Namespace ASE_Assignment

Classes

AppCanvas

Provides a graphical canvas for drawing shapes, lines, and other elements. Supports setting Pen color, moving the drawing cursor, and performing various rendering operations with custom colors, sizes, and positioning.

AppCircle

Represents a command to draw a circle on the canvas.

AppCommandFactory

Provides a factory for creating command objects based on command strings. Implements the Factory pattern to create appropriate command instances for drawing, variable manipulation, and control flow operations.

AppHexagon

Represents a command to draw a hexagon on the canvas. Configures the hexagon's center position and size, and provides execution and parameter checking functionality.

AppMoveTo

Represents a command to move the drawing cursor to a specific position.

AppPen

Represents a command to set the pen color for drawing operations.

AppRectangle

Represents a command to draw a rectangle on the canvas.

AppStickman

Represents a command to draw a stickman figure on the canvas. Configures the stickman's position and size, and provides execution and parameter checking functionality.

AppTriangle

Represents a command to draw a triangle on the canvas. Configures the triangle's width and height, and provides execution and parameter checking functionality.

AppWrite

Represents a command to write text on the canvas. Configures the text content and provides execution and parameter checking functionality.

BOOSEInterpreterUI

Represents the main form interface for interacting with the BOOSE canvas and executing commands. Provides methods for handling form events, running commands, and refreshing the canvas display.

CommandHandlers

Handles command processing and execution for the BOOSE interpreter. Manages variables, arrays, control structures, and drawing operations.

CustomCommand

Represents a custom command that executes a specified action.

ExpressionParser

Provides parsing and evaluation capabilities for mathematical expressions. Supports basic arithmetic operations, variables, and parentheses.

Class AppCanvas

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Provides a graphical canvas for drawing shapes, lines, and other elements. Supports setting `Pen` color, moving the drawing cursor, and performing various rendering operations with custom colors, sizes, and positioning.

```
public class AppCanvas : ICanvas
```

Inheritance

object ← AppCanvas

Implements

ICanvas

Inherited Members

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

Constructors

AppCanvas()

Initializes a new instance of the AppCanvas class, setting default dimensions and clearing the canvas to white.

```
public AppCanvas()
```

Properties

PenColour

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

Property Value

object

Xpos

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

Property Value

int

Ypos

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

Property Value

int

Methods

Circle(int, bool)

Draws a circle at the current position on the canvas. Allows setting the circle `radius` and whether it should be filled.

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

Parameters

`radius` int

The radius of the circle in pixels.

`filled` bool

true to fill the circle; false for an outline.

Exceptions

CanvasException

Thrown if `radius` is zero or negative, or if the Graphics object is uninitialized.

Clear()

Clears the canvas by filling it with white, erasing all previous content.

```
public void Clear()
```

Exceptions

CanvasException

Thrown if the `Graphics` object is uninitialized.

DrawStickman(int, int, int)

Draws a stickman figure at the specified position with the given `size`.

```
public void DrawStickman(int centerX, int centerY, int size)
```

Parameters

`centerX` int

The `x`-coordinate of the stickman's center.

`centerY` int

The `y`-coordinate of the stickman's center.

`size` int

The `size` of the stickman, affecting head and limb proportions.

Remarks

The `size` parameter affects the proportions of each body part. This method draws a stickman with a head, body, arms, and legs centered at the specified coordinates.

Exceptions

CanvasException

Thrown if `size` is zero or negative, or if the Graphics object is uninitialized.

DrawTo(int, int)

Draws a straight line from the current position to the specified coordinates. Updates the cursor position to the endpoint after drawing.

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

Parameters

`toX` int

The `x`-coordinate to draw to.

`toY` int

The `y`-coordinate to draw to.

Remarks

This method assumes the Graphics object is initialized and the coordinates are within the canvas size.

Exceptions

CanvasException

Thrown if the specified coordinates are out of bounds or if the Graphics object is uninitialized.

Hex(int, int, int)

Draws a hexagon centered at the specified position with a given `size`.

```
public void Hex(int centerX, int centerY, int size)
```

Parameters

`centerX` int

The `x`-coordinate of the hexagon center.

`centerY` int

The `y`-coordinate of the hexagon center.

`size` int

The radius of the hexagon.

Exceptions

CanvasException

Thrown if `size` is zero or negative, or if the Graphics object is uninitialized.

MoveTo(int, int)

Moves the drawing cursor to a new position without drawing a line.

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

Parameters

`x` int

The new `x`-coordinate for the cursor.

`y` int

The new `y`-coordinate for the cursor.

Exceptions

CanvasException

Thrown if the specified coordinates are negative.

Rect(int, int, bool)

Draws a rectangle at the current cursor position. Supports setting the `width`, `height`, and whether it should be filled.

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

Parameters

`width` int

The width of the rectangle in pixels.

`height` int

The height of the rectangle in pixels.

`filled` bool

true to fill the rectangle; false for an outline.

Remarks

Ensure that `width` and `height` are positive values and within canvas bounds.

Exceptions

CanvasException

Thrown if `width` or `height` is zero or negative, or if the Graphics object is uninitialized.

Reset()

Resets the canvas by clearing it to white and setting the cursor position to the origin.

```
public void Reset()
```

Exceptions

BOOSEException

Thrown if an error occurs during the reset process.

