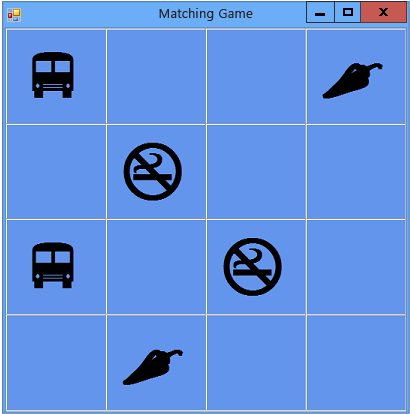
**Memóriajáték készítése C#-ban**

Ebből oktatóanyagból megtanulhatod, hogy hogyan készíthetsz egy memóriajátékot, ahol a játékosnak párosítania kell az elrejtett képeket.

Az alábbi lépéseken fog haladni az útmutató:

* Objektumok tárolása, ikonok a [Lista <T>](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1) objektumban.
* foreach ciklus használata C# -ban, hogy a lista elemein végig haladjon.
* Ablak állapotának követése referenciaváltozók segítségével.
* Eseménykezelő készítése, amely eseményekre reagál, ami több objektummal is használható
* Időzítő létrehozása, amely visszaszámol, majd a kezdés után egyszer elindít egy eseményt.

Amikor elkészülsz, alkalmazásnak a következő képhez hasonlónak kell kinéznie:

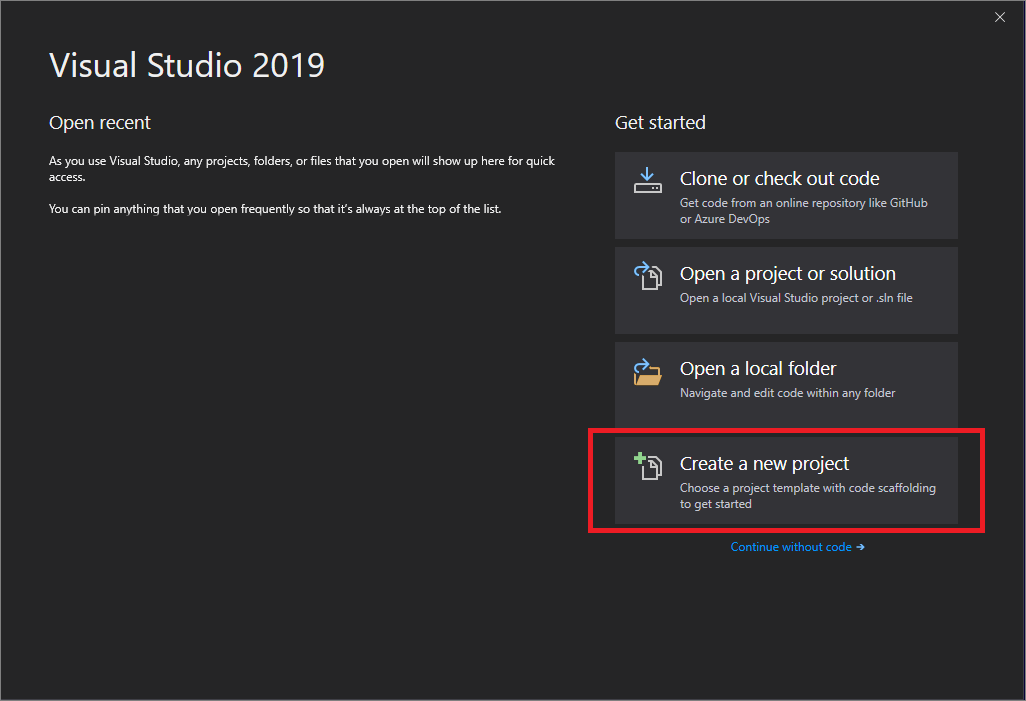


# 1. lépés: Projekt készítése, majd táblázat hozzáadása az űrlaphoz

A játék létrehozásának első lépése a projekt létrehozása és egy táblázat hozzáadása az űrlaphoz. A táblázat segít az ikonok igazításában az elkészülő 4x4-es rácshoz. Jó pár tulajdonságot is be kell beállítani megjelenés javítása érdekében.

