

Namespace TestProjectSampurna

Classes

[CircleCommandTest](#)

Contains unit tests for the [CircleCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when drawing circles on the canvas.

[ClearCommandTest](#)

Unit tests for the [ClearCommand](#) class. Ensures the command validates parameters correctly and triggers a canvas clear operation when executed.

[DrawToCommandTest](#)

Contains unit tests for the [DrawToCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when drawing lines on the canvas.

[IfsAndLoopstest](#)

End-to-end unit tests for IF / IF-ELSE / WHILE / FOR commands. These tests execute real BOOSE programs using ExtendedParser and validate behaviour via canvas output.

[MethodAndCallCommandTests](#)

Contains end-to-end unit tests for BOOSE method definition and invocation.

[MockCanvas](#)

A mock implementation of BOOSE.ICanvas used for unit testing. It records method calls, argument values, and internal drawing state without performing any real graphical operations.

[MoveToCommandTest](#)

Contains unit tests for the [MoveToCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when moving the canvas cursor.

[MultilineProgramTest](#)

Provides unit tests for executing multiline BOOSE programs. Verifies drawing commands, colour changes, pen commands, and conditional or sequential execution using MockCanvas.

[PenCommandTest](#)

Contains unit tests for the [PenCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when setting the pen colour.

[RectangleCommandTest](#)

Contains unit tests for the [RectangleCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when drawing rectangles on the canvas.

[ResetCommandTest](#)

Unit tests for the [`resetCommand`](#) class. Ensures correct validation, compilation behavior, and execution effects on both the stored program and the canvas.

[TriangleCommandTest](#)

[VariableTest](#)

Minimal variable facility tests required for: Int, Real, Boolean, Array, Poke, Peek.

[WriteCommandTest](#)

Contains unit tests for the [`WriteCommand`](#) class. These tests verify correct parameter validation and execution behaviour when writing text to the canvas.

Class CircleCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains unit tests for the [CircleCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when drawing circles on the canvas.

```
[TestClass]
public class CircleCommandTest
```

Inheritance

[object](#) ← CircleCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Execute_InvalidExpression_Throws()

Ensures that executing the command with an invalid (non-numeric) radius expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_InvalidExpression_Throws()
```

Execute_NegativeRadius_Throws()

Ensures that executing the command with a negative radius throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_NegativeRadius_Throws()
```

Execute_ValidRadius_DrawsCircle()

Ensures that executing the command with a valid radius draws exactly one unfilled circle with the expected radius.

```
[TestMethod]  
public void Execute_ValidRadius_DrawsCircle()
```

Execute_ZeroRadius_Throws()

Ensures that executing the command with a radius of zero throws a BOOSE.CommandException.

```
[TestMethod]  
[ExpectedException(typeof(CommandException))]  
public void Execute_ZeroRadius_Throws()
```

Set_Empty_Throws()

Ensures that calling [Set\(StoredProgram, string\)](#) with an empty parameter string throws a BOOSE.CommandException.

```
[TestMethod]  
[ExpectedException(typeof(CommandException))]  
public void Set_Empty_Throws()
```

Set_WithRadius_Succeeds()

Ensures that providing a valid radius during [Set\(StoredProgram, string\)](#) succeeds without immediately drawing a circle.

```
[TestMethod]  
public void Set_WithRadius_Succeeds()
```

Setup()

Creates a fresh mock canvas, stored program, and circle command instance before each test runs.

```
[TestInitialize]  
public void Setup()
```

Class ClearCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Unit tests for the [ClearCommand](#) class. Ensures the command validates parameters correctly and triggers a canvas clear operation when executed.

```
[TestClass]  
public class ClearCommandTest
```

Inheritance

[object](#) ← ClearCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

CheckParameters_WithValues_Throws()

Ensures that [CheckParameters\(\)](#) throws when parameters exist.

```
[TestMethod]  
[ExpectedException(typeof(CommandException))]  
public void CheckParameters_WithValues_Throws()
```

Compile_NoError()

Confirms [Compile\(\)](#) does nothing and does not throw exceptions.

```
[TestMethod]  
public void Compile_NoError()
```

Execute_CallsClearOnce()

Ensures that executing the command calls `canvas.Clear()` exactly once.

```
[TestMethod]
public void Execute_CallsClearOnce()
```

Set_NoParameters_Success()

Confirms that calling `Set()` with empty parameters succeeds.

```
[TestMethod]
public void Set_NoParameters_Success()
```

Set_WithParameters_Throws()

Ensures that providing parameters to `Set()` throws an exception.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_WithParameters_Throws()
```

Setup()

Initializes required objects before every test run.

```
[TestInitialize]
public void Setup()
```

Class DrawToCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains unit tests for the [DrawToCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when drawing lines on the canvas.

```
[TestClass]
public class DrawToCommandTest
```

Inheritance

[object](#) ← DrawToCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Execute_AllowsCommaSeparatedParams()

Ensures that comma-separated coordinate parameters are accepted and correctly parsed by the command.

```
[TestMethod]
public void Execute_AllowsCommaSeparatedParams()
```

Execute_InvalidXExpression_Throws()

Ensures that executing the command with an invalid (non-numeric) X-coordinate expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_InvalidXExpression_Throws()
```

Execute_InvalidYExpression_Throws()

Ensures that executing the command with an invalid (non-numeric) Y-coordinate expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_InvalidYExpression_Throws()
```

Execute_ValidParameters_CallsDrawTo()

Ensures that executing the command with valid parameters results in exactly one call to `DrawTo` on the canvas with the expected coordinates.

```
[TestMethod]
public void Execute_ValidParameters_CallsDrawTo()
```

Set_InvalidCount_Throws()

Ensures that providing an incorrect number of parameters to [Set\(StoredProgram, string\)](#) throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_InvalidCount_Throws()
```

Set_ValidParameters_DoesNotDraw()

Ensures that calling [Set\(StoredProgram, string\)](#) with valid parameters succeeds and does not immediately draw a line.

```
[TestMethod]
public void Set_ValidParameters_DoesNotDraw()
```

Setup()

Creates a fresh mock canvas, stored program, and draw-to command instance before each test runs.

```
[TestInitialize]  
public void Setup()
```

Class IfsAndLoopstest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

End-to-end unit tests for IF / IF-ELSE / WHILE / FOR commands. These tests execute real BOOSE programs using ExtendedParser and validate behaviour via canvas output.

```
[TestClass]  
public class IfsAndLoopstest
```

Inheritance

[object](#) ← IfsAndLoopstest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

For_LoopsAndExecutesBody()

FOR: loop iterates from 1 to 5 with step 1. Expected circles: 10, 20, 30, 40, 50.

```
[TestMethod]  
public void For_LoopsAndExecutesBody()
```

IfElse_False_RunsElseBlock()

IF-ELSE: when condition is false, ELSE block runs.

```
[TestMethod]  
public void IfElse_False_RunsElseBlock()
```

IfElse_True_RunsThenBlock()

IF-ELSE: when condition is true, THEN block runs and ELSE is skipped.

```
[TestMethod]  
public void IfElse_True_RunsThenBlock()
```

If_False_SkipsBody()

IF: when the condition is false, the body is skipped.

```
[TestMethod]  
public void If_False_SkipsBody()
```

If_True_RunsBody()

IF: when the condition is true, the body executes.

```
[TestMethod]  
public void If_True_RunsBody()
```

Setup()

Creates a fresh canvas, program, and parser before each test.

```
[TestInitialize]  
public void Setup()
```

While_LoopsExpectedNumberOfTimes()

WHILE: loop repeats until condition becomes false. Expected circles: 100, 85, 70, 55.

```
[TestMethod]  
public void While_LoopsExpectedNumberOfTimes()
```

Class MethodAndCallCommandTests

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains end-to-end unit tests for BOOSE method definition and invocation.

```
[TestClass]
public class MethodAndCallCommandTests
```

Inheritance

[object](#) ← MethodAndCallCommandTests

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

These tests validate that:

- Method blocks are skipped during normal execution unless explicitly called.
- `call` executes a method body and updates the method return variable.
- Incorrect method syntax and missing methods raise appropriate exceptions.

Methods

Call_ExecuteMethodAndSetsReturnVariable()

Ensures that `call` executes the method body and stores the result in the method return variable.

```
[TestMethod]
public void Call_ExecuteMethodAndSetsReturnVariable()
```

Remarks

The script defines two integer methods (multiply and divide), calls them with arguments, and asserts that the method return variables contain the expected results.

Call_MissingMethod_ThrowsAtRuntime()

Ensures that calling a method which does not exist in the program throws a BOOSE.CommandException at runtime.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Call_MissingMethod_ThrowsAtRuntime()
```

Remarks

This test builds a program and parses a `call` to a missing method name. In this implementation, the call command defers validation until execution, so the exception is expected during [Run\(\)](#).

EndMethod_WithoutMethod_Throws()

Ensures that an `endmethod` statement without a preceding `method` causes the parser to throw a BOOSE.ParserException.

```
[TestMethod]
[ExpectedException(typeof(ParserException))]
public void EndMethod_WithoutMethod_Throws()
```

MethodDefinition_IsSkippedDuringNormalFlow()

Ensures that defining a method does not execute its body during normal program flow when there is no corresponding `call` statement.

```
[TestMethod]
public void MethodDefinition_IsSkippedDuringNormalFlow()
```

Remarks

The test defines a method that would set its return variable to 99. Because the method is never called, the return variable should remain at its default value.

Class MockCanvas

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

A mock implementation of BOOSE.ICanvas used for unit testing. It records method calls, argument values, and internal drawing state without performing any real graphical operations.

```
public class MockCanvas : ICanvas
```

Inheritance

[object](#) ← MockCanvas

Implements

ICanvas

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CircleCallCount

Number of times Circle was called.

```
public int CircleCallCount { get; }
```

Property Value

[int](#)

ClearCallCount

Number of times Clear was called.

```
public int ClearCallCount { get; }
```

Property Value

[int ↗](#)

DrawToCallCount

Number of times DrawTo was called.

```
public int DrawToCallCount { get; }
```

Property Value

[int ↗](#)

LastBlue

Last blue colour component used.

```
public int LastBlue { get; }
```

Property Value

[int ↗](#)

LastFilled

Indicates whether the last shape was filled.

```
public bool LastFilled { get; }
```

Property Value

[bool ↗](#)

LastGreen

Last green colour component used.

```
public int LastGreen { get; }
```

Property Value

[int ↗](#)

LastHeight

Last rectangle/triangle height recorded.

```
public int LastHeight { get; }
```

Property Value

[int ↗](#)

LastRadius

Last circle radius recorded.

```
public int LastRadius { get; }
```

Property Value

[int ↗](#)

LastRed

Last red colour component used.

```
public int LastRed { get; }
```

Property Value

[int](#)

LastText

Stores the last text written using WriteText.

```
public string LastText { get; }
```

Property Value

[string](#)

LastWidth

Last rectangle/triangle width recorded.

```
public int LastWidth { get; }
```

Property Value

[int](#)

LastX

Last X position updated by MoveTo or DrawTo.

```
public int LastX { get; }
```

Property Value

[int](#)

LastY

Last Y position updated by MoveTo or DrawTo.

```
public int LastY { get; }
```

Property Value

[int ↗](#)

MoveToCallCount

Number of times MoveTo was called.

```
public int MoveToCallCount { get; }
```

Property Value

[int ↗](#)

PenColour

The current pen colour.

```
public object PenColour { get; set; }
```

Property Value

[object ↗](#)

RectCallCount

Number of times Rect was called.

```
public int RectCallCount { get; }
```

Property Value

[int ↗](#)

ResetCallCount

Number of times Reset was called.

```
public int ResetCallCount { get; }
```

Property Value

[int↗](#)

SetColourCallCount

Number of times SetColour was called.

```
public int SetColourCallCount { get; }
```

Property Value

[int↗](#)

TriCallCount

Number of times Tri was called.

```
public int TriCallCount { get; }
```

Property Value

[int↗](#)

WriteTextCallCount

Number of times WriteText was called.

```
public int WriteTextCallCount { get; }
```

Property Value

[int ↗](#)

Xpos

The current X position of the pen.

