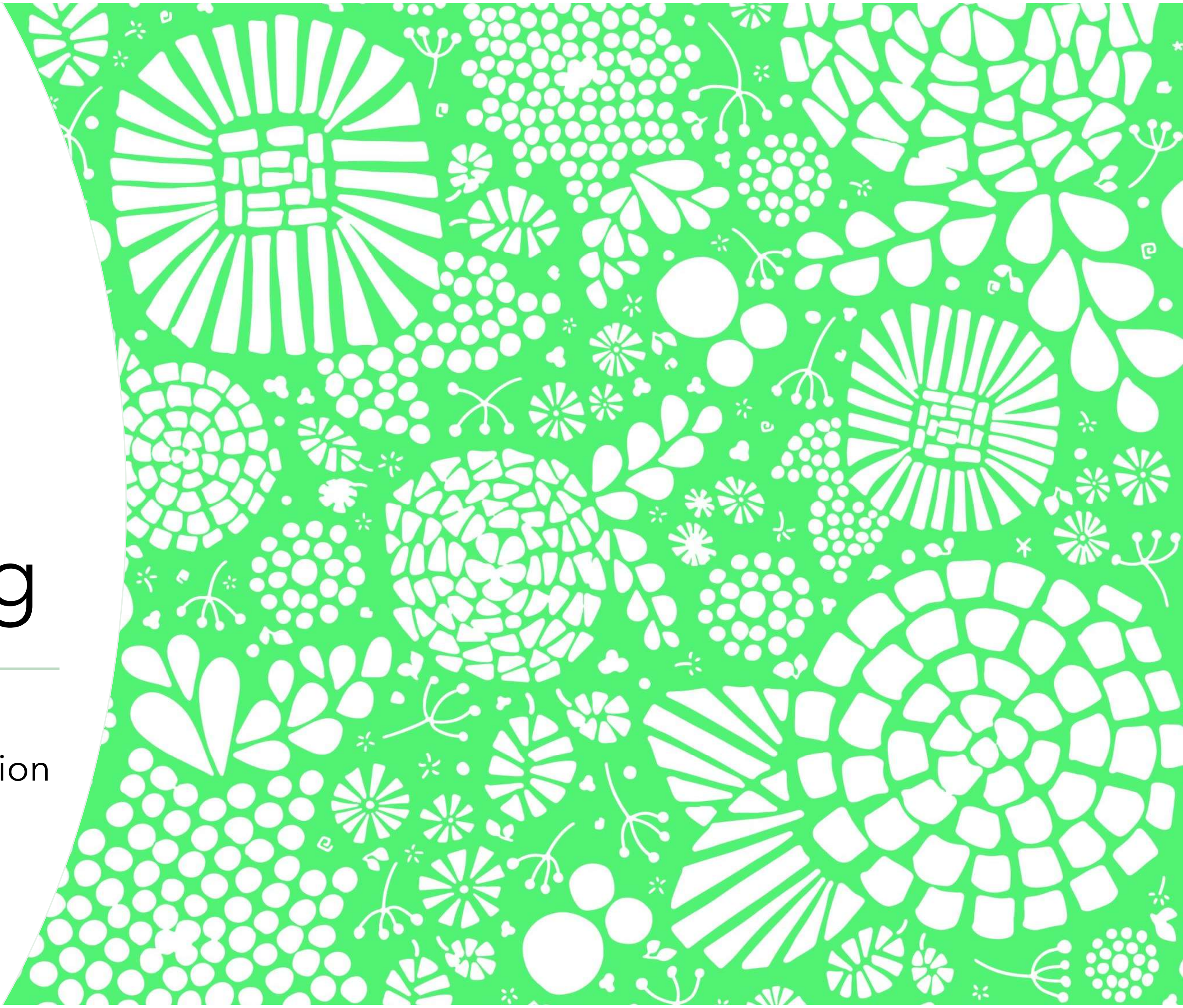
CO453 Application Programming

Session 2: Refactoring and Selection



Session Summary

- The principles of creating readable and maintainable code
- Refactoring code to reduce code “smells”
- Code metrics to measure quality
- C# method return values and parameters
- C# Selecting alternative actions
- Black box testing of code

Programming Best Practice

See "Coding Guide"
In the Wiki

Microsoft Naming Conventions

Architectural Principles

Writing Readable Code

Code Refactoring

Agile Alliance

Test Driven Development

Readable Code

Link

- Good Comments
- Consistent Indentation
- Group Code Lines
- Consistent Naming
- Short Lines, Short Methods
- Naming Files & Folders

Architectural Principles

Link

- Separation of Concerns
- Single Responsibility
- Encapsulation
- (DRY) Don’t Repeat Yourself

Pair Programming (Extreme Programming)

Definition

- Pair programming consists of two programmers sharing a single workstation (one screen, keyboard and mouse among the pair). The programmer at the keyboard is usually called the “driver”, the other, also actively involved in the programming task but focusing more on overall direction is the “navigator”; it is expected that the programmers swap roles every few minutes or so.

Swap every 30 minutes

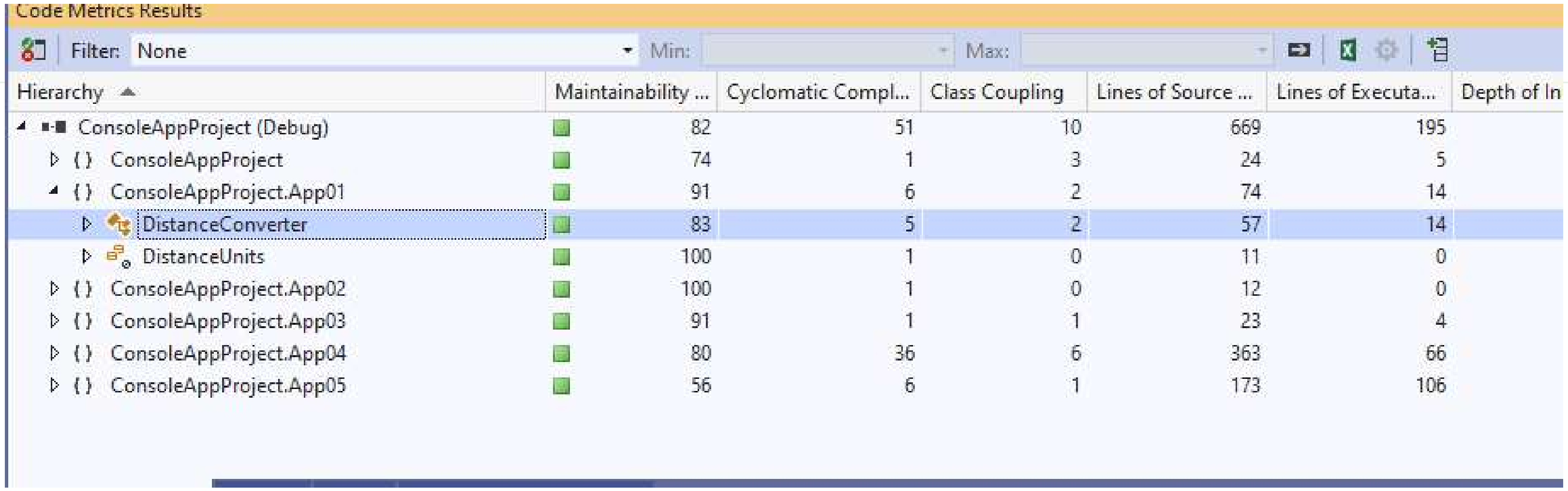
Example Dialog

- **Driver**: I am going to add a method called “Foo”
- **Navigator**: Would it not be better called “Bar”?
- **Driver**: I am going to add some code to do “this”
- **Navigator**: I think that would make the method too long!

Pair Programming - Practice

- Each students gets the same marks.
- Both names added to each class and to shared repository.
- Students can “divorce” or “separate” if not happy.
- Same pair for only one single App.

Measuring Quality: Code Metrics


Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability ...	Cyclomatic Compl...	Class Coupling	Lines of Source ...	Lines of Executa...	Depth of In
ConsoleAppProject (Debug)	🟩 82	51	10	669	195	
ConsoleAppProject	🟩 74	1	3	24	5	
ConsoleAppProject.App01	🟩 91	6	2	74	14	
DistanceConverter	🟩 83	5	2	57	14	
DistanceUnits	🟩 100	1	0	11	0	
ConsoleAppProject.App02	🟩 100	1	0	12	0	
ConsoleAppProject.App03	🟩 91	1	1	23	4	
ConsoleAppProject.App04	🟩 80	36	6	363	66	
ConsoleAppProject.App05	🟩 56	6	1	173	106	

Maintainability
- Green 20 -1 00
- Yellow 10 – 19
- Red 0 – 9
- (Higher the better)

Complexity
- Lower the better

Lines of Code
- Lower the better

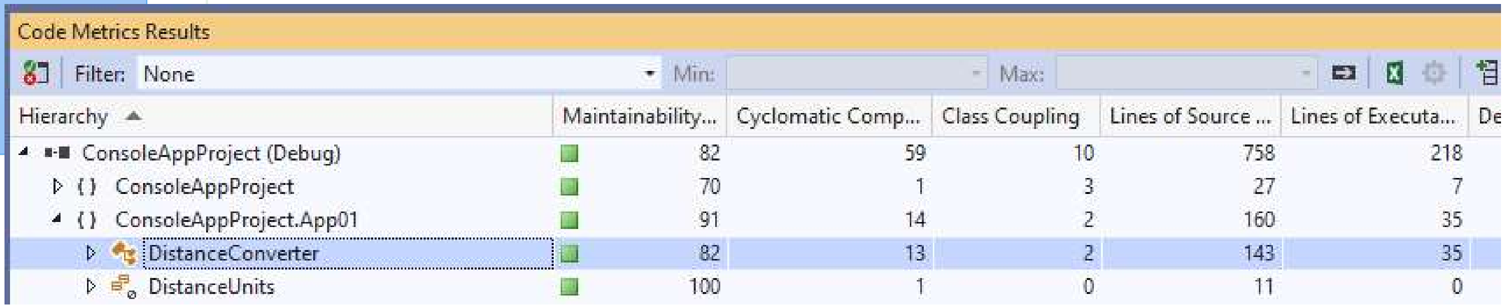
Coupling
- Lower the better

Inheritance Depth
- Lower the better

DistanceConverter: Week 1 Issues

In adding two more features to the original program converting miles to feet we are left with three possible issues

- User can't select between 3 conversions
- OutputHeading() may be too Specific
- Contains unnecessary duplication and needs:
  - General Input Distance
  - General Output Distance
  - General Convert Distance



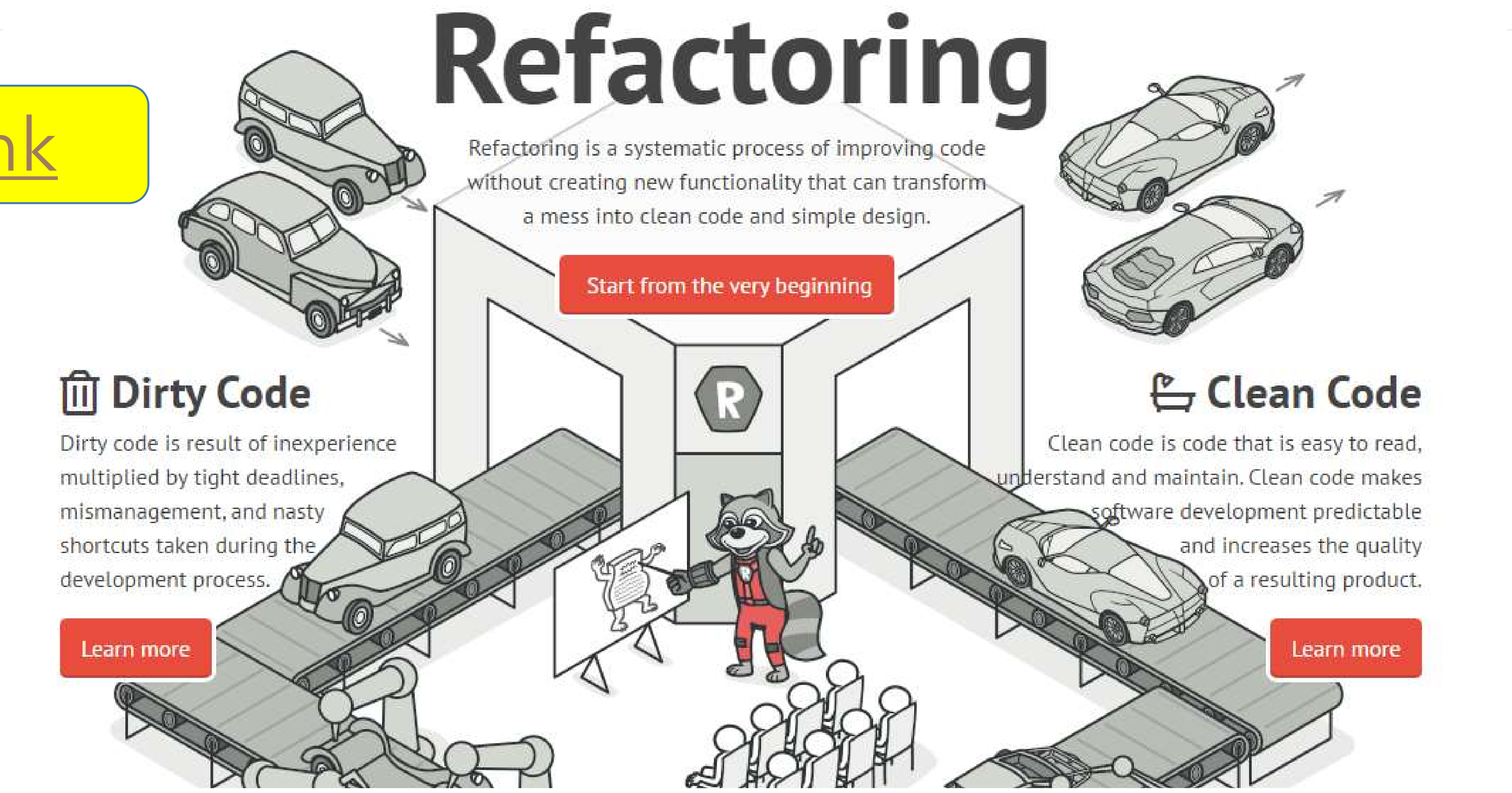
Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability...	Cyclomatic Comp...	Class Coupling	Lines of Source ...	Lines of Executa...	De
ConsoleAppProject (Debug)	🟩 82	59	10	758	218	
ConsoleAppProject	🟩 70	1	3	27	7	
ConsoleAppProject.App01	🟩 91	14	2	160	35	
DistanceConverter	🟩 82	13	2	143	35	
DistanceUnits	🟩 100	1	0	11	0	

Refactoring – Quality Improvement

Link



Refactoring

Refactoring is a systematic process of improving code without creating new functionality that can transform a mess into clean code and simple design.

Start from the very beginning

Dirty Code

Dirty code is result of inexperience multiplied by tight deadlines, mismanagement, and nasty shortcuts taken during the development process.

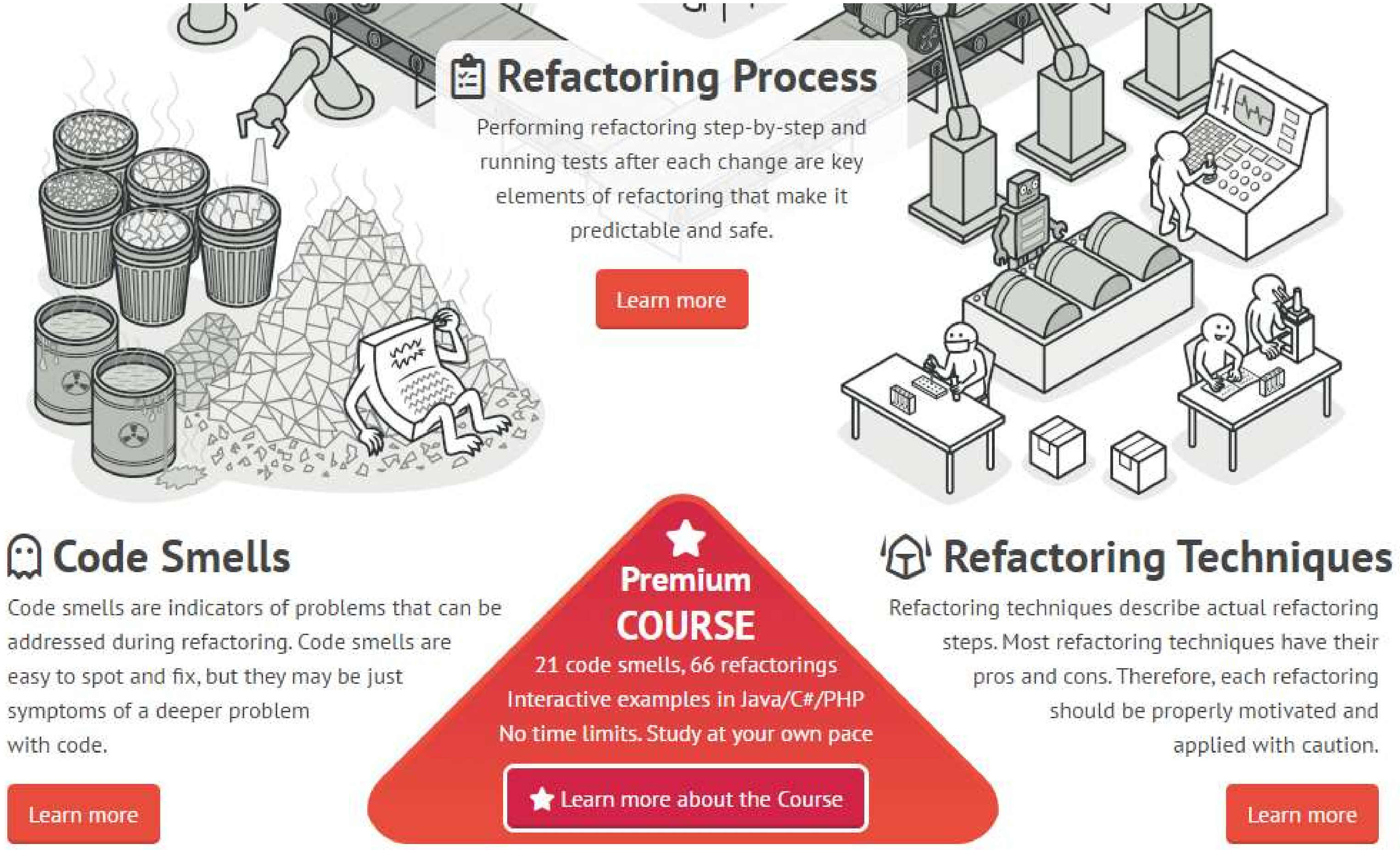
Learn more

Clean Code

Clean code is code that is easy to read, understand and maintain. Clean code makes software development predictable and increases the quality of a resulting product.

