Namespace DrawingApplication

Classes

[Application_UI](#)

UI For DrawingApplication

[Array](#)

A Class that implements Array without restrictions through calling the existing BOOSE.Array Class and modifying its methods

[AsCanvas](#)

A class that implements the abstract commands of ICanvas.

[AsCommandFactory](#)

A Class of BOOSE Commands

[AsParser](#)

Parser splits the program in to indivdual lines then runs the line through AsStoredProgram so it can be run and checks if the commands exist by checking AsCommandFactory

[AsStoredProgram](#)

A Class that extends StoredProgram to remove restrictions of size of program

[Boolean](#)

[Clear](#)

Class to implement the Clear() command from the AsCommandFactory.

[CompoundCommand](#)

[Else](#)

[End](#)

[For](#)

[If](#)

A class that implements the if command from the AsCommandFactory.

[Int](#)

A Class that implements the Variable Int without restrictions through calling the existing BOOSE.Int Class and modifying its restriction method

[Method](#)

A Class that implements the Variable Method without restrictions through calling the existing BOOSE.Method Class and modifying its restriction method

[Real](#)
   A Class that implements the Variable Real without restrictions through calling the existing BOOSE.Real Class and modifying its restriction method

[Rect](#)
   Class to implement the Rect() command from the AsCommandFactory.

[Reset](#)
   A class that implements the Reset() command from the AsCommandFactory.

[Tri](#)
   A class that implements the Tri() command from the AsCommandFactory.

[Triforce](#)
   Class to implement the Triforce() command from the AsCommandFactory.

[While](#)

[Write](#)
   A Class that outputs and Expression to the Console and Canvas. This is done by Evaluating the

[WriteText](#)
   A class that implements the WriteText() command from the AsCommandFactory.

Class Application_UI

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

UI For DrawingApplication

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

Inheritance

[object](#) ← [MarshalByRefObject](#) ← [Component](#) ← [Control](#) ← [ScrollableControl](#) ←
[ContainerControl](#) ← [Form](#) ← Application_UI

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)](#) ,

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

Application_UI()

```
public Application_UI()
```

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 Array

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A Class that implements Array without restrictions through calling the existing BOOSE.Array Class and modifying its methods

```
public class Array : Array, ICommand
```

Inheritance

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

Implements

ICommand

Inherited Members

Array.PEEK , Array.POKE , Array.type , Array.rows , Array.columns , Array.valueInt , Array.valueReal , Array.intArray , Array.realArray , Array.pokeValue , Array.peekVar , Array.rowS , Array.columnS , Array.row , Array.column , Array.Rows , Array.Columns , Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList , Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) , [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

Array()

Constructor called from AsCommandFactory

```
public Array()
```

Methods

ArrayRestrictions()

Method to imposes a limit on the amount of Arrays

```
public void ArrayRestrictions()
```

Exceptions

RestrictionException

An exception is given when the amount of Arrays exceeds its Max

CheckParameters(string[])

Checks if the correct number of parameteres

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

Parameters

parameterList string[]

Exceptions

CommandException

If the commmand is formated incorrecty or invalid parameters

Compile()

A method that checks that the Array is formated correctly

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

Exceptions

CommandException

An exception if the Array command isnt formated correctly

Execute()

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

GetIntArray(int, int)

```
public override int GetIntArray(int row, int col)
```

Parameters

row int↗

col int↗

Returns

int↗

GetRealArray(int, int)

```
public override double GetRealArray(int row, int col)
```

Parameters

row int↗

col int↗

Returns

double↗

ProcessArrayParametersCompile(bool)

```
protected override void ProcessArrayParametersCompile(bool peekOrPoke)
```

Parameters

peekOrPoke [bool](#)⧉

ProcessArrayParametersExecute(bool)

```
protected override void ProcessArrayParametersExecute(bool peekOrPoke)
```

Parameters

peekOrPoke [bool](#)⧉

ReduceRestrictionCounter()

