عنوان: **پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #2**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۰:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: OOP, C#, Project, Paint

در ادامه مطلب پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #1 به تشریح مابقی کلاس‌های برنامه می‌پردازیم.

با توجه به تجزیه و تحلیل انجام شده تمامی اشیا از کلاس پایه به نام Shape ارث بری دارند حال به توضیح کدهای این کلاس می‌پردازیم. (به دلیل اینکه توضیحات این کلاس در دو پست نوشته خواهد شد برای این کلاس‌ها از partial class استفاده شده است)

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Net;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Shape (Base Class)
    /// </summary>
    public abstract partial class Shape
    {
        #region Fields (1)

        private Brush _backgroundBrush;

        #endregion Fields

        #region Properties (16)

        /// <summary>
        /// Gets or sets the brush.
        /// </summary>
        /// <value>
        /// The brush.
        /// </value>
        public Brush BackgroundBrush
        {
            get { return _backgroundBrush ?? (_backgroundBrush = new SolidBrush(BackgroundColor)); }
            private set
            {
                _backgroundBrush = value ?? new SolidBrush(BackgroundColor);
            }
        }

        /// <summary>
        /// Gets or sets the color of the background.
        /// </summary>
        /// <value>
        /// The color of the background.
        /// </value>
        public Color BackgroundColor { get; set; }

        /// <summary>
        /// Gets or sets the end point.
        /// </summary>
        /// <value>
        /// The end point.
        /// </value>
        public PointF EndPoint { get; set; }

        /// <summary>
        /// Gets or sets the color of the fore.
        /// </summary>
        /// <value>
        /// The color of the fore.
        /// </value>
        public Color ForeColor { get; set; }

        /// <summary>
        /// Gets or sets the height.
        /// </summary>
        /// <value>
```

```
        /// The height.
        /// </value>
        public float Height
        {
            get
            {
                return Math.Abs(StartPoint.Y - EndPoint.Y);
            }
            set
            {
                if (value > 0)
                    EndPoint = new PointF(EndPoint.X, StartPoint.Y + value);
            }
        }

        /// <summary>
        /// Gets or sets a value indicating whether this instance is fill.
        /// </summary>
        /// <value>
        ///   <c>true</c> if this instance is fill; otherwise, <c>false</c>.
        /// </value>
        public bool IsFill { get; set; }

        /// <summary>
        /// Gets or sets a value indicating whether this instance is selected.
        /// </summary>
        /// <value>
        /// <c>true</c> if this instance is selected; otherwise, <c>false</c>.
        /// </value>
        public bool IsSelected { get; set; }

        /// <summary>
        /// Gets or sets my pen.
        /// </summary>
        /// <value>
        /// My pen.
        /// </value>
        public Pen Pen
        {
            get
            {
                return new Pen(ForeColor, Thickness);
            }
        }

        /// <summary>
        /// Gets or sets the type of the shape.
        /// </summary>
        /// <value>
        /// The type of the shape.
        /// </value>
        public ShapeType ShapeType { get; protected set; }

        /// <summary>
        /// Gets the size.
        /// </summary>
        /// <value>
        /// The size.
        /// </value>
        public SizeF Size
        {
            get
            {
                return new SizeF(Width, Height);
            }
        }

        /// <summary>
        /// Gets or sets the start point.
        /// </summary>
        /// <value>
        /// The start point.
        /// </value>
        public PointF StartPoint { get; set; }

        /// <summary>
        /// Gets or sets the thickness.
        /// </summary>
        /// <value>
        /// The thickness.
        /// </value>
```

```
        public byte Thickness { get; set; }

        /// <summary>
        /// Gets or sets the width.
        /// </summary>
        /// <value>
        /// The width.
        /// </value>
        public float Width
        {
            get
            {
                return Math.Abs(StartPoint.X - EndPoint.X);
            }
            set
            {
                if (value > 0)
                    EndPoint = new PointF(StartPoint.X + value, EndPoint.Y);
            }
        }

        /// <summary>
        /// Gets or sets the X.
        /// </summary>
        /// <value>
        /// The X.
        /// </value>
        public float X
        {
            get
            {
                return StartPoint.X;
            }
            set
            {
                if (value > 0)
                    StartPoint = new PointF(value, StartPoint.Y);
            }
        }

        /// <summary>
        /// Gets or sets the Y.
        /// </summary>
        /// <value>
        /// The Y.
        /// </value>
        public float Y
        {
            get
            {
                return StartPoint.Y;
            }
            set
            {
                if (value > 0)
                    StartPoint = new PointF(StartPoint.X, value);
            }
        }

        /// <summary>
        /// Gets or sets the index of the Z.
        /// </summary>
        /// <value>
        /// The index of the Z.
        /// </value>
        public int Zindex { get; set; }

        #endregion Properties
        }
}
```

ابتدا به تشریح خصوصیات کلاس می‌پردازیم:

خصوصیات:

BackgroundColor : در صورتی که شی مورد نظر به صورت توپررسم شود، این خاصیت رنگ پس زمینه شی را مشخص می‌کند.
BackgroundBrush : خاصیتی است که با توجه به خاصیت BackgroundColor یک الگوی پر کردن زمینه شی می‌سازد.
StartPoint : نقطه شروع شی را در خود نگهداری می‌کند.
EndPoint : نقطه انتهای شی را در خود نگهداری می‌کند. (قبلا گفته شد که هر شی را در صورتی که در یک مستطیل فرض کنیم یک نقطه شروع و یک نقطه پایان دارد)
ForeColor : رنگ قلم ترسیم شی مورد نظر را تعیین می‌کند.
Height : ارتفاع شی مورد نظر را تعیین می‌کند (این خصوصیت اختلاف عمودی StartPoint.Y و EndPoint.Y را محاسبه می‌کند و در زمان مقدار دهی EndPoint جدیدی ایجاد می‌کند).
Width : عرض شی مورد نظر را تعیین می‌کند (این خصوصیت اختلاف افقیStartPoint.X و EndPoint.X را محاسبه می‌کند و در زمان مقدار دهی EndPoint جدیدی ایجاد می‌کند).
IsFill : این خصوصیت تعیین کننده توپر و یا توخالی بودن شی است.
IsSelected : این خاصیت تعیین می‌کند که آیا شی انتخاب شده است یا خیر (در زمان انتخاب شی چهار مربع کوچک روی شی رسم می‌شود).
Pen : قلم خط ترسیم شی را مشخص می‌کند. (قلم با ضخامت دلخواه)
ShapeType : این خصوصیت نوع شی را مشخص می‌کند (این خاصیت بیشتر برای زمان پیش نمایش ترسیم شی در زمان اجراست البته به نظر خودم اضافه هست اما راه بهتری به ذهنم نرسید)
Size : با استفاده از خصوصیات Height و Width ایجاد شده و تعیین کننده Size شی است.
Thickness : ضخامت خط ترسیمی شی را مشخص می‌کند، این خاصیت در خصوصیت Pen استفاده شده است.
X : مقدار افقی نقطه شروع شی را تعیین می‌کند در واقع StartPoint.X را برمی‌گرداند (این خاصیت اضافی بوده و جهت راحتی کار استفاده شده می‌توان آن را ننوشت).
Y : مقدار عمودی نقطه شروع شی را تعیین می‌کند در واقع StartPoint.Y را برمی‌گرداند (این خاصیت اضافی بوده و جهت راحتی کار استفاده شده می‌توان آن را ننوشت).
Zindex : در زمان ترسیم اشیا ممکن است اشیا روی هم ترسیم شوند، در واقع Zindex تعیین کننده عمق شی روی بوم گرافیکی است.

در پست بعدی به توضیح متدهای این کلاس می‌پردازیم.

نظرات خوانندگان

نویسنده: بتیسا
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۰:۲۶

با سلام

از مطلب مفیدی که تهیه کردید ممنون.
می‌شود از طریق خاصیت Brush که فعلا فقط خواندنی هست، طرح‌های مختلفی برای پس زمینه اشیاع ایجاد کرد. مانند Paint.net و یا MS Paint.

اگر به صورت زیر تعریف کنیم فکر می‌کنم کمی کامل‌تر باشه!