Learn more

refactoring.guru/refactoring



Refactoring Process

Performing refactoring step-by-step and running tests after each change are key elements of refactoring that make it predictable and safe.

Learn more

Code Smells

Code smells are indicators of problems that can be addressed during refactoring. Code smells are easy to spot and fix, but they may be just symptoms of a deeper problem with code.

Learn more

Premium COURSE

21 code smells, 66 refactorings
Interactive examples in Java/C#/PHP
No time limits. Study at your own pace

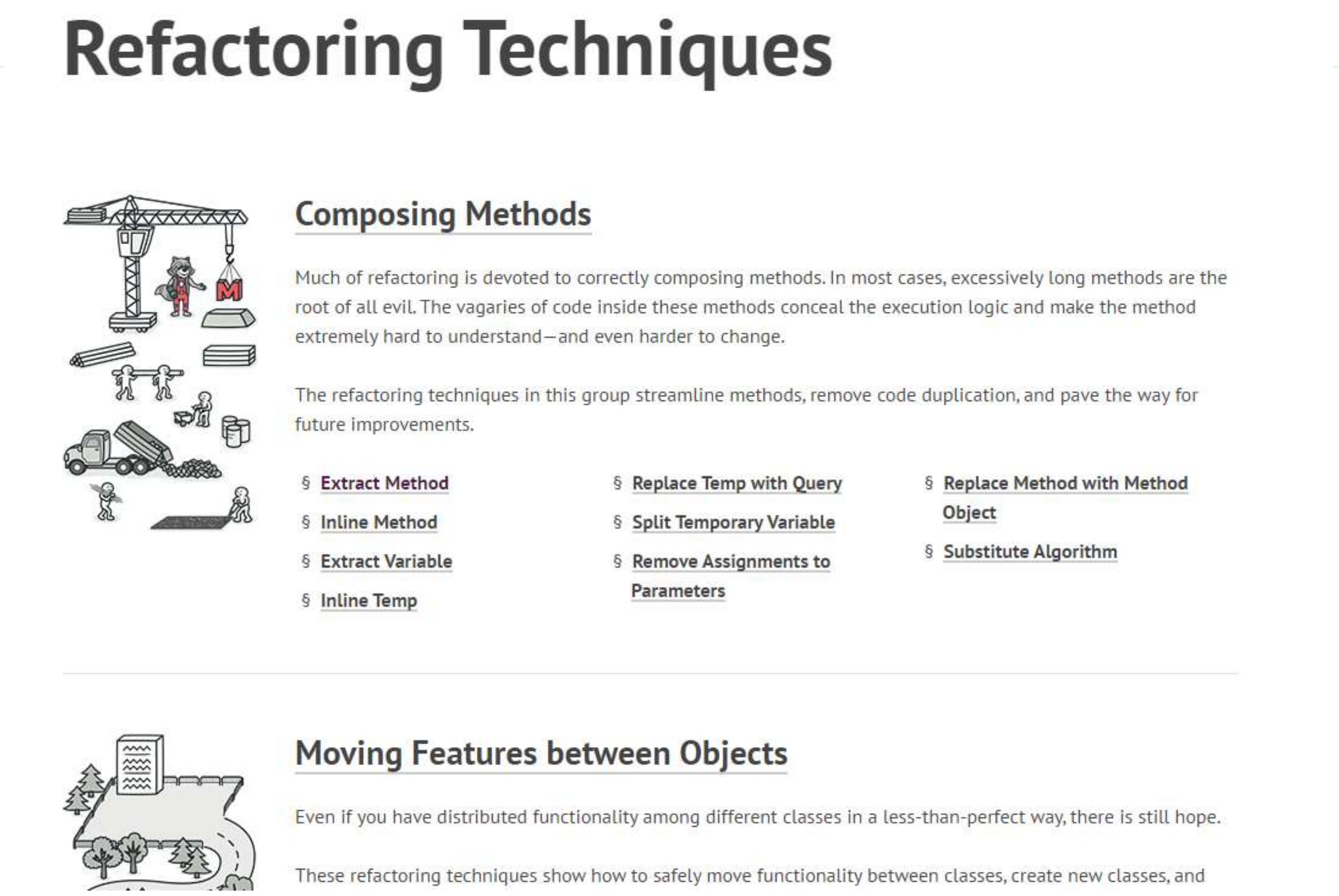
Learn more about the Course

Refactoring Techniques

Refactoring techniques describe actual refactoring steps. Most refactoring techniques have their pros and cons. Therefore, each refactoring should be properly motivated and applied with caution.

Learn more

68 Ways to Refactor



Refactoring Techniques

Composing Methods

Much of refactoring is devoted to correctly composing methods. In most cases, excessively long methods are the root of all evil. The vagaries of code inside these methods conceal the execution logic and make the method extremely hard to understand—and even harder to change.

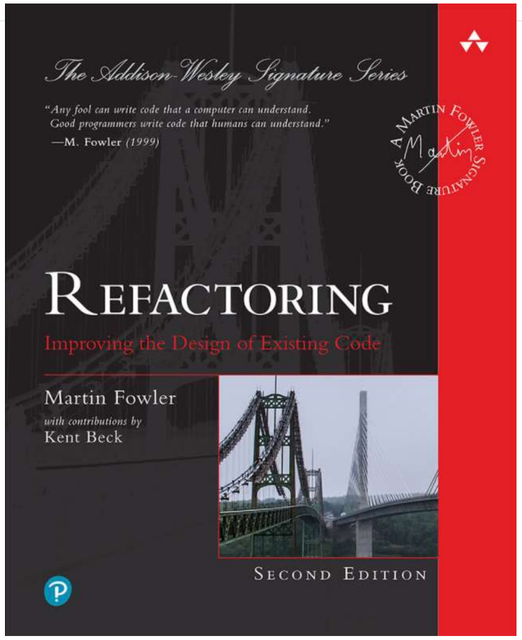
The refactoring techniques in this group streamline methods, remove code duplication, and pave the way for future improvements.

- § Extract Method
- § Inline Method
- § Extract Variable
- § Inline Temp
- § Replace Temp with Query
- § Split Temporary Variable
- § Remove Assignments to Parameters
- § Replace Method with Method Object
- § Substitute Algorithm

Moving Features between Objects

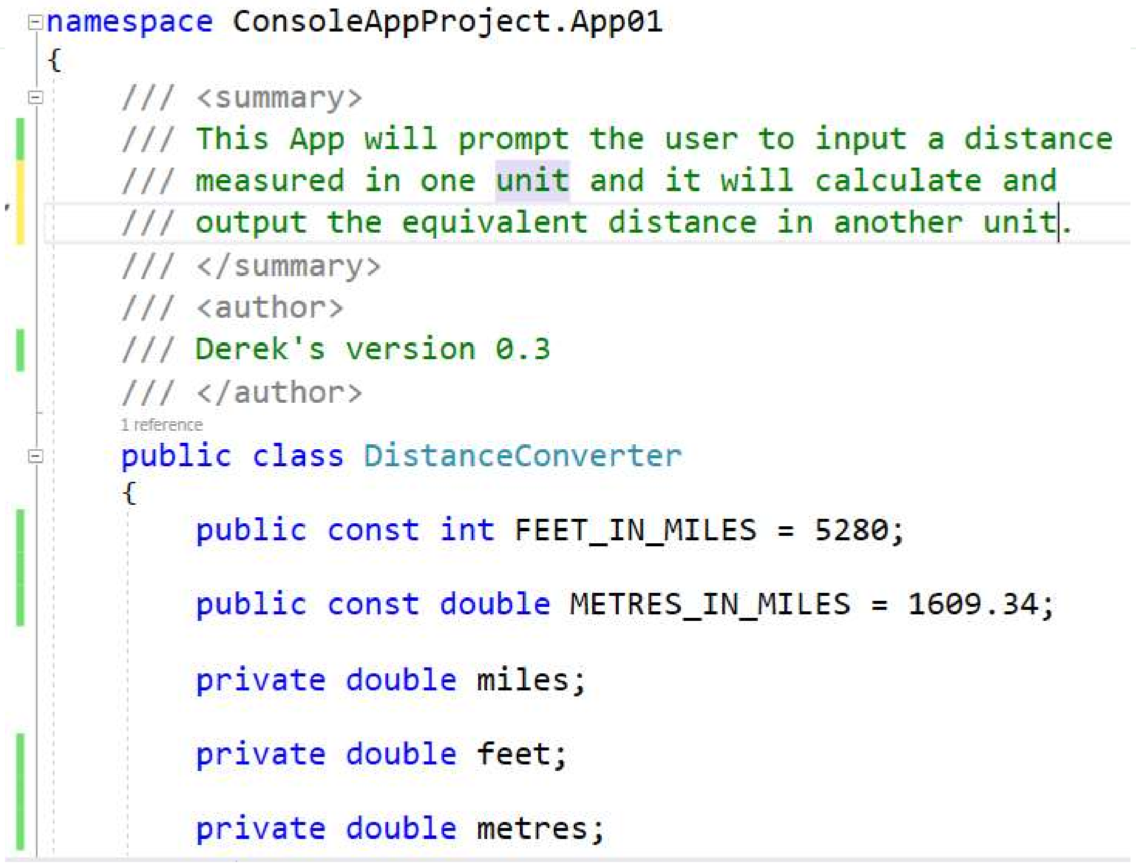
Even if you have distributed functionality among different classes in a less-than-perfect way, there is still hope.

These refactoring techniques show how to safely move functionality between classes, create new classes, and


The Addison-Wesley Signature Series
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand." —M. Fowler (1999)
A Martin Fowler Signature Book
Refactoring
Improving the Design of Existing Code
Martin Fowler
with contributions by Kent Beck
Second Edition

Click on book

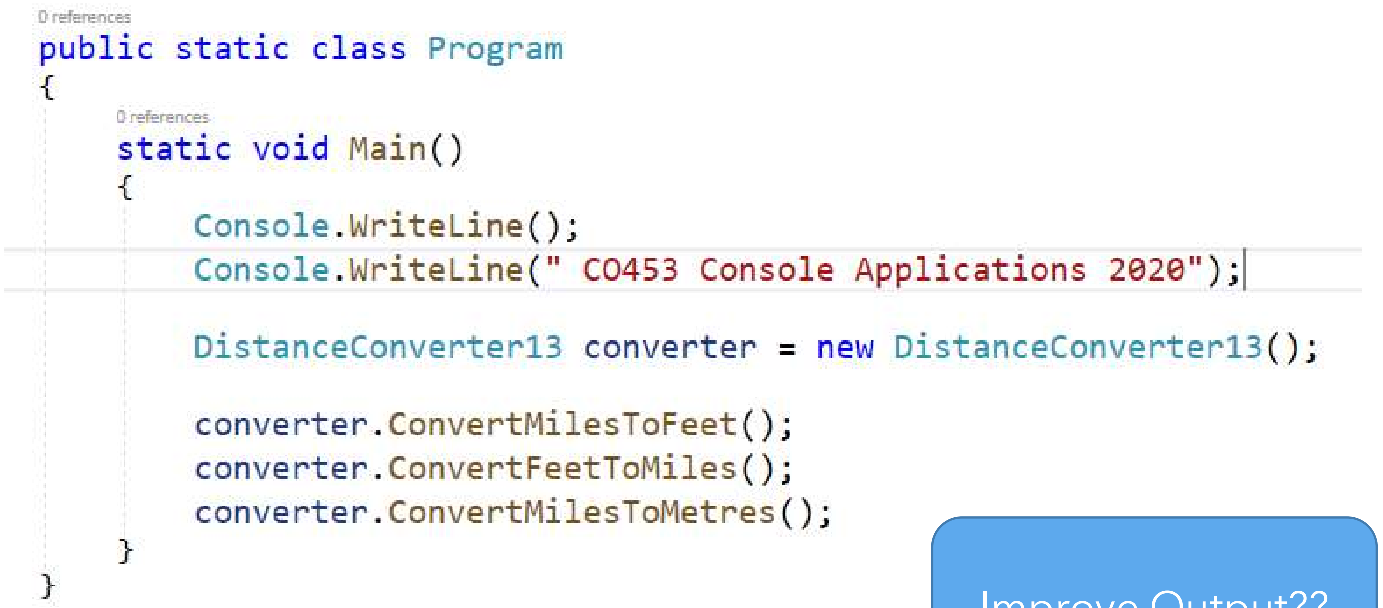
Distance Converter Version 0.3



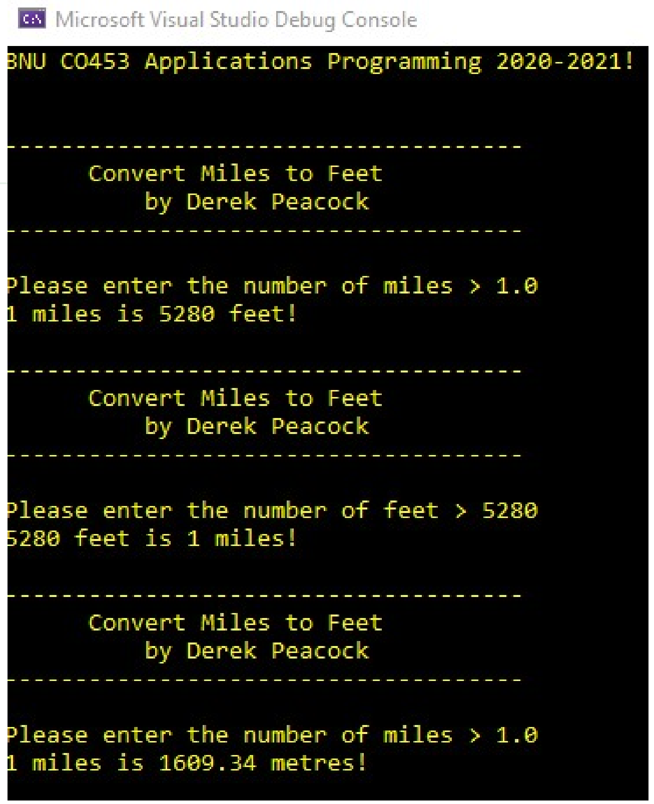
```
namespace ConsoleAppProject.App01
{
    /// <summary>
    /// This App will prompt the user to input a distance
    /// measured in one unit and it will calculate and
    /// output the equivalent distance in another unit.
    /// </summary>
    /// <author>
    /// Derek's version 0.3
    /// </author>
    1 reference
    public class DistanceConverter
    {
        public const int FEET_IN_MILES = 5280;

        public const double METRES_IN_MILES = 1609.34;

        private double miles;

        private double feet;

        private double metres;
```

Did you remember to update the class comment?