Reduces the Max Array limit

```
protected void ReduceRestrictionCounter()
```

SetIntArray(int, int, int)

```
public override void SetIntArray(int val, int row, int col)
```

Parameters

val [int](#)⧉

row [int](#)⧉

col [int](#)⧉

SetRealArray(double, int, int)

```csharp
public override void SetRealArray(double val, int row, int col)
```

Parameters

val double↗

row int↗

col int↗

Class AsCanvas

Namespace: DrawingApplication

Assembly: DrawingApplication.dll

A class that implements the abstract commands of ICanvas.

```
public class AsCanvas : ICanvas
```

Inheritance

object ← AsCanvas

Implements

ICanvas

Inherited Members

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

Constructors

AsCanvas()

Constructor that calls Set().

```
public AsCanvas()
```

Properties

PenColour

Gets or sets the color of the pen used for drawing.

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

Property Value

object

Xpos

Gets or sets the X position on the canvas.

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

Property Value

int

Ypos

Gets or sets the Y position on the canvas.

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

Property Value

int

Methods

Circle(int, bool)

Draws a circle on the canvas/bitmap.

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

Parameters

`radius` int

The radius of the circle.

`filled` bool

Indicates whether the circle is filled or not.

Exceptions

CanvasException

Thrown if the radius is invalid.

Clear()

Clears the canvas/bitmap

```
public void Clear()
```

DrawTo(int, int)

Draws a line from the current position to a specified (x, y) position.

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

Parameters

x int ⧉

The x-coordinate.

y int ⧉

The y-coordinate.

Exceptions

CanvasException

Thrown if the position is invalid.

MoveTo(int, int)

Moves the current position on the canvas to the specified (x, y) coordinates.

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

Parameters

x int⤤

　The x-coordinate.

y int⤤

　The y-coordinate.

Exceptions

CanvasException

　Thrown if the position is invalid.

Rect(int, int, bool)

Draws a rectangle on the canvas/bitmap.

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

Parameters

width int⤤

　The width of the rectangle.

height int⤤

　The height of the rectangle.

filled bool⤤

　Indicates whether the rectangle is filled or not.

Exceptions

CanvasException

    Thrown if the dimensions are invalid.

Reset()

Resets the position on the canvas/bitmap

```
public void Reset()
```

Set(int, int)

Initializes the canvas with specified dimensions.

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

Parameters

`width` [int↗]

    The width of the PictureBox.

`height` [int↗]

    The height of the PictureBox.

SetColour(int, int, int)

Sets the color of the pen using RGB values.

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

Parameters

`red` [int↗]

    The red component (0-255).

green int⧉

    The green component (0-255).

blue int⧉

    The blue component (0-255).

Exceptions

CanvasException

    Thrown if RGB values are invalid.

Tri(int, int)

Draws a triangle on the canvas/bitmap.

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

Parameters

width int⧉

    The width of the triangle.

height int⧉

    The height of the triangle.

Exceptions

CanvasException

    Thrown if the values are not valid.

WriteText(string)

Writes text onto the canvas/bitmap.

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

Parameters

text string⤢

  The text to write.

getBitmap()

Returns the bitmap containing the graphics.

```
public object getBitmap()
```

Returns

object⤢

  The Bitmap

Class AsCommandFactory

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A Class of BOOSE Commands

```
public class AsCommandFactory : CommandFactory, ICommandFactory
```

Inheritance

[object](#) ← CommandFactory ← AsCommandFactory

Implements

ICommandFactory

Inherited Members

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

Methods

MakeCommand(string)

This is a class off all the commands

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

Parameters

`commandType` [string](#)

Returns

ICommand

Exceptions

FactoryException

Class AsParser

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

Parser splits the program in to indivdual lines then runs the line through AsStoredProgram so it can be run and checks if the commands exist by checking AsCommandFactory

```
public class AsParser : IParser
```

