Program:	**CPA2-5**
Course:	**INFO-5060**
Professor:	**Tony Haworth**
Project:	***Scrabble™ Scores***
Due Date:	**Monday, Feb 6th, 2023 by 11:59 pm**
Last Update:	**Jan 19th** *– Corrected a typo (QuiddlerLibrary should have been ScrabbleLibrary)*

This project may be done individually or in groups of two.
Students in different sections of the course may work together.
Sign-up for a project group in FOL even if you work alone!

Description

You will create a .NET 6.0 (.NET Core) library assembly using C# that provides game services for the board game Scrabble™. You must also implement a .NET 6.0 client that demonstrates the services provided by the class library by managing user inputs and generating all output in the console.

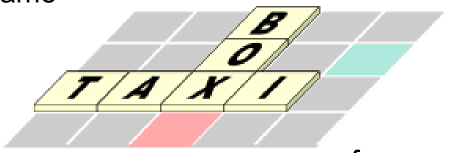
Purpose

On completing this project you should be able to:

1. develop a .NET class library assembly and a console client
2. use the *interface* type in C# to define component interfaces
3. integrate managed and unmanaged components into a .NET application

Background

You will **partially** implement the game *Scrabble™* (https://www.scrabblepages.com/scrabble/rules/), a multiplayer game in which the players take turns forming words on a game board. The words are formed from tiles and each tile contains one letter along with its point value. The tiles are distributed to the player from a bag and each player starts with seven random tiles which the player places on their *rack*. The objective is to form a word with the highest possible point value. The player can use any subset of *tiles* from her/his *rack* to do this. Each tile may only be used once within the word and the score for the word is based on the sum of the point values for each tile. (The scoring is actually more complicated than this, but we'll keep it simpler for this project.)



The complete set of *Tiles* (omitting blank *tiles*) can be described as follows:

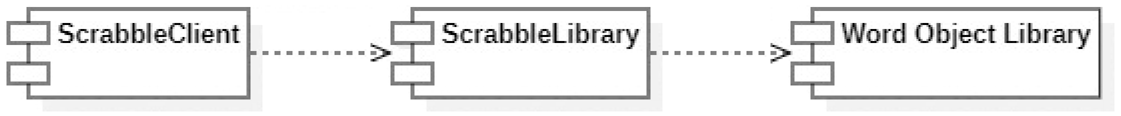
Number of Tiles in the Bag (98 total)

Tile	*Count*
j, k, q, x, z	1
b, c, f, h, m, p, v, w, y	2
g	3
d, l, s, u	4
n, r, t	6
o	8
a, i	9
e	12

Tile Point Values

Tile	*Points*
a, e, i, l, n, o, r, s, t, u	1
d, g	2
b, c, m, p	3
f, h, v, w, y	4
k	5
j, x	8
q, z	10