Testing the three features



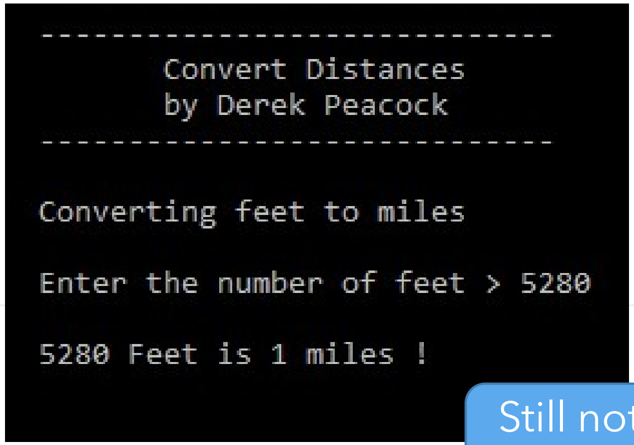
```
0 references
public static class Program
{
    0 references
    static void Main()
    {
        Console.WriteLine();
        Console.WriteLine(" CO453 Console Applications 2020");

        DistanceConverter13 converter = new DistanceConverter13();

        converter.ConvertMilesToFeet();
        converter.ConvertFeetToMiles();
        converter.ConvertMilesToMetres();
    }
}
```



```
BNU CO453 Applications Programming 2020-2021!


 -------------------------------------
       Convert Miles to Feet
           by Derek Peacock
 -------------------------------------

Please enter the number of miles > 1.0
1 miles is 5280 feet!

 -------------------------------------
       Convert Miles to Feet
           by Derek Peacock
 -------------------------------------

Please enter the number of feet > 5280
5280 feet is 1 miles!

 -------------------------------------
       Convert Miles to Feet
           by Derek Peacock
 -------------------------------------

Please enter the number of miles > 1.0
1 miles is 1609.34 metres!
```

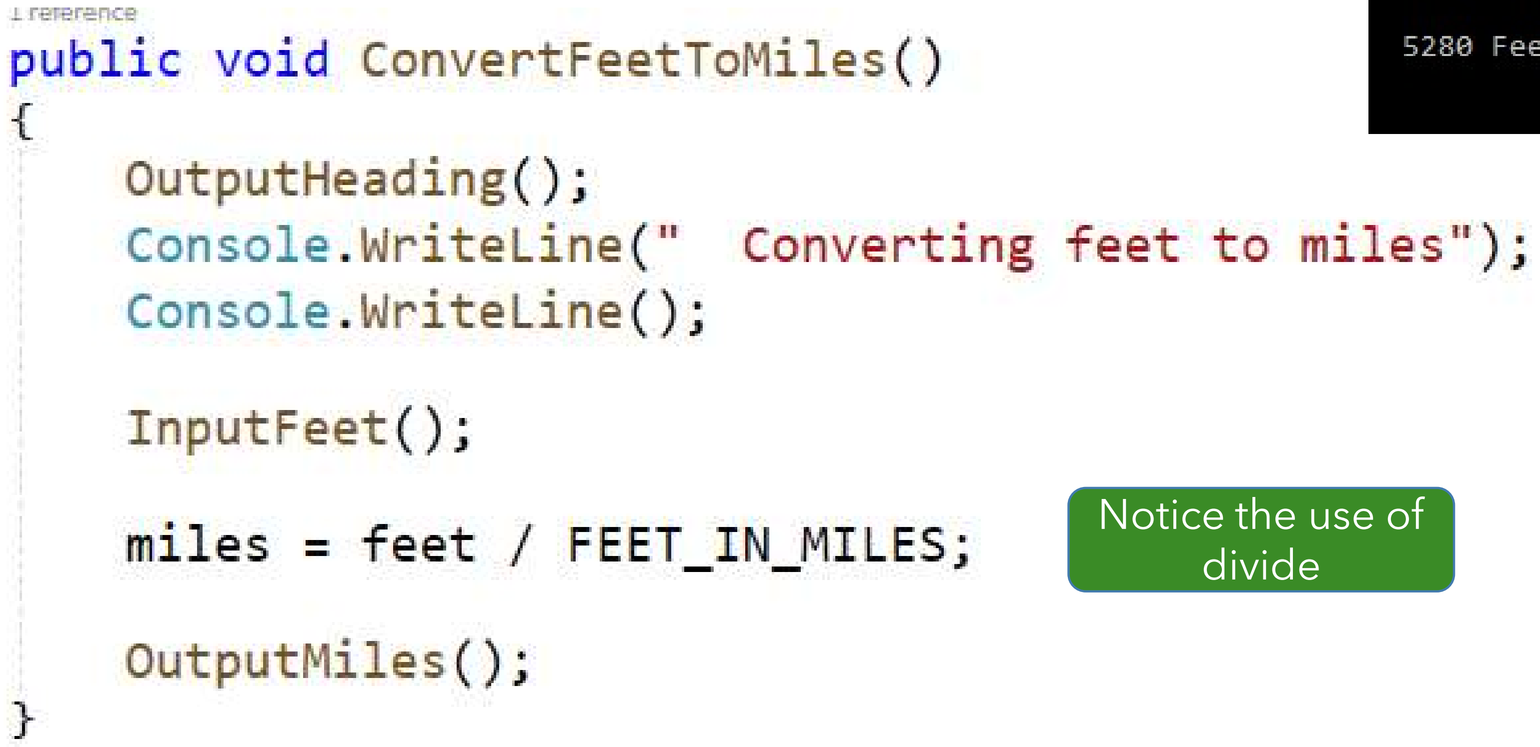
Improve Output??

Improving Output



```
-----------------------------
        Convert Distances
        by Derek Peacock
-----------------------------

Converting feet to miles

Enter the number of feet > 5280

5280 Feet is 1 miles !
```

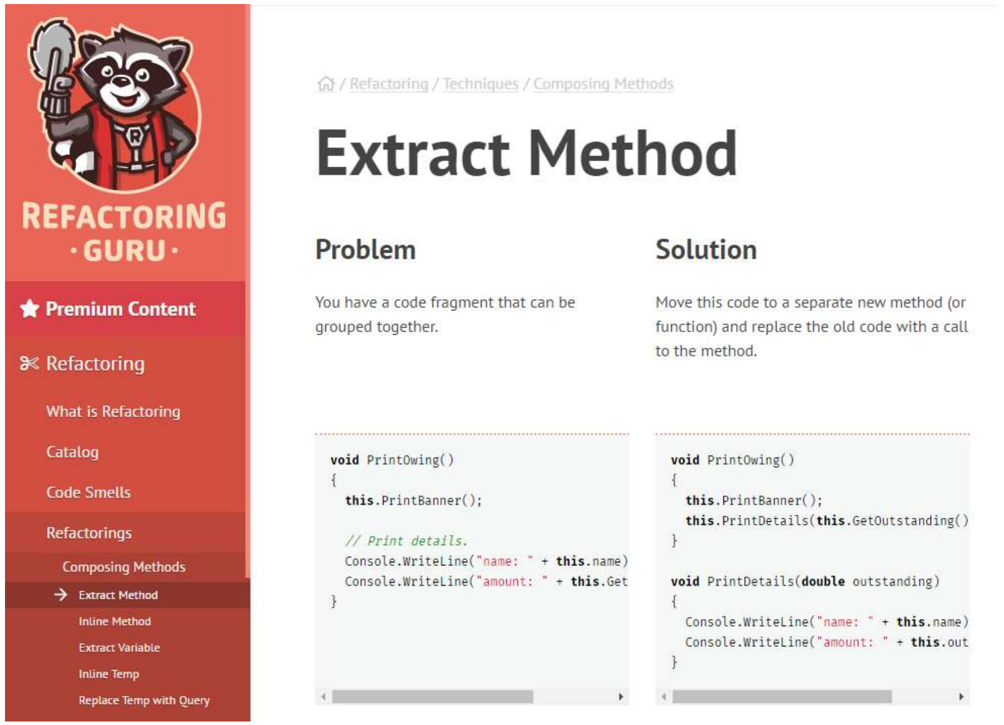
Still not perfect?



```
1 reference
public void ConvertFeetToMiles()
{
    OutputHeading();
    Console.WriteLine("  Converting feet to miles");
    Console.WriteLine();

    InputFeet();

    miles = feet / FEET_IN_MILES;

    OutputMiles();
}
```

Notice the use of divide

Refactor: Extract Method



/ Refactoring / Techniques / Composing Methods

Extract Method

Problem

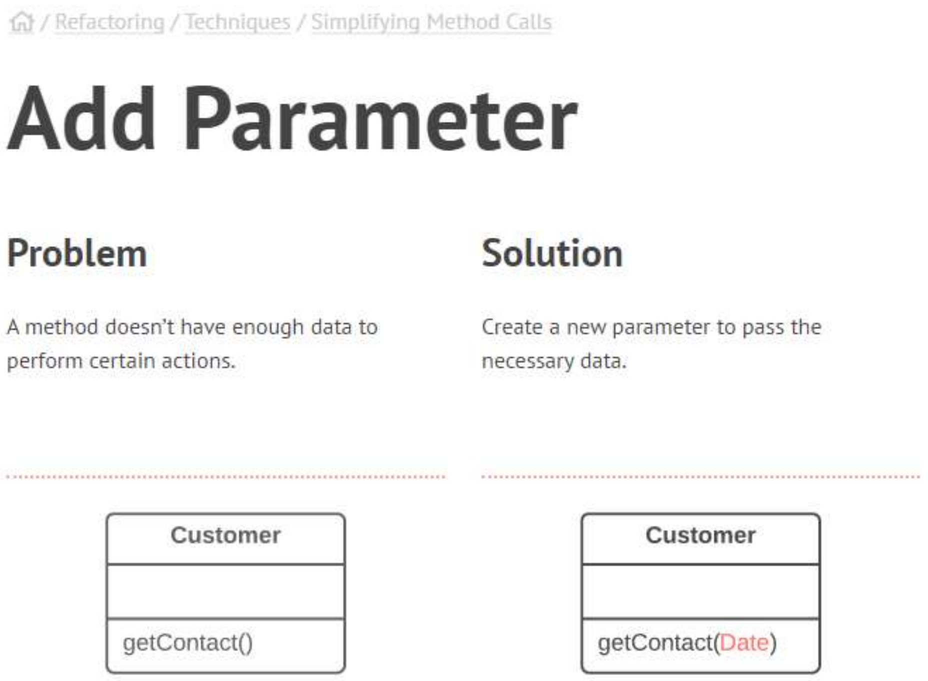
You have a code fragment that can be grouped together.

```
void PrintOwing()
{
  this.PrintBanner();

  // Print details.
  Console.WriteLine("name: " + this.name)
  Console.WriteLine("amount: " + this.Get
}
```

Solution

Move this code to a separate new method (or function) and replace the old code with a call to the method.

```
void PrintOwing()
{
  this.PrintBanner();
  this.PrintDetails(this.GetOutstanding()
}

void PrintDetails(double outstanding)
{
  Console.WriteLine("name: " + this.name)
  Console.WriteLine("amount: " + this.out
}
```



/ Refactoring / Techniques / Simplifying Method Calls

Add Parameter

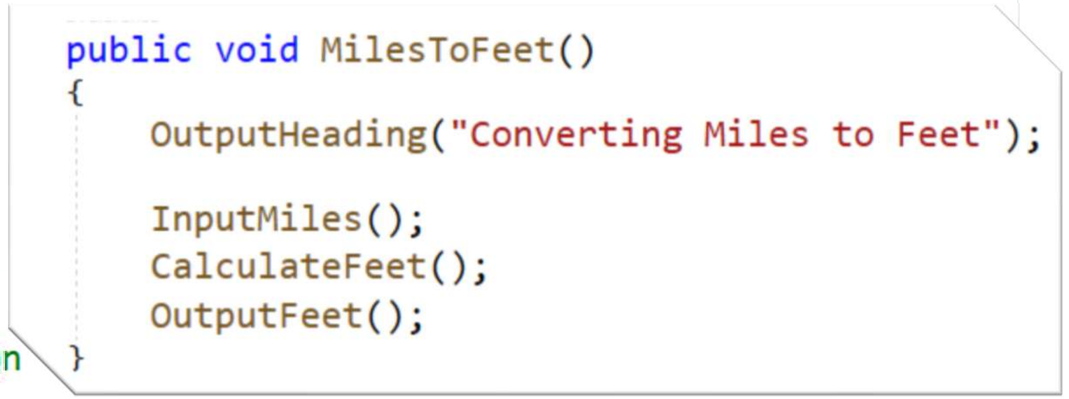
Problem

A method doesn't have enough data to perform certain actions.

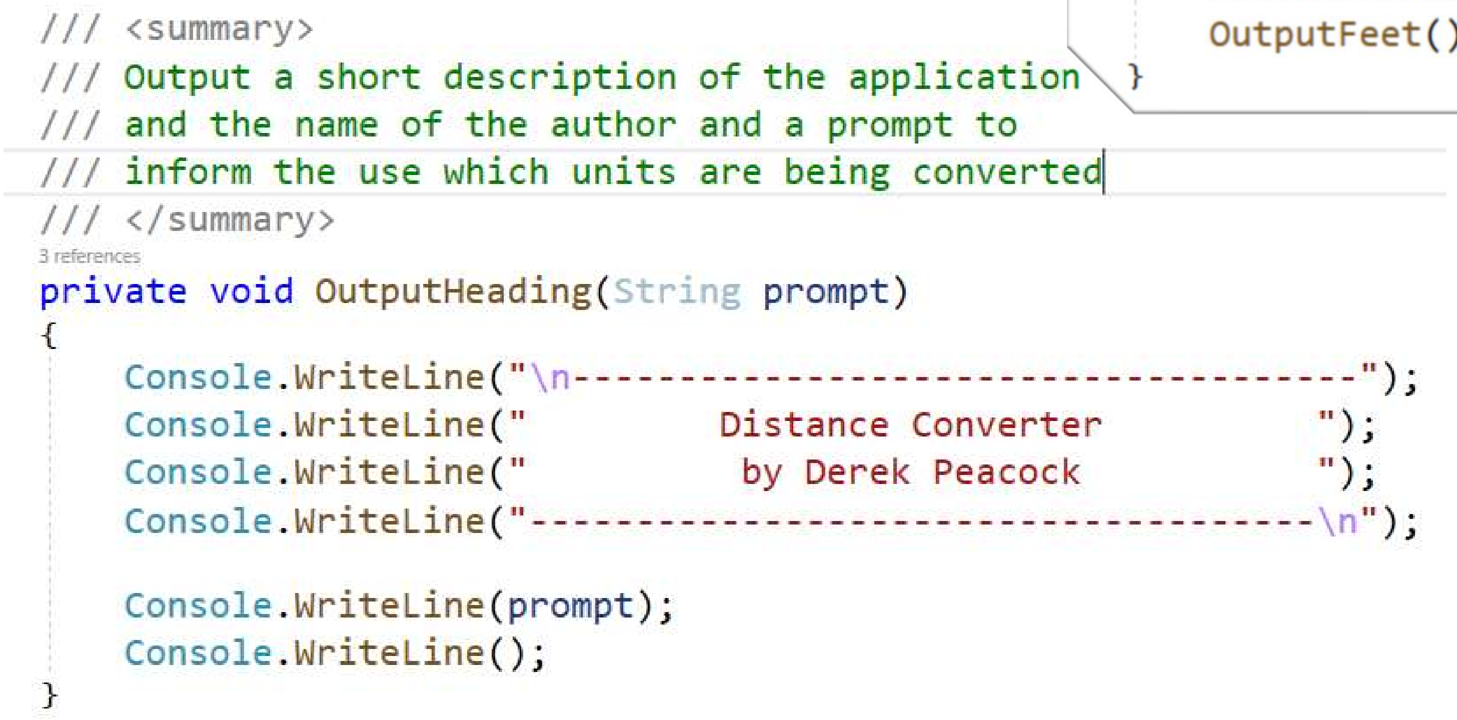
Solution

Create a new parameter to pass the necessary data.

OutputHeading()



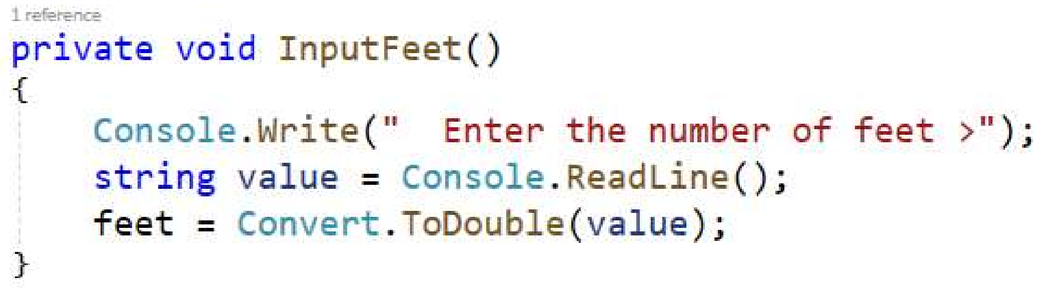
```
public void MilesToFeet()
{
    OutputHeading("Converting Miles to Feet");

    InputMiles();
    CalculateFeet();
    OutputFeet();
}
```



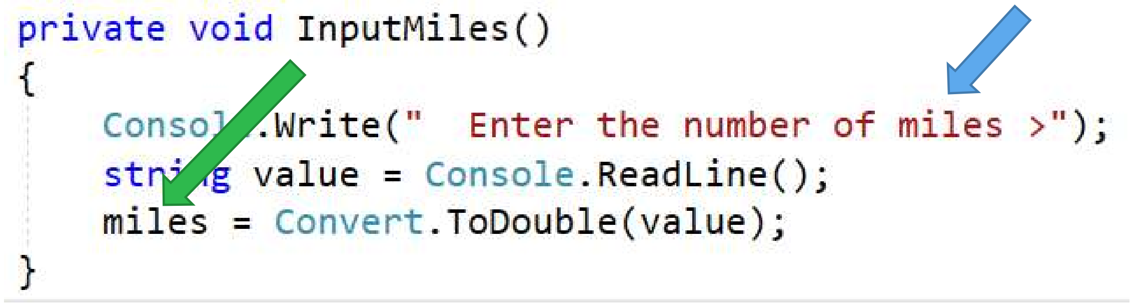
```
/// <summary>
/// Output a short description of the application
/// and the name of the author and a prompt to
/// inform the use which units are being converted
/// </summary>
3 references
private void OutputHeading(String prompt)
{
    Console.WriteLine("\n-----------------------------------");
    Console.WriteLine("         Distance Converter         ");
    Console.WriteLine("          by Derek Peacock          ");
    Console.WriteLine("-----------------------------------\n");

    Console.WriteLine(prompt);
    Console.WriteLine();
}
```

1. Extract as method
2. Add Parameter

More Duplication



```
1 reference
private void InputFeet()
{
    Console.Write("  Enter the number of feet >");
    string value = Console.ReadLine();
    feet = Convert.ToDouble(value);
}
```

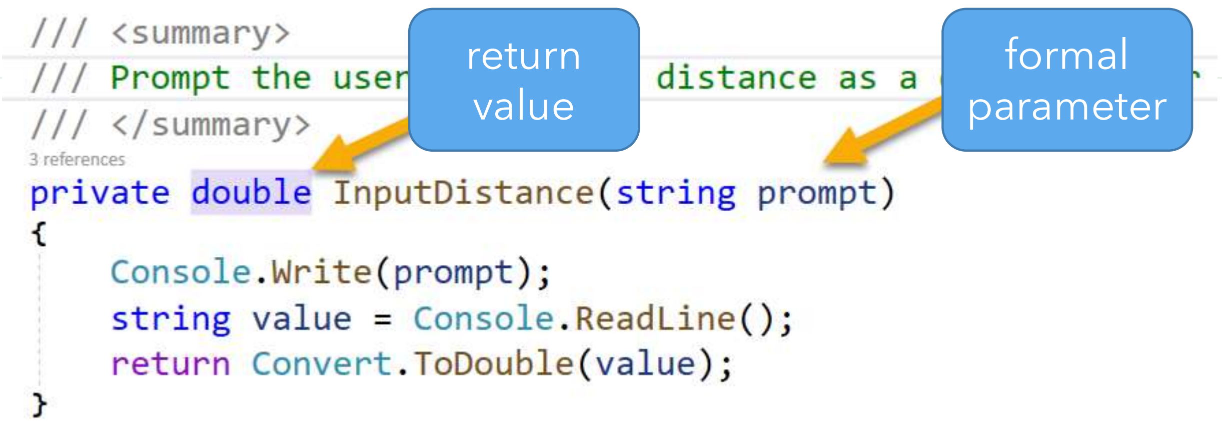


```
private void InputMiles()
{
    Console.Write("  Enter the number of miles >");
    string value = Console.ReadLine();
    miles = Convert.ToDouble(value);
}
```

What is different?

