```
#region Using declarations
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;
using System.Xml.Serialization;
using NinjaTrader.Cbi;
using NinjaTrader.Gui;
using NinjaTrader.Gui.Chart;
using NinjaTrader.Gui.SuperDom;
using NinjaTrader.Gui.Tools;
using NinjaTrader.Data;
using NinjaTrader.NinjaScript;
using NinjaTrader.Core.FloatingPoint;
using NinjaTrader.NinjaScript.Indicators;
using NinjaTrader.NinjaScript.DrawingTools;
#endregion

//This namespace holds Strategies in this folder and is required. Do not change it.
namespace NinjaTrader.NinjaScript.Strategies
{
    public class RSITestv2 : Strategy
    {
        private RSI RSI1;
        private Stochastics Stochastics1;

        protected override void OnStateChange()
        {
            if (State == State.SetDefaults)
            {
                Description                                 = @"Enter the description for your new custom Strategy here.";
                Name                                        = "RSITestv2";
                Calculate                                   = Calculate.OnBarClose;
                EntriesPerDirection                         = 1;
                EntryHandling                               = EntryHandling.AllEntries;
                IsExitOnSessionCloseStrategy                = true;
                ExitOnSessionCloseSeconds                   = 30;
                IsFillLimitOnTouch                          = false;
                MaximumBarsLookBack                         = MaximumBarsLookBack.TwoHundredFiftySix;
                OrderFillResolution                         = OrderFillResolution.Standard;
                Slippage                                    = 0;
                StartBehavior                               = StartBehavior.WaitUntilFlat;
                TimeInForce                                 = TimeInForce.Gtc;
                TraceOrders                                 = false;
                RealtimeErrorHandling                       = RealtimeErrorHandling.StopCancelClose;
                StopTargetHandling                          = StopTargetHandling.PerEntryExecution;
                BarsRequiredToTrade                         = 20;
                // Disable this property for performance gains in Strategy Analyzer optimizations
                // See the Help Guide for additional information
                IsInstantiatedOnEachOptimizationIteration   = true;
```

```
            }
            else if (State == State.Configure)
            {
            }
            else if (State == State.DataLoaded)
            {
                RSI1                = RSI(Close, 14, 3);
                Stochastics1                = Stochastics(Close, 7, 14, 3);
            }
        }

        protected override void OnBarUpdate()
        {
            if (BarsInProgress != 0)
                return;

            if (CurrentBars[0] < 1)
            return;

             // Set 1
            {
            if (
                 // Overbought
                ((RSI1.Avg[0] >= 70)
                 || (IsFalling(Stochastics1.D) == true)))
            {
                Draw.ArrowUp(this, @"RSITestv2 Arrow up_1", false, 0, 0, Brushes.Maroon);
            }

             // Set 2
            else if (
                 // Oversold
                ((RSI1.Avg[0] <= 30)
                 || (IsRising(Stochastics1.D) == true)))
            {
                Draw.ArrowDown(this, @"RSITestv2 Arrow down_1", false, 0, 0, Brushes.DarkGreen);
            }
            }
        }
    }
}
```