Протокол №6

Тема:

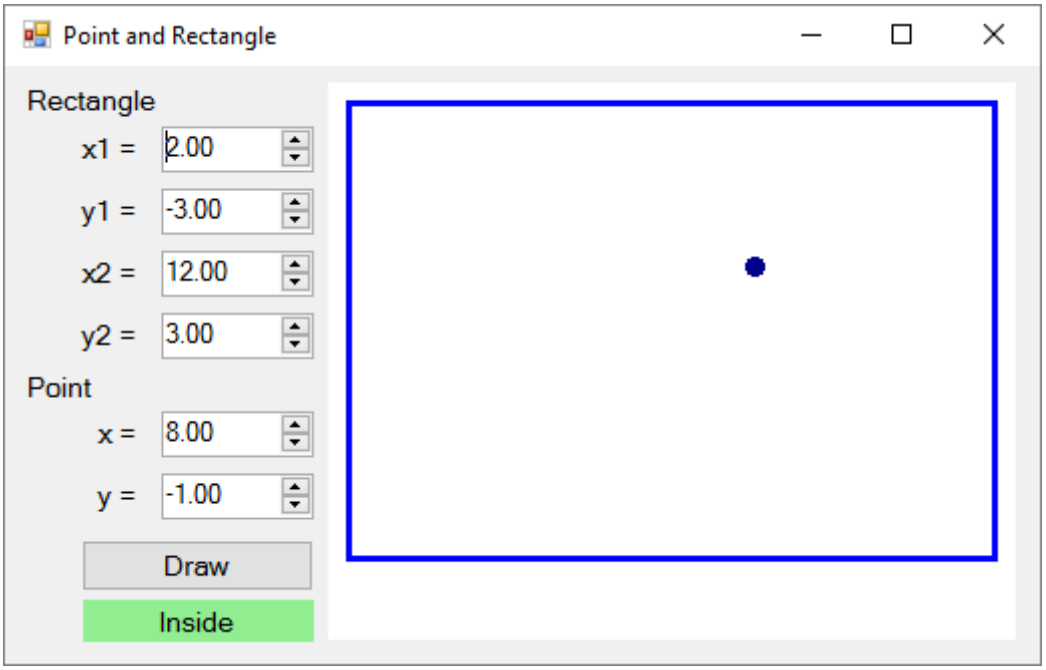
Използване на конструкции за контрол на изпълнението

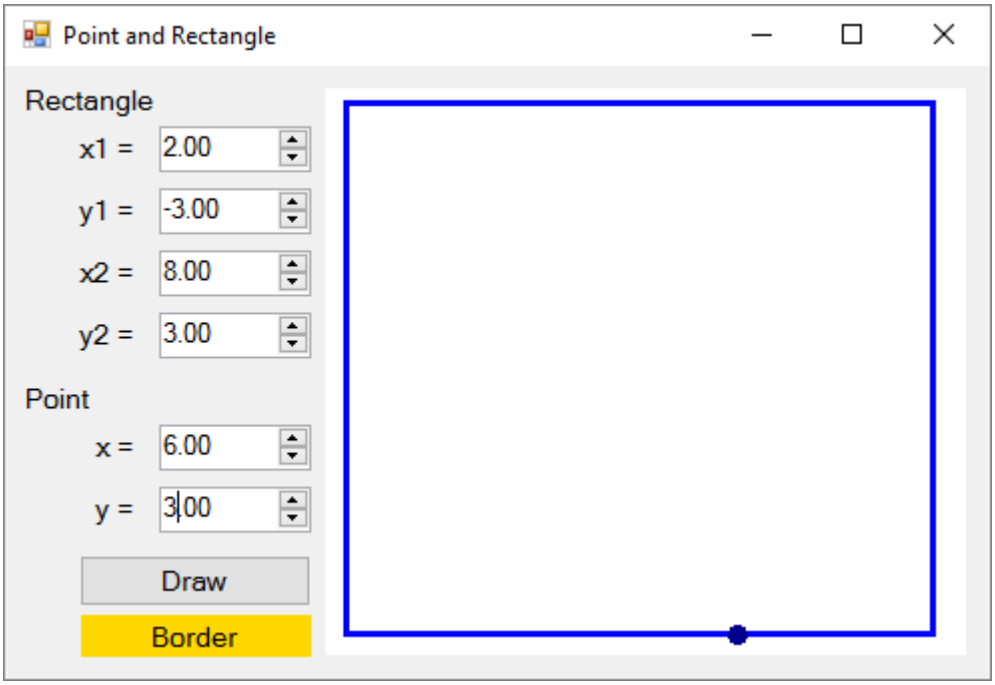
Дата: Изготвил:

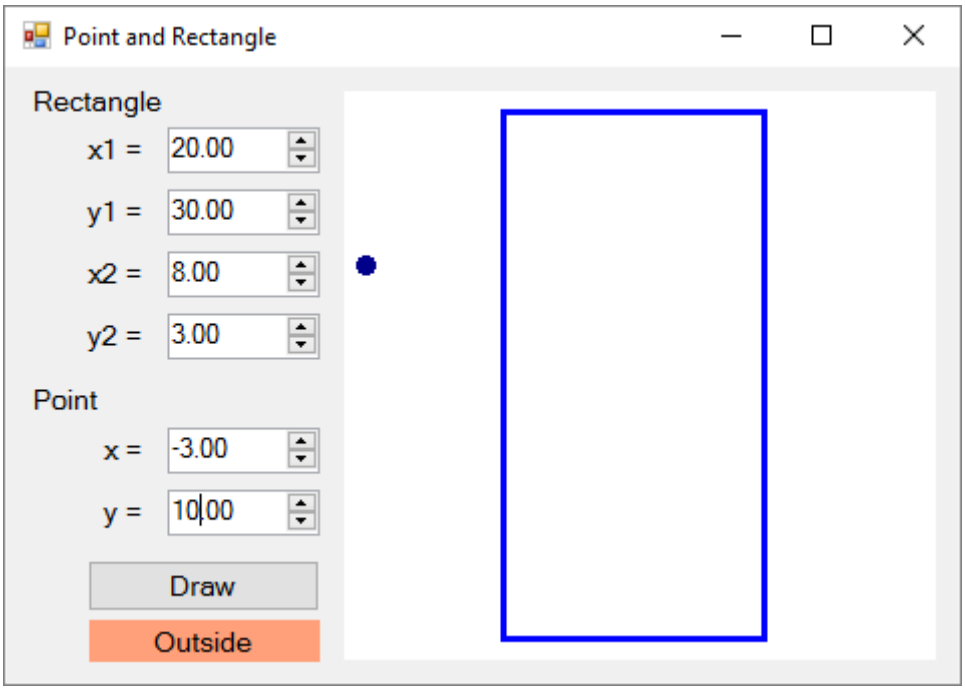
Въпроси и задачи:

Точка и правоъгълник – графично (GUI) приложение

Задачата, която си поставяме е да се разработи графично (**GUI**) приложение за **визуализация на точка и правоъгълник**. Приложението трябва да изглежда приблизително по следния начин:

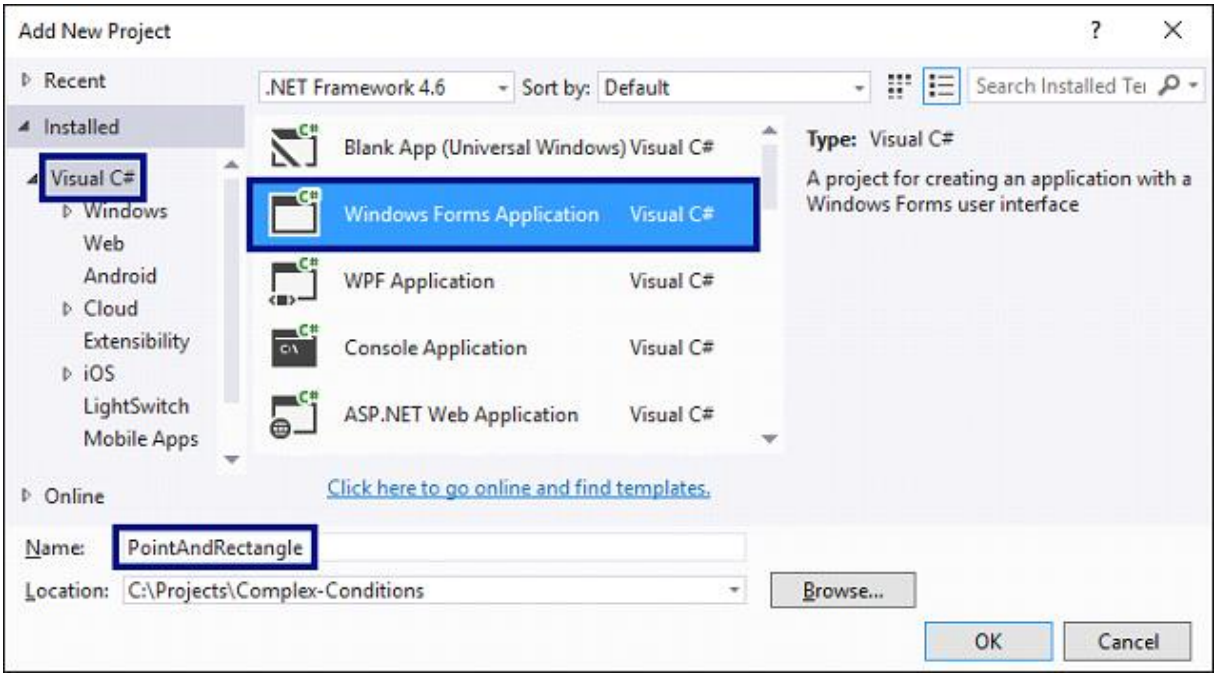






От контролите вляво се задават координатите на **два от ъглите на правоъгълник** (десетични числа) и координатите на **точка**. Приложението **визуализира графично** правоъгълника и точката и изписва дали точката е **вътре** в правоъгълника (**Inside**), **вън** от него (**Outside**) или на някоя от стените му (**Border**). Приложението **премества и мащабира** координатите на правоъгълника и точката, за да бъдат максимално големи, но да се събират в полето за визуализация в дясната страна на приложението.