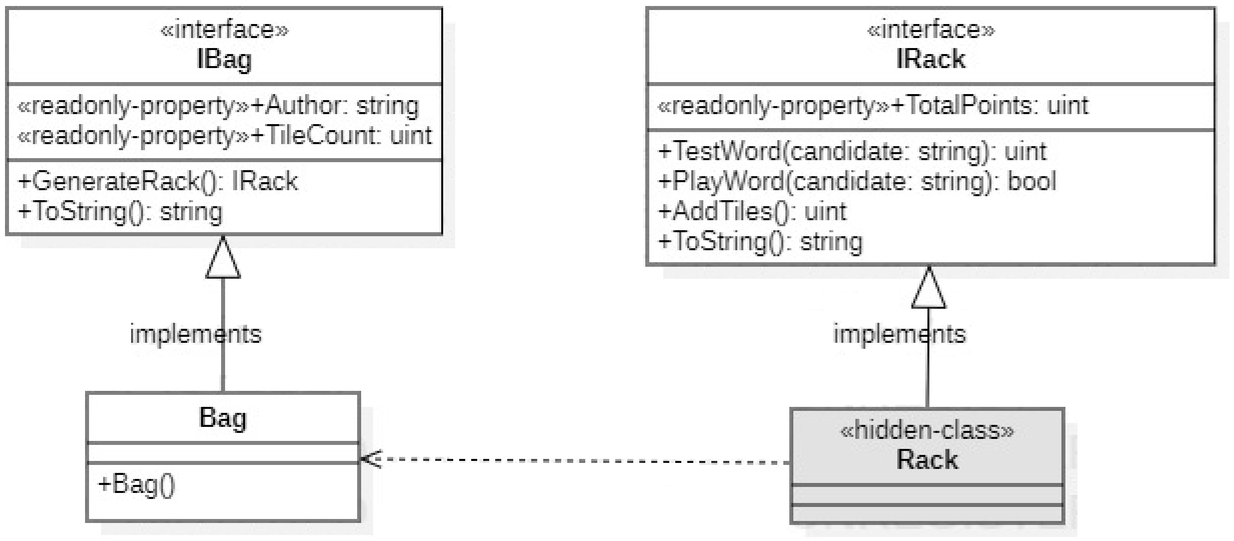
Component Architecture



The diagram above illustrates the software components involved in this project. You will be responsible for developing the *ScrabbleLibrary.dll* library and a client. The *Microsoft Word Object Library* is an existing native/unmanaged (COM) component that should already be available on any computer that has *Microsoft Office* installed and will be used to spell-check words to ensure they are valid. Note that to use the Word Object Library your *ScrabbleLibrary* must also reference the *Microsoft Office Object Library*.

Exposed Types

ScrabbleLibrary


«interface»
IBag
«readonly-property»+Author: string
«readonly-property»+TileCount: uint
+GenerateRack(): IRack
+ToString(): string
implements
Bag
+Bag()
«interface»
IRack
«readonly-property»+TotalPoints: uint
+TestWord(candidate: string): uint
+PlayWord(candidate: string): bool
+AddTiles(): uint
+ToString(): string
implements
«hidden-class»
Rack

The *ScrabbleLibrary* component must expose the *IBag* and *IRack* interfaces and the *Bag* class shown in the diagram above and further described below. These types must belong to the *ScrabbleLibrary* namespace. It must not expose any other types. Note that the Rack class should not be visible to any client of ScrabbleLibrary.dll.

IMPORTANT: For full marks, the interfaces must be defined exactly as shown in the diagram without any additional members. The only exception to this is that your interfaces may extend the standard *IDisposable* interface for finalization purposes. The interfaces only show methods and properties that must be exposed to an external client. However, *y*ou may also include any other members in your *implementation classes* (*Bag* and *Rack*) as needed to make your implementation functional as long as they are not visible to the client. You may also add classes or other types as required to complete your implementation as long as these classes are also not visible to the client.

Required Public Features of ScrabbleLibrary:

Bag Class	
Bag	A default constructor method that initializes the Bag with a full set of tiles.

IBag Interface Properties	
Author	Read-only property that returns a string identifying you as the author of the library *(see the sample output).*
TileCount	Read-only property that indicates how many tiles remain in the *Bag.*

IBag Interface Methods	
GenerateRack	Returns an *IRack* interface reference to a new Rack object already populated with seven *tiles.* One exception to this would be if there were fewer than seven tiles remaining in the Bag when the method was called.
ToString	Returns a description of the contents of the Bag as a single string object. Each letter that is present in the Bag is represented in the format **`a(9)`** which indicates there are 9 'a' tiles in the bag. The tiles should be listed in alphabetical order and letters that are absent should also be absent from the string returned. Letters should be delimited by whitespace (see the sample output).

IRack Interface Properties	
TotalPoints	Read-only property that represents the total points scored with the current Rack object.

IRack Interface Methods	
TestWord	Accepts a string containing a *candidate* word and returns its potential point value based on two criteria (if a candidate word fails to meet either criteria the method return 0): 1) The letters of the *candidate* string are a subset of the letters in the current Rack object. 2) The candidate provided is a valid word as tested using the *Application* class's *CheckSpelling()* method.
PlayWord	This method first tests the *candidate* word like *TestWord()* does. If the word is valid it removes the letters of the word from the Rack object, updates the value of the *TotalPoints* property by adding to it the points scored by the current word and then returns true. If the word is invalid it simply returns false
AddTiles	Does nothing if the Rack already has seven tiles or if the Bag is empty, otherwise it adds tiles to the rack from the Bag until either the rack contains seven tiles or the Bag is empty. The method then returns the number of tiles in the Rack (usually 7).