Inheritance

[object](#) ← AsParser

Implements

IParser

Inherited Members

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

Constructors

AsParser(CommandFactory, StoredProgram)

Constructor

```
public AsParser(CommandFactory Factory, StoredProgram Program)
```

Parameters

`Factory` CommandFactory

CommandFactory to check if command exist

`Program` StoredProgram

StoredProgram to run the program

Methods

ParseCommand(string)

Parses a command so it has functionality

```
public ICommand ParseCommand(string Line)
```

Parameters

Line string↗

   lines of the program

Returns

ICommand

Exceptions

ParserException

   An exception if varible doesnt exist or is invalid

ParseProgram(string)

Parsers the Program by checking each line as a command to check if the syntax of the commands are correct if not an error is given

```
public void ParseProgram(string program)
```

Parameters

program string↗

   StoredProgram to check syntax

Exceptions

ParserException

an error given if the syntax is invalid

Class AsStoredProgram

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A Class that extends StoredProgram to remove restrictions of size of program

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

Inheritance

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

Implements

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

Inherited Members

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

Constructors

AsStoredProgram(ICanvas)

Constructor to make an instance of AsStoredProgram

```
public AsStoredProgram(ICanvas canvas)
```

Parameters

`canvas` ICanvas

Methods

Run()

A method that runs the program by going through by command after command

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

Exceptions

RestrictionException

An exception that is reached if program size limit is reached

StoredProgramException

An exception to catch syntax errors or infinite loops

Class Boolean

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

```
public class Boolean : Boolean, ICommand
```

Inheritance

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

Implements

ICommand

Inherited Members

Boolean.Compile() , Boolean.Execute() , Boolean.BoolValue , Evaluation.expression ,
Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
[Evaluation.CheckParameters(string[])](#) , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression ,
Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) ,
[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

Boolean()

```
public Boolean()
```

Methods

Restrictions()

```
public override void Restrictions()
```

Class Clear

Namespace: DrawingApplication

Assembly: DrawingApplication.dll

Class to implement the Clear() command from the AsCommandFactory.

```
public class Clear : CommandOneParameter, ICommand
```

Inheritance

object ← Command ← CanvasCommand ← CommandOneParameter ← Clear

Implements

ICommand

Inherited Members

CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,
CommandOneParameter.CheckParameters(string[]) , CanvasCommand.yPos , CanvasCommand.xPos ,
CanvasCommand.canvas , CanvasCommand.Canvas , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , Command.Set(StoredProgram, string) ,
Command.Compile() , 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

Clear()

Default constructor that is called from AsCommandFactory.

```
public Clear()
```

Clear(Canvas)

Constructor that initializes the Clear command with a specific canvas.

```
public Clear(Canvas c)
```

Parameters

c Canvas

   The canvas on which the clear operation will be performed.

Methods

Execute()

Executes the Clear command by calling the Clear() method of the AsCanvas class.

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

Class CompoundCommand

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

```
public class CompoundCommand : ConditionalCommand, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← Boolean ← ConditionalCommand ← CompoundCommand

Implements

ICommand

Derived

[Else](#), [End](#)

Inherited Members

ConditionalCommand.endLineNumber , ConditionalCommand.Execute() ,
ConditionalCommand.EndLineNumber , ConditionalCommand.Condition ,
ConditionalCommand.LineNumber , ConditionalCommand.CondType ,
ConditionalCommand.ReturnLineNumber , Boolean.Restrictions() , Boolean.BoolValue ,
Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
[Evaluation.CheckParameters(string[])](#) , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression ,
Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) ,
[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

CompoundCommand()

Controls the variable Ends for if/while/for loops

```
public CompoundCommand()
```

Exceptions

RestrictionException

An Exception is given if the variable limit is reached

Properties

CorrespondingCommand

Getter and Setter