Set(int, int)

Sets the canvas size and initializes the drawing area.

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

Parameters

`xsize` int

The width of the canvas in pixels.

`ysize` int

The height of the canvas in pixels.

Remarks

This method sets up the canvas with new dimensions and should be called before drawing any content.

Exceptions

BOOSEException

Thrown if an error occurs during the initialization of the Graphics object.

SetColour(int, int, int)

Sets the color of the drawing pen using RGB values.

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

Parameters

`red` int

The red component of the color (0-255).

`green` int

The green component of the color (0-255).

`blue` int

The blue component of the color (0-255).

Remarks

This method updates the pen color and retains the current `penSize`.

Exceptions

CanvasException

Thrown if any of the color values are outside the range of 0 to 255.

Tri(int, int)

Draws a triangle at the current position with the specified dimensions.

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

Parameters

`width` int

The base `width` of the triangle in pixels.

`height` int

The `height` of the triangle in pixels.

Exceptions

CanvasException

Thrown if `width` or `height` is zero or negative, or if the Graphics object is uninitialized.

WriteText(string)

Renders text at the current cursor position.

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

Parameters

`text` string

The `string` of text to display.

Remarks

Use this method to display text on the canvas; ensure the Graphics object is initialized.

Exceptions

CanvasException

Thrown if `text` is null or empty, or if the Graphics object is uninitialized.

getBitmap()

Returns the bitmap image representing the current state of the canvas.

```
public object getBitmap()
```

Returns

object

The `Bitmap` object for the canvas.

Exceptions

BOOSEException

Thrown if there is an error accessing the `Bitmap` object.

Class AppCircle

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to draw a circle on the canvas.

```
public class AppCircle : ICommand
```

Inheritance

object ← AppCircle

Implements

ICommand

Inherited Members

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

Constructors

AppCircle(AppCanvas, int, bool)

Initializes a new instance of the circle drawing command.

```
public AppCircle(AppCanvas canvas, int radius, bool filled = false)
```

Parameters

`canvas` AppCanvas

The canvas to draw on.

`radius` int

The radius of the circle in pixels.

`filled` bool

Optional parameter to fill the circle.

Exceptions

ArgumentException

Thrown when radius is less than or equal to zero.

Methods

CheckParameters(string[])

Validates and sets the circle parameters.

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

Parameters

`parameters` string[]

Array containing radius and optional fill parameter.

Exceptions

ArgumentException

Thrown when parameters are invalid.

Compile()

```
public void Compile()
```

Execute()

Executes the circle drawing command.

```
public void Execute()
```

Set(StoredProgram, string)

```
public void Set(StoredProgram program, string name)
```

Parameters

`program` StoredProgram

`name` string

Class AppCommandFactory

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Provides a factory for creating command objects based on command strings. Implements the Factory pattern to create appropriate command instances for drawing, variable manipulation, and control flow operations.

```
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

Supports multiple command types:

- Drawing commands (moveto, circle, rect, etc.)
- Variable operations (int, real, array)
- Control structures (if, while, for)
- Method declarations and calls

Constructors

AppCommandFactory(AppCanvas)

Initializes a new instance of the AppCommandFactory class.

```
public AppCommandFactory(AppCanvas canvas)
```

Parameters

`canvas` AppCanvas

The AppCanvas on which commands will operate.

Methods

Clear()

```
public void Clear()
```

MakeCommand(string)

Creates a command based on the provided command string.

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

Parameters

`commandType` string

The command string to parse. Available commands: - moveto x,y : Moves cursor to coordinates (AppMoveTo) - circle radius [filled] : Draws circle with optional fill (AppCircle) - rect width height [filled] : Draws rectangle with optional fill (AppRectangle) - tri width height : Draws triangle (AppTriangle) - hex x y size : Draws hexagon (AppHexagon) - pen r g b : Sets pen color (RGB values 0-255) (AppPen) - write text : Writes text at current position (AppWrite) - stickman x y size : Draws stickman figure (AppStickman)

Returns

ICommand

An ICommand instance for the specified command.

Exceptions

ArgumentException

Thrown when command is invalid or parameters are incorrect.

Class AppHexagon

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to draw a hexagon on the canvas. Configures the hexagon's center position and size, and provides execution and parameter checking functionality.

```
public class AppHexagon : ICommand
```

Inheritance

object ← AppHexagon

Implements

ICommand

Inherited Members

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

Constructors

AppHexagon(AppCanvas, int, int, int)

Initializes a new instance of the AppHexagon class.

```
public AppHexagon(AppCanvas canvas, int centerX, int centerY, int size)
```

Parameters

`canvas` AppCanvas

The AppCanvas on which to draw the hexagon.

`centerX` int

The x-coordinate of the hexagon's center position.

`centerY` int

The y-coordinate of the hexagon's center position.

`size` int

The size (radius) of the hexagon.

Methods

CheckParameters(string[])

Checks and validates the parameters for the hexagon command.

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

Parameters

`parameters` string[]

An array of parameters for the hexagon command.

Remarks

The `parameters` array must contain three integers representing `_centerX`, `_centerY`, and `_size` respectively.

Exceptions

ArgumentException

Thrown if there are insufficient parameters, if parsing fails, or if `_size` is non-positive.

Compile()

Compiles the hexagon command, preparing it for execution.