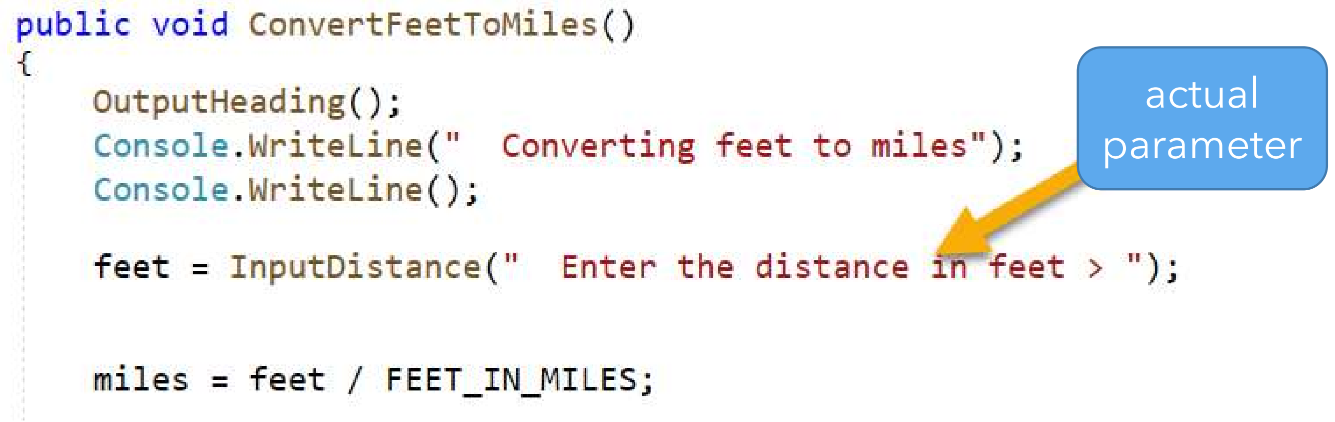
1. The prompt to the user
2. The value read and stored

Method Return Values



```
/// <summary>
/// Prompt the user          distance as a
/// </summary>
3 references
private double InputDistance(string prompt)
{
    Console.Write(prompt);
    string value = Console.ReadLine();
    return Convert.ToDouble(value);
}
```

- Fewer lines of Code (-9)
- Less complex (9 -> 8)
- Easier to maintain
- Performs the same function

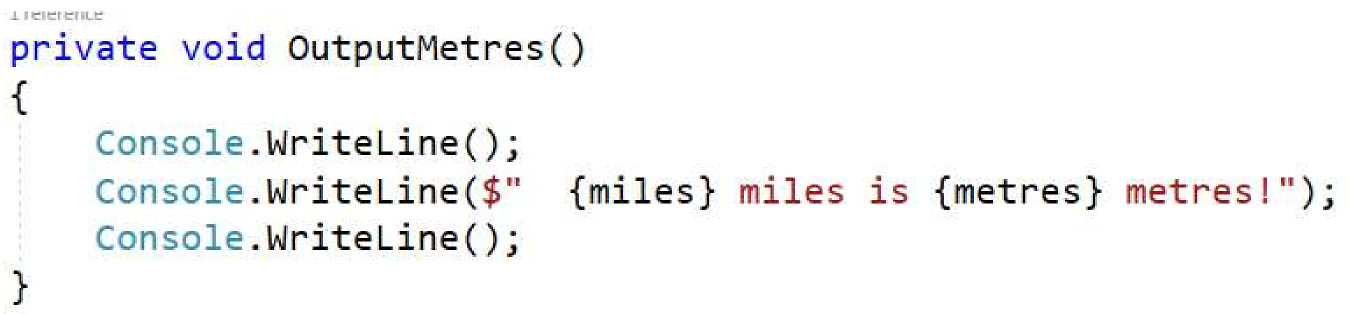


```
public void ConvertFeetToMiles()
{
    OutputHeading();
    Console.WriteLine("  Converting feet to miles");
    Console.WriteLine();

    feet = InputDistance("  Enter the distance in feet > ");

    miles = feet / FEET_IN_MILES;
```

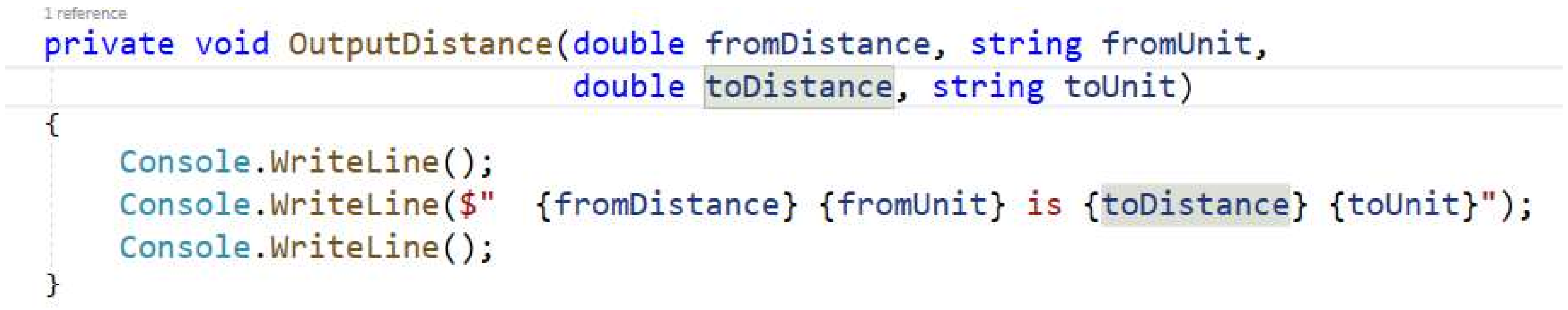
Generalising Output Methods

- Fewer lines of Code (35 -> 26)
- Less complex (9 -> 6)
- Easier to maintain
- Performs the same function

```
OutputDistance(miles, "miles", feet, "feet");
```

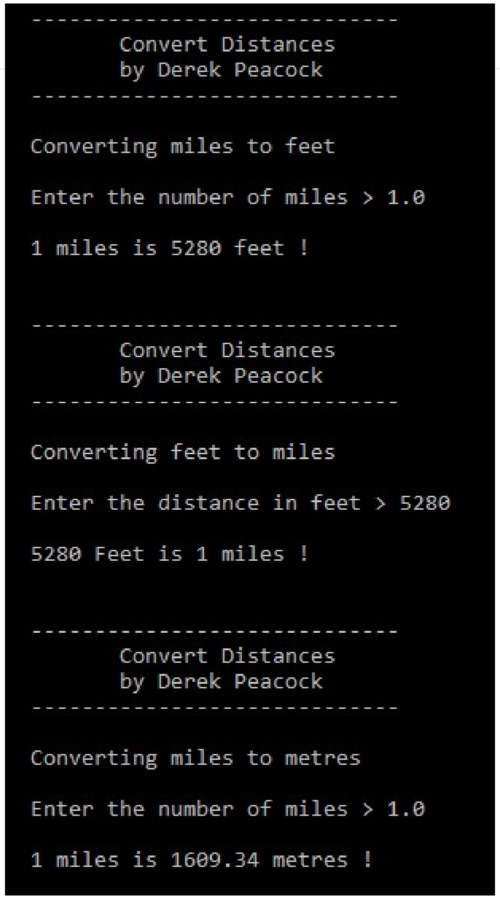


```
1 reference
private void OutputMetres()
{
    Console.WriteLine();
    Console.WriteLine($"  {miles} miles is {metres} metres!");
    Console.WriteLine();
}
```



```
1 reference
private void OutputDistance(double fromDistance, string fromUnit,
                            double toDistance, string toUnit)
{
    Console.WriteLine();
    Console.WriteLine($"  {fromDistance} {fromUnit} is {toDistance} {toUnit}");
    Console.WriteLine();
}
```

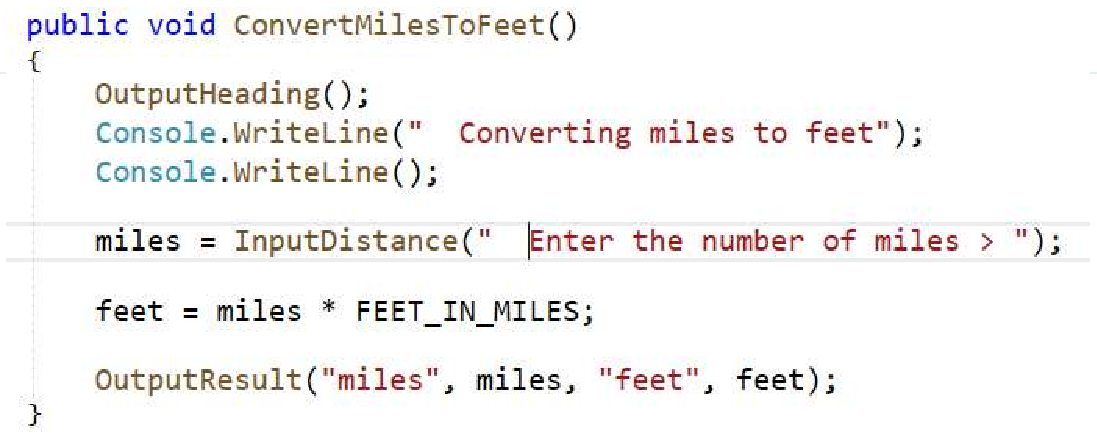
Test the three methods



```
-----------------------------
       Convert Distances
       by Derek Peacock
-----------------------------

Converting miles to feet

Enter the number of miles > 1.0

1 miles is 5280 feet !


-----------------------------
       Convert Distances
       by Derek Peacock
-----------------------------

Converting feet to miles

Enter the distance in feet > 5280

5280 Feet is 1 miles !


-----------------------------
       Convert Distances
       by Derek Peacock
-----------------------------

Converting miles to metres

Enter the number of miles > 1.0

1 miles is 1609.34 metres !
```

- Each time you refactor you need to re-test all features

Generalising Conversion Methods

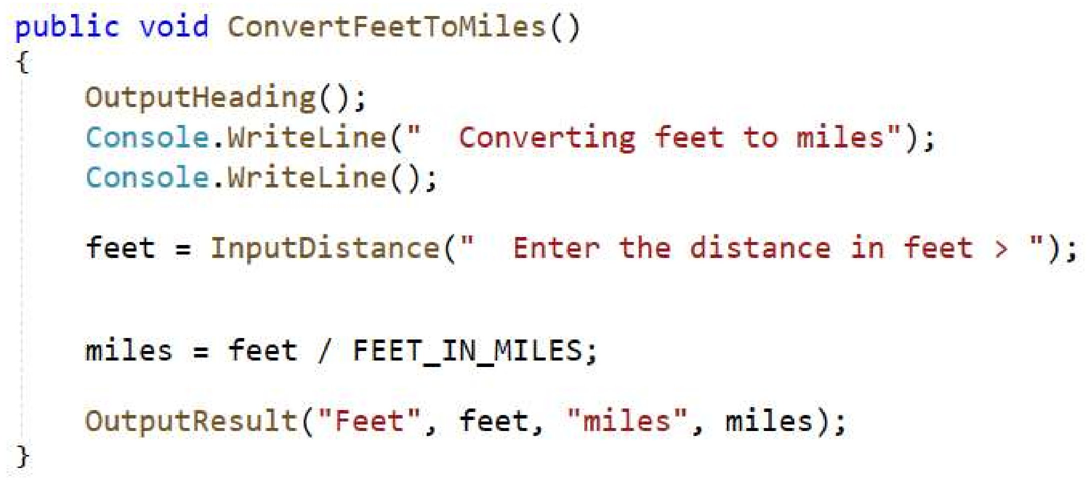


```
public void ConvertMilesToFeet()
{
    OutputHeading();
    Console.WriteLine("  Converting miles to feet");
    Console.WriteLine();

    miles = InputDistance("  Enter the number of miles > ");

    feet = miles * FEET_IN_MILES;

    OutputResult("miles", miles, "feet", feet);
}
```

Imperial Units
Inch, Feet,
Yard, Mile

Metric Units
Centimetre
Metre,
Kilometre

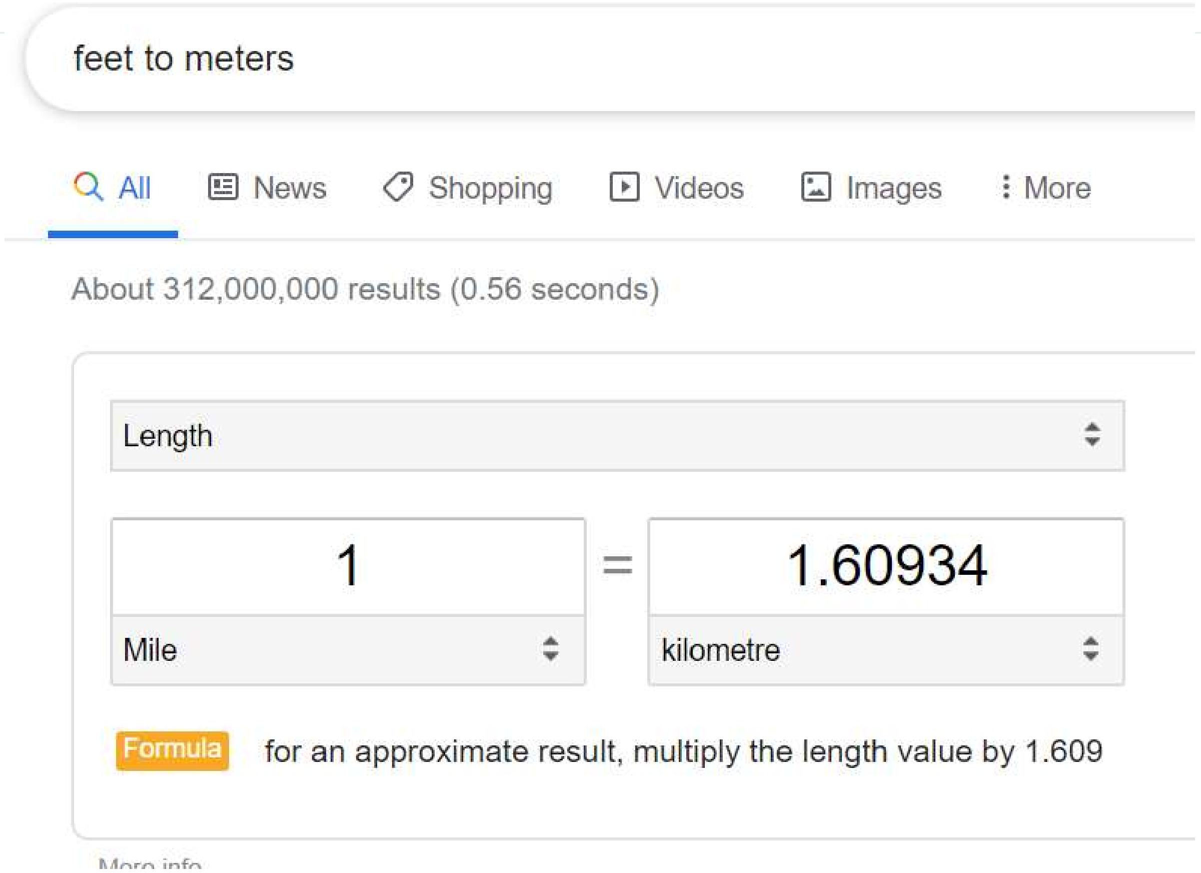
No Possible
Conversions
7 x 7 – 7 = 42



```
public void ConvertFeetToMiles()
{
    OutputHeading();
    Console.WriteLine("  Converting feet to miles");
    Console.WriteLine();

    feet = InputDistance("  Enter the distance in feet > ");


    miles = feet / FEET_IN_MILES;

    OutputResult("Feet", feet, "miles", miles);
}
```

Needs a
new
design!

