Table of Contents

Introduction

This documentation describes the functions, classes and utilities provided within the Speculous FuzzyLogic Assembly for the .Net Framework V2.

The aim of the assembly is to provide supporting functionality for the common functions, logical evaluations and schemas used in Fuzzy Logic.

What is Fuzzy Logic

It is assumed that the developer utilising this assembly will have a basic grounding in Fuzzy Logic theorem. For an introduction to Fuzzy Logic refer to the following links:-

- Fuzzy Systems – A Tutorial (http://www.ortech-engr.com/fuzzy/tutor.txt **)**
- Fuzzy Logic Introduction (www.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/jp6/article2.html)
- Stanford Encyclopaedia of Philosophy (http://plato.stanford.edu/entries/logic-fuzzy/).

Recommended books are:-

1. AI Application Programming - 2nd Edition – M. Tim Jones
2. AI for Game Developer – David M. Bourg & Glenn Seemann

Got suggestion for improvements or want to report a bug?

To suggest improvements or report bugs, please visit www.speculous.com

Licensing

License

The functionality found within this beta test product is free distributable and can be used in its compiled form within third poarty products without change. Any attempt to disassemble, change, duplicate functionality or update this product without express permission from Speculous Software Limited is strictly prohibited.

The compiled software product, it's source code, it's design, it's functionality/logic and its documentation are copyrighted by Speculous Software Limited.

Licensing

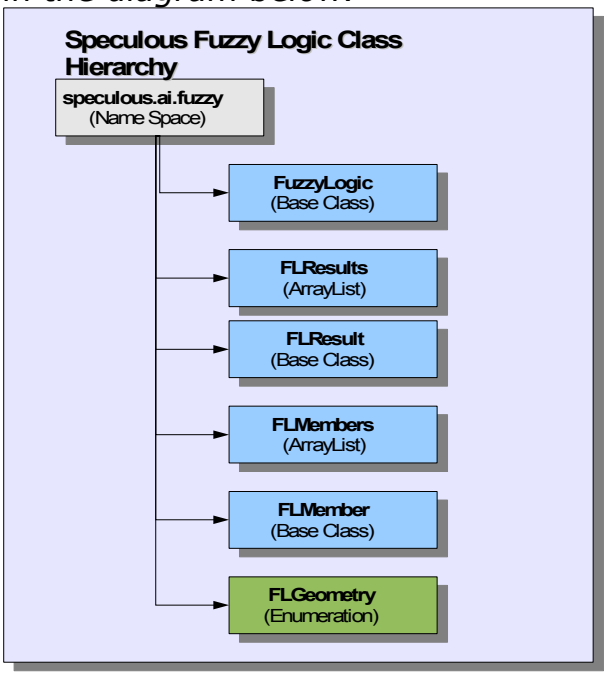
This assembly is free to be used in its compiled for for any purpose. This assembly must not be decoded, back-ward engineered in any way changed.

Please contact us at www.speculous.com and let us know what you are using this assembly for.

Assembly Outline

Assembly Structure

The classes and enumerations provided by Speculous FuzzyLogic are detailed in the diagram below.



Description

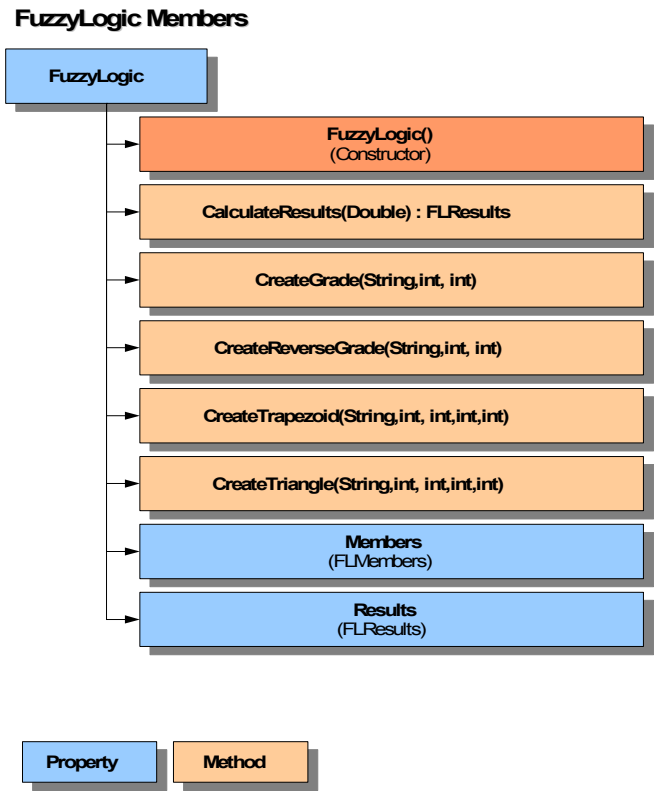
The assembly name space is **speculous.ai.fuzzy**. This assembly exposes the following classes and enumerations:-

- FuzzyLogic – This Class provided the core functionality and container for all Fuzzy Logic support. Typically this will be the only class type employed to create objects.
- FLMembers – A collection of Fuzzy Logic members (or geometries) defining the rules by which data is evaluated.
- FLMember – A class used to define a single member (or geometry).
- FLResults – A collection of results produce from the processing of data against a FLMember Collection.
- FLResult – A single result relating to a single FLMember.
- FLGeometry – An enumeration of the supported geometry types that can be used to define FLMembers.