```
public int Xpos { get; set; }
```

Property Value

[int ↗](#)

Ypos

The current Y position of the pen.

```
public int Ypos { get; set; }
```

Property Value

[int ↗](#)

Methods

Circle(int, bool)

Records a Circle command and stores the radius and fill mode.

```
public void Circle(int radius, bool filled)
```

Parameters

`radius` [int ↗](#)

`filled` [bool ↗](#)

Clear()

Records a Clear command.

```
public void Clear()
```

DrawTo(int, int)

Records a DrawTo operation and updates internal pen position.

```
public void DrawTo(int x, int y)
```

Parameters

x [int](#)

y [int](#)

MoveTo(int, int)

Records a MoveTo operation and updates internal pen position.

```
public void MoveTo(int x, int y)
```

Parameters

x [int](#)

y [int](#)

Rect(int, int, bool)

Records a rectangle command and stores width, height, and fill mode.

```
public void Rect(int width, int height, bool filled)
```

Parameters

width [int](#)

height [int](#)

filled [bool](#)

Reset()

Resets pen position and colour to defaults and increments Reset call counter.

```
public void Reset()
```

ResetCounters()

Resets all recorded method call counters. Used between test cases to ensure clean state.

```
public void ResetCounters()
```

Set(int, int)

Sets the canvas size (not used in mock implementation).

```
public void Set(int width, int height)
```

Parameters

width [int](#)

height [int](#)

SetColour(int, int, int)

Records a colour change and stores RGB components.

```
public void SetColour(int red, int green, int blue)
```

Parameters

red [int ↗](#)

green [int ↗](#)

blue [int ↗](#)

Tri(int, int)

Records a triangle command and stores width and height.

```
public void Tri(int width, int height)
```

Parameters

width [int ↗](#)

height [int ↗](#)

WriteText(string)

Records text written by the WriteText command.

```
public void WriteText(string text)
```

Parameters

text [string ↗](#)

getBitmap()

Returns a dummy bitmap for compatibility with ICanvas.

```
public object getBitmap()
```

Returns

object

Class MoveToCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains unit tests for the [MoveToCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when moving the canvas cursor.

```
[TestClass]
public class MoveToCommandTest
```

Inheritance

[object](#) ← MoveToCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Execute_AllowsCommaSeparatedParams()

Ensures that comma-separated coordinate parameters are accepted and correctly parsed by the command.

```
[TestMethod]
public void Execute_AllowsCommaSeparatedParams()
```

Execute_ShouldMoveCursor_WhenParametersValid()

Ensures that executing the command with valid parameters moves the canvas cursor to the expected coordinates.

```
[TestMethod]
public void Execute_ShouldMoveCursor_WhenParametersValid()
```

Execute_ShouldThrow_WhenXParameterInvalid()

Ensures that executing the command with an invalid (non-numeric) X-coordinate expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_ShouldThrow_WhenXParameterInvalid()
```

Execute_ShouldThrow_WhenYParameterInvalid()

Ensures that executing the command with an invalid (non-numeric) Y-coordinate expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_ShouldThrow_WhenYParameterInvalid()
```

Set_ShouldThrow_WhenMissingParameters()

Ensures that providing fewer than the required number of parameters to [Set\(StoredProgram, string\)](#) throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_ShouldThrow_WhenMissingParameters()
```

Setup()

Creates a fresh mock canvas, stored program, and move-to command instance before each test runs.

```
[TestInitialize]
public void Setup()
```

Class MultilineProgramTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Provides unit tests for executing multiline BOOSE programs. Verifies drawing commands, colour changes, pen commands, and conditional or sequential execution using MockCanvas.

```
[TestClass]
public class MultilineProgramTest
```

Inheritance

[object](#) ← MultilineProgramTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Cleanup()

Resets all MockCanvas counters after every test to guarantee clean and reproducible test execution.

```
[TestCleanup]
public void Cleanup()
```

Multiple_Program_Test()

Verifies that multiple sequential BOOSE commands execute correctly. Confirms movement, drawing, shape creation, colour change, and rectangle drawing all occur exactly once.

```
[TestMethod]
public void Multiple_Program_Test()
```

Setup()

Initializes the test components before each test run. Creates fresh instances of MockCanvas, command factory, stored program, and parser.

```
[TestInitialize]  
public void Setup()
```

Class PenCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains unit tests for the [PenCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when setting the pen colour.

```
[TestClass]  
public class PenCommandTest
```

Inheritance

[object](#) ← PenCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Execute_AllowsCommaSeparatedParams()

Ensures that comma-separated RGB parameters are accepted and correctly parsed by the command.

```
[TestMethod]  
public void Execute_AllowsCommaSeparatedParams()
```

Execute_ShouldSetPenColour_WhenValidRGB()

Ensures that executing the command with valid RGB values sets the pen colour on the canvas correctly.

```
[TestMethod]  
public void Execute_ShouldSetPenColour_WhenValidRGB()
```

Execute_ShouldThrow_WhenBIsNotNumber()

Ensures that executing the command with an invalid (non-numeric) blue colour expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_ShouldThrow_WhenBIsNotNumber()
```

Execute_ShouldThrow_WhenGIsNotNumber()

Ensures that executing the command with an invalid (non-numeric) green colour expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_ShouldThrow_WhenGIsNotNumber()
```

Execute_ShouldThrow_WhenRIsNotNumber()

Ensures that executing the command with an invalid (non-numeric) red colour expression throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_ShouldThrow_WhenRIsNotNumber()
```

Set_ShouldThrow_WhenNotThreeParameters()

Ensures that providing fewer or more than three parameters to [Set\(StoredProgram, string\)](#) throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_ShouldThrow_WhenNotThreeParameters()
```

Setup()

Creates a fresh mock canvas, stored program, and pen command instance before each test runs.

```
[TestInitialize]  
public void Setup()
```

Class RectangleCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains unit tests for the [RectangleCommand](#) class. These tests verify correct parameter parsing, validation, and execution behaviour when drawing rectangles on the canvas.

```
[TestClass]  
public class RectangleCommandTest
```

Inheritance

[object](#) ← RectangleCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Execute_AllowsCommaSeparatedParams()

Ensures that comma-separated parameters are accepted and correctly parsed by the rectangle command.

```
[TestMethod]  
public void Execute_AllowsCommaSeparatedParams()
```

Execute_ShouldDrawFilledRectangle_WhenTruePassed()

Ensures that executing the command with the optional `true` fill parameter draws a filled rectangle.

```
[TestMethod]  
public void Execute_ShouldDrawFilledRectangle_WhenTruePassed()
```

Execute_ShouldDrawRectangle_WithValidParameters()

Ensures that executing the command with valid width and height draws an unfilled rectangle with the expected dimensions.

```
[TestMethod]  
public void Execute_ShouldDrawRectangle_WithValidParameters()
```

Execute_ShouldDrawUnfilledRectangle_WhenFalsePassed()

Ensures that executing the command with the optional `false` fill parameter draws an unfilled rectangle.

```
[TestMethod]  
public void Execute_ShouldDrawUnfilledRectangle_WhenFalsePassed()
```

Execute_ShouldThrow_WhenHeightIsNegative()

Ensures that executing the command with a negative height throws a BOOSE.CommandException.

```
[TestMethod]  
[ExpectedException(typeof(CommandException))]  
public void Execute_ShouldThrow_WhenHeightIsNegative()
```

Execute_ShouldThrow_WhenWidthExpressionInvalid()

Ensures that executing the command with an invalid (non-numeric) width expression throws a BOOSE.CommandException.

```
[TestMethod]  
[ExpectedException(typeof(CommandException))]  
public void Execute_ShouldThrow_WhenWidthExpressionInvalid()
```

Execute_ShouldThrow_WhenWidthIsZero()

Ensures that executing the command with a width of zero throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_ShouldThrow_WhenWidthIsZero()
```

Set_ShouldThrow_WhenFillIsInvalid_FormatException()

Ensures that providing an invalid fill parameter value causes a [FormatException](#) to be thrown.

```
[TestMethod]
[ExpectedException(typeofFormatException)]
public void Set_ShouldThrow_WhenFillIsInvalid_FormatException()
```

Remarks

This occurs because [Parse\(string\)](#) is used and the value is not validated inside the command.

Set_ShouldThrow_WhenMissingParameters()

Ensures that calling [Set\(StoredProgram, string\)](#) with fewer than the required parameters throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_ShouldThrow_WhenMissingParameters()
```

Set_ShouldThrow_WhenTooManyParameters()

Ensures that calling [Set\(StoredProgram, string\)](#) with more than the allowed number of parameters throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_ShouldThrow_WhenTooManyParameters()
```

Set_WithTwoParams_Succeeds_AndDoesNotDraw()

Ensures that providing only width and height parameters to [Set\(StoredProgram, string\)](#) succeeds and does not immediately draw a rectangle.

```
[TestMethod]  
public void Set_WithTwoParams_Succeeds_AndDoesNotDraw()
```

Setup()

Creates a fresh mock canvas, stored program, and rectangle command instance before each test runs.

```
[TestInitialize]  
public void Setup()
```

Class ResetCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Unit tests for the [resetCommand](#) class. Ensures correct validation, compilation behavior, and execution effects on both the stored program and the canvas.

```
[TestClass]
public class ResetCommandTest
```

Inheritance

[object](#) ← ResetCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

CheckParameters_WithValues_Throws()

Ensures that [CheckParameters\(\)](#) throws when parameters exist.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void CheckParameters_WithValues_Throws()
```

Compile_NoError()

Confirms [Compile\(\)](#) performs no actions and does not throw.

```
[TestMethod]
public void Compile_NoError()
```

Execute_ResetsProgramAndCanvas()

Ensures `Execute()` calls both:

- `program.ResetProgram()`
- `canvas.Reset()`

```
[TestMethod]
public void Execute_ResetsProgramAndCanvas()
```

Set_NoParameters_Success()

Ensures `Set()` succeeds when no parameters are given.

```
[TestMethod]
public void Set_NoParameters_Success()
```

Set_WithParameters_Throws()

Ensures supplying parameters to `Set()` throws an exception.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_WithParameters_Throws()
```

Setup()

Creates fresh instances before each test run.

```
[TestInitialize]
public void Setup()
```

Class TriangleCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

```
[TestClass]
public class TriangleCommandTest
```

Inheritance

[object](#) ← TriangleCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Execute_InvalidExpression_Throws()

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_InvalidExpression_Throws()
```

Execute_NegativeSize_Throws()

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_NegativeSize_Throws()
```

Execute_TooManyTokens_TreatedAsInvalidExpression_Throws()

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_TooManyTokens_TreatedAsInvalidExpression_Throws()
```

Execute_ValidSize_DrawsTriangle()

```
[TestMethod]
public void Execute_ValidSize_DrawsTriangle()
```

Execute_ZeroSize_Throws()

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Execute_ZeroSize_Throws()
```

Set_Empty_Throws()

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_Empty_Throws()
```

Set_TrimsWhitespace_Succeeds()

```
[TestMethod]
public void Set_TrimsWhitespace_Succeeds()
```

Setup()

```
[TestInitialize]
public void Setup()
```

Class VariableTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Minimal variable facility tests required for: Int, Real, Boolean, Array, Poke, Peek.

```
[TestClass]  
public class VariableTest
```

Inheritance

[object](#) ← VariableTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

ArrayCommand_CreatesIntArray()

Array can be created with a valid size and default values exist.

```
[TestMethod]  
public void ArrayCommand_CreatesIntArray()
```

BooleanCommand_InitialisesValue()

Boolean variable can be declared and initialised.

```
[TestMethod]  
public void BooleanCommand_InitialisesValue()
```

IntCommand_InitialisesValue()

Int variable can be declared and initialised from an expression.

```
[TestMethod]
public void IntCommand_InitialisesValue()
```

PeekCommand_ReadsArrayToInt()

Peek reads an array element into a variable (int target).

```
[TestMethod]
public void PeekCommand_ReadsArrayToInt()
```

PokeCommand_WritesIntoArray()

Poke writes a value into an array element.

```
[TestMethod]
public void PokeCommand_WritesIntoArray()
```

RealCommand_InitialisesValue()

Real variable can be declared and initialised from an expression.

```
[TestMethod]
public void RealCommand_InitialisesValue()
```

Setup()

```
[TestInitialize]
public void Setup()
```

Class WriteCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains unit tests for the [WriteCommand](#) class. These tests verify correct parameter validation and execution behaviour when writing text to the canvas.

```
[TestClass]  
public class WriteCommandTest
```

Inheritance

[object](#) ← WriteCommandTest

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Methods

Execute_CalledOnce_WritesOnce()

Ensures that executing the command writes text exactly once to the canvas.

```
[TestMethod]  
public void Execute_CalledOnce_WritesOnce()
```

Execute_UnquotedLiteral_WritesText()

Ensures that executing the command with an unquoted literal string writes the expected text to the canvas.

```
[TestMethod]  
public void Execute_UnquotedLiteral_WritesText()
```

Set_Empty_Throws()

Ensures that calling [Set\(StoredProgram, string\)](#) with an empty parameter string throws a BOOSE.CommandException.

```
[TestMethod]
[ExpectedException(typeof(CommandException))]
public void Set_Empty_Throws()
```

Set_WhitespaceOnly_Throws()

Ensures that providing a parameter consisting only of whitespace to [Set\(StoredProgram, string\)](#) throws a BOOSE.CommandException.

```
[TestMethod]
public void Set_WhitespaceOnly_Throws()
```

Setup()

Creates a fresh mock canvas, stored program, and write command instance before each test runs.

```
[TestInitialize]
public void Setup()
```

Namespace WinFormsApp1

Classes

[AppCommandFactory](#)

Application-specific command factory responsible for creating BOOSE command instances based on parsed command names.

