Namespace chia.dotnet.clvm

Classes

CompileOptions

Represents the options for compiling a CLVM program.

Cons

Represents a cons cell in a program.

OperatorsType

Represents a collection of operators used in the CLVM language.

ParseError

Position

Represents a position in a source code file, specified by line and column numbers.

Program

Represents a CLVM program.

ProgramOutput

Represents the output of a CLVM program execution.

RunOptions

Represents the options for running a CLVM program.

Delegates

Operator

Represents a delegate for an operator function.

Class CompileOptions

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents the options for compiling a CLVM program.

```
public record CompileOptions : RunOptions, IEquatable<RunOptions>,
IEquatable<CompileOptions>
```

Inheritance

object ← RunOptions ← CompileOptions

Implements

IEquatable<RunOptions>, IEquatable<CompileOptions>

Inherited Members

RunOptions.MaxCost , RunOptions.Operators , RunOptions.Strict , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

IncludeFilePaths

Gets or sets the include file paths used during compilation.

```
public IDictionary<string, IDictionary<string, string>> IncludeFilePaths { get; init; }
```

Property Value

IDictionary<string, IDictionary<string, string>>

Represents the options for compiling a CLVM program.

Class Cons

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents a cons cell in a program.

```
public class Cons : Tuple<Program, Program>, IStructuralComparable, IStructuralEquatable,
IComparable, ITuple
```

Inheritance

object ← Tuple<Program, Program> ← Cons

Implements

IStructuralComparable, IStructuralEquatable, IComparable, ITuple

Inherited Members

Tuple<Program, Program>.Equals(object) , Tuple<Program, Program>.GetHashCode() , Tuple<Program, Program>.ToString() , Tuple<Program, Program>.Item1 , Tuple<Program, Program>.Item2 , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

https://en.wikipedia.org/wiki/Cons/

Constructors

Cons(Program, Program)

Represents a cons cell in a program.

```
public Cons(Program item1, Program item2)
```

Parameters

`item1` Program

Represents a cons cell in a program.

`item2` Program

Represents a cons cell in a program.

Remarks

https://en.wikipedia.org/wiki/Cons/

Delegate Operator

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents a delegate for an operator function.

```
public delegate ProgramOutput Operator(Program args)
```

Parameters

`args` Program

The arguments passed to the operator.

Returns

ProgramOutput

The output of the operator.

Class OperatorsType

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents a collection of operators used in the CLVM language.

```
public record OperatorsType : IEquatable<OperatorsType>
```

Inheritance

object ← OperatorsType

Implements

IEquatable<OperatorsType>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

OperatorsType()

Initializes a new instance of the OperatorsType class.

```
public OperatorsType()
```

Properties

Apply

Gets or sets the apply operator symbol.

```
public string Apply { get; init; }
```

Property Value

string

Represents a collection of operators used in the CLVM language.

Operators

Gets or sets the dictionary of operators.

```
public IDictionary<string, Operator> Operators { get; init; }
```

Property Value

IDictionary <string, Operator>

Represents a collection of operators used in the CLVM language.

Quote

Gets or sets the quote operator symbol.

```
public string Quote { get; init; }
```

Property Value

string

Represents a collection of operators used in the CLVM language.

Unknown

Gets or sets the unknown operator function.

```
public Func<Program, Program, ProgramOutput> Unknown { get; set; }
```

Property Value

Func <Program, Program, ProgramOutput>

Represents a collection of operators used in the CLVM language.

Class ParseError

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

```
public class ParseError : Exception, ISerializable
```

Inheritance

object ← Exception ← ParseError

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

ParseError()

```
public ParseError()
```

ParseError(string)

```
public ParseError(string message)
```

Parameters

`message` string

ParseError(string, Exception)

```
public ParseError(string message, Exception inner)
```

Parameters

`message` string

`inner` Exception

Class Position

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents a position in a source code file, specified by line and column numbers.

```
public class Position
```

Inheritance

object ← Position

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Position(string, int)

Initializes a new instance of the Position class with the specified source code and index.

```
public Position(string source, int index)
```

Parameters

`source` string

The source code.

`index` int

The index of the position in the source code.