Real Example (Google)



feet to meters

All News Shopping Videos Images More

About 312,000,000 results (0.56 seconds)

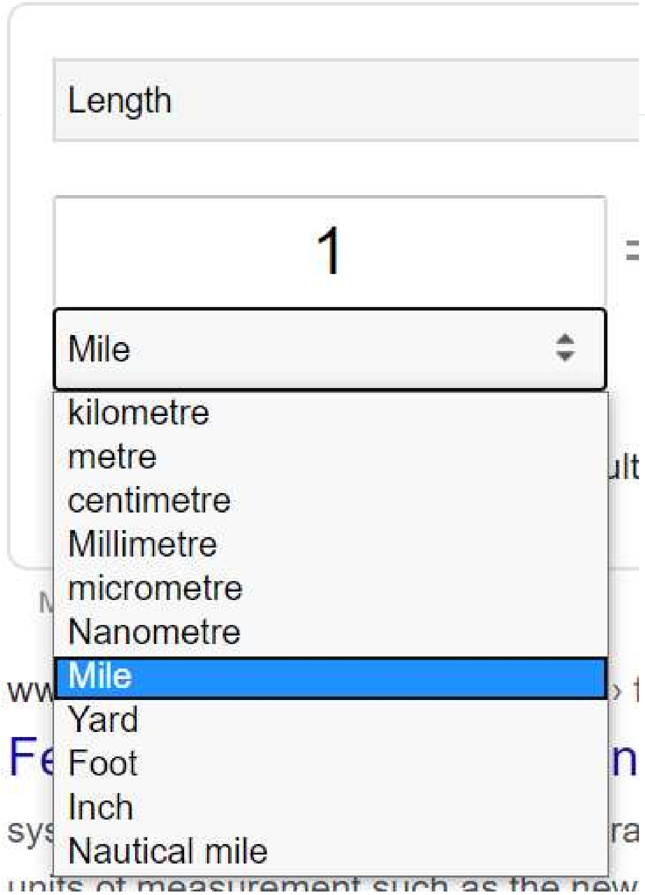
Length

1 = 1.60934

Mile kilometre

Formula for an approximate result, multiply the length value by 1.609

More info



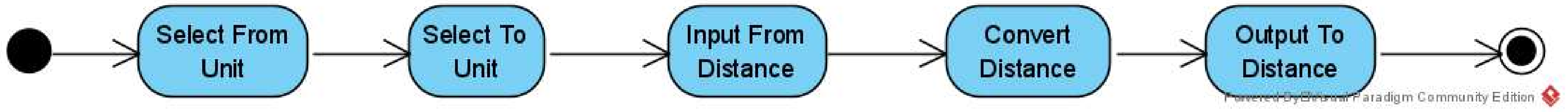
Length

1

Mile

kilometre
metre
centimetre
Millimetre
micrometre
Nanometre
Mile
Yard
Foot
Inch
Nautical mile

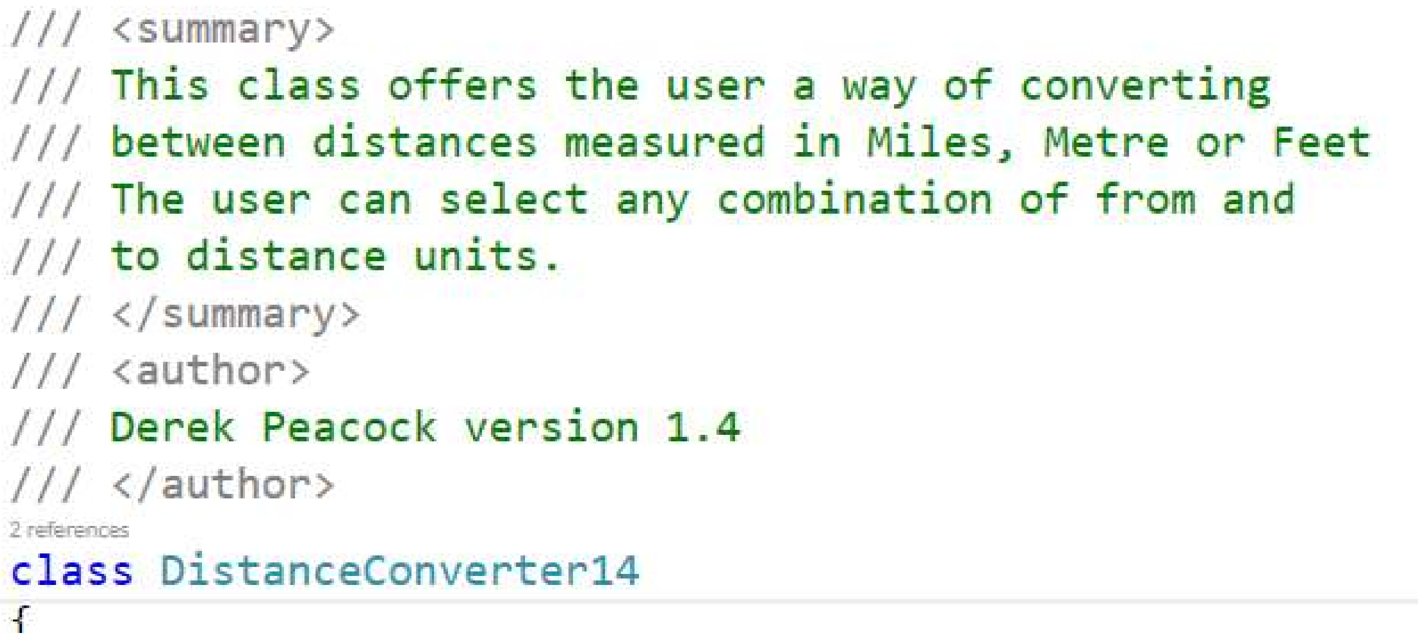
New Design for Distance Converter (v1.4)



This involves offering the user a list of distance units for them to **select** which one they want to convert from and which one the want to convert to.

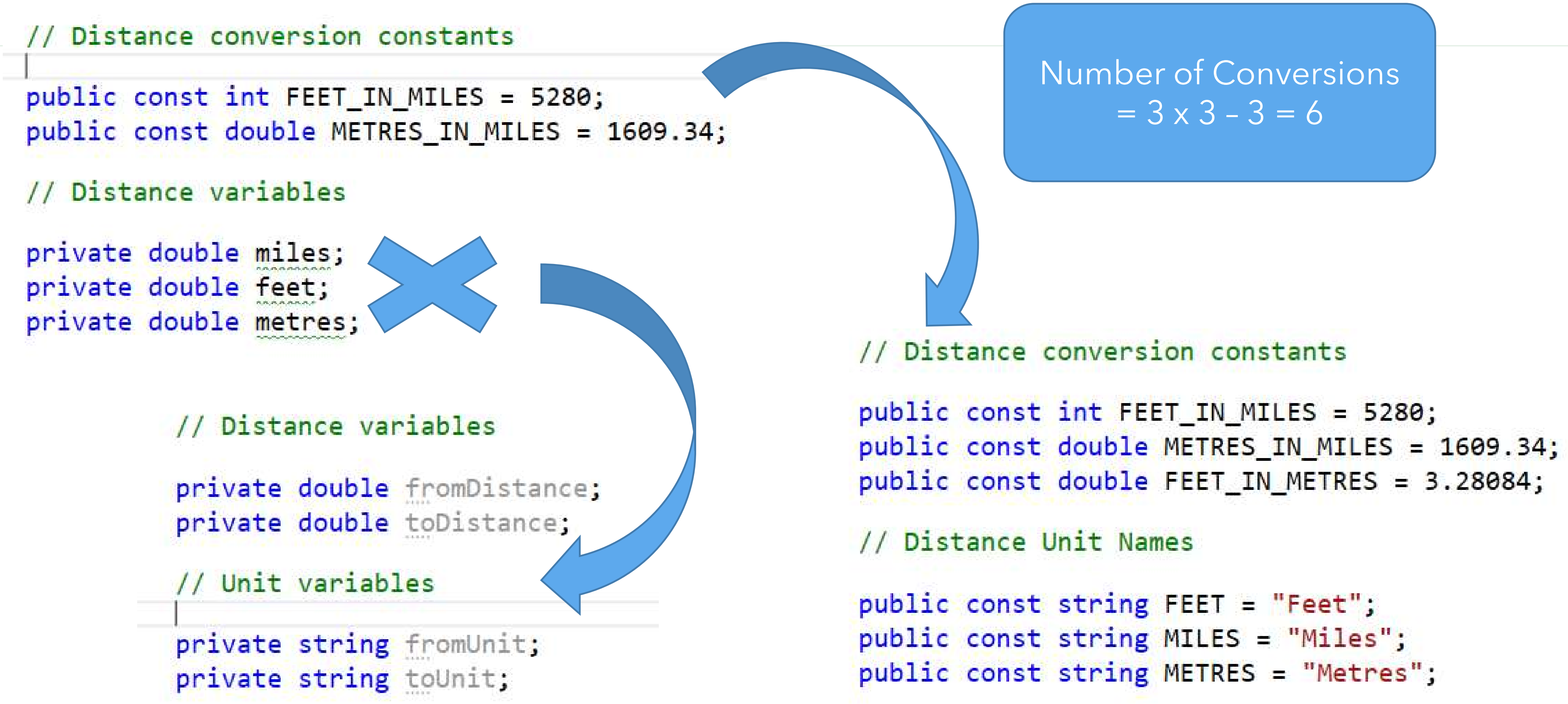
This is a major program **refactor** and an **enhancement**!

Class Comment version 1.4



```
/// <summary>
/// This class offers the user a way of converting
/// between distances measured in Miles, Metre or Feet
/// The user can select any combination of from and
/// to distance units.
/// </summary>
/// <author>
/// Derek Peacock version 1.4
/// </author>
2 references
class DistanceConverter14
{
```

Generalise Variables



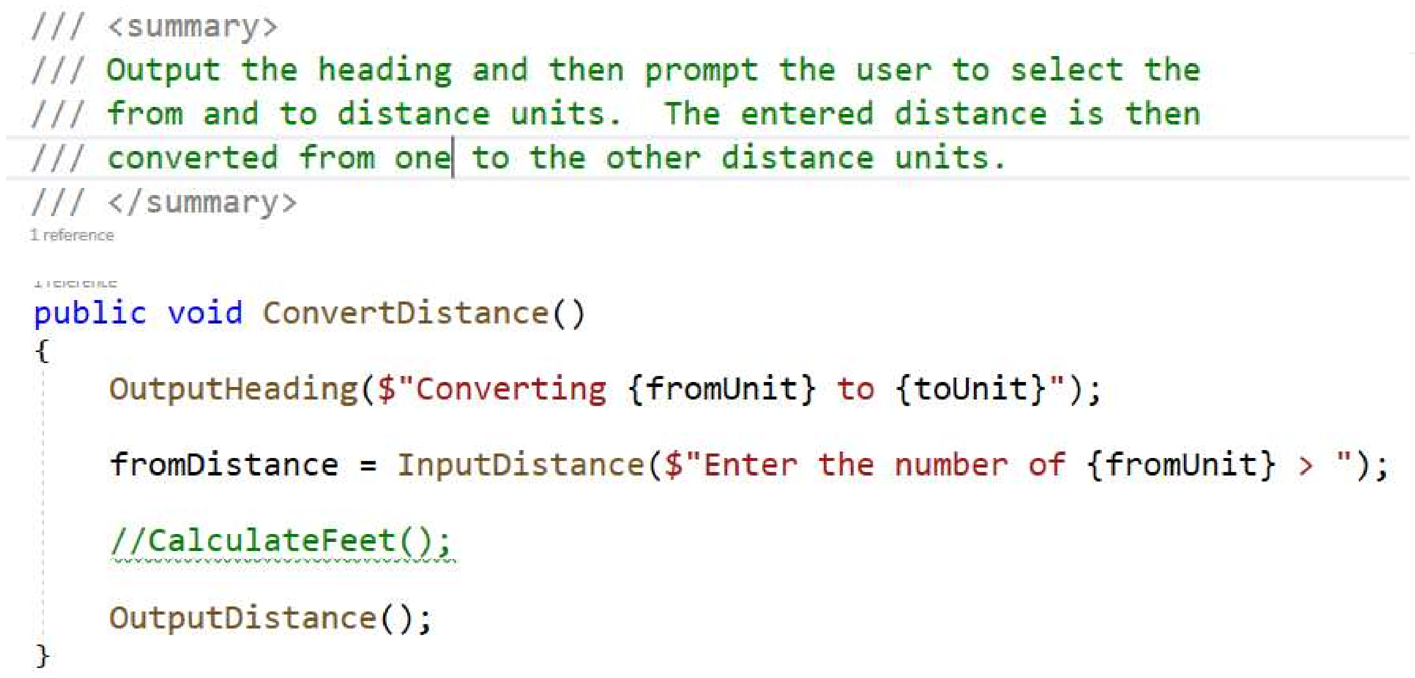
```
// Distance conversion constants

public const int FEET_IN_MILES = 5280;
public const double METRES_IN_MILES = 1609.34;

// Distance variables

private double miles;
private double feet;
private double metres;
```

```
    // Distance variables

    private double fromDistance;
    private double toDistance;

    // Unit variables

    private string fromUnit;
    private string toUnit;
```

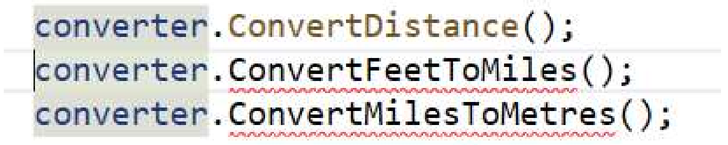
```
// Distance conversion constants

public const int FEET_IN_MILES = 5280;
public const double METRES_IN_MILES = 1609.34;
public const double FEET_IN_METRES = 3.28084;

// Distance Unit Names

public const string FEET = "Feet";
public const string MILES = "Miles";
public const string METRES = "Metres";
```

Number of Conversions
= 3 x 3 - 3 = 6

ConvertDistance()



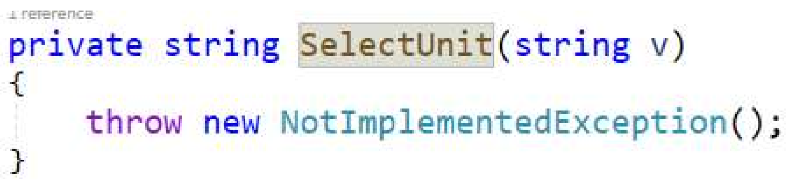
```
/// <summary>
/// Output the heading and then prompt the user to select the
/// from and to distance units.  The entered distance is then
/// converted from one to the other distance units.
/// </summary>
1 reference
public void ConvertDistance()
{
    OutputHeading($"Converting {fromUnit} to {toUnit}");

    fromDistance = InputDistance($"Enter the number of {fromUnit} > ");

    //CalculateFeet();

    OutputDistance();
}
```



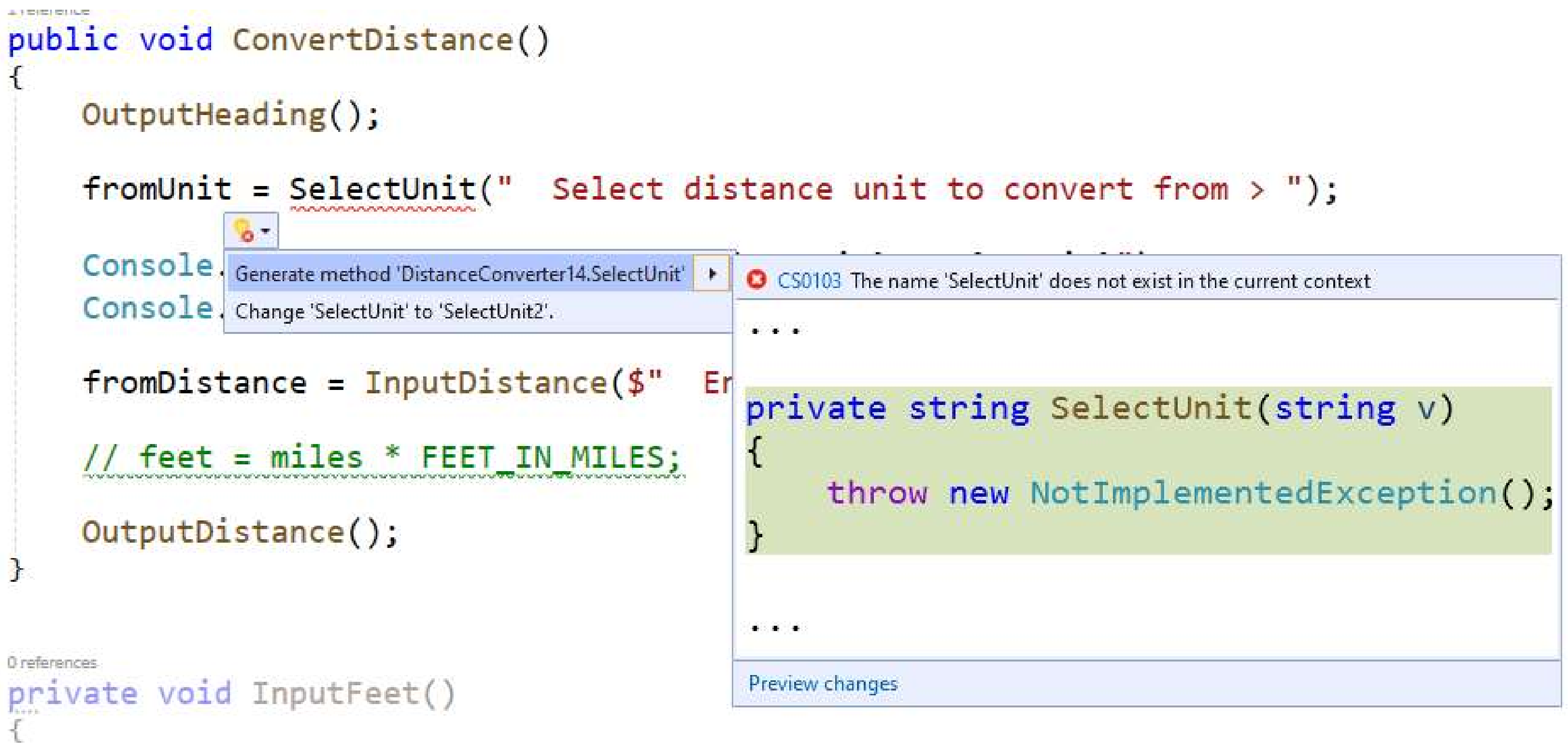
```
converter.ConvertDistance();
converter.ConvertFeetToMiles();
converter.ConvertMilesToMetres();
```