FuzzyLogic Class

Class Structure

The diagram below describes the class structure:-



Description

This class provides the cornerstone for the application of fuzzy logic. This class provide the functionality.

Example

FuzzyLogic oFuzzy = new FuzzyLogic();

FuzzyLogic Constructor

Syntax

FuzzyLogic()

Parameters:-

None

Description

This constructor is used in conjunction with the **new** keyword to create a new Fuzzy Logic object.

Example

FuzzyLogic oFuzzy = new FuzzyLogic();

FuzzyLogic.GenerateResults

Syntax

GenerateResults(dblData):FLResults

Parameters:-

***dblData** – Double – This provides the data to be evaluated..*

Returns a FLResults collection containing the results of the evaluation of the data.

Description

This function evaluates the value provided (dblData) against each of rules (members) in the **FuzyLogic.Members** collection. The evaluation of each member (or geometry) will generate a single result into the returned **FLResults** collection. These results are also generated into the **FuzzyLogic.Results** collection.

Example

flResults oFLRes = oFuzzy.GenerateResults(22.75);

FuzzyLogic.CreateGrade

Syntax

CreateGrade(sName, iX1,iX2)

Parameters:-

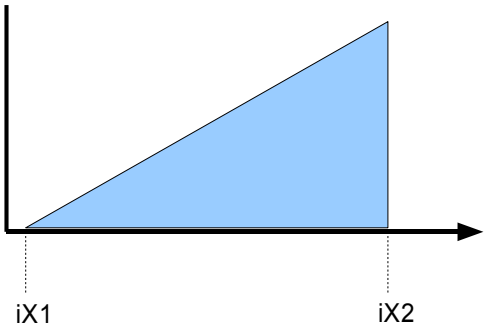
sName *– The name of the new grade member created.*

iX1 *– int– The starting point of the grade member upon the x-axis.*
iX2 *– int– The ending point of the grade member upon the x-axis.*

No Return value.

Description

This helper function creates a new FLMember object with the FuzzyLogic.Members collection with a grade geometry defined by the iX1 and iX2 parameters. The illustration below shows a fuzzy logic grade with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oFuzzy.CreateGrade("A Little", 0, 35);

FuzzyLogic.CreateReverseGrade

Syntax

CreateReverseGrade(sName, iX1,iX2)

Parameters:-

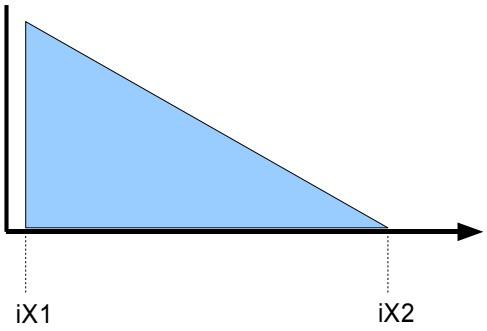
sName *– The name of the new reverse grade member created.*

iX1 *– int– The starting point of the grade member upon the x-axis.*
iX2 *– int– The ending point of the grade member upon the x-axis.*

No Return value.

Description

This helper function creates a new FLMember object with the FuzzyLogic.Members collection with a reverse grade geometry defined by the iX1 and iX2 parameters. The illustration below shows a fuzzy logic reverse grade with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oFuzzy.CreateReverseGrade("A Lot", 65, 100);

FuzzyLogic.CreateTrapezoid

Syntax

CreateTrapezoid(sName, iX1,iX2,iX3,iX4)

Parameters:-

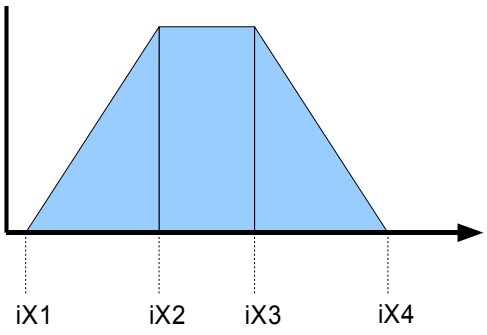
sName *– The name of the new trapezoid member created.*

iX1 *– int– The starting point of the trapezoid member upon the x-axis.*
iX2 *– int– The starting apex of the trapezoid member upon the x-axis.*
iX3 *– int– The ending apex of the trapezoid member upon the x-axis.*
iX4 *– int– The ending point of the trapezoid member upon the x-axis.*

No Return value.