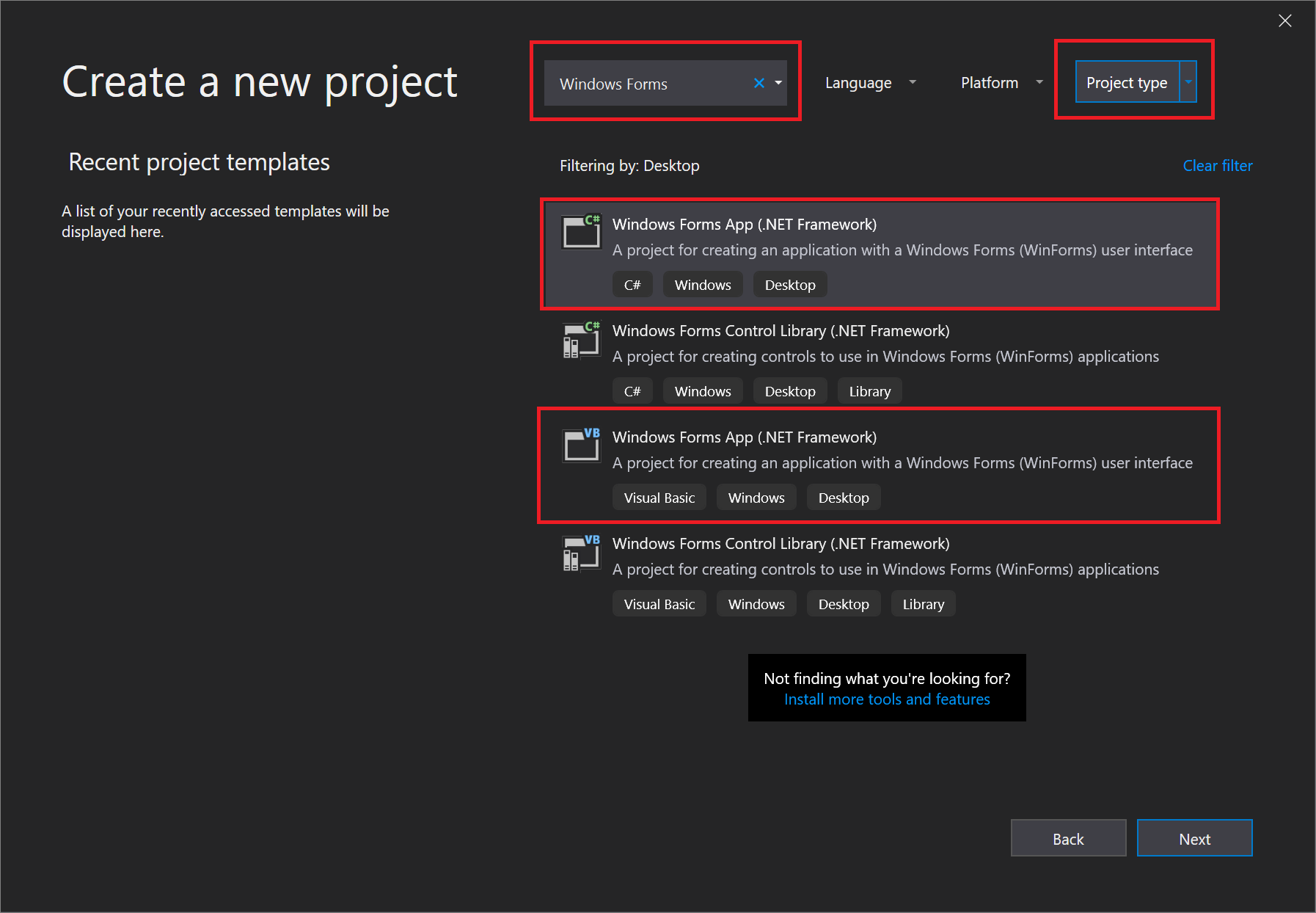
## Projekt létrehozása és táblázat hozzáadása az űrlaphoz

1. A kezdőablakban válaszd a **Create a new Project (Új projekt létrehozása) lehetőséget**.



1. A **Create a new Project** ablakban írd be a Windows Forms kulcsszót a keresőmezőbe. Ezután válaszd a **Desktop (Asztali alkalmazás)** elemet a **Project type (Projekt típus)** listából.

A **Projekt típus** szűrő alkalmazása után válaszd a **Windows Forms App (.NET Framework)** sablont a **C#** számára, majd válaszd a **Next (Tovább) lehetőséget**.



