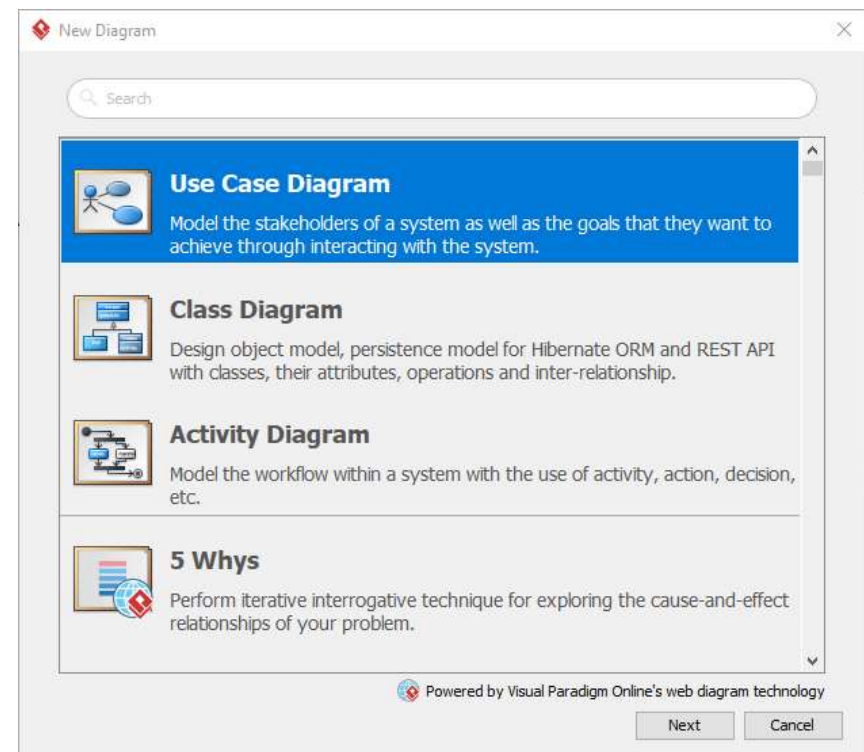
Other download

Get Community Edition
FREE for non-commercial use

Get VP-Viewer
FREE VP project browser

Download Old Versions

Create a new Project then
add a new Use Case
Diagram

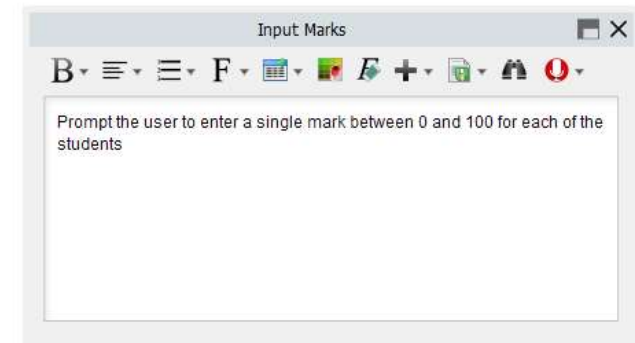
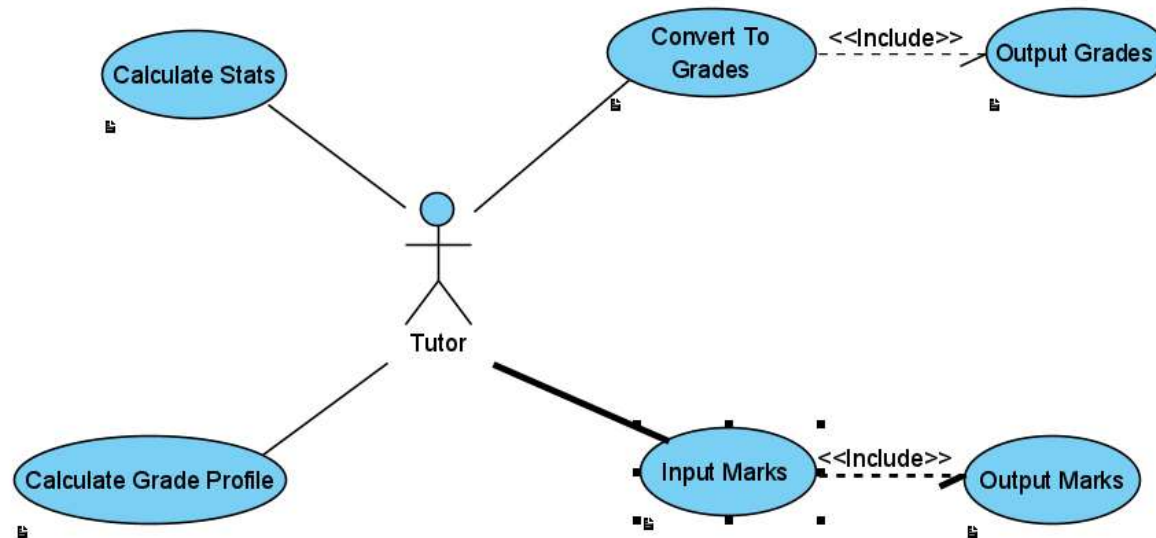


