

Plezuro scripting language Documentation

P l e z u r o

1 Authors

University : Silesian University of Technology
Faculty : Faculty of Applied Mathematics
Academic year : 2013/2014
Path : Computer Science
Semester : IV
Names

- Piotr Sroczkowski
Idea, scripting language, IDE, documentation, almost all
- Daniel Mikulski
Tests, numerical integration

2 Technical information

Language : c# 5.0
Platform : Mono 3.2.8
Compiler : gmcs 3.2.8.0
Version control system : git 1.9.1
Public repository address : <https://github.com/oprogramador/repo>
Licence : GNU GPL 2.0

3 User interface specification

3.1 Short description

A scripting language has been implemented. On its base a non-relational database works.

3.2 Short tutorial

3.2.1 Simple example

```
$i=2;
$n=0;
:: while( {n<1000},
{
    $k = 2;
    $ispr = true;
    :: while( {k*k<=i},
    {
        :: if( i%k==0, { ispr=false }, {0});
        k=k+1
    });
    :: if( ispr , {n=n+1}, {0});
    i=i+1
});
i-1
```

3.2.2 Comments

```
//this is a comment

/*
```

```
Another comment
*/
```

3.2.3 Variables

At declaring before the variable name you should write '\$', it determines the variable scope.

```
$a = 12;
a++;
$b = a*2;
a+b^3
```

3.2.4 Cloning vs reference

```
$a = 21;
$b = a;
b++;
b.::println(); //it prints '22'
(a==b).::println(); //it prints 'true'

$c = 4;
$d := c;
d++;
c.::println(); //it prints '4'
(c==d).::println(); //it prints 'false'

$e = 2;
$f = e.::clone();
f++;
e.::println(); //it prints '2'
(e==f).::println(); //it prints 'false'

(1==1).::println(); //it prints 'true'
(1===1).::println(); //it prints 'false'
```

null

3.2.5 Built-in classes (types)

```
//number
$x = 2.3e45;
$y = 0xff; //hexadecimal
$z = 072; //octal
$a = 0b11011; //binary

//string
$b = 'aaaaaaaaaaaaaaaaaaaaa';
$c = "wfefwfwf";

//list
$d = [1,2,3,4];
$e = ::array(1,9,3);

//dictionary
$f = ::dic(1,2,3,4);
```

```

//set
$g = :: set (3,4,5);

//error
$h = 1/0;

//class
$i = 1:: class ();

//package
$j = i:: package ();

//pair
$k = 3:4;

//procedure
$l = {1+2}

(x,y,z,a,b,c,d,e,f,g,h,i,j,k,l)

```

3.2.6 Tuples

```

($a, $b) = (1,3);
a:: printl ();
b:: printl ();

$c = 5;
$d = 6;
(a,b,c,d) = (b,c,d,a);

a:: printl ();
(a,b,c,d):: printl ();

a <-> c;
(a,b,c,d)

```

3.2.7 Conditional expressions

```

$x = 2;
:: if( x<0, {x++}, {x--});

$a = x>0 ? 'yes' : 'no';
$b = (x>0):: if({ 'yes' }, { 'no' });
$c = :: if(x>0, { 'yes' }, { 'no' });

a,b,c //it prints ("yes","yes","yes")

```

3.2.8 Loops

```

$i = 0;
:: while({i<20},{
    i:: printl ();
    i++
});

```

```
[1,2,3,4,5]::each({ args::println() });
::range(30,70,6.5)::each({ args::println() })
```

3.2.9 Procedures

```
$f = {
    ($x,$y,$z) = args;
    ::println ( 'args='+args );
    ::println ( 'x='+x );
    ::println ( 'y='+y );
    ::println ( 'z='+z );
    x+y*z };

f::applyF([2,3,4])::println();
f::time()::println(); //executing time in milliseconds; x,y,z are undefined here
{f::applyF([2,3,4])}::time()::println() //executed time; x,y,z are defined
```

3.2.10 File operations

```
$txt = 'abc.txt'::fromF();

'xyz.txt'::toF($txt*20)
```

3.2.11 Html table generation

```
::toF('1.html', [([1,2,3]),[4,5]]::html());
::toF('2.html', [::dic('name','Jean', 'city', 'Marseille'), ::dic('name','Tom', 'city','Mi
```