```
public void Compile()
```

Execute()

Executes the hexagon drawing command by rendering the hexagon on the canvas.

```
public void Execute()
```

Set(StoredProgram, string)

Sets the stored program and command string for the hexagon command.

```
public void Set(StoredProgram program, string commandString)
```

Parameters

`program` StoredProgram

The BOOSE.StoredProgram associated with this command.

`commandString` string

The string representation of the command.

Class AppMoveTo

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to move the drawing cursor to a specific position.

```
public class AppMoveTo : ICommand
```

Inheritance

object ← AppMoveTo

Implements

ICommand

Inherited Members

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

Constructors

AppMoveTo(AppCanvas, int, int)

Initializes a new instance of the MoveTo command.

```
public AppMoveTo(AppCanvas canvas, int x, int y)
```

Parameters

`canvas` AppCanvas

The canvas to draw on.

`x` int

X coordinate.

`y` int

Y coordinate.

Methods

CheckParameters(string[])

Validates and sets the movement parameters.

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

Parameters

`parameters` string[]

Array containing x and y coordinates.

Exceptions

ArgumentException

Thrown when parameters are invalid.

Compile()

Compiles the command.

```
public void Compile()
```

Execute()

Executes the move command, updating the canvas cursor position.

```
public void Execute()
```

Set(StoredProgram, string)

Sets program state for the command.

```
public void Set(StoredProgram program, string name)
```

Parameters

`program` StoredProgram

`name` string

Class AppPen

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to set the pen color for drawing operations.

```
public class AppPen : ICommand
```

Inheritance

object ← AppPen

Implements

ICommand

Inherited Members

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

Constructors

AppPen(AppCanvas, int, int, int)

Initializes a new instance of the AppPen class.

```
public AppPen(AppCanvas canvas, int r, int g, int b)
```

Parameters

`canvas` AppCanvas

The canvas to draw on.

`r` int

Red component (0-255).

`g` int

Green component (0-255).

`b` int

Blue component (0-255).

Exceptions

ArgumentException

Thrown when color values are out of range.

Methods

CheckParameters(string[])

Validates and sets the pen color parameters.

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

Parameters

`parameters` string[]

Array containing RGB values.

Exceptions

ArgumentException

Thrown when parameters are invalid or out of range.

Compile()

Compiles the command.

```
public void Compile()
```

Execute()

Executes the pen color change command.

```
public void Execute()
```

Set(StoredProgram, string)

Sets program state for the command.

```
public void Set(StoredProgram program, string name)
```

Parameters

`program` StoredProgram

`name` string

Class AppRectangle

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to draw a rectangle on the canvas.

```
public class AppRectangle : ICommand
```

Inheritance

object ← AppRectangle

Implements

ICommand

Inherited Members

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

Constructors

AppRectangle(AppCanvas, int, int, bool)

Initializes a new instance of the rectangle command.

```
public AppRectangle(AppCanvas canvas, int width, int height, bool filled = false)
```

Parameters

`canvas` AppCanvas

The canvas to draw on.

`width` int

Width of the rectangle.

`height` int

Height of the rectangle.

`filled` bool

Whether the rectangle should be filled.

Exceptions

ArgumentException

Thrown when width or height is less than or equal to zero.

Methods

CheckParameters(string[])

Validates and sets the rectangle parameters.

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

Parameters

`parameters` string[]

Array containing width, height, and optional fill parameter.

Exceptions

ArgumentException

Thrown when parameters are invalid.

Compile()

Compiles the command.

```
public void Compile()
```

Execute()

Executes the rectangle drawing command.

```
public void Execute()
```

Set(StoredProgram, string)

Sets program state for the command.

```
public void Set(StoredProgram program, string name)
```

Parameters

`program` StoredProgram

`name` string

Class AppStickman

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to draw a stickman figure on the canvas. Configures the stickman's position and size, and provides execution and parameter checking functionality.

```
public class AppStickman : ICommand
```

Inheritance

object ← AppStickman

Implements

ICommand

Inherited Members

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

Constructors

AppStickman(AppCanvas, int, int, int)

Initializes a new instance of the AppStickman class.

```
public AppStickman(AppCanvas canvas, int centerX, int centerY, int size)
```

Parameters

`canvas` AppCanvas

The AppCanvas on which to draw the stickman.

`centerX` int

The x-coordinate of the stickman's center position.

`centerY` int

The y-coordinate of the stickman's center position.

`size` int

The size of the stickman, defining its proportions.

Methods

CheckParameters(string[])

Checks and validates the parameters for the stickman command.

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

Parameters

`parameters` string[]

An array of parameters for the stickman command.

Remarks

The parameters array must contain three integers representing the `parameters` for `_centerX`, `_centerY`, and `_size` respectively.

Exceptions

ArgumentException

Thrown if there are insufficient parameters, if parsing fails, or if `_size` is non-positive.

Compile()

Compiles the stickman command, preparing it for execution.

```
public void Compile()
```

Execute()

Executes the stickman drawing command by rendering the stickman on the canvas.

```
public void Execute()
```

Set(StoredProgram, string)

Sets the stored program and command string for the stickman command.

