Plezuro scripting language Documentation

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
  Some scripts

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;
{n<first}.while({
        $k := 2;
        $ispr = true;
        {k*k<=i}.while({
                {i%k==0}.if({ispr=false});
                k++
        });
        {ispr}.if({n++});
        i++
});
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.::printl(); //it prints '22'
(a===b).::printl(); //it prints 'true'

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

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

(1==1).::printl(); //it prints 'true'
(1===1).::printl(); //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 = 'aaaaaaaaaaaaaaaaa';
$c = "wfefwfwf";
$cc = '''xxx
yyy

zzz''';

//list
$d = [1,2,3,4];

//dictionary
$f = #[1,2,3,4];
```

```
//set
$g = $[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,cc,d,e,f,g,h,i,j,k,l)
```

3.2.6 Indexing

```
'abcdefghijklmnopqr'[([1..5,0,0,2..12..3])].toS.printl;
[12,13,14,15,16][1..4]
```

3.2.7 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)
```

3.2.8 Conditional expressions

```
$x = 90;
$y = true;
{x>0}.if{
        'aa'.printl
}.if({x<40},{
        'bb'.printl
}).elif({y},{
        'cc'.printl
}).else{
        'dd'.printl
}
```

3.2.9 Loops

```
$i = 13;
{i<20}.while({
        i.printl;
        i++
});

[1,2,3,4,5].each({ this.printl });

[108,2,3,4,5,20,90].each({ args.printl });

30..70..6.5.each({ args.printl })
```

3.2.10 Procedures

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

f.::applyF([2,3,4]).::printl();
f(4,5,6).::printl();
f.::time().::printl(); //executing time in milliseconds; x,y,z are undefined here
{f(4,5,6)}.::time().::printl(); //executed time; x,y,z are defined
```

3.2.11 File operations

```
$txt = 'abc.txt'.fromF;
{txt.class==String}.if({ (txt*4).toF('xyz.txt') })
```

3.2.12 Html table generation

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

3.2.13 User-defined classes

```
Lang
<< 'Person'.::newClass($[Object], #[
        'init',{
                @this << ('age': (vals*2))
        },
        '+',{
                (@this['age'] += vals)
        },
        "get-age",{
                @this['age']
        },
        "set-age",{
                @this['age'] = vals
        },
        "str",{
```

```
                "I'm␣"+(@this['age'])+'␣years␣old.'
        },
        "destroy",{
                ::printl('person␣destroy');
        }
]);
$Per = $[@Lang['Person']];
Lang << 'Dog'.::newClass( Per, #[
        'init',{
                ($age, $race) = vals;
                @(@Lang['Person'])['init'](this, age);
                @this << ('race': race);
        }
]);

$p = (@Lang['Person'](14));
@p['age']++;
p+50;
$d = (@Lang['Dog'](13,'Akbash'));
d+3;
(''+d).::printl();
//((@Lang['Person']).::set('age'))(d,100);
@p,@d
```

3.2.14 Two argument operators precedence (from those executed at the end)

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

*

/

^

Together

- ^^
- =~

..

:

3.2.15 One argument operators

!

&&

**

#

++

--

@

3.2.16 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 singe 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.
  - printl
    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 (following instructions for Debian derivatives). You can use : `http://www.linuxmint.com/download.php`.
2. Install mono. Use terminal commmand : *sudo apt-get install monodevelop mono-complete.*
3. Install git : *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. Run : *./plezuro.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