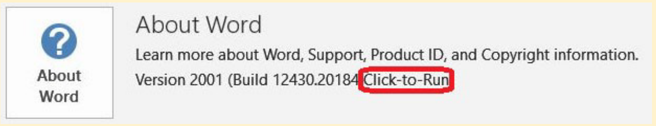
Required Hidden Features of ScrabbleLibrary:

Rack Class
This is a hidden class that implements *IRack.* You may want this class to include a one-argument constructor that accepts a Bag object. This is because a Rack object should only be constructed by a Bag's *GenerateRack()* method and any Rack object needs a reference to its Bag object to allow it to obtain tiles from the Bag.

Microsoft Word Object Library

Prerequisites:

- You must have a desktop ("Click-to-run") version of Microsoft Word installed.

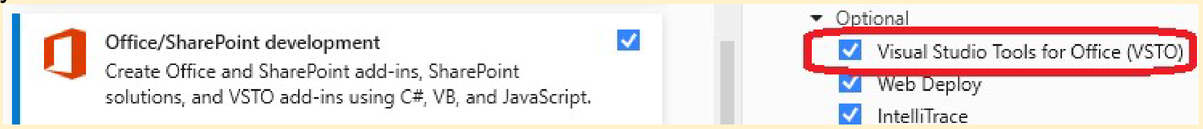


About Word

Learn more about Word, Support, Product ID, and Copyright information.

Version 2001 (Build 12430.20184 Click-to-Run

- You must have *Visual Studio Tools for Office* installed! This may already have been installed when you installed Visual Studio.



Office/SharePoint development

Create Office and SharePoint add-ins, SharePoint solutions, and VSTO add-ins using C#, VB, and JavaScript.

Optional

✓ Visual Studio Tools for Office (VSTO)

✓ Web Deploy

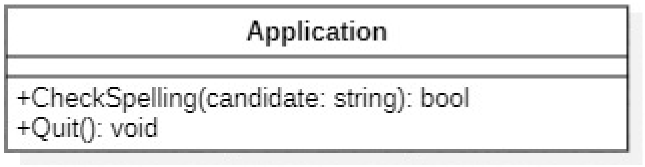
✓ IntelliTrace

- Your ScrabbleLibrary must reference both of the following <u>unmanaged</u> components:
  - *Microsoft Word Object Library* and *Microsoft Office Object Library*.

This is an <u>EXISTING</u> unmanaged library component compiled to native code rather than CIL. It exposes many classes, but for this project you will only require the following class and methods.

<u>Application Class</u>

- An object of this type is obtained by instantiating *Microsoft.Office.Interop.Word.Application*. This will first require project references to the COM components *Microsoft Word ##.0 Object Library* (version 15.0 or later will do) <u>and</u> *Microsoft Office ##.0 Object Library.*
- The *CheckSpelling()* method also has several optional parameters but only an argument for the *word* parameter must be provided when calling it. Note that only words in **<u>lowercase</u>** format will be properly spell-checked. Your *ScrabbleLibrary* will use this to validate words passed to the *IRack* methods *TestWord()* and *PlayWord()*.



Application

+CheckSpelling(candidate: string): bool
+Quit(): void

Your *ScrabbleLibrary.dll* should be designed to "finalize" the *Application* object by calling the object's *Quit* method before it gets cleaned-up by the garbage collector. This will unload an instance of MS Word that was loaded by your library.

The requirements continue on the next page.

Other Requirements

ScrabbleLibrary

- Must be implemented as a .NET 6.0 (.NET Core) class library assembly using C#
- Must work with your own client
- Must recognize letters in lowercase format. May optionally recognize UPPERCASE letters as well.
- Must not directly produce any output, not even error messages