[ArrayCommand](#)

Implements the BOOSE `array` command, allowing declaration and management of integer or real arrays within a BOOSE program.

[AssignCommand](#)

Implements the BOOSE `assign` command, which assigns the result of an expression to an existing variable.

[BooleanCommand](#)

Implements the BOOSE `boolean` command, which declares and manages boolean variables within a BOOSE program.

[CallCommand](#)

Implements the BOOSE `call` command, which invokes a user-defined method and passes arguments to it.

[CircleCommand](#)

Implements the BOOSE `circle` command, which draws a circle on the canvas using the current pen settings.

[ClearCommand](#)

Command that clears the canvas to its default background color.

[DrawToCommand](#)

Implements the BOOSE `drawto` command, which draws a line from the current canvas position to a specified coordinate.

[ElseCommand](#)

Implements the BOOSE `else` command.

[EndForCommand](#)

Implements the BOOSE `endfor` command.

[EndIfCommand](#)

Implements the BOOSE `endif` command.

[EndMethodCommand](#)

Implements the BOOSE `endmethod` command.

[EndWhileCommand](#)

Implements the BOOSE `endwhile` command.

[ExtendedParser](#)

Extends the BOOSE BOOSE.Parser to support additional syntax features, improved error aggregation, and integration with [ExtendedStoredProgram](#).

[ExtendedStoredProgram](#)

Extends BOOSE.StoredProgram to provide full runtime support for methods, variables, call stacks, and execution control.

[ForCommand](#)

Implements the BOOSE `for` loop command.

[Form1](#)

Main UI form for the BOOSE drawing interpreter.

[IfCommand](#)

Implements the BOOSE `if` command for conditional execution.

[IntCommand](#)

Implements the BOOSE `int` command for declaring and managing integer variables.

[MethodCommand](#)

Implements the BOOSE `method` command for defining user methods.

[MoveToCommand](#)

Implements the BOOSE `moveto` command, which moves the current canvas position without drawing.

[PeekCommand](#)

Implements the BOOSE `peek` command, which retrieves a value from an array and assigns it to an existing variable.

[PenCommand](#)

Implements the BOOSE `pen` command, which sets the current drawing colour.

[PokeCommand](#)

Implements the BOOSE `poke` command, which assigns a value to an array element.

[RealCommand](#)

Implements the BOOSE `real` command for declaring and managing floating-point variables.

[RectangleCommand](#)

Implements the BOOSE `rect` command, which draws a rectangle on the canvas.

[TriangleCommand](#)

Implements the BOOSE `tri` command, which draws a triangle on the canvas.

[WhileCommand](#)

Implements the BOOSE `while` command for conditional looping.

[WriteCommand](#)

Implements the BOOSE `write` command, which displays text on the canvas.

[canvasApp](#)

Canvas implementation used by BOOSE commands to draw shapes, lines, text, and manage pen state inside a bitmap.

[resetCommand](#)

Represents the RESET command. Clears the stored program and resets the drawing canvas. Syntax:

`reset`

Class AppCommandFactory

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Application-specific command factory responsible for creating BOOSE command instances based on parsed command names.

```
public class AppCommandFactory : CommandFactory, ICommandFactory
```

Inheritance

[object](#) ← CommandFactory ← AppCommandFactory

Implements

ICommandFactory

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

This factory extends BOOSE.CommandFactory and injects a shared BOOSE.ICanvas instance into all drawing-related commands so they can render onto the same canvas.

Constructors

AppCommandFactory(ICanvas)

Initializes a new instance of the [AppCommandFactory](#) class.

```
public AppCommandFactory(ICanvas canvas)
```

Parameters

canvas ICanvas

The canvas implementation that drawing commands will operate on.

Methods

MakeCommand(string)

Creates a concrete BOOSE.ICommand instance for the given command name.

```
public override ICommand MakeCommand(string rawName)
```

Parameters

rawName [string](#)

The raw command name as parsed from the BOOSE program text.

Returns

ICommand

An initialized BOOSE.ICommand corresponding to the command name.

Exceptions

CommandException

Thrown when the command name is null, empty, or consists only of whitespace.

Class ArrayCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **array** command, allowing declaration and management of integer or real arrays within a BOOSE program.

```
public class ArrayCommand : Evaluation, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← ArrayCommand

Implements

ICommand

Inherited Members

Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
[Evaluation.ProcessExpression\(string\)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Value ,
Evaluation.Local , Command.IsDouble , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , [Command.ProcessParameters\(string\)](#) ,
Command.ToString() , Command.Program , Command.Name , Command.ParameterList ,
Command.Parameters , Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#)

Remarks

This command supports one-dimensional arrays of type **int** or **real**. The array size is evaluated at compile time using the BOOSE expression system.

Properties

Size

Gets the size of the array.

```
public int Size { get; }
```

Property Value

[int](#)

Methods

CheckParameters(string[])

Validates parameters for the command.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

The parameter list to validate.

Remarks

Parameter validation is handled during [Set\(StoredProgram, string\)](#) and [Compile\(\)](#).

Compile()

Compiles the array command by evaluating its size and allocating storage.

```
public override void Compile()
```

Exceptions

[CommandException](#)

Thrown if the evaluated array size is less than or equal to zero.

Execute()

Executes the array command.

```
public override void Execute()
```

Remarks

Array declarations do not perform any runtime action.

GetIntArray(int)

Retrieves a value from an integer array.

```
public int GetIntArray(int index)
```

Parameters

index [int](#)

The zero-based array index.

Returns

[int](#)

The integer value at the specified index.

Exceptions

CommandException

Thrown if this is not an integer array or the index is out of range.

GetRealArray(int)

Retrieves a value from a real (double) array.

```
public double GetRealArray(int index)
```

Parameters

`index` [int](#)

The zero-based array index.

Returns

[double](#)

The real value at the specified index.

Exceptions

`CommandException`

Thrown if this is not a real array or the index is out of range.

IsIntArray()

Determines whether this array is an integer array.

`public bool IsIntArray()`

Returns

[bool](#)

`true` if the array type is `int`; otherwise, `false`.

IsRealArray()

Determines whether this array is a real (double) array.

`public bool IsRealArray()`

Returns

[bool](#)

`true` if the array type is `real`; otherwise, `false`.

Set(StoredProgram, string)

Parses and initializes the array declaration command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current BOOSE.StoredProgram instance.

Params string

The parameter string in the form: <int|real> <name> <size>.

Exceptions

CommandException

Thrown when parameters are missing, malformed, or the array type is unsupported.

SetIntArray(int, int)

Sets a value in an integer array.

```
public void SetIntArray(int val, int index)
```

Parameters

val int

The value to store.

index int

The zero-based array index.

Exceptions

CommandException

Thrown if this is not an integer array or the index is out of range.

SetRealArray(double, int)

Sets a value in a real (double) array.

```
public void SetRealArray(double val, int index)
```

Parameters

val [double](#)

The value to store.

index [int](#)

The zero-based array index.

Exceptions

CommandException

Thrown if this is not a real array or the index is out of range.

Class AssignCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `assign` command, which assigns the result of an expression to an existing variable.

```
public class AssignCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← AssignCommand

Implements

ICommand

Inherited Members

Command.IsDBNull , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString()
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

The BOOSE parser rewrites assignment statements of the form `x = expr` into `assign x = expr` before execution. This command supports assignment to `int`, `real`, and `boolean` variables.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

The list of parameters supplied to the command.

Remarks

Parameter validation is handled during [Set\(StoredProgram, string\)](#) and [Compile\(\)](#), so no additional checks are required here.

Compile()

Performs compile-time validation of the assignment.

```
public override void Compile()
```

Exceptions

CommandException

Thrown if the target variable does not exist.

Execute()

Executes the assignment by evaluating the expression and storing the result in the target variable.

```
public override void Execute()
```

Exceptions

CommandException

Thrown if the variable does not exist or is of an unsupported type.

Set(StoredProgram, string)

Parses and initializes the assignment command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program `StoredProgram`

The current BOOSE.StoredProgram instance.

Params `string` ↗

The parameter string in the form `<var> = <expr>`.

Exceptions

`CommandException`

Thrown if the assignment syntax is invalid or incomplete.

Class BooleanCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **boolean** command, which declares and manages boolean variables within a BOOSE program.

```
public class BooleanCommand : Evaluation, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← BooleanCommand

Implements

ICommand

Inherited Members

Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
[Evaluation.ProcessExpression\(string\)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Local ,
Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.Program , Command.Name ,
Command.ParameterList , Command.Parameters , Command.Paramsint , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

A boolean variable may be declared with or without an initial value. If no expression is provided, the value defaults to **false**. Boolean values are internally represented as integers where **true** maps to **1** and **false** maps to **0**.

Properties

BoolValue

Gets or sets the current boolean value of the variable.

```
public bool BoolValue { get; set; }
```

Property Value

[bool](#) ↗

Value

Gets the integer representation of the boolean value.

```
public override int Value { get; }
```

Property Value

[int](#) ↗

Remarks

This allows boolean variables to participate in numeric expressions.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#) ↗ []

The parameter list supplied to the command.

Remarks

All validation is handled during [Set\(StoredProgram, string\)](#) and [Compile\(\)](#), so no additional checks are required.

Compile()

Compiles the boolean declaration by registering the variable and evaluating its initial value.

```
public override void Compile()
```

Remarks

If the variable does not already exist in the program, it is added during compilation.

Execute()

Evaluates the boolean expression and updates the variable value.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the boolean declaration command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current BOOSE.StoredProgram instance.

Params string ↗

The parameter string in the form <name> or <name> = <expr>.

Exceptions

CommandException

Thrown when the parameter list is empty or malformed.

ToString()

Returns the string representation of the boolean value.

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

Returns

string ↗

"**true**" if the value is true; otherwise, "**false**".

Class CallCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `call` command, which invokes a user-defined method and passes arguments to it.

```
public class CallCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← CallCommand

Implements

ICommand

Inherited Members

Command.IsDBNull , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

This command requires an [ExtendedStoredProgram](#) in order to support method lookup, parameter passing, and return handling. The call operation sets up parameter variables, jumps to the method body, and stores the return address on the program stack.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

The parameter list supplied to the command.

Remarks

All validation is handled during execution.

Compile()

Performs compile-time validation for the call command.

```
public override void Compile()
```

Remarks

Method existence and parameter validation are deferred until execution because methods may be declared later in the program.

