Contents

COAS Delphi Help generator

This helpfile is generated by COAS' help generator for Delphi units and components.

For comments please contact jt@coas.com
Please visit http://www.geocities.com/SiliconValley/Vista/5524 for the latest version.

PROGRAMS

UNITS

uExecution
uObjVM
uValues
ValStack
Value
ValueList
ValueOwner
ValueSet
ValueSubSet
ValueUtils
VariableRef
VarLangValue
VarValue
ClassValue

VisualValue unit

see TVisualValue

USES

LangValue, Classes

TYPES

TVisualValue
TVisualValueClass

TVisualValue class

Properties Methods

Unit
VisualValue

Declaration
```
TVisualValue = class ( ILangValue )
```

Description
TVisualValue born to be a parent for all VCL ILangValue s

Properties

LangName LangOwner

Methods

Create CreateSame

Tasks

LangName property

Unit
VisualValue

Applies to
TVisualValue

Declaration
```
property LangName : string
```

Description
Name of TVisualValue in language

Create method

Unit
VisualValue

Applies to
TVisualValue

Declaration
```
constructor Create ( o : TComponent ) ;override;
```

LangOwner property

Unit
VisualValue

Applies to
TVisualValue

Declaration
```
property LangOwner : IValueOwner
```

Description
Owner of TVisualValue in language for example VM for procedure a Class instance for method

CreateSame method

Unit
VisualValue

Applies to
TVisualValue

Declaration
```
function CreateSame ( o : TComponent ) : TVisualValue ;virtual;
```

TVisualValueClass class of tvisualvalue

Unit
VisualValue

Declaration
```
TVisualValueClass = class of TVisualValue
```

Tasks

CodeGenStack unit

USES

Classes, ObjCodeGeneration, Code

TYPES

TCodeGenStack

TCodeGenStack class

Methods

Unit
CodeGenStack

Declaration
```
TCodeGenStack = class
```

Methods

Create	Count	Destroy
Top Second	Push	
PopDef	Pop	RemoveTop

Tasks

Create method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
constructor Create ;
```

Count method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
function Count : integer ;
```

Destroy method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
destructor Destroy ;override;
```

Top method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
function Top : TObjCodeGeneration ;
```

Second method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
function Second : TObjCodeGeneration ;
```

Push method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
procedure Push ( a : TObjCodeGeneration ) ;
```

PopDef method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
procedure PopDef ;
```

Pop method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
procedure Pop ;
```

RemoveTop method

Unit
CodeGenStack

Applies to
TCodeGenStack

Declaration
```
procedure RemoveTop ;
```

ConstValues unit

USES

LangValue, NamedValuesList, ProxyValue, VisualValue

TYPES

TConstValues

TConstValues class

Methods

Unit
ConstValues

Declaration
```
TConstValues = class ( ILangValue )
```

Description
Constant value set

Methods

Create	Destroy	HasValue
AddValue	CreateEqu	RemoveValue
AddTo		

Tasks

Create method

Unit
ConstValues

Applies to
TConstValues

Declaration
```
constructor Create ;
```

Destroy method

Unit
ConstValues

Applies to
TConstValues

Declaration
```
destructor Destroy ;override;
```

HasValue method

Unit
ConstValues

Applies to
TConstValues

Declaration
```
function HasValue ( const s : string ) : boolean ;override;
```

AddValue method

Unit
ConstValues

Applies to
TConstValues

Declaration
```
procedure AddValue ( const s : string ;a : ILangValue ) ;override;
```

CreateEqu method

Unit
ConstValues

Applies to
TConstValues

Declaration
```
function CreateEqu : ILangValue ;override;
```

RemoveValue method

Unit
ConstValues

Applies to
TConstValues

Declaration
```
procedure RemoveValue ( a : ILangValue ) ;override;
```

AddTo method

Unit
ConstValues

Applies to
TConstValues

Declaration
```
procedure AddTo ( a : IValueOwner ) ;
```

Decompiler unit

USES

Code, Classes, Decomps, ObjList, OpCodes, SysUtils

ROUTINES

Decompile

Decompile proc

Unit
Decompiler

Declaration
```
procedure Decompile ( c : TCode ;s : TStrings ) ;
```

Description
Decompile code c and add string representation to s

Tasks

Decomps unit

USES

Code, OpCodes, SysUtils, ObjList

TYPES

TOpDecomp
TOpCodeDecomp
TIntDecomp
TStrDecomp
TFloatDecomp
TBoolDecomp

TOpDecomp class

Properties Methods

Unit
Decomps

Declaration
```
TOpDecomp = class
```

Description
TOpDecomp provides interface to decompile operation

Properties

OpCode

Methods

AsString Length

Tasks

OpCode property

Unit
Decomps

Applies to
TOpDecomp

Declaration
```
OpCode : TOpCode ;
```

Description
Code of operation

AsString method

Unit
Decomps

Applies to
TOpDecomp

Declaration
```
function AsString ( c : TCode ;Pos : integer ) : string ;virtual;abstract;
```

Description
AsString returns string representation of op in given code. C is code, Pos is position of operation being decompiling

Length method

Unit
Decomps

Applies to
TOpDecomp

Declaration
```
function Length ( c : TCode ;Pos : integer ) : integer ;virtual;abstract;
```

Description
Length returns length of current operation in bytes.
Meanings of parameters are same as in AsString function