Description

This helper function creates a new FLMember object with the FuzzyLogic.Members collection with a *trapezoid* geometry defined by the iX1, iX2, iX3 and iX4 parameters. The illustration below shows a fuzzy logic *trapezoid* with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oFuzzy.CreateTrapezoid("High", 35,45,55,65);

FuzzyLogic.CreateTriangle

Syntax

CreateTriangle(sName, iX1,iX2,iX3)

Parameters:-

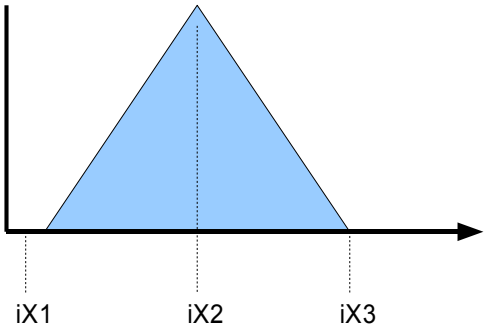
sName *– The name of the new triangle member created.*

iX1 *– int– The starting point of the triangle member upon the x-axis.*
iX2 *– int– The apex of the triangle member upon the x-axis.*
iX3 *– int– The ending point of the triangle member upon the x-axis.*

No Return value.

Description

This helper function creates a new FLMember object with the FuzzyLogic.Members collection with a *triangle* geometry defined by the iX1, iX2 and iX3 parameters. The illustration below shows a fuzzy logic *triangle* with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oFuzzy.CreateTriangle("High",10,50,90);

FuzzyLogic.Memebers Collection

Syntax

FuzzyLogic.Memebers

Description

This **FLMembers** collection holds the members (geometry rules) defining the fuzzy logic set. The members of this set (found in the Items[] array) can be created manually by adding an **FLMember** object to the collection or via the **Createxxxx** helper functions within the FuzzyLogic Object.

See **FLMembers** class definition for more information.

Example

FlMember oMbr = FuzzyLogic.Memebers.Items[0];

FuzzyLogic.Results Collection

Syntax

FuzzyLogic.Results

Description

This **FLResults** collection holds the results of evaluating the member rules (see **FuzzyLogic.Members** collection) against a value using the **FuzzyLogic.CalculateResults** Method. This collection of **FLResult** object are automatically generated.

See **FLResults** class definition for more information. Also see **FuzzyLogic.GenerateResults**.

Example

FlResult oRes = FuzzyLogic.Results.Items[0];

FuzzyLogic.Registered

Syntax

FuzzyLogic.Registered = True | False

Description

This property is read-only and indicates if the licence information provided in the constructor was valid and therefore the object is acting in a registered mode.

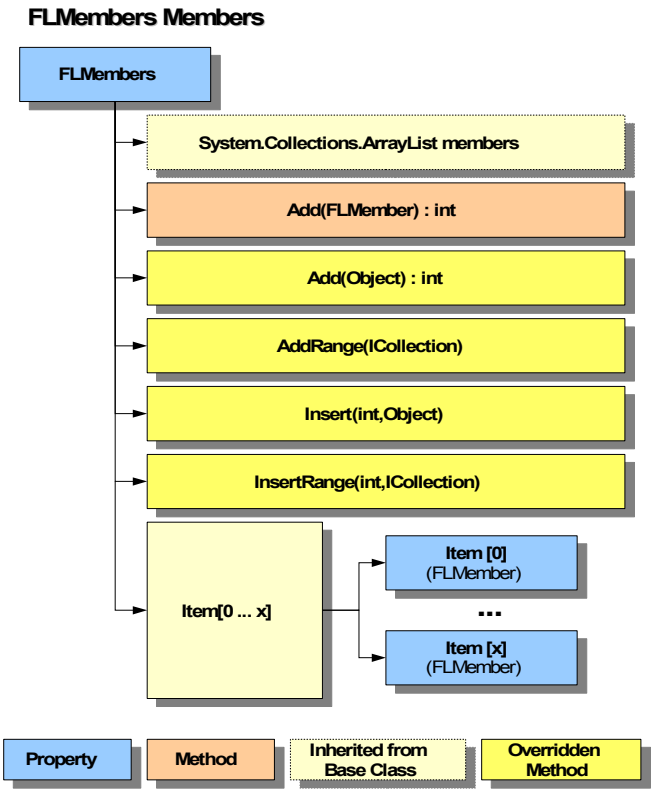
Example

```
 If (oFuzzy.Registered)
{
....
}
```

FLMembers Class (Collection)

Class Structure

The diagram below describes the class structure:-



Description

This class inherits and modifies the functionality of the **System.Collections.ArrayList** Class. It holds a collection of FLMember type objects.