3.2.12 Operator precedence (from those executed at the end)

```
;
:=
=
,
<->
<<
>>
?
|
&
<=>
>=
>
<=
<
!=
==
===
=~
```

+
 -
 %
 *
 /
 ^
 Together
 • ^^
 • =~
 ..
 :

3.2.13 Built-in packages, classes, methods, operators and constants

- package Lang
 - class Boolean
 - Extends: [Object]
 - * Short description : Boolean value
 - * Operators:
 - ?
 - Arguments: (Boolean b, Pair p)
 - Returned type: Object
 - Short description : If b is *true*, it returns the first value of pair p, in other case it returns the second value.
 - |
 - Arguments: (Boolean a, Boolean b)
 - Returned type: Boolean
 - Short description : Logic alternative
 - &
 - Arguments: (Boolean a, Boolean b)
 - Returned type: Boolean
 - Short description : Logical conjunction
 - !
 - Arguments: (Boolean b)
 - Returned type: Boolean Short description : Logical negation
 - * Methods:
 - if
 - Arguments: (Boolean b, Procedure t, Procedure f)
 - Returned type: Object
 - Short description : Conditional instruction - if b is *true*, the procedure t is executed, otherwise the procedure f is executed.
 - * Constants:
 - true
 - Short description : True
 - false
 - Short description : False
 - class Class
 - Extends: [Object]

- * Short description : Class
- * Methods:
 - parents
 - Arguments: (Class c)
 - Returned type: List
 - Short description : It returns all base classes (there is multiple inheritance).
 - package
 - Arguments: (Class c)
 - Returned type: Package
 - Short description : It returns the package that the class belongs to.
- class Dictionary
 - Extends: [Object]
 - * Short description : Dictionary container
 - * Operators:
 - <<
 - Arguments: (Dictionary d, Pair p)
 - Returned type: Dictionary
 - Short description : It adds a pair key-value to the dictionary.
 - * Methods:
 - ref
 - Arguments: (Dictionary d, Object key)
 - Returned type: Object
 - Short description : It returns the reference to the value indicated by the key.
 - len
 - Arguments: (Dictionary d)
 - Returned type: Number
 - Short description : It returns the length of the dictionary.
 - contains
 - Arguments: (Dictionary d, Object key)
 - Returned type: Boolean
 - Short description : Information whether the dictionary contains the key.
 - keys
 - Arguments: (Dictionary d)
 - Returned type: List
 - Short description : It returns the list of all the keys.
 - remove
 - Arguments: (Dictionary d, Object key)
 - Returned type: Dictionary
 - Short description : It returns the new dictionary with the removed key.
- class DotFunc
 - Extends: [Object]
 - * Short description : Pair (function, first argument)
 - * Operators:
 - ^^
 - Arguments: (DotFunc d, Object o)
 - Returned type: Object
 - Short description : It calls the function with the arguments. The first argument is stored, the next ones are contained inside object o (Empty class object is treated as no arguments, Tuple as multiple arguments, other classes as single argument).
- class Empty
 - Extends: [Object]

- * Short description : Empty value
- * Methods:
 - array
 - Arguments: (Empty e)
 - Returned type: List
 - Short description : It returns an empty list.
- class Error
 - Extends: [Object]
 - * Short description : Error
 - * Methods:
 - msg
 - Arguments: (Error e)
 - Returned type: String
 - Short description : It returns the error message.
- class List
 - Extends: [Object]
 - * Short description : List collection
 - * Operators:
 - <<
 - Arguments: (List l, Object o)
 - Returned type: List
 - Short description : Pushing o object to l list.
 - >>
 - Arguments: (List l, Reference r)
 - Returned type: List
 - Short description : Popping an object from l list to r reference.
 - +
 - Arguments: (List a, List b)
 - Returned type: List
 - Short description : Two lists concatenation.
 - *
 - Arguments: (List l, Number n)
 - Returned type: Object
 - Short description : n-times copying of l list.
 - * Methods:
 - get
 - Arguments: (List l, Number n)
 - Returned type: Object
 - Short description : It returns n-th element of l list.
 - len
 - Arguments: (List l)
 - Returned type: Number
 - Short description : It returns l list length.
 - ref
 - Arguments: (List l, Number n)
 - Returned type: Reference
 - Short description : It returns the reference to n-th element of l list.
 - each
 - Arguments: (List l, Procedure p)
 - Returned type: Object
 - Short description : Iteration of l list, executing of p procedure for each element.