**Megjegyzés**

Ha nem látod a **Windows Forms App (.NET Framework)** sablont, telepíthetd az **Create a new Project** ablakból. A **Not finding what you’re looking for (Nem találja, amit keres?)**üzenet alatt válaszd az **Install more tools and features (További eszközök és szolgáltatások telepítése)** linket.

Az „Új eszközök és szolgáltatások telepítése” link az „Új projekt létrehozása” ablak „Nem található meg a keresett” üzenetéből

Ezután a Visual Studio Installer alkalmazásban válaszd a Válassza a**. NET desktop development (.NET asztali fejlesztés)** modult.



Válaszd ki a Visual Studio Installer **Modify (Módosítás)** gombját. Előugorhat egy kérdés a munka mentésére; ha megjelenik, akkor mentsd el a projektet. Majd válaszd a **Continue (Folytatás)** lehetőséget a modul telepítéséhez.

1. A **Configure your new project (Új projekt konfigurálása**) ablakban írd be a Memóriajáték címet a **Project name (Projekt neve)** mezőbe. Ezután válaszd a **Create (Létrehozás) lehetőséget**.

## Az űrlap tulajdonságainak beállítása

1. A **Properties (Tulajdonságok)** ablakban állítsd be az alábbi űrlaptulajdonságokat.
   1. Írd át az űrlap **Text (Szöveg)** tulajdonságát **Form1-ről** **Memóriajáték-ra** . Ez a szöveg a játék ablakának tetején jelenik meg.
   2. Állítsd be az űrlap méretét 550 pixel szélességű és 550 pixel magasságra. Ez beállítható úgy, hogy a **Size** tulajdonságot **550, 550 értékre** állítod be, vagy húzd az űrlap sarkát, amíg meg nem jelenik a megfelelő méret az integrált fejlesztési környezet (IDE) jobb alsó sarkában.
2. Az eszközkészlet megjelenítéséhez válaszd az **ToolBox (Eszköztár)** fület az IDE bal oldalán.
3. Húzd az űrlapra [TableLayoutPanel](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.tablelayoutpanel) vezérlőt az eszközkészlet **Containers** kategóriájából, majd állítsd be a következő tulajdonságokat.
   1. Állítsa be a **BackColor (Háttérszín)** értékét **CornflowerBlue színre**. Ehhez nyisd meg a **BackColor** dialógust a **Properties** ablakban a **BackColor** tulajdonság melletti legördülő nyíl kiválasztásával. Ezután válaszd a **Web** fület a **BackColor** párbeszédpanelen az elérhető **színnevek** listájának megtekintéséhez.

**Megyjegyzés**

A színek nem ábécé sorrendben vannak, a **CornflowerBlue** a lista alján található.

* 1. Állítsd a **Dock** tulajdonságot **Fill (Kitöltés) értékére** melletti legördülő elem nagy középső gomb kiválasztásával. Ez eloszlatja a táblázatot úgy, hogy az a teljes űrlapot lefedje.
  2. Állítsd a **CellBorderStyle (Cellahatár stílus)** tulajdonságot **Inset (Belső) értékre**. Ezután kirajzolja határokat a táblázatban lévő cellák között.
  3. Válaszd ki a TableLayoutPanel jobb felső sarkában található háromszög gombot a feladatmenü megjelenítéséhez.
  4. A feladatmenüben kattints kétszer az **Add row (Sor hozzáadása) lehetőségre sorok beillesztéséhez és kattints** kétszer az **Add Colum (Oszlop hozzáadása)** elemre további két **oszlop** hozzáadásához.
  5. A feladatmenüben válaszd az **Edit Rows and Columns (Sorok és oszlopok szerkesztése) elemet, hogy megjelenlen a Column and Row Styles (Oszlop- és sorstílusok)** ablak. Válaszd ki az oszlopokat, jelöld ki a **Percent (Százalék)** gombot, majd állítsd az oszlopok szélességét a teljes szélesség 25 százalékára. Ezután válaszd ki a **Rows (Sorok)** elemet az ablak tetején lévő legördülő menüből, és állítsd be az egyes sorok magasságát 25 százalékra. Ha kész, zárd be az ablakot az **OK** gombbal.

A TableLayoutPanel most 4x4-es rácsosnak kell lennie, tizenhat azonos méretű négyzet alakú cellával. Ezekben a sorokban és oszlopokban jelennek meg később az ikonképek.

