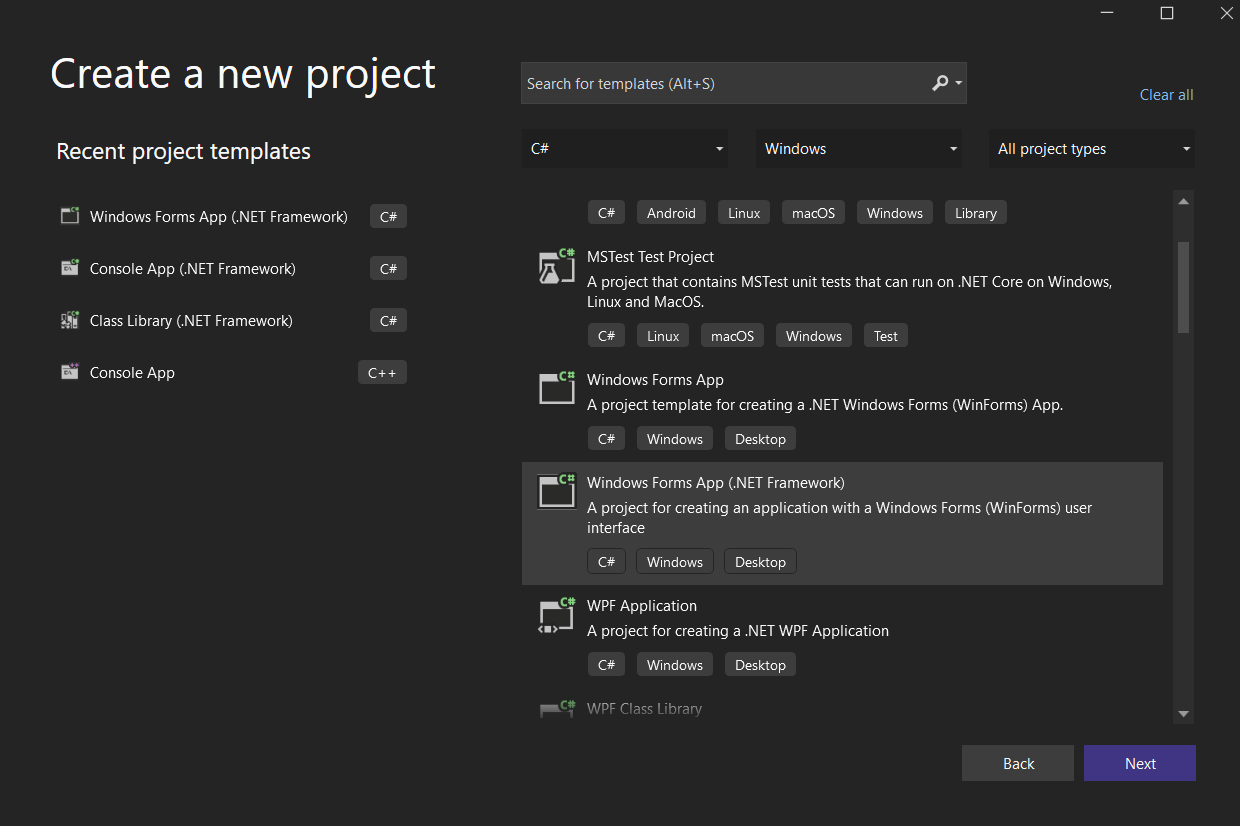
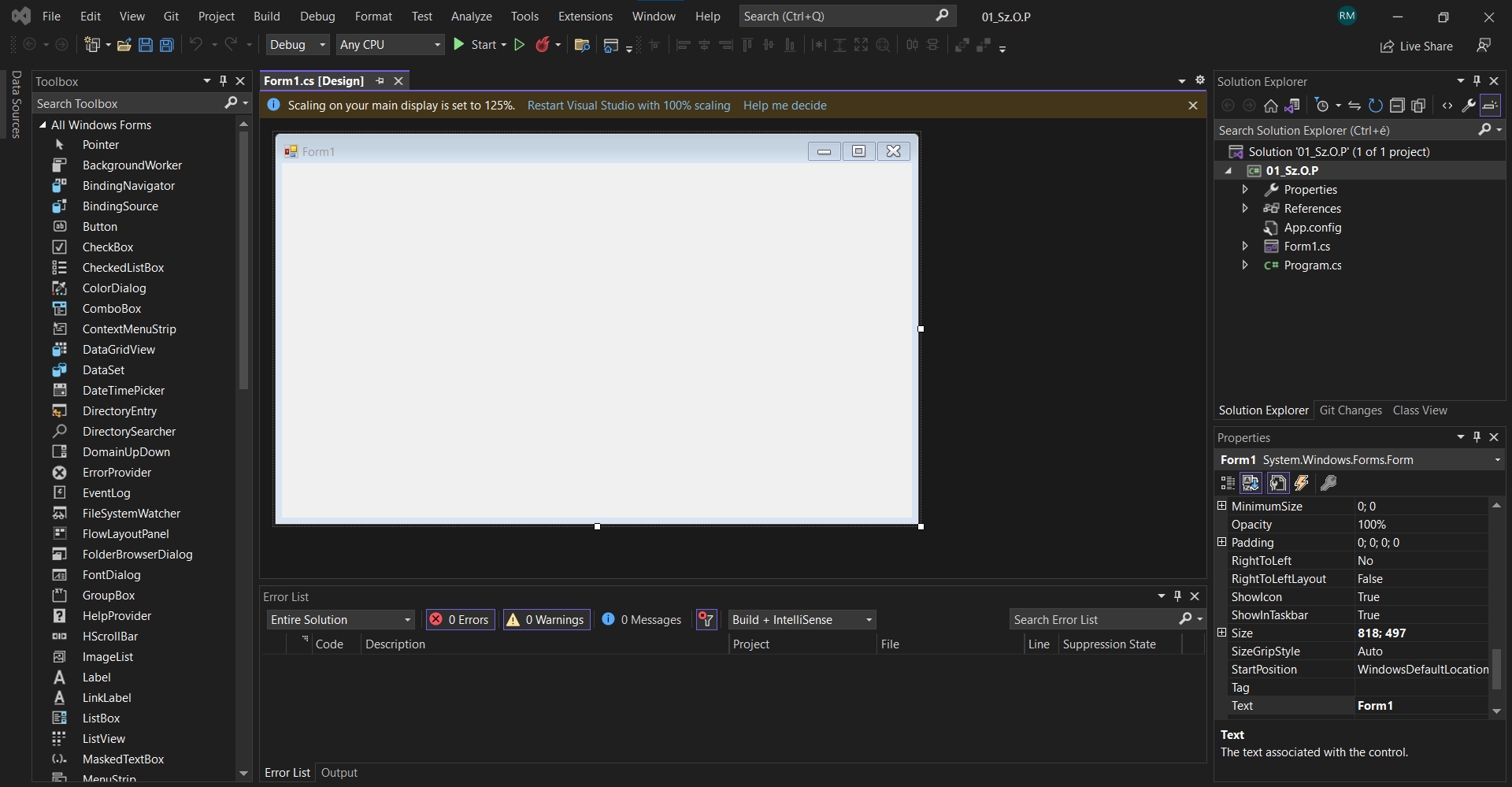
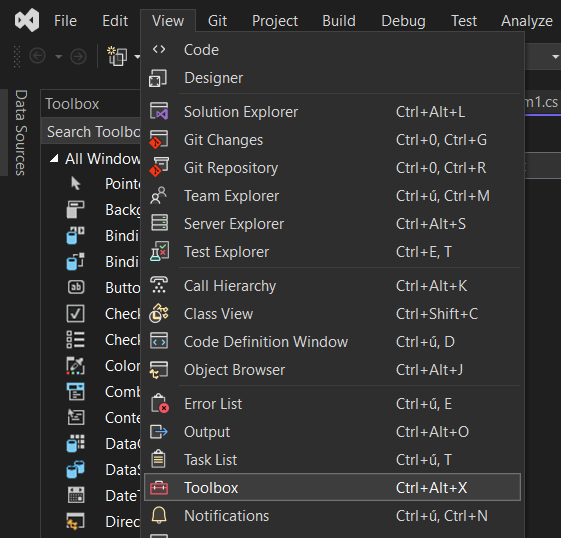
# Windows form alkalmazás készítése