- Delete two convert methods
- Rename last convert method
- Use new variables for distance and units
- Comment out conversion
- Remove parameters from OutputDistance
- Remove convert calls from Program class

Add SelectUnit()



```
1 reference
private string SelectUnit(string v)
{
    throw new NotImplementedException();
}
```



```
1 reference
public void ConvertDistance()
{
    OutputHeading();

    fromUnit = SelectUnit("  Select distance unit to convert from > ");

    Console.
    Console.

    fromDistance = InputDistance($"  Er

    // feet = miles * FEET_IN_MILES;

    OutputDistance();
}

0 references
private void InputFeet()
{
```

Generate method 'DistanceConverter14.SelectUnit'

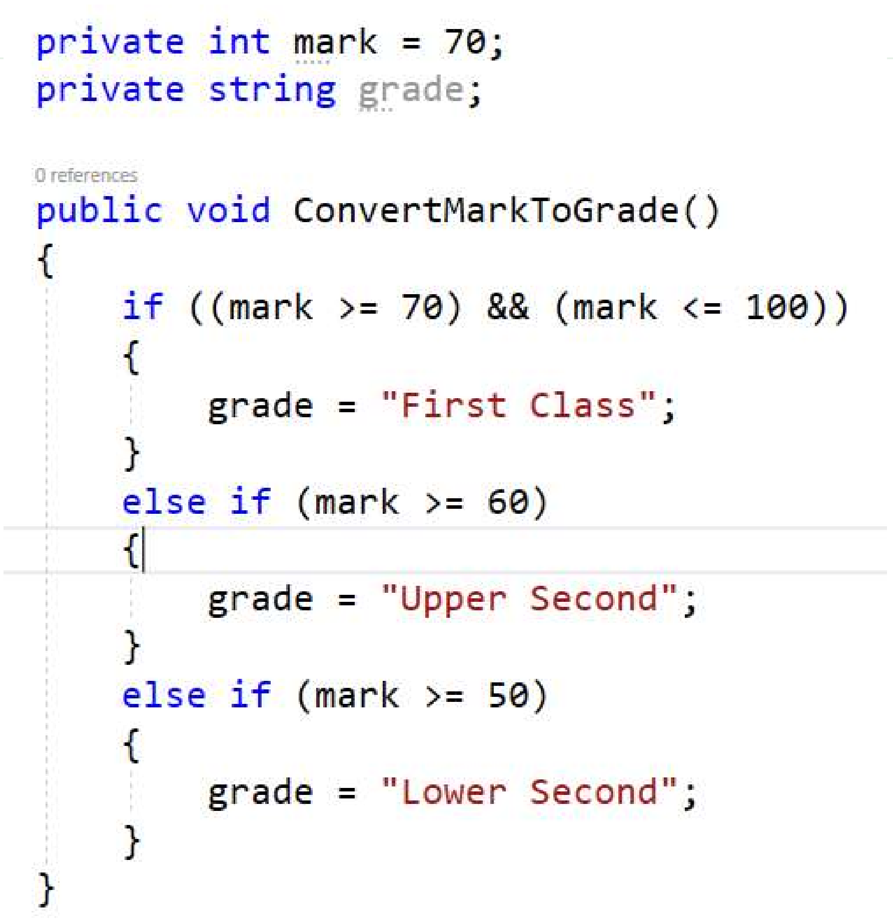
Change 'SelectUnit' to 'SelectUnit2'.

CS0103 The name 'SelectUnit' does not exist in the current context

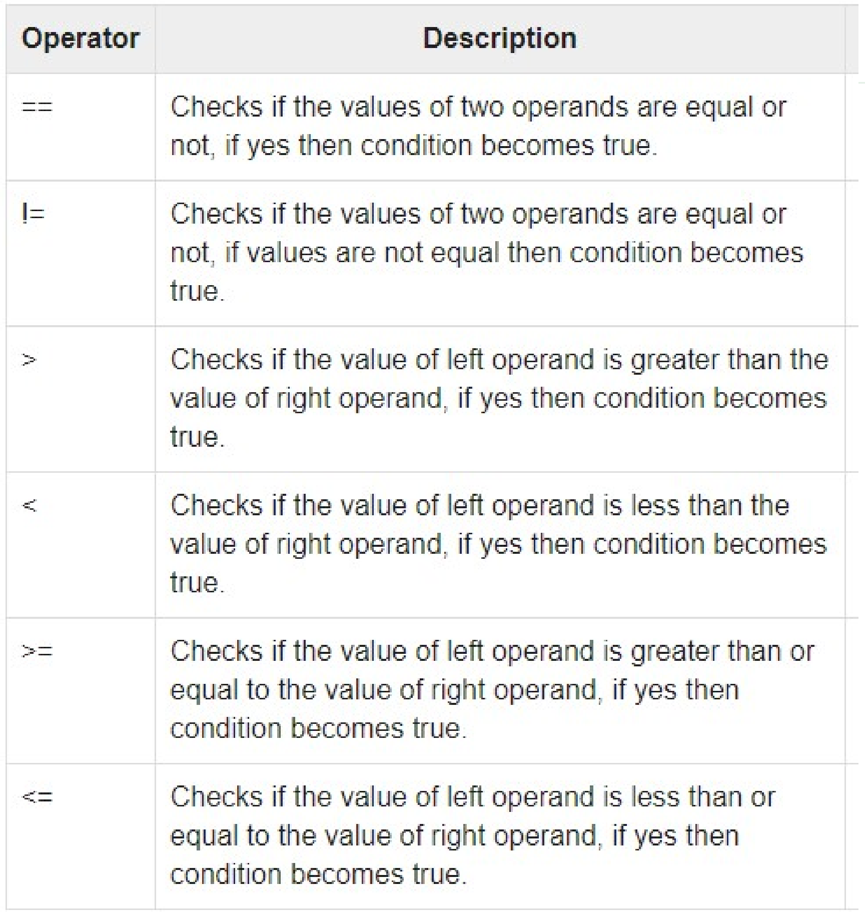
```
...

private string SelectUnit(string v)
{
    throw new NotImplementedException();
}

...
```

Preview changes

C# Selection & Relational Operators

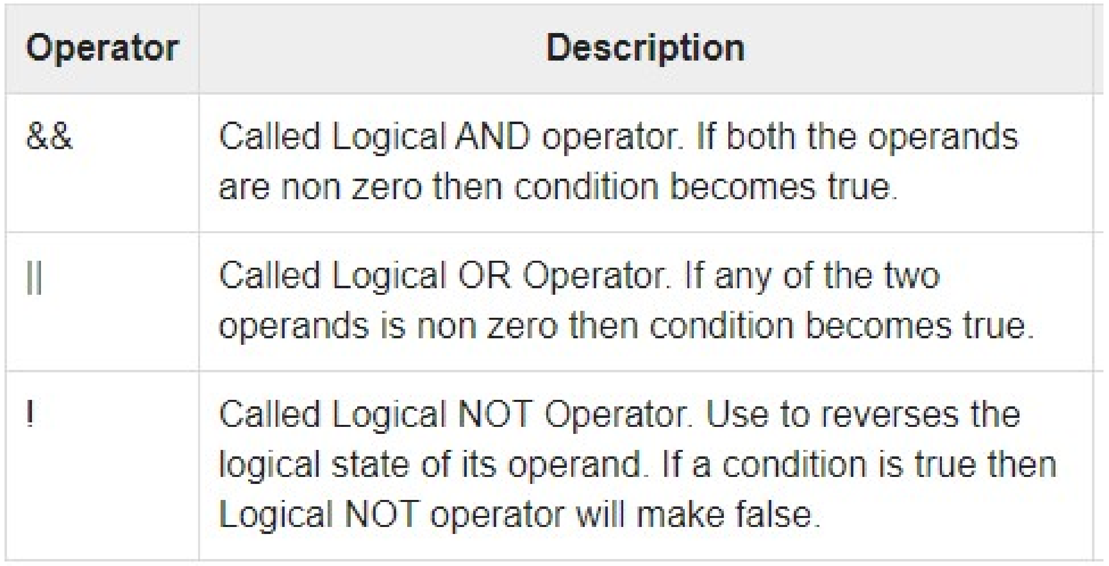


```
private int mark = 70;
private string grade;

0 references
public void ConvertMarkToGrade()
{
    if ((mark >= 70) && (mark <= 100))
    {
        grade = "First Class";
    }
    else if (mark >= 60)
    {
        grade = "Upper Second";
    }
    else if (mark >= 50)
    {
        grade = "Lower Second";
    }
}
```



Operator	Description
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

Logical Operators



Operator	Description
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.
||	Called Logical OR Operator. If any of the two operands is non zero then condition becomes true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

(3 > 1) && (5 < 3) is **false**

(3 > 1) || (5 < 3) is **true**

!(3 > 1) is **false**

!(5 < 3) is **true**

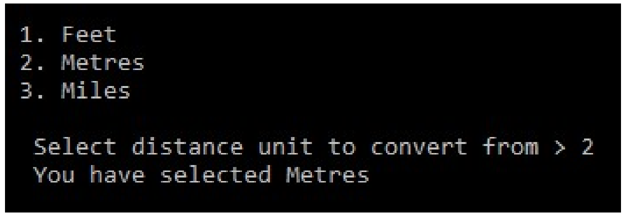
https://www.tutorialspoint.com/csharp/csharp_operators.htm

SelectUnit() Method

```
/// <summary>
/// Display a menu of distance units and then prom
/// user to select one and return it.
/// </summary>
2 references
private string SelectUnit(string prompt)
{
    Console.WriteLine();
    Console.WriteLine($" 1. {FEET}");
    Console.WriteLine($" 2. {METRES}");
    Console.WriteLine($" 3. {MILES}");
    Console.WriteLine();

    Console.Write(prompt);
    string choice = Console.ReadLine();
```

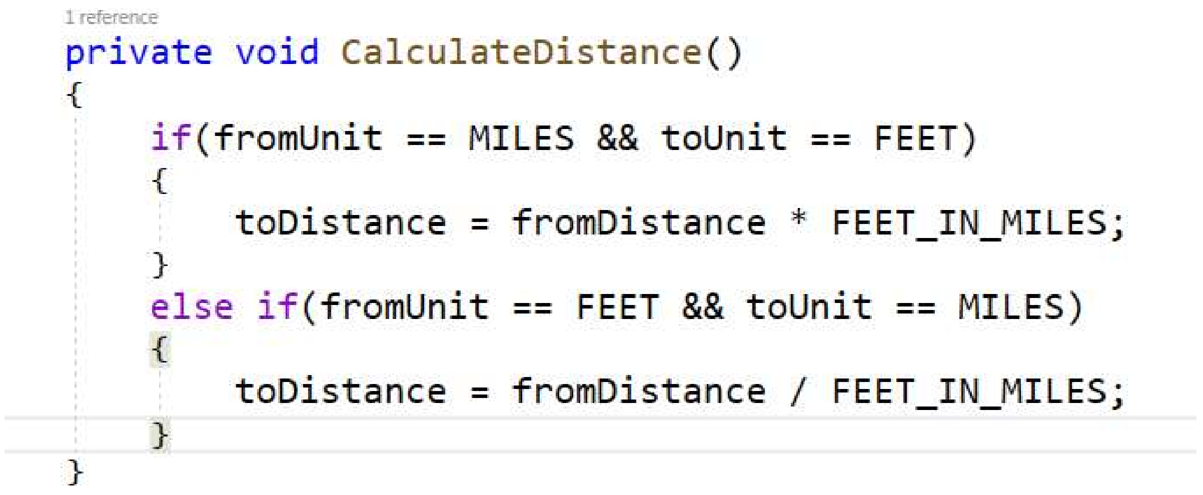
```
if (choice == "1")
{
    unit = FEET;
}
else if (choice == "2")
{
    unit = METRES;
}
else if (choice == "3")
{
    unit = MILES;
}

Console.WriteLine($"  You have selected {unit}");
Console.WriteLine();

return unit;
```

What does this not check for?

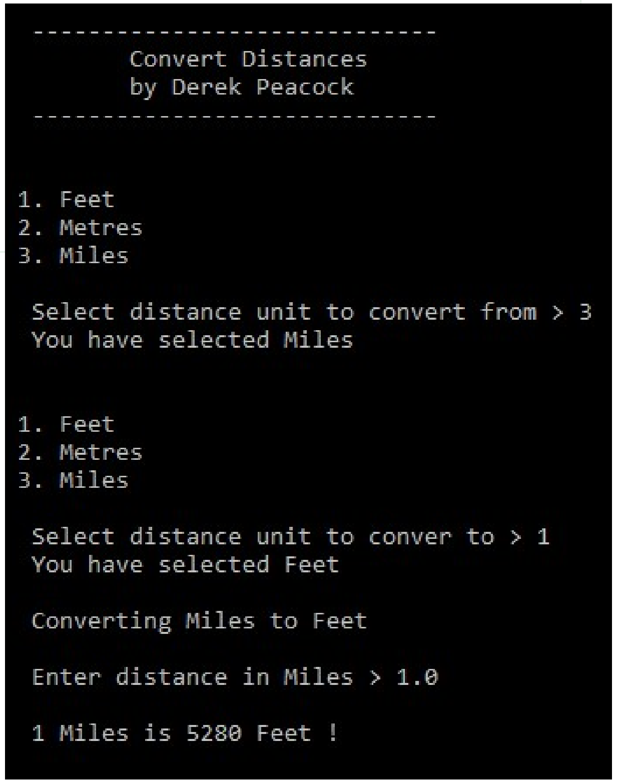


```
1. Feet
2. Metres
3. Miles

 Select distance unit to convert from > 2
 You have selected Metres
```

ConvertDistance() Method



```
1 reference
private void CalculateDistance()
{
    if(fromUnit == MILES && toUnit == FEET)
    {
        toDistance = fromDistance * FEET_IN_MILES;
    }
    else if(fromUnit == FEET && toUnit == MILES)
    {
        toDistance = fromDistance / FEET_IN_MILES;
    }
}
```



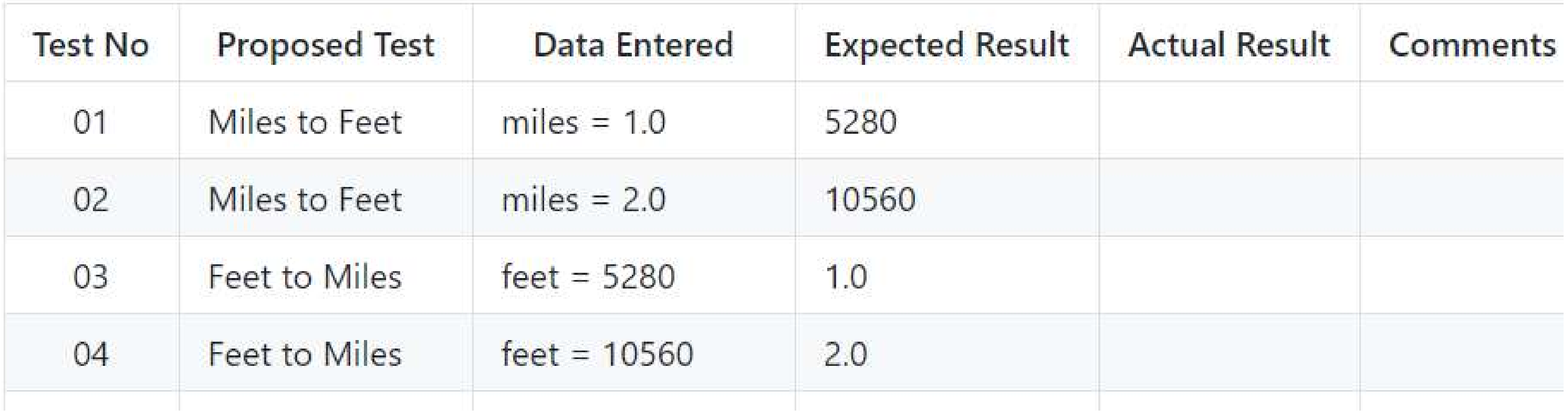
```
 -----------------------------
        Convert Distances
        by Derek Peacock
 -----------------------------


1. Feet
2. Metres
3. Miles

 Select distance unit to convert from > 3
 You have selected Miles


1. Feet
2. Metres
3. Miles

 Select distance unit to conver to > 1
 You have selected Feet

 Converting Miles to Feet

 Enter distance in Miles > 1.0

 1 Miles is 5280 Feet !
```