1. Ügyelj arra, hogy a TableLayoutPanel legyen kiválasztva az a szerkesztőben. Ennek ellenőrzéséhez a **Tulajdonságok** ablak tetején kell megjelennie a **tableLayoutPanel1 névnek**. Ha nincs kiválasztva, akkor klikkelj az űrlapon a TableLayoutPanel elemre, vagy a **Tulajdonságok** ablak tetején lévő legördülő menüben válaszd ki.

Miközben a TableLayoutPanel van kiválasztva, nyisd meg az eszköztárt, és adj hozzá egy Label (Címke) elemet (a **Common Controls** kategóriában találod) a táblázat bal felső cellájába. Válaszd ki a címke elemet az IDE-ben. Állítsd be a következő tulajdonságait.

* 1. Ügyelj arra, hogy a címke **BackColor** tulajdonság értéke **CornflowerBlue**.
  2. Állítsd az **AutoSize** tulajdonságot **False (Hamis) értékre.**
  3. Állítsd a **Dock** tulajdonságot **Fill** értékre.
  4. Állítsd a **TextAlign (Szöveg igazítás)** tulajdonságot **MiddleCenter (Közép-Közép)-re** úgy, hogy a tulajdonság melletti legördülő menüt megnyitod, majd kiválasztod a középső gombot. Ez igazítja az ikont a cella közepére.
  5. Válaszd a **Font (Betűkészlet)** tulajdonságot. Megjelenik egy választó ( **...** ) gomb.
  6. Kattints a három pontra, és állítsd be a **Font** értéket **Webdings-re**, a **Font Style (Betűstílus)** legyen **Bold (Félkövér)** és a **Size** a **48** .
  7. Állítsd a címke **Text** tulajdonságát **c** betűre.

Most már a TableLayoutPanel bal felső cellájában egy fekete keretes doboznak kell lennie kék háttérrel.

**Megjegyzés**

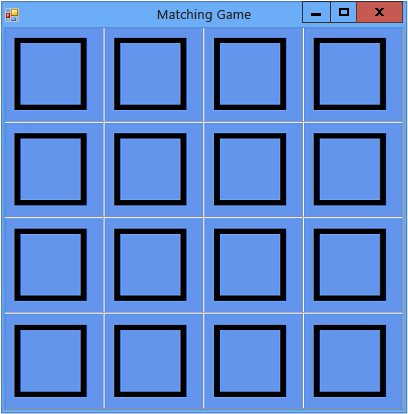
A Webdings egy beépített betűtípus a Windows operációs rendszerben. A játékban a játékosnak párt kell keresnie az ikonoknak, így ezt a betűtípust fogja megjeleníteni az ikonokat. A **c** betű helyett próbáld ki a többi karaktert is a címke **Text** tulajdonságban, hogy lásd milyen ikonok jelennek meg. A felkiáltójel egy pók, a nagy N betű egy szem, a vessző pedig egy chilipaprika.

1. Válaszd ki a Label elemet, és másold át a TableLayoutPanel következő cellájába. (Nyomd meg a **Ctrl** + **C** billentyűket, vagy a menüsoron válassza az **Edit (Szerkesztés)** > **Copy (Másolás) lehetőséget**.) Ezután illeszd be. (Nyomd meg a **Ctrl** + **V** billentyűket, vagy a menüsoron válassza az **Edit** > **Paste (Beillesztés) elemet**.) Az első címke másolata megjelenik a TableLayoutPanel második cellájában. Illeszd be újra, és egy másik címke jelenik meg a harmadik cellában. Illeszd be a címkéket az összes cellába.

**Megjegyzés**

Ha túl sokszor illesztesz be, akkor az IDE új sort ad hozzá a TableLayoutPanelhez, hogy ott legyen helye az új címkének. Az új cella eltávolításához válaszd a **Ctrl** + **Z** billentyűket, vagy a menüsoron válaszd az **Edit (Szerkesztés)** > **Undo (Visszavonás) lehetőséget**.

Az játék alapjai elkészültek. A végén a következő képhez hasonlóan kell kinéznie:

  
A játék kezdeti formája

# 2. lépés: Véletlenszerű objektumok és az ikonlista elkészítése

Ebben a fejezetben hozzuk létre a szimbólumpárokat a játékhoz. Mindegyik szimbólum hozzáadódik két véletlenszerű cellához az űrlapon található TableLayoutPanel alkalmazásban.

