

# Да се променя цвета на изрисуваните четириъгълници, ако се изберат с мишката

- Да се реализира метод `Contains(Point pt)` на класа `Rectangle`, който връща `true`, ако подадената точка е в четириъгълника;
- Да се реализира метод `List<Rectangle> WhereContains(int x, int y)`, който връща списък с избрани четириъгълници, като се използва метода `Contains`.
- Да се прихване `MouseDown` събитието на формата, за да се прави нов четириъгълник при натискане на десен бутон и да се избират вече съществуващите четириъгълници при натискане на ляв бутон (`e.Button == MouseButton.Left`);
- Избраните и оцветени в червено четириъгълници да се записват в полето `List<Rectangle> _selectedRectangles`, за да може при пускане на левия бутон отново да се изрисуват в синьо.

# Contains()

```
public class Rectangle
{
    public Point Position { get; set; }

    public int Width { get; set; }

    public int Height { get; set; }

    public Color Color { get; set; }

    public bool Contains(Point point)
    {
        return
            Position.X < point.X && (Position.X + Width) > point.X &&
            Position.Y < point.Y && (Position.Y + Height) > point.Y;
    }
}
```

# WhereContains()

```
private List<Rectangle> WhereContains(int x, int y)
{
    List<Rectangle> resultList = new List<Rectangle>();
    foreach (var rectangle in _rectangles)
    {
        if (rectangle.Contains(new Point(x, y)))
        {
            resultList.Add(rectangle);
        }
    }

    return resultList;
}
```

# MouseDown()

```
private void FormMain_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Right) // натиснат на десния бутон
    {
        // създава се и се инициализира нова форма
        Rectangle rectangle = new Rectangle();
        rectangle.Position = e.Location;
        rectangle.Width = 60;
        rectangle.Height = 60;
        rectangle.Color = Color.Blue;

        // добавя се в списъка с форми
        _rectangles.Add(rectangle);

        // получава graphics обект, за да изрисува новата фигура
        // използва using конструкция, тъй като graphics обекта имплементира IDisposable!
        using (var graphics = this.CreateGraphics())
        {
            rectangle.Paint(graphics);
        }
    }
}
```

```
else if (e.Button == System.Windows.Forms.MouseButtons.Left) // натиснат левия бутон
{
    using (var graphics = this.CreateGraphics())
    {
        _selectedRectangles = WhereContains(e.Location.X, e.Location.Y);
        foreach (var selectedRectangle in _selectedRectangles)
        {
            selectedRectangle.Color = Color.Red;
            selectedRectangle.Paint(graphics);
        }
    }
}
```

# MouseUp()

```
private void FormMain_MouseUp(object sender, MouseEventArgs e)
{
    if (_selectedRectangles == null)
    {
        return;
    }

    using (var graphics = this.CreateGraphics())
    {
        foreach (var selectedRectangle in _selectedRectangles)
        {
            selectedRectangle.Color = Color.Blue;
            selectedRectangle.Paint(graphics);
        }
    }

    _selectedRectangles = null;
}
```

# Прихващане на събитие

```
private void FormMain_MouseDown(object sender, MouseEventArgs e)
{
    MessageBox.Show("Mouse down at x:" + e.X + " y:" + e.Y);
}
public FormMain()
{
    InitializeComponent();

    this.MouseDown += this.FormMain_MouseDown;
}
```

# Прихващане на събитие чрез анонимен метод (.NET 2.0)

```
public FormMain()
{
    InitializeComponent();

    this.MouseDown += delegate(object s, MouseEventArgs e)
    {
        MessageBox.Show("Mouse down at x:" + e.X + " y:" + e.Y);
    };
}
```

# Прихващане на събитие чрез анонимен метод (.NET 2.0)

```
private int someVariable;
private void FormMain_MouseDown(object sender, MouseEventArgs e)
{
    MessageBox.Show("Some var:" + someVariable);
}
public FormMain()
{
    InitializeComponent();

    someVariable = 10;

    this.MouseDown += this.FormMain_MouseDown;
}
```

```
public FormMain()
{
    InitializeComponent();

    int someVariable = 10;

    this.MouseDown += delegate(object s, MouseEventArgs e)
    {
        MessageBox.Show("Some var:" + someVariable);
    };
}
```



# WhereContains()

```
else if (e.Button == System.Windows.Forms.MouseButtons.Left)
{
    using (var graphics = this.CreateGraphics())
    {
        _selectedRectangles = RectangleExtension.WhereContains(
            _rectangles,
            e.Location);

        foreach (var selectedRectangle in _selectedRectangles)
        {
            selectedRectangle.Color = Color.Red;
            selectedRectangle.Paint(graphics);
        }
    }
}
```

```
public static class RectangleExtension
{
    public static List<Rectangle> WhereContains(
        List<Rectangle> rectangles,
        Point point)
    {
        List<Rectangle> resultList = new List<Rectangle>();
        foreach (var rectangle in rectangles)
        {
            if (rectangle.Contains(point))
            {
                resultList.Add(rectangle);
            }
        }

        return resultList;
    }
}
```