Új projekt készítésekor ügyeljünk, hogy .NET Framework-ös legyen a project.



Névadás után megjelenik előttünk ez:  


Ha a Toolbox menü nem jelenne meg, az a következő helyen érhető el:

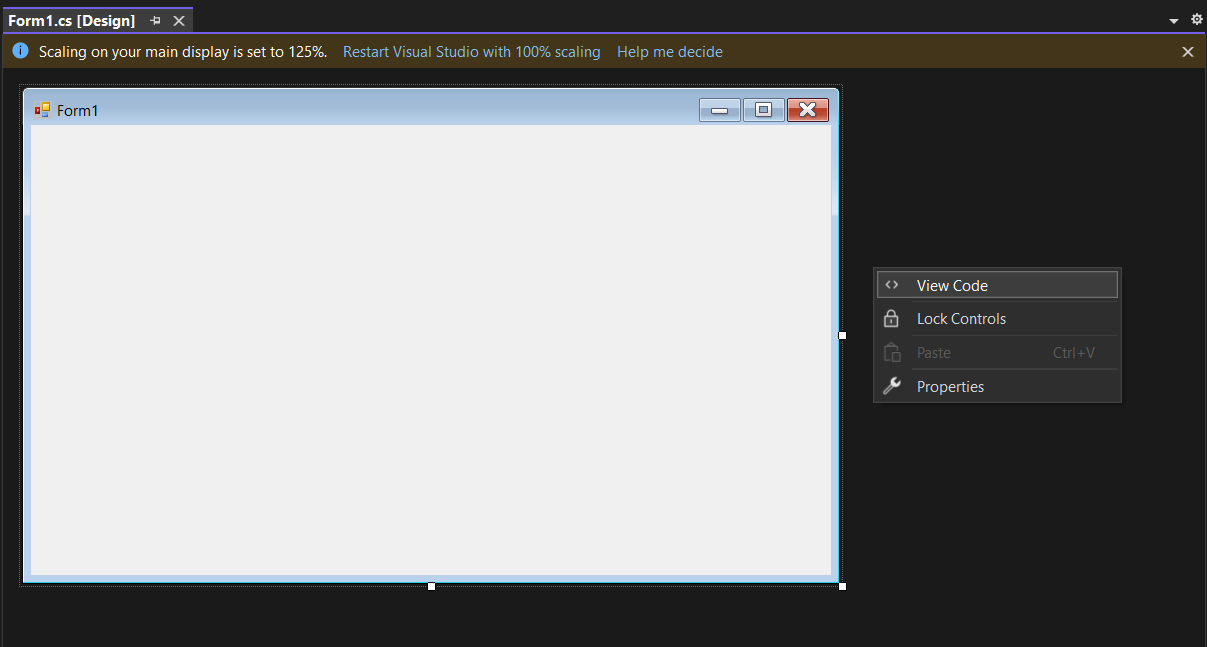


A Toolbox menü segítségével rakhatunk a Form-unkra különböző irányításokat, pl.: gombot, képdobozt, stb…

Ha szeretnénk Formunkra valamelyik irányítást rárakni, vagy dupla kattintással, vagy odahúzással megtehetjük ezt.

A Toolbox-on kívül magában a programkódban is elhelyezhetünk ilyeneket.

Ha a felső fülek között csak a Design fül látható, akkor ezen ablakban jobb klikkel elérhető menüponton belül, a View code-ra rámenve szerkeszthetjük a programunk kódját.



Irányítást akár magában a kódban is hozzáadhatunk:

public Form1() // A form konstruktora

{

InitializeComponent();

Button btn = new Button();