Ehhez két new utasítás segítségével két objektumot fogunk létrehozni. Az első egy [véletlenszerű](https://docs.microsoft.com/en-us/dotnet/api/system.random) objektum. Ezzel lehet véletlenszerűen kiválasztani a TableLayoutPanel celláját. A második objektum, egy [Lista <T>](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1) objektum, amit a véletlenszerűen kiválasztott szimbólumok tárolására fogunk használni.

## Véletlenszerű objektum és ikonok listájának hozzáadása

1. A **Solution Explorer-ben** válaszd az Form1.cs fájlt, majd a menüsoron válassza a **View (Nézet)** > **Code (Kód) lehetőséget**. Alternatív megoldásként nyomd meg az **F7** billentyűt, vagy duplán kattintva a **Form1-re** a **Solution Explorer-ben**.

Megjelenik a Form1 mögött található kódrészlet.

1. A meglévő kódhoz add hozzá a következő kódot.

public partial class Form1 : Form

{

// A Random objektum fogja kiválasztani az ikonokat a cellákba

Random random = new Random();

// Mindegyik betű egy érdekes ikon

// a Webdings font készletében

// és mindegyik ikon kétszer szerepel a listában

List<string> icons = new List<string>()

{

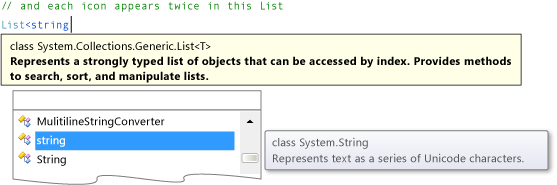
"!", "!", "N", "N", ",", ",", "k", "k",

"b", "b", "v", "v", "w", "w", "z", "z"

};

Feltétlenül helyezd be a kódot a nyitó kapcsos zárójel után és közvetlenül az osztály deklaráció után (public partial class Form1 : Form).

1. A Lista objektum hozzáadásakor támaszkodj nyugodtan a megnyíló **IntelliSense** súgóra.

  
***IntelliSense*** ablak

**Megjegyzés**

Az IntelliSense ablak csak akkor jelenik meg, ha a kódot kézzel írod be. Ha másolással és beillesztéssel raktad be a kódot, akkor nem jelenik meg.

Ha a kódot (és a megjegyzéseket) kis részeletekben tekinted át, könnyebb lesz megérteni. A programok listás objektumokat használhatnak sokféle elem nyomon követésére. Egy lista számokat, valódi / hamis értékeket, szöveget vagy egyéb objektumokat tartalmazhatnak. Rendelkezhet olyan listaobjektummal is, amely más listaobjektumokat tartalmaznak. A listában szereplő tételeket elemeknek nevezzük, és minden lista csak egy típusú elemet tartalmaz. Tehát a számok listája csak számokat tartalmazhat - nem adható szöveg ehhez a listához. Hasonlóképpen, nem adhat számokat a szöveges listához.

Amikor egy List objektumot készítesz a new kulcsszó felhasználásával, meg kell határoznod, hogy milyen típusú adatokat szeretnél tárolni az objektumban. Az **IntelliSense** ablak tetején található eszköztipp ezért jeleníti meg a listában szereplő elemek típusát.

1. C#-ban a listát egyetlen állítással lehet létrehozni. Ennek oka az, hogy a C # nyelvnek vannak gyűjteményindítói , amelyek előkészítik a listát az értékek elfogadásához.

Amikora a listát példányosítod a new kulcsszóval, akkor az új Lista objektum létrehozása után a program kitölti azt a kapcsoszárójelben megadott adatokkal. A hozzáadott kódsorban egy ikont tartalmazó karakter listát hoztuk létre, úgy, ami 16 karaktert tartalmaz.

Ezeknek a karakternek mindegyike egy betû, és mindegyik megfelel egy ikonnak. Tehát a játékban lesz egy pár felkiáltójel, nagybetűs N betű, vessző és így tovább. (Ha ezeket a karaktereket a Webdings betűkészletre állítod, szimbólumokként jelennek meg, például busz, kerékpár, pók stb.) A listaobjektum összesen tizenhat karakterláncot tartalmaz a TableLayoutPanel minden cellájához.

# 3. lépés: Ikonok rendelése címkékhez

Ha az ikonok mindig ugyan azokban a cellákban jelennek meg, úgy a játék nem lesz túl nagy kihívás. Ennek elkerülése érdekében véletlenszerűen kell az ikonokat hozzárendelni egy-egy címkéhez az AssignIconsToSquares() metódussal.