- where
Arguments: (List l, Procedure p)
Returned type: List
Short description : Selection of such elements that procedure p returns *true*.
- map
Arguments: (List l, Procedure p)
Returned type: List
Short description : Mapping of p procedure throw l list.
- sort
Arguments: (List l)
Returned type: List
Short description : Sorting.
- orderBy
Arguments: (List l, Procedure p)
Returned type: List
Short description : Sorting by the value that p procedure returns.
- orderByD
Arguments: (List l, Procedure p)
Returned type: List
Short description : The same as orderBy but descending.
- groupBy
Arguments: (List l, Procedure p)
Returned type: List
Short description : Grouping by the value that p procedure returns.
- reverse
Arguments: (List l)
Returned type: List
Short description : List reversing.
- max
Arguments: (List l)
Returned type: Object
Short description : It returns the max value.
- min
Arguments: (List l)
Returned type: Object
Short description : It returns the min value.
- median
Arguments: (List l)
Returned type: Object
Short description : It returns the median.
- remove
Arguments: (List l, Number n)
Returned type: List
Short description : It returns the list with removed element at n index.
- toSet
Arguments: (List l)
Returned type: Set
Short description : It converts to the set collection.
- html
Arguments: (List l)
Returned type: String
Short description : It returns an html table.

- class NullClass
 - Extends: [Object]
 - * Short description : Null value
 - * Constants:
 - null
 - Short description : Null
- class Number
 - Extends: [Object]
 - * Short description : Real number
 - * Operators:
 - +
 - Arguments: (Number a, Number b)
 - Returned type: Number
 - Short description : Addition.
 - -
 - Arguments: (Number a, Number b)
 - Returned type: Number
 - Short description : Subtraction.
 - *
 - Arguments: (Number a, Number b)
 - Returned type: Number
 - Short description : Multiplication.
 - /
 - Arguments: (Number a, Number b)
 - Returned type: Number
 - Short description : Division.
 - ^
 - Arguments: (Number a, Number b)
 - Returned type: Number
 - Short description : Power.
 - ++
 - Arguments: (Number a)
 - Returned type: Number
 - Short description : Incrementation.
 - --
 - Arguments: (Number a)
 - Returned type: Number
 - Short description : Decrementation.
 - * Methods:
 - chr
 - Arguments: (Number n)
 - Returned type: String
 - Short description : It returns the character with ASCII code n.
 - sin
 - Arguments: (Number n)
 - Returned type: Number
 - Short description : Sine.
 - cos
 - Arguments: (Number n)
 - Returned type: Number
 - Short description : Cosine.

- `tan`
Arguments: (Number n)
Returned type: Number
Short description : Tangent.
- `asin`
Arguments: (Number n)
Returned type: Number
Short description : Arcsine.
- `acos`
Arguments: (Number n)
Returned type: Number
Short description : Arccosine.
- `atan`
Arguments: (Number n)
Returned type: Number
Short description : Arctangent.
- `sinh`
Arguments: (Number n)
Returned type: Number
Short description : Hyperbolic sine.
- `cosh`
Arguments: (Number n)
Returned type: Number
Short description : Hyperbolic cosine.
- `tanh`
Arguments: (Number n)
Returned type: Number
Short description : Hyperbolic tangent.
- `round`
Arguments: (Number n)
Returned type: Number
Short description : Rounding.
- `floor`
Arguments: (Number n)
Returned type: Number
Short description : Flooring.
- `ceil`
Arguments: (Number n)
Returned type: Number
Short description : Ceiling.
- `abs`
Arguments: (Number n)
Returned type: Number
Short description : Absolut value.
- `ln`
Arguments: (Number n)
Returned type: Number
Short description : Natural logarithm.
- `sqrt`
Arguments: (Number n)
Returned type: Number
Short description : Square root.