```
private Brush _backgroundBrush;

        /// <summary>
        /// Gets or sets the brush.
        /// </summary>
        /// <value>
        /// The brush.
        /// </value>
        public Brush BackgroundBrush
        {
            get
            {
                return _backgroundBrush;
            }
            private set
            {
                _backgroundBrush = (value != null) ? value : new SolidBrush(BackgroundColor);
            }
        }
        //-------------------------------------[Methode for set brush]---------------------------------
-----
        public virtual void SetBackgroundBrushAsHatch(HatchStyle hatchStyle)
        {
            HatchBrush brush = new HatchBrush(hatchStyle, BackgroundColor);
            BackgroundBrush = brush;
        }

        public virtual void SetBackgroundBrushAsSolid()
        {
            SolidBrush brush = new SolidBrush(BackgroundColor);
            BackgroundBrush = brush;
        }

        public virtual void SetBackgroundBrushAsLinearGradient()
        {
            LinearGradientBrush brush = new LinearGradientBrush(StartPoint, EndPoint, ForeColor,
BackgroundColor);
            BackgroundBrush = brush;
        }
```

که اگر بخواهیم میتونیم باز بیشتر Customize بکنیمشون.

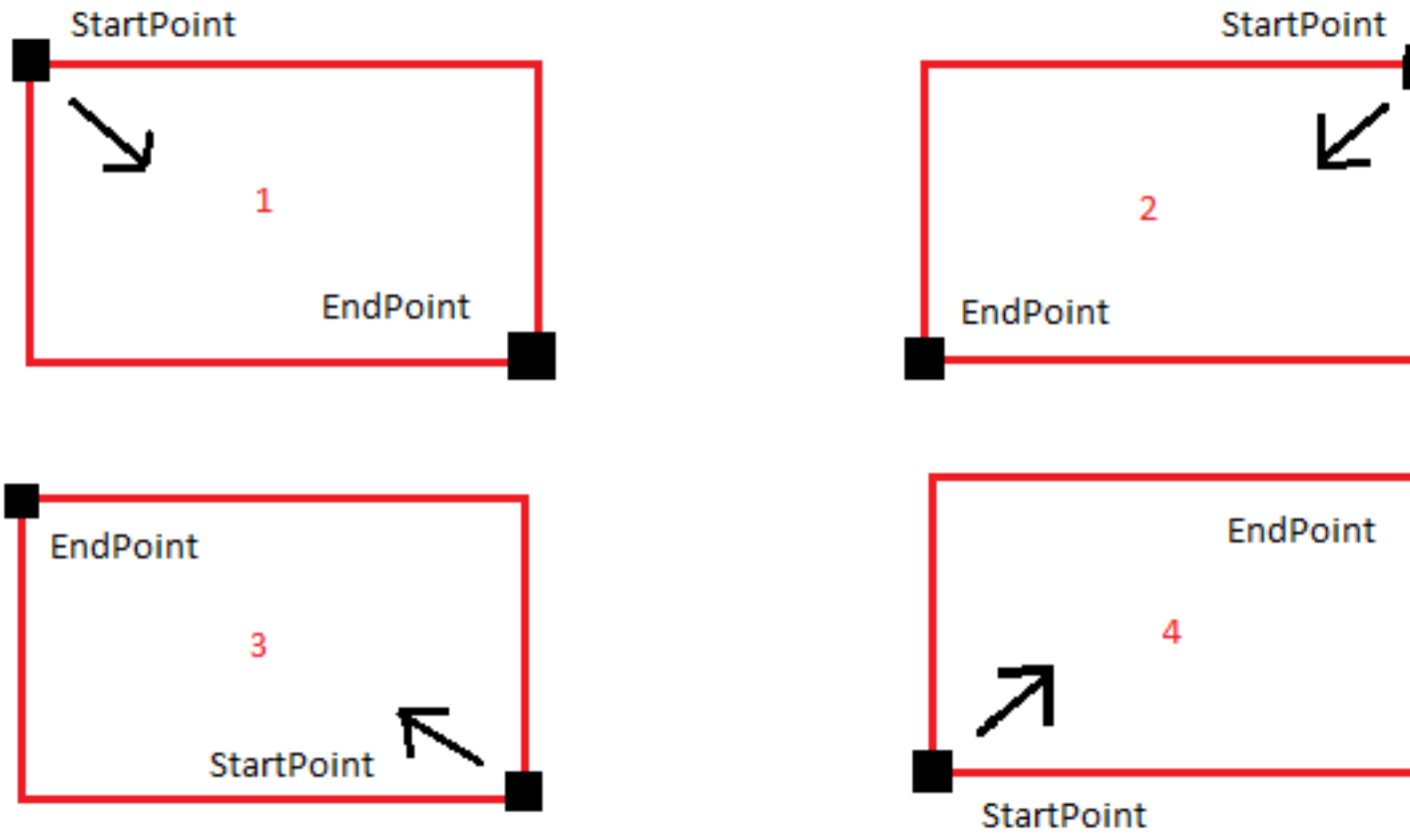
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۰:۴۰

بله کاملا حق با شماست خیلی کارها می‌شه روش انجام داد (قصد آموزش یک مبحث به زبان ساده بود)==> نظر شما اعمال شد.
سعی می‌کنم در زمان ارائه پروژه نهایی همه اینها اعمال بشه

عنوان: **پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #3**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۲:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: OOP, C#, Project, Paint

در ادامه مطالب قبل
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #1
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #2

قبل از شروع توضیحات متدهای کلاس Shape در ادامه پست‌های قبل در ^ و ^ ابتدا به تشریح یک تصویر می‌پردازیم.



خوب همانگونه که در تصویر بالا مشاده می‌نمایید، برای رسم یک شی چهار حالت متفاوت ممکن است پیش بیاید. (دقت کنید که ربع اول محور مختصات روی بوم گرافیکی قرار گرفته است، در واقع گوشه بالا و سمت چپ بوم گرافیکی نقطه (0 و 0) محور مختصات است و عرض بوم گرافیکی محور Xها و ارتفاع بوم گرافیکی محور Yها را نشان می‌دهد)

در این حالت StartPoint.X < EndPoint.X و StartPoint.Y < EndPoint.Y خواهد بود. (StartPoint نقطه ای است که ابتدا ماوس شروع به ترسیم می‌کند، و EndPoint زمانی است که ماوس رها شده و پایان ترسیم را مشخص می‌کند.)
در این حالت StartPoint.X > EndPoint.X و StartPoint.Y > EndPoint.Y خواهد بود.
در این حالت StartPoint.X > EndPoint.X و StartPoint.Y > EndPoint.Y خواهد بود.
در این حالت StartPoint.X < EndPoint.X و StartPoint.Y > EndPoint.Y خواهد بود.

ابتدا یک کلاس کمکی به صورت استاتیک تعریف می‌کنیم که متدی جهت پیش نمایش رسم شی در حالت جابجایی ، رسم، و تغییر اندازه دارد.