Properties

Column

Gets the column number of the position.

```
public int Column { get; init; }
```

Property Value

int

Represents a position in a source code file, specified by line and column numbers.

Line

Gets the line number of the position.

```
public int Line { get; init; }
```

Property Value

int

Represents a position in a source code file, specified by line and column numbers.

Methods

ToString()

Returns a string that represents the current position in the format "line:column".

```
public override string ToString()
```

Returns

string

A string representation of the position.

Class Program

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents a CLVM program.

```
public class Program
```

Inheritance

object ← Program

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

Program(byte[])

Initializes a new instance of the Program class with a byte array value.

```
public Program(byte[] value)
```

Parameters

`value` byte[]

The byte array value to initialize with.

Program(Cons)

Initializes a new instance of the Program class with a Cons value.

```
public Program(Cons value)
```

Parameters

`value` Cons

The Cons value to initialize with.

Fields

False

Represents the False program.

```
public static readonly Program False
```

Field Value

Program

Represents a CLVM program.

Nil

Represents the Nil program.

```
public static readonly Program Nil
```

Field Value

Program

Represents a CLVM program.

True

Represents the True program.

```
public static readonly Program True
```

Field Value

Program

Represents a CLVM program.

Properties

Atom

Gets the atom value of the Program.

```
public byte[] Atom { get; }
```

Property Value

byte[]

Represents a CLVM program.

Exceptions

InvalidOperationException

Thrown when the Program is not an atom.

Cons

Gets the Cons value of the Program.

```
public Cons Cons { get; }
```

Property Value

Cons

Represents a CLVM program.

Exceptions

InvalidOperationException

Thrown when the Program is not a Cons.

First

Gets the first Program in the Cons.

```
public Program First { get; }
```

Property Value

Program

Represents a CLVM program.

IsAtom

Gets a value indicating whether the Program is an atom.

```
public bool IsAtom { get; }
```

Property Value

bool

Represents a CLVM program.

IsCons

Gets a value indicating whether the Program is a Cons.

```
public bool IsCons { get; }
```

Property Value

bool

Represents a CLVM program.

IsNull

Gets a value indicating whether the Program is null.

```
public bool IsNull { get; }
```

Property Value

bool

Represents a CLVM program.

Position

Gets or privately sets the position of the Program.

```
public Position? Position { get; }
```

Property Value

Position

Represents a CLVM program.

PositionSuffix

Gets the position suffix of the Program.

```
public string PositionSuffix { get; }
```

Property Value

string

Represents a CLVM program.

Rest

Gets the rest of the Programs in the Cons.

```
public Program Rest { get; }
```

Property Value

Program

Represents a CLVM program.

Value

Gets the value of the Program.

```
public object Value { get; }
```

Property Value

object

Represents a CLVM program.

Methods

At(Position)

Sets the position of the Program and returns the Program.

```
public Program At(Position position)
```

Parameters

`position` Position

The position to set.

Returns

Program

The Program with the set position.

Compile(CompileOptions?)

Compiles the Program with the given options.

```
public ProgramOutput Compile(CompileOptions? options = null)
```

Parameters

`options` CompileOptions

The options to use when compiling the Program. If null, default options are used.

Returns

ProgramOutput

A ProgramOutput representing the result of the compilation.

Curry(IEnumerable<Program>)

Curries the Program with a list of arguments.

```
public Program Curry(IEnumerable<Program> args)
```

Parameters

`args` IEnumerable<Program>

The list of arguments to curry the Program with.

Returns

Program

A new Program that is the result of currying the original Program with the arguments.

Define(Program)

Defines a new Program within the current Program.

```
public Program Define(Program program)
```

Parameters

`program` Program

The Program to define within the current Program.

Returns

Program

A new Program with the defined Program inserted.

DefineAll(IEnumerable<Program>)

Defines multiple Programs within the current Program.

```
public Program DefineAll(IEnumerable<Program> programs)
```

Parameters

`programs` IEnumerable<Program>

The Programs to define within the current Program.

Returns

Program

A new Program with all the defined Programs inserted.

Deserialize(byte[])

Deserializes a byte array into a Program.

```
public static Program Deserialize(byte[] bytes)
```

Parameters

`bytes` byte[]

The byte array to deserialize.