Execute()

Executes the method call by binding arguments, setting up the call stack, and transferring control to the method body.

```
public override void Execute()
```

Exceptions

StoredProgramException

Thrown if the current program does not support method execution.

CommandException

Thrown if the method does not exist, parameter counts do not match, or the method definition cannot be retrieved.

Set(StoredProgram, string)

Parses and initializes the method call command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current BOOSE.StoredProgram instance.

Params [string](#)

The parameter string in the form <methodName> [arg1 arg2 ...].

Exceptions

CommandException

Thrown when the method name is missing or parameters are malformed.

Class CircleCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `circle` command, which draws a circle on the canvas using the current pen settings.

```
public class CircleCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← CircleCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString()
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

The radius of the circle is provided as an integer expression and is evaluated at runtime. The circle is drawn at the current canvas position.

Constructors

CircleCommand(ICanvas)

Initializes a new instance of the [CircleCommand](#) class.

```
public CircleCommand(ICanvas canvas)
```

Parameters

`canvas` ICanvas

The canvas used to render the circle.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

The parameter list supplied to the command.

Remarks

Parameter validation is handled during [Set\(StoredProgram, string\)](#).

Compile()

Performs compile-time validation for the command.

```
public override void Compile()
```

Remarks

No compile-time checks are required for the circle command.

Execute()

Executes the circle command by evaluating the radius expression and drawing the circle on the canvas.

```
public override void Execute()
```

Exceptions

CommandException

Thrown if the evaluated radius is less than or equal to zero.

Set(StoredProgram, string)

Parses and initializes the circle drawing command.

```
public override void Set(StoredProgram program, string param)
```

Parameters

program StoredProgram

The current BOOSE.StoredProgram instance.

param string ↗

The parameter string representing the circle radius expression.

Exceptions

CommandException

Thrown when the radius parameter is missing or invalid.

Class ClearCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Command that clears the canvas to its default background color.

```
public class ClearCommand : ICommand
```

Inheritance

[object](#) ← ClearCommand

Implements

ICommand

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

ClearCommand(ICanvas)

Creates a new instance of the [ClearCommand](#) class.

```
public ClearCommand(ICanvas canvas)
```

Parameters

canvas ICanvas

The canvas that will be cleared.

Methods

CheckParameters(string[])

Validates that no parameters were provided.

```
public void CheckParameters(string[] Parameters)
```

Parameters

Parameters [string](#)[]

Array of parameters.

Exceptions

CommandEvent

Thrown when parameters exist.

Compile()

No compile-time behavior is needed for the clear command.

```
public void Compile()
```

Execute()

Executes the clear operation on the canvas.

```
public void Execute()
```

Set(StoredProgram, string)

Sets command configuration and validates parameters. The clear command does not support parameters.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

Stored program context (ignored).

Params [string](#)

Parameter string (should be empty).

Exceptions

CommandEvent

Thrown when parameters are supplied.

Class DrawToCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `drawto` command, which draws a line from the current canvas position to a specified coordinate.

```
public class DrawToCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← DrawToCommand

Implements

ICommand

Inherited Members

Command.IsDBNull , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString()
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

The target coordinates are provided as integer expressions and are evaluated at runtime. After drawing, the current canvas position is updated to the new location.

Constructors

DrawToCommand(ICanvas)

Initializes a new instance of the [DrawToCommand](#) class.

```
public DrawToCommand(ICanvas canvas)
```

Parameters

`canvas` ICanvas

The canvas used to render the line.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

The parameter list supplied to the command.

Remarks

Parameter validation is handled during [Set\(StoredProgram, string\)](#).

Compile()

Performs compile-time validation for the command.

```
public override void Compile()
```

Remarks

No compile-time checks are required for the draw-to command.

Execute()

Executes the draw-to command by evaluating the coordinate expressions and drawing a line to the specified point.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the draw-to command.

```
public override void Set(StoredProgram program, string param)
```

Parameters

program StoredProgram

The current BOOSE.StoredProgram instance.

param [string](#)

The parameter string containing the target X and Y expressions.

Exceptions

CommandException

Thrown when the parameter list does not contain exactly two values.

Class ElseCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **else** command.

```
public class ElseCommand : Command, ICommand
```

Inheritance

[object](#) ↗ ← Command ← ElseCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) ↗ , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗

Remarks

The **else** command links itself to the most recent [IfCommand](#) on the compile stack. At runtime, when execution reaches **else**, it jumps to the command immediately after the matching **endif** (thereby skipping the else block when the IF condition was true).

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#) ↗ []

The tokenised parameter list.

Compile()

Links this `else` command with the most recent [IfCommand](#) on the compile stack and records its position in that `if`.

```
public override void Compile()
```

Exceptions

CommandException

Thrown when `else` is used without a matching `if`, or when the program is not an [ExtendedStoredProgram](#).

Execute()

Executes the `else` command by jumping to the first command after the matching `endif`.

```
public override void Execute()
```

Exceptions

StoredProgramException

Thrown if this `else` is not linked to a matching `endif`.

Set(StoredProgram, string)

Parses and initializes the `else` command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

`Program` StoredProgram

The current program instance.

Params [string](#)

Expected to be empty.

Exceptions

CommandException

Thrown when unexpected parameters are supplied.

SetEndIfIndex(int)

Sets the index of the matching `endif`.

```
public void SetEndIfIndex(int idx)
```

Parameters

`idx` [int](#)

The index of the `endif` command.

Class EndForCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `endfor` command.

```
public class EndForCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← EndForCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

Compile()

Links this command to its corresponding [ForCommand](#).

```
public override void Compile()
```

Execute()

Increments the loop variable and jumps back to the matching `for`.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses the endfor command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

`Program` StoredProgram

`Params` [string](#)

Class EndIfCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `endif` command.

```
public class EndIfCommand : Command, ICommand
```

Inheritance

[object](#) ↗ ← Command ← EndIfCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) ↗ , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗

Remarks

The `endif` command completes compile-time linking by popping the matching [IfCommand](#) from the compile stack, setting its end index, and (when present) linking the corresponding [ElseCommand](#) to this `endif`.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

`parameterList` [string](#) ↗ []

The tokenised parameter list.

Compile()

Links the matching `if` and optional `else` commands.

```
public override void Compile()
```

Exceptions

CommandException

Thrown when `endif` is used without a matching `if`, or when the program is not an [ExtendedStoredProgram](#).

Execute()

Executes the `endif` command.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the `endif` command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

`Program` StoredProgram

The current program instance.

`Params` [string](#)

Optional parameter "if" may be supplied (e.g. `end if`).

Exceptions

CommandException

Thrown when unexpected parameters are supplied.

Class EndMethodCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `endmethod` command.

```
public class EndMethodCommand : Command, ICommand
```

Inheritance

[object](#) ↗ ← Command ← EndMethodCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) ↗ , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#) ↗ []

Compile()

Links this command to its corresponding [MethodCommand](#).

```
public override void Compile()
```

Execute()

Returns execution to the caller after method completion.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses the endmethod command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

Params string ↗

Class EndWhileCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `endwhile` command.

```
public class EndWhileCommand : Command, ICommand
```

Inheritance

[object](#) ↗ ← Command ← EndWhileCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) ↗ , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ ,
[object.GetType\(\)](#) ↗ , [object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#) ↗ []

Compile()

Links this command to its matching [WhileCommand](#).

```
public override void Compile()
```

Execute()

Jumps execution back to the matching `while` command.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses the endwhile command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

`Program` StoredProgram

The current program instance.

`Params` `string` ↗

The parameter string.

Class ExtendedParser

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Extends the BOOSE.Boose.Parser to support additional syntax features, improved error aggregation, and integration with [ExtendedStoredProgram](#).

```
public class ExtendedParser : Parser, IParser
```

Inheritance

[object](#) ← Parser ← ExtendedParser

Implements

IParser

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

ExtendedParser(CommandFactory, ExtendedStoredProgram)

Initializes a new instance of the [ExtendedParser](#) class.

```
public ExtendedParser(CommandFactory factory, ExtendedStoredProgram program)
```

Parameters

factory CommandFactory

The BOOSE.CommandFactory used to create command instances.

program [ExtendedStoredProgram](#)

The [ExtendedStoredProgram](#) that receives parsed commands.

Exceptions

[ArgumentNullException](#)

Thrown when `factory` or `program` is null.

Methods

ParseCommand(string)

Parses a single line of BOOSE source code into an BOOSE.ICommand.

```
public override ICommand ParseCommand(string line)
```

Parameters

`line` [string](#)

The line of source text to parse.

Returns

ICommand

A fully initialized BOOSE.ICommand instance, or `null` if the line is empty or a comment.

ParseProgram(string)

Parses the supplied program text into commands, compiles them, and stores them in the associated [ExtendedStoredProgram](#).

```
public override void ParseProgram(string programText)
```

Parameters

`programText` [string](#)

The BOOSE program source text.

Exceptions

ParserException

Thrown when one or more lines fail to parse or compile. The exception message contains aggregated errors with line numbers.

Class ExtendedStoredProgram

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Extends BOOSE.StoredProgram to provide full runtime support for methods, variables, call stacks, and execution control.

```
public class ExtendedStoredProgram : StoredProgram, IList, ICollection, IEnumerable,  
ICloneable, IStoredProgram
```

Inheritance

[object](#) ← [ArrayList](#) ← [StoredProgram](#) ← [ExtendedStoredProgram](#)

Implements

[IList](#), [ICollection](#), [IEnumerable](#), [ICloneable](#), [IStoredProgram](#)

Inherited Members

[StoredProgram.IsValidProgram\(\)](#), [StoredProgram.SetSyntaxStatus\(bool\)](#),
[StoredProgram.AddMethod\(Method\)](#), [StoredProgram.GetMethod\(string\)](#),
[StoredProgram.GetVariable\(int\)](#), [StoredProgram.FindVariable\(Evaluation\)](#),
[StoredProgram.FindVariable\(string\)](#), [StoredProgram.GetVarValue\(string\)](#),
[StoredProgram.UpdateVariable\(string, int\)](#), [StoredProgram.UpdateVariable\(string, double\)](#),
[StoredProgram.UpdateVariable\(string, bool\)](#), [StoredProgram.DeleteVariable\(string\)](#),
[StoredProgram.IsExpression\(string\)](#), [StoredProgram.EvaluateExpressionWithString\(string\)](#),
[StoredProgram.EvaluateExpression\(string\)](#), [StoredProgram.Push\(ConditionalCommand\)](#),
[StoredProgram.Pop\(\)](#), [StoredProgram.Add\(Command\)](#), [StoredProgram.NextCommand\(\)](#),
[StoredProgram.ResetProgram\(\)](#), [StoredProgram.Commandsleft\(\)](#), [ArrayList.Adapter\(IList\)](#),
[ArrayList.Add\(object\)](#), [ArrayList.AddRange\(ICollection\)](#),
[ArrayList.BinarySearch\(int, int, object, IComparer\)](#), [ArrayList.BinarySearch\(object\)](#),
[ArrayList.BinarySearch\(object, IComparer\)](#), [ArrayList.Clone\(\)](#), [ArrayList.Contains\(object\)](#),
[ArrayList.CopyTo\(Array\)](#), [ArrayList.CopyTo\(Array, int\)](#), [ArrayList.CopyTo\(int, Array, int, int\)](#),
[ArrayList.FixedSize\(ArrayList\)](#), [ArrayList.FixedSize\(IList\)](#), [ArrayList.GetEnumerator\(\)](#),
[ArrayList.GetEnumerator\(int, int\)](#), [ArrayList.GetRange\(int, int\)](#), [ArrayList.IndexOf\(object\)](#),
[ArrayList.IndexOf\(object, int\)](#), [ArrayList.IndexOf\(object, int, int\)](#), [ArrayList.Insert\(int, object\)](#),
[ArrayList.InsertRange\(int, ICollection\)](#), [ArrayList.LastIndexOf\(object\)](#),
[ArrayList.LastIndexOf\(object, int\)](#), [ArrayList.LastIndexOf\(object, int, int\)](#),
[ArrayList.ReadOnly\(ArrayList\)](#), [ArrayList.ReadOnly\(IList\)](#), [ArrayList.Remove\(object\)](#),
[ArrayList.RemoveAt\(int\)](#), [ArrayList.RemoveRange\(int, int\)](#), [ArrayList.Repeat\(object, int\)](#),
[ArrayList.Reverse\(\)](#), [ArrayList.Reverse\(int, int\)](#), [ArrayList.SetRange\(int, ICollection\)](#),

[ArrayList.Sort\(\)](#) , [ArrayList.Sort\(IComparer\)](#) , [ArrayList.Sort\(int, int, IComparer\)](#) ,
[ArrayList.Synchronized\(ArrayList\)](#) , [ArrayList.Synchronized\(IList\)](#) , [ArrayList.ToArray\(\)](#) ,
[ArrayList.ToArray\(Type\)](#) , [ArrayList.TrimToSize\(\)](#) , [ArrayList.Capacity](#) , [ArrayList.IsFixedSize](#) ,
[ArrayList.IsReadOnly](#) , [ArrayList.IsSynchronized](#) , [ArrayList.this\[int\]](#) , [ArrayList.SyncRoot](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

ExtendedStoredProgram(ICanvas)

Initializes a new instance of the [ExtendedStoredProgram](#) class.