```
using System;
using System.Drawing;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Helpers
    /// </summary>
    public static class Helpers
    {
        /// <summary>
        /// Draws the preview.
        /// </summary>
        /// <param name="g">The g.</param>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundBrush">The background brush.</param>
        /// <param name="shapeType">Type of the shape.</param>
        public static void DrawPreview(Graphics g, PointF startPoint, PointF endPoint, Color foreColor,
byte thickness, bool isFill, Brush backgroundBrush, ShapeType shapeType)
        {
            float x = 0, y = 0;
            float width = Math.Abs(endPoint.X - startPoint.X);
            float height = Math.Abs(endPoint.Y - startPoint.Y);
            if (startPoint.X <= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = startPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = endPoint.X;
                y = endPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = endPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X <= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = startPoint.X;
                y = endPoint.Y;
            }

            switch (shapeType)
            {
                case ShapeType.Ellipse:
                    if (isFill)
                        g.FillEllipse(backgroundBrush, x, y, width, height);
                    //else
                    g.DrawEllipse(new Pen(foreColor, thickness), x, y, width, height);
                    break;
                case ShapeType.Rectangle:
                    if (isFill)
                        g.FillRectangle(backgroundBrush, x, y, width, height);
                    //else
                    g.DrawRectangle(new Pen(foreColor, thickness), x, y, width, height);
                    break;
                case ShapeType.Circle:
                    float raduis = Math.Max(width, height);

                    if (isFill)
                        g.FillEllipse(backgroundBrush, x, y, raduis, raduis);
                    //else
                    g.DrawEllipse(new Pen(foreColor, thickness), x, y, raduis, raduis);
                    break;
                case ShapeType.Square:
                    float side = Math.Max(width, height);

                    if (isFill)
                        g.FillRectangle(backgroundBrush, x, y, side, side);
                    //else
                    g.DrawRectangle(new Pen(foreColor, thickness), x, y, side, side);
```

```
                    break;
                case ShapeType.Line:
                    g.DrawLine(new Pen(foreColor, thickness), startPoint, endPoint);
                    break;
                case ShapeType.Diamond:
                    var points = new PointF[4];
                    points[0] = new PointF(x + width / 2, y);
                    points[1] = new PointF(x + width, y + height / 2);
                    points[2] = new PointF(x + width / 2, y + height);
                    points[3] = new PointF(x, y + height / 2);
                    if (isFill)
                        g.FillPolygon(backgroundBrush, points);
                    //else
                    g.DrawPolygon(new Pen(foreColor, thickness), points);
                    break;
                case ShapeType.Triangle:
                    var tPoints = new PointF[3];
                    tPoints[0] = new PointF(x + width / 2, y);
                    tPoints[1] = new PointF(x + width, y + height);
                    tPoints[2] = new PointF(x, y + height);
                    if (isFill)
                        g.FillPolygon(backgroundBrush, tPoints);
                    //else
                    g.DrawPolygon(new Pen(foreColor, thickness), tPoints);
                    break;
            }
            if (shapeType != ShapeType.Line)
            {
                g.DrawString(String.Format("({0},{1})", x, y), new Font(new FontFamily("Tahoma"), 10),
new SolidBrush(foreColor), x - 20, y - 25);
                g.DrawString(String.Format("({0},{1})", x + width, y + height), new Font(new
FontFamily("Tahoma"), 10), new SolidBrush(foreColor), x + width - 20, y + height + 5);
            }
            else
            {
                g.DrawString(String.Format("({0},{1})", startPoint.X, startPoint.Y), new Font(new
FontFamily("Tahoma"), 10), new SolidBrush(foreColor), startPoint.X - 20, startPoint.Y - 25);
                g.DrawString(String.Format("({0},{1})", endPoint.X, endPoint.Y), new Font(new
FontFamily("Tahoma"), 10), new SolidBrush(foreColor), endPoint.X - 20, endPoint.Y + 5);
            }

        }
    }
}
```

متد های این کلاس:

DrawPreview : این متد پیش نمایشی برای شی در زمان ترسیم، جابجایی و تغییر اندازه آماده می‌کند، پارامترهای آن عبارتند از : **بوم گرافیکی** ، **نقطه شروع** ، **نقطه پایان** و **رنگ قلم ترسیم** پیش نمایش شی، **ضخامت خط** ، **آیا شی توپر باشد** ؟، **الگوی پر کردن پس زمینه شی** ، و **نوع شی ترسیمی** می‌باشد.

در ادامه پست‌های قبل ادامه کد کلاس Shape را تشریح می‌کنیم.

```
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Net;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Shape (Base Class)
    /// </summary>
    public abstract partial class Shape
    {
#region Constructors (2)

        /// <summary>
        /// Initializes a new instance of the <see cref="Shape" /> class.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="zIndex">Index of the z.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
```

```
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundColor">Color of the background.</param>
        protected Shape(PointF startPoint, PointF endPoint, int zIndex, Color foreColor, byte
thickness, bool isFill, Color backgroundColor)
        {
            CalulateLocationAndSize(startPoint, endPoint);
            Zindex = zIndex;
            ForeColor = foreColor;
            Thickness = thickness;
            IsFill = isFill;
            BackgroundColor = backgroundColor;
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="Shape" /> class.
        /// </summary>
        protected Shape() { }

#endregion Constructors

#region Methods (10)

// Public Methods (9)

        /// <summary>
        /// Draws the specified g.
        /// </summary>
        /// <param name="g">The g.</param>
        public virtual void Draw(Graphics g)
        {
            if (!IsSelected) return;
            float diff = Thickness + 4;
            Color myColor = Color.DarkSeaGreen;
            g.DrawString(String.Format("({0},{1})", StartPoint.X, StartPoint.Y), new Font(new
FontFamily("Tahoma"), 10), new SolidBrush(myColor), StartPoint.X - 20, StartPoint.Y - 25);
            g.DrawString(String.Format("({0},{1})", EndPoint.X, EndPoint.Y), new Font(new
FontFamily("Tahoma"), 10), new SolidBrush(myColor), EndPoint.X - 20, EndPoint.Y + 5);
            if (ShapeType != ShapeType.Line)
            {
                g.DrawRectangle(new Pen(myColor), X, Y, Width, Height);

                //  1 2 3
                //  8   4
                //  7 6 5
                var point1 = new PointF(StartPoint.X - diff / 2, StartPoint.Y - diff / 2);
                var point2 = new PointF((StartPoint.X - diff / 2 + EndPoint.X) / 2, StartPoint.Y - diff
/ 2);
                var point3 = new PointF(EndPoint.X - diff / 2, StartPoint.Y - diff / 2);
                var point4 = new PointF(EndPoint.X - diff / 2, (EndPoint.Y + StartPoint.Y) / 2 - diff /
2);
                var point5 = new PointF(EndPoint.X - diff / 2, EndPoint.Y - diff / 2);
                var point6 = new PointF((StartPoint.X - diff / 2 + EndPoint.X) / 2, EndPoint.Y - diff /
2);
                var point7 = new PointF(StartPoint.X - diff / 2, EndPoint.Y - diff / 2);
                var point8 = new PointF(StartPoint.X - diff / 2, (EndPoint.Y + StartPoint.Y) / 2 - diff
/ 2);


                g.FillRectangle(new SolidBrush(myColor), point1.X, point1.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point2.X, point2.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point3.X, point3.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point4.X, point4.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point5.X, point5.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point6.X, point6.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point7.X, point7.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point8.X, point8.Y, diff, diff);
            }
            else
            {
                var point1 = new PointF(StartPoint.X - diff / 2, StartPoint.Y - diff / 2);
                var point2 = new PointF(EndPoint.X - diff / 2, EndPoint.Y - diff / 2);
                g.FillRectangle(new SolidBrush(myColor), point1.X, point1.Y, diff, diff);
                g.FillRectangle(new SolidBrush(myColor), point2.X, point2.Y, diff, diff);
            }
        }