TOpCodeDecomp class

Unit
Decomps

Declaration
```
TOpCodeDecomp = class ( TOpDecomp )
```

Tasks

TIntDecomp class

Unit
Decomps

Declaration
```
TIntDecomp = class ( TOpCodeDecomp )
```

Tasks

TStrDecomp class

Unit
Decomps

Declaration
```
TStrDecomp = class ( TOpCodeDecomp )
```

Tasks

TFloatDecomp class

Unit
Decomps

Declaration
```
TFloatDecomp = class ( TOpCodeDecomp )
```

Tasks

TBoolDecomp class

Unit
Decomps

Declaration
```
TBoolDecomp = class ( TOpCodeDecomp )Register;
```

Tasks

DirIter unit

USES

Iter, SysUtils

TYPES

TDirIter

TDirIter class

Properties Methods

Unit
DirIter

Declaration
```
TDirIter = class ( TIter )
```

Properties

FileName SearchRec

Methods

Create Next IsEnd
Destroy

Tasks

Create method

Unit
DirIter

Applies to
TDirIter

Declaration
```
constructor Create ( const S : string ;Attr : integer ) ;
```

Next method

Unit
DirIter

Applies to
TDirIter

Declaration
```
procedure Next ;override;
```

IsEnd method

Unit
DirIter

Applies to
TDirIter

Declaration
```
function IsEnd : Boolean ;override;
```

Destroy method

Unit
DirIter

Applies to
TDirIter

Declaration
```
destructor Destroy ;override;
```

FileName property

Unit
DirIter

Applies to
TDirIter

Declaration
```
property FileName : string Read only
```

SearchRec property

Unit
DirIter

Applies to
TDirIter

Declaration
```
property SearchRec : TSearchRec Read only
```

Dirs unit

USES

SysUtils

ROUTINES

OriginDir func

Unit
Dirs

Declaration
```
function OriginDir : string ;
```

Tasks

OriginDirFile func

Unit
Dirs

Declaration
```
function OriginDirFile ( const s : string ) : String ;
```

Tasks

DirFile func

Unit
Dirs

Declaration
```
function DirFile ( const s , f : string ) : string ;
```

Tasks

UniqueFile func

Unit
Dirs

Declaration
```
function UniqueFile ( const Dir , Prefix , Suffix : string ) : string ;
```

Tasks

Execution unit

USES

uExecution, uObjVM, Code, RootValue, ProxyValue, ObjOp

TYPES

TExecution

TExecution class

Methods

Unit
Execution

Declaration
```
TExecution = class ( IExecution )
```

Description
Implementation of IExecution interface

Methods

Create Destroy Step

Tasks

Create method

Unit
Execution

Applies to
TExecution

Declaration
```
Constructor Create ( aVM : IObjVM ;aCode : TCode ) ;
```

Description
Creates an execution of aCode on aVM virtual machine

Destroy method

Unit
Execution

Applies to
TExecution

Declaration
```
destructor Destroy ;override;
```

Description
Destroys execution

Step method

Unit
Execution

Applies to
TExecution

Declaration
```
procedure Step ;override;
```

Description
Step - execute one Operation

FreeMember unit

USES

MemberValue, LangValue

TYPES

TFreeMember

TFreeMember class

Unit
FreeMember

Declaration
```
TFreeMember = class ( TMemberValue )
```

Description
TFreeMember realizes Free procedure for objects

Tasks

InputQueryProc unit

USES

LangValue, VarLangValue, Dialogs

TYPES

TInputQueryProc

TInputQueryProc class

Unit
InputQueryProc

Declaration
```
TInputQueryProc = class ( ILangValue )
```

Tasks

Iter unit

TYPES

TIter

TIter class

Unit
Iter

Declaration
```
TIter = class
```

Tasks

LangProc unit

see TLangProc

USES

VisualValue, LangValue, Classes

TYPES

TExecEvent
TLangProc

TExecEvent type

Unit
LangProc

Declaration
```
TExecEvent = procedure ( Sender : TVisualValue ;S : IValStack ;MustReturn :
boolean ) of object ;
```

Description
TExecEvent is an event to implement Exec method of TLangProc
see Exec method reference for details

Tasks

TLangProc class

Methods Events

Unit
LangProc

Declaration
```
TLangProc = class ( TVisualValue )
```

Methods

Exec CreateSame

Events

OnExec

Tasks

Exec method

Unit
LangProc

Applies to
TLangProc

Declaration
```
procedure Exec ( S : IValStack ;MustReturn : boolean ) ;override;
```

CreateSame method

Unit
LangProc

Applies to
TLangProc

Declaration
```
function CreateSame ( o : TComponent ) : TVisualValue ;override;
```

OnExec event

Unit
LangProc

Applies to
TLangProc

Declaration
```
property OnExec : TExecEvent
```

LangRef unit

USES

LangValue

TYPES

TLangRef

TLangRef class

Properties Methods

Unit
LangRef

Declaration
```
TLangRef = class ( ILangValue )
```

Description
TLangRef is need to implement .Ref value

Properties

Target

Methods

Create	Exec	CreateEqu

Tasks

Target property

Unit
LangRef

Applies to
TLangRef

Declaration
```
Target : ILangValue ;
```

Create method

Unit
LangRef

Applies to
TLangRef

Declaration
```
constructor Create ( aTarget : ILangValue ) ;
```

Exec method

Unit
LangRef

Applies to
TLangRef

Declaration
```
procedure Exec ( S : IValStack ;MustReturn : boolean ) ;override;
```