```
public ExtendedStoredProgram(ICanvas canvas)
```

Parameters

canvas ICanvas

The canvas used for drawing commands.

Properties

Count

Gets the total number of commands in the program.

```
public int Count { get; }
```

Property Value

[int](#)

PC

Gets the current program counter.

```
public int PC { get; }
```

Property Value

[int ↗](#)

Methods

Add(ICommand)

Adds a command to the program.

```
public void Add(ICommand cmd)
```

Parameters

cmd ICommand

The command to add.

AddVariable(Evaluation)

Adds a variable to the program.

```
public override void AddVariable(Evaluation v)
```

Parameters

v Evaluation

The variable to add.

Clear()

Clears all runtime state including commands, variables, and stacks.

```
public void Clear()
```

GetCommandAt(int)

Retrieves the command at the specified index.

```
public ICommand GetCommandAt(int index)
```

Parameters

`index` [int](#)

The command index.

Returns

`ICommand`

The command at the given index.

GetMethodstartIndex(string)

Gets the starting command index for a method.

```
public int GetMethodstartIndex(string methodName)
```

Parameters

`methodName` [string](#)

The method name.

Returns

[int](#)

The command index where the method begins.

GetVariable(string)

Retrieves a variable by name.

```
public override Evaluation GetVariable(string name)
```

Parameters

name [string](#)

The variable name.

Returns

Evaluation

The variable instance, or [null](#) if not found.

Jump(int)

Sets the program counter to a new position.

```
public void Jump(int target)
```

Parameters

target [int](#)

The target command index.

MethodExists(string)

Determines whether a method with the given name exists.

```
public bool MethodExists(string methodName)
```

Parameters

`methodName` [string](#)

The method name.

Returns

[bool](#)

`true` if the method exists; otherwise, `false`.

PopCompile()

Pops the most recent object from the compile-time stack.

`public object PopCompile()`

Returns

[object](#)

The popped object.

PopReturn()

Pops the most recent return address from the call stack.

`public int PopReturn()`

Returns

[int](#)

The return address.

PushCompile(object)

Pushes an object onto the compile-time stack.

```
public void PushCompile(object o)
```

Parameters

o [object](#)

The object to push.

PushReturn(int)

Pushes a return address onto the call stack.

```
public void PushReturn(int address)
```

Parameters

address [int](#)

The address to return to.

RegisterMethod(string, int)

Registers a method name with its starting command index.

```
public void RegisterMethod(string methodName, int startIndex)
```

Parameters

methodName [string](#)

The method name.

startIndex [int](#)

The index where the method begins.

Run()

Executes the program from the beginning until completion.

```
public override void Run()
```

SetOrCreateNumeric(string, double, string)

Sets or creates a numeric variable used during method calls.

```
public void SetOrCreateNumeric(string name, double value, string type)
```

Parameters

name [string](#)

The variable name.

value [double](#)

The numeric value.

type [string](#)

The variable type ([int](#), [real](#), or [boolean](#)).

VariableExists(string)

Determines whether a variable with the given name exists.

```
public override bool VariableExists(string name)
```

Parameters

name [string](#)

The variable name.

Returns

[bool](#)

`true` if the variable exists; otherwise, `false`.

Class ForCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **for** loop command.

```
public class ForCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← ForCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

Compile()

Registers the loop with the compile stack.

```
public override void Compile()
```

Execute()

Executes the loop condition and initialisation logic.

```
public override void Execute()
```

GetForIndex()

Gets the index of this **for** command.

```
public int GetForIndex()
```

Returns

[int](#)

The command index.

Set(StoredProgram, string)

Parses the for-loop declaration.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current program.

Params [string](#)

The loop declaration parameters.

SetEndForIndex(int)

Sets the index of the matching `endfor` command.

```
public void SetEndForIndex(int idx)
```

Parameters

`idx` [int](#)

The command index.

Class Form1

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Main UI form for the BOOSE drawing interpreter.

```
public class Form1 : Form, IDropTarget, ISynchronizeInvoke, IWin32Window,  
IBindableComponent, IComponent, IDisposable, IContainerControl
```

Inheritance

```
object ↳ ← MarshalByRefObject ↳ ← Component ↳ ← Control ↳ ← ScrollableControl ↳ ←  
ContainerControl ↳ ← Form ↳ ← Form1
```

Implements

```
IDropTarget ↳ , ISynchronizeInvoke ↳ , IWin32Window ↳ , IBindableComponent ↳ , IComponent ↳ ,  
IDisposable ↳ , IContainerControl ↳
```

Inherited Members

```
Form.SetVisibleCore(bool) ↳ , Form.Activate() ↳ , Form.ActivateMdiChild(Form) ↳ ,  
Form.AddOwnedForm(Form) ↳ , Form.AdjustFormScrollbars(bool) ↳ , Form.Close() ↳ ,  
Form.CreateAccessibilityInstance() ↳ , Form.CreateControlsInstance() ↳ , Form.CreateHandle() ↳ ,  
Form.DefWndProc(ref Message) ↳ , Form.ProcessMnemonic(char) ↳ , Form.CenterToParent() ↳ ,  
Form.CenterToScreen() ↳ , Form.LayoutMdi(MdiLayout) ↳ , Form.OnActivated(EventArgs) ↳ ,  
Form.OnBackgroundImageChanged(EventArgs) ↳ ,  
Form.OnBackgroundImageLayoutChanged(EventArgs) ↳ , Form.OnClosing(CancelEventArgs) ↳ ,  
Form.OnClosed(EventArgs) ↳ , Form.OnFormClosing(FormClosingEventArgs) ↳ ,  
Form.OnFormClosed(FormClosedEventArgs) ↳ , Form.OnCreateControl() ↳ ,  
Form.OnDeactivate(EventArgs) ↳ , Form.OnEnabledChanged(EventArgs) ↳ , Form.OnEnter(EventArgs) ↳ ,  
Form.OnFontChanged(EventArgs) ↳ , Form.OnGotFocus(EventArgs) ↳ ,  
Form.OnHandleCreated(EventArgs) ↳ , Form.OnHandleDestroyed(EventArgs) ↳ ,  
Form.OnHelpButtonClicked(CancelEventArgs) ↳ , Form.OnLayout(LayoutEventArgs) ↳ ,  
Form.OnLoad(EventArgs) ↳ , Form.OnMaximizedBoundsChanged(EventArgs) ↳ ,  
Form.OnMaximumSizeChanged(EventArgs) ↳ , Form.OnMinimumSizeChanged(EventArgs) ↳ ,  
Form.OnInputLanguageChanged(InputLanguageChangedEventArgs) ↳ ,  
Form.OnInputLanguageChanging(InputLanguageChangingEventArgs) ↳ ,  
Form.OnVisibleChanged(EventArgs) ↳ , Form.OnMdiChildActivate(EventArgs) ↳ ,  
Form.OnMenuStart(EventArgs) ↳ , Form.OnMenuComplete(EventArgs) ↳ ,  
Form.OnPaint(PaintEventArgs) ↳ , Form.OnResize(EventArgs) ↳ ,  
Form.OnDpiChanged(DpiChangedEventArgs) ↳ , Form.OnGetDpiScaledSize(int, int, ref Size) ↳ ,
```

[Form.OnRightToLeftLayoutChanged\(EventArgs\)](#) , [Form.OnShown\(EventArgs\)](#) ,
[Form.OnTextChanged\(EventArgs\)](#) , [Form.ProcessCmdKey\(ref Message, Keys\)](#) ,
[Form.ProcessDialogKey\(Keys\)](#) , [Form.ProcessDialogChar\(char\)](#) ,
[Form.ProcessKeyPreview\(ref Message\)](#) , [Form.ProcessTabKey\(bool\)](#) ,
[Form.RemoveOwnedForm\(Form\)](#) , [Form.Select\(bool, bool\)](#) ,
[Form.ScaleMinAxisSize\(float, float, bool\)](#) ,
[Form.GetScaledBounds\(Rectangle, SizeF, BoundsSpecified\)](#) ,
[Form.ScaleControl\(SizeF, BoundsSpecified\)](#) , [Form.SetBoundsCore\(int, int, int, int, BoundsSpecified\)](#) ,
[Form.SetClientSizeCore\(int, int\)](#) , [Form.SetDesktopBounds\(int, int, int, int\)](#) ,
[Form.SetDesktopLocation\(int, int\)](#) , [Form.Show\(IWin32Window\)](#) , [Form.ShowDialog\(\)](#) ,
[Form.ShowDialog\(IWin32Window\)](#) , [Form.ToString\(\)](#) , [Form.UpdateDefaultButton\(\)](#) ,
[Form.OnResizeBegin\(EventArgs\)](#) , [Form.OnResizeEnd\(EventArgs\)](#) ,
[Form.OnStyleChanged\(EventArgs\)](#) , [Form.ValidateChildren\(\)](#) ,
[Form.ValidateChildren\(ValidationConstraints\)](#) , [Form.WndProc\(ref Message\)](#) , [Form.AcceptButton](#) ,
[Form.ActiveForm](#) , [Form.ActiveMdiChild](#) , [Form.AllowTransparency](#) , [Form.AutoScroll](#) ,
[Form.AutoSize](#) , [Form.AutoSizeMode](#) , [Form.AutoValidate](#) , [Form.BackColor](#) ,
[Form.FormBorderStyle](#) , [Form.CancelButton](#) , [Form.ClientSize](#) , [Form.ControlBox](#) ,
[Form.CreateParams](#) , [Form.DefaultImeMode](#) , [Form.DefaultSize](#) , [Form.DesktopBounds](#) ,
[Form/DesktopLocation](#) , [Form/DialogResult](#) , [Form/HelpButton](#) , [Form/Icon](#) , [Form/IsMdiChild](#) ,
[Form/IsMdiContainer](#) , [Form/IsRestrictedWindow](#) , [Form/KeyPreview](#) , [Form/Location](#) ,
[Form/MaximizedBounds](#) , [Form/MaximumSize](#) , [Form/MainMenuStrip](#) , [Form/MinimumSize](#) ,
[Form/MaximizeBox](#) , [Form/MdiChildren](#) , [Form/MdiChildrenMinimizedAnchorBottom](#) ,
[Form/MdiParent](#) , [Form/MinimizeBox](#) , [Form/Modal](#) , [Form/Opacity](#) , [Form/OwnedForms](#) ,
[Form/Owner](#) , [Form/RestoreBounds](#) , [Form/RightToLeftLayout](#) , [Form>ShowInTaskbar](#) ,
[Form>ShowIcon](#) , [Form>ShowWithoutActivation](#) , [Form/Size](#) , [Form/SizeGripStyle](#) ,
[Form/StartPosition](#) , [Form/Text](#) , [Form/TopLevel](#) , [Form/TopMost](#) , [Form/TransparencyKey](#) ,
[Form/WindowState](#) , [Form/AutoSizeChanged](#) , [Form/AutoValidateChanged](#) ,
[Form/HelpButtonClicked](#) , [Form/MaximizedBoundsChanged](#) , [Form/MaximumSizeChanged](#) ,
[Form/MinimumSizeChanged](#) , [Form/Activated](#) , [Form/Deactivate](#) , [Form/FormClosing](#) ,
[Form/FormClosed](#) , [Form/Load](#) , [Form/MdiChildActivate](#) , [Form/MenuComplete](#) ,
[Form/MenuStart](#) , [Form/InputLanguageChanged](#) , [Form/InputLanguageChanging](#) ,
[Form/RightToLeftLayoutChanged](#) , [Form/Shown](#) , [Form/DpiChanged](#) , [Form/ResizeBegin](#) ,
[Form/ResizeEnd](#) , [ContainerControl.OnAutoValidateChanged\(EventArgs\)](#) ,
[ContainerControl.OnMove\(EventArgs\)](#) , [ContainerControl.OnParentChanged\(EventArgs\)](#) ,
[ContainerControl.PerformLayout\(\)](#) , [ContainerControl.RescaleConstantsForDpi\(int, int\)](#) ,
[ContainerControl.Validate\(\)](#) , [ContainerControl.Validate\(bool\)](#) ,
[ContainerControl.AutoScaleDimensions](#) , [ContainerControl.AutoScaleFactor](#) ,
[ContainerControl.AutoScaleMode](#) , [ContainerControl.BindingContext](#) ,
[ContainerControl.CanEnableIme](#) , [ContainerControl.ActiveControl](#) ,
[ContainerControl.CurrentAutoScaleDimensions](#) , [ContainerControl.ParentForm](#) ,