        /// <summary>
        /// Points the in sahpe.
        /// </summary>
        /// <param name="point">The point.</param>
        /// <param name="tolerance">The tolerance.</param>
```

```
        /// <returns>
        ///   <c>true</c> if [has point in sahpe] [the specified point]; otherwise, <c>false</c>.
        /// </returns>
        public virtual bool HasPointInSahpe(PointF point, byte tolerance = 5)
        {
            return point.X > (StartPoint.X - tolerance) && point.X < (EndPoint.X + tolerance) &&
point.Y > (StartPoint.Y - tolerance) && point.Y < (EndPoint.Y + tolerance);
        }

        /// <summary>
        /// Moves the specified location.
        /// </summary>
        /// <param name="location">The location.</param>
        /// <returns></returns>
        public virtual PointF Move(Point location)
        {
            StartPoint = new PointF(location.X, location.Y);
            EndPoint = new PointF(location.X + Width, location.Y + Height);
            return StartPoint;
        }

        /// <summary>
        /// Moves the specified dx.
        /// </summary>
        /// <param name="dx">The dx.</param>
        /// <param name="dy">The dy.</param>
        /// <returns></returns>
        public virtual PointF Move(int dx, int dy)
        {
            StartPoint = new PointF(StartPoint.X + dx, StartPoint.Y + dy);
            EndPoint = new PointF(EndPoint.X + dx, EndPoint.Y + dy);
            return StartPoint;
        }

        /// <summary>
        /// Resizes the specified dx.
        /// </summary>
        /// <param name="dx">The dx.</param>
        /// <param name="dy">The dy.</param>
        /// <returns></returns>
        public virtual SizeF Resize(int dx, int dy)
        {
            EndPoint = new PointF(EndPoint.X + dx, EndPoint.Y + dy);
            return new SizeF(Width, Height);
        }

        /// <summary>
        /// Resizes the specified start point.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="currentPoint">The current point.</param>
        public virtual void Resize(PointF startPoint, PointF currentPoint)
        {
            var dx = (int)(currentPoint.X - startPoint.X);
            var dy = (int)(currentPoint.Y - startPoint.Y);
            if (startPoint.X >= X - 5 && startPoint.X <= X + 5)
            {
                StartPoint = new PointF(currentPoint.X, StartPoint.Y);
                if (ShapeType == ShapeType.Circle || ShapeType == ShapeType.Square)
                {
                    Height = Width;
                }
            }
            else if (startPoint.X >= EndPoint.X - 5 && startPoint.X <= EndPoint.X + 5)
            {
                Width += dx;
                if (ShapeType == ShapeType.Circle || ShapeType == ShapeType.Square)
                {
                    Height = Width;
                }
            }
            else if (startPoint.Y >= Y - 5 && startPoint.Y <= Y + 5)
            {
                Y = currentPoint.Y;
                if (ShapeType == ShapeType.Circle || ShapeType == ShapeType.Square)
                {
                    Width = Height;
                }
            }
            else if (startPoint.Y >= EndPoint.Y - 5 && startPoint.Y <= EndPoint.Y + 5)
            {
```

```
                Height += dy;
                if (ShapeType == ShapeType.Circle || ShapeType == ShapeType.Square)
                {
                    Width = Height;
                }
            }
        }

        /// <summary>
        /// Sets the background brush as hatch.
        /// </summary>
        /// <param name="hatchStyle">The hatch style.</param>
        public virtual void SetBackgroundBrushAsHatch(HatchStyle hatchStyle)
        {
            var brush = new HatchBrush(hatchStyle, BackgroundColor);
            BackgroundBrush = brush;
        }

        /// <summary>
        /// Sets the background brush as linear gradient.
        /// </summary>
        public virtual void SetBackgroundBrushAsLinearGradient()
        {
            var brush = new LinearGradientBrush(StartPoint, EndPoint, ForeColor, BackgroundColor);
            BackgroundBrush = brush;
        }

        /// <summary>
        /// Sets the background brush as solid.
        /// </summary>
        public virtual void SetBackgroundBrushAsSolid()
        {
            var brush = new SolidBrush(BackgroundColor);
            BackgroundBrush = brush;
        }
// Private Methods (1)

        /// <summary>
        /// Calulates the size of the location and.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        private void CalulateLocationAndSize(PointF startPoint, PointF endPoint)
        {
            float x = 0, y = 0;
            float width = Math.Abs(endPoint.X - startPoint.X);
            float height = Math.Abs(endPoint.Y - startPoint.Y);
            if (startPoint.X <= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = startPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = endPoint.X;
                y = endPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = endPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X <= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = startPoint.X;
                y = endPoint.Y;
            }
            StartPoint = new PointF(x, y);
            EndPoint = new PointF(X + width, Y + height);
        }