CreateEqu method

Unit
LangRef

Applies to
TLangRef

Declaration
```
function CreateEqu : ILangValue ;override;
```

LangValMessages unit

USES

SysUtils

ROUTINES

CantBeSTring proc

Unit
LangValMessages

Declaration
```
procedure CantBeSTring ;
```

Tasks

CantBeInteger proc

Unit
LangValMessages

Declaration
```
procedure CantBeInteger ;
```

Tasks

CantBeBoolean proc

Unit
LangValMessages

Declaration
```
procedure CantBeBoolean ;
```

Tasks

CantBeVariant proc

Unit
LangValMessages

Declaration
```
procedure CantBeVariant ;
```

Tasks

CantBeFloat proc

Unit
LangValMessages

Declaration
```
procedure CantBeFloat ;
```

Tasks

ContainsNoValues proc

Unit
LangValMessages

Declaration
```
procedure ContainsNoValues ;
```

Tasks

NotExecutable proc

Unit
LangValMessages

Declaration
```
procedure NotExecutable ;
```

Tasks

NeedParameter proc

Unit
LangValMessages

Declaration
```
procedure NeedParameter ;
```

Tasks

LangValMsgs unit

CONST

LangValMsgs Constants

Declaration

```
const sErrCantBeString = 63001 ;

const sErrCantBeInt = 63002 ;

const sErrCantBeBool = 63003 ;

const sErrCantBeVar = 63004 ;

const sErrCantBeFloat = 63005 ;

const sErrNoValues = 63006 ;

const sErrNotExecutable = 63007 ;

const sErrNeedParameter = 63008 ;
```

LangValue unit

contains TLangValue - Value for macro language

USES

LangValMessages, Classes

TYPES

ILangValue
IValueOwner
IValStack

ILangValue class

Properties Methods

Unit
LangValue

Declaration
```
ILangValue = class ( IValueOwner )
```

Description
ILangValue is a class providing all language objects functionality.
Variables,Values,Procedures,Objects are implementation of ILangValue.

Properties

AsInteger	AsString	AsBoolean
AsVariant	AsFloat	Values

Methods

Exec	HasValue	CreateEqu
GetValue	SetValue	

Tasks

AsInteger property

Unit
LangValue

Applies to
ILangValue

Declaration
```
property AsInteger : Integer
```

Description
Returns integer representation of ILangValue if posible otherwise raises exception

AsString property

Unit
LangValue

Applies to
ILangValue

Declaration
```
property AsString : STring
```

Description
Returns string representation of ILangValue if posible otherwise raises exception

AsBoolean property

Unit
LangValue

Applies to
ILangValue

Declaration
```
property AsBoolean : Boolean
```

Description
Returns boolean representation of ILangValue if posible otherwise raises exception

AsVariant property

Unit
LangValue

Applies to
ILangValue

Declaration
```
property AsVariant : Variant
```

Description
Returns Variant representation of ILangValue if posible otherwise raises exception

AsFloat property

Unit
LangValue

Applies to
ILangValue

Declaration
```
property AsFloat : Extended
```

Description
Returns float representation of ILangValue if posible otherwise raises exception

Exec method

Unit
LangValue

Applies to
ILangValue

Declaration
```
procedure Exec ( S : IValStack ;MustReturn : boolean ) ;virtual;
```

Description
Execute value on given stack. if MustReturn is set to true method have to leave one value to stack. It is expected that Exec clears stack by DropFrame method.

HasValue method

Unit
LangValue

Applies to
ILangValue

Declaration
```
function HasValue ( const s : string ) : boolean ;virtual;
```

Description
If this value has value named s returns true.

Values property

Unit
LangValue

Applies to
ILangValue

Declaration
```
property Values [ const s : string ] : ILangValue Read only
```

Description
Accessor for holded values

CreateEqu method

Unit
LangValue

Applies to
ILangValue

Declaration
```
function CreateEqu : ILangValue ;virtual;abstract;
```

Description
Creates value equal to itself

GetValue method

Unit
LangValue

Applies to
ILangValue

Declaration
```
function GetValue : ILangValue ;
```

Description
Returns value placed inside. For example for variable

SetValue method

Unit
LangValue

Applies to
ILangValue

Declaration
```
procedure SetValue ( a : ILangValue ) ;
```

Description
Sets value to given

IValueOwner class

Methods

Unit
LangValue

Declaration
```
IValueOwner = class ( TComponent )
```

Description
IValueOwner is interface of something that can contain ILangValue

Methods

AddValue RemoveValue

Tasks

AddValue method

Unit
LangValue

Applies to
IValueOwner

Declaration
```
procedure AddValue ( const s : string ;Value : ILangValue ) ;virtual;abstract;
```

RemoveValue method

Unit
LangValue

Applies to
IValueOwner

Declaration
```
procedure RemoveValue ( Value : ILangValue ) ;virtual;abstract;
```

IValStack class

Properties Methods

Unit
LangValue

Declaration
```
IValStack = class
```

Description
IValStack is an interface to stack of ILangValue. Implementaton see in ValStack unit