Returns

Program

A new Program deserialized from the byte array.

Exceptions

ParseError

Thrown when the byte array is unexpectedly empty.

DeserializeHex(string)

Deserializes a hexadecimal string into a Program.

```
public static Program DeserializeHex(string hex)
```

Parameters

`hex` string

The hexadecimal string to deserialize.

Returns

Program

A new Program deserialized from the hexadecimal string.

Equals(Program)

Determines whether the specified Program is equal to the current Program.

```
public bool Equals(Program value)
```

Parameters

`value` Program

The Program to compare with the current Program.

Returns

bool

true if the specified Program is equal to the current Program; otherwise, false.

FromBigInt(BigInteger)

Creates a new Program from a BigInteger.

```
public static Program FromBigInt(BigInteger value)
```

Parameters

`value` BigInteger

The BigInteger to convert.

Returns

Program

A new Program.

FromBool(bool)

Creates a new Program from a boolean value.

```
public static Program FromBool(bool value)
```

Parameters

`value` bool

The boolean value to convert.

Returns

Program

A new Program.

FromBytes(byte[])

Creates a program from a byte array.

```
public static Program FromBytes(byte[] value)
```

Parameters

`value` byte[]

The byte array.

Returns

Program

The created program.

FromCons(Program, Program)

Creates a program from two cons cells.

```
public static Program FromCons(Program program1, Program program2)
```

Parameters

`program1` Program

The first program.

`program2` Program

The second program.

Returns

Program

The created program.

FromHex(string)

Creates a new Program from a hexadecimal string.

```
public static Program FromHex(string value)
```

Parameters

`value` string

The hexadecimal string to convert.

Returns

Program

A new Program.

FromInt(long)

Creates a new Program from a long integer.

```
public static Program FromInt(long value)
```

Parameters

`value` long

The long integer to convert.

Returns

Program

A new Program.

FromJacobianPoint(JacobianPoint)

Creates a new Program from a JacobianPoint.

```
public static Program FromJacobianPoint(JacobianPoint value)
```

Parameters

`value` JacobianPoint

The JacobianPoint to convert.

Returns

Program

A new Program.

FromList(IEnumerable<Program>)

Creates a new Program from a list of Programs.

```
public static Program FromList(IEnumerable<Program> programs)
```

Parameters

`programs` IEnumerable<Program>

The list of Programs to convert.

Returns

Program

A new Program created from the list of Programs.

FromPrivateKey(PrivateKey)

Creates a new Program from a PrivateKey.

```
public static Program FromPrivateKey(PrivateKey value)
```

Parameters

`value` PrivateKey

The PrivateKey to convert.

Returns

Program

A new Program.

FromSource(string)

Creates a new Program from a source string.

```
public static Program FromSource(string source)
```

Parameters

`source` string

The source string to parse.

Returns

Program

A new Program parsed from the source string.

Exceptions

ParseError

Thrown when the source string is unexpectedly empty.

FromText(string)

Creates a new Program from a text string.

```
public static Program FromText(string value)
```

Parameters

`value` string

The text string to convert.

Returns

Program

A new Program.

Hash()

Computes the hash of the Program.

```
public byte[] Hash()
```

Returns

byte[]

A byte array representing the hash of the Program.

HashHex()

Computes the hash of the Program and returns it as a hexadecimal string.

```
public string HashHex()
```

Returns

string

A string representing the hash of the Program in hexadecimal.

Run(Program, RunOptions?)

Runs the Program with the given environment and options.

```
public ProgramOutput Run(Program environment, RunOptions? options = null)
```

Parameters

`environment` Program

The environment to use when running the Program.

`options` RunOptions

The options to use when running the Program. If null, default options are used.

Returns

ProgramOutput

A ProgramOutput representing the result of the run.

Serialize()

Serializes the Program to a byte array.

```
public byte[] Serialize()
```

Returns

byte[]

A byte array representing the serialized Program.

SerializeHex()

Serializes the Program to a hexadecimal string.

```
public string SerializeHex()
```

Returns

string

A string representing the serialized Program in hexadecimal.