```
public void Set(StoredProgram program, string commandString)
```

Parameters

`program` StoredProgram

The BOOSE.StoredProgram associated with this command.

`commandString` string

The string representation of the command.

Class AppTriangle

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to draw a triangle on the canvas. Configures the triangle's width and height, and provides execution and parameter checking functionality.

```
public class AppTriangle : ICommand
```

Inheritance

object ← AppTriangle

Implements

ICommand

Inherited Members

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

Constructors

AppTriangle(AppCanvas, int, int)

Initializes a new instance of the AppTriangle class.

```
public AppTriangle(AppCanvas canvas, int width, int height)
```

Parameters

`canvas` AppCanvas

The AppCanvas on which to draw the triangle.

`width` int

The base width of the triangle in pixels.

`height` int

The height of the triangle in pixels.

Methods

CheckParameters(string[])

Checks and validates the parameters for the triangle command.

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

Parameters

`parameters` string[]

An array of parameters for the triangle command.

Remarks

The `parameters` array must contain two integers representing `_width` and `_height` respectively.

Exceptions

ArgumentException

Thrown if there are insufficient parameters, if parsing fails, or if `_width` or `_height` is non-positive.

Compile()

Compiles the triangle command, preparing it for execution.

```
public void Compile()
```

Execute()

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

```
public void Execute()
```

Set(StoredProgram, string)

Sets the stored program and command string for the triangle command.

```
public void Set(StoredProgram program, string commandString)
```

Parameters

`program` StoredProgram

The BOOSE.StoredProgram associated with this command.

`commandString` string

The string representation of the command.

Class AppWrite

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a command to write text on the canvas. Configures the text content and provides execution and parameter checking functionality.

```
public class AppWrite : ICommand
```

Inheritance

object ← AppWrite

Implements

ICommand

Inherited Members

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

Constructors

AppWrite(AppCanvas, string, ExpressionParser)

Initializes a new instance of the AppWrite class.

```
public AppWrite(AppCanvas canvas, string text, ExpressionParser parser)
```

Parameters

`canvas` AppCanvas

The AppCanvas on which to write the text.

`text` string

The text content to write on the canvas.

`parser` ExpressionParser

The ExpressionParser to evaluate expressions.

Exceptions

ArgumentException

Thrown if the text is empty.

Methods

CheckParameters(string[])

Validates and sets the text parameters.

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

Parameters

`parameters` string[]

Array containing text to write.

Exceptions

ArgumentException

Thrown when text is empty or invalid.

Compile()

Compiles the command.

```
public void Compile()
```

Execute()

Executes the write command, displaying text on the canvas.

```
public void Execute()
```

Set(StoredProgram, string)

Sets program state for the command.

```
public void Set(StoredProgram program, string name)
```

Parameters

`program` StoredProgram

`name` string

Class BOOSEInterpreterUI

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents the main form interface for interacting with the BOOSE canvas and executing commands. Provides methods for handling form events, running commands, and refreshing the canvas display.

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

Inheritance

object ← MarshalByRefObject ← Component ← Control ← ScrollableControl ← ContainerControl ← Form ← BOOSEInterpreterUI

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.ScaleMinMaxSize(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.PerformAutoScale() , 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.OnInvalidated(InvalidateEventArgs) , Control.OnKeyDown(KeyEventArgs) ,
Control.OnKeyPress(KeyPressEventArgs) , Control.OnKeyUp(KeyEventArgs) ,
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

BOOSEInterpreterUI()

Initializes a new instance of the BOOSEInterpreterUI class, setting up the canvas, command factory, stored program, and parser.

```
public BOOSEInterpreterUI()
```

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 CommandHandlers

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Handles command processing and execution for the BOOSE interpreter. Manages variables, arrays, control structures, and drawing operations.

```
public class CommandHandlers
```

Inheritance

object ← CommandHandlers

Inherited Members

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

Remarks

Supports the following operations:

- Variable declaration and assignment (int, real)
- Array operations (declaration, element access)
- Control structures (if, while, for)
- Method declarations and calls

Constructors

CommandHandlers(AppCanvas)

Initializes a new instance of the CommandHandlers class.

```
public CommandHandlers(AppCanvas canvas)
```

Parameters

`canvas` AppCanvas

The canvas to draw on.

Properties

IsExecutingBlock

Gets whether the current block is being executed based on control flow state.

```
public bool IsExecutingBlock { get; }
```

Property Value

bool

Variables

Gets the dictionary of program variables and their values.

```
public Dictionary<string, double> Variables { get; }
```

Property Value

Dictionary <string, double>

Methods

Clear()

Clears all program state including variables, arrays, control flow stacks, and resets the canvas.

```
public void Clear()
```

GetArrayElement(string, int)

Gets the value at the specified index in the named array.

```
public double GetArrayElement(string arrayName, int index)
```

Parameters

`arrayName` string

The name of the array to access.

`index` int

The index of the element to retrieve.

Returns

double

The value at the specified index.

Exceptions

ArgumentException

Thrown when the array doesn't exist or index is out of bounds.

HandleArrayCommand(string[])

Handles array-related commands including declaration and operations.

```
public ICommand HandleArrayCommand(string[] parts)
```

Parameters

`parts` string[]

Command parts containing array operation details.

Returns

ICommand

Remarks

Supports:

- array type name size : Declares new array
- name[index] = value : Assigns value to array element
- var = name[index] : Retrieves value from array element

Exceptions

ArgumentException

Thrown when array operation is invalid.

HandleArrayDeclaration(string, double[])

Declares a new array with the specified name and initial values.