Properties

FrameValues

Methods

Pop Push DropFrame
FrameCount

Tasks

Pop method

Unit
LangValue

Applies to
IValStack

Declaration
```
function Pop : ILangValue ;virtual;abstract;
```

Description
Pops a value from top of stack

Push method

Unit
LangValue

Applies to
IValStack

Declaration
```
procedure Push ( a : ILangValue ) ;virtual;abstract;
```

Description
Pushes a value to stack

DropFrame method

Unit
LangValue

Applies to
IValStack

Declaration
```
procedure DropFrame ;virtual;abstract;
```

Description
Removes a procedure call frame from top of stack

FrameValues property

Unit
LangValue

Applies to
IValStack

Declaration
```
property FrameValues [ No : Integer ] : ILangValue Read only
```

Description
Provides access to topmost frame values

FrameCount method

Unit
LangValue

Applies to
IValStack

Declaration
```
function FrameCount : integer ;virtual;abstract;
```

Description
Counts values in frame

LangValueList unit

USES

LangValue, ObjList

TYPES

TLangValueList

TLangValueList class

Properties Methods

Unit
LangValueList

Declaration
```
TLangValueList = class
```

Properties

Items

Methods

Create	Destroy	Add
Count	Remove	Delete

Tasks

Create method

Unit
LangValueList

Applies to
TLangValueList

Declaration
```
constructor Create ;
```

Destroy method

Unit
LangValueList

Applies to
TLangValueList

Declaration
```
destructor Destroy ;override;
```

Items property

Unit
LangValueList

Applies to
TLangValueList

Declaration
```
property Items [ No : Integer ] : ILangValue Read only
```

Add method

Unit
LangValueList

Applies to
TLangValueList

Declaration
```
procedure Add ( a : ILangValue ) ;
```

Count method

Unit
LangValueList

Applies to
TLangValueList

Declaration
```
function Count : integer ;
```

Remove method

Unit
LangValueList

Applies to
TLangValueList

Declaration
```
procedure Remove ( No : Integer ) ;
```

Delete method

Unit
LangValueList

Applies to
TLangValueList

Declaration
```
procedure Delete ( No : Integer ) ;
```

LangValues unit

USES

Classes, LangValues

TYPES

TLangValues

TLangValues class

Unit
LangValues

Declaration
```
TLangValues = class
```

Tasks

LangValueStack unit

LangVariable unit

USES

LangValue

TYPES

TLangVariable

TLangVariable class

Properties Methods

Unit
LangVariable

Declaration
```
TLangVariable = class ( ILangValue )
```

Description
Variable for language

Properties

Value

Methods

Create	Destroy	Exec
HasValue	CreateEqu	

Tasks

Value property

Unit
LangVariable

Applies to
TLangVariable

Declaration
```
Value : ILangValue ;
```

Create method

Unit
LangVariable

Applies to
TLangVariable

Declaration
```
constructor Create ;
```

Destroy method

Unit
LangVariable

Applies to
TLangVariable

Declaration
```
destructor Destroy ;override;
```

Exec method

Unit
LangVariable

Applies to
TLangVariable

Declaration
```
procedure Exec ( S : IValStack ;MustReturn : boolean ) ;override;
```

HasValue method

Unit
LangVariable

Applies to
TLangVariable

Declaration
```
function HasValue ( const s : string ) : boolean ;override;
```

CreateEqu method

Unit
LangVariable

Applies to
TLangVariable

Declaration
```
function CreateEqu : ILangValue ;override;
```

LangVariables unit

USES

LangValue, NamedValuesList, LangVariable, ProxyValue

TYPES

TLangVariables

TLangVariables class

Methods

Unit
LangVariables

Declaration
```
TLangVariables = class ( ILangValue )
```

Methods

Create	Destroy	HasValue

Tasks

Create method

Unit
LangVariables

Applies to
TLangVariables

Declaration
```
constructor Create ;
```

Destroy method

Unit
LangVariables

Applies to
TLangVariables

Declaration
```
destructor Destroy ;override;
```

HasValue method

Unit
LangVariables

Applies to
TLangVariables

Declaration
```
function HasValue ( const s : string ) : boolean ;override;
```

MemberValue unit

USES

LangValue

TYPES

TMemberValue

TMemberValue class

Properties Methods

Unit
MemberValue

Declaration
```
TMemberValue = class ( ILangValue )
```

Properties

Parent

Methods

Create

Tasks

Parent property

Unit
MemberValue

Applies to
TMemberValue

Declaration
```
Parent : ILangValue ;
```

Create method

Unit
MemberValue

Applies to
TMemberValue

Declaration
```
constructor Create ( aParent : ILangValue ) ;
```

NamedValuesList unit

USES

LangValue, ObjStr

TYPES

TNamedValuesList

TNamedValuesList class

Properties Methods

Unit
NamedValuesList

Declaration
```
TNamedValuesList = class
```

Properties

Items	Names

Methods

Create	Destroy	Count
ValByName	Add	DeleteValue

Tasks

Create method

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
constructor Create ;
```

Destroy method

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
destructor Destroy ;override;
```

Items property

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
property Items [ No : integer ] : ILangValue Read only
```

Names property

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
property Names [ No : integer ] : string Read only
```

Count method

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
function Count : integer ;
```

ValByName method

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
function ValByName ( const a : string ) : ILangValue ;
```

Add method

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
procedure Add ( const Name : string ;a : ILangValue ) ;
```

DeleteValue method

Unit
NamedValuesList

Applies to
TNamedValuesList

Declaration
```
procedure DeleteValue ( a : ILangValue ) ;
```

NodeStack unit

Stack of parse tree nodes

USES

PrseTree, Classes

TYPES

TNodeStack

TNodeStack class

Methods

Unit
NodeStack

Declaration
```
TNodeStack = class
```

Methods

Create Destroy Push
Pop

Tasks

Create method

Unit
NodeStack

Applies to
TNodeStack

Declaration
```
constructor Create ;
```

Destroy method

Unit
NodeStack

Applies to
TNodeStack

Declaration
```
destructor Destroy ;override;
```

Push method

Unit
NodeStack

Applies to
TNodeStack

Declaration
```
procedure Push ( a : PNode ) ;
```

Pop method

Unit
NodeStack

Applies to
TNodeStack

Declaration
```
function Pop : PNode ;
```

ObjCodeGeneration unit

USES

Code, OpCodes, IntStack, NodeStack

TYPES

TObjCodeGeneration

TObjCodeGeneration class

Properties Methods

Unit
ObjCodeGeneration

Declaration
```
TObjCodeGeneration = class
```

Properties

Cur Stack	NodeStack

Methods

Create	Destroy	Code
cOpCode	cByte	cHalt
cAdd	cSub	cMul
cDivcNegate	cAnd	
cOr cNot	cEqu	
cNEcG	cL	
cLE cGE	cGet	
cSet	cExec	cEval
cGetElem	cRoot	cStr
cFloat	cInt	cBool
cNop	cJZ	cJNZ
cJMP	cJRZ	cDcrR
cToR	cFromR	cRDrop
cIF cTHEN	cELSE	
cTIMES	cLOOP	cBEGIN
cWHILE	cREPEAT	cGen

Tasks

Cur property

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
Cur : Integer ;
```

Stack property

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
Stack : TIntStack ;
```

NodeStack property

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
NodeStack : TNodeStack ;
```