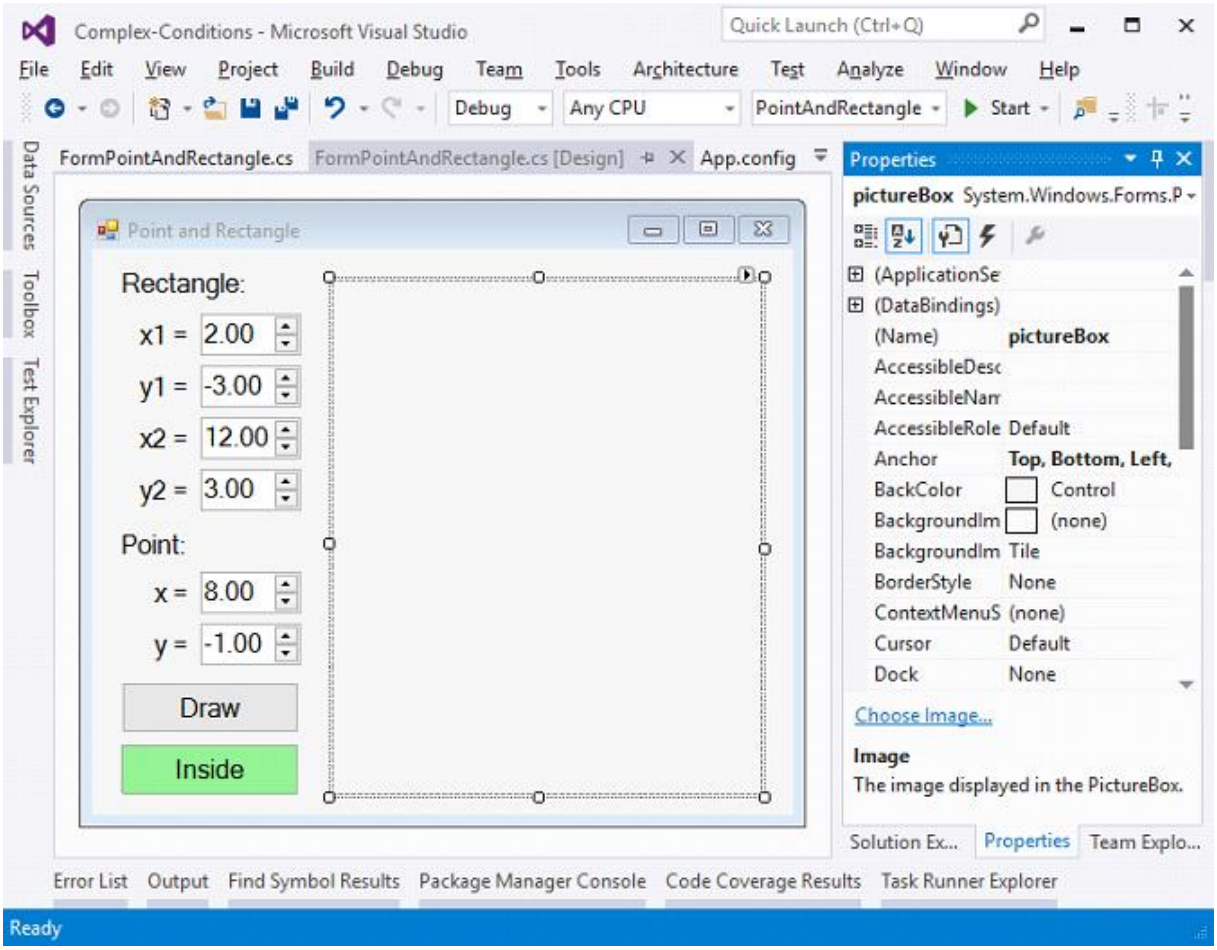
Създаваме нов проект **Windows Forms Application** с подходящо име, например “Point-and-Rectangle”:



Нареждаме контролите във формата, както е показано на фигурата по-долу:

- 6 кутийки за въвеждане на число (`NumericUpDown`).
- Надписи (`Label`) пред всяка кутийка за въвеждане на число.
- Бутон (`Button`) за изчертаване на правоъгълника и точката.
- Текстов блок за резултата (`Label`).

Нагласяме **размерите** и **свойствата** на контролите, за да изглеждат приблизително като на картинката:



Задаваме следните препоръчителни настройки на контролите:

- За **главната форма (`Form`)**, която съдържа всички контроли:
  - (name) = **`FormPointAndRectangle`**
  - **`Text`** = **`Point and Rectangle`**
  - **`Font.Size`** = **`12`**
  - **`Size`** = **`700`**, **`410`**
  - **`MinimumSize`** = **`500`**, **`400`**
  - **`FormBorderStyle`** = **`FixedSingle`**
- За **полетата за въвеждане на число (`NumericUpDown`)**:
  - (name) = **`numericUpDownX1`**; **`numericUpDownY1`**; **`numericUpDownX2`**; **`numericUpDownY2`**; **`numericUpDownX`**; **`numericUpDownY`**
  - **`Value`** = **`2`**; **`-3`**; **`12`**; **`3`**; **`8`**; **`-1`**
  - **`Minimum`** = **`-100000`**
  - **`Maximum`** = **`100000`**
  - **`DecimalPlaces`** = **`2`**
- За **бутона (`Button`)** за **визуализация** на правоъгълника и точката:
  - (name) = **`buttonDraw`**
  - **`Text`** = **`Draw`**
- За **текстовия блок за резултата (`Label`)**:
  - (name) = **`labelLocation`**
  - **`AutoSize`** = **`false`**
  - **`BackColor`** = **`PaleGreen`**