FLMembers Inherited Class Modifications

Description

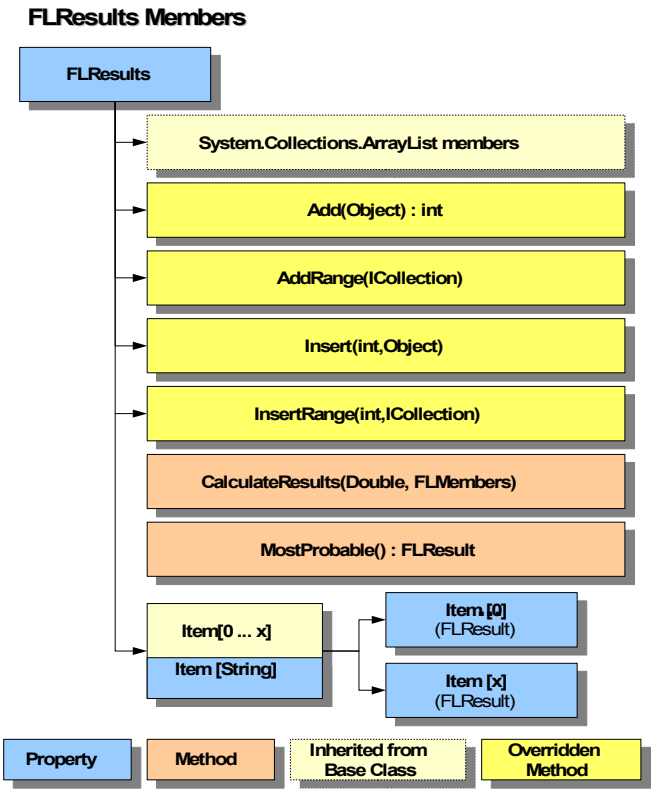
The FLMember class is derived from the ArrayList collection. Most functionality of the ArrayList class is implemented without change. The only changes made are designed to enforce types contained within the collection. The table below details the variation (over-ridden) functionality.

Method/Property	Modification
Add(FLMember)	A new implementation specifically for adding FLMember types has been provided
Add(Object)	Will accept any object as a parameter but will only add objects of FLMember type.
AddRange(Object)	Will accept any object as a parameter but will only add collections of FLMembers type.
Insert(int, Object)	Will accept any object as a parameter but will only add objects of FLMember type.
InsertRange(int, Object)	Will accept any object as a parameter but will only add collections of FLMembers type.

FLResults Class (Collection)

Class Structure

The diagram below describes the class structure:-



Description

This class inherits and modifies the functionality of the **System.Collections.ArrayList** Class. It holds a collection of FLMember type objects. It also exposes specific functionality for the manipulation of FLResult objects.

FLResults Inherited Class Modifications

Description

The FLResults class is derived from the ArrayList collection. Most functionality of the ArrayList class is implemented without change. The only changes made are designed to enforce types contained within the collection.
The table below details the variation (over-ridden) functionality.

Method/Property	Modification
Add(Object)	Will accept any object as a parameter but will only add objects of FLResult type.
AddRange(Object)	Will accept any object as a parameter but will only add collections of FLResult s type.
Insert(int, Object)	Will accept any object as a parameter but will only add objects of FLResult type.
InsertRange(int, Object)	Will accept any object as a parameter but will only add collections of FLResults type.
Items[String]	This variation of the Items[] property allows for individual results to be indexed in the array list using their FLResult.Name property. This name is derived from the FLMember relating to the result. e.g. *FLResult oRes=oResults.Items["Heavy"]*

FLResults.CreateResults

Syntax

GenerateResults(dblData, oMembers)

Parameters:-

***dblData** – Double – This provides the data to be evaluated.*

***oMembers** – FLMembers – The FLMembers collection used in the evaluation.*

Description

This function evaluates the value provided (dblData) against each of rules (members) in the oMembers collection. The evaluation of each member (or geometry) will generate a single result into this **FLResults** collection.

Example

oFLRes.CreateResults(22.75, oMembers);

FLResults.MostProbable

Syntax

MostProbable()

Parameters:-

None.

Returns an FLResult object.

Description

This method return an FLResult object containing the most probable result from the FLResults collection. "Most probable" is judged as the result in the collection that is nearest to absolute true (i.e. 1.00). If two or more result are equally most probable, the first in index order within the collection is returned.