Create method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
constructor Create ( c : TCode ) ;
```

Destroy method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
destructor Destroy ;override;
```

Code method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
function Code : TCode ;
```

cOpCode method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cOpCode ( OpCode : Integer ) ;
```

cByte method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cByte ( a : Byte ) ;
```

cHalt method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cHalt ;
```

cAdd method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cAdd ;
```

cSub method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cSub ;
```

cMul method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cMul ;
```

cDiv method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cDiv ;
```

cNegate method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cNegate ;
```

cAnd method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cAnd ;
```

Description
Boolean operations

cOr method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cOr ;
```

cNot method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cNot ;
```

cEqu method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cEqu ;
```

Description
Comparison ops

cNE method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cNE ;
```

cG method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cG ;
```

cL method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cL ;
```

cLE method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cLE ;
```

cGE method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cGE ;
```

cGet method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cGet ;
```

Description
Value operations

cSet method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cSet ;
```

cExec method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cExec ;
```

cEval method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cEval ;
```

cGetElem method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cGetElem ;
```

cRoot method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cRoot ;
```

cStr method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cStr ( const a : string ) ;
```

Description
Constant operations

cFloat method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cFloat ( a : Extended ) ;
```

cInt method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cInt ( a : Integer ) ;
```

cBool method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cBool ( a : boolean ) ;
```

cNop method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cNop ;
```

Description
Execution flow operations

cJZ method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cJZ ( a : Integer ) ;
```

cJNZ method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cJNZ ( a : Integer ) ;
```

cJMP method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cJMP ( a : Integer ) ;
```

cJRZ method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cJRZ ( a : Integer ) ;
```

cDcrR method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cDcrR ;
```

cToR method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cToR ;
```

cFromR method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cFromR ;
```

Description
Moves value from return stack to data stack

cRDrop method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cRDrop ;
```

Description
Drops value from reaturn stack

cIF method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cIF ;
```

Description
Structural Macros
<e> IF <1> ELSE <2> THEN - Like FORTH

cTHEN method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cTHEN ;
```

cELSE method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cELSE ;
```

cTIMES method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cTIMES ;
```

Description
<e> TIMES <Action> LOOP

cLOOP method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cLOOP ;
```

cBEGIN method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cBEGIN ;
```

Description
BEGIN <e> WHILE <Action> REPEAT

cWHILE method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cWHILE ;
```

cREPEAT method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cREPEAT ;
```

cGen method

Unit
ObjCodeGeneration

Applies to
TObjCodeGeneration

Declaration
```
procedure cGen ( Gen : TObjCodeGeneration ) ;
```

Description
Adds Gen code to current generation

ObjectValue unit

USES

LangValue, Classes, ConstValues, MemberValue, FreeMember

TYPES

TObjectValue
TDefObjectValue

TObjectValue class

Properties Methods

Unit
ObjectValue

Declaration
```
TObjectValue = class ( TConstValues )
```

Properties

Target

Methods

Create

Tasks

Target property

Unit
ObjectValue

Applies to
TObjectValue

Declaration
```
Target : TObject ;
```

Create method

Unit
ObjectValue

Applies to
TObjectValue

Declaration
```
constructor Create ( aTarget : TObject ) ;
```

TDefObjectValue class

Unit
ObjectValue

Declaration
```
TDefObjectValue = class ( TObjectValue )
```

Tasks

ObjOp unit

USES

uExecution

TYPES

TObjOp
TObjOpClass

TObjOp class

Properties Methods

Unit
ObjOp

Declaration
```
TObjOp = class
```

Properties

OpCode

Methods

Create Destroy Execute

Tasks

OpCode property

Unit
ObjOp

Applies to
TObjOp

Declaration
```
OpCode : Integer ;
```

Create method

Unit
ObjOp

Applies to
TObjOp

Declaration
```
Constructor Create ;virtual;
```

Destroy method

Unit
ObjOp

Applies to
TObjOp

Declaration
```
Destructor Destroy ;override;
```

Execute method

Unit
ObjOp

Applies to
TObjOp

Declaration
```
procedure Execute ( a : IExecution ) ;virtual;abstract;
```

TObjOpClass class of tobjop

Unit
ObjOp

Declaration
```
TObjOpClass = class of TObjOp
```

Tasks

ObjOps unit

USES

ObjOp, OpList, OpCodes, SysUtils

ObjVM unit

USES

uObjVM, Classes, ObjOp, OpList, RootValue, Code, uExecution, LangVariables, ObjOps, ConstValues, LangValue

TYPES

TObjVM

TObjVM class

Properties Methods

Unit
ObjVM

Declaration
```
TObjVM = class ( IObjVM )
```

Description
TObjVM is a virtual machine for "objective" languages

Properties

Constants

Methods

StartExecution	Create	Destroy
AddValue	RemoveValue	

Tasks

Constants property

Unit
ObjVM

Applies to
TObjVM

Declaration
```
Constants : TConstValues ;
```

StartExecution method

Unit
ObjVM

Applies to
TObjVM

Declaration
```
function StartExecution ( c : TCode ) : IExecution ;override;
```

Create method

Unit
ObjVM

Applies to
TObjVM

Declaration
```
constructor Create ( o : TComponent ) ;override;
```

Destroy method

Unit
ObjVM

Applies to
TObjVM

Declaration
```
destructor Destroy ;override;
```

AddValue method

Unit
ObjVM

Applies to
TObjVM

Declaration
```
procedure AddValue ( const Name : string ;a : ILangValue ) ;override;
```

RemoveValue method

Unit
ObjVM

Applies to
TObjVM

Declaration
```
procedure RemoveValue ( a : ILangValue ) ;override;
```

OpCodes unit

Opcodes constants

CONST

ocAdd
ocSub
ocMul
ocDiv
ocNegate
ocAnd
ocOr
ocNot
ocEqu
ocNE
ocG
ocL
ocLE
ocGE
ocGet
ocSet
ocExec
ocEval
ocGetElem
ocRoot
ocStr
ocFloat
ocInt
ocBool
ocNop
ocHalt
ocJZ
ocJNZ
ocJMP
ocJRZ
ocDcrR
ocToR
ocFromR
ocRDrop

TYPES

TOpCode

OpCodes Constants

Declaration