[ScrollableControl.ScrollStateAutoScrolling](#) , [ScrollableControl.ScrollStateHScrollVisible](#) ,
[ScrollableControl.ScrollStateVScrollVisible](#) , [ScrollableControl.ScrollStateUserHasScrolled](#) ,
[ScrollableControl.ScrollStateFullDrag](#) , [ScrollableControl.GetScrollState\(int\)](#) ,
[ScrollableControl.OnMouseWheel\(MouseEventArgs\)](#) ,
[ScrollableControl.OnRightToLeftChanged\(EventArgs\)](#) ,
[ScrollableControl.OnPaintBackground\(PaintEventArgs\)](#) ,
[ScrollableControl.OnPaddingChanged\(EventArgs\)](#) , [ScrollableControl.SetDisplayRectLocation\(int, int\)](#) ,
[ScrollableControl.ScrollControlIntoView\(Control\)](#) , [ScrollableControl.ScrollToControl\(Control\)](#) ,
[ScrollableControl.OnScroll\(ScrollEventArgs\)](#) , [ScrollableControl.SetAutoScrollMargin\(int, int\)](#) ,
[ScrollableControl.SetScrollState\(int, bool\)](#) , [ScrollableControl.AutoScrollMargin](#) ,
[ScrollableControl.AutoScrollPosition](#) , [ScrollableControl.AutoScrollMinSize](#) ,
[ScrollableControl.DisplayRectangle](#) , [ScrollableControl.HScroll](#) , [ScrollableControl.HorizontalScroll](#) ,
[ScrollableControl.VScroll](#) , [ScrollableControl.VerticalScroll](#) , [ScrollableControl.Scroll](#) ,
[Control.GetAccessibilityObjectById\(int\)](#) , [Control.SetAutoSizeMode\(AutoSizeMode\)](#) ,
[Control.GetAutoSizeMode\(\)](#) , [Control.GetPreferredSize\(Size\)](#) ,
[Control.AccessibilityNotifyClients\(AccessibleEvents, int\)](#) ,
[Control.AccessibilityNotifyClients\(AccessibleEvents, int, int\)](#) , [Control.BeginInvoke\(Delegate\)](#) ,
[Control.BeginInvoke\(Action\)](#) , [Control.BeginInvoke\(Delegate, params object\[\]\)](#) ,
[Control.BringToFront\(\)](#) , [Control.Contains\(Control\)](#) , [Control.CreateGraphics\(\)](#) ,
[Control.CreateControl\(\)](#) , [Control.DestroyHandle\(\)](#) , [Control.DoDragDrop\(object, DragDropEffects\)](#) ,
[Control.DoDragDrop\(object, DragDropEffects, Bitmap, Point, bool\)](#) ,
[Control.DrawToBitmap\(Bitmap, Rectangle\)](#) , [Control.EndInvoke\(IAsyncResult\)](#) , [Control.FindForm\(\)](#) ,
[Control.GetTopLevel\(\)](#) , [Control.RaiseKeyEvent\(object, KeyEventArgs\)](#) ,
[Control.RaiseMouseEvent\(object, MouseEventArgs\)](#) , [Control.Focus\(\)](#) ,
[Control.FromChildHandle\(nint\)](#) , [Control.FromHandle\(nint\)](#) ,
[Control.GetChildAtPoint\(Point, GetChildAtPointSkip\)](#) , [Control.GetChildAtPoint\(Point\)](#) ,
[Control.GetContainerControl\(\)](#) , [Control.GetNextControl\(Control, bool\)](#) ,
[Control.GetStyle\(ControlStyles\)](#) , [Control.Hide\(\)](#) , [Control.InitLayout\(\)](#) , [Control.Invalidate\(Region\)](#) ,
[Control.Invalidate\(Region, bool\)](#) , [Control.Invalidate\(\)](#) , [Control.Invalidate\(bool\)](#) ,
[Control.Invalidate\(Rectangle\)](#) , [Control.Invalidate\(Rectangle, bool\)](#) , [Control.Invoke\(Action\)](#) ,
[Control.Invoke\(Delegate\)](#) , [Control.Invoke\(Delegate, params object\[\]\)](#) ,
[Control.Invoke<T>\(Func<T>\)](#) , [Control.InvokePaint\(Control, PaintEventArgs\)](#) ,
[Control.InvokePaintBackground\(Control, PaintEventArgs\)](#) , [Control.IsKeyLocked\(Keys\)](#) ,
[Control.IsInputChar\(char\)](#) , [Control.IsInputKey\(Keys\)](#) , [Control.IsMnemonic\(char, string\)](#) ,
[Control.LogicalToDeviceUnits\(int\)](#) , [Control.LogicalToDeviceUnits\(Size\)](#) ,
[Control.ScaleBitmapLogicalToDevice\(ref Bitmap\)](#) , [Control.NotifyInvalidate\(Rectangle\)](#) ,
[Control.InvokeOnClick\(Control, EventArgs\)](#) , [Control.OnAutoSizeChanged\(EventArgs\)](#) ,
[Control.OnBackColorChanged\(EventArgs\)](#) , [Control.OnBindingContextChanged\(EventArgs\)](#) ,
[Control.OnCausesValidationChanged\(EventArgs\)](#) , [Control.OnContextMenuStripChanged\(EventArgs\)](#) ,
[Control.OnCursorChanged\(EventArgs\)](#) , [Control.OnDataContextChanged\(EventArgs\)](#) ,

[Control.OnDockChanged\(EventArgs\)](#) , [Control.OnForeColorChanged\(EventArgs\)](#) ,
[Control.OnNotifyMessage\(Message\)](#) , [Control.OnParentBackColorChanged\(EventArgs\)](#) ,
[Control.OnParentBackgroundImageChanged\(EventArgs\)](#) ,
[Control.OnParentBindingContextChanged\(EventArgs\)](#) , [Control.OnParentCursorChanged\(EventArgs\)](#) ,
[Control.OnParentDataContextChanged\(EventArgs\)](#) , [Control.OnParentEnabledChanged\(EventArgs\)](#) ,
[Control.OnParentFontChanged\(EventArgs\)](#) , [Control.OnParentForeColorChanged\(EventArgs\)](#) ,
[Control.OnParentRightToLeftChanged\(EventArgs\)](#) , [Control.OnParentVisibleChanged\(EventArgs\)](#) ,
[Control.OnPrint\(PaintEventArgs\)](#) , [Control.OnTabIndexChanged\(EventArgs\)](#) ,
[Control.OnTabStopChanged\(EventArgs\)](#) , [Control.OnClick\(EventArgs\)](#) ,
[Control.OnClientSizeChanged\(EventArgs\)](#) , [Control.OnControlAdded\(ControlEventArgs\)](#) ,
[Control.OnControlRemoved\(ControlEventArgs\)](#) , [Control.OnLocationChanged\(EventArgs\)](#) ,
[Control.OnDoubleClick\(EventArgs\)](#) , [Control.OnDragEnter\(DragEventArgs\)](#) ,
[Control.OnDragOver\(DragEventArgs\)](#) , [Control.OnDragLeave\(EventArgs\)](#) ,
[Control.OnDragDrop\(DragEventArgs\)](#) , [Control.OnGiveFeedback\(GiveFeedbackEventArgs\)](#) ,
[Control.InvokeGotFocus\(Control, EventArgs\)](#) , [Control.OnHelpRequested\(HelpEventArgs\)](#) ,
[Control.OnInvalidate\(InvalidateEventArgs\)](#) , [Control.OnKeyDown\(KeyboardEventArgs\)](#) ,
[Control.OnKeyPress\(KeyPressEventArgs\)](#) , [Control.OnKeyUp\(KeyboardEventArgs\)](#) ,
[Control.OnLeave\(EventArgs\)](#) , [Control.InvokeLostFocus\(Control, EventArgs\)](#) ,
[Control.OnLostFocus\(EventArgs\)](#) , [Control.OnMarginChanged\(EventArgs\)](#) ,
[Control.OnMouseDoubleClick\(MouseEventArgs\)](#) , [Control.OnMouseClick\(MouseEventArgs\)](#) ,
[Control.OnMouseCaptureChanged\(EventArgs\)](#) , [Control.OnMouseDown\(MouseEventArgs\)](#) ,
[Control.OnMouseEnter\(EventArgs\)](#) , [Control.OnMouseLeave\(EventArgs\)](#) ,
[Control.OnDpiChangedBeforeParent\(EventArgs\)](#) , [Control.OnDpiChangedAfterParent\(EventArgs\)](#) ,
[Control.OnMouseHover\(EventArgs\)](#) , [Control.OnMouseMove\(MouseEventArgs\)](#) ,
[Control.OnMouseUp\(MouseEventArgs\)](#) ,
[Control.OnQueryContinueDrag\(QueryContinueDragEventArgs\)](#) ,
[Control.OnRegionChanged\(EventArgs\)](#) , [Control.OnPreviewKeyDown\(PreviewKeyDownEventArgs\)](#) ,
[Control.OnSizeChanged\(EventArgs\)](#) , [Control.OnChangeUICues\(UICuesEventArgs\)](#) ,
[Control.OnSystemColorsChanged\(EventArgs\)](#) , [Control.OnValidating\(CancelEventArgs\)](#) ,
[Control.OnValidated\(EventArgs\)](#) , [Control.PerformLayout\(\)](#) , [Control.PerformLayout\(Control, string\)](#) ,
[Control.PointToClient\(Point\)](#) , [Control.PointToScreen\(Point\)](#) ,
[Control.PreProcessMessage\(ref Message\)](#) , [Control.PreProcessControlMessage\(ref Message\)](#) ,
[Control.ProcessKeyEventArgs\(ref Message\)](#) , [Control.ProcessKeyMessage\(ref Message\)](#) ,
[Control.RaiseDragEvent\(object, DragEventArgs\)](#) , [Control.RaisePaintEvent\(object, PaintEventArgs\)](#) ,
[Control.RecreateHandle\(\)](#) , [Control.RectangleToClient\(Rectangle\)](#) ,
[Control.RectangleToScreen\(Rectangle\)](#) , [Control.ReflectMessage\(nint, ref Message\)](#) ,
[Control.Refresh\(\)](#) , [Control.ResetMouseEventArgs\(\)](#) , [Control.ResetText\(\)](#) , [Control.ResumeLayout\(\)](#) ,
[Control.ResumeLayout\(bool\)](#) , [Control.Scale\(SizeF\)](#) , [Control.Select\(\)](#) ,
[Control.SelectNextControl\(Control, bool, bool, bool, bool\)](#) , [Control.SendToBack\(\)](#) ,
[Control.SetBounds\(int, int, int, int\)](#) , [Control.SetBounds\(int, int, int, int, BoundsSpecified\)](#) ,