```
public ConditionalCommand CorrespondingCommand { get; set; }
```

Property Value

ConditionalCommand

Methods

Compile()

Compiles

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

ReduceRestrictions()

Reduces the variable limit

```
protected void ReduceRestrictions()
```

Class Else

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

```
public class Else : CompoundCommand, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← Boolean ← ConditionalCommand ← [CompoundCommand](#) ← Else

Implements

ICommand

Inherited Members

[CompoundCommand.CorrespondingCommand](#) , [CompoundCommand.ReduceRestrictions()](#) ,
ConditionalCommand.endLineNumber , ConditionalCommand.EndLineNumber ,
ConditionalCommand.Condition , ConditionalCommand.LineNumber , ConditionalCommand.CondType ,
ConditionalCommand.ReturnLineNumber , Boolean.Restrictions() , Boolean.BoolValue ,
Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
[Evaluation.CheckParameters(string[])](#) , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression ,
Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) ,
[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

Else()

```
public Else()
```

Properties

CorrespondingEnd

```
public End CorrespondingEnd { get; set; }
```

Property Value

[End](#)

Methods

Compile()

Compiles

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

Execute()

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

Class End

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

```
public class End : CompoundCommand, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← Boolean ← ConditionalCommand ← [CompoundCommand](#) ← End

Implements

ICommand

Inherited Members

[CompoundCommand.CorrespondingCommand](#) , [CompoundCommand.ReduceRestrictions()](#) , ConditionalCommand.endLineNumber , ConditionalCommand.EndLineNumber , ConditionalCommand.Condition , ConditionalCommand.LineNumber , ConditionalCommand.CondType , ConditionalCommand.ReturnLineNumber , Boolean.Restrictions() , Boolean.BoolValue , Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value , [Evaluation.CheckParameters(string[])](#) , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList , Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) , [Command.ProcessParameters(string)](#) , Command.ToString() , Command.Program , Command.Name , Command.ParameterList , Command.Parameters , Command.Paramsint , [object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Methods

Compile()

Comiles the loops and checks if the there is a loop that need an end

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

Exceptions

CommandException

An exception if loop is missing

Execute()

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

Exceptions

RestrictionException

CommandException

Class For

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

```
public class For : ConditionalCommand, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← Boolean ← ConditionalCommand ← For

Implements

ICommand

Inherited Members

ConditionalCommand.endLineNumber , ConditionalCommand.EndLineNumber , ConditionalCommand.Condition , ConditionalCommand.LineNumber , ConditionalCommand.CondType , ConditionalCommand.ReturnLineNumber , Boolean.Restrictions() , Boolean.BoolValue , Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value , [Evaluation.CheckParameters(string[])](#) , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList , Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) , [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)](#)

Properties

From

Getter and Setter for From

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

Property Value

[int](#)

LoopControlV

```
public Evaluation LoopControlV { get; }
```

Property Value

Evaluation

Step

Getter and Setter for Step

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

Property Value

int⧉

To

Getter and Setter for To

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

Property Value

int⧉

Methods

Compile()

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

Execute()

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

Class If

Namespace: DrawingApplication

Assembly: DrawingApplication.dll

A class that implements the if command from the AsCommandFactory.

```
public class If : ConditionalCommand, ICommand
```

Inheritance

object ← Command ← Evaluation ← Boolean ← ConditionalCommand ← If

Implements

ICommand

Inherited Members

ConditionalCommand.endLineNumber , ConditionalCommand.Compile() ,
ConditionalCommand.Execute() , ConditionalCommand.EndLineNumber ,
ConditionalCommand.Condition , ConditionalCommand.LineNumber , ConditionalCommand.CondType ,
ConditionalCommand.ReturnLineNumber , Boolean.Restrictions() , Boolean.BoolValue ,
Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
Evaluation.CheckParameters(string[]) , Evaluation.ProcessExpression(string) , Evaluation.Expression ,
Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , Command.Set(StoredProgram, string) ,
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)