```
const ocAdd = -1 ;
```

Arithmetic operations

```
const ocSub = -2 ;

const ocMul = -3 ;

const ocDiv = -4 ;

const ocNegate = -5 ;

const ocAnd = -11 ;
```

Boolean operations

```
const ocOr = -12 ;

const ocNot = -13 ;

const ocEqu = -14 ;
```

Comparison ops

```
const ocNE = -15 ;

const ocG = -16 ;

const ocL = -17 ;

const ocLE = -18 ;

const ocGE = -19 ;

const ocGet = -21 ;
```

Value operations

```
const ocSet = -22 ;

const ocExec = -23 ;

const ocEval = -24 ;

const ocGetElem = -25 ;

const ocRoot = -26 ;

const ocStr = -31 ;
```

Constant operations

```
const ocFloat = -32 ;

const ocInt = -33 ;
```

```
const ocBool = -34 ;

const ocNop = -40 ;
```
Execution flow operations
```
const ocHalt = -41 ;

const ocJZ = -42 ;

const ocJNZ = -43 ;

const ocJMP = -44 ;

const ocJRZ = -45 ;
```
Jump if top of Return stack is Zero
```
const ocDcrR = -46 ;
```
DeCRement top of return stack
```
const ocToR = -47 ;
```
Moves value to return stack
```
const ocFromR = -48 ;
```
Moves value from return stack to data stack
```
const ocRDrop = -49 ;
```
Drops value from reaturn stack

TOpCode type

Unit
OpCodes

Declaration
```
TOpCode = integer ;
```

Tasks

OpList unit

USES

ObjOp, ObjList

TYPES

TOpList

TOpList class

Methods

Unit
OpList

Declaration
```
TOpList = class
```

Methods

Create	Destroy	Register
Find		

Tasks

Create method

Unit
OpList

Applies to
TOpList

Declaration
```
constructor Create ;
```

Destroy method

Unit
OpList

Applies to
TOpList

Declaration
```
destructor Destroy ;override;
```

Register method

Unit
OpList

Applies to
TOpList

Declaration
```
procedure Register ( OpCode : integer ;OpClass : TObjOpClass ) ;
```

Find method

Unit
OpList

Applies to
TOpList

Declaration
```
function Find ( OpCode : integer ) : TObjOp ;
```

ProxyValue unit

contains TProxyValue

USES

VisualValue, LangValue, Classes

TYPES

TProxyValue

TProxyValue class

Properties Methods

Unit
ProxyValue

Declaration
```
TProxyValue = class ( TVisualValue )
```

Description
TProxyValue is an alias to any other ILangValue.
See Target property for details

Properties

Target

Methods

Exec	CreateEqu	CreateFrom
HasValue	Create	CreateSame

Tasks

Exec method

Unit
ProxyValue

Applies to
TProxyValue

Declaration
```
procedure Exec ( S : IValStack ;MustReturn : boolean ) ;override;
```

CreateEqu method

Unit
ProxyValue

Applies to
TProxyValue

Declaration
```
function CreateEqu : ILangValue ;override;
```

CreateFrom method

Unit
ProxyValue

Applies to
TProxyValue

Declaration
```
constructor CreateFrom ( aTarget : ILangValue ) ;
```

HasValue method

Unit
ProxyValue

Applies to
TProxyValue

Declaration
```
function HasValue ( const s : string ) : boolean ;override;
```

Target property

Unit
ProxyValue

Applies to
TProxyValue

Declaration
```
property Target : ILangValue
```

Description
ILangValue represented by proxy. All call to proxy is implemented
by repeating this calls to Target. See source of TProxyValue for details

Create method

Unit
ProxyValue

Applies to
TProxyValue

Declaration
```
constructor Create ( o : TComponent ) ;override;
```

CreateSame method

Unit
ProxyValue

Applies to
TProxyValue

Declaration
```
function CreateSame ( o : TComponent ) : TVisualValue ;override;
```

PtrStack unit

USES

Classes

TYPES

TPtrStack

TPtrStack class

Unit
PtrStack

Declaration
```
TPtrStack = class ( TList )
```

Tasks

RegObjVM unit

RootValue unit

USES

LangValue, LangValueList

TYPES

TRootValue

TRootValue class

Properties Methods

Unit
RootValue

Declaration
```
TRootValue = class ( ILangValue )
```

Properties

ValueList

Methods

HasValue Create Destroy
Add CreateEqu

Tasks

ValueList property

Unit
RootValue

Applies to
TRootValue

Declaration
```
ValueList : TLangValueList ;
```

Description
List of Values included in Root

HasValue method

Unit
RootValue

Applies to
TRootValue

Declaration
```
function HasValue ( const s : string ) : boolean ;override;
```

Create method

Unit
RootValue

Applies to
TRootValue

Declaration
```
constructor Create ;
```

Destroy method

Unit
RootValue

Applies to
TRootValue

Declaration
```
destructor Destroy ;override;
```

Add method

Unit
RootValue

Applies to
TRootValue

Declaration
```
procedure Add ( a : ILangValue ) ;
```

CreateEqu method

Unit
RootValue

Applies to
TRootValue

Declaration
```
function CreateEqu : ILangValue ;override;
```

ShowProc unit

USES

LangValue, VarLangValue, Dialogs

TYPES

TShowProc

TShowProc class

Unit
ShowProc

Declaration
```
TShowProc = class ( ILangValue )
```

Tasks

StrIter unit

USES

Iter

TYPES

TStrIter

TStrIter class

Unit
StrIter

Declaration
```
TStrIter = class ( TIter )
```

Tasks

TObjCodeGeneration unit

uExecution unit

USES

Code, LangValue, Forms, RootValue, ConstValues

TYPES

IExecution

IExecution class

Properties Methods

Unit
uExecution

Declaration
```
IExecution = class
```

Properties

Root		Consts	Code
IP	IsEnd	Source	
Pos	Return		

Methods

BeginExec	Step	EndExec
Run		

Tasks

Root property

Unit
uExecution

Applies to
IExecution

Declaration
```
Root : TRootValue ;
```

Description
Root - root value

Consts property

Unit
uExecution

Applies to
IExecution

Declaration
```
Consts : TConstValues ;
```

Description
Constants - execution depended

Code property

Unit
uExecution

Applies to
IExecution

Declaration
```
Code : TCode ;
```

Description
Code - currently executing code

IP property

Unit
uExecution

Applies to
IExecution

Declaration
```
IP : integer ;
```

Description
IP - Instruction pointer - address of currently executing instruction

IsEnd property

Unit
uExecution

Applies to
IExecution

Declaration
```
IsEnd : boolean ;
```

Description
IsEnd - determines when execution must be stopped

Source property

Unit
uExecution

Applies to
IExecution

Declaration
```
Source : TObject ;
```

Description
Source - source of Code (Reserved for debug purporses)

Pos property

Unit
uExecution

Applies to
IExecution

Declaration
```
Pos : Integer ;
```

Return property

Unit
uExecution

Applies to
IExecution

Declaration
```
Return : IValStack ;
```

Description
Return is a stack for storing some execution-flow information
such as return addresses (not implemented) and FOR loop variables

BeginExec method

Unit
uExecution

Applies to
IExecution

Declaration
```
procedure BeginExec ;virtual;
```

Description
BeginExec - see TExecution reference

Step method

Unit
uExecution

Applies to
IExecution

Declaration
```
procedure Step ;virtual;abstract;
```

Description
Step - see TExecution reference

EndExec method

Unit
uExecution

Applies to
IExecution

Declaration
