Here are the packages with brief descriptions (if available):

- **P1_Base**

Generated by *doxygen* 1.8.13
Dong Minh Doxygen

P1_Base Namespace
Reference
# Classes

```
class Program
```

Generated by [doxygen](https://www.doxygen.org) 1.8.13
Here are the classes, structs, unions and interfaces with brief descriptions:

<table>
<thead>
<tr>
<th>▼ N P1_Base</th>
<th>Program</th>
</tr>
</thead>
</table>

Generated by [doxygen](http://www.doxygen.nl) 1.8.13
Collaboration diagram for P1_Base.Program:

```
+ Factorization()
+ GCD()
+ LCD()
+ CF()
- Main()
```
### Static Public Member Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static List&lt; int &gt; Factorization (int num)</code></td>
<td>The factorization will find the prime numbers, place it in the list and repeat until factorization can no longer be done. After that, return the list so it can be displayed. <a href="#">More...</a></td>
</tr>
<tr>
<td><code>static int GCD (int a, int b)</code></td>
<td>The greatest common denominator (GCD) will take in the numbers and will find the GCD value, then return it to display later. <a href="#">More...</a></td>
</tr>
<tr>
<td><code>static int LCD (int a, int b)</code></td>
<td>The least common denominator (LCD) will take in the numbers and will find the LCD value, then return it to display later. <a href="#">More...</a></td>
</tr>
<tr>
<td><code>static int CF (int a, int b)</code></td>
<td>The common factors will be found when two numbers are input into this <a href="#">More...</a></td>
</tr>
</tbody>
</table>
### Static Private Member Functions

<table>
<thead>
<tr>
<th>static void</th>
<th><strong>Main</strong> (string[] args)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The main function to run the entire program. This program will ask the user to input two numbers, then the program will calculate the GCF and LCD automatically. Also, it will display to the user with some output.</td>
<td>More...</td>
</tr>
</tbody>
</table>
Detailed Description

Definition at line 9 of file Program.cs.
Member Function Documentation
CF()

```csharp
static int P1_Base.Program.CF ( int a,
    int b )
```

The common factors will be found when two numbers are input into this

**Pseudo Code**

Continue to loop until the highest common factor is found

**Parameters**

- `a` Integer A Value
- `b` Integer B Value

**Returns**

Returns the List

Definition at line 140 of file `Program.cs`.

Here is the caller graph for this function:

```
P1_Base.Program.CF
```

```
P1_Base.Program.Main
```
◆ Factorization()

```csharp
static List<int> P1_Base.Program.Factorization ( int num )
```

The factorization will find the prime numbers, place it in the list and repeat until factorization can no longer be done. After that, return the list so it can be displayed.

**Pseudo Code**

- If there are any 2s, take it out.
- If there are any other primes, take it out.
- If the number is not 1, then whatever is left is prime.

**Parameters**

- **num** Integer list

**Returns**

Returns the List

Definition at line 29 of file *Program.cs*.

Here is the caller graph for this function:
GCD()

static int P1_Base.Program.GCD ( int a,  
    int b  
  )

The greatest common demonitor (GCD) will take in the numbers and will find the GCD value, then return it to display later.

Pseudo Code

If a is greater than b, a = a % b
If b is greater than a, b = b % a

Parameters
  a Integer A Value
  b Integer B Value

Returns
Returns values of a and b

Definition at line 76 of file Program.cs.

Here is the caller graph for this function:
◆ LCD()

static int P1_Base.Program.LCD ( int a,
    int b
  )

The least common denominator (LCD) will take in the numbers and will find the LCD value, then return it to display later.

**Pseudo Code**

Store the values if a > b or b > a for num1 and num2 respectively (both)

Keep looping for the lowest common denominator until the LCD is satisfied

**Parameters**

a Integer A value
b Integer B value

**Returns**

Returns the number for LCD

Definition at line 104 of file Program.cs.

Here is the caller graph for this function:

[Diagram showing caller graph with nodes labeled P1 Base Program LCD and P1 Base Program Main connected by an arrow]
Main()

static void P1_Base.Program.Main ( string [] args )

The main function to run the entire program. This program will ask the user to input two numbers, then the program will calculate the GCF and LCD automatically. Also, it will display to the user with some output.

Parameters

args

Returns

Loops if Y/y, Stops if N/n

Definition at line 163 of file Program.cs.