```
public void HandleArrayDeclaration(string name, double[] values)
```

Parameters

`name` string

The name of the array to declare.

`values` double[]

The initial values for the array.

Exceptions

ArgumentException

Thrown when the array name is empty or null.

HandleArrayOperation(string[])

Handles array operations including element access and modification.

```
public ICommand HandleArrayOperation(string[] parts)
```

Parameters

`parts` string[]

Array containing the operation details (operation type, array name, index, value).

Returns

ICommand

An ICommand that performs the specified array operation.

Remarks

Supports two operations:

- poke: Sets an array element value (poke arrayName index = value)
- peek: Retrieves an array element value (peek varName = arrayName index)

Exceptions

ArgumentException

Thrown when operation syntax is invalid or array/index is not found.

HandleControlStructure(string[])

Processes control flow structures like if, else, and end statements.

```
public ICommand HandleControlStructure(string[] parts)
```

Parameters

`parts` string[]

Array containing the control structure command and its parameters.

Returns

ICommand

An ICommand that manages control flow.

Exceptions

ArgumentException

Thrown when control structure syntax is invalid.

InvalidOperationException

Thrown when control structures are improperly nested.

HandleDrawingCommand(string, string[])

Processes drawing commands like moveto, circle, rect, etc.

```
public ICommand HandleDrawingCommand(string command, string[] parts)
```

Parameters

`command` string

The drawing command to execute.

`parts` string[]

Array of command parameters.

Returns

ICommand

An ICommand for the specified drawing operation.

Exceptions

ArgumentException

Thrown when command parameters are invalid.

HandleVariableCommand(string[])

Processes variable declarations and assignments.

```
public ICommand HandleVariableCommand(string[] parts)
```

Parameters

`parts` string[]

Command parts containing variable operation details.

Returns

ICommand

Remarks

Supports:

- int name = value : Declares integer variable
- real name = value : Declares floating-point variable
- name = expression : Assigns result of expression to variable

Exceptions

ArgumentException

Thrown when variable operation is invalid.

SetArrayElement(string, int, double)

Sets the value at the specified index in the named array.

```
public void SetArrayElement(string arrayName, int index, double value)
```

Parameters

`arrayName` string

The name of the array to modify.

`index` int

The index of the element to set.

`value` double

The value to set at the specified index.

Exceptions

ArgumentException

Thrown when the array doesn't exist or index is out of bounds.

ShouldExecuteCommand()

Determines whether the current command should be executed based on control flow state.

```
public bool ShouldExecuteCommand()
```

Returns

bool

True if the command should be executed; otherwise, false.

Class CustomCommand

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Represents a custom command that executes a specified action.

```
public class CustomCommand : ICommand
```

Inheritance

object ← CustomCommand

Implements

ICommand

Inherited Members

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

Constructors

CustomCommand(Action)

Initializes a new instance of the CustomCommand class.

```
public CustomCommand(Action action)
```

Parameters

`action` Action

The action to execute when the command is run.

Methods

CheckParameters(string[])

Checks the parameters for the command and validates them.

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

Parameters

`parameters` string[]

An array of parameters to validate.

Compile()

Compiles the command, preparing it for execution.

```
public void Compile()
```

Execute()

Executes the command by performing the associated action.

```
public void Execute()
```

Set(StoredProgram, string)

Sets the stored program and command string for the command.

```
public void Set(StoredProgram program, string commandString)
```

Parameters

`program` StoredProgram

The BOOSE.StoredProgram associated with this command.

`commandString` string

The string representation of the command.

Class ExpressionParser

Namespace: ASE_Assignment

Assembly: BOOSEInterpreter.dll

Provides parsing and evaluation capabilities for mathematical expressions. Supports basic arithmetic operations, variables, and parentheses.

```
public class ExpressionParser
```

Inheritance

object ← ExpressionParser

Inherited Members

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

Constructors

ExpressionParser(Dictionary<string, double>)

Initializes a new instance of the ExpressionParser class.

```
public ExpressionParser(Dictionary<string, double> variables)
```

Parameters

`variables` Dictionary<string, double>

Dictionary containing variable names and their values.

Methods

EvaluateExpression(string)

Evaluates a mathematical expression and returns its result.

```
public double EvaluateExpression(string expression)
```

Parameters

`expression` string

The mathematical expression to evaluate.

Returns

double

The calculated result of the expression.

Exceptions

InvalidOperationException

Thrown when the expression is invalid or contains undefined variables.

Tokenize(string)

Breaks an expression string into tokens (numbers, operators, variables, parentheses).