```
procedure EndExec ;virtual;abstract;
```

Description
EndExec - see TExecution reference

Run method

Unit
uExecution

Applies to
IExecution

Declaration
```
procedure Run ;virtual;
```

Description
Run - run execution until IsEnd value is setted to True

uObjVM unit

USES

Classes, Code, uExecution, LangValue, RootValue, OpList

TYPES

IObjVM

IObjVM class

Properties Methods

Unit
uObjVM

Declaration
```
IObjVM = class ( IValueOwner )
```

Properties

Ops Root

Methods

StartExecution	Run

Tasks

Ops property

Unit
uObjVM

Applies to
IObjVM

Declaration
```
Ops : TOpList ;
```

Root property

Unit
uObjVM

Applies to
IObjVM

Declaration
```
Root : TRootValue ;
```

StartExecution method

Unit
uObjVM

Applies to
IObjVM

Declaration
```
function StartExecution ( c : TCode ) : IExecution ;virtual;abstract;
```

Run method

Unit
uObjVM

Applies to
IObjVM

Declaration
```
procedure Run ( c : TCode ) ;virtual;
```

uValues unit

USES

Value

TYPES

TValues

TValues class

Properties Methods

Unit
uValues

Declaration
```
TValues = class
```

Properties

Values	Count

Methods

Add Clear

Tasks

Values property

Unit
uValues

Applies to
TValues

Declaration
```
property Values [ No : Integer ] : TValue Read only
```

Count property

Unit
uValues

Applies to
TValues

Declaration
```
property Count : integer
```

Add method

Unit
uValues

Applies to
TValues

Declaration
```
procedure Add ( a : TValue ) ;virtual;abstract;
```

Clear method

Unit
uValues

Applies to
TValues

Declaration
```
procedure Clear ;virtual;abstract;
```

ValStack unit

USES

LangValue, LangValueList, LangValMessages

TYPES

TValStack

TValStack class

Properties Methods

Unit
ValStack

Declaration
```
TValStack = class ( IValStack )
```

Properties

Items

Methods

Pop Push	DropFrame	
FrameCount	Create	Destroy

Tasks

Items property

Unit
ValStack

Applies to
TValStack

Declaration
```
Items : TLangValueList ;
```

Pop method

Unit
ValStack

Applies to
TValStack

Declaration
```
function Pop : ILangValue ;override;
```

Push method

Unit
ValStack

Applies to
TValStack

Declaration
```
procedure Push ( a : ILangValue ) ;override;
```

DropFrame method

Unit
ValStack

Applies to
TValStack

Declaration
```
procedure DropFrame ;override;
```

FrameCount method

Unit
ValStack

Applies to
TValStack

Declaration
```
function FrameCount : integer ;override;
```

Create method

Unit
ValStack

Applies to
TValStack

Declaration
```
constructor Create ;
```

Destroy method

Unit
ValStack

Applies to
TValStack

Declaration
```
destructor Destroy ;override;
```

Value unit

USES

Classes

TYPES

TValue

TValue class

Methods Events

Unit
Value

Declaration
```
TValue = class ( TPersistent )
```

Methods

Assign

Events

v

Tasks

v event

Unit
Value

Applies to
TValue

Declaration
```
v : Variant ;
```

Assign method

Unit
Value

Applies to
TValue

Declaration
```
procedure Assign ( a : TPersistent ) ;override;
```

ValueList unit

USES

uValues, Value, ObjList

TYPES

TValueList

TValueList class

Methods

Unit
ValueList

Declaration
```
TValueList = class ( TValues )
```

Methods

Add Create Destroy
Clear

Tasks

Add method

Unit
ValueList

Applies to
TValueList

Declaration
```
procedure Add ( a : TValue ) ;override;
```

Create method

Unit
ValueList

Applies to
TValueList

Declaration
```
constructor Create ;
```

Destroy method

Unit
ValueList

Applies to
TValueList

Declaration
```
destructor Destroy ;override;
```

Clear method

Unit
ValueList

Applies to
TValueList

Declaration
```
procedure Clear ;override;
```

ValueOwner unit

USES

Classes

TYPES

TValueOwner

TValueOwner class

Unit
ValueOwner

Declaration
```
TValueOwner = class ( TComponent )
```

Tasks

ValueSet unit

USES

uValues, Value, Classes

TYPES

TValueSet

TValueSet class

Methods

Unit
ValueSet

Declaration
```
TValueSet = class ( TValues )
```

Methods

Add Create Destroy
Clear

Tasks

Add method

Unit
ValueSet

Applies to
TValueSet

Declaration
```
procedure Add ( a : TValue ) ;override;
```

Create method

Unit
ValueSet

Applies to
TValueSet

Declaration
```
constructor Create ;
```

Destroy method

Unit
ValueSet

Applies to
TValueSet

Declaration
```
destructor Destroy ;override;
```

Clear method

Unit
ValueSet

Applies to
TValueSet

Declaration
```
procedure Clear ;override;
```

ValueSubSet unit

USES

uValues, Value

TYPES

TValueSubSet

TValueSubSet class

Methods

Unit
ValueSubSet

Declaration
```
TValueSubSet = class ( TValues )
```

Methods

Create

Tasks

Create method

Unit
ValueSubSet

Applies to
TValueSubSet

Declaration
```
constructor Create ( aValues : TValues ;aStart , aCount : Integer ) ;
```

ValueUtils unit

Useful procedures

USES

LangValue

ROUTINES

ReturnTrue proc

Unit
ValueUtils

Declaration
```
procedure ReturnTrue ( s : IValStack ;MustReturn : boolean ) ;
```

Tasks

ReturnBool proc

Unit
ValueUtils

Declaration
```
procedure ReturnBool ( s : IValStack ;MustReturn : boolean ;a : boolean ) ;
```

Tasks

ReturnString proc

Unit
ValueUtils

Declaration
```
procedure ReturnString ( s : IValStack ;MustReturn : boolean ;const str :
string ) ;
```

Tasks

ReturnVar proc

Unit
ValueUtils

Declaration
```
procedure ReturnVar ( s : IValStack ;MustReturn : boolean ;avar : Variant ) ;
```

Tasks

SetObjectOf proc

Unit
ValueUtils

Declaration
```
procedure SetObjectOf ( val : ILangValue ;a : TObject ) ;
```

Tasks

GetObjectOf func

Unit
ValueUtils

Declaration
```
function GetObjectOf ( val : ILangValue ) : TObject ;
```

Tasks

VariableRef unit

USES

LangValue, LangVariable

TYPES

TVariableRef

TVariableRef class

Unit
VariableRef

Declaration
```
TVariableRef = class ( ILangValue )
```

Tasks

VarLangValue unit

USES

LangValue

TYPES

TVarLangValue

TVarLangValue class

Methods

Unit
VarLangValue

Declaration
```
TVarLangValue = class ( ILangValue )
```

Methods

CreateEqu

Tasks

CreateEqu method

Unit
VarLangValue

Applies to
TVarLangValue

Declaration
```
function CreateEqu : ILangValue ;override;
```

VarValue unit

USES

LangValue

TYPES

TVarValue

TVarValue class

Properties Methods

Unit
VarValue

Declaration
```
TVarValue = class ( ILangValue )
```

Properties

Value

Methods

Exec

Tasks

Exec method

Unit
VarValue

Applies to
TVarValue

Declaration
```
procedure Exec ( S : IValStack ;MustReturn : boolean ) ;override;
```

Value property

Unit
VarValue

Applies to
TVarValue

Declaration
```
property Value : variant
```

ClassValue unit

USES

VisualValue, ConstValues, LangValue, ObjectValue, Classes, ValueUtils, ProxyValue

CONST

nCreate

TYPES

TClassValue
TCreateValue

ClassValue Constants

Declaration