    - `TextAlign` = `MiddleCenter`
- За **полето с чертежа (`PictureBox`)**:
    - (name) = `pictureBox`
    - `Anchor` = `Top`, `Bottom`, `Left`, `Right`

Следва да хванем следните **събития**, за да напишем C# кода, който ще се изпълни при настъпването им:

- Събитието `Click` на бутона `buttonDraw` (извиква се при натискане на бутона).
- Събитието `ValueChanged` на контролите за въвеждане на числа `numericUpDownX1`, `numericUpDownY1`, `numericUpDownX2`, `numericUpDownY2`, `numericUpDownX` и `numericUpDownY` (извиква се при промяна на стойността в контролата за въвеждане на число).
- Събитието `Load` на формата `FormPointAndRectangle` (извиква се при стартиране на приложението, преди да се появи главната форма на екрана).
- Събитието `Resize` на формата `FormPointAndRectangle` (извиква се при промяна на размера на главната формата).

Всички изброени по-горе събития ще изпълняват едно и също действие – `Draw()`, което ще визуализира правоъгълника и точката и ще показва дали тя е вътре, вън или на някоя от страните. Кодът трябва да изглежда така:

```
private void buttonDraw_Click(object sender, EventArgs e)
{
    Draw();
}

private void FormPointAndRectangle_Load(object sender, EventArgs e)
{
    Draw();
}

private void FormPointAndRectangle_Resize(object sender, EventArgs
e)
{
    Draw();
}

private void numericUpDownX1_ValueChanged(object sender, EventArgs
e)
{
    Draw();
}
```

```
 /* TODO: имплементирайте по същия начин манипулатори на събития
    numericUpDownY1_ValueChanged,
    numericUpDownX2_ValueChanged,
    numericUpDownY2_ValueChanged,
    numericUpDownX_ValueChanged and
    numericUpDownY_ValueChanged */

private void Draw()
{
    // TODO: приложете това малко по-късно …
}
```

Нека започнем от по-лесната част: **печат на информация къде е точката спрямо правоъгълника** (Inside, Outside или Border). Кодът трябва да изглежда така:

```
private void Draw()
{
    // Вземете правоъгълника и координатите на точката от формуляра
    var x1 = this.numericUpDownX1.Value;
    var y1 = this.numericUpDownY1.Value;
    var x2 = this.numericUpDownX2.Value;
    var y2 = this.numericUpDownY2.Value;
    var x = this.numericUpDownX.Value;
    var y = this.numericUpDownY.Value;

    // Показва местоположението на точката: Inside / Border /
Outside
    DisplayPointLocation(x1, y1, x2, y2, x, y);
}

private void DisplayPointLocation(
    decimal x1, decimal y1, decimal x2, decimal y2, decimal x,
decimal y)
{
    var left = Math.Min(x1, x2);
    var right = Math.Max(x1, x2);
    var top = Math.Min(y1, y2);
    var bottom = Math.Max(y1, y2);
    if (x > left && x < right && …)
```

```
    {
        this.labelLocation.Text = "Inside";
        this.labelLocation.BackColor = Color.LightGreen;
    }
    else if (… || y < top || y > bottom)
    {
        this.labelLocation.Text = "Outside";
        this.labelLocation.BackColor = Color.LightSalmon;
    }
    else
    {
        this.labelLocation.Text = "Border";
        this.labelLocation.BackColor = Color.Gold;
    }
}
```

Горният код взима координатите на правоъгълника и точките и проверява дали точката е вътре, вън или на страната на правоъгълника. При визуализацията на резултата се сменя и цвета на фона на текстовия блок, който го съдържа.