```
lib/Co.cs: class Co
lib/HtmlArrayTable.cs: class HtmlArrayTable : HtmlTable
lib/ITypeConvertible.cs: interface ITypeConvertible
lib/HtmlTableFactory.cs: static class HtmlTableFactory
lib/MyCrypto.cs: class MyCrypto
lib/HtmlDicTable.cs: class HtmlDicTable : HtmlTable
lib/Integral.cs: class Integral
lib/SimpleTypeConverter.cs: static class SimpleTypeConverter
lib/HtmlTable.cs: abstract class HtmlTable
lib/SString.cs: class SString
Engine/StaticParser.cs: static class StaticParser
Engine/Parser.cs: class Parser
Engine/Evaluator.cs: class Evaluator : IPrintable
Engine/IOutputable.cs: interface IOutputable : ITextable
Engine/RPN.cs: class RPN
Engine/Tokenizer.cs: class Tokenizer
Engine/ErrorText.cs: class ErrorText
Engine/Engine.cs: class Engine
Engine/SyntaxException.cs: class SyntaxException : Exception
Engine/IOMap.cs: class IOMap : Dictionary<ITextable, IOutputable>, IRefreshable
Engine/SymbolMap.cs: class SymbolMap : ConcurrentDictionary<string, object>
Engine/SymbolException.cs: class SymbolException : Exception
Engine/RPNTypes.cs: enum RPNTypes
Engine/TokenConverter.cs: class TokenConverter
Program.cs: class Program
Tests/TestUnit.cs:    class TestUnit
DataFixtures/DataFixtures.cs: class DataFixtures : SetT
MyTypes/MyClasses/ClassT.cs: public class ClassT : IItem, IVariable, ICallable
MyTypes/MyClasses/DictionaryT.cs: public class DictionaryT : SortedDictionary<IVariable,IVariable>, IVa
MyTypes/MyClasses/PointerT.cs: class PointerT : Pointer<ReferenceT>, IVariable
MyTypes/MyClasses/DotFunc.cs: class DotFunc : IVariable, ICallable
```

```
MyTypes/MyClasses/ErrorT.cs: class ErrorT : IVariable
MyTypes/MyClasses/TupleT.cs: public class TupleT : SList<IVariable>, IVariable
MyTypes/MyClasses/NullType.cs: class NullType : IVariable
MyTypes/MyClasses/Number.cs: class Number : Pointer<double>, IVariable
MyTypes/MyClasses/StringT.cs: class StringT : Pointer<string>, IVariable, IIndexable
MyTypes/MyClasses/StringT.cs: class MiniParser : IParseable
MyTypes/MyClasses/ReferenceT.cs: public class ReferenceT : Pointer<IVariable>, IVariable, ITypeConvertil
MyTypes/MyClasses/MyRandom.cs: class MyRandom : Number
MyTypes/MyClasses/CallFunc.cs: class CallFunc : IVariable
MyTypes/MyClasses/ProcedureT.cs: public class ProcedureT : OStack, ICallable
MyTypes/MyClasses/IfObject.cs: class IfObject : IVariable
MyTypes/MyClasses/StopPoint.cs: class StopPoint : IVariable
MyTypes/MyClasses/PairT.cs: public class PairT : IVariable
MyTypes/MyClasses/SoftLink.cs: class SoftLink : Pointer<string>, IVariable
MyTypes/MyClasses/MyObject.cs: class MyObject : IVariable
MyTypes/MyClasses/BooleanT.cs: class BooleanT : Pointer<bool>, IVariable
MyTypes/MyClasses/EmptyT.cs: class EmptyT : IVariable
MyTypes/MyClasses/RangeT.cs: class RangeT : IVariable, IEnumerable
MyTypes/MyClasses/BuiltinFunc.cs: class BuiltinFunc : IVariable, ICallable
MyTypes/MyClasses/Callable.cs: class Callable
MyTypes/MyClasses/MyClass.cs: class MyClass : ClassT
MyTypes/MyClasses/ObjectT.cs: class ObjectT : IVariable
MyTypes/MyClasses/SetT.cs: public class SetT : SortedSet<IVariable>, IVariable
MyTypes/MyClasses/ListT.cs: public class ListT : SList<ICompCloneable>, IVariable, IIndexable
MyTypes/MyClasses/PackageT.cs: public class PackageT : List<IItem>, IItem, IVariable
MyTypes/MyClasses/Method.cs: public class Method : IVariable, ICallable
MyTypes/MyClasses/BuiltinClass.cs: class BuiltinClass : ClassT
MyTypes/AccessModifier.cs: public enum AccessEnum
MyTypes/AccessModifier.cs: public class AccessModifier
MyTypes/IVariable.cs: public interface IVariable : ICompCloneable, IStringable, ITuplable
MyTypes/IStepable.cs: interface IStepable : IVariable
MyTypes/Variable.cs: static class Variable
MyTypes/ITuplable.cs: public interface ITuplable
MyTypes/VariableFactory.cs: class VariableFactory
MyTypes/CommandNotFoundException.cs: class CommandNotFoundException : Exception
MyTypes/InfinityException.cs: class InfinityException : NumberException
MyTypes/IIndexable.cs: interface IIndexable
MyTypes/IITem.cs: public interface IItem : IVariable
MyTypes/IStringable.cs: public interface IStringable
MyTypes/ObjectContainer.cs: class ObjectContainer : List<IVariable>
MyTypes/NaNException.cs: class NaNException : NumberException
MyTypes/NotComparableException.cs: class NotComparableException : Exception
MyTypes/NumberException.cs: class NumberException : Exception
MyTypes/UndefinedException.cs: class UndefinedException : Exception
MyTypes/CircularInheritanceException.cs: class CircularInheritanceException : Exception
MyTypes/LambdaConverter.cs: static class LambdaConverter
MyTypes/ICallable.cs: public interface ICallable : IComparable
MyTypes/NoMethodException.cs: class NoMethodException : Exception
MyTypes/ModuleNotFoundException.cs: class ModuleNotFoundException : Exception
Gui/MyMenu.cs: public class MyMenu : MainMenu
Gui/FormAdapter.cs: public class FormAdapter
Gui/VisualSyntax.cs: class VisualSyntax
Gui/MainWindow.cs: class MainWindow : Form
Gui/Clickable.cs: public interface Clickable
```