- fib
 - Arguments: (Number n)
 - Returned type: Number
 - Short description : N-th element of the Fibonacci sequence.
- * Constants:
 - pi
 - Short description : Pi number
 - e
 - Short description : E number
- class Object
 - Extends: []
 - * Short description : Any object
 - * Operators:
 - .
 - Arguments: (Object a, SoftLink s)
 - Returned type: DotFunc
 - Short description : DotFunc creation.
 - ;
 - Arguments: (Object a, Object b)
 - Returned type: Object
 - Short description : It returns b object.
 - ,
 - Arguments: (Object a, Object b)
 - Returned type: Object
 - Short description : Tuple creation.
 - ;
 - Arguments: (Reference a, Reference b, Reference c)
 - Returned type: Number
 - Short description : Swapping a and b.
 - :
 - Arguments: (Object a, Object b)
 - Returned type: Pair
 - Short description : Pair creation.
 - <=>
 - Arguments: (Object a, Object b)
 - Returned type: Number
 - Short description : It returns 1 if a is greater than b, 0 if equal, -1 if less.
 - >=
 - Arguments: (Object a, Object b)
 - Returned type: Boolean
 - Short description : It informs whether a is greater or equal to b.
 - >
 - Arguments: (Object a, Object b)
 - Returned type: Boolean
 - Short description : It informs whether a is greater then b.
 - <=
 - Arguments: (Object a, Object b)
 - Returned type: Boolean
 - Short description : It informs whether a is less or equal to b.
 - <
 - Arguments: (Object a, Object b)

- Returned type: Boolean
- Short description : It informs whether a is less than b.
- !=
 - Arguments: (Object a, Object b)
 - Returned type: Boolean
 - Short description : It informs whether a is not equal to b.
- ==
 - Arguments: (Object a, Object b)
 - Returned type: Boolean
 - Short description : It informs whether a equal to b.
- ===
 - Arguments: (Object a, Object b)
 - Returned type: Boolean
 - Short description : It informs whether a is b (the same object).
- &&
 - Arguments: (Reference r)
 - Returned type: Pointer
 - Short description : It returns the pointer to r.
- :=
 - Arguments: (Object a, Object b)
 - Returned type: Object
 - Short description : Cloning b into a, you can clone all the tuples.
- =
 - Arguments: (Object a, Object b)
 - Returned type: Boolean
 - Short description : Ascharactering b to a (reference, you can ascharacter all the tuples).
- * Methods:
 - class
 - Arguments: (Object o)
 - Returned type: Class
 - Short description : It returns the class of o object.
 - print
 - Arguments: (Object o)
 - Returned type: Object
 - Short description : Printing o to the console.
 - println
 - Arguments: (Object o)
 - Returned type: Object
 - Short description : Printing the o to the console as the new line.
 - clone
 - Arguments: (Object o)
 - Returned type: Object
 - Short description : Cloning.
 - lent
 - Arguments: (Object o)
 - Returned type: Number
 - Short description : It returns the length of o (for Tuple object length of the tuple, for Empty object 0, for other classes objects 1.
 - set
 - Arguments: (Object o)
 - Returned type: Set
 - Short description : Set creation..

- dic
 - Arguments: (Object o)
 - Returned type: Dictionary
 - Short description : Dictionary creation.
- class Package
 - Extends: [Object]
 - * Short description : Package (collection of classes and other packages)
 - * Operators:
 - * Methods:
 - package
 - Arguments: (Package p)
 - Returned type: Package
 - Short description : It returns the parent package.
 - * Constants:
 - true
 - Short description : True
 - false
 - Short description : False
- class Pair
 - Extends: [Object]
 - * Short description : Ordered pair (key, value)
 - * Methods:
 - key
 - Arguments: (Pair p)
 - Returned type: Object
 - Short description : It returns the key.
 - value
 - Arguments: (Pair p)
 - Returned type: Object
 - Short description : It returns the value.
- class Pointer
 - Extends: [Object]
 - * Short description : Pointer to an object
 - * Operators:
 - **
 - Arguments: (Pointer p)
 - Returned type: Object
 - Short description : It returns the object that p pointer points to.
- class Procedure
 - Extends: [Object]
 - * Short description : Procedure that gives parameters and returns a value
 - * Methods:
 - apply
 - Arguments: (Procedure p)
 - Returned type: Object
 - Short description : Calling procedure without parameters.
 - applyF
 - Arguments: (Procedure p, List l)
 - Returned type: Object
 - Short description : Calling procedure with parameters.