Class Int

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A Class that implements the Variable Int without restrictions through calling the existing BOOSE.Int Class and modifying its restriction method

```
public class Int : Int, ICommand
```

Inheritance

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

Implements

ICommand

Inherited Members

Int.Compile() , Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value , [Evaluation.CheckParameters(string[])](#) , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList , Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) , [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

Int()

Constructor that is called from AsCommandFactory.cs

```
public Int()
```

Methods

Execute()

Parses the value of the Int to see if valid then updates the variable

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

Exceptions

StoredProgramException

An exeption if value isnt valid

Restrictions()

Method to impose a limit on the amount of Int Variables

```
public override void Restrictions()
```

Exceptions

RestrictionException

An exception is given when the amount of int variables exceeds its Max

Class Method

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A Class that implements the Variable Method without restrictions through calling the existing BOOSE.Method Class and modifying its restriction method

```
public class Method : Method, ICommand
```

Inheritance

[object](#) ← Command ← Evaluation ← Boolean ← ConditionalCommand ← CompoundCommand ← Method ← Method

Implements

ICommand

Inherited Members

Method.Compile() , Method.LocalVariables , Method.MethodName , Method.Type , CompoundCommand.ReduceRestrictions() , CompoundCommand.CorrespondingCommand , ConditionalCommand.endLineNumber , ConditionalCommand.EndLineNumber , ConditionalCommand.Condition , ConditionalCommand.LineNumber , ConditionalCommand.CondType , ConditionalCommand.ReturnLineNumber , Boolean.Restrictions() , Boolean.BoolValue , Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value , [Evaluation.ProcessExpression(string)](#) , Evaluation.Expression , Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList , Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) , [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

Method()

Constructor that is called by AsCommandFactory and set limit on the amount of methods

```
public Method()
```

Exceptions

RestrictionException

> An Exception when the variable limit is reached

Methods

CheckParameters(string[])

needed due to interface but doesnt do anything

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

Parameters

parameter [string](#)[]

Execute()

Create the method variable by calling this to the Program and gets the line number for the method

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

Class Real

Namespace: DrawingApplication

Assembly: DrawingApplication.dll

A Class that implements the Variable Real without restrictions through calling the existing BOOSE.Real Class and modifying its restriction method

```
public class Real : Real, ICommand
```

Inheritance

object ← Command ← Evaluation ← Real ← Real

Implements

ICommand

Inherited Members

Real.Compile() , Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value , [Evaluation.CheckParameters(string[])](Evaluation.CheckParameters(string[])) , Evaluation.ProcessExpression(string) , Evaluation.Expression , Evaluation.VarName , Evaluation.Local , Command.program , Command.parameterList , Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](Command.Set(StoredProgram,string)) , Command.ProcessParameters(string) , Command.ToString() , Command.Program , Command.Name , Command.ParameterList , Command.Parameters , Command.Paramsint , object.Equals(object) , [object.Equals(object, object)](object.Equals(object,object)) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , [object.ReferenceEquals(object, object)](object.ReferenceEquals(object,object))

Constructors

Real()

Constructor Called from AsCommandFactory

```
public Real()
```

Properties

Value

Getter and Setter

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

Property Value

[double ↗](#)

Methods

Execute()

Parses the value of the Real to see if valid then updates the variable

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

Exceptions

StoredProgramException

An exeption if value isnt valid

Restrictions()

Method to imposes a limit on the amount of Real Variables

```
public override void Restrictions()
```

Exceptions

RestrictionException

An exception is given when the amount of Real variables exceeds its Max

Class Rect

Namespace: DrawingApplication

Assembly: DrawingApplication.dll

Class to implement the Rect() command from the AsCommandFactory.

```
public class Rect : CommandTwoParameters, ICommand
```

Inheritance

object ← Command ← CanvasCommand ← CommandOneParameter ← CommandTwoParameters ← Rect