Client

- Must be implemented as a .NET 6.0 console application using C#
- Must only make calls on the library's *Bag* and *Rack* objects via the *IBag* and *IRack* interfaces.
- Must demonstrate the features of the *ScrabbleLibrary* assembly by doing the following:
  - Create one *Bag* object
  - Show *ScrabbleLibrary* author information using *IBag.Author*
  - Show the tiles in the bag via *IBag.ToString()* and the number of tiles via *IBag.TileCount* each time these change
  - Create any number of populated Rack objects via *IBag.GenerateRack()*
  - For each Rack (player):
    - Show the tiles in the player's Rack via *IRack.ToString()*
    - Prompt the user to input a word and test the word using *IRack.TestWord()*.
    - If the word is worth points give the user the option to play the word, then play it using *IRack.PlayWord.*
    - If the word is worth 0 points allow the user further chances to input a valid word.
    - After the user has player a word, show the total points for the player using *IRack.TotalPoints,*
    - At the end of the user's turn (even if they don't play a word) show the contents of their Rack and of the Bag.
  - Obtain the points for a word via *IRack.TestWord()*
  - Play a word via *IRack.PlayWord()*
  - Repopulate a Rack with *IRack.AddTiles()* after playing a word
  - After each Rack (player) has taken a turn, give the user the choice to either quit, or allow each player to play another turn using the same Rack objects. However, when the Bag is empty the client should definitely stop processing turns.
  - When the program is quitting it should report the final values of *IRack.TotalPoints* for each Rack (player).
- Must incorporate exception handling for all calls on *ScrabbleLibrary* methods to minimize any possibility of unhandled exceptions from occurring.

The requirements continue on the next page.

Sample Output

```
Testing ScrabbleLibrary [Author: T. Haworth - January 16, 2023]

Bag initialized with the following 98 tiles..
a(9)  b(2)  c(2)  d(4)  e(12) f(2)  g(3)  h(2)  i(9)  j(1)  k(1)  l(4)  m(2)
n(6)  o(8)  p(2)  q(1)  r(6)  s(4)  t(6)  u(4)  v(2)  w(2)  x(1)  y(2)  z(1)

Enter the number of players (1-8): 2

Racks for 2 players were populated.
Bag now contains the following 84 tiles..
a(9)  b(1)  c(2)  d(2)  e(11) f(2)  g(3)  h(2)  i(7)  j(1)  l(3)  m(2)  n(6)
o(6)  p(2)  q(1)  r(4)  s(4)  t(6)  u(3)  v(1)  w(2)  x(1)  y(2)  z(1)

-----------------------------------------------------------------------------
                                  Player 1
-----------------------------------------------------------------------------
Your rack contains [dvoriud].
Test a word for its points value? (y/n): y
Enter a word using the letters [dvoriud]: car
The word [car] is worth 0 points.
Test a word for its points value? (y/n): y
Enter a word using the letters [dvoriud]: door
The word [door] is worth 0 points.
Test a word for its points value? (y/n): y
Enter a word using the letters [dvoriud]: druid
The word [druid] is worth 7 points.
Do you want to play the word [druid]? (y/n): y

        ------------------------------
        Word Played:         druid
        Total Points:        7
        Rack now contains:   [vonplet]
        ------------------------------

Bag now contains the following 79 tiles..
a(9)  b(1)  c(2)  d(2)  e(10) f(2)  g(3)  h(2)  i(7)  j(1)  l(2)  m(2)  n(5)
o(6)  p(1)  q(1)  r(4)  s(4)  t(5)  u(3)  v(1)  w(2)  x(1)  y(2)  z(1)

-----------------------------------------------------------------------------
                                  Player 2
-----------------------------------------------------------------------------
Your rack contains [oerbikl].
Test a word for its points value? (y/n): y
Enter a word using the letters [oerbikl]: biker
The word [biker] is worth 11 points.
Do you want to play the word [biker]? (y/n): y

        ------------------------------
        Word Played:         biker
        Total Points:        11
        Rack now contains:   [olaiehs]
        ------------------------------
```

```
Bag now contains the following 74 tiles..
a(8)  b(1)  c(2)  d(2)  e(9)  f(2)  g(3)  h(1)  i(6)  j(1)  l(2)  m(2)  n(5)
o(6)  p(1)  q(1)  r(4)  s(3)  t(5)  u(3)  v(1)  w(2)  x(1)  y(2)  z(1)

Would you like each player to take another turn? (y/n): y

-----------------------------------------------------------------------------
                                  Player 1
-----------------------------------------------------------------------------
Your rack contains [vonplet].
Test a word for its points value? (y/n): y
Enter a word using the letters [vonplet]: plot
The word [plot] is worth 6 points.
Do you want to play the word [plot]? (y/n): y

        ------------------------------
        Word Played:         plot
        Total Points:        13
        Rack now contains:   [vnewazb]
        ------------------------------

Bag now contains the following 70 tiles..
a(7)  c(2)  d(2)  e(9)  f(2)  g(3)  h(1)  i(6)  j(1)  l(2)  m(2)  n(5)  o(6)
p(1)  q(1)  r(4)  s(3)  t(5)  u(3)  v(1)  w(1)  x(1)  y(2)

-----------------------------------------------------------------------------
                                  Player 2
-----------------------------------------------------------------------------
Your rack contains [olaiehs].
Test a word for its points value? (y/n): y
Enter a word using the letters [olaiehs]: shoal
The word [shoal] is worth 8 points.
Do you want to play the word [shoal]? (y/n): y

        ------------------------------
        Word Played:         shoal
        Total Points:        19
        Rack now contains:   [iegndie]
        ------------------------------

Bag now contains the following 65 tiles..
a(7)  c(2)  d(1)  e(8)  f(2)  g(2)  h(1)  i(5)  j(1)  l(2)  m(2)  n(4)  o(6)
p(1)  q(1)  r(4)  s(3)  t(5)  u(3)  v(1)  w(1)  x(1)  y(2)

Would you like each player to take another turn? (y/n): n

Retiring the game.

The final scores are...
-----------------------------------------------------------------------------
Player 1: 13 points
Player 2: 19 points
-----------------------------------------------------------------------------
```