```
Gui/IOPanel.cs: class IOPanel : Panel
Gui/IClickable.cs: public interface IClickable
Gui/MainPanel.cs: class MainPanel : Panel
Gui/IOBox.cs: class IOBox : RichTextBox
Gui/MyItem.cs: public class MyItem
Gui/InputBox.cs: class InputBox : IOBox, ITextable
Gui/OutputBox.cs: class OutputBox : IOBox, IOutputable
MyCollections/DefaultType.cs: class DefaultType
MyCollections/WList.cs: public class WList<T> : List<T>
MyCollections/IPrintable.cs: public interface IPrintable : IEvalable, IVariable
MyCollections/EmptyArgException.cs: class EmptyArgException : Exception
MyCollections/CList.cs: public class CList<T> : WList<T> where T: IComparable
MyCollections/OStack.cs: public class OStack : WStack<object>
MyCollections/GeneralIndexer.cs: static class GeneralIndexer
MyCollections/TypeTrans.cs: public static class TypeTrans
MyCollections/SList.cs: public class SList<T> : CList<T>, ICompCloneable where T : ICompCloneable
MyCollections/ConcurrentDictionary.cs: public class ConcurrentDictionary<TKey, TValue> : Dictionary<TKe
MyCollections/IParseable.cs: interface IParseable
MyCollections/IValuable.cs: interface IValuable
MyCollections/Pointer.cs: public class Pointer<T>
MyCollections/TokenTypes.cs: public enum TokenTypes
MyCollections/TokenTypes.cs: public static class TokenTypesExtension
MyCollections/Token.cs: public class Token
MyCollections/IEvalable.cs: public interface IEvalable
MyCollections/ICompCloneable.cs: public interface ICompCloneable : IComparable, ICloneable
MyCollections/NullArg.cs: class NullArg
MyCollections/WStack.cs: public class WStack<T> : Stack<T>, IVariable
MyCollections/General.cs: static class General
MyCollections/SortedSet.cs: public class SortedSet<T> : SortedDictionary<T,int>
MyCollections/ITextable.cs: public interface ITextable
Maths/NumberCalcul.cs: static class NumberCalcul
Info/Help.cs: class Help
Info/Info.cs: class Info
Controller/IRefreshable.cs: interface IRefreshable
Controller/Controller.cs:    class Controller
```