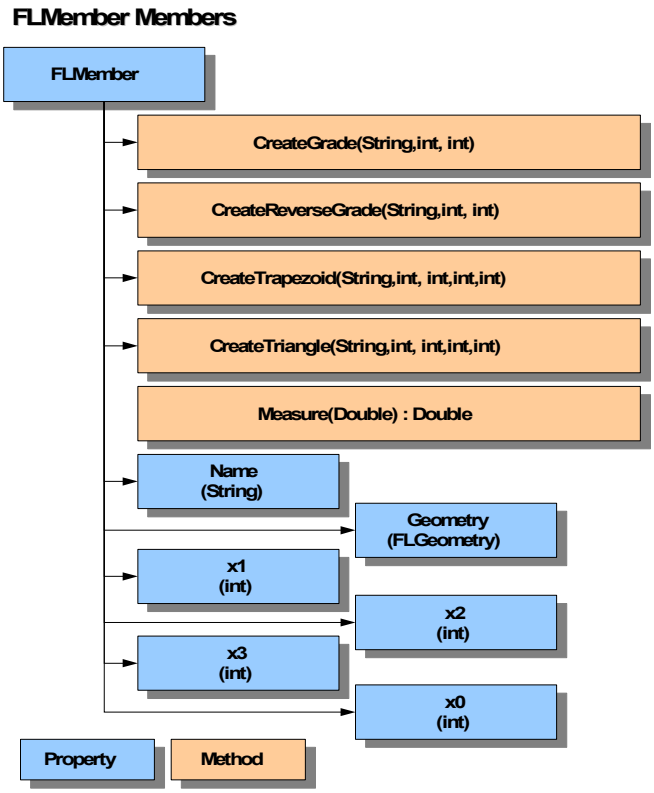
Example

FLResult oResMP = oResults.MostProbable();

FLMember Class

Class Structure

The diagram below describes the class structure:-



Description

This class define the dataset and functionality of an individual fuzzy logic member (FLMember).

FLMember.MakeGrade

Syntax

MakeGrade(sName, iX1,iX2)

Parameters:-

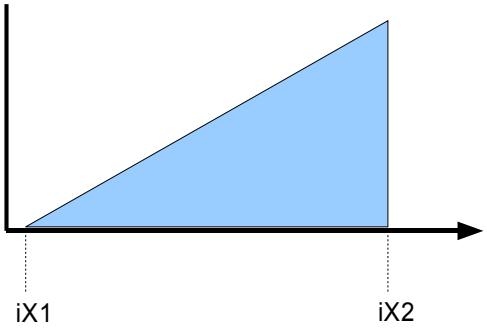
sName *– The name of the new grade member created.*

iX1 *– int– The starting point of the grade member upon the x-axis.*
iX2 *– int– The ending point of the grade member upon the x-axis.*

No Return value.

Description

This helper function configures the FLMember object to act as a fuzzy logic grade geometry defined by the iX1 and iX2 parameters. The illustration below shows a fuzzy logic grade with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oMemb.MakeGrade("A Little", 0, 35);

FLMember.MakeReverseGrade

Syntax

MakeReverseGrade(sName, iX1,iX2)

Parameters:-

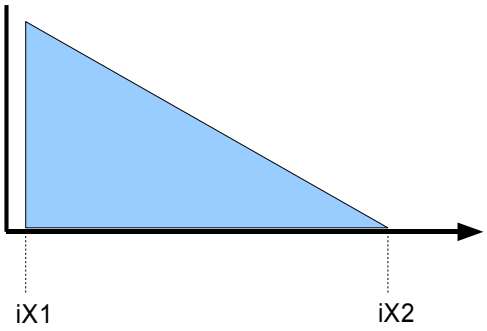
sName *– The name of the new reverse grade member created.*

iX1 *– int– The starting point of the grade member upon the x-axis.*
iX2 *– int– The ending point of the grade member upon the x-axis.*

No Return value.

Description

This helper function configures the FLMember object to act as a fuzzy logic reverse grade geometry defined by the iX1 and iX2 parameters. The illustration below shows a fuzzy logic reverse grade with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oMemb.MakeReverseGrade("A Lot", 65, 100);

FLMember.MakeTrapezoid

Syntax

MakeTrapezoid(sName, iX1,iX2,iX3,iX4)

Parameters:-

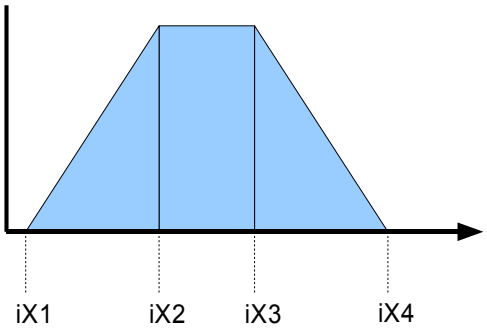
***sName** – The name of the new trapezoid member created.*

***iX1** – int– The starting point of the trapezoid member upon the x-axis.*
***iX2** – int– The starting apex of the trapezoid member upon the x-axis.*
***iX3** – int– The ending apex of the trapezoid member upon the x-axis.*
***iX4** – int– The ending point of the trapezoid member upon the x-axis.*

No Return value.

Description

This helper function configures the FLMember object to act as a fuzzy logic *trapezoid* geometry defined by the iX1, iX2, iX3 and iX4 parameters. The illustration below shows a fuzzy logic *trapezoid* with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oMemb.MakeTrapezoid("High", 35,45,55,65);

FLMember.MakeTriangle

Syntax

MakeTriangle(sName, iX1,iX2,iX3)

Parameters:-

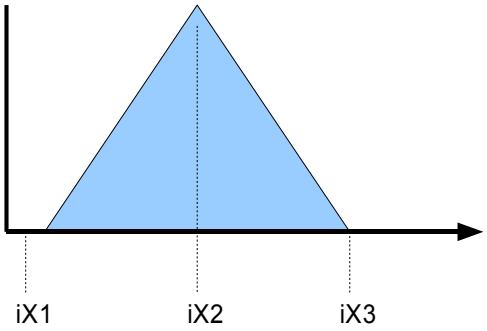
sName *– The name of the new triangle member created.*

iX1 *– int– The starting point of the triangle member upon the x-axis.*
iX2 *– int– The apex of the triangle member upon the x-axis.*
iX3 *– int– The ending point of the triangle member upon the x-axis.*

No Return value.