#endregion Methods
    }
}
```

حال به تشریح **سازنده** کلاس می‌پردازیم:

Shape : پارامترهای این سازنده به ترتیب عبارتند از **نقطه شروع** ، **نقطه پایان** ، **عمق شی** ، **رنگ قلم** ، **ضخامت خط** ، **آیا شی توپر باشد** ؟، و **رنگ پر کردن شی** ، در این سازنده ابتدا توسط متدی به نام CalulateLocationAndSize(startPoint, endPoint); b نقاط ابتدا و انتهای شی مورد نظر تنظیم می‌شود، در متد مذکور بررسی می‌شود در صورتی که نقاط شروع و پایان یکی از حالت‌های 1 ، 2، 3، 4 از تصویر ابتدا پست باشد همگی تبدیل به حالت 1 خواهد شد.

سپس به تشریح **متدهای** کلاس Shape می‌پردازیم:
Draw : این متد دارای یک پارامتر ورودی است که **بوم گرافیکی** مورد نظر می‌باشد، در واقع شی مورد نظر خود را بروی این بوم گرافیکی ترسیم می‌کند. در کلاس پایه کار این متد زیاد پیچیده نیست، در صورتی که شی در حالت انتخاب باشد (= IsSelected true) بروی شی مورد نظر 8 مربع کوچک ترسیم می‌شود و اگر شی مورد نظر خط باشد دو مربع کوچک در طرفین خط رسم می‌شود که نشان دهنده انتخاب شدن شی مورد نظر است. این متد به صورت virtual تعریف شده است یعنی کلاس هایی که از Shape ارث میبرند می‌توانند این متد را برای خود از نو بازنویسی کرده (override کنند) و تغییر رفتار دهند.
HasPointInSahpe : این متد نیز به صورت virtual تعریف شده است دارای خروجی بولین می‌باشد. پارامترهای این متد عبارتند از **یک نقطه** و **یک عدد** که نشان دهنده تلرانش نقطه بر حسب پیکسل می‌باشد. کار این متد این است که یک نقطه را گرفته و بررسی می‌کند که آیا نقطه مورد نظر با تلرانس وارد شده آیا در داخل شی واقع شده است یا خیر (مثلا وجود نقطه در مستطیل یا وجود نقطه در دایره فرمول‌های متفاوتی دارند که در اینجا پیش فرض برای تمامی اشیا حالت مستطیل در نظر گرفته شده که می‌توانید آنها را بازنویسی (override) کنید).

Move : این متد به عنوان پارامتر **یک نقطه** را گرفته و شی مورد نظر را به آن نقطه منتقل می‌کند در واقع نقطه شروع و پایان ترسیم شی را تغییر می‌دهد.

Move : این متد نیز برای جابجایی شی به کار می‌رود، این متد دارای پارامترهای **جابجابی در راستای محور Xها** , **جابجایی در راستای محور Yها** ؛ و شی مورد نظر را به آن نقطه منتقل می‌کند در واقع نقطه شروع و پایان ترسیم شی را با توجه به پارامترهای ورودی تغییر می‌دهد.
Resize : این متد نیز برای تغییر اندازه شی به کار می‌رود، این متد دارای پارامترهای **تغییر اندازه در راستای محور Xها** , **تغییر اندازه در راستای محور Yها** می‌باشد و نقطه پایان شی مورد نظر را تغییر می‌دهد اما نقطه شروع تغییری نمی‌کند.

Resize : این متد نیز برای تغییر اندازه شی به کار می‌رود، در زمان تغییر اندازه شی با ماوس ابتدا یک نقطه شروع وجود دارد که ماوس در آن نقطه کلیک شده و شروع به درگ کردن شی جهت تغییر اندازه می‌کند (پارامتر اول این متد نقطه شروع درگ کردن جهت تغییر اندازه را مشخص می‌کند startPoint)، سپس در یک نقطه ای درگ کردن تمام می‌شود در این نقطه باید شی تغییر اندازه پیدا کرده و ترسیم شود (پارامتر دوم این متد نقطه مذکور می‌باشد currentLocation). سپس با توجه با این دو نقطه بررسی می‌شود که تغییر اندازه در کدام جهت صورت گرفته است و اعداد جهت تغییرات نقاط شروع و پایان شی مورد نظر محاسبه می‌شوند. (مثلا تغییر اندازه در مستطیل از ضلع بالا به طرفین، یا از ضلع سمت راست به طرفین و ....). البته برای مربع و دایره باید کاری کنیم که طول و عرض تغییر اندازه یکسان باشد.

CalulateLocationAndSize : این متد که در سازنده کلاس استفاده شده در واقع دو نقطه شروع و پایان را گرفته و با توجه به تصویر ابتدای پست حالت‌های 1 و 2 و3 و 4 را به حالت 1 تبدیل کرده و StartPoint و EndPoint را اصلاح می‌کند.
SetBackgroundBrushAsHatch : این متد یک الگوی Brush گرفته و با توجه به رنگ پس زمینه شی خصوصیت BackgroundBrush را مقداردهی می‌کند.

SetBackgroundBrushAsLinearGradient : این متد با توجه به خصوصیت ForeColor و BackgroundColor یک Gradiant Brush ساخته و آن را به خصوصیت
BackgroundBrush نسبت می‌کند.
SetBackgroundBrushAsSolid : یک الگوی پر کردن توپر برای شی مورد نظر با توجه به خصوصیت BackgroundColor شی ایجاد کرده و آن را به خصوصیت BackgroundBrush شی نسبت می‌دهد.

تذکر : متدهای Move، Resize و HasPointInShape به صورت virtual تعریف شده تا کلاس‌های مشتق شده در صورت نیاز خود کد رفتار مورد نظر خود را override کرده یا از همین رفتار استفاده نمایند.

خوشحال می‌شم در صورتی که در Refactoring کد نوشته شده با من همکاری کنید.

در پست‌های آینده به بررسی و پیاده سازی دیگر کلاس‌ها خواهیم پرداخت.

نظرات خوانندگان

نویسنده: بتیسا
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۰:۴۱

با سلام

از مطلب مفیدتون ممنونم
در متد **DrawPreview** اصلی که نوشته شده در بخش هایی که اشیاء توپر رسم می‌شوند مانند خط 143، 149 و... بجای استفاده از خصوصیت Brush که در بخش قبل برای پس زمینه در نظر گرفته شده بود هر بار یک براش ایجاد شده که می‌توانیم به صورت زیر اصلاح کنیم.

```
case ShapeType.Ellipse:
                    if (isFill)
                        g.FillEllipse(new SolidBrush(backgroundColor), x, y, width, height);
                    //else
                    g.DrawEllipse(new Pen(foreColor, thickness), x, y, width, height);
                    break;

//--------------------------------[Change to]-------=>

               case ShapeType.Ellipse:
                    if (isFill)
                        g.FillEllipse(Brush, x, y, width, height);
                    //else
                    g.DrawEllipse(new Pen(foreColor, thickness), x, y, width, height);
                    break;
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۰:۴۶

بله دوست گلم میشد اینکارو انجام داد
اما با توجه به اینکه متد **DrawPreview** به صورت static تعریف شده نمی‌توان از خصوصیات غیر استاتیک کلاس در آن استفاده کرد.
درسته؟

نویسنده: بتیسا
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۰:۵۸

بله به static بودن متد توجه نکرده بودم

نویسنده: بتیسا
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۱:۱۹

برای برطرف کردن این مسئله هم می‌توانیم همانطور که در ورودی foreColor را دریافت کردیم brush را نیز دریافت کنیم.

```
public static void DrawPreview(Graphics g, PointF startPoint, PointF endPoint, Color foreColor, byte
thickness, bool isFill, Color backgroundColor, ShapeType shapeType)
//--------------------------------[Change to]-------=>
public static void DrawPreview(Graphics g, PointF startPoint, PointF endPoint, Color foreColor, byte
thickness, bool isFill, Brush backgroundBrush , ShapeType shapeType)


//-------------------------------------------------
    case ShapeType.Ellipse:
                    if (isFill)
                        g.FillEllipse(new SolidBrush(backgroundColor), x, y, width, height);
                    //else
```

```
                    g.DrawEllipse(new Pen(foreColor, thickness), x, y, width, height);
                    break;

//--------------------------------[Change to]-------=>

                case ShapeType.Ellipse:
                    if (isFill)
                        g.FillEllipse(backgroundBrush, x, y, width, height);
                    //else
                    g.DrawEllipse(new Pen(foreColor, thickness), x, y, width, height);
                    break;
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۲:۷

درسته اما در قسمت اینترفیس کاربر باید Brushهای مورد نظر ساخته شده و به این متد پاس داده شود، در مراحل پایانی فکر می‌کنم بهتر منظورم بتونم برسونم، به دلایلی (که در پست‌های آینده گفته میشه) از این روش‌ها استفاده نکردم.

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۴:۱۷

چرا drawpreview به صورت استاتیک تعریف شده؟ چرا دوبار تعریف شده؟ و چرا این کلاس پایه اطلاعات زیادی در مورد رسم زیر مجموعه‌های خودش داره؟ آیا بهتر نیست جرئیات ترسیم هر شیء با override شدن به زیر کلاس‌های مشتق شده واگذار بشن؟ چرا این متدها از خاصیت‌های کلاس تعریف شده استفاده نمی‌کنن و دوباره این خاصیت‌ها رو به صورت پارامتر دریافت کردن؟ (همون بحث اعلام استقلال متد تعریف شده به صورت استاتیک و اینکه چرا؟) و اگر این کلاس پایه تا این اندازه لازم هست در مورد رسم دایره و سایر اشکال اطلاعات داشته باشد، چه ضرورتی به abstarct بودن آن هست؟ اصلا چه ضرورتی به تعریف اشیاء مشتق شده از آن هست؟

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۷:۱۱

جز متد **DrawPreview** هیچکدام از متدها پارامتری دریافت نمی‌کنند، اون هم به دلیل اینه که می‌خوام پیش نمایشی از یک شی ترسیم کنم البته می‌تونستیم اون توی یک کلاس جدا بنویسیم که این کلاس زیاد شلوغ نشه فکر می‌کنم با نوشتن کلاس‌های مشتق شده بهتر بشه توی این زمینه‌ها بحث کرد.

پ.ن : متد **DrawPreview** به یک کلاس ثالث منتقل شد.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۱۷:۵۹

نظر شما اعمال شد

نویسنده: مسعود بهرامی
تاریخ: ۱۳۹۲/۰۹/۰۱ ۱۶:۵۸

با اجازه دوست عزیزم مهندس فتح الهی
من به نظرم Helpers رو اگه به شکل زیر Re factor کنیم بهتر باشه :)
اول یه کلاس تعریف می‌کنیم و اطلاعات لازم برای ترسیم پیش نمایش رو تو اون کلاس میزاریم