BNU CO453

4

UML: Use Case Diagram (analysis)

Use Case descriptions
should become
method comments



Use Cases can
become
Methods

Student Grades Class

```
/// <summary>
/// Grade A is First Class    : 70 - 100
/// Grade B is Upper Second  : 60 - 69
/// Grade C is Lower Second   : 50 - 59
/// Grade D is Third Class    : 40 - 49
/// Grade F is Fail           : 0 - 39
/// </summary>
```

```
0 references
public enum Grades
{
    [Description("Fail")]
    F,
    [Description("Third Class")]
    D,
    [Description("Lower Second")]
    C,
    [Description("Upper Second")]
    B,
    [Description("First Class")]
    A
}
```

What
grade is
missing?

Using public properties
makes testing easier

3 references

```
public class StudentGrades
{
    1 reference
    public string[] Students { get; set; }

    4 references | 0/4 passing
    public int [] Marks { get; set; }

    7 references | 0/1 passing
    public int [] GradeProfile { get; set; }

    1 reference | 0/1 passing
    public double Mean { get; set; }

    1 reference | 0/1 passing
    public int Minimum { get; set; }

    1 reference | 0/1 passing
    public int Maximum { get; set; }
}
```


StudentGrades Constructor

```
/// <summary>
/// Class Constructor called when an object
/// is created and sets up an array of students.
/// </summary>
3 references
public StudentGrades()
{
    Students = new string[]
    {
        "Daniel", "Dylan", "Eric",
        "Georgia", "Hasan", "Hamza",
        "Jack", "Liam", "Shan",
        "Shamial"
    };

    GradeProfile = new int[(int)Grade.A + 1];
    Marks = new int[Students.Length];
}
```

- Constructor is used to initialise variables to default values
- Use your own list of students names
- At least 10 student names
- An empty marks array is created
- An empty Grade Profile is created
- (int) casts a Grade to an int

Other Methods

```
/// <summary>
/// Input a mark between 0 - 100 for each
/// student and store it in the Marks array
/// </summary>
```

0 references

```
public void InputMarks()
{
    throw new NotImplementedException();
}
```

```
/// <summary>
/// List all the students and display their
/// name and current mark
/// </summary>
```

0 references

```
public void OutputMarks()
{
    throw new NotImplementedException();
}
```

```
/// <summary>
/// Convert a student mark to a grade
/// from F (Fail) to A (First Class)
/// </summary>
```

1 reference | 0/1 passing

```
public Grade ConvertToGrade(int mark)
{
    throw new NotImplementedException();
}
```

```
/// <summary>
/// Calculate and display the minimum, maximum
/// and mean mark for all the students
/// </summary>
```

3 references | 0/3 passing

```
public void CalculateStats()
{
    throw new NotImplementedException();
}
```


Methods Continued

```
/// <summary>
/// List all the students and display their
/// name, mark and grade
/// </summary>
```