Помислете как **да допишете** недовършените (нарочно) условия в `if` **проверките**! Кодът по-горе **нарочно не се компилира**, защото целта му е да помислите как и защо работи и да **допишете сами липсващите части**. Остава да се имплементира най-сложната част: визуализация на правоъгълника и точката в контролата `pictureBox` с преоразмеряване. Може да си помогнем с **кода по-долу**, който прави малко изчисления и рисува син правоъгълник и тъмносиньо кръгче (точката) според зададените във формата координати. За съжаление сложността на кода надхвърля изучавания до момента материал и е сложно да се обясни в детайли как точно работи. Оставени са коментари за ориентация. Това е пълната версия на действието `Draw()`:

```
private void Draw()
{
  // Взима правоъгълника и координатите на точката от формуляра
  var x1 = this.numericUpDownX1.Value;
  var y1 = this.numericUpDownY1.Value;
  var x2 = this.numericUpDownX2.Value;
  var y2 = this.numericUpDownY2.Value;
  var x = this.numericUpDownX.Value;
  var y = this.numericUpDownY.Value;

  // Показва местоположението на точка: Inside / Border / Outside
```

```
  DisplayPointLocation(x1, y1, x2, y2, x, y);

  /* Изчислете коефициента на мащаба (съотношението) за диаграмата, съдържаща правоъгълник и точка, за да се поберат добре в кутията на картината*/
  var minX = Min(x1, x2, x);
  var maxX = Max(x1, x2, x);
  var minY = Min(y1, y2, y);
  var maxY = Max(y1, y2, y);
  var diagramWidth = maxX - minX;
  var diagramHeight = maxY - minY;
  var ratio = 1.0m;
  var offset = 10;
  if (diagramWidth != 0 && diagramHeight != 0)
  {
    var ratioX = (pictureBox.Width - 2 * offset - 1) / diagramWidth;
    var ratioY = (pictureBox.Height - 2 * offset - 1) / diagramHeight;
    ratio = Math.Min(ratioX, ratioY);
  }

  // Изчислява мащабираните координати на правоъгълника
  var rectLeft = offset + (int)Math.Round((Math.Min(x1, x2) - minX) * ratio);
  var rectTop = offset + (int)Math.Round((Math.Min(y1, y2) - minY) * ratio);
  var rectWidth = (int)Math.Round(Math.Abs(x2 - x1) * ratio);
  var rectHeight = (int)Math.Round(Math.Abs(y2 - y1) * ratio);
  var rect = new Rectangle(rectLeft, rectTop, rectWidth, rectHeight);

  // Изчислява координатите на изчислената точка
  var pointX = (int)Math.Round(offset + (x - minX) * ratio);
  var pointY = (int)Math.Round(offset + (y - minY) * ratio);
  var pointRect = new Rectangle(pointX - 2, pointY - 2, 5, 5);

  // Начертава правоъгълника и точка
```

```
  pictureBox.Image = new Bitmap(pictureBox.Width,
pictureBox.Height);
  using (var g = Graphics.FromImage(pictureBox.Image))
  {
    // Начертава фона (white area)
    g.Clear(Color.White);

    // Начертава правоъгълника (scalled to the picture box size)
    var pen = new Pen(Color.Blue, 3);
    g.DrawRectangle(pen, rect);

    // Начертава точката (scalled to the picture box size)
    pen = new Pen(Color.DarkBlue, 5);
    g.DrawEllipse(pen, pointRect);
  }
}

private decimal Min(decimal val1, decimal val2, decimal val3)
{
  return Math.Min(val1, Math.Min(val2, val3));
}

private decimal Max(decimal val1, decimal val2, decimal val3)
{
  return Math.Max(val1, Math.Max(val2, val3));
}
```

В горния код се срещат доста **преобразувания на типове**, защото се работи с различни типове числа (десетични числа, реални числа и цели числа) и понякога се изисква да се преминава между тях.

Накрая **компилираме кода**. Ако има грешки, ги отстраняваме. Най-вероятната **причина** за грешка е **несъответстващо име на някоя от контролите** или ако **сте написали кода на неправилно място**.

Стартираме приложението и го **тестваме**. Въвеждаме различни данни, за да видим дали се държи коректно.