## Véletlenszerű ikon hozzárendelése minden címkéhez

1. A következő kód hozzáadása előtt nézd át a működését is.

/// <summary>

/// Minden ikont hozzárendel egy véletlenszerű négyzethez

/// </summary>

private void AssignIconsToSquares()

{

// A TableLayoutPanel-nek 16 címkéje van,

// és a listának 16 ikonja,

// tehát véletlenszerűen mindegyik ikont hozzá kell adni

// mindegyik címkéhez a nézetben

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

int randomNumber = random.Next(icons.Count);

iconLabel.Text = icons[randomNumber];

// iconLabel.ForeColor = iconLabel.BackColor;

icons.RemoveAt(randomNumber);

}

}

}

1. Add hozzá a AssignIconsToSquares()metódust a korábban bemutattak szerint. Helyezheted közvetlenül a [2. lépésben](https://docs.microsoft.com/en-us/visualstudio/ide/step-2-add-a-random-object-and-a-list-of-icons?view=vs-2019) hozzáadott kód alá: [Véletlenszerű objektumok és az ikonlista elkészítése](https://docs.microsoft.com/en-us/visualstudio/ide/step-2-add-a-random-object-and-a-list-of-icons?view=vs-2019).

A For Each ciklus bármikor használható, amikor ugyanazt a műveletet többször el akarod végezni. Ebben az esetben ugyanazokat a parancsokat kell végrehajtani a [TableLayoutPanel](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.tablelayoutpanel) minden címkéjén, ahogy a kód leírta. Az első sor létrehoz egy control nevű változót, ami az egyes vezérlőelemeket egyenként tárolja, míg a ciklusban lévő utasításokat végrehajtja rajta.

foreach (Control control in tableLayoutPanel1.Controls)

{

// A végrehajtandó utasítások

// az egyes címkékhez

// Az állítások az iconLabel segítségével

// férnek hozzá mindegyik címke tulajdonságához és metódusaihoz

}

**Megjegyzés**

Az "iconLabel" és a "control" elnevezést azért használtuk, mert leíró jellegűek. Ezeket a neveket bármilyen névvel helyettesítheted, és a kód pontosan ugyanúgy fog működni, amíg a ciklusban lévő utasításokban összes nevet megváltoztatod.

Az AssignIconsToSquares() metódus minden, a TableLayoutPanel címkén keresztül ismétlődik, és mindegyikre ugyanazokat az utasításokat hajtja végre. Ezek az utasítások véletlenszerű ikont húznak a [2. lépésben](https://docs.microsoft.com/en-us/visualstudio/ide/step-2-add-a-random-object-and-a-list-of-icons?view=vs-2019) hozzáadott listából [: Adjunk hozzá egy véletlenszerű objektumot és az ikonok listáját](https://docs.microsoft.com/en-us/visualstudio/ide/step-2-add-a-random-object-and-a-list-of-icons?view=vs-2019). (Ezért kellett minden ikont kétszer hozzáadni, így lesznek egy párban a véletlenszerű címkékhez rendelve.)

Vizsgáld meg közelebbről a foreach belsejében lévő kódot.

Label iconLabel = control as Label;

if (iconLabel != null)

{

int randomNumber = random.Next(icons.Count);

iconLabel.Text = icons[randomNumber];

// iconLabel.ForeColor = iconLabel.BackColor;

icons.RemoveAt(randomNumber);

}

Az első sor a **control** változót átalakítja az **iconLabel** nevű címkeelemmé. Az ezt követő sor egy olyan if állítás, amely ellenőrzi, hogy sikeres volt-e az átalakítás. Ha a konvertálás sikeres, akkor az utasításban szereplő if utasítások futnak le.

Az if feltétel alatt lévő utasítás első sora egy **randomNumber** nevű változót hoz létre, amely egy véletlenszerű számot tartalmaz, amely megfelel az **ikonlistában** szereplő elemek egyikének. Ehhez a [Random](https://docs.microsoft.com/en-us/dotnet/api/system.random) objektum [Next()](https://docs.microsoft.com/en-us/dotnet/api/system.random.next#System_Random_Next) metódusát használja , amelyet korábban hoztunk létre. A metódus egy véletlenszerű számot ad vissza.