# Създаване на универсален метод за търсене (.NET 2.0)

```
else if (e.Button == System.Windows.Forms.MouseButtons.Left)
{
    using (var graphics = this.CreateGraphics())
    {
        _selectedRectangles = RectangleExtension.Where(
            _rectangles,
            delegate(Rectangle rectangle)
            {
                return rectangle.Contains(e.Location);
            });

        foreach (var selectedRectangle in _selectedRectangles)
        {
            selectedRectangle.Color = Color.Red;
            selectedRectangle.Paint(graphics);
        }
    }
}
```

```
public delegate Boolean FindRectangle(Rectangle rectangle);
```

```
public static class RectangleExtension
{
```

```
    public static List<Rectangle> Where(
        List<Rectangle> rectangles,
        FindRectangle findRectangleDelegate)
    {
        List<Rectangle> resultList = new List<Rectangle>();
        foreach (var rectangle in rectangles)
        {
            if (findRectangleDelegate(rectangle))
            {
                resultList.Add(rectangle);
            }
        }

        return resultList;
    }
}
```

# Прихващане на събитие чрез ламбда израз (.NET 3.0)

```
public FormMain()
{
    InitializeComponent();

    int someVariable = 10;

    this.MouseDown += delegate(object s, MouseEventArgs e)
    {
        MessageBox.Show("Some var:" + someVariable);
    };
}
```

```
public FormMain()
{
    InitializeComponent();

    int someVariable = 10;

    this.MouseDown += (s, e) => {
        MessageBox.Show("Some var:" + someVariable);
    };
}
```

# Създаване на универсален метод за търсене (.NET 3.0)

```
else if (e.Button == System.Windows.Forms.MouseButtons.Left)
{
    using (var graphics = this.CreateGraphics())
    {
        _selectedRectangles = RectangleExtension.Where(
            _rectangles,
            r => r.Contains(e.Location));
        foreach (var selectedRectangle in _selectedRectangles)
        {
            selectedRectangle.Color = Color.Red;
            selectedRectangle.Paint(graphics);
        }
    }
}
```

```
public delegate Boolean FindRectangle(Rectangle rectangle);
```

```
public static class RectangleExtension
{
```

```
    public static List<Rectangle> Where(
        List<Rectangle> rectangles,
        FindRectangle findRectangleDelegate)
    {
        List<Rectangle> resultList = new List<Rectangle>();
        foreach (var rectangle in rectangles)
        {
            if (findRectangleDelegate(rectangle))
            {
                resultList.Add(rectangle);
            }
        }

        return resultList;
    }
}
```

```
}
```

# Разширяващи методи (Extension methods)

```
public static List<Rectangle> Where(  
    this List<Rectangle> rectangles,  
    FindRectangle findRectangleDelegate)  
{
```