0 references

```
public void OutputGrades()
{
    throw new NotImplementedException();
}
```

```
/// <summary>
/// Calculate and display the proportion of
/// students achieving each of the grades
/// </summary>
```

1 reference | 0/1 passing

```
public void CalculateGradeProfile()
{
    throw new NotImplementedException();
}
```

Test Driven Development (TDD)

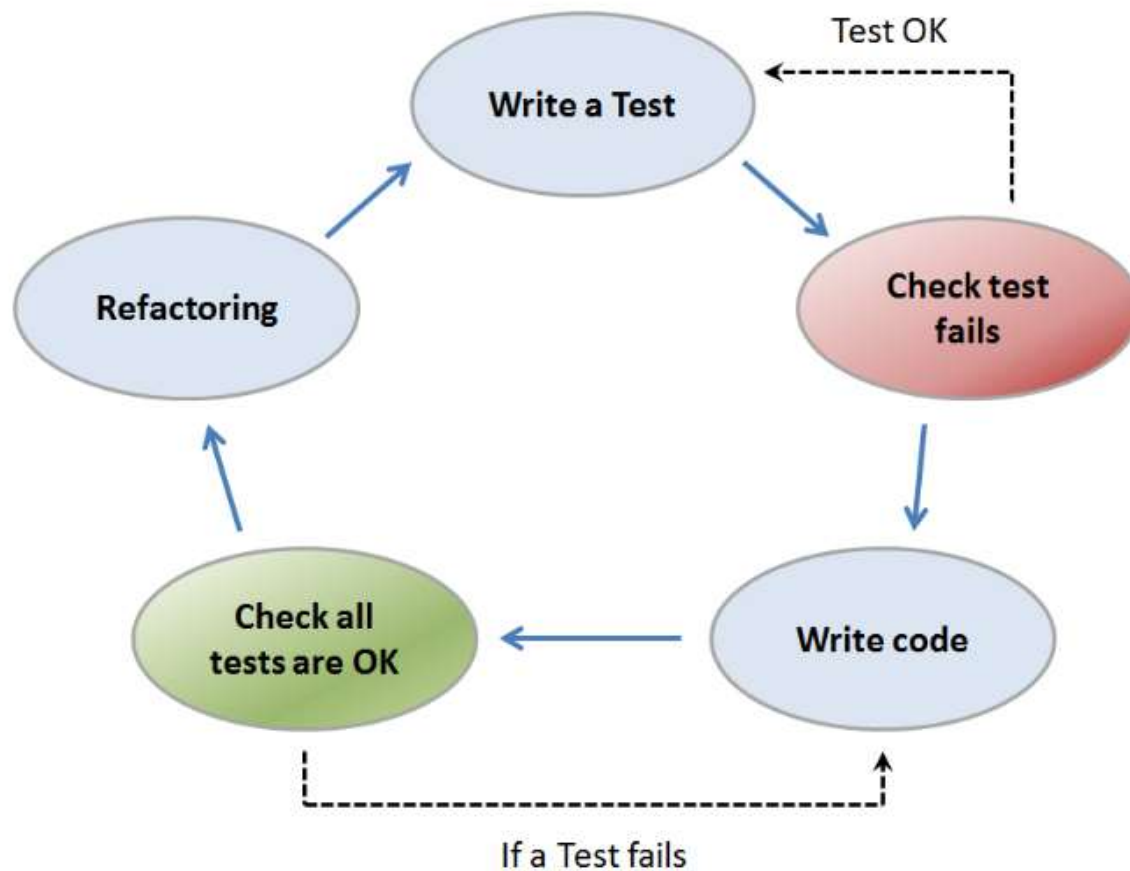


Figure 1 : TDD Cycle

jobsite



Asp.Net

Jobs > Asp.Net > South East > London

592 Asp.Net jobs in London + 20

How many tests are needed to check that any mark can be converted to the correct grade?

