- **LineStrategy**

```
using System;
using System.Drawing;
using System.Windows.Forms;


   namespace Strategy
   {
       /// <summary>
       /// Implements the line drawing strategy
       /// </summary>
       public class LineStrategy : TwoPointStrategy
       {
           #region Protected Methods
           protected override void Draw(object sender, PaintEventArgs e)
           {
               if (_points != null)
               {
                   var graphics = e.Graphics;
                   graphics.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
                   graphics.DrawLine(new Pen(_color, _thickness),
_points[0], _points[1]);
               }
           }
           #endregion


           #region Public Methods
           public override string GetDescription()
           {
               if (!_hasDrawn)
               {
```

```
                    return "Nothing drawn";
                }

                if (_points != null)
                {
                    return $"Draw line from ({_points[0].X},
    {_points[0].Y}) to ({_points[1].X}, {_points[1].Y})";
                }
                return "Something wrong";
            }
            #endregion
        }
    }
```

- **View - ChangeCurrentHandler**

```
        /// <summary>
        /// Removes the old AddedPaintHandler to the pictureBox.Paint event
and adds
        /// the one encapsulated in the strategy.
        /// </summary>
        /// <param name="strategy">New strategy encapsulating a
PaintHandler</param>
        public void ChangeCurrentHandler(Strategy strategy)
        {
            pictureBox.Paint -= _currentAddedPaintHandler;
            _currentAddedPaintHandler = strategy.GetDraw();
            pictureBox.Paint += _currentAddedPaintHandler;
        }
```

- **DrawingMemento**

```
/// <summary>
/// Constructor for the DrawingMemento - initializes the image
/// </summary>
/// <param name="drawing">Image/drawing to be wrapped</param>
/// <param name="drawing">The description of that image</param>
public DrawingMemento(Image drawing, string description)
{
    _drawing = drawing;
    _description = description;
}
```

- **UndoStack – Add**

```
/// <summary>
/// Adds a drawing to the state stack
/// </summary>
/// <param name="drawing">The saved drawing</param>
/// <returns>void</returns>
public void Add(DrawingMemento drawing)
{
    if (_stack.Count != 0 && _current != null)
    {
        while (_current != _stack.Last)
        {
            _stack.RemoveLast();
        }
    }
```

```
            _stack.AddLast(drawing);
            _current = _stack.Last;
        }
```

- **Presenter – ChoosePaintingTool**

```
        /// <summary>
        /// Updates the current strategy to match the selected tool
        /// </summary>
        /// <param name="paintingTool">The new selected painting tool</param>
        /// <param name="borderColor">The color of the border of the painting
tool</param>
        /// <param name="fillColor">The fill color of the painting
tool</param>
        public void ChoosePaintingTool(PaintingTool paintingTool, Color
borderColor, Color fillColor, float thickness)
        {
            if (_currentStrategy.Done)
            {
                CaptureAndAddMemento();
            }

            var newStrategy = _model.GetPaintingStrategy(paintingTool);
            _view.ChangeCurrentHandler(newStrategy);
            _currentStrategy = newStrategy;
            ColorChanged(borderColor);
            FillColorChanged(fillColor);
            ThicknessChanged(thickness);
        }
```