Description

This helper function c configures the FLMember object to act as a fuzzy logic *triangle* geometry defined by the iX1, iX2 and iX3 parameters. The illustration below shows a fuzzy logic *triangle* with the parameters indicated.



The name provide in the sName parameter is typically a plain language description of the member (e.g. Big, Heavy, Wide, Far, etc).

Example

oMemb.MakeTriangle("High",10,50,90);

FLMember.Measure

Syntax

Measure(dblData)

Parameters:-

***dblData** – Double – This provides the data to be evaluated.*

The return value is a Double.

Description

This function returns a result, of type Double, representing the fuzzy logic trueness of dblData when evaluated against this members rule (or geometry). Fuzzy Logic trueness is measures in a scale of 0.00 to 1.00 where 0 is absolutely false and 1.00 if absolutely true.

e.g.

0.00 == Absolutely false
0.25 == Mostly false
0.50 == Indifferent (neither true nor false)
0.75 == Mostly true.
1.00 == Absolutely true

Example

```
FLMember oMemb= new FLMember();
oMemb.MakeTrapeziod("High",10,20,30,40);
double dblRes=oMemb.Measure(25);
```

dblRes would contain the value 1.00 (absolute truth) because the measured value (25) sits on the trapezoids apex (between 20 and 30).

FLMember.Name

Syntax

FLMember.Name = [String]

Description

This property contain the name of the FLMember object .

Example

FLMember oMemb= new FLMember();
oMemb.MakeTrapeziod("High",10,20,30,40);
String sName=oMemb.Name;

sName would contain the string "High".

FLMember.Geometry

Syntax

FLMember.Geometry = [speculous.ai.fuzzy.FLGeometry]

Description

This property contain the geometry type (e.g. Trapezoid, Triangle, Grade, Reverse Grade) associated with this FLMember object .

Example

```
FLMember oMemb= new FLMember();
oMemb.MakeTrapeziod("High",10,20,30,40);
bool IsTriangle = (oMemb.Geometry==FLGeometry.Triangle);
```

IsTriangle would contain the false because the geomrty created was a trapezoid.

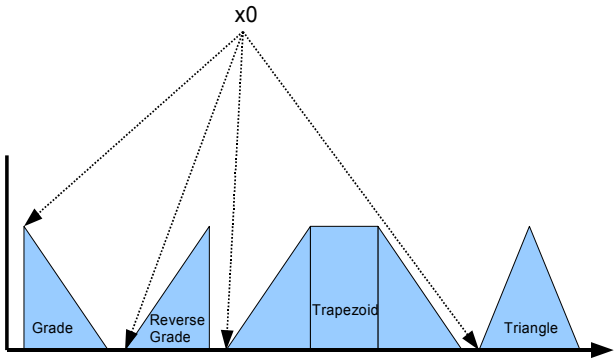
FLMember.x0

Syntax

FLMember.x0 = [double]

Description

This property defines the start point (x-axis) for the geometry type associated with this FLMember object. The diagram below illustrates the point upon each geometry type.



Example

FLMember oMemb= new FLMember();
oMemb.MakeTrapeziod("High",10,20,30,40);
double dblxPoint=oMemb.x0;

dblxPoint would contain 10.00.

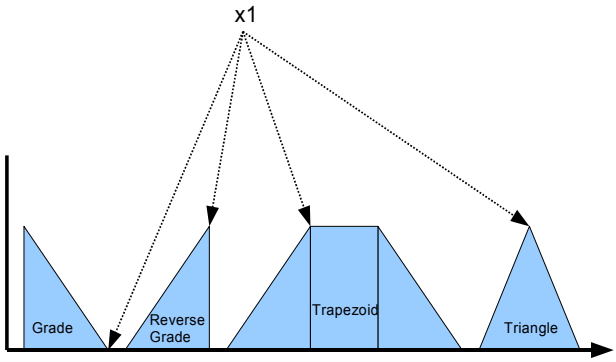
FLMember.x1

Syntax

FLMember.x1= [double]

Description

This property defines the 2nd point (x-axis) for the geometry type associated with this FLMember object. The diagram below illustrates the point upon each geometry type.



Example

FLMember oMemb= new FLMember();
oMemb.MakeTrapeziod("High",10,20,30,40);
double dblxPoint=oMemb.x1;

dblxPoint would contain 20.00.