100 or 12 or 6?

jobsite



Asp.Net Tdd

Jobs > Asp.Net Tdd > South East > London

298 Asp.Net Tdd jobs in London;3

Testing ConvertToGrade()

```
/// <summary>
/// Grade A is First Class : 70 - 100
/// Grade B is Upper Second : 60 - 69
/// Grade C is Lower Second : 50 - 59
/// Grade D is Third Class : 40 - 49
/// Grade F is Fail : 0 - 39
/// </summary>
```

27 references

```
public enum Grade
{
    F, D, C, B, A
}
```

Only need to check the
boundary values

e.g. 70 is lowest A
100 is the highest A
Total = 10

Testing Convert0ToGrade()

[TestClass]

0 references

```
public class StudentGradesTest
```

```
{
```

```
    private readonly StudentGrades
```

```
    studentGrades = new StudentGrades();
```

Create a new
Test Class

```
public void Convert0ToGradeF()
```

```
{
```

```
    // Arrange
```

```
    Grade expectedGrade = Grade.F;
```

```
    // Act
```

```
    Grade actualGrade = studentGrades.ConvertToGrade(0);
```

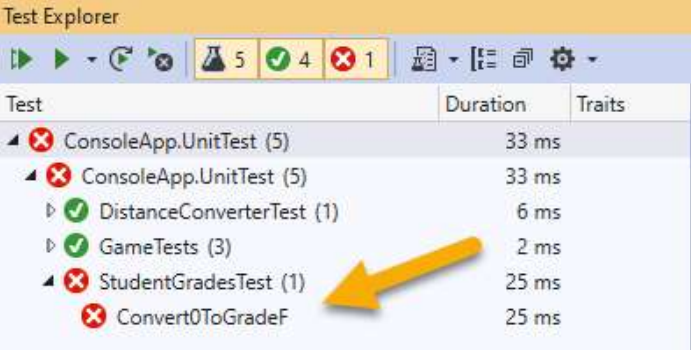
```
    // Assert
```

```
    Assert.AreEqual(expectedGrade, actualGrade);
```

```
}
```

Add Test which
fails

Add a test for each grade
boundary (10 tests)




Test	Duration	Traits
ConsoleApp.UnitTest (5)	33 ms	
ConsoleApp.UnitTest (5)	33 ms	
DistanceConverterTest (1)	6 ms	
GameTests (3)	2 ms	
StudentGradesTest (1)	25 ms	
Convert0ToGradeF	25 ms	

Convert0ToGrade()

```
2 references | 0/1 passing  
public Grade ConvertToGrade(int mark)  
{  
    if (mark >= 0 && mark < 40)  
    {  
        return Grade.F;  
    }  
    else  
    {  
        return Grade.D;  
    }  
}
```

Add minimum code till
the test passes

The **magic** numbers need to be
replaced by constants



Test	Duration	Traits
✓ ConsoleApp.UnitTest (5)	9 ms	
✓ ConsoleApp.UnitTest (5)	9 ms	
✓ DistanceConverterTest (1)	7 ms	
✓ GameTests (3)	2 ms	
✓ StudentGradesTest (1)	< 1 ms	
✓ Convert0ToGradeF	< 1 ms	

ConvertToGrade() - Refactored

Refactor magic numbers
by using constants


/ references

```
public class StudentGrades
{
    public const int LowestMark = 0;
    public const int HighestMark = 100;

    public const int LowestGradeD = 40;
    public const int LowestGradeC = 50;
    public const int LowestGradeB = 60;
    public const int LowestGradeA = 70;
}
```

2 references | 1/1 passing

```
public Grade ConvertToGrade(int mark)
{
    if (mark >= LowestMark &&
        mark < LowestGradeD)
    {
        return Grade.F;
    }
    else
    {
        return Grade.D;
    }
}
```

Test Explorer		
		
Test	Duration	Traits
ConsoleApp.UnitTest (5)	9 ms	
ConsoleApp.UnitTest (5)	9 ms	
DistanceConverterTest (1)	7 ms	
GameTests (3)	2 ms	
StudentGradesTest (1)	< 1 ms	
ConvertToGradeF	< 1 ms	