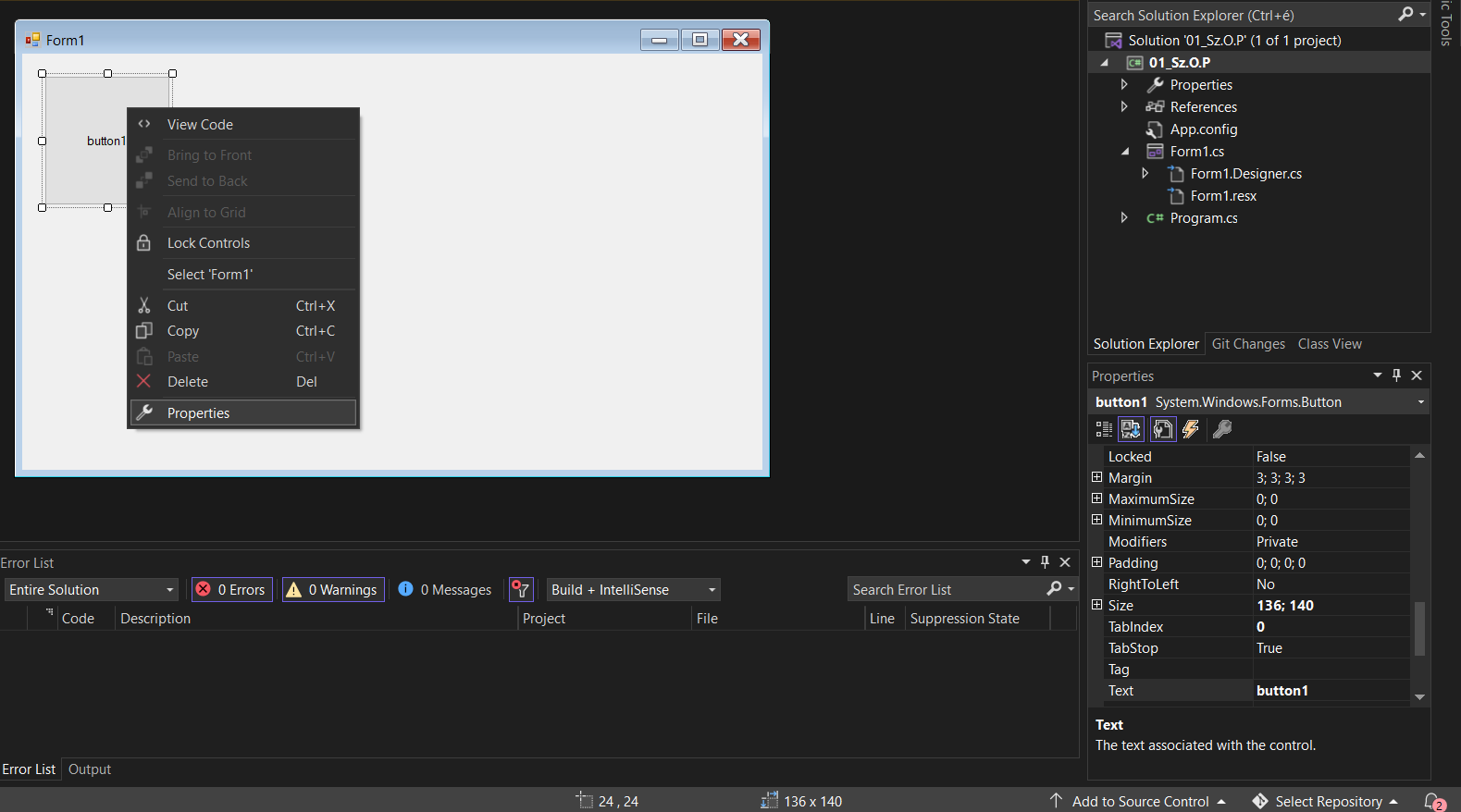
btn.Size = new Size(100, 100);

btn.Location = new Point(10, 10);

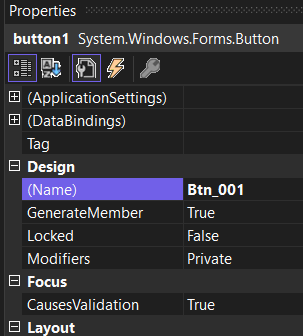
this.Controls.Add(btn);

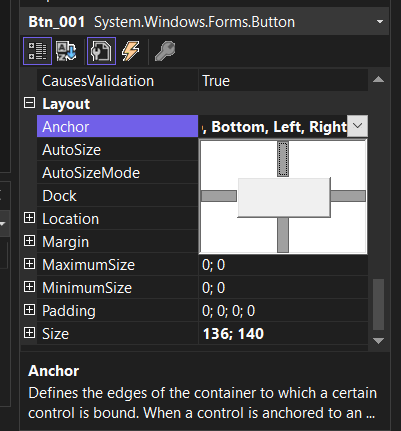
}

Az ilyen objektumoknak vannak külön-külön tulajdonságai, ezt ha még nem látjuk az IDE felületén (általában bal alsó sarok), akkor jobb klikk az objektumra, és ott a Properties menüpont.