```
public List<string> Tokenize(string expression)
```

Parameters

`expression` string

The mathematical expression to tokenize.

Returns

List <string>

A list of tokens extracted from the expression.

Namespace ASE_Tests

Classes

AppCircleTests
: Tests for the AppCircle class. Verifies circle drawing functionality and parameter validation.

AppCommandFactoryTests
: Tests for the AppCommandFactory class. Verifies the factory's ability to create appropriate command instances from string inputs and handle invalid commands correctly.

AppHexagonTests
: Tests for the AppHexagon class. Verifies hexagon drawing functionality and parameter validation.

AppMoveToTests
: Tests for the AppMoveTo class. Verifies cursor movement functionality and coordinate validation.

AppPenTests
: Tests for the AppPen class. Verifies pen color setting functionality and RGB value validation.

AppRectangleTests
: Tests for the AppRectangle class. Verifies rectangle drawing functionality and parameter validation.

AppStickmanTests
: Tests for the AppStickman class. Verifies stickman drawing functionality and parameter validation.

AppTriangleTests
: Tests for the AppTriangle class. Verifies triangle drawing functionality and parameter validation.

AppWriteTests
: Tests for the AppWrite class. Verifies text writing functionality including string literals, expressions, and variable substitution.

CommandHandlersTests
: Tests for the CommandHandlers class. Verifies program storage, variable handling, and control flow functionality.

ExpressionParserTests
: Tests for the ExpressionParser class. Verifies expression parsing and evaluation functionality.

Class AppCircleTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppCircle class. Verifies circle drawing functionality and parameter validation.

```
[TestClass]
public class AppCircleTests
```

Inheritance

object ← AppCircleTests

Inherited Members

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

Methods

CheckParameters_InsufficientParameters_ThrowsException()

Tests that parameter validation rejects insufficient parameters.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_InsufficientParameters_ThrowsException()
```

CheckParameters_NonNumericRadius_ThrowsException()

Tests that parameter validation rejects non-numeric radius.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericRadius_ThrowsException()
```

CheckParameters_ValidParameters_SetsProperties()

Tests that parameter validation accepts valid parameters.

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

Constructor_NegativeRadius_ThrowsException()

Tests that constructor rejects negative radius.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_NegativeRadius_ThrowsException()
```

Constructor_ValidRadius_CreatesCircle()

Tests that constructor accepts valid radius value.

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

Constructor_WithFilled_CreatesFilledCircle()

Tests that constructor correctly handles filled circle creation.

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

Constructor_ZeroRadius_ThrowsException()

Tests that constructor rejects zero radius.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
```

```
public void Constructor_ZeroRadius_ThrowsException()
```

Execute_FilledCircle_DrawsCorrectly()

Tests that Execute method correctly draws filled circle. The actual drawing cannot be verified, only the execution completion.

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

Execute_UnfilledCircle_DrawsCorrectly()

Tests that Execute method correctly draws unfilled circle. The actual drawing cannot be verified, only the execution completion.

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

SetUp()

Initializes test environment before each test.

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

Class AppCommandFactoryTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppCommandFactory class. Verifies the factory's ability to create appropriate command instances from string inputs and handle invalid commands correctly.

```
[TestClass]
public class AppCommandFactoryTests
```

Inheritance

object ← AppCommandFactoryTests

Inherited Members

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

Methods

MakeCommand_Circle_ReturnsValidCommand()

Tests that Circle command is created with valid radius.

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

MakeCommand_EmptyCommand_ThrowsException()

Tests that empty commands throw appropriate exceptions.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void MakeCommand_EmptyCommand_ThrowsException()
```

MakeCommand_Hexagon_ReturnsValidCommand()

Tests that Hexagon command is created with valid parameters.

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

MakeCommand_InvalidCommand_ThrowsException()

Tests that invalid commands throw appropriate exceptions.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void MakeCommand_InvalidCommand_ThrowsException()
```

MakeCommand_MoveTo_ReturnsValidCommand()

Tests that MoveTo command is created with valid coordinates.

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

MakeCommand_Pen_ReturnsValidCommand()

Tests that Pen command is created with valid RGB values.

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

MakeCommand_Rectangle_ReturnsValidCommand()

Tests that Rectangle command is created with valid dimensions.

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

MakeCommand_Triangle_ReturnsValidCommand()

Tests that Triangle command is created with valid dimensions.

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

MakeCommand_Variable_ReturnsValidCommand()

Tests that variable declaration commands are handled correctly.

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

MakeCommand_Write_ReturnsValidCommand()

Tests that Write command is created with valid text.

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

SetUp()

Initializes test environment before each test.

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

Class AppHexagonTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppHexagon class. Verifies hexagon drawing functionality and parameter validation.

```
[TestClass]
public class AppHexagonTests
```

Inheritance

object ← AppHexagonTests

Inherited Members

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

Methods

CheckParameters_EmptyParameters_ThrowsException()

Tests that parameter validation rejects empty parameter array.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_EmptyParameters_ThrowsException()
```

CheckParameters_InsufficientParameters_ThrowsException()

Tests that parameter validation rejects insufficient parameters.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_InsufficientParameters_ThrowsException()
```

CheckParameters_NonNumericSize_ThrowsException()

Tests that parameter validation rejects non-numeric size.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericSize_ThrowsException()
```

CheckParameters_NonNumericX_ThrowsException()

Tests that parameter validation rejects non-numeric x coordinate.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericX_ThrowsException()
```

CheckParameters_NonNumericY_ThrowsException()

Tests that parameter validation rejects non-numeric y coordinate.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericY_ThrowsException()
```

CheckParameters_ValidParameters_SetsProperties()

Tests that parameter validation accepts valid parameters.