```
public class ShapeSpecification
    {
```

```
        public PointF StartPoint{get;set;}
        public PointF EndPoint{get;set;}
        public Color ForeColor{get;set;}
        public byte Thickness{get;set;}
        public bool IsFill{get;set;}
        public Brush BackgroundBrush{get;set;}
    }
```

یه کلاس دیگه هم نقاط ابتدا و انتها و طول و عرض رو تو خودش داره

```
public class StartPoints
    {
        public float XPoint { get; set; }
        public float YPoint { get; set; }
        public float Width { get; set; }
        public float Height { get; set; }
    }
```

حالا یه اینترفیس تعریف می‌کنیم که فقط یه متد داره به نام Draw

```
public interface IPeiview
    {
        void Draw(ShapeSpecification shapeScepification);
    }
```

حالا می‌رسیم به کلاس Helpers اصلیمون که میتونه هم استاتیک باشه و هم معمولی به دو شکل زیر

```
public class Helpers
    {
        private readonly IPeiview peiview;

        public Helpers(IPeiview peiview)
        {
            this.peiview = peiview;
        }

        public void Draw(ShapeSpecification shapeSpecification)
        {
            peiview.Draw(shapeSpecification);
        }
    }
```

```
public static  class Helpers
    {

        public static void Draw(ShapeSpecification shapeSpecification, IPeiview peiview)
        {
            peiview.Draw(shapeSpecification);
        }
    }
```

که فقط یه متد ساده Draw داره و اونم تابع Draw اینترفیسی که بش دادیم رو صدا میزینه

یه کلاس دیگه هم تعریف میکنیم که مسئولیتش تشخیص بوم‌های چهارگانه است برای شروع نقطه‌ی ترسیم

```
public static class AreaParser
    {
        public static StartPoints Parse(PointF startPoint, PointF endPoint)
        {
            var startPoints = new StartPoints();

            startPoints.Width = Math.Abs(endPoint.X - startPoint.X);
            startPoints.Height = Math.Abs(endPoint.Y - startPoint.Y);

            if (startPoint.X <= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                startPoints.XPoint = startPoint.X;
                startPoints.YPoint = startPoint.Y;
            }
```

```
            else if (startPoint.X >= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                startPoints.XPoint = endPoint.X;
                startPoints.YPoint = endPoint.Y;
            }

            else if (startPoint.X >= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                startPoints.XPoint = endPoint.X;
                startPoints.YPoint = startPoint.Y;
            }

            else if (startPoint.X <= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                startPoints.XPoint = startPoint.X;
                startPoints.YPoint = endPoint.Y;
            }
            return startPoints;
        }
    }
```

نکته: این کلاس رو اگه با Func ایجاد کنیم خیلی بهتر و تمیزتر وقشنکتر هم میشد که من می‌گزرم فعلا ازش
حالا ما هر شکل جدید که اضافه کنیم به پروژه Paint خودمون و قصد پیش نمایش اونو داشته باشیم فقط کافیه یه کلاس برا پیش نمایشش ایجاد کنیم که کلاس Ipreview رو Implement کنه و متد Draw مخصوص به خودش را داشته باشه و از شر Swithهای طولانی خلاص میشیم مثلا من برای دایره اینکارو کردم

```
public class CirclePreview:IPeiview
    {
        private readonly Graphics graphics;

        public CirclePreview(Graphics graphics)
        {
            this.graphics = graphics;
        }
        public void Draw(ShapeSpecification shapeScepification)
        {
            var startPoints = AreaParser.Parse(shapeScepification.StartPoint,
shapeScepification.EndPoint);

            float raduis = Math.Max(startPoints.Width, startPoints.Height);

            if (shapeScepification.IsFill)
                this.graphics.FillEllipse(shapeScepification.BackgroundBrush, startPoints.XPoint,
startPoints.YPoint, raduis, raduis);
            else
                this.graphics.DrawEllipse(new Pen(shapeScepification.ForeColor,
shapeScepification.Thickness), startPoints.XPoint, startPoints.YPoint, raduis, raduis);
        }
    }
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۰۱ ۲۳:۲۷

ظاهرا همه چیز مرتبه و درست هست.
من بررسی می‌کنم و نتیجش به شما اطلاع میدم
در هر صورت از وقتی که گذاشتین متشکرم

عنوان: **پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #4**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۱/۲۲ ۲۳:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: C#, OOP, Project, Paint

در ادامه پست قبل ، در این پست به بررسی کلاس Triangle جهت رسم مثلث و کلاس Diamond جهت رسم لوزی می‌پردازیم.

```
using System.Drawing;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Triangle
    /// </summary>
    public class Triangle : Shape
    {
        #region Constructors (2)

        /// <summary>
        /// Initializes a new instance of the <see cref="Triangle" /> class.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="zIndex">Index of the z.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundColor">Color of the background.</param>
        public Triangle(PointF startPoint, PointF endPoint, int zIndex, Color foreColor, byte
thickness, bool isFill, Color backgroundColor)
            : base(startPoint, endPoint, zIndex, foreColor, thickness, isFill, backgroundColor)
        {
            ShapeType = ShapeType.Triangle;
        }


        /// <summary>
        /// Initializes a new instance of the <see cref="Triangle" /> class.
        /// </summary>
        public Triangle()
        {
            ShapeType = ShapeType.Triangle;
        }

        #endregion Constructors

        #region Methods (1)

        // Public Methods (1)

        /// <summary>
        /// Draws the specified g.
        /// </summary>
        /// <param name="g">The g.</param>
        public override void Draw(Graphics g)
        {
            var points = new PointF[3];
            points[0] = new PointF(X + Width / 2, Y);
            points[1] = new PointF(X + Width, Y + Height);
            points[2] = new PointF(X, Y + Height);
            if (IsFill)
                g.FillPolygon(BackgroundBrush, points);
            g.DrawPolygon(new Pen(ForeColor, Thickness), points);
            base.Draw(g);
        }

        #endregion Methods
    }
}
```

همانگونه که مشاهده می‌کنید کلاس مثلث از کلاس Shape ارث برده و تشکیل شده از یک سازنده و بازنویسی (override) متد Draw می‌باشد، البته متد HasPointInSahpe در کلاس پایه قاعدتا باید بازنویسی شود، برای تشخیص وجود نقطه در شکل مثلث،

(اگر دوستان فرمولش می‌دونن ممنون می‌شم در اختیار بذارن). در متد Draw سه نقطه مثلث در نظر گرفته شده که بر طبق آن با استفاده از متدهای رسم منحنی اقدام به رسم مثلث توپر یا تو خالی نموده‌ایم.