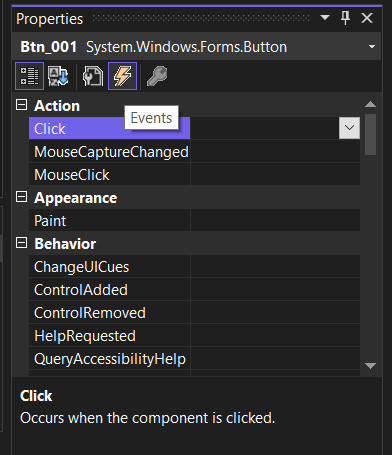


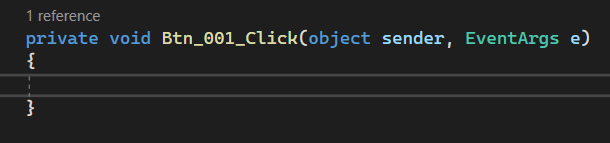
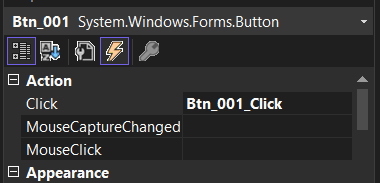
Általában az objektum nevét érdemes megváltoztatni, hogy jobban mutassuk, hogy mit is csinálna. (pl.: jobb csak annyit beírni, hogy LoginBtn, mint hogy Button1 és Button25 között megkeressük a nekünk valót)

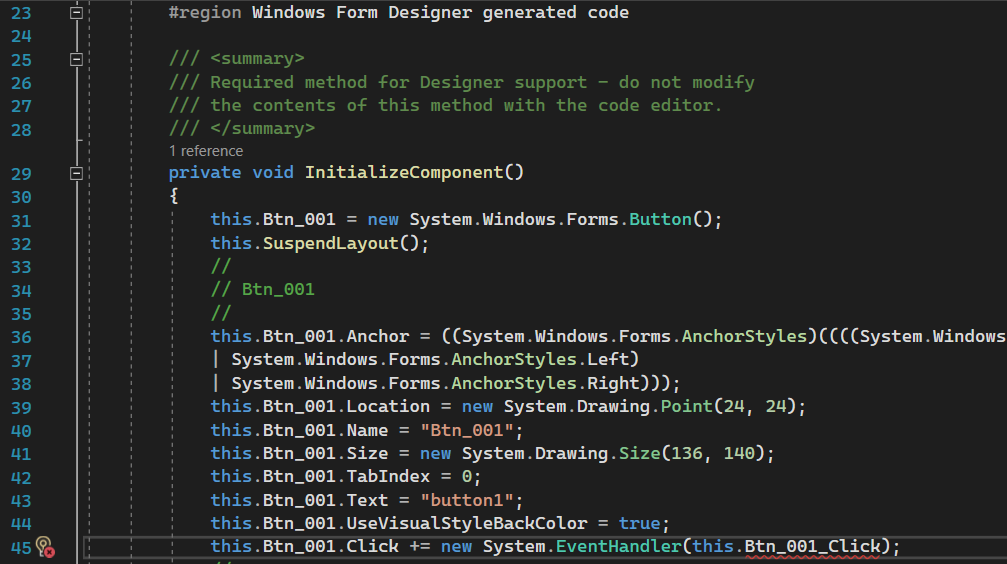
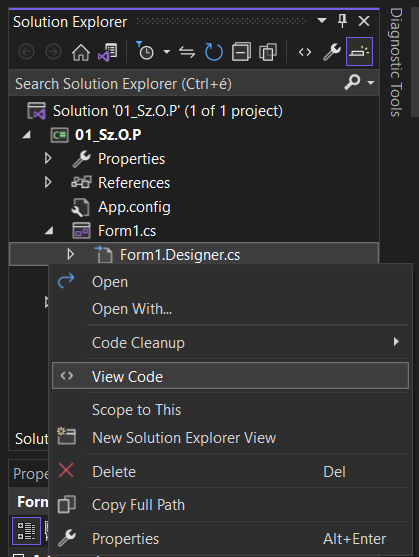


Ha az ablak méretével változó Control-t szeretnénk létrehozni, ahhoz az Anchor tulajdonságát kell megváltoztatni:

Mindegyik Control-hoz tartozik sok esemény (Event), melyen ugyanabban a menüben lehet elérni, mint a tulajdonságait.