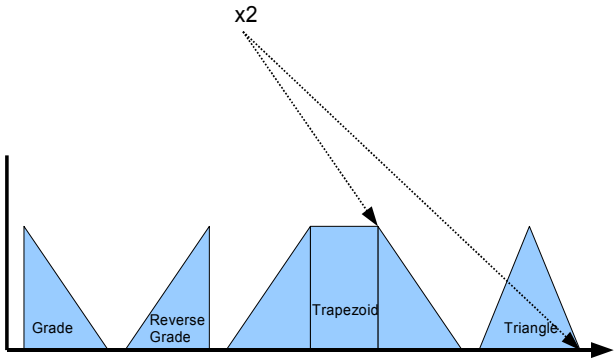
FLMember.x2

Syntax

FLMember.x2= [double]

Description

This property defines the 3nd point (x-axis) for the geometry type (Triangle and Trapezoid only) associated with this FLMember object. The diagram below illustrates the point upon each geometry type.



Example

FLMember oMemb= new FLMember();
oMemb.MakeTrapeziod("High",10,20,30,40);
double dblxPoint=oMemb.x2;

dblxPoint would contain 30.00.

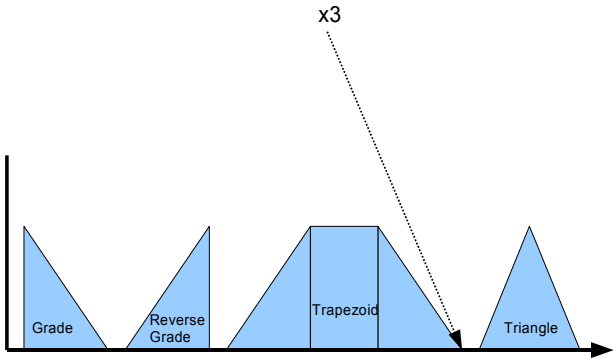
FLMember.x3

Syntax

FLMember.x3= [double]

Description

This property defines the 3nd point (x-axis) for the geometry type (Trapezoid only) associated with this FLMember object. The diagram below illustrates the point upon each geometry type.



Example

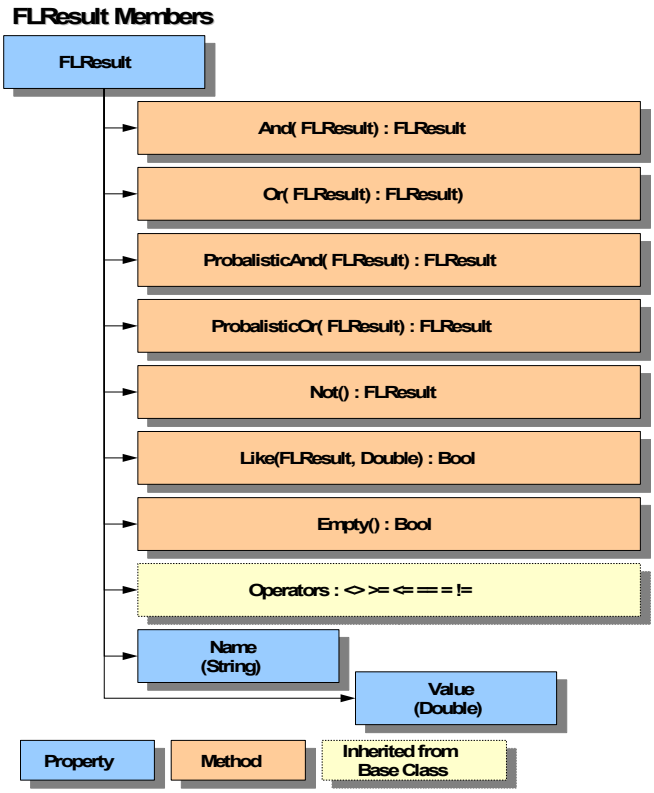
FLMember oMemb= new FLMember();
oMemb.MakeTrapeziod("High",10,20,30,40);
double dblxPoint=oMemb.x3;

dblxPoint would contain 40.00.

FLResult Class

Class Structure

The diagram below describes the class structure:-



Description

This class define the dataset and functionality of an individual fuzzy logic result (FLResult).

FLResult.Name

Syntax

FLResult.Name = [String]

Description

This property contain the name of the FLResult object .

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

String sName=oResults.Items[0].Name;
```

sName would contain the string "High".

FLResult.Value

Syntax

FLResult.Name = [double]

Description

This property contain a value representing the trueness of the result. See FLMember.Measure for details of how the value is calculated.

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

double dblRes=oResults.Items["Low"].Value;
```

dblRes would contain 0.00 (false).

FLResult Operators

Description

The FLResult class supports the use of operators between two FLResult objects.

e.g.

If (oRes1==oRes2)
{
....
}

The table below describes the operation and its application.