کلاس لوزی نیز دقیقا مانند کلاس مثلث عمل می‌کند.

```
using System.Drawing;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Diamond
    /// </summary>
    public class Diamond : Shape
    {
        #region Constructors (2)

        /// <summary>
        /// Initializes a new instance of the <see cref="Diamond" /> class.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="zIndex">Index of the z.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundColor">Color of the background.</param>
        public Diamond(PointF startPoint, PointF endPoint, int zIndex, Color foreColor, byte thickness,
bool isFill, Color backgroundColor)
            : base(startPoint, endPoint, zIndex, foreColor, thickness, isFill, backgroundColor)
        {
            ShapeType = ShapeType.Diamond;
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="Diamond" /> class.
        /// </summary>
        public Diamond()
        {
            ShapeType = ShapeType.Diamond;
        }

        #endregion Constructors

        #region Methods (1)

        // Public Methods (1)

        /// <summary>
        /// Draws the specified g.
        /// </summary>
        /// <param name="g">The g.</param>
        public override void Draw(Graphics g)
        {
            var points = new PointF[4];
            points[0] = new PointF(X + Width / 2, Y);
            points[1] = new PointF(X + Width, Y + Height / 2);
            points[2] = new PointF(X + Width / 2, Y + Height);
            points[3] = new PointF(X, Y + Height / 2);
            if (IsFill)
                g.FillPolygon(BackgroundBrush, points);
            g.DrawPolygon(new Pen(ForeColor, Thickness), points);
            base.Draw(g);
        }

        #endregion Methods
    }
}
```

این کلاس نیز از کلاس Shape ارث برده و دارای یک سازنده بوده و متد Draw را ازنو بازنویسی می‌کند، این متد نیز با استفاده از چهار نقطه و استفاده از متد رسم منحنی در دات نت اقدام به طراحی لوزی توپر یا تو خالی می‌کند، متد HasPointInSahpe در کلاس پایه قاعدتا باید بازنویسی شود، برای تشخیص وجود نقطه در شکل لوزی، برای رسم لوزی توپر نیز خصوصیت BackgroundBrush استفاده کرده و شی توپر را رسم می‌کند.

مباحث رسم مستطیل و مربع، دایره و بیضی در پست‌های بعد بررسی خواهند شد.

پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #1
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #2
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #3

موفق وموید باشید.

عنوان: **پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #5**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۰۳ ۰:۵۰
آدرس: www.dotnettips.info
برچسب‌ها: C#, OOP, Project, Paint

در ادامه مطلب پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #4 به تشریح مابقی کلاس‌های برنامه می‌پردازیم.

در این پست به شرح کلاس Rectangle جهت رسم مستطیل و Square جهت رسم مربع می‌پردازیم

```
using System.Drawing;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Rectangle
    /// </summary>
    public class Rectangle : Shape
    {
        #region Constructors (2)

        /// <summary>
        /// Initializes a new instance of the <see cref="Rectangle" /> class.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="zIndex">Index of the z.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundColor">Color of the background.</param>
        public Rectangle(PointF startPoint, PointF endPoint, int zIndex, Color foreColor, byte
thickness, bool isFill, Color backgroundColor)
            : base(startPoint, endPoint, zIndex, foreColor, thickness, isFill, backgroundColor)
        {
            ShapeType = ShapeType.Rectangle;
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="Rectangle" /> class.
        /// </summary>
        public Rectangle()
        {
            ShapeType = ShapeType.Rectangle;
        }

        #endregion Constructors

        #region Methods (1)

        // Public Methods (1)

        /// <summary>
        /// Draws the specified g.
        /// </summary>
        /// <param name="g">The g.</param>
        public override void Draw(Graphics g)
        {
            if (IsFill)
                g.FillRectangle(BackgroundBrush, StartPoint.X, StartPoint.Y, Width, Height);
            g.DrawRectangle(Pen, StartPoint.X, StartPoint.Y, Width, Height);
            base.Draw(g);
        }

        #endregion Methods
    }
}
```

کلاس Rectangle از کلاس پایه طراحی شده در ^ ارث بری دارد. این کلاس ساده بوده و تنها شامل یک سازنده و متد ترسیم شی مستطیل می‌باشد.

کلاس بعدی کلاس Square می‌باشد، که از کلاس بالا (Rectangle) ارث بری داشته است، کدهای این کلاس را در زیر مشاهده می‌کنید.

```
using System;
using System.Drawing;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Square
    /// </summary>
    public class Square : Rectangle
    {
#region Constructors (2)

        /// <summary>
        /// Initializes a new instance of the <see cref="Square" /> class.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="zIndex">Index of the z.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundColor">Color of the background.</param>
        public Square(PointF startPoint, PointF endPoint, int zIndex, Color foreColor, byte thickness,
bool isFill, Color backgroundColor)
        {
            float x = 0, y = 0;
            float width = Math.Abs(endPoint.X - startPoint.X);
            float height = Math.Abs(endPoint.Y - startPoint.Y);
            if (startPoint.X <= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = startPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = endPoint.X;
                y = endPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = endPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X <= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = startPoint.X;
                y = endPoint.Y;
            }
            StartPoint = new PointF(x, y);
            var side = Math.Max(width, height);
            EndPoint = new PointF(x+side, y+side);
            ShapeType = ShapeType.Square;
            Zindex = zIndex;
            ForeColor = foreColor;
            Thickness = thickness;
            BackgroundColor = backgroundColor;
            IsFill = isFill;
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="Square" /> class.
        /// </summary>
        public Square()
        {
            ShapeType = ShapeType.Square;
        }

#endregion Constructors
    }
}
```

این کلاس شامل دو سازنده می‌باشد که سازنده دوم فقط نوع شی را تعیین می‌کند و بقیه کارهای آن مانند مستطیل است، در واقع می‌توان از یک دیدگاه گفت که مربع یک مستطیل است که اندازه طول و عرض آن یکسان است. در سازنده اول (نحوه ترسیم

شکل) ابتدا نقاط ابتدا و انتهای رسم شکل تعیین شده و سپس با توجه به پارامترهای محاسبه شده نوع شی جهت ترسیم و دیگر خصوصیات کلاس مقدار دهی می‌شود، با این تفاوت که در نقطه EndPoint طول و عرض مربع برابر با بزرگترین مقدار طول و عرض وارد شده در سازنده کلاس تعیین شده و مربع شکل می‌گیرد. مابقی متدهای ترسیم و ... طبق کلاس پایه مستطیل و Shape تعیین می‌شود.

مطالب قبل:

پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #1

پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #2

پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #3

پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #4

نظرات خوانندگان

نویسنده: کاوه احمدی
تاریخ: ۱۳۹۱/۱۲/۰۳ ۱۰:۵۸

امروز فرصتی دست داد نگاهی اجمالی به این پروژه بیندازم. به نظرم کد نوشته شده تا به اینجا شی گرا محسوب نمی‌شود. یعنی برخی اهدافی که به واسطه آن پارادایم شی گرایی شکل گرفته در آن رعایت نشده است.
به طور مشخص منظورم متد DrawPreview است که در بخش سوم در کلاس Helpers نوشته شده. تکرار کد شدیدی که در دستور switch این متد دیده می‌شود به سادگی قابل حذف است. کد فوق 2 مشکل اساسی دارد: اول آنکه با زیاد شدن تعداد اشیای قابل رسم، این دستور switch بسیار طولانی شده (با تکرار کد) و کد ناخوانا می‌شود و دوم آنکه با اضافه شدن هر شی قابل رسم جدید به پروژه یک case باید به این دستور اضافه شود. یعنی تغییر در یک بخش از نرم‌افزار منجر به تغییر در سایر بخش‌ها (کلاس Helpers) می‌شود. بدیهی است پارادایم شی‌گرا برای جلوگیری از چنین مسائلی شکل گرفته. در غیر این صورت این کد همان کدهای ساخت‌یافته است که در قالب کلاس نوشته شده. به نظر می‌آید بهتر باشد یک اینترفیس drawable در نظر گرفته می‌شد، در این متد از آن استفاده می‌شد و اشیای قابل رسم آنرا پیاده‌سازی می‌کردند. یک راه بسیار ساده و کارامد

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۳ ۱۱:۲۵

البته میشه این قسمت کلاس پایه رو که if و else و switch زیاد داره، با توجه به مطلب « کمپین ضد IF ! » بهبود بخشید.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۰۳ ۲۳:۴

پاسخ شما کاملا صحیح است، توی همون پست گفتم که خیلی خوشحال میشم دوستان ایده ای به من بدهند که این قسمت با استفاده از Action و Func طراحی کنم، خودم راهی به ذهنم نرسید. بله کاملا شما درست می‌فرمایید، راه حل چیست؟

پ. ن: البته این متد کاملا قابل حذف است و می‌تواند در سیستم استفاده نشود. فقط جهت پیش نمایش رسم اشیا بکار می‌رود.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۲:۵

@کاوه احمدی
من خیلی سعی کردم طبق الگوی « کمپین ضد IF ! » عمل کردم، و پیش رفتم درست شد، اما به دلیل اینکه در زمان رسم شی در برنامه کاربری (اینترفیس) در زمان MouseMove پیش نمایش شی رسم می‌شود و در زمان MouseUp خود شی رسم می‌شود این امکان نداشتم تا از شی نمونه سازی کنم و طبق اون الگو پیش برم ، لطفا در صورتی که روشم اشتباست اصلاح بفرمایین.
موفق ومومید باشید.