TestCalculateMean()

[TestMethod]

0 references

```
public void TestCalculateMean()
```

```
{
```

```
    // Arrange
```

```
    int[] statsMarks = new int[]
```

```
    {
```

```
        10, 20, 30, 40, 50, 60, 70, 80, 90, 100
```

```
    };
```

```
    studentGrades.Marks = statsMarks;
```

```
    double expectedMean = 55.0;
```

```
    // Act
```

```
    studentGrades.CalculateStats();
```

```
    // Assert
```

```
    Assert.AreEqual(expectedMean, studentGrades.Mean);
```

```
}
```

No code exists
so the test fails

3 references | 0/2 passing

```
public void CalculateStats()
```

```
{
```

```
    double total = 0;
```

```
    foreach(int mark in Marks)
```

```
    {
```

```
        total += mark;
```

```
    }
```

```
    Mean = total / Marks.Length;
```

```
}
```

TestCalculateMin()

```
public void TestCalculateMin()
{
    // Arrange
    studentGrades.Marks = StatsMarks;
    int expectedMin = 10;

    // Act
    studentGrades.CalculateStats();

    // Assert
    Assert.AreEqual(expectedMin, studentGrades.Minimum);
}
```

[TestClass]

0 references

public class StudentGradesTest

{

private readonly StudentGrades

studentGrades = new StudentGrades();

private readonly int[] StatsMarks = new int[]

{

10, 20, 30, 40, 50, 60, 70, 80, 90, 100

};

studentGrades and
statsMarks moved so that
they can be used by more
than one test method

CalculateStats()

3 references | 3/3 passing

```
public void CalculateStats()
{
    Minimum = Marks[0];
    Maximum = Marks[0];

    double total = 0;

    foreach(int mark in Marks)
    {
        if (mark > Maximum) Maximum = mark;
        if (mark < Minimum) Minimum = mark;
        total += mark;
    }

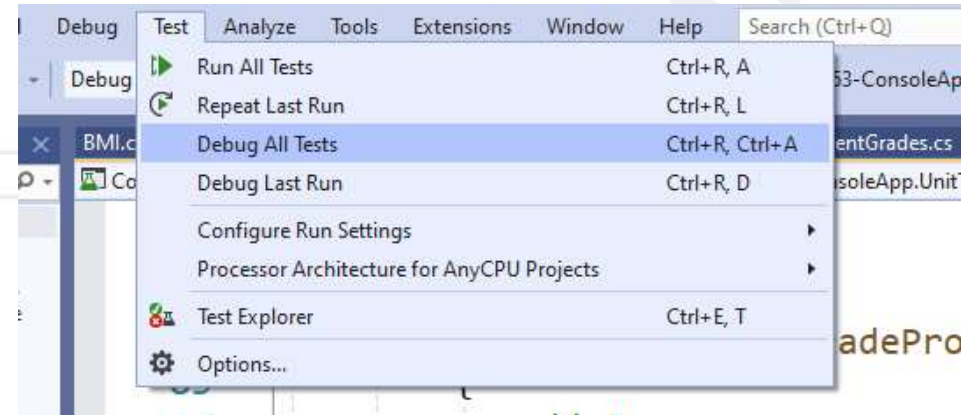
    Mean = total / Marks.Length;
}
```

Code for Calculate Stats
is added until it passes

Test	Duration	Traits
ConsoleApp.UnitTest (7)	12 ms	
ConsoleApp.UnitTest (7)	12 ms	
DistanceConverterTest (1)	8 ms	
GameTests (3)	3 ms	
StudentGradesTest (3)	1 ms	
Convert0ToGradeF	1 ms	
TestCalculateMean	< 1 ms	
TestCalculateMin	< 1 ms	

Debugging – Add a Breakpoint

```
117 public void CalculateStats()  
118 {  
119     Minimum = Marks[0];  
120     Maximum = Marks[0];  
121  
122     double total = 0;  
123  
124     foreach(int mark in Marks)  
125     {  
126         if (mark > Maximum) Maximum = mark;  
127         if (mark < Minimum) Minimum = mark;  
128         total += mark;  
129     }  
130  
131     Mean = total / Marks.Length;  
132 }  
133
```