Operator	Syntax	Description
==	*[oRes1] == [oRes2]*	Returns true if *[oRes1]*.Value is equal to *[oRes2]*.Value
>	*[oRes1] >[oRes2]*	Returns true if *[oRes1]*.Value is greater than *[oRes2]*.Value
<	*[oRes1] < [oRes2]*	Returns true if *[oRes1]*.Value is smaller than *[oRes2]*.Value
>=	*[oRes1] >= [oRes2]*	Returns true if *[oRes1]*.Value is greater than or equal to *[oRes2]*.Value
<=	*[oRes1] <= [oRes2]*	Returns true if *[oRes1]*.Value is greater than or equal to *[oRes2]*.Value
!=	*[oRes1] != [oRes2]*	Returns true if *[oRes1]*.Value is greater than or smaller than (i.e. Not equal to) *[oRes2]*.Value

FLResult.And()

Syntax

oResult.And(FLResult)

Parameters:-

***FLResult** – FLResult – The comparator FLResult object.*

Returns a new FLResult object.

Description

This method performs a fuzzy logic AND operation between two FLResult objects (the source object and the parameter object). The AND operation is defined as a Min() function between the two object values (FLResult.Value). The resultant FLResult object provides the result of the operation. The name property of the result object reflects the two source objects and the operation performed.

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

flResult oRes=oResults.Items["High"].And(oResults.Items["Low"]);
```

oRes.Name would contain the string "High AND Low". oRes.Value would contain 0.00 (i.e. The value (lowest) from *oResults.Items["Low"]*).

FLResult.Or()

Syntax

oResult.Or(FLResult)

Parameters:-

***FLResult** – FLResult – The comparator FLResult object.*

Returns a new FLResult object.

Description

This method performs a fuzzy logic OR operation between two FLResult objects (the source object and the parameter object). The AND operation is defined as a Max() function between the two object values (FLResult.Value). The resultant FLResult object provides the result of the operation. The name property of the result object reflects the two source objects and the operation performed.

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

flResult oRes=oResults.Items["High"].Or(oResults.Items["Low"]);
```

oRes.Name would contain the string "High OR Low". oRes.Value would contain 1.00 (i.e. The value (lowest) from *oResults.Items["High"]*).

FLResult.ProbalisticAnd()

Syntax

oResult.ProbalisticAnd(FLResult)

Parameters:-

***FLResult** – FLResult – The comparator FLResult object.*

Returns a new FLResult object.

Description

This method performs a fuzzy logic PROBALISTIC AND operation between two FLResult objects (the source object and the parameter object). The AND operation is defined as the multiplication of the two object values (FLResult.Value). The resultant FLResult object provides the result of the operation. The name property of the result object reflects the two source objects and the operation performed.

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

flResult oRes=oResults.Items["High"].ProbalisticAnd(oResults.Items["Low"]);
```

oRes.Name would contain the string "PROB - High AND Low". oRes.Value would contain 0.00 (i.e. *1.00 x 0.00*).

FLResult.ProbalisticOr()

Syntax

oResult.ProbalisticOr(FLResult)

Parameters:-

***FLResult** – FLResult – The comparator FLResult object.*

Returns a new FLResult object.

Description

This method performs a fuzzy logic PROBALISTIC OR operation between two FLResult objects (the source object and the parameter object). The AND operation is defined as the subtraction of the two object values (FLResult.Value). The resultant FLResult object provides the result of the operation. The name property of the result object reflects the two source objects and the operation performed.

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

flResult oRes=oResults.Items["High"].ProbalisticOr(oResults.Items["Low"]);
```

oRes.Name would contain the string "PROB - High AND Low". oRes.Value would contain 1.00 (i.e. *1.00 - 0.00*).

FLResult.Not()

Syntax

oResult.Not()

Returns a double value.

Description

This method performs a fuzzy logic NOT upon the FLResult object. The calculation for NOT is 1.00-FLResult.Value.

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

double dblRes=oResults.Items["High"].Not();
```

The value of dblRes would be 0 (i.e. 1.00 – 1.00).

FLResult.Empty()

Syntax

oResult.Empty()

Returns a boolean value.

Description

This method performs returns true if the value of the result is empty (i.e. 0 or absolute false).

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

flResult oRes=oResults.Items["High"].ProbalisticOr(oResults.Items["Low"]);
```

oResults.Items["High"].Empty would return false.
oResults.Items["Low"].Empty would return true.

FLResult.Like(FLRes)

Syntax

oResult.Like(FLResult, dblVariance)

Parameters:-

***FLResult** – FLResult – The comparator FLResult object.*
***dblVariance** – Double – This provides tolerance (+/-) allowed.*

Returns a new FLResult object.

Description

This method return true if the source FLResult object is "like" the parameter FLResult Object. Like is defines as the name property of each being identical and the values being the same within a tollerance (plus/minus the variance).

Example

```
FLMembers oMembers= new FLMembers();
FLResults oResults = new FLResults();

oMembers.CreateTrapeziod("High",10,20,30,40);
oMembers.CreateTriangle("Low",0,20);

oResults.CreateResults(20,oMembers);

bool IsLike=oResults.Items["High"].Like(oResults.Items["Low"],1.00);
```

IsLike would contain the value false, as although the value are within the tolerance of each other the names are different.

Required Files/Assemblies

Outside of the .NET 3.5 framework there are no other libraries required.