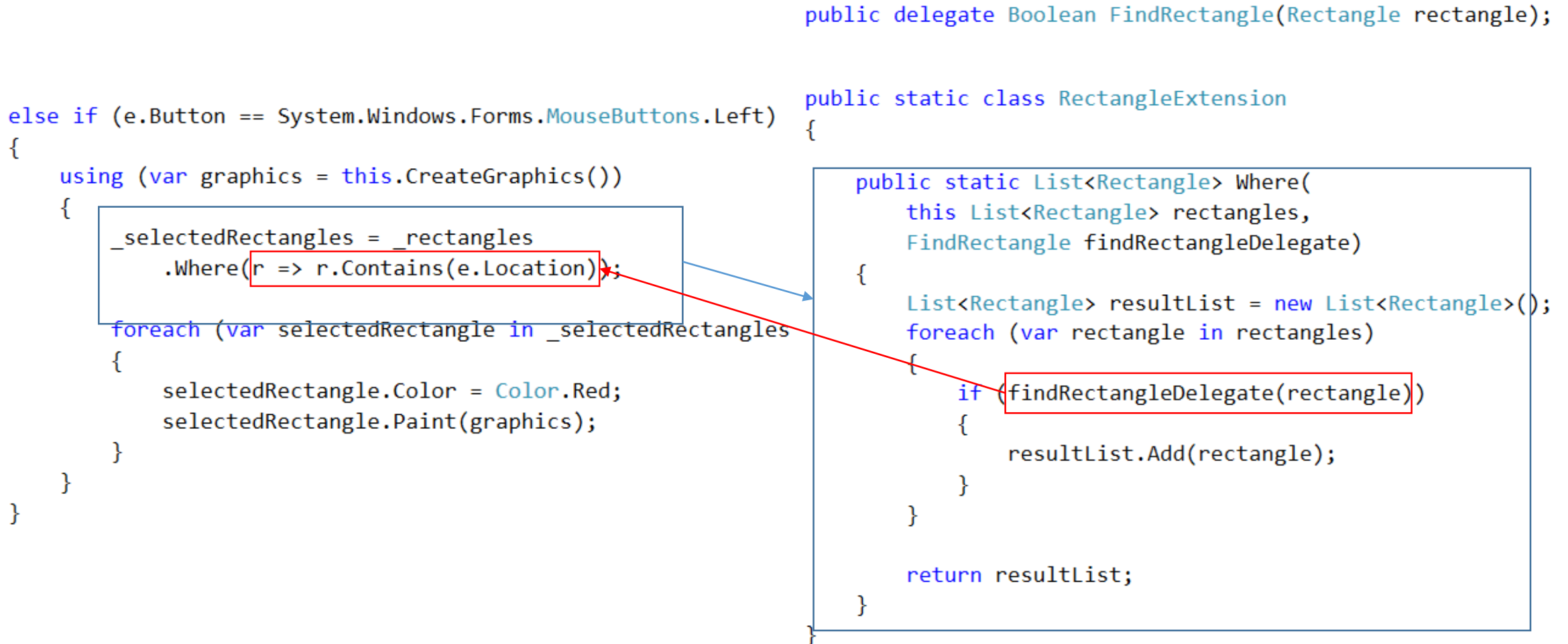
# Разширяващи методи (Extension methods)

```
public delegate Boolean FindRectangle(Rectangle rectangle);

else if (e.Button == System.Windows.Forms.MouseButtons.Left)
{
    using (var graphics = this.CreateGraphics())
    {
        _selectedRectangles = _rectangles
            .Where(r => r.Contains(e.Location));
        foreach (var selectedRectangle in _selectedRectangles)
        {
            selectedRectangle.Color = Color.Red;
            selectedRectangle.Paint(graphics);
        }
    }
}

public static class RectangleExtension
{
    public static List<Rectangle> Where(
        this List<Rectangle> rectangles,
        FindRectangle findRectangleDelegate)
    {
        List<Rectangle> resultList = new List<Rectangle>();
        foreach (var rectangle in rectangles)
        {
            if (findRectangleDelegate(rectangle))
            {
                resultList.Add(rectangle);
            }
        }

        return resultList;
    }
}
```



# Language-Integrated Query (LINQ)

```
else if (e.Button == System.Windows.Forms.MouseButtons.Left)
{
    using (var graphics = this.CreateGraphics())
    {
        _selectedRectangles = _rectangles
            .Where(r => r.Contains(e.Location))
            .ToList();

        foreach (var selectedRectangle in _selectedRectangles)
        {
            selectedRectangle.Color = Color.Red;
            selectedRectangle.Paint(graphics);
        }
    }
}
```

# Да се модифицира условието за избиране на правоъгълници

- При щракване с левия бутон да се изберат всички четириъгълници стоящи отдясно на мястото на което е щракнато;
- Да се използва LINQ филтър.



Да се модифицира условието за избиране  
на правоъгълници

```
_selectedRectangles = _rectangles  
    .Where(r => r.Position.X > e.Location.X)  
    .ToList();
```

# Интерфейса IEnumerable

- Представя колекция от обекти (може да е масив, списък и др.)
- Повечето LINQ методи са разширяващи методи за този интерфейс
- Повечето LINQ методи връщат като резултат обект наследяващ IEnumerable, което позволява каскадно извикване на множество методи

# Сортиране с LINQ

```
string[] names = new string[] { "Ivan", "Dragan", "Zivko", "Petyr", "Dimityr" };  
  
string[] orderedNames = names  
    .OrderBy(o => o)  
    .ToArray();
```

# Филтриране с LINQ

```
string[] names = new string[] { "Ivan", "Dragan", "Zivko", "Petyr", "Dimityr" };  
  
string[] orderedNames = names  
    .Where(c => c.StartsWith("D"))  
    .ToArray();
```

# Филтриране и сортиране - каскадно

```
string[] names = new string[] { "Ivan", "Dragan", "Zivko", "Petyr", "Dimityr" };  
  
string[] orderedNames = names  
    .Where(c => c.StartsWith("D"))  
    .OrderBy(o => o)  
    .ToArray();
```

# Избор на последен елемент

```
string[] names = new string[] { "Ivan", "Dragan", "Zivko", "Petyr", "Dimityr" };  
  
string lastName = names  
    .Where(c => c.StartsWith("D"))  
    .OrderBy(o => o )  
    .LastOrDefault();
```

# Проекция

```
string[] names = new string[] { "Ivan", "Dragan", "Zivko", "Petyr", "Dimityr" };  
  
char lastNameFirstLetter = names  
    .Where(c => c.StartsWith("D"))  
    .OrderBy(o => o)  
    .Select(s => s[0])  
    .LastOrDefault();
```