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- D:/Google Drive/ODU/Documents/2018/MSIM 408 - Introduction to Game Development/P1/P1_Base(1)/P1_Base/P1_Base/Program.cs

Generated by doxygen 1.8.13
## Class Index

<table>
<thead>
<tr>
<th>p</th>
</tr>
</thead>
</table>

[Program (P1_Base)]

Generated by [Doxygen 1.8.13](https://www.doxygen.org/)
Here is a list of all class members with links to the classes they belong to:

- CF() : \texttt{P1\_Base.Program}
- Factorization() : \texttt{P1\_Base.Program}
- GCD() : \texttt{P1\_Base.Program}
- LCD() : \texttt{P1\_Base.Program}
- Main() : \texttt{P1\_Base.Program}

Generated by \texttt{doxygen} 1.8.13
Dong_Minh_Doxygen

- CF() : \texttt{P1\_Base\_Program}
- Factorization() : \texttt{P1\_Base\_Program}
- GCD() : \texttt{P1\_Base\_Program}
- LCD() : \texttt{P1\_Base\_Program}
- Main() : \texttt{P1\_Base\_Program}

Generated by doxygen 1.8.13
Here is a list of all files with brief descriptions:

- D:
  - Google Drive
  - ODU
    - Documents
    - 2018
      - MSIM 408 - Introduction to Game Development
        - P1
          - P1_Base(1)
          - P1_Base
        - obj
        - Debug
          - TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs
          - TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs
          - TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs
        - Properties
          - AssemblyInfo.cs
          - Program.cs

Generated by doxygen 1.8.13
Dong Minh Doxygen

D: Directory Reference
### Directories

<table>
<thead>
<tr>
<th>directory</th>
<th>Google Drive</th>
</tr>
</thead>
</table>

Generated by doxygen 1.8.13
Dong_Minh_Doxygen

Google Drive Directory Reference
## Directories

| directory | ODU |

Generated by [doxygen](https://doxygen.net) 1.8.13
<table>
<thead>
<tr>
<th>D:</th>
<th>Google Drive</th>
<th>ODU</th>
</tr>
</thead>
</table>

**ODU Directory Reference**
Dong_Minh_Doxygen

<table>
<thead>
<tr>
<th>Documents Directory Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>D:</td>
</tr>
</tbody>
</table>
Directories

| directory | 2018 |

Generated by doxygen 1.8.13
Dong Minh Doxygen

2018 Directory Reference
Directories

directory  MSIM 408 - Introduction to Game Development

Generated by doxygen 1.8.13
Dong_Minh_Doxygen

MSIM 408 - Introduction to Game Development
Directory Reference
## Directories

| directory | P1 |

Generated by [doxygen](https://www.doxygen.org/) 1.8.13
Dong Minh Doxygen

P1 Directory Reference
<table>
<thead>
<tr>
<th>Directory</th>
<th>P1_Base(1)</th>
</tr>
</thead>
</table>

Generated by **doxygen** 1.8.13
| directory | P1_Base |
Dong Minh Doxygen

P1_Base Directory Reference
<table>
<thead>
<tr>
<th>directory</th>
<th>P1_Base</th>
</tr>
</thead>
</table>

Generated by [doxygen](http://www.doxygen.org) 1.8.13
Dong Minh Doxygen

P1_Base Directory Reference
## Directories

<table>
<thead>
<tr>
<th>directory</th>
<th>obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory</td>
<td>Properties</td>
</tr>
</tbody>
</table>
## Files

<table>
<thead>
<tr>
<th>file</th>
<th>Program.cs [code]</th>
</tr>
</thead>
</table>

Generated by [doxygen](https://www.doxygen.org/) 1.8.13
Dong Minh Doxygen

obj Directory Reference
<table>
<thead>
<tr>
<th>directory</th>
<th>Debug</th>
</tr>
</thead>
</table>

Generated by doxygen 1.8.13
Debug Directory Reference
# Files

<table>
<thead>
<tr>
<th>file</th>
<th>TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs</td>
</tr>
<tr>
<td>file</td>
<td>TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs</td>
</tr>
</tbody>
</table>

Generated by doxygen 1.8.13
Go to the source code of this file.
TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

Go to the source code of this file.
Go to the source code of this file.
<table>
<thead>
<tr>
<th>Properties Directory Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
</tr>
<tr>
<td>Directory</td>
</tr>
<tr>
<td>Reference</td>
</tr>
</tbody>
</table>
## Files

| file | AssemblyInfo.cs [code] |

Generated by [doxygen](http://doxygen.org) 1.8.13
Go to the source code of this file.
Go to the source code of this file.
## Classes

<table>
<thead>
<tr>
<th>class</th>
<th>P1_Base.Program</th>
</tr>
</thead>
</table>


Namespaces

namespace P1_Base
This is the complete list of members for `P1_Base.Program`, including all inherited members.