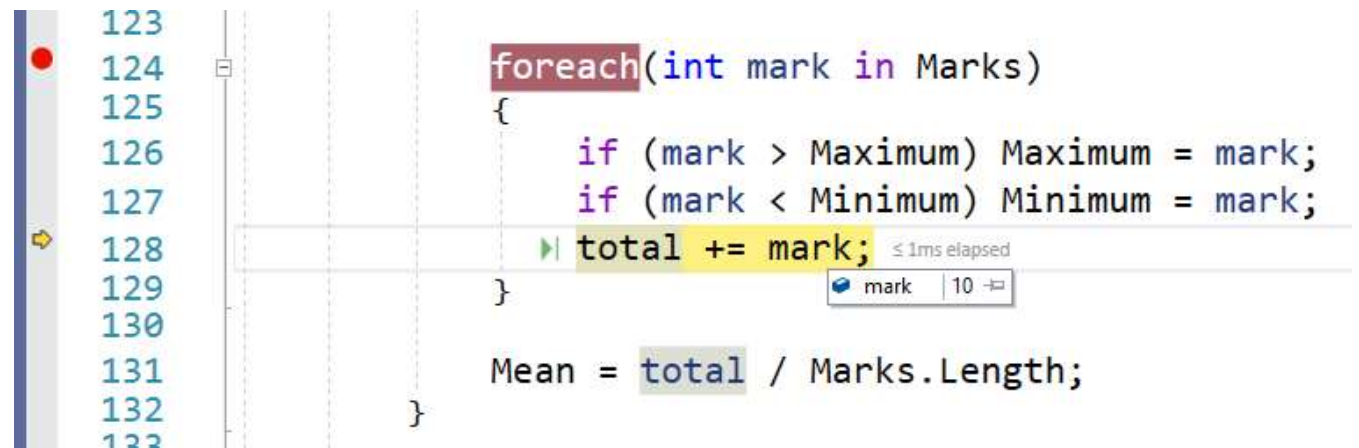
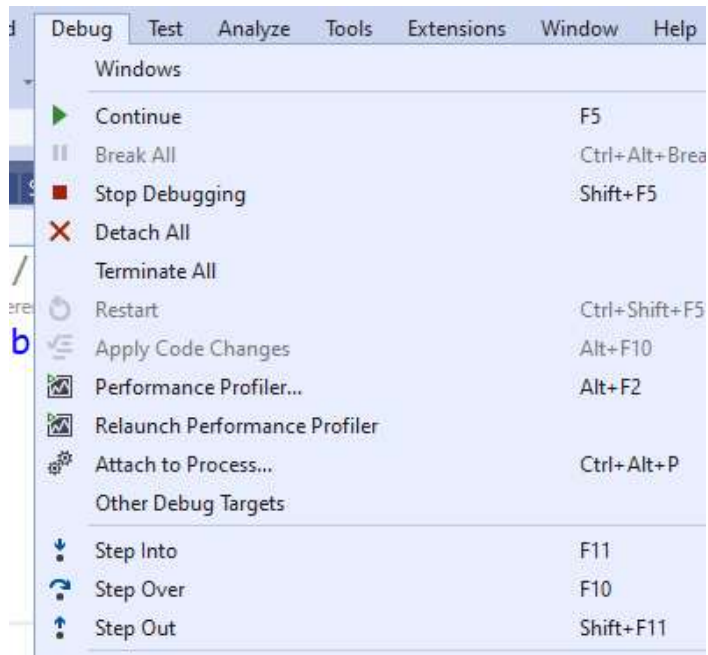
Debugging – Inspecting variables

```
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130
```

```
public void CalculateStats()  
{  
    Minimum = Marks[0];  
    Maximum = Marks[9];  
  
    double total = 0;  
  
    foreach (int mark in Marks)  
    {  
        if (mark > Maximum) Maximum = mark;  
        if (mark < Minimum) Minimum = mark;  
        total += mark;  
    }  
}
```