This single method can now complete all 6 conversions and offers the user a choice

Testing App01: Valid Data

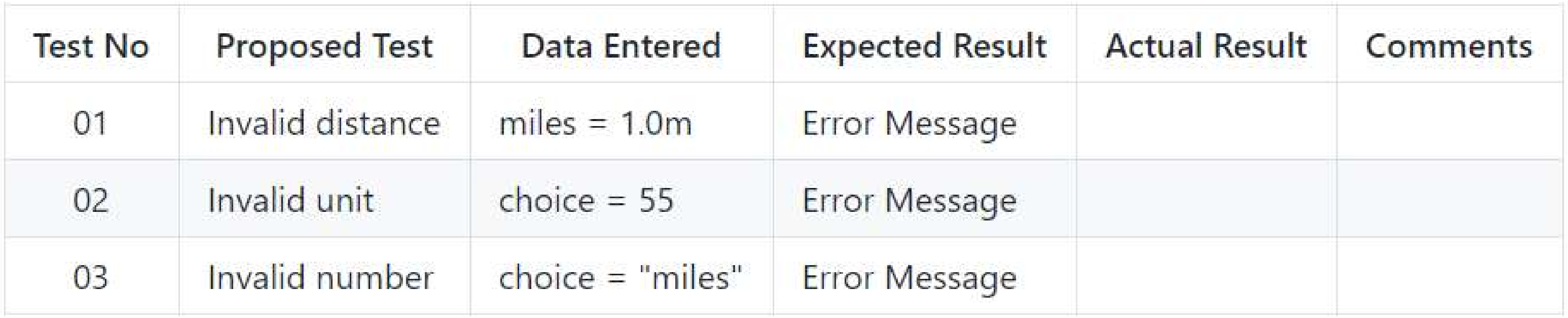
Black Box Testing

Needs 12 tests



Test No	Proposed Test	Data Entered	Expected Result	Actual Result	Comments
01	Miles to Feet	miles = 1.0	5280		
02	Miles to Feet	miles = 2.0	10560		
03	Feet to Miles	feet = 5280	1.0		
04	Feet to Miles	feet = 10560	2.0		

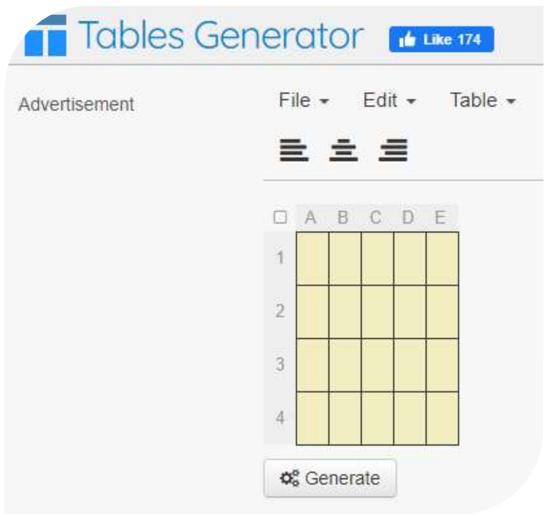
Testing App01: Invalid Data

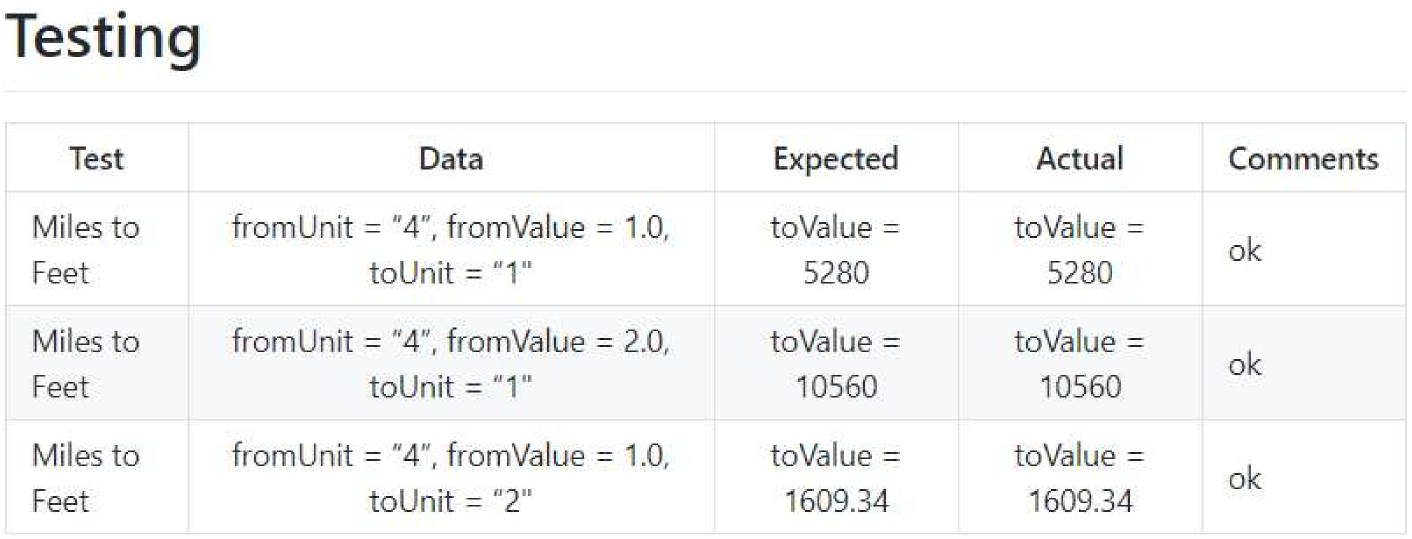


Test No	Proposed Test	Data Entered	Expected Result	Actual Result	Comments
01	Invalid distance	miles = 1.0m	Error Message		
02	Invalid unit	choice = 55	Error Message		
03	Invalid number	choice = "miles"	Error Message		

Markdown Tables

```
# Testing
| Test  | Data   | Expected   | Actual  |   Comments |
|---|:-:|:-:|:-:|---|
| Miles to Feet  | fromUnit = “4”, fromValue = 1.0, toUnit = “1"  | toValue = 5280  | toValue = 5280  |ok   |
| Miles to Feet  | fromUnit = “4”, fromValue = 2.0, toUnit = “1"  | toValue = 10560  | toValue = 10560  |ok   |
| Miles to Feet  | fromUnit = “4”, fromValue = 1.0, toUnit = “2"  | toValue = 1609.34  | toValue = 1609.34  |ok   |
```





Testing

Test	Data	Expected	Actual	Comments
Miles to Feet	fromUnit = “4”, fromValue = 1.0, toUnit = “1"	toValue = 5280	toValue = 5280	ok
Miles to Feet	fromUnit = “4”, fromValue = 2.0, toUnit = “1"	toValue = 10560	toValue = 10560	ok
Miles to Feet	fromUnit = “4”, fromValue = 1.0, toUnit = “2"	toValue = 1609.34	toValue = 1609.34	ok

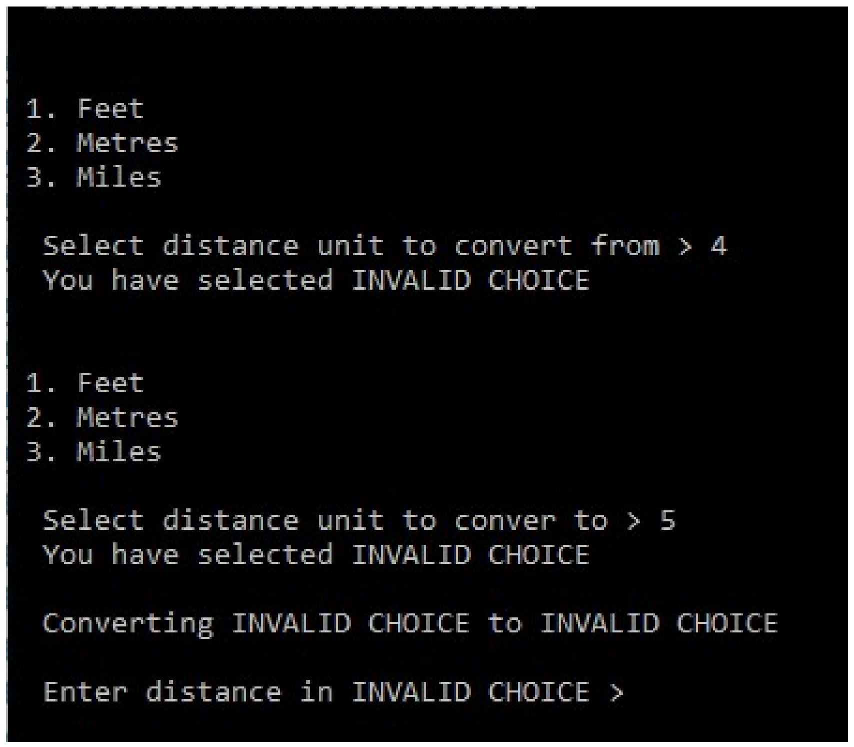
Adding Data Validation

```
string unit = "INVALID CHOICE";

if (choice == "1")
{
    unit = FEET;
}
else if (choice == "2")
{
    unit = METRES;
}
else if (choice == "3")
{
    unit = MILES;
}

Console.WriteLine($"  You have selected {unit}");
Console.WriteLine();

return unit;
```

Alternative selection method

```
switch (choice)
{
    case "1": unit = FEET; break;
    case "2": unit = METRES; break;
    case "3": unit = MILES; break;

    default: unit = "INVALID CHOICE"; break;
}
```

Problems of Invalid Data

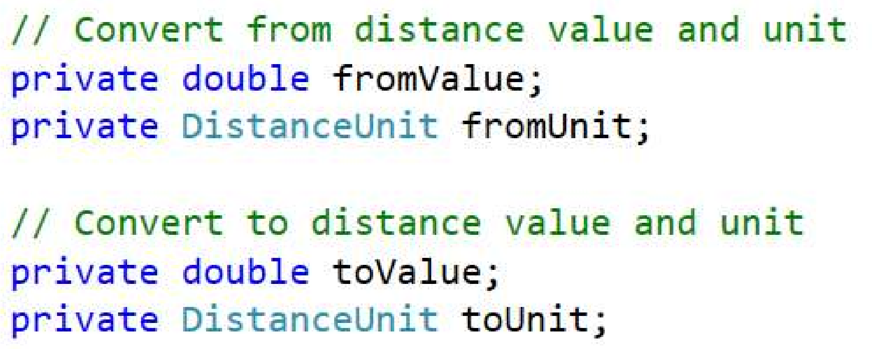


```
1. Feet
2. Metres
3. Miles

 Select distance unit to convert from > 4
 You have selected INVALID CHOICE


1. Feet
2. Metres
3. Miles

 Select distance unit to conver to > 5
 You have selected INVALID CHOICE

 Converting INVALID CHOICE to INVALID CHOICE

 Enter distance in INVALID CHOICE >
```

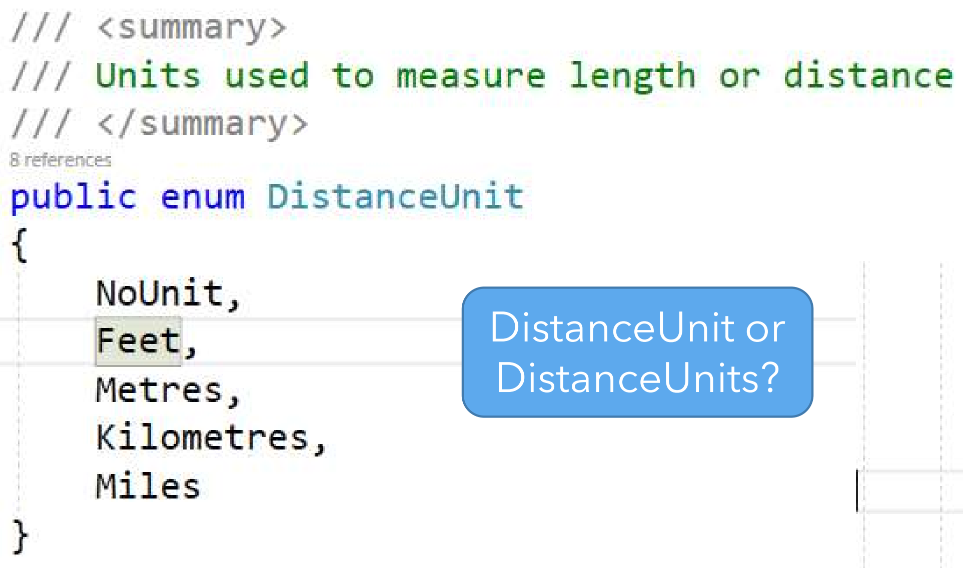
How should this be improved for the user?

Needs the use of repetition introduced later!

Using Enumerations

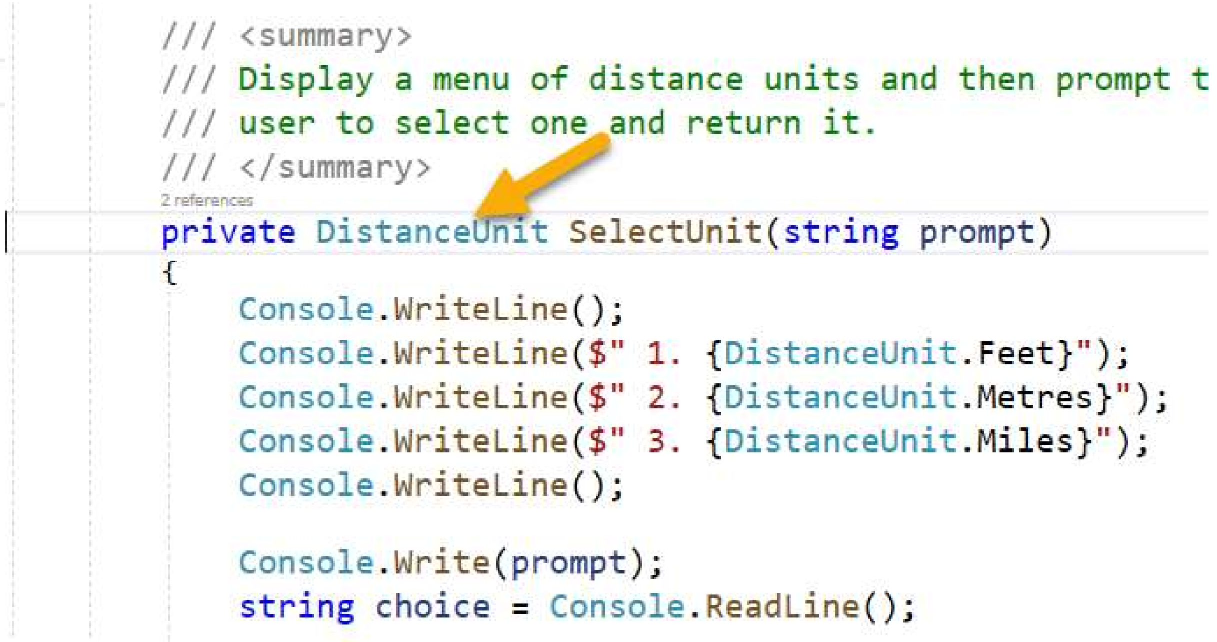


```
// Convert from distance value and unit
private double fromValue;
private DistanceUnit fromUnit;

// Convert to distance value and unit
private double toValue;
private DistanceUnit toUnit;
```



```
/// <summary>
/// Units used to measure length or distance
/// </summary>
8 references
public enum DistanceUnit
{
    NoUnit,
    Feet,
    Metres,
    Kilometres,
    Miles
}
```