Implements

ICommand

Inherited Members

CommandTwoParameters.param2 , CommandTwoParameters.param2unprocessed , CommandOneParameter.param1 , CommandOneParameter.param1unprocessed , CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas , Command.program , Command.parameterList , Command.parameters , Command.paramsint , Command.Set(StoredProgram, string) , Command.Compile() , 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

Rect()

Default constructor.

```
public Rect()
```

Rect(Canvas, int, int)

Constructor that initializes the Rect command with a specific canvas and dimensions.

```
public Rect(Canvas c, int width, int height)
```

Parameters

c Canvas

The canvas on which the rectangle will be drawn.

width int⧉

The width of the rectangle.

height int⧉

The height of the rectangle.

Methods

CheckParameters(string[])

Checks the number of parameters provided for the Rect command.

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

Parameters

parameterList string⧉[]

The list of parameters to check.

Exceptions

CommandException

Thrown if the number of parameters is invalid.

Execute()

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

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

Exceptions

RestrictionException

Thrown if the rectangle size exceeds allowed limits.

Class Reset

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A class that implements the Reset() command from the AsCommandFactory.

```
public class Reset : CommandOneParameter, ICommand
```

Inheritance

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

Implements

ICommand

Inherited Members

CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,
[CommandOneParameter.CheckParameters(string[])](#) , CanvasCommand.yPos , CanvasCommand.xPos ,
CanvasCommand.canvas , CanvasCommand.Canvas , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) ,
Command.Compile() , [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

Reset()

Default constructor.

```
public Reset()
```

Reset(Canvas)

Constructor that initializes the Reset command with a specific canvas.

```
public Reset(Canvas c)
```

Parameters

c Canvas

   The canvas to be reset.

Methods

Execute()

Executes the Reset command by calling the Reset method on the canvas.

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

Class Tri

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A class that implements the Tri() command from the AsCommandFactory.

```
public class Tri : CommandTwoParameters, ICommand
```

Inheritance

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

Implements
ICommand

Inherited Members
CommandTwoParameters.param2 , CommandTwoParameters.param2unprocessed ,
CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,
Command.program , Command.parameterList , Command.parameters , Command.paramsint ,
[Command.Set(StoredProgram, string)](#) , Command.Compile() , [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

Tri()

Default constructor.

```
public Tri()
```

Tri(Canvas, int, int)

Constructor that initializes the Tri command with a specific canvas and dimensions.

```
public Tri(Canvas c, int width, int height)
```

Parameters

c Canvas

The canvas on which the triangle will be drawn.

width int⧉

The width of the triangle.

height int⧉

The height of the triangle.

Methods

CheckParameters(string[])

Checks the number of parameters provided for the Tri command.

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

Parameters

parameterList string⧉[]

The list of parameters to check.

Exceptions

CommandException

Thrown if the number of parameters is invalid.

Execute()

Executes the Tri command by calling the Tri method on the canvas.

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

Exceptions

RestrictionException

    Thrown if the dimensions exceed allowed limits.

Class Triforce

Namespace: DrawingApplication

Assembly: DrawingApplication.dll

Class to implement the Triforce() command from the AsCommandFactory.

```
public class Triforce : CommandOneParameter, ICommand
```

Inheritance

object ← Command ← CanvasCommand ← CommandOneParameter ← Triforce

Implements

ICommand

Inherited Members

CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,
Command.program , Command.parameterList , Command.parameters , Command.paramsint ,
[Command.Set(StoredProgram, string)](Command.Set) , Command.Compile() , [Command.ProcessParameters(string)](Command.ProcessParameters) ,
Command.ToString() , Command.Program , Command.Name , Command.ParameterList ,
Command.Parameters , Command.Paramsint , [object.Equals(object)](object.Equals) , [object.Equals(object, object)](object.Equals) ,
[object.GetHashCode()](object.GetHashCode) , [object.GetType()](object.GetType) , [object.MemberwiseClone()](object.MemberwiseClone) ,
[object.ReferenceEquals(object, object)](object.ReferenceEquals)

Constructors

Triforce()

Default constructor.

```
public Triforce()
```

Triforce(Canvas, int)

Constructor that initializes the Rect command with parameters

```
public Triforce(Canvas c, int size)
```

Parameters

c Canvas

The canvas on which the triforce will be drawn.

size int↗

The size of the triforce.