ToBigInt()

Converts the Program to a BigInteger.

```
public BigInteger ToBigInt()
```

Returns

BigInteger

A BigInteger representing the Program.

Exceptions

Exception

Thrown when the Program is a Cons.

ToBool()

Converts the Program to a boolean.

```
public bool ToBool()
```

Returns

bool

A boolean representing the Program.

Exceptions

Exception

Thrown when the Program is a Cons.

ToBytes()

Converts the Program to a byte array.

```
public byte[] ToBytes()
```

Returns

byte[]

A byte array representing the Program.

Exceptions

Exception

Thrown when the Program is a Cons.

ToHex()

Converts the Program to a hexadecimal string.

```
public string ToHex()
```

Returns

string

A string representing the Program in hexadecimal.

ToInt()

Converts the Program to a long integer.

```
public long ToInt()
```

Returns

long

A long integer representing the Program.

Exceptions

Exception

Thrown when the Program is a Cons.

ToJacobianPoint()

Converts the Program to a JacobianPoint.

```
public JacobianPoint ToJacobianPoint()
```

Returns

JacobianPoint

A JacobianPoint representing the Program.

Exceptions

Exception

Thrown when the Program is a Cons or the Atom length is not 48 or 96.

ToList(bool)

Converts the Program to a list of Programs.

```
public IList<Program> ToList(bool strict = false)
```

Parameters

`strict` bool

A boolean indicating whether to enforce strict list conversion. Defaults to false.

Returns

IList<Program>

A list of Programs representing the Program.

Exceptions

Exception

Thrown when the Program is not a strict list and strict is true.

ToPrivateKey()

Converts the Program to a PrivateKey.

```
public PrivateKey ToPrivateKey()
```

Returns

PrivateKey

A PrivateKey representing the Program.

Exceptions

Exception

Thrown when the Program is a Cons.

ToSource(bool)

Converts the Program to its source code representation.

```
public string ToSource(bool showKeywords = true)
```

Parameters

`showKeywords` bool

A boolean indicating whether to include keywords in the source code representation. Defaults to true.

Returns

string

A string representing the source code of the Program.

ToString()

Returns a string that represents the current Program.

```
public override string ToString()
```

Returns

string

A string that represents the current Program.

ToText()

Converts the Program to a text string.

```
public string ToText()
```

Returns

string

A string representing the Program in text.

Exceptions

Exception

Thrown when the Program is a Cons.

Uncurry()

Uncurries the Program into a tuple containing the original Program and a list of arguments.

```
public Tuple<Program, IEnumerable<Program>>? Uncurry()
```

Returns

Tuple<Program, IEnumerable<Program>>

A tuple containing the original Program and a list of arguments, or null if the Program cannot be uncurried.

Class ProgramOutput

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents the output of a CLVM program execution.

```
public record ProgramOutput : IEquatable<ProgramOutput>
```

Inheritance

object ← ProgramOutput

Implements

IEquatable<ProgramOutput>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Cost

Gets or initializes the cost of executing the CLVM program.

```
public BigInteger Cost { get; init; }
```

Property Value

BigInteger

Represents the output of a CLVM program execution.

Value

Gets or initializes the value produced by the CLVM program.

```
public Program Value { get; init; }
```

Property Value

[Program](#)

Represents the output of a CLVM program execution.

Class RunOptions

Namespace: chia.dotnet.clvm

Assembly: chia-dotnet-clvm.dll

Represents the options for running a CLVM program.

```
public record RunOptions : IEquatable<RunOptions>
```

Inheritance

object ← RunOptions

Implements

IEquatable<RunOptions>

Derived

CompileOptions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

MaxCost

Gets or sets the maximum cost allowed for executing the program.

```
public BigInteger? MaxCost { get; init; }
```

Property Value

BigInteger?

Represents the options for running a CLVM program.

Operators

Gets or sets the type of operators to be used in the program.

```
public OperatorsType Operators { get; init; }
```

Property Value

OperatorsType

Represents the options for running a CLVM program.

Strict

Gets or sets a value indicating whether strict mode is enabled.

```
public bool Strict { get; init; }
```

Property Value

bool

Represents the options for running a CLVM program.