Ha egy Control-ra duplán kattintunk, akkor a leggyakrabban használt eseményhez hozza létre a metódust. Ezen felül az event fülben is duplán kell kattintani a megfelelő lenyíló mezőbe, hogy létrehozzon újat. Ha létezne az event-hez megfelelő paraméterű metódus, akkor ott a lenyíló menüben is ki lehet választani.

Ha ezt a metódust netalántán nem a Properties menüben törölnénk, hanem a kódnézetben, akkor az IDE hibát fog jelezni. Ezt orvosolni a következőképp kell: (megfelelő sor törlése, ez jelen esetben a 45.)

Form kezelésről jelenleg ennyi.

# Szálkezelés

Szálkezeléshez kell használni a System.Threading névteret.



// Szál leállítása

t.Abort();

// Szál "visszafűzése"

t.Join();

// a megadott ideig (ms) vár

Thread.Sleep(2500);

// új thread objektum létrehozása

// paramétere metódusreferenciát vár

Thread t = new Thread(DrawRandomLine);

// és elindítása

t.Start();

Jelenleg a DrawRandomLine egy paraméter nélküli metódus. Ha szeretnénk paraméteres metódust egy szál konstruktorának adni, akkor az csak egyparaméteres lehet, és az is object típusú. Hogy ne így kelljen ezt megadni, használhatunk lambda kifejezést is.

private void Canvas\_MouseDown(object sender, MouseEventArgs e)

{

Thread t = new Thread(() => DrawLine(e.X, e.Y));

t.Start();

}

private void DrawLine(int x, int y)

{

// A Canvas közepétől rajzolja a vonalat

g.DrawLine(Pens.Black, Canvas.Width / 2, Canvas.Height / 2, x, y);

// a megadott ideig (ms) vár

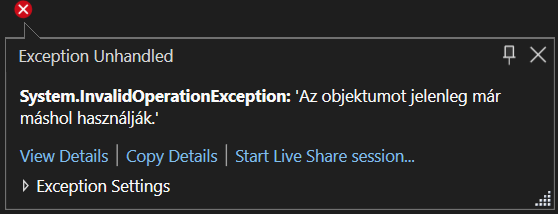
Thread.Sleep(2500);

// "Törli" a vonalat

g.DrawLine(Pens.White, Canvas.Width / 2, Canvas.Height / 2, x, y);

}

Ha sokat kattintunk a vászonra, akkor egy idő után kivételt kapunk:



Ezt lock-olással oldhatjuk meg.

Szálkezelésről jelenleg ennyi

// helytelen lock-olás, a metódus teljes futása alatt ellehetetlenítjük

// a megfelelő erőforráshoz a hozzáférést

private void DrawLine(int x, int y)

{

lock (g)

{

// A Canvas közepétől rajzolja a vonalat

g.DrawLine(Pens.Black, Canvas.Width / 2, Canvas.Height / 2, x, y);

// a megadott ideig (ms) vár

Thread.Sleep(2500);

// "Törli" a vonalat

g.DrawLine(Pens.White, Canvas.Width / 2, Canvas.Height / 2, x, y);

}

}

// Helyes lock-olás, csak a kritikus szakaszban foglaljuk le

// a kívánt erőforrást

private void DrawLine(int x, int y)

{

lock (g) // A Canvas közepétől rajzolja a vonalat

{

g.DrawLine(Pens.Black, Canvas.Width / 2, Canvas.Height / 2, x, y);

}

// a megadott ideig (ms) vár

Thread.Sleep(2500);

lock (g) // "Törli" a vonalat

{

g.DrawLine(Pens.White, Canvas.Width / 2, Canvas.Height / 2, x, y);

}

}