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

Constructor_InvalidSize_ThrowsException()

Tests that constructor rejects zero size.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidSize_ThrowsException()
```

Constructor_NegativeSize_ThrowsException()

Tests that constructor rejects negative size.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_NegativeSize_ThrowsException()
```

Constructor_ValidParameters_CreatesHexagon()

Tests that constructor accepts valid x, y, and size values.

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

Execute_DrawsHexagon()

Tests that Execute method draws the hexagon. The actual drawing cannot be verified, only the execution completion.

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

SetUp()

Initializes test environment before each test.

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

Class AppMoveToTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppMoveTo class. Verifies cursor movement functionality and coordinate validation.

```
[TestClass]
public class AppMoveToTests
```

Inheritance

object ← AppMoveToTests

Inherited Members

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

Methods

CheckParameters_EmptyParameters_ThrowsException()

Tests that parameter validation rejects empty parameter array.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_EmptyParameters_ThrowsException()
```

CheckParameters_NonNumericX_ThrowsException()

Tests that parameter validation rejects non-numeric x coordinate.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericX_ThrowsException()
```

CheckParameters_NonNumericY_ThrowsException()

Tests that parameter validation rejects non-numeric y coordinate.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericY_ThrowsException()
```

Constructor_ValidCoordinates_CreatesMoveTo()

Tests that constructor accepts valid coordinates.

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

Execute_LargeCoordinates_MovesCorrectly()

Tests that Execute method handles large coordinate values correctly.

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

Execute_MultipleMoves_UpdatesCorrectly()

Tests that multiple moves update canvas position correctly.

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

Execute_NegativeCoordinates_ThrowsException()

Tests that Execute method rejects negative coordinates.

```
[TestMethod]
[ExpectedException(typeof(CanvasException))]
```

```
public void Execute_NegativeCoordinates_ThrowsException()
```

Execute_NegativeX_ThrowsException()

Tests that Execute method rejects negative x coordinate.

```
[TestMethod]
[ExpectedException(typeof(CanvasException))]
public void Execute_NegativeX_ThrowsException()
```

Execute_NegativeY_ThrowsException()

Tests that Execute method rejects negative y coordinate.

```
[TestMethod]
[ExpectedException(typeof(CanvasException))]
public void Execute_NegativeY_ThrowsException()
```

Execute_UpdatesCanvasPosition()

Tests that Execute method updates canvas position correctly.

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

SetUp()

Initializes test environment before each test.

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

Class AppPenTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppPen class. Verifies pen color setting functionality and RGB value validation.

```
[TestClass]
public class AppPenTests
```

Inheritance

object ← AppPenTests

Inherited Members

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

Methods

CheckParameters_InvalidParameters_ThrowsException()

Tests that parameter validation rejects invalid RGB values.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_InvalidParameters_ThrowsException()
```

Constructor_InvalidGreen_ThrowsException()

Tests that constructor rejects RGB green value below 0.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidGreen_ThrowsException()
```

Constructor_InvalidRed_ThrowsException()

Tests that constructor rejects RGB red value above 255.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidRed_ThrowsException()
```

Constructor_ValidColors_CreatesPen()

Tests that constructor accepts valid RGB values.

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

Execute_SetsCorrectColor()

Tests that Execute method sets the correct RGB color on the canvas.

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

SetUp()

Initializes test environment before each test.

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

Class AppRectangleTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppRectangle class. Verifies rectangle drawing functionality and parameter validation.

```
[TestClass]
public class AppRectangleTests
```

Inheritance

object ← AppRectangleTests

Inherited Members

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

Methods

CheckParameters_InsufficientParameters_ThrowsException()

Tests that parameter validation rejects insufficient parameters.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_InsufficientParameters_ThrowsException()
```

CheckParameters_NonNumericHeight_ThrowsException()

Tests that parameter validation rejects non-numeric height.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericHeight_ThrowsException()
```

CheckParameters_NonNumericWidth_ThrowsException()

Tests that parameter validation rejects non-numeric width.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericWidth_ThrowsException()
```

CheckParameters_ValidParameters_SetsProperties()

Tests that parameter validation accepts valid parameters.

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

Constructor_InvalidHeight_ThrowsException()

Tests that constructor rejects negative height.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidHeight_ThrowsException()
```

Constructor_InvalidWidth_ThrowsException()

Tests that constructor rejects zero width.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidWidth_ThrowsException()
```

Constructor_ValidDimensions_CreatesRectangle()

Tests that constructor accepts valid width and height values.

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

Constructor_WithFilled_CreatesFilledRectangle()

Tests that constructor correctly handles filled rectangle creation.

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

Execute_DrawsRectangle()

Tests that Execute method draws the rectangle. The actual drawing cannot be verified, only the execution completion.

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

SetUp()

Initializes test environment before each test.

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

Class AppStickmanTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppStickman class. Verifies stickman drawing functionality and parameter validation.