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۶ ۱۶:۵۵

سلام

می‌تونید از ابزارهای تزریق وابستگی برای تامین وهله مورد نیاز استفاده کنید.

عنوان: **پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #6**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۲/۰۵ ۰:۵
آدرس: www.dotnettips.info
برچسب‌ها: C#, OOP, Project, Paint

در ادامه پست پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #5 ، در این پست به تشریح کلاس دایره و بیضی می‌پردازیم.

ابتدا به تشریح کلاس ترسیم بیضی (Ellipse) می‌پردازیم.

```
using System.Drawing;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Ellipse Draw
    /// </summary>
    public class Ellipse : Shape
    {
        #region Constructors (2)

        /// <summary>
        /// Initializes a new instance of the <see cref="Ellipse" /> class.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="zIndex">Index of the z.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundColor">Color of the background.</param>
        public Ellipse(PointF startPoint, PointF endPoint, int zIndex, Color foreColor, byte thickness,
bool isFill, Color backgroundColor)
            : base(startPoint, endPoint, zIndex, foreColor, thickness, isFill, backgroundColor)
        {
            ShapeType = ShapeType.Ellipse;
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="Ellipse" /> class.
        /// </summary>
        public Ellipse()
        {
            ShapeType = ShapeType.Ellipse;
        }

        #endregion Constructors

        #region Methods (1)

        // Public Methods (1)

        /// <summary>
        /// Draws the specified g.
        /// </summary>
        /// <param name="g">The g.</param>
        public override void Draw(Graphics g)
        {
            if (IsFill)
                g.FillEllipse(BackgroundBrush, StartPoint.X, StartPoint.Y, Width, Height);
            g.DrawEllipse(Pen, StartPoint.X, StartPoint.Y, Width, Height);
            base.Draw(g);
        }

        #endregion Methods
    }
}
```

این کلاس از شی Shape ارث برده و دارای دو سازنده ساده می‌باشد که نوع شی ترسیمی را مشخص می‌کنند، در متد Draw نیز با توجه به توپر یا توخالی بودن شی ترسیم آن انجام میشود، در این کلاس باید متد **HasPointInShape** بازنویسی (override) شود، در این متد باید تعیین شود که یک نقطه در داخل بیضی قرار گرفته است یا خیر که متاسفانه فرمول بیضی خاطرم نبود. البته به

صورت پیش فرض نقطه با توجه به چهارگوشی که بیضی را احاطه می‌کند سنجیده می‌شود.

کلاس دایره (Circle) از کلاس بالا (Ellipse) ارث بری دارد که کد آن را در زیر مشاهده می‌نمایید.

```
using System;
using System.Drawing;

namespace PWS.ObjectOrientedPaint.Models
{
    /// <summary>
    /// Circle
    /// </summary>
    public class Circle : Ellipse
    {
#region Constructors (2)

        /// <summary>
        /// Initializes a new instance of the <see cref="Circle" /> class.
        /// </summary>
        /// <param name="startPoint">The start point.</param>
        /// <param name="endPoint">The end point.</param>
        /// <param name="zIndex">Index of the z.</param>
        /// <param name="foreColor">Color of the fore.</param>
        /// <param name="thickness">The thickness.</param>
        /// <param name="isFill">if set to <c>true</c> [is fill].</param>
        /// <param name="backgroundColor">Color of the background.</param>
        public Circle(PointF startPoint, PointF endPoint, int zIndex, Color foreColor, byte thickness,
bool isFill, Color backgroundColor)
        {
            float x = 0, y = 0;
            float width = Math.Abs(endPoint.X - startPoint.X);
            float height = Math.Abs(endPoint.Y - startPoint.Y);
            if (startPoint.X <= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = startPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = endPoint.X;
                y = endPoint.Y;
            }
            else if (startPoint.X >= endPoint.X && startPoint.Y <= endPoint.Y)
            {
                x = endPoint.X;
                y = startPoint.Y;
            }
            else if (startPoint.X <= endPoint.X && startPoint.Y >= endPoint.Y)
            {
                x = startPoint.X;
                y = endPoint.Y;
            }
            StartPoint = new PointF(x, y);
            var side = Math.Max(width, height);
            EndPoint = new PointF(x + side, y + side);
            ShapeType = ShapeType.Circle;
            Zindex = zIndex;
            ForeColor = foreColor;
            Thickness = thickness;
            BackgroundColor = backgroundColor;
            IsFill = isFill;
        }

        /// <summary>
        /// Initializes a new instance of the <see cref="Circle" /> class.
        /// </summary>
        public Circle()
        {
            ShapeType = ShapeType.Circle;
        }

#endregion Constructors

#region Methods (1)

// Public Methods (1)

        /// <summary>
        /// Points the in sahpe.
```

```
        /// </summary>
        /// <param name="point">The point.</param>
        /// <param name="tolerance">The tolerance.</param>
        /// <returns>
        ///   <c>true</c> if [has point in sahpe] [the specified point]; otherwise, <c>false</c>.
        /// </returns>
        public override bool HasPointInSahpe(PointF point, byte tolerance = 5)
        {
            float width = Math.Abs(EndPoint.X+tolerance - StartPoint.X-tolerance);
            float height = Math.Abs(EndPoint.Y+tolerance - StartPoint.Y-tolerance);
            float diagonal = Math.Max(height, width);
            float raduis = diagonal / 2;
            float dx = Math.Abs(point.X - (X + Width / 2));
            float dy = Math.Abs(point.Y - (Y + height / 2));
            return (dx + dy <= raduis);
        }

#endregion Methods
    }
}
```

این کلاس شامل دو سازنده می‌باشد، که در سازنده اول با توجه به نقاط ایتدا و انتهای ترسیم شکل مقدار طول و عرض مستطیل احاطه کننده دایره محاسبه شده و باتوجه به آنها بزرگترین ضلع به عنوان قطر دایره در نظر گرفته می‌شود و EndPoint شکل مورد نظر تعیین می‌شود.

در متد **HasPointInShape** با استفاده از فرمول دایره تعیین می‌شود که آیا نقطه پارامتر ورودی متد در داخل دایره واقع شده است یا خیر (جهت انتخاب شکل برای جابجایی یا تغییر اندازه).
در پست‌های بعد به پیاده سازی اینترفیس نرم افزار خواهیم پرداخت.

موفق و موید باشید

در ادامه مطالب قبل:
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #1
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #2
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #3
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #4
پیاده سازی پروژه نقاشی (Paint) به صورت شی گرا #5

نظرات خوانندگان

نویسنده: بتیسا
تاریخ: ۱۳۹۱/۱۲/۰۷ ۹:۸

با سلام برای پیدا کردن نقطه در بیضی من چند لینک پیدا کردم امیدوارم که به کارتون بیاد
لینک اول از ویکی پدیا
لینک دوم از stackoverflow
لینک سوم mathforum
لینک چهارم mathopenref