Tips and Suggestions

- The interfaces *IBag* and *IRack* may be implemented either implicitly or explicitly. For example:

```
public class Bag : IBag
{
    public string ToString()
    {
        // implicit
```

```
public class Bag : IBag
{
    string IBag.ToString()
    {
        // explicit
```

- To assign points for candidate words in your *ScrabbleLibrary.dll* you'll first need to ensure the candidate word consists of letters that are a subset of the available letters in the current rack object. You can try searching for an algorithm online. However, one algorithm involves creating a data structure of pairs with the key value being a letter and the data value being the frequency (# of instances) of the letter in the rack. You could use a similar data structure for the candidate word. The algorithm could then loop through the second data structure comparing the frequency of each letter with the frequency of the same letter in the first data structure. One suitable data structure for this in .NET is the *Dictionary*<> type.
- You may find the performance of your client to be poor (i.e. sluggish) if your library creates a new *Application* object for every word it needs to validate. To avoid this, consider creating a single reusable instance of *Application* that will exist until the client app terminates.

Grading Scheme

1. The School of I.T. Policy on Missed Evaluations and Evaluation Deadlines applies to this project.
2. Projects submitted by the submission deadline will be evaluated as follows:

Description	**Marks**
ScrabbleLibrary is a .NET 6.0 class library project and correctly defines, implements and exposes the published interfaces *IBag* and *IRack* and the public class *Bag*, but hides all other types.	10
These features of the *IBag* and *IRack* interfaces in *ScrabbleLibrary* work according to the requirements provided 1. IBag : Author, GenerateRack(), TileCount, ToString() 2. IRack : TestWord(), PlayWord(), AddTiles(), ToString(), TotalPoints	15
ScrabbleLibrary uses the *Microsoft Word Object Library* component's *Application* class effectively to validate candidate words. It also invokes the component's *Quit()* method before the client application terminates.	5
Client submitted effectively demonstrates all *ScrabbleLibrary* features exclusively via the IBag and IRack interfaces and handles exceptions thrown by the *ScrabbleLibrary*	10
Maximum deduction for generating output from ScrabbleLibrary.dll	-5
Maximum deduction for poor coding style (unclear and inconsistent or undescriptive naming of code elements, poor code alignment, missing and/or inadequate inline and header comments)	-5
Maximum deduction for an incomplete submission (missing source code or required VS project files)	-40
Maximum Total	**40**

2. Submissions to the FOL drop box should be made on time! A deduction will apply to late submissions as follows:
    - A penalty of 20% will be deducted if your submission is received < 24 hours late.
    - Your project will receive a mark of 0 (zero) if it is more than 24 hours late.

Submit

Submit one archive file (zip, rar or 7z file) containing your complete Visual Studio solution(s) with all your source code. I should only have to rebuild your project before testing without changing any project settings or source code.

If working with another student only one of you needs to submit your solution, but you will both be responsible for ensuring you submit it on time.

Project Corrections

If any corrections or changes are necessary they will be posted on the course web site and you will be notified of any changes in class. It is your responsibility to check the site periodically for changes to the project. Additional resources relating to the project may also be posted.