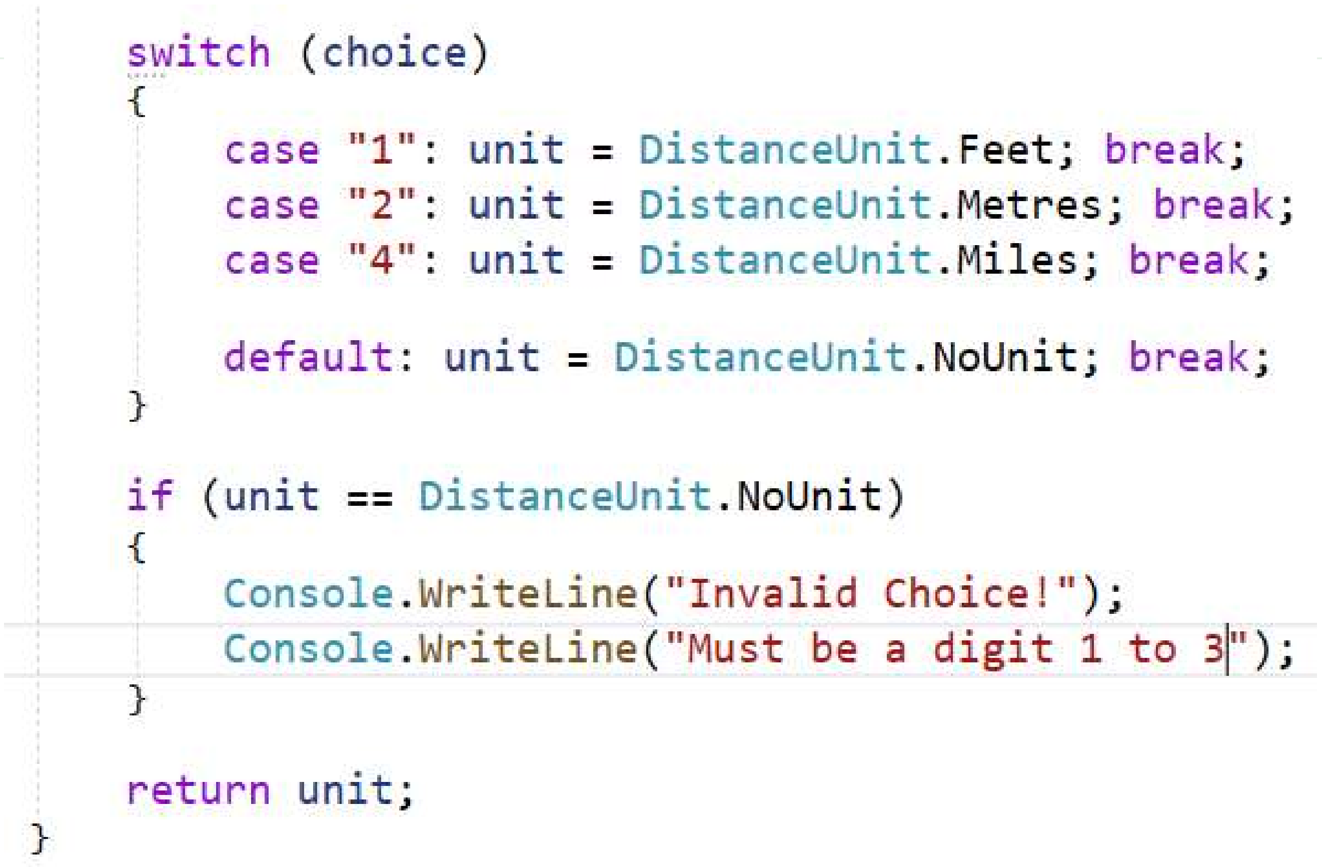
DistanceUnit or DistanceUnits?



```
/// <summary>
/// Display a menu of distance units and then prompt t
/// user to select one and return it.
/// </summary>
2 references
private DistanceUnit SelectUnit(string prompt)
{
    Console.WriteLine();
    Console.WriteLine($" 1. {DistanceUnit.Feet}");
    Console.WriteLine($" 2. {DistanceUnit.Metres}");
    Console.WriteLine($" 3. {DistanceUnit.Miles}");
    Console.WriteLine();

    Console.Write(prompt);
    string choice = Console.ReadLine();
```

Use whenever an attribute has a limited set of values

Using Enumerations



```
    switch (choice)
    {
        case "1": unit = DistanceUnit.Feet; break;
        case "2": unit = DistanceUnit.Metres; break;
        case "4": unit = DistanceUnit.Miles; break;

        default: unit = DistanceUnit.NoUnit; break;
    }

    if (unit == DistanceUnit.NoUnit)
    {
        Console.WriteLine("Invalid Choice!");
        Console.WriteLine("Must be a digit 1 to 3");
    }

    return unit;
}
```

The enumeration provides a structure instead of an unrelated set of string constants.

Week 2 Independent Study

Complete

Complete all 6 distance conversions and test fully.

Add Evaluation

App 01 version 1.4

Create

Create a new App02 folder and a class that will calculate a user's BMI index in either Imperial or Metric units

App 02 version 1.0

Add

Add a menu for the user of the **Program** to select whether they want to run the Distance Converter or the BMI calculator class

BMI – Leave testing till next week

Features: Distance Converter

The user can select any distance conversion between feet, metres and miles

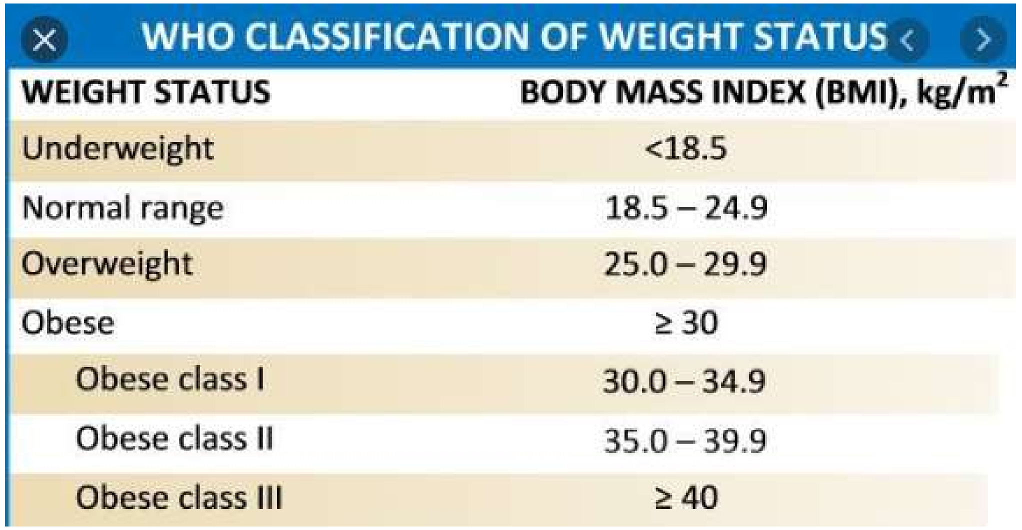
The program will convert the distance entered in one unit into the equivalent distance in the other unit.

APP 02 Version 1.0 - Body Mass Index

Calculating BMI (Body Mass Index)

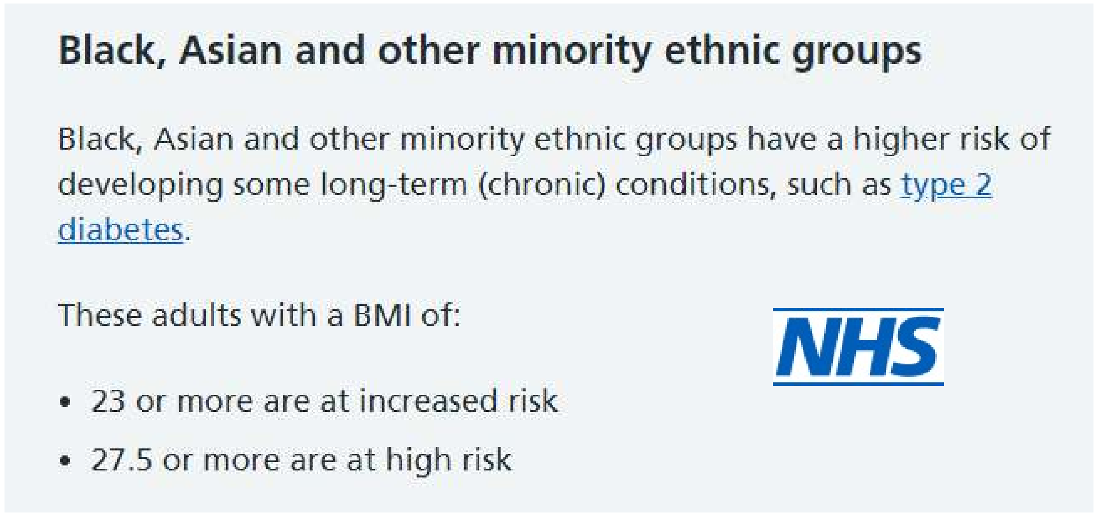
- BMI = weight in kg/(height in metres)^2
- BMI = weight in pounds x 703/(height in inches)^2

A lot of people in the uk still measure height in feet and inches and weight in stones and pounds



WHO CLASSIFICATION OF WEIGHT STATUS

WEIGHT STATUS	BODY MASS INDEX (BMI), kg/m^2
Underweight	<18.5
Normal range	18.5 – 24.9
Overweight	25.0 – 29.9
Obese	≥ 30
Obese class I	30.0 – 34.9
Obese class II	35.0 – 39.9
Obese class III	≥ 40



Black, Asian and other minority ethnic groups

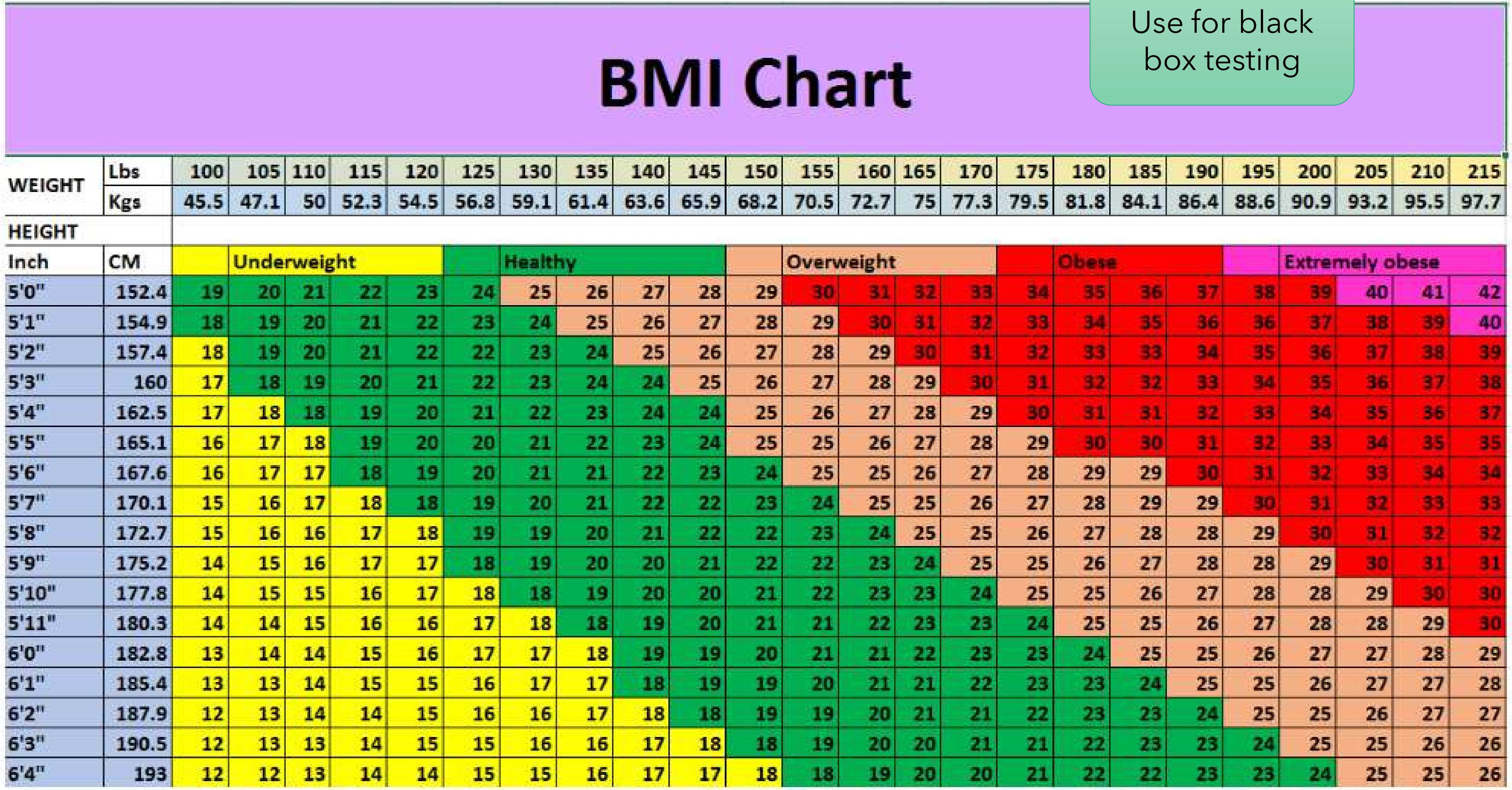
Black, Asian and other minority ethnic groups have a higher risk of developing some long-term (chronic) conditions, such as type 2 diabetes.

These adults with a BMI of:

- 23 or more are at increased risk
- 27.5 or more are at high risk

NHS

Use for black box testing



BMI Chart

WEIGHT	Lbs	100	105	110	115	120	125	130	135	140	145	150	155	160	165	170	175	180	185	190	195	200	205	210	215
	Kgs	45.5	47.1	50	52.3	54.5	56.8	59.1	61.4	63.6	65.9	68.2	70.5	72.7	75	77.3	79.5	81.8	84.1	86.4	88.6	90.9	93.2	95.5	97.7
HEIGHT																									
Inch	CM		Underweight					Healthy					Overweight					Obese				Extremely obese			
5'0"	152.4	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
5'1"	154.9	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	36	37	38	39	40
5'2"	157.4	18	19	20	21	22	22	23	24	25	26	27	28	29	30	31	32	33	33	34	35	36	37	38	39
5'3"	160	17	18	19	20	21	22	23	24	24	25	26	27	28	29	30	31	32	32	33	34	35	36	37	38
5'4"	162.5	17	18	18	19	20	21	22	23	24	24	25	26	27	28	29	30	31	31	32	33	34	35	36	37
5'5"	165.1	16	17	18	19	20	20	21	22	23	24	25	25	26	27	28	29	30	30	31	32	33	34	35	35
5'6"	167.6	16	17	17	18	19	20	21	21	22	23	24	25	25	26	27	28	29	29	30	31	32	33	34	34
5'7"	170.1	15	16	17	18	18	19	20	21	22	22	23	24	25	25	26	27	28	29	29	30	31	32	33	33
5'8"	172.7	15	16	16	17	18	19	19	20	21	22	22	23	24	25	25	26	27	28	28	29	30	31	32	32
5'9"	175.2	14	15	16	17	17	18	19	20	20	21	22	22	23	24	25	25	26	27	28	28	29	30	31	31
5'10"	177.8	14	15	15	16	17	18	18	19	20	20	21	22	23	23	24	25	25	26	27	28	28	29	30	30
5'11"	180.3	14	14	15	16	16	17	18	18	19	20	21	21	22	23	23	24	25	25	26	27	28	28	29	30
6'0"	182.8	13	14	14	15	16	17	17	18	19	19	20	21	21	22	23	23	24	25	25	26	27	27	28	29
6'1"	185.4	13	13	14	15	15	16	17	17	18	19	19	20	21	21	22	23	23	24	25	25	26	27	27	28
6'2"	187.9	12	13	14	14	15	16	16	17	18	18	19	19	20	21	21	22	23	23	24	25	25	26	27	27
6'3"	190.5	12	13	13	14	15	15	16	16	17	18	18	19	20	20	21	21	22	23	23	24	25	25	26	26
6'4"	193	12	12	13	14	14	15	15	16	17	17	18	18	19	20	20	21	22	22	23	23	24	25	25	26

Features: BMI Calculator

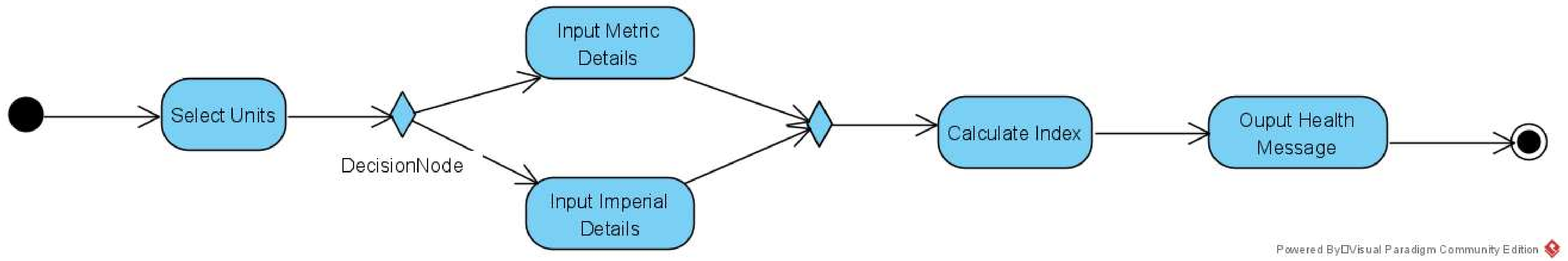
1. The user can select to enter their height and weight in metric or imperial units

2. The program will calculate their BMI index

3. The program will display an appropriate health message based on the BMI index.

4. The program will display a health message for Black Asian and other ethnic minority groups.

App 02: Design



References

- 2008 Microsoft Naming Conventions
- 2019 Microsoft Architectural Principles
- 2011 Best Practice Readable Code
- Refactoring
- C# If Statement
- Switch Statement
- Agile Alliance

```
irror_mod = modifier_ob.
mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
_operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
_operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

selection at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.activ
("Selected" + str(modifier
irror_ob.select = 0
bpy.context.selected_obj
ata.objects[one.name].se

int("please select exactl

- OPERATOR CLASSES ----

types.Operator):
X mirror to the selected
ject.mirror_mirror_x"
rror X"

ontext):
xt.active_object is not
```