<table>
<thead>
<tr>
<th>Function</th>
<th>Class</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF(int a, int b)</td>
<td>P1_Base.Program</td>
<td>static</td>
</tr>
<tr>
<td>Factorization(int num)</td>
<td>P1_Base.Program</td>
<td>static</td>
</tr>
<tr>
<td>GCD(int a, int b)</td>
<td>P1_Base.Program</td>
<td>static</td>
</tr>
<tr>
<td>LCD(int a, int b)</td>
<td>P1_Base.Program</td>
<td>static</td>
</tr>
<tr>
<td>Main(string[] args)</td>
<td>P1_Base.Program</td>
<td>private static</td>
</tr>
</tbody>
</table>
namespace P1_Base
{

class Program
{

    public static List<int> Factorization(int num)
    {
        List<int> result = new List<int>();

        // Take out the 2s
        while (num % 2 == 0)
        {
            result.Add(2);
            num /= 2;
        }

        // Take out other primes.
        int factor = 3;
        while (factor * factor <= num)
        {
            if (num % factor == 0)
            {

            }

            factor += 2;
        }
    }
}
}
result.Add(factor);
    num /= factor;
} else factor += 2;
}

// If num is not 1, then whatever is left is prime
if (num > 1)
    result.Add(num);

return result;
} // end of Factorization function

public static int GCD(int a, int b)
{
    while (a != 0 && b != 0)
    {
        if (a > b)
            a %= b;
        else
            b %= a;
    }
    return a == 0 ? b : a;
} // End GCD

public static int LCD(int a, int b)
{
    int num1, num2;
    if (a > b)
    {
        num1 = a;
        num2 = b;
    }
else
{
    num1 = b;
    num2 = a;
}

for (int i = 1; i < num2; i++)
{
    if ((num1 * i) % num2 == 0)
        return i * num1;
}
return num1 * num2;

public static int CF(int a, int b)
{
    int i, j, hcf = 1;

    j = (a < b) ? a : b;
    for (i = 1; i <= j; i++)
        if (a % i == 0 && b % i == 0)
            hcf = i;

    return hcf;
}

static void Main(string[] args)
{
    int a = -1, b = -1;
    string sa, sb;
    bool isContinue = true;

while (isContinue)
{
    bool aValid = false, bValid = false;
    Console.WriteLine("Enter the first number:");
    while (!aValid)
    {
        sa = Console.ReadLine();
        try
        {
            a = Int32.Parse(sa);
        }
        catch (Exception e)
        {
            Console.WriteLine("{0} is not a valid integer.", sa);
        }
        if (a < 1 || a > 100)
            Console.WriteLine("Please enter a number between 1 and 100.");
        else
            aValid = true;
    }
    Console.WriteLine("Enter the second number:");
    while (!bValid)
    {
        sb = Console.ReadLine();
        try
        {
            b = Int32.Parse(sb);
        }
        catch (Exception e)
        {
            // Handle exceptions
        }
    }
}
Console.WriteLine("{0} is not a valid integer.", sb);

if (b < 1 || b > 100)
    Console.WriteLine("Please enter a number between 1 and 100.");
else
    bValid = true;

// **Enter your code here.**

// Get the first factor text and display it
Console.WriteLine("The factors of {0} are:", a);
var factorsA = string.Join(" ", Factorization(a));
Console.WriteLine(factorsA);

// Get the second factor text and display it
Console.WriteLine("The factors of {0} are:", b);
var factorsB = string.Join(" ", Factorization(b));
Console.WriteLine(factorsB);

// Find the highest common
number then factor and display it

```csharp
Console.WriteLine("The common factors of \{0\} and \{1\} are:", a, b);
int newCommonFactor = CF(a, b);
string primeCommonFactors = string.Join(" ", Factorization(newCommonFactor));

Console.WriteLine(primeCommonFactors);

// Get the GCF and LCD and display it
Console.WriteLine("The GCF of \{0\} and \{1\} is: \{2\}", a, b, GCD(a, b));
Console.WriteLine("The LCD of \{0\} and \{1\} is: \{2\}", a, b, LCD(a, b));

// **End writing own code**


Console.WriteLine("Do you want to continue? Y/N");
string newLoop = Console.ReadLine();
if (newLoop[0] == 'Y' || newLoop[0] == 'y')
{
    Console.WriteLine();
isContinue = true;
}
else
{
    isContinue = false;
}
```

Dong Minh Doxygen

TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs

Go to the documentation of this file.

Generated by doxygen 1.8.13
Go to the documentation of this file.
Dong Minh Doxygen

TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs

Go to the documentation of this file.

Generated by doxygen 1.8.13
Dong_Minh_Doxygen

AssemblyInfo.cs

Go to the documentation of this file.

```csharp
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("P1_Base")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Microsoft")]
[assembly: AssemblyProduct("P1_Base")]
[assembly: AssemblyCopyright("Copyright © Microsoft 2015")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
```
// Version information for an assembly consists of the following four values:

// Major Version
// Minor Version
// Build Number
// Revision

// You can specify all the values or you can default the Build and Revision Numbers by using the '*' as shown below:

[assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]