A [Count](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1.count?view=netframework-4.8#System_Collections_Generic_List_1_Count) tulajdonság szabja meg a véletlenszerű számok felső tartományát, ahonnan választani lehet a véletlen számot. A következő sor az ikonlisták egyikét hozzárendeli a címke [Szöveg](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.label.text#System_Windows_Forms_Label_Text) tulajdonságához. A kommentált vonalat később ismertetjük ebben a fejezetben. Végül az if nyilatkozat utolsó sora eltávolítja a listáról az űrlaphoz hozzáadott ikont.

Ne feled, ha nem vagy biztos benne, hogy a kód valamely része mit csinál, akkor az egérmutatót helyezd egy kód elem fölé, és olvasd el a kapott szövegbuborékot. A Visual Studio debug módjával a program futása közben is végig lépdelhetsz minden kódsoron. Lásd: [Hogyan léphetek végig a kódon Visual Studio hibakeresőjével](https://docs.microsoft.com/en-us/visualstudio/debugger/navigating-through-code-with-the-debugger?view=vs-2019).

1. Ahhoz, hogy feltöltődjön a játéktábla ikonokkal meg kell hívnunk az AssignIconsToSquares() metódust ahogy a program elindul. Az InitializeComponent() metódus alatt a **Form1**konstruktor metódusához illesszük be a megírt metódust, így az űrlapon már azelőtt lefut, mielőtt megjelenik. A konstruktorok akkor futnak le, amikor új objektumot, például osztályt vagy struktúrát hozzunk létre. További információkért lásd: [konstruktorok (C # programozási útmutató)](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors).

public Form1()

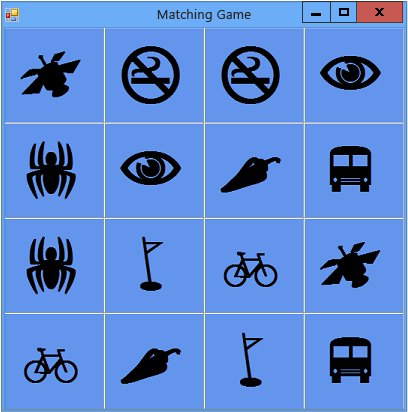
{

InitializeComponent();

AssignIconsToSquares();

}

1. Mentsd el a programot, és futtasd le. Meg kell jelennie az ablaknak, amelyen véletlenszerű ikonok vannak társítva az egyes címkékhez.
2. Zárd be a programot, majd futtasd le újra. Figyeld meg, hogy az egyes címkékhez különböző ikonok vannak hozzárendelve, ahogy az a következő képen látható.

  
Memóriajáték véletlenszerű ikonokkal

Az ikonok most még láthatók, mert még nem rejtetted el őket. Ha el akarod rejteni őket a játékban, állítsd be egyes címkék **ForeColor** tulajdonságát ugyanolyan színűre, mint a **BackColor** tulajdonsága.

**Tipp**

A vezérlők, például a címkék elrejtésének másik módja az, hogy a **Visible (Látható)** tulajdonságot **False értékre** állítod.

1. Az ikonok elrejtéséhez állítsd le a programot, és távolítsd el a megjegyzéseket a For Each ciklus következő sorában:

iconLabel.ForeColor = iconLabel.BackColor;

1. A menüsoron válaszd a **Save All (Összes mentése)** gombot a program mentéséhez, majd futtasd le. Úgy tűnik, hogy az ikonok eltűntek - csak a kék háttér jelenik meg. Azonban az ikonok véletlenszerűen vannak lettek kiosztva, és továbbra is ott vannak. Mivel az ikonok színe ugyanolyan, mint a háttéré, így el vannak rejtve a játékostól. Hiszen, ha mindent látna, akkor nem is lenne olyan nagy kihívás a játék!

# 4. lépés: Klikk esemény hozzáadása minden címkéhez

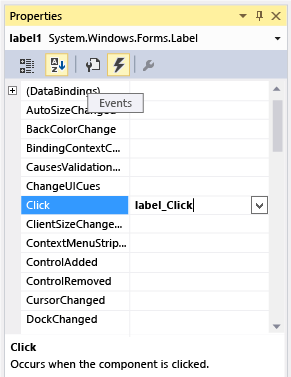
A memóriajáték a következőképpen működik:

1. Amikor egy játékos kiválaszt egy rejtett ikonos cellát, akkor