- while
 - Arguments: (Procedure a, Procedure b)
 - Returned type: Object
 - Short description : *while* loop, *a* procedure determines the condition, *b* procedure is executed inside.
- integral
 - Arguments: (Procedure p, Number beg, Number end)
 - Returned type: Object
 - Short description : Numerical integral.
- time
 - Arguments: (Procedure p)
 - Returned type: Number
 - Short description : It counts p procedure executing time in milliseconds.
- class Reference
 - Extends: [Object]
 - * Short description : Reference to an object, an additional class, each object has a reference but no object is an instance of the Reference class.
- class Set
 - Extends: [Object]
 - * Short description : Set collection
 - * Operators:
 - <<
 - Arguments: (Set s, Object o)
 - Returned type: Object
 - Short description : Pushing o object to s set.
 - * Methods:
 - len
 - Arguments: (Set s)
 - Returned type: Object
 - Short description : It returns the set length.
 - max
 - Arguments: (Set s)
 - Returned type: Object
 - Short description : It returns the max value.
 - min
 - Arguments: (Set s)
 - Returned type: Object
 - Short description : It returns the min value.
 - contains
 - Arguments: (Set s, Object o)
 - Returned type: Boolean
 - Short description : It informs whether the set contains the value.
 - join
 - Arguments: (Set a, Set b)
 - Returned type: Set
 - Short description : Set intersection.
 - except
 - Arguments: (Set a, Set b)
 - Returned type: Set
 - Short description : Set complement.

- union
 - Arguments: (Set a, Set b)
 - Returned type: Set
 - Short description : Set union.
- remove
 - Arguments: (Set s, Object o)
 - Returned type: Object
 - Short description : It returns the set with removed value.
- toList
 - Arguments: (Set s)
 - Returned type: Object
 - Short description : Conversion to list.
- len
 - Arguments: (Set s)
 - Returned type: Object
 - Short description : It returns the set length.
- class SoftLink
 - Extends: [Object]
 - * Short description : Soft link
 - * Operators:
 - ^^
 - Arguments: (SoftLink s, Object o)
 - Returned type: Object
 - Short description : Execution of the procedure pointer by the link for the arguments.
- class String
 - Extends: [Object]
 - * Short description : Text string
 - * Operators:
 - +
 - Arguments: (String s, Object o)
 - Returned type: String
 - Short description : Concatenation.
 - *
 - Arguments: (String s, Number n)
 - Returned type: String
 - Short description : N-times copying.
 - #
 - Arguments: (String s)
 - Returned type: Object
 - Short description : Inserting of calculated values inside the string.
 - =~
 - Arguments: (String regex, String s)
 - Returned type: Boolean
 - Short description : It informs whether s string matches regex Regex.
 - * Methods:
 - len
 - Arguments: (String s)
 - Returned type: Number
 - Short description : It returns the string length.
 - get
 - Arguments: (String s, Number n)

- Returned type: String
- Short description : It returns the n-th character.
- reverse
 - Arguments: (String s)
 - Returned type: String
 - Short description : It returns the reversed string.
- ord
 - Arguments: (String s)
 - Returned type: Number
 - Short description : It returns the ASCII code of the first character.
- fromF
 - Arguments: (String s)
 - Returned type: String
 - Short description : It reads the file content into string.
- toF
 - Arguments: (String s, String f)
 - Returned type: Boolean
 - Short description : It writes s string to f file, the returned value informs about the success.
- put
 - Arguments: (String f, String s)
 - Returned type: Boolean
 - Short description : It writes s string to f file, the returned value informs about the success.
- putA
 - Arguments: (String f, String s)
 - Returned type: Boolean
 - Short description : It appends s string to f file, the returned value informs about the success.
- append
 - Arguments: (String s, String f)
 - Returned type: Boolean
 - Short description : It appends s string to f file, the returned value informs about the success.
- load
 - Arguments: (String s)
 - Returned type: Object
 - Short description : It executes the module written in a file.
- eval
 - Arguments: (String s)
 - Returned type: Object
 - Short description : It returns the code inside a string.
- class Tuple
 - Extends: [Object]
 - * Short description : Tuple collection, each tuple contains minimum 2 elements.

4 Developer specification

4.1 How to download, compile and run?

1. Install any distribution of GNU/Linux operating system (next instructions for Debian derivatives).
You can use : <http://www.linuxmint.com/download.php>.