```
[TestClass]
public class AppStickmanTests
```

Inheritance

object ← AppStickmanTests

Inherited Members

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

Methods

CheckParameters_InsufficientParameters_ThrowsException()

Tests that parameter validation rejects insufficient parameters.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_InsufficientParameters_ThrowsException()
```

CheckParameters_NonNumericSize_ThrowsException()

Tests that parameter validation rejects non-numeric size.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericSize_ThrowsException()
```

CheckParameters_NonNumericX_ThrowsException()

Tests that parameter validation rejects non-numeric x coordinate.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericX_ThrowsException()
```

CheckParameters_NonNumericY_ThrowsException()

Tests that parameter validation rejects non-numeric y coordinate.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericY_ThrowsException()
```

CheckParameters_ValidParameters_SetsProperties()

Tests that parameter validation accepts valid parameters.

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

Constructor_InvalidSize_ThrowsException()

Tests that constructor rejects zero size.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidSize_ThrowsException()
```

Constructor_NegativeSize_ThrowsException()

Tests that constructor rejects negative size.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_NegativeSize_ThrowsException()
```

Constructor_ValidParameters_CreatesStickman()

Tests that constructor accepts valid x, y, and size values.

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

Execute_DrawsStickman()

Tests that Execute method draws the stickman. The actual drawing cannot be verified, only the execution completion.

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

SetUp()

Initializes test environment before each test.

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

Class AppTriangleTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppTriangle class. Verifies triangle drawing functionality and parameter validation.

```
[TestClass]
public class AppTriangleTests
```

Inheritance

object ← AppTriangleTests

Inherited Members

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

Methods

CheckParameters_EmptyParameters_ThrowsException()

Tests that parameter validation rejects empty parameter array.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_EmptyParameters_ThrowsException()
```

CheckParameters_InsufficientParameters_ThrowsException()

Tests that parameter validation rejects insufficient parameters.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_InsufficientParameters_ThrowsException()
```

CheckParameters_NonNumericHeight_ThrowsException()

Tests that parameter validation rejects non-numeric height.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericHeight_ThrowsException()
```

CheckParameters_NonNumericWidth_ThrowsException()

Tests that parameter validation rejects non-numeric width.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_NonNumericWidth_ThrowsException()
```

CheckParameters_ValidParameters_SetsProperties()

Tests that parameter validation accepts valid parameters.

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

Constructor_InvalidHeight_ThrowsException()

Tests that constructor rejects negative height.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidHeight_ThrowsException()
```

Constructor_InvalidWidth_ThrowsException()

Tests that constructor rejects zero width.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_InvalidWidth_ThrowsException()
```

Constructor_ValidDimensions_CreatesTriangle()

Tests that constructor accepts valid width and height values.

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

Execute_ValidTriangle_DrawsCorrectly()

Tests that Execute method draws the triangle. The actual drawing cannot be verified, only the execution completion.

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

SetUp()

Initializes test environment before each test.

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

Class AppWriteTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the AppWrite class. Verifies text writing functionality including string literals, expressions, and variable substitution.

```
[TestClass]
public class AppWriteTests
```

Inheritance

object ← AppWriteTests

Inherited Members

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

Methods

CheckParameters_InsufficientParameters_ThrowsException()

Tests parameter validation for insufficient parameters.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void CheckParameters_InsufficientParameters_ThrowsException()
```

Constructor_EmptyText_ThrowsException()

Tests that constructor rejects empty text input.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void Constructor_EmptyText_ThrowsException()
```

Constructor_ValidText_CreatesWrite()

Tests that constructor accepts valid text input.

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

ProcessText_Expression_HandlesCorrectly()

Tests handling of expressions with variables.

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

ProcessText_MultipleExpressions_HandlesCorrectly()

Tests handling of multiple expressions and concatenations.

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

ProcessText_StringConcatenation_HandlesCorrectly()

Tests string concatenation with variables.

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

ProcessText_StringLiteral_HandlesCorrectly()

Tests handling of string literal text.

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

SetUp()

Initializes test environment before each test.

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

Class CommandHandlersTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the CommandHandlers class. Verifies program storage, variable handling, and control flow functionality.

```
[TestClass]
public class CommandHandlersTests
```

Inheritance

object ← CommandHandlersTests

Inherited Members

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

Methods

Arrays_Operations_WorkCorrectly()

Tests that array operations work correctly including initialization, element access, and modification.

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

IfElse_Condition_ExecutesCorrectBranch()

Tests that if-else control flow executes correct branch.

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

IfElse_ElseBlock_ExecutesCorrectly()

Tests that else blocks are handled correctly.

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

IfElse_NestedConditions_ExecutesCorrectly()

Tests that nested if statements work correctly.

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

SetUp()

Initializes test environment before each test.

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

Variables_SetAndGet_StoresCorrectly()

Tests that variables can be stored and retrieved correctly.

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

Class ExpressionParserTests

Namespace: ASE_Tests

Assembly: ASE Tests.dll

Tests for the ExpressionParser class. Verifies expression parsing and evaluation functionality.

```
[TestClass]
public class ExpressionParserTests
```

Inheritance

object ← ExpressionParserTests

Inherited Members

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

Methods

EvaluateExpression_ArithmeticExpression_ReturnsCorrectResult()

Tests that arithmetic expressions are evaluated correctly.

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

EvaluateExpression_InvalidExpression_ThrowsException()

Tests that invalid expressions throw appropriate exception.

```
[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void EvaluateExpression_InvalidExpression_ThrowsException()
```

EvaluateExpression_NumericValue_ReturnsCorrectResult()

Tests that numeric expressions are evaluated correctly.

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

EvaluateExpression_VariableValue_ReturnsCorrectResult()

Tests that variable expressions are evaluated correctly.

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

SetUp()

Initializes test environment before each test.

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