Locals	
Search (Ctrl+E) 🔍 ⏪ ⏩ Search Depth: 3 🔽 A	
Name	Value
⏏ this	{CO453_ConsoleApp}
⏏ Grade	X
▸ ⏏ GradeProfile	{int[7]}
▸ ⏏ Marks	{int[10]}
⏏ Maximum	10
⏏ Mean	0
⏏ Minimum	10
▸ ⏏ Students	{string[10]}
▸ ⏏ Static members	
⏏ total	0

Debugging – Stepping Over



TestCalculateGradeProfile

[TestMethod]

0 references

```
public void TestGradeProfile()
```

```
{
```

```
    // Arrange
```

```
    studentGrades.Marks = StatsMarks;
```

```
    bool expectedProfile = false;
```

```
    // Act
```

```
    studentGrades.CalculateGradeProfile();
```

```
    expectedProfile = ((studentGrades.GradeProfile[0] == 3) &&  
        (studentGrades.GradeProfile[1] == 1) &&  
        (studentGrades.GradeProfile[2] == 1) &&  
        (studentGrades.GradeProfile[3] == 1) &&  
        (studentGrades.GradeProfile[4] == 4));
```

```
    // Assert
```

```
    Assert.IsTrue(expectedProfile);
```

```
}
```

[TestClass]

0 references

```
public class StudentGradesTest
```

```
{
```

```
    private readonly StudentGrades
```

```
        studentGrades = new StudentGrades();
```

```
    private readonly int[] StatsMarks = new int[]
```

```
    {
```

```
        10, 20, 30, 40, 50, 60, 70, 80, 90, 100
```

```
    };
```

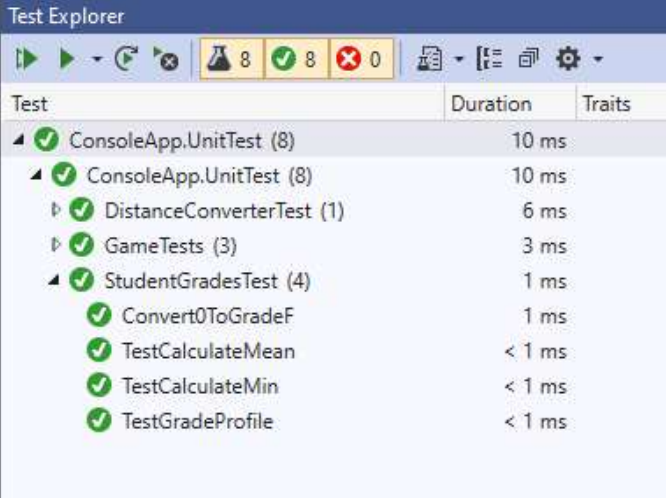
No code exists
so the test fails

CalculateGradeProfile()

```
public void CalculateGradeProfile()
{
    for(int i = 0; i < GradeProfile.Length; i++)
    {
        GradeProfile[i] = 0;
    }

    foreach(int mark in Marks)
    {
        Grade grade = ConvertToGrade(mark);
        GradeProfile[(int)grade]++;
    }

    OutputGradeProfile();
}
```



Test	Duration	Traits
✓ ConsoleApp.UnitTest (8)	10 ms	
✓ ConsoleApp.UnitTest (8)	10 ms	
✓ DistanceConverterTest (1)	6 ms	
✓ GameTests (3)	3 ms	
✓ StudentGradesTest (4)	1 ms	
✓ Convert0ToGradeF	1 ms	
✓ TestCalculateMean	< 1 ms	
✓ TestCalculateMin	< 1 ms	
✓ TestGradeProfile	< 1 ms	

The enumeration
Grade is stored
internally as an integer
but still needs to be
cast as an int