Methods

CheckParameters(string[])

Checks the number of parameters provided for the triforce command.

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

Parameters

parameterList string↗[]

The list of parameters to check.

Exceptions

CommandException

Thrown if the number of parameters is invalid.

Execute()

Executes multiple Tri commands to draw a triforce on the canvas.

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

Exceptions

RestrictionException

Thrown if the size exceeds allowed limits.

Class While

Namespace: DrawingApplication

Assembly: DrawingApplication.dll

```
public class While : ConditionalCommand, ICommand
```

Inheritance

object ← Command ← Evaluation ← Boolean ← ConditionalCommand ← While

Implements

ICommand

Inherited Members

ConditionalCommand.endLineNumber , ConditionalCommand.Compile() ,
ConditionalCommand.Execute() , ConditionalCommand.EndLineNumber ,
ConditionalCommand.Condition , ConditionalCommand.LineNumber , ConditionalCommand.CondType ,
ConditionalCommand.ReturnLineNumber , Boolean.Restrictions() , Boolean.BoolValue ,
Evaluation.expression , Evaluation.evaluatedExpression , Evaluation.varName , Evaluation.value ,
Evaluation.CheckParameters(string[]) , Evaluation.ProcessExpression(string) , Evaluation.Expression ,
Evaluation.VarName , Evaluation.Value , Evaluation.Local , Command.program , Command.parameterList ,
Command.parameters , Command.paramsint , Command.Set(StoredProgram, string) ,
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)

Class Write

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A Class that outputs and Expression to the Console and Canvas. This is done by Evaluating the

```csharp
public class Write : CanvasCommand, ICommand
```

Inheritance

[object](#) ← Command ← CanvasCommand ← Write

Implements

ICommand

Inherited Members

CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas , Command.program , Command.parameterList , Command.parameters , Command.paramsint , [Command.Set(StoredProgram, string)](#) , Command.Compile() , [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

Write()

Constructor

```csharp
public Write()
```

Methods

CheckParameters(string[])

A mehtod that checks the amount of parmeters and if they are valid

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

Parameters

parameter [string](#)[]

Exceptions

CommandException

Execute()

Get an Expresion from the user and outputs it to the console and canvas

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

Class WriteText

Namespace: [DrawingApplication](#)

Assembly: DrawingApplication.dll

A class that implements the WriteText() command from the AsCommandFactory.

```
public class WriteText : CommandOneParameter, ICommand
```

Inheritance

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

Implements

ICommand

Inherited Members

CommandOneParameter.param1 , CommandOneParameter.param1unprocessed ,
CanvasCommand.yPos , CanvasCommand.xPos , CanvasCommand.canvas , CanvasCommand.Canvas ,
Command.program , Command.parameterList , Command.parameters , Command.paramsint ,
[Command.Set(StoredProgram, string)](#) , Command.Compile() , [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

WriteText()

Default constructor.

```
public WriteText()
```

WriteText(Canvas, string)

Constructor that initializes the WriteText command with a specific canvas and text.

```
public WriteText(Canvas c, string text)
```

Parameters

c Canvas

The canvas where text will be written.

text [string↗](#)

The text to be written.

Methods

CheckParameters(string[])

Checks the number of parameters provided for the WriteText command.

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

Parameters

parameterList [string↗](#)[]

The list of parameters to check.

Exceptions

CommandException

Thrown if the number of parameters is invalid.

Execute()

Executes the WriteText command by calling the WriteText method on the canvas.

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

Exceptions

RestrictionException

Thrown if the text is empty.