```
const nCreate = 'Create' ;
```

TClassValue class

Methods

Unit
ClassValue

Declaration
```
TClassValue = class ( TVisualValue )
```

Description
TClassValue is value to implement classes in objVM language

Methods

HasValue	AddValue	RemoveValue
CreateObject	Create	Destroy

Tasks

HasValue method

Unit
ClassValue

Applies to
TClassValue

Declaration
```
function HasValue ( const s : string ) : boolean ;override;
```

AddValue method

Unit
ClassValue

Applies to
TClassValue

Declaration
```
procedure AddValue ( const aName : string ;a : ILangValue ) ;override;
```

RemoveValue method

Unit
ClassValue

Applies to
TClassValue

Declaration
```
procedure RemoveValue ( a : ILangValue ) ;override;
```

CreateObject method

Unit
ClassValue

Applies to
TClassValue

Declaration
```
function CreateObject ( o : TComponent ) : ILangValue ;
```

Create method

Unit
ClassValue

Applies to
TClassValue

Declaration
```
constructor Create ( o : TComponent ) ;override;
```

Destroy method

Unit
ClassValue

Applies to
TClassValue

Declaration
```
destructor Destroy ;override;
```

TCreateValue class

Methods

Unit
ClassValue

Declaration
```
TCreateValue = class ( ILangValue )
```

Methods

CreateFrom Exec

Tasks

CreateFrom method

Unit
ClassValue

Applies to
TCreateValue

Declaration
```
Constructor CreateFrom ( a : TClassValue ) ;
```

Exec method

Unit
ClassValue

Applies to
TCreateValue

Declaration
```
procedure Exec ( s : IValStack ;MustReturn : boolean ) ;override;
```