

# Namespace TestProjectSampurna

## Classes

### [ClearCommandTest](#)

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

### [DrawToCommandTest](#)

Unit tests for the WinFormsApp1.DrawToCommand class. Ensures parameters are parsed, validated, compiled, and executed correctly.

### [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](#)

Unit tests for the [MoveToCommand](#) class. Verifies correct parameter parsing, error handling, and that the canvas cursor is moved to the correct coordinates.

### [MultilineProgramTest](#)

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

### [PenCommandTest](#)

Unit tests for the [PenCommand](#) class. Ensures RGB parsing, validation, and canvas interaction behave correctly.

### [RectangleCommandTest](#)

Unit tests for the [rectangleCommand](#) class. Verifies rectangle drawing, fill handling, error validation, and parameter parsing.

### [ResetCommandTest](#)

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

### [TriangleCommandTest](#)

Unit tests for the [TriangleCommand](#) class. Ensures correct validation, compilation and execution behaviour.

### [WriteCommandTest](#)

Unit tests for the WinFormsApp1.WriteCommand class. Ensures messages are parsed, validated, cleaned, and executed correctly.

## circleCommandtest

Contains unit tests for the [circleCommand](#) class. Ensures proper parameter validation and correct execution behavior.

# 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

Unit tests for the WinFormsApp1.DrawToCommand class. Ensures parameters are parsed, validated, compiled, and executed correctly.

```
[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

### CheckParameters\_InvalidCount\_Throws()

Verifies that providing too few or too many parameters throws a CommandException.

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

### Compile\_InvalidX\_Throws()

Ensures Compile() fails when a non-integer X value is provided.

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

## Compile\_InvalidY\_Throws()

Ensures Compile() fails when a non-integer Y value is provided.

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

## Compile\_ValidParameters\_Success()

Confirms that valid parameters ("100 200") compile successfully.

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

## Execute\_ValidParameters\_CallsDrawTo()

Ensures that executing the command causes canvas.DrawTo() to be called exactly once with the expected coordinates.

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

## Setup()

Initializes a fresh canvas and command before each test.

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

# 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

Unit tests for the [MoveToCommand](#) class. Verifies correct parameter parsing, error handling, and that the canvas cursor is moved to the correct coordinates.

```
[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

### CheckParameters\_ShouldThrow\_WhenIncorrectParameterCount()

Ensures [CheckParameters\(string\[\]\)](#) throws a BOOSE.CommandException if parameter count is incorrect.

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

### Execute\_ShouldMoveCursor\_WhenParametersValid()

Ensures valid X and Y parameters correctly move the canvas cursor.

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

## Set\_ShouldThrow\_WhenMissingParameters()

Ensures that missing parameters throw a BOOSE.CommandException.

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

## Set\_ShouldThrow\_WhenXParameterInvalid()

Ensures the command throws a BOOSE.CommandException when the X-coordinate cannot be parsed.

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

## Set\_ShouldThrow\_WhenYParameterInvalid()

Ensures the command throws a BOOSE.CommandException when the Y-coordinate cannot be parsed.

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

## Setup()

Initializes the mock canvas, stored program, and MoveTo command before each unit test.

```
[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

Unit tests for the [PenCommand](#) class. Ensures RGB parsing, validation, and canvas interaction behave correctly.

```
[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\_ShouldSetPenColour\_WhenValidRGB()

Verifies that valid RGB parameters correctly set the pen colour on the canvas through [SetColour\(int, int, int\)](#).

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

### Set\_ShouldThrow\_WhenBIsNotNumber()

Ensures an exception is thrown when the B (blue) component is not a valid numeric value.

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

## Set\_ShouldThrow\_WhenBIsOutOfRange()

Ensures an exception is thrown when the B (blue) component exceeds the allowed range (0–255).

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

## Set\_ShouldThrow\_WhenGIsNotNumber()

Ensures an exception is thrown when the G (green) component contains invalid (non-numeric) input.

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

## Set\_ShouldThrow\_WhenGIsOutOfRange()

Ensures an exception is thrown when the G (green) component is outside the allowed RGB range (0–255).

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

## Set\_ShouldThrow\_WhenNotThreeParameters()

Ensures the command throws an exception when fewer than three parameters are supplied to the pen command.

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

## Set\_ShouldThrow\_WhenRIsNotNumber()

Ensures an exception is thrown when the R (red) component is not a valid integer value.

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

## Set\_ShouldThrow\_WhenRIsOutOfRange()

Ensures an exception is thrown when the R (red) component is outside the allowed range (0–255).