[Control.SizeFromClientSize\(Size\)](#) , [Control.SetStyle\(ControlStyles, bool\)](#) , [Control.SetTopLevel\(bool\)](#) ,
[Control.RtlTranslateAlignment\(HorizontalAlignment\)](#) ,
[Control.RtlTranslateAlignment\(LeftRightAlignment\)](#) ,
[Control.RtlTranslateAlignment\(ContentAlignment\)](#) ,
[Control.RtlTranslateHorizontal\(HorizontalAlignment\)](#) ,
[Control.RtlTranslateLeftRight\(LeftRightAlignment\)](#) , [Control.RtlTranslateContent\(ContentAlignment\)](#) ,
[Control.Show\(\)](#) , [Control.SuspendLayout\(\)](#) , [Control.Update\(\)](#) , [Control.UpdateBounds\(\)](#) ,
[Control.UpdateBounds\(int, int, int, int\)](#) , [Control.UpdateBounds\(int, int, int, int, int, int\)](#) ,
[Control.UpdateZOrder\(\)](#) , [Control.UpdateStyles\(\)](#) , [Control.OnImeModeChanged\(EventArgs\)](#) ,
[Control.AccessibilityObject](#) , [Control.AccessibleDefaultActionDescription](#) ,
[Control.AccessibleDescription](#) , [Control.AccessibleName](#) , [Control.AccessibleRole](#) ,
[Control.AllowDrop](#) , [Control.Anchor](#) , [Control.AutoScrollOffset](#) , [Control.LayoutEngine](#) ,
[Control.DataContext](#) , [Control.BackgroundImage](#) , [Control.BackgroundImageLayout](#) ,
[Control.Bottom](#) , [Control.Bounds](#) , [Control.CanFocus](#) , [Control.CanRaiseEvents](#) ,
[Control.CanSelect](#) , [Control.Capture](#) , [Control.CausesValidation](#) ,
[Control.CheckForIllegalCrossThreadCalls](#) , [Control.ClientRectangle](#) , [Control.CompanyName](#) ,
[Control.ContainsFocus](#) , [Control.ContextMenuStrip](#) , [Control.Controls](#) , [Control.Created](#) ,
[Control.Cursor](#) , [Control.DataBindings](#) , [Control.DefaultBackColor](#) , [Control.DefaultCursor](#) ,
[Control.DefaultFont](#) , [Control.DefaultForeColor](#) , [Control.DefaultMargin](#) ,
[Control.DefaultMaximumSize](#) , [Control.DefaultMinimumSize](#) , [Control.DefaultPadding](#) ,
[Control.DeviceDpi](#) , [Control.IsDisposed](#) , [Control.Disposing](#) , [Control.Dock](#) ,
[Control.DoubleBuffered](#) , [Control.Enabled](#) , [Control.Focused](#) , [Control.Font](#) ,
[Control.FontHeight](#) , [Control.ForeColor](#) , [Control.Handle](#) , [Control.HasChildren](#) , [Control.Height](#) ,
[Control.IsHandleCreated](#) , [Control.InvokeRequired](#) , [Control.IsAccessible](#) ,
[Control.IsAncestorSiteInDesignMode](#) , [Control.IsMirrored](#) , [Control.Left](#) , [Control.Margin](#) ,
[Control.ModifierKeys](#) , [Control.MouseButtons](#) , [Control.mousePosition](#) , [Control.Name](#) ,
[Control.Parent](#) , [Control.ProductName](#) , [Control.ProductVersion](#) , [Control.RecreatingHandle](#) ,
[Control.Region](#) , [Control.RenderRightToLeft](#) , [Control.ResizeRedraw](#) , [Control.Right](#) ,
[Control.RightToLeft](#) , [Control.ScaleChildren](#) , [Control.Site](#) , [Control.TabIndex](#) , [Control.TabStop](#) ,
[Control.Tag](#) , [Control.Top](#) , [Control.TopLevelControl](#) , [Control.ShowKeyboardCues](#) ,
[Control.ShowFocusCues](#) , [Control.UseWaitCursor](#) , [Control.Visible](#) , [Control.Width](#) ,
[Control.PreferredSize](#) , [Control.Padding](#) , [Control.ImeMode](#) , [Control.ImeModeBase](#) ,
[Control.PropagatingImeMode](#) , [Control.BackColorChanged](#) , [Control.BackgroundImageChanged](#) ,
[Control.BackgroundImageLayoutChanged](#) , [Control.BindingContextChanged](#) ,
[Control.CausesValidationChanged](#) , [Control.ClientSizeChanged](#) ,
[Control.ContextMenuStripChanged](#) , [Control.CursorChanged](#) , [Control.DockChanged](#) ,
[Control.EnabledChanged](#) , [Control.FontChanged](#) , [Control.ForeColorChanged](#) ,
[Control.LocationChanged](#) , [Control.MarginChanged](#) , [Control.RegionChanged](#) ,
[Control.RightToLeftChanged](#) , [Control.SizeChanged](#) , [Control.TabIndexChanged](#) ,
[Control.TabStopChanged](#) , [Control.TextChanged](#) , [Control.VisibleChanged](#) , [Control.Click](#) ,

[Control.ControlAdded](#) , [Control.ControlRemoved](#) , [Control.DataContextChanged](#) ,
[Control.DragDrop](#) , [Control.DragEnter](#) , [Control.DragOver](#) , [Control.DragLeave](#) ,
[Control.GiveFeedback](#) , [Control.HandleCreated](#) , [Control.HandleDestroyed](#) ,
[Control.HelpRequested](#) , [Control.Invalidated](#) , [Control.PaddingChanged](#) , [Control.Paint](#) ,
[Control.QueryContinueDrag](#) , [Control.QueryAccessibilityHelp](#) , [Control.DoubleClick](#) ,
[Control.Enter](#) , [Control.GotFocus](#) , [Control.KeyDown](#) , [Control.KeyPress](#) , [Control.KeyUp](#) ,
[Control.Layout](#) , [Control.Leave](#) , [Control.LostFocus](#) , [Control.MouseClick](#) ,
[Control.MouseDoubleClick](#) , [Control.MouseCaptureChanged](#) , [Control.MouseDown](#) ,
[Control.MouseEnter](#) , [Control.MouseLeave](#) , [Control.DpiChangedBeforeParent](#) ,
[Control.DpiChangedAfterParent](#) , [Control.MouseHover](#) , [Control.MouseMove](#) , [Control.MouseUp](#) ,
[Control.MouseWheel](#) , [Control.Move](#) , [Control.PreviewKeyDown](#) , [Control.Resize](#) ,
[Control.ChangeUICues](#) , [Control.StyleChanged](#) , [Control.SystemColorsChanged](#) ,
[Control.Validating](#) , [Control.Validated](#) , [Control.ParentChanged](#) , [Control.ImeModeChanged](#) ,
[Component.Dispose\(\)](#) , [Component.GetService\(Type\)](#) , [Component.Container](#) ,
[Component.DesignMode](#) , [Component.Events](#) , [Component.Disposed](#) ,
[MarshalByRefObject.GetLifetimeService\(\)](#) , [MarshalByRefObject.InitializeLifetimeService\(\)](#) ,
[MarshalByRefObject.MemberwiseClone\(bool\)](#) , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) ,
[object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#)

Constructors

Form1()

Initializes the form UI and sets up the canvas, command factory, and interpreter runtime.

```
public Form1()
```

Methods

Application_ThreadException(object, ThreadExceptionEventArgs)

Handles unhandled exceptions raised on the UI thread.

```
public void Application_ThreadException(object sender, ThreadExceptionEventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [ThreadExceptionEventArgs](#)

Thread exception event arguments.

Dispose(bool)

Clean up any resources being used.

```
protected override void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

true if managed resources should be disposed; otherwise, false.

Form1_Load(object, EventArgs)

Designer-wired handler to keep the WinForms designer compiling.

```
public void Form1_Load(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

ProgramWindow_TextChanged(object, EventArgs)

Designer-wired handler to keep the WinForms designer compiling.

```
public void ProgramWindow_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

RunFullProgram()

Executes the multi-line program from the ProgramWindow.

```
public void RunFullProgram()
```

RunSingleCommand()

Executes a single command entered in the singleCommandBox.

```
public void RunSingleCommand()
```

checkBtn_Click(object, EventArgs)

Handles the Check button click. Parses and compiles the program to validate syntax without executing it.

```
public void checkBtn_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

debuggerWindow_TextChanged(object, EventArgs)

Designer-wired handler to keep the WinForms designer compiling.

```
public void debuggerWindow_TextChanged(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

exitToolStripMenuItem_Click(object, EventArgs)

Exits the application.

```
public void exitToolStripMenuItem_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

loadCommandToolStripMenuItem_Click(object, EventArgs)

Loads a BOOSE program file into the ProgramWindow.

```
public void loadCommandToolStripMenuItem_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

menuToolStripMenuItem_Click(object, EventArgs)

Designer-wired handler to keep the WinForms designer compiling.

```
public void menuToolStripMenuItem_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

outputBox_Paint(object, PaintEventArgs)

Handles the PictureBox paint event and draws the current canvas bitmap.

```
public void outputBox_Paint(object sender, PaintEventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [PaintEventArgs](#)

Paint event arguments.

runBtn_Click(object, EventArgs)

Handles the Run button click. Runs a single command if provided; otherwise runs the full program.

```
public void runBtn_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

saveCommandToolStripMenuItem_Click(object, EventArgs)

Saves the current program text from ProgramWindow to a file.

```
public void saveCommandToolStripMenuItem_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

saveImageToolStripMenuItem_Click(object, EventArgs)

Saves the current canvas bitmap as a PNG image.

```
public void saveImageToolStripMenuItem_Click(object sender, EventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [EventArgs](#)

Event arguments.

singleCommandBox_KeyDown(object, KeyEventArgs)

Handles Enter key press in the single command input box to run the command immediately.

```
public void singleCommandBox_KeyDown(object sender, KeyEventArgs e)
```

Parameters

sender [object](#)

The event sender.

e [KeyEventArgs](#)

Key event arguments.

Class IfCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **if** command for conditional execution.

```
public class IfCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← IfCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

The **if** command evaluates a boolean condition at runtime and controls program flow by jumping to either the matching **else** or the matching **endif** when the condition is false.

Linking between **if**, **else** and **endif** occurs at compile time using the [ExtendedStoredProgram](#) compile stack.

Properties

ElseIndex

Gets the command index of the matching **else** statement, or -1 if none exists.

```
public int ElseIndex { get; }
```

Property Value

[int](#)

EndIfIndex

Gets the command index of the matching `endif` statement, or -1 if not yet linked.

```
public int EndIfIndex { get; }
```

Property Value

[int](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

`parameterList` [string](#)[]

The tokenised parameter list.

Remarks

Parameter validation is handled in [Set\(StoredProgram, string\)](#).

Compile()

Registers this `if` command on the compile stack so that `else` and `endif` can link to it.

```
public override void Compile()
```

Exceptions

CommandException

Thrown if the program is not an [ExtendedStoredProgram](#).

Execute()

Executes the `if` command by evaluating the condition and jumping over the appropriate block when the condition is false.

```
public override void Execute()
```

Remarks

If the condition evaluates to `true`, execution continues normally into the IF body.

If the condition evaluates to `false`:

- When an `else` exists, execution jumps to the first command after `else`.
- When no `else` exists, execution jumps to the first command after `endif`.

Exceptions

StoredProgramException

Thrown if this command is not linked to a matching `endif`.

Set(StoredProgram, string)

Parses and initializes the `if` command with the supplied condition.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current program instance.

Params [string](#)

The boolean condition expression.

Exceptions

CommandException

Thrown when no condition expression is supplied.

SetElseIndex(int)

Sets the index of the associated `else` command.

```
public void SetElseIndex(int idx)
```

Parameters

idx [int](#)

The index of the `else` command.

SetEndIfIndex(int)

Sets the index of the associated `endif` command.

```
public void SetEndIfIndex(int idx)
```

Parameters

idx [int](#)

The index of the `endif` command.