OutputGradeProfile()

```
private void OutputGradeProfile()
{
    Grade grade = Grade.X;
    Console.WriteLine();

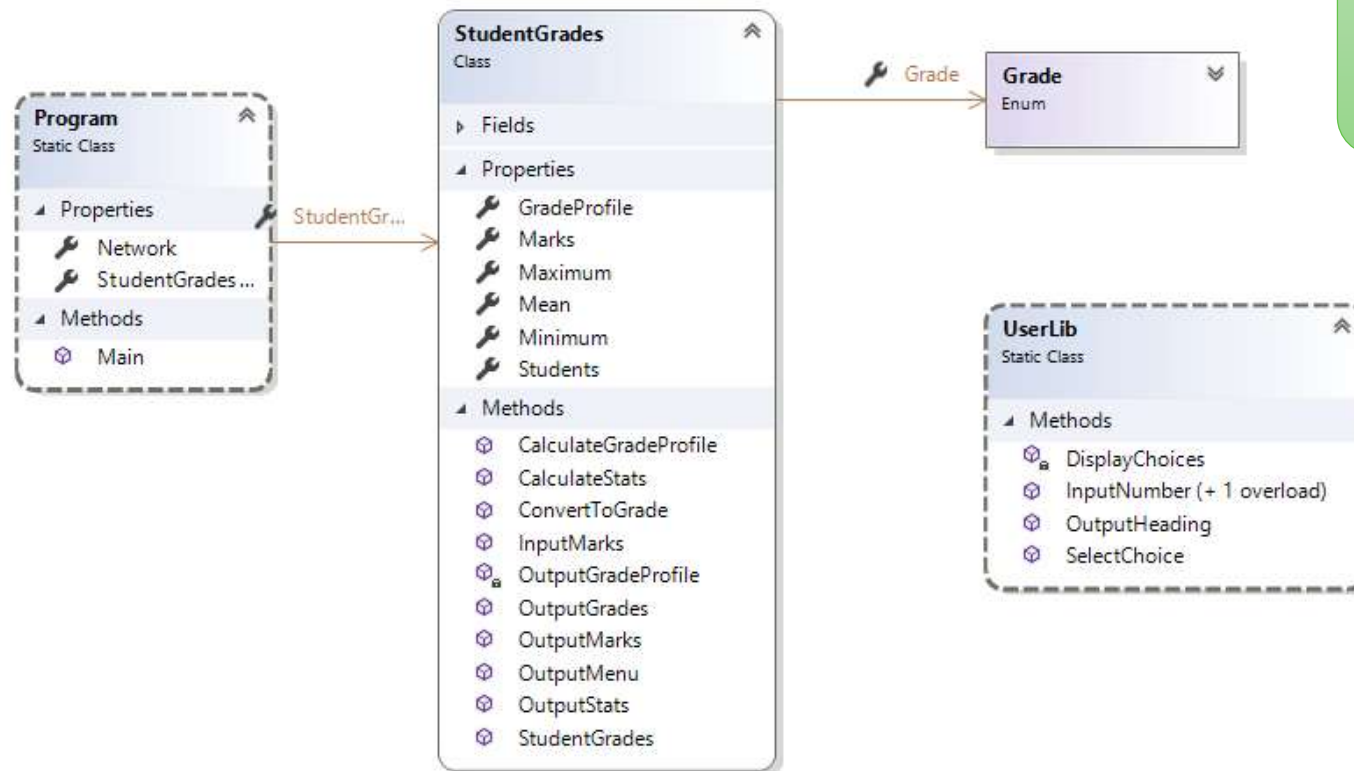
    foreach(int count in GradeProfile)
    {
        int percentage = count * 100 / Marks.Length;
        Console.WriteLine($"Grade {grade} {percentage}% Count {count}");
        grade++;
    }

    Console.WriteLine();
}
```

Remaining methods involve user input and output that cannot be Unit Tested

Automated Testing records user input, but the tools are not available with community edition of Visual Studio

UML Class Diagram



Final Design

Independent Study



App01 Distance Converter

Finalized and Documented



App02 BMI Calculator

Finalized and Documented



App03 Student Grades

Complete the application methods to process student marks into grades and all unit tests