2. Install mono. Use terminal command : *sudo apt-get install monodevelop mono-complete.*
3. Install git, although it should be built-in in your distribution : *sudo apt-get install git*
4. Create a new directory and go inside it : *mkdir project1; cd project1*
5. Download the project : *git download https://github.com/oprogramador/repo.git; cd repo*
6. Compile : *./make.sh*
7. Go one directory level up : *cd ..*
8. Run : *./calc.exe*

You can also try compiling it on Windows using either Visual Studio or Mono.

4.2 Code

4.2.1 Files, namespaces (adequate to directories), classes, interfaces, enumerations, inheritance

```

Gui/IOPanel.cs: class IOPanel : Panel
Gui/IClickable.cs: public interface IClickable
Gui/InputBox.cs: class InputBox : IOBox, ITexttable
Gui/MyItem.cs: public class MyItem
Gui/MyMenu.cs: public class MyMenu : MainMenu
Gui/OutputBox.cs: class OutputBox : IOBox, IOutputtable
Gui/FormAdapter.cs: public class FormAdapter
Gui/IOBox.cs: class IOBox : RichTextBox
Gui/MainPanel.cs: class MainPanel : Panel
Gui/Clickable.cs: public interface Clickable
Gui/MainWindow.cs: class MainWindow : Form
Gui/VisualSyntax.cs: class VisualSyntax
MyTypes/IStepable.cs: interface IStepable : IVariable
MyTypes/CircularInheritanceException.cs: class CircularInheritanceException : Exception
MyTypes/ITuplable.cs: public interface ITuplable
MyTypes/NotComparableException.cs: class NotComparableException : Exception
MyTypes/InfinityException.cs: class InfinityException : NumberException
MyTypes/IVariable.cs: public interface IVariable : ICompCloneable, IStringable, ITuplable
MyTypes/LambdaConverter.cs: static class LambdaConverter
MyTypes/IStringable.cs: public interface IStringable
MyTypes/IItem.cs: public interface IItem : IVariable
MyTypes/NoMethodException.cs: class NoMethodException : Exception
MyTypes/AccessModifier.cs: public enum AccessEnum
MyTypes/AccessModifier.cs: public class AccessModifier
MyTypes/ModuleNotFoundException.cs: class ModuleNotFoundException : Exception
MyTypes/NaNException.cs: class NaNException : NumberException
MyTypes/VariableFactory.cs: class VariableFactory
MyTypes/NumberException.cs: class NumberException : Exception
MyTypes/UndefinedException.cs: class UndefinedException : Exception
MyTypes/MyClasses/ObjectT.cs: class ObjectT : IVariable
MyTypes/MyClasses/BuiltinClass.cs: class BuiltinClass : ClassT
MyTypes/MyClasses/CallFunc.cs: class CallFunc : IVariable
MyTypes/MyClasses/PairT.cs: public class PairT : IVariable
MyTypes/MyClasses/RangeT.cs: class RangeT : IVariable, IEnumerable
MyTypes/MyClasses/ClassT.cs: public class ClassT : IItem, IVariable, ICallable
MyTypes/MyClasses/SoftLink.cs: class SoftLink : Pointer<string>, IVariable