Class IntCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `int` command for declaring and managing integer variables.

```
public class IntCommand : Evaluation, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← IntCommand

Implements

ICommand

Inherited Members

Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
[Evaluation.ProcessExpression\(string\)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Local ,
Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.Program , Command.Name ,
Command.ParameterList , Command.Parameters , Command.Paramsint , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Properties

Value

Gets or sets the current integer value of the variable.

```
public override int Value { get; set; }
```

Property Value

[int](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

Compile()

Registers the variable and evaluates its initial value.

```
public override void Compile()
```

Execute()

Evaluates the expression and updates the integer value.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the integer declaration command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program [StoredProgram](#)

The current program instance.

Params [string](#)[]

The parameter string in the form <name> or <name> = <expr>.

ToString()

Returns the string representation of the integer value.

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

Returns

string ↗

The current integer value as a string.

Class MethodCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **method** command for defining user methods.

```
public class MethodCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← MethodCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

Compile()

Registers the method and creates its return variable.

```
public override void Compile()
```

Execute()

Skips execution of the method body during normal program flow.

```
public override void Execute()
```

GetMethodName()

Gets the method name.

```
public string GetMethodName()
```

Returns

[string](#)

GetMethodstartIndex()

Gets the command index where the method begins.

```
public int GetMethodstartIndex()
```

Returns

[int](#)

GetParameters()

Gets the list of method parameters.

```
public List<(string type, string name)> GetParameters()
```

Returns

`List<(string type, string name)>`

GetReturnType()

Gets the method return type.

`public string GetReturnType()`

Returns

`string`

Set(StoredProgram, string)

Parses the method declaration.

`public void Set(StoredProgram Program, string Params)`

Parameters

`Program` `StoredProgram`

The current program instance.

`Params` `string`

The method declaration parameters.

SetEndMethodIndex(int)

Sets the index of the matching `endmethod`.

`public void SetEndMethodIndex(int idx)`

Parameters

idx [int](#)

Class MoveToCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `moveto` command, which moves the current canvas position without drawing.

```
public class MoveToCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← MoveToCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

MoveToCommand(ICanvas)

Initializes a new instance of the [MoveToCommand](#) class.

```
public MoveToCommand(ICanvas canvas)
```

Parameters

`canvas` ICanvas

The canvas used for positioning.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

Compile()

Performs compile-time validation.

```
public override void Compile()
```

Execute()

Executes the move-to command by updating the current canvas position.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the move-to command.

```
public override void Set(StoredProgram program, string param)
```

Parameters

program [StoredProgram](#)

The current program instance.

param [string](#)[]

The parameter string containing the target X and Y expressions.

Class PeekCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `peek` command, which retrieves a value from an array and assigns it to an existing variable.

```
public class PeekCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← PeekCommand

Implements

ICommand

Inherited Members

Command.IsDBNull , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

Compile()

Validates that the array and destination variable exist and are compatible.

```
public override void Compile()
```

Execute()

Retrieves the array value at the specified index and assigns it to the destination variable.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the peek command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current program instance.

Params string ↗

The parameter string in the form <var> = <array> <index>.

Class PenCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **pen** command, which sets the current drawing colour.

```
public class PenCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← PenCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

PenCommand(ICanvas)

Initializes a new instance of the [PenCommand](#) class.

```
public PenCommand(ICanvas canvas)
```

Parameters

canvas ICanvas

The canvas used for drawing.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

Compile()

Performs compile-time validation.

```
public override void Compile()
```

Execute()

Executes the pen command by setting the canvas drawing colour.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the pen command.

```
public override void Set(StoredProgram program, string param)
```

Parameters

program [StoredProgram](#)

The current program instance.

param [string](#)[]

The parameter string containing the RGB colour expressions.

Class PokeCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `poke` command, which assigns a value to an array element.

```
public class PokeCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← PokeCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

Compile()

Validates that the target array exists and is valid.

```
public override void Compile()
```

Execute()

Stores the evaluated value into the specified array index.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the poke command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current program instance.

Params [string](#)

The parameter string in the form <array> <index> = <value>.

Class RealCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE **real** command for declaring and managing floating-point variables.

```
public class RealCommand : Evaluation, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← RealCommand

Implements

ICommand

Inherited Members

Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
[Evaluation.ProcessExpression\(string\)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Local ,
Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.Program , Command.Name ,
Command.ParameterList , Command.Parameters , Command.Paramsint , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Properties

RealValue

Gets or sets the current real (double) value of the variable.

```
public double RealValue { get; set; }
```

Property Value

[double](#)

Value

Gets the integer representation of the real value.

```
public override int Value { get; }
```

Property Value

[int ↗](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string ↗](#)[]

Compile()

Registers the variable and evaluates its initial value.

```
public override void Compile()
```

Execute()

Evaluates the expression and updates the real value.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the real declaration command.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The current program instance.

Params string

The parameter string in the form <name> or <name> = <expr>.

ToString()

Returns the string representation of the real value using invariant culture.

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

Returns

string

The real value as a formatted string.

Class RectangleCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `rect` command, which draws a rectangle on the canvas.

```
public class RectangleCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← RectangleCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

RectangleCommand(ICanvas)

Initializes a new instance of the [RectangleCommand](#) class.

```
public RectangleCommand(ICanvas canvas)
```

Parameters

`canvas` ICanvas

The canvas used for drawing.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

Compile()

Performs compile-time validation.

```
public override void Compile()
```

Execute()

Executes the rectangle command by drawing a rectangle on the canvas.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the rectangle drawing command.

```
public override void Set(StoredProgram program, string param)
```

Parameters

program [StoredProgram](#)

The current program instance.

param [string](#)[]

The parameter string containing the width, height, and optional fill flag.

Class TriangleCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `tri` command, which draws a triangle on the canvas.

```
public class TriangleCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← TriangleCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

TriangleCommand(ICanvas)

Initializes a new instance of the [TriangleCommand](#) class.

```
public TriangleCommand(ICanvas canvas)
```

Parameters

`canvas` ICanvas

The canvas used for drawing.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

Compile()

Performs compile-time validation.

```
public override void Compile()
```

Execute()

Executes the triangle command by drawing a triangle on the canvas.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the triangle drawing command.

```
public override void Set(StoredProgram program, string param)
```

Parameters

program [StoredProgram](#)

The current program instance.

param [string](#)[]

The parameter string representing the triangle size.

Class WhileCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `while` command for conditional looping.

```
public class WhileCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← WhileCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] parameterList)
```

Parameters

parameterList [string](#)[]

Compile()

Registers the loop with the compile stack.

```
public override void Compile()
```

Execute()

Evaluates the loop condition and controls program flow.

```
public override void Execute()
```

GetWhileIndex()

Gets the index of this `while` command.

```
public int GetWhileIndex()
```

Returns

`int`

The command index.

Set(StoredProgram, string)

Parses the while-loop condition.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

`Program` `StoredProgram`

The current program instance.

`Params` `string`

The condition expression.

SetEndWhileIndex(int)

Sets the index of the matching `endwhile` command.

```
public void SetEndWhileIndex(int index)
```

Parameters

`index` [int](#)

The command index.

Class WriteCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the BOOSE `write` command, which displays text on the canvas.

```
public class WriteCommand : Command, ICommand
```

Inheritance

[object](#) ← Command ← WriteCommand

Implements

ICommand

Inherited Members

Command.IsDouble , Command.program , Command.parameterList , Command.parameters ,
Command.paramsint , [Command.ProcessParameters\(string\)](#) , Command.ToString() ,
Command.Program , Command.Name , Command.ParameterList , Command.Parameters ,
Command.Paramsint , [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) ,
[object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Constructors

WriteCommand(ICanvas)

Initializes a new instance of the [WriteCommand](#) class.

```
public WriteCommand(ICanvas canvas)
```

Parameters

`canvas` ICanvas

The canvas used for text output.

Methods

CheckParameters(string[])

Validates command parameters.

```
public override void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

Compile()

Performs compile-time validation.

```
public override void Compile()
```

Execute()

Evaluates the expression and writes the resulting text to the canvas.

```
public override void Execute()
```

Set(StoredProgram, string)

Parses and initializes the write command.

```
public override void Set(StoredProgram program, string param)
```

Parameters

program [StoredProgram](#)

The current program instance.

param [string](#)[]

The expression or text to be displayed on the canvas.

Class canvasApp

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Canvas implementation used by BOOSE commands to draw shapes, lines, text, and manage pen state inside a bitmap.

```
public class canvasApp : ICanvas
```

Inheritance

[object](#) ← canvasApp

Implements

ICanvas

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

canvasApp()

Creates a new canvas with default size and initializes the drawing surface.

```
public canvasApp()
```

Properties

PenColour

Current pen colour used for drawing.

```
public object PenColour { get; set; }
```

Property Value

[object](#)

Xpos

Current X drawing coordinate.

```
public int Xpos { get; set; }
```

Property Value

[int](#)

Ypos

Current Y drawing coordinate.

```
public int Ypos { get; set; }
```

Property Value

[int](#)

Methods

Circle(int, bool)

Draws a circle using the current pen settings.

```
public void Circle(int radius, bool filled)
```

Parameters

`radius` [int](#)

Circle radius.

`filled` `bool`

Whether the circle is filled.

Clear()

Clears the canvas to a light gray background.

```
public void Clear()
```

DrawTo(int, int)

Draws a line from the current position to a target coordinate.

```
public void DrawTo(int toX, int toY)
```

Parameters

`toX` `int`

Target X coordinate.

`toY` `int`

Target Y coordinate.

MoveTo(int, int)

Moves the cursor to a new position without drawing.

```
public void MoveTo(int x, int y)
```

Parameters

`x` `int`

New X coordinate.

y [int](#)

New Y coordinate.

Rect(int, int, bool)

Draws a rectangle from the current position.

```
public void Rect(int width, int height, bool filled)
```

Parameters

[width](#) [int](#)

Rectangle width.

[height](#) [int](#)

Rectangle height.

[filled](#) [bool](#)

Whether the rectangle is filled.

Reset()

Resets the cursor position to origin (0,0).

```
public void Reset()
```

Set(int, int)

Sets canvas size and resets cursor position.

```
public void Set(int xsize, int ysize)
```

Parameters

xsize [int](#)

Canvas width.

ysize [int](#)

Canvas height.

SetColour(int, int, int)

Changes the drawing pen colour.

```
public void SetColour(int red, int green, int blue)
```

Parameters

red [int](#)

Red component (0-255).

green [int](#)

Green component (0-255).

blue [int](#)

Blue component (0-255).

Tri(int, int)

Draws a triangle from the current position.

```
public void Tri(int width, int height)
```

Parameters

width [int](#)

Triangle base width.

height [int](#)

Triangle height.

WriteText(string)

Renders a text string at the current canvas position.

```
public void WriteText(string text)
```

Parameters

text [string](#)

Text to draw.

getBitmap()

Returns the internal bitmap used for drawing.

```
public object getBitmap()
```

Returns

[object](#)

Bitmap object.

Class resetCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Represents the RESET command. Clears the stored program and resets the drawing canvas. Syntax: `reset`

```
public class resetCommand : ICommand
```

Inheritance

[object](#) ← resetCommand

Implements

ICommand

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

resetCommand(ICanvas)

Creates a new instance of the resetCommand.

```
public resetCommand(ICanvas canvas)
```

Parameters

canvas ICanvas

The drawing canvas to reset.

Methods

CheckParameters(string[])

Ensures no parameters were supplied.

```
public void CheckParameters(string[] p)
```

Parameters

p [string](#)[]

The parameter array.

Exceptions

CommandException

Thrown if parameters exist.

Compile()

No compilation required for reset.

```
public void Compile()
```

Execute()

Executes the reset operation by clearing the program and resetting the canvas.

```
public void Execute()
```

Set(StoredProgram, string)

Sets the program reference and validates that no parameters are provided.

```
public void Set(StoredProgram Program, string Params)
```

Parameters

Program StoredProgram

The stored program.

Params [string](#)

Parameters passed to the command (should be empty).

Exceptions

CommandEvent

Thrown if parameters are present.