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

## Setup()

Initializes a fresh mock canvas, stored program, and [PenCommand](#) instance before each test.

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

# Class RectangleCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Unit tests for the [rectangleCommand](#) class. Verifies rectangle drawing, fill handling, error validation, and parameter parsing.

```
[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\_ShouldDrawFilledRectangle\_WhenTruePassed()

Ensures the rectangle is drawn as filled when the parameter "true" is supplied.

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

### Execute\_ShouldDrawRectangle\_WithValidParameters()

Ensures a valid width and height cause a rectangle to be drawn and that the fill flag defaults to false.

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

### Execute\_ShouldDrawUnfilledRectangle\_WhenFalsePassed()

Ensures the rectangle is drawn unfilled when the parameter "false" is supplied.

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

## Set\_ShouldThrow\_WhenFillIsInvalid()

Ensures the optional fill parameter only accepts the values "true" or "false", throwing an exception for anything else.

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

## Set\_ShouldThrow\_WhenHeightIsInvalid()

Ensures the command throws an exception when the height parameter is not a positive integer.

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

## Set\_ShouldThrow\_WhenMissingParameters()

Ensures the command throws an exception when fewer than the required two or three parameters are supplied.

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

## Set\_ShouldThrow\_WhenWidthIsInvalid()

Ensures the command throws an exception when the width parameter is non-numeric or invalid.

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

## Setup()

Initializes a new mock canvas, stored program, and rectangle command instance before each test.

```
[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

Unit tests for the [TriangleCommand](#) class. Ensures correct validation, compilation and execution behaviour.

```
[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

### Setup()

Initializes a new canvas and stored program before each test.

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

### TriangleCommand\_Execute\_ValidParameters()

Ensures the command correctly executes and calls canvas.Tri().

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

### TriangleCommand\_InvalidHeight\_ThrowsException()

Ensures the command throws a CommandException when height is not numeric.

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

## TriangleCommand\_InvalidWidth\_ThrowsException()

Ensures the command throws a CommandException when width is not numeric.

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

## TriangleCommand\_NegativeDimensions\_ThrowsException()

Ensures negative dimensions are rejected by the compiler.

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

## TriangleCommand\_NoParameters\_ThrowsException()

Ensures the command throws a CommandException when no parameters are provided.

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

## TriangleCommand\_TooManyParameters\_ThrowsException()

Ensures the command throws a CommandException when too many parameters are supplied.

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

# Class WriteCommandTest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Unit tests for the WinFormsApp1.WriteCommand class. Ensures messages are parsed, validated, cleaned, and executed correctly.

```
[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\_ValidMessage\_CallsWriteText()

Confirms Execute() calls WriteText exactly once.

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

### Parameters\_AfterSet\_StoredCorrectly()

Ensures parameters are stored correctly for BOOSE compatibility.

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

### Set\_DoubleQuotedMessage\_RemovesQuotes()

Ensures Set() removes surrounding double quotes.

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

## Set\_EmptyString\_Throws()

Ensures WriteCommand fails when message is empty.

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

## Set\_QuotedEmptyString\_Throws()

Ensures WriteCommand fails when message contains empty quotes.

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

## Set\_SingleQuotedMessage\_RemovesQuotes()

Ensures Set() removes surrounding single quotes.

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

## Set\_UnquotedMessage\_Success()

Ensures Set() accepts a simple unquoted message.

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

## Setup()

Creates fresh Canvas, Program, and WriteCommand before every test.

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

# Class cricleCommandtest

Namespace: [TestProjectSampurna](#)

Assembly: TestProjectSampurna.dll

Contains unit tests for the [circleCommand](#) class. Ensures proper parameter validation and correct execution behavior.

```
[TestClass]
public class cricleCommandtest
```

## Inheritance

[object](#) ← cricleCommandtest

## Inherited Members

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

# Methods

## Circle\_InvalidFillParameter\_ThrowsException()

Ensures that invalid fill parameters (other than 'true' or 'false') cause a BOOSE.CommandException.

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

## Circle\_InvalidRadius\_ThrowsException()

Ensures that a non-numeric radius triggers a BOOSE.CommandException.

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

## Circle\_LargeRadius\_ExeсutesCorrectly()

Ensures that very large radius values are accepted and executed properly.

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

## Circle\_MissingParameters\_ThrowsException()

Ensures that an empty parameter list triggers a BOOSE.CommandException.

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

## Circle\_NegativeRadius\_ThrowsException()

Ensures that negative radius values are rejected with a BOOSE.CommandException.

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

## Circle\_TooManyParameters\_ThrowsException()

Ensures that providing more than two parameters causes a BOOSE.CommandException.

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

## Circle\_ValidRadius\_ExeсutesCorrectly()

Verifies that executing a circle command with a valid radius results in a proper call to the canvas.

```
[TestMethod]
public void Circle_ValidRadius_Exe
```

## Circle\_WithFalse\_Exe

Verifies that specifying 'false' as the fill parameter produces an unfilled circle.

```
[TestMethod]
public void Circle_WithFalse_Exe
```

## Circle\_WithTrue\_Exe

Verifies that specifying 'true' as the fill parameter produces a filled circle.

```
[TestMethod]
public void Circle_WithTrue_Exe
```

## Circle\_ZeroRadius\_ThrowsException()

Ensures that a radius of zero is rejected with a BOOSE.CommandException.

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

## Setup()

Initializes required objects before each test method runs.

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

# Namespace WinFormsApp1

## Classes

### [AppCommandFactory](#)

Factory class responsible for creating application-specific BOOSE commands. Extends the BOOSE.CommandFactory to support custom drawing commands.

### [ClearCommand](#)

Command that clears the canvas to its default background color.

### [DrawToCommand](#)

Implements the BOOSE `drawto` command. Draws a line from the current canvas position to a new (x, y) coordinate.

### [Form1](#)

Main form of the BOOSE drawing application. Handles execution of commands, program parsing, debugging output, file operations, and drawing through the ICanvas implementation.

### [MoveToCommand](#)

Represents the BOOSE `moveto` command. Moves the canvas cursor to a specified (x, y) position without drawing.

### [PenCommand](#)

Represents a command that changes the pen colour on the canvas.

### [TriangleCommand](#)

Represents the BOOSE command "triangle", used to draw a triangle on the canvas. Syntax: `triangle <width> <height>`

### [WriteCommand](#)

Represents the `write` command in the BOOSE language. This command displays text at the canvas's current pen position. Supports both quoted and unquoted messages.

Examples:

```
write "Hello World"
write 'Test Message'
write Hello
```

### [canvasApp](#)

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

### circleCommand

Represents the BOOSE command "circle", used to draw a circle on the canvas. Supports: circle <radius> [true/false] for filled or outlined.

### rectangleCommand

Implements the RECT command used to draw rectangles on the canvas. Supports optional fill parameter (true/false).

### resetCommand

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

# Class AppCommandFactory

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Factory class responsible for creating application-specific BOOSE commands. Extends the BOOSE.CommandFactory to support custom drawing commands.

```
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\(\)](#)

## Constructors

### AppCommandFactory(ICanvas)

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

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

Parameters

**canvas** ICanvas

The canvas instance used by commands to draw.

## Methods

### MakeCommand(string)

Creates a command instance based on the raw command name string. Supports custom application commands such as moveto, drawto, circle, rect, etc.

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

## Parameters

rawName [string](#)

The command keyword entered by the user.

## Returns

ICommand

An instance of BOOSE.ICommand if matched; otherwise falls back to base factory.

## Exceptions

CommandException

Thrown when the command name is null, empty, or unrecognized.

# 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. Draws a line from the current canvas position to a new (x, y) coordinate.

```
public class DrawToCommand : CommandTwoParameters, ICommand
```

## Inheritance

[object](#) ← Command ← CanvasCommand ← CommandOneParameter ← CommandTwoParameters ← DrawToCommand

## Implements

ICommand

## Inherited Members

CommandTwoParameters.param2 , CommandTwoParameters.param2unprocessed ,  
CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,  
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,  
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

## DrawToCommand(ICanvas)

Creates a new instance of [DrawToCommand](#).

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

## Parameters

`canvas` ICanvas

The canvas used for drawing.

## Methods

### CheckParameters(string[])

Ensures exactly two parameters are provided: X and Y.

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

#### Parameters

p [string](#)[]

Tokenized parameters.

#### Exceptions

CommandException

Thrown if the number of parameters is incorrect.

### Compile()

Converts the raw parameter values into integers.

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

#### Exceptions

CommandException

Thrown if X or Y cannot be parsed into valid integers.

### Execute()

Executes the command by drawing a line from the current canvas position to the specified (x, y) coordinate.

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

## Set(StoredProgram, string)

Sets the command input parameters and prepares for validation and compilation.

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

### Parameters

**Program** StoredProgram

The BOOSE stored program instance.

**Params** string ↗

The raw parameter string, expected in the format "x y".

# Class Form1

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Main form of the BOOSE drawing application. Handles execution of commands, program parsing, debugging output, file operations, and drawing through the ICanvas implementation.

```
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\(KeyEventEventArgs\)](#) ,  
[Control.OnKeyPress\(KeyPressEventEventArgs\)](#) , [Control.OnKeyUp\(KeyEventEventArgs\)](#) ,  
[Control.OnLeave\(EventArgs\)](#) , [Control.InvokeLostFocus\(Control, EventArgs\)](#) ,  
[Control.OnLostFocus\(EventArgs\)](#) , [Control.OnMarginChanged\(EventArgs\)](#) ,  
[Control.OnMouseDoubleClick\(MouseEventArgs\)](#) , [Control.OnMouseClicked\(MouseEventArgs\)](#) ,  
[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\(Control, bool, bool, bool\)](#) , [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.Accessible](#) ,  
[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 main form and sets up BOOSE components.

```
public Form1()
```

## Methods

### 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.

# Class MoveToCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Represents the BOOSE `moveto` command. Moves the canvas cursor to a specified (x, y) position without drawing.

```
public class MoveToCommand : CommandTwoParameters, ICommand
```

## Inheritance

[object](#) ← Command ← CanvasCommand ← CommandOneParameter ← CommandTwoParameters ← MoveToCommand

## Implements

ICommand

## Inherited Members

CommandTwoParameters.param2 , CommandTwoParameters.param2unprocessed ,  
CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,  
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,  
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)

Creates a new instance of [MoveToCommand](#) using the provided canvas.

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

## Parameters

`canvas` ICanvas

The canvas object where drawing operations occur.

## Methods

### CheckParameters(string[])

Validates that the correct number of parameters are provided.

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

#### Parameters

p [string](#)[]

Array of parameter strings.

#### Exceptions

CommandException

Thrown when incorrect parameter count is used.

### Compile()

Compiles and converts the string parameters into integer coordinates.

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

#### Exceptions

CommandException

Thrown if parameters cannot be converted to integers.

### Execute()

Executes the [moveto](#) command and updates the canvas cursor position.

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

## Set(StoredProgram, string)

Sets and parses the parameters for the command.

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

### Parameters

**Program** StoredProgram

The current stored BOOSE program.

**Params** string ↗

The raw parameter string (e.g., "100 200").

### Exceptions

CommandException

Thrown when parameters are missing or invalid.

# Class PenCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Represents a command that changes the pen colour on the canvas.

```
public class PenCommand : CommandThreeParameters, ICommand
```

## Inheritance

[object](#) ← Command ← CanvasCommand ← CommandOneParameter ← CommandTwoParameters ← CommandThreeParameters ← PenCommand

## Implements

ICommand

## Inherited Members

CommandThreeParameters.param3 , CommandThreeParameters.param3unprocessed ,  
CommandTwoParameters.param2 , CommandTwoParameters.param2unprocessed ,  
CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,  
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,  
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

Command syntax:

```
pen <r> <g> <b>
```

Each value must be between 0 and 255.

## Constructors

PenCommand(ICanvas)

Creates a new [PenCommand](#) instance using the provided canvas.

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

## Parameters

**canvas** ICanvas

Canvas where colour will be applied.

# Methods

## CheckParameters(string[])

Validates the number of parameters passed to the command.

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

## Parameters

**p** [string](#)[]

Parameter array.

## Exceptions

CommandException

Thrown when parameter count is not exactly 3.

## Compile()

Compiles and converts parameter strings into usable integer RGB values.

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

## Exceptions

## CommandException

Thrown when values are non-numeric or outside the valid range (0–255).

## Execute()

Executes the pen colour change on the associated canvas.

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

## Set(StoredProgram, string)

Sets and parses the raw parameter string for this command.

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

### Parameters

**Program** StoredProgram

The stored program reference.

**Params** [string](#)

Parameter string containing RGB values.

### Exceptions

CommandException

Thrown if parameters are missing or invalid.

# Class TriangleCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Represents the BOOSE command "triangle", used to draw a triangle on the canvas. Syntax: triangle <width> <height>

```
public class TriangleCommand : CommandOneParameter, ICommand
```

## Inheritance

[object](#) ← Command ← CanvasCommand ← CommandOneParameter ← TriangleCommand

## Implements

ICommand

## Inherited Members

CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,  
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,  
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)

Creates a new instance of TriangleCommand.

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

## Parameters

**canvas** ICanvas

Drawing canvas.

# Methods

## CheckParameters(string[])

Ensures exactly 2 parameters are supplied.

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

### Parameters

parameters `string[]`

## Compile()

Converts parameters into executable numeric values.

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

### Exceptions

CommandException

Thrown for invalid numbers.

## Execute()

Executes the triangle command on the canvas.

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

## Set(StoredProgram, string)

Sets the program and parses the command parameters.

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

## Parameters

**Program** `StoredProgram`

Stored program.

**Params** `string` ↗

Raw parameter string.

## Exceptions

`CommandException`

Thrown for invalid syntax.

# Class WriteCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Represents the `write` command in the BOOSE language. This command displays text at the canvas's current pen position. Supports both quoted and unquoted messages.

Examples:

```
write "Hello World"  
write 'Test Message'  
write Hello
```

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

## Inheritance

[object](#) ← Command ← WriteCommand

## 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\(object, object\)](#)

# Constructors

## WriteCommand(ICanvas)

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

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

## Parameters

`canvas` ICanvas

The canvas where text will be rendered.

## Exceptions

[ArgumentNullException](#)

Thrown if `canvas` is `null`.

## Methods

### CheckParameters(string[])

Performs compile-time validation. Not required for `write` since validation occurs in [Set\(StoredProgram, string\)](#).

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

#### Parameters

`parameters` [string](#)[]

Parameter array passed for validation.

### Compile()

Performs any compile-time processing. The `write` command has no compile step.

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

### Execute()

Executes the command by drawing the stored message on the canvas. Called by BOOSE.StoredProgram.Run().

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

## Set(StoredProgram, string)

Parses and stores the message text to be written. This method is called by the BOOSE parser.

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

### Parameters

**Program** StoredProgram

The current program context.

**Params** string ↗

The raw message string, possibly quoted.

### Exceptions

CommandException

Thrown if no valid message text is provided.

# 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 circleCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Represents the BOOSE command "circle", used to draw a circle on the canvas. Supports: circle <radius> [true/false] for filled or outlined.

```
public class circleCommand : CommandOneParameter, ICommand
```

## Inheritance

[object](#) ← Command ← CanvasCommand ← CommandOneParameter ← circleCommand

## Implements

ICommand

## Inherited Members

CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,  
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,  
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

### circleCommand(ICanvas)

Creates a new instance of [circleCommand](#).

```
public circleCommand(ICanvas canvas)
```

## Parameters

**canvas** ICanvas

The canvas on which the circle will be drawn.

# Methods

## CheckParameters(string[])

Validates the number of parameters for the circle command.

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

### Parameters

parameters `string[]`

Array of parameters.

### Exceptions

CommandException

Thrown if parameter count is invalid.

## Compile()

Converts parameters into executable values (parses radius and fill flag).

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

### Exceptions

CommandException

Thrown if radius is invalid or the fill flag is not "true"/"false".

## Execute()

Executes the circle command on the canvas.

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

## Set(StoredProgram, string)

Sets the program and parses the command parameters.

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

### Parameters

**Program** StoredProgram

Stored BOOSE program instance.

**Params** string ↗

Command parameters as a string.

### Exceptions

CommandEvent

Thrown if the parameter format is invalid.

# Class rectangleCommand

Namespace: [WinFormsApp1](#)

Assembly: WinFormsApp1.dll

Implements the RECT command used to draw rectangles on the canvas. Supports optional fill parameter (true/false).

```
public class rectangleCommand : CommandTwoParameters, ICommand
```

## Inheritance

[object](#) ← Command ← CanvasCommand ← CommandOneParameter ← CommandTwoParameters ← rectangleCommand

## Implements

ICommand

## Inherited Members

CommandTwoParameters.param2 , CommandTwoParameters.param2unprocessed ,  
CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,  
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,  
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)

Creates a new rectangleCommand instance.

```
public rectangleCommand(ICanvas canvas)
```

## Parameters

**canvas** ICanvas

The drawing canvas.

## Methods

### CheckParameters(string[])

Validates the number of parameters passed to the RECT command.

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

#### Parameters

`parameters string[]`

Parameter array.

#### Exceptions

CommandException

Thrown if parameter count is not 2 or 3.

### Compile()

Converts parameters into numerical values and evaluates optional fill flag.

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

#### Exceptions

CommandException

Thrown if width, height, or fill option is invalid.

### Execute()

Executes the RECT command and draws a rectangle on the canvas.

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

## Set(StoredProgram, string)

Sets the command parameters from program input.

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

### Parameters

**Program** StoredProgram

Stored program instance.

**Params** string ↗

Raw parameter string.

### Exceptions

CommandException

Thrown if invalid number of parameters.

# 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.