```

```

MyTypes/MyClasses/BuiltinFunc.cs: class BuiltinFunc : IVariable, ICallable
MyTypes/MyClasses/ErrorT.cs: class ErrorT : IVariable
MyTypes/MyClasses/StopPoint.cs: class StopPoint : IVariable
MyTypes/MyClasses/ProcedureT.cs: public class ProcedureT : OStack, ICallable
MyTypes/MyClasses/BooleanT.cs: class BooleanT : Pointer<bool>, IVariable
MyTypes/MyClasses/ListT.cs: public class ListT : SList<ICompCloneable>, IVariable
MyTypes/MyClasses/TupleT.cs: public class TupleT : SList<IVariable>, IVariable
MyTypes/MyClasses/MyClass.cs: class MyClass : ClassT
MyTypes/MyClasses/PointerT.cs: class PointerT : Pointer<ReferenceT>, IVariable
MyTypes/MyClasses/SetT.cs: class SetT : SortedSet<IVariable>, IVariable
MyTypes/MyClasses/Method.cs: public class Method : IVariable, ICallable
MyTypes/MyClasses/Callable.cs: class Callable
MyTypes/MyClasses/NullType.cs: class NullType : IVariable
MyTypes/MyClasses/StringT.cs: class StringT : Pointer<string>, IVariable
MyTypes/MyClasses/StringT.cs: class MiniParser : IParseable
MyTypes/MyClasses/Number.cs: class Number : Pointer<double>, IVariable
MyTypes/MyClasses/MyObject.cs: class MyObject : IVariable
MyTypes/MyClasses/PackageT.cs: public class PackageT : List<IItem>, IItem, IVariable
MyTypes/MyClasses/DictionaryT.cs: public class DictionaryT : SortedDictionary<IVariable, IVariable>, IVariable
MyTypes/MyClasses/ReferenceT.cs: public class ReferenceT : Pointer<IVariable>, IVariable, ITypeConvertible
MyTypes/MyClasses/DotFunc.cs: class DotFunc : IVariable, ICallable
MyTypes/MyClasses/EmptyT.cs: class EmptyT : IVariable
MyTypes/ICallable.cs: public interface ICallable : IComparable
MyTypes/ObjectContainer.cs: class ObjectContainer : List<IVariable>
MyTypes/Variable.cs: static class Variable
Maths/NumberCalcul.cs: static class NumberCalcul
Engine/IOutputable.cs: interface IOutputable : ITexttable
Engine/Tokenizer.cs: class Tokenizer
Engine/SymbolMap.cs: class SymbolMap : ConcurrentDictionary<string, object>
Engine/Engine.cs: class Engine
Engine/ErrorText.cs: class ErrorText
Engine/StaticParser.cs: static class StaticParser
Engine/RPNTypes.cs: enum RPNTypes
Engine/SyntaxException.cs: class SyntaxException : Exception
Engine/RPN.cs: class RPN
Engine/Evaluator.cs: class Evaluator : IPrintable
Engine/IOMap.cs: class IOMap : Dictionary<ITexttable, IOutputable>, IRefreshable
Engine/TokenConverter.cs: class TokenConverter
Engine/SymbolException.cs: class SymbolException : Exception
Engine/Parser.cs: class Parser
Program.cs: class Program
lib/HtmlTable.cs: abstract class HtmlTable
lib/SimpleTypeConverter.cs: static class SimpleTypeConverter
lib/HtmlArrayTable.cs: class HtmlArrayTable : HtmlTable
lib/ITypeConvertible.cs: interface ITypeConvertible
lib/Integral.cs: class Integral
lib/Co.cs: class Co
lib/HtmlDicTable.cs: class HtmlDicTable : HtmlTable
lib/HtmlTableFactory.cs: static class HtmlTableFactory
DataFixtures/DataFixtures.cs: class DataFixtures : SetT
MyCollections/ICompCloneable.cs: public interface ICompCloneable : IComparable, ICloneable
MyCollections/Pointer.cs: public class Pointer<T>
MyCollections/IEvalable.cs: public interface IEvalable
MyCollections/TokenTypes.cs: public enum TokenTypes

```

```

MyCollections/TokenTypes.cs: public static class TokenTypesExtension
MyCollections/Token.cs: public class Token
MyCollections/ITextable.cs: public interface ITexttable
MyCollections/TypeTrans.cs: public static class TypeTrans
MyCollections/WStack.cs: public class WStack<T> : Stack<T>, IVariable
MyCollections/IValuable.cs: interface IValuable
MyCollections/IParseable.cs: interface IParseable
MyCollections/SList.cs: public class SList<T> : CList<T>, ICompCloneable where T : ICompCloneable
MyCollections/SortedSet.cs: class SortedSet<T> : SortedDictionary<T,int>
MyCollections/General.cs: static class General
MyCollections/EmptyArgException.cs: class EmptyArgException : Exception
MyCollections/CList.cs: public class CList<T> : WList<T> where T: IComparable
MyCollections/WList.cs: public class WList<T> : List<T>
MyCollections/NullArg.cs: class NullArg
MyCollections/ConcurrentDictionary.cs: public class ConcurrentDictionary<TKey, TValue> : Dictionary<TKey, TValue>
MyCollections/OStack.cs: public class OStack : WStack<object>
MyCollections/IPrintable.cs: public interface IPrintable : IEvalable, IVariable
MyCollections/DefaultType.cs: class DefaultType
Info/Help.cs: class Help
Info/Info.cs: class Info
Controller/Controller.cs: class Controller
Controller/IRefreshable.cs: interface IRefreshable
Tests/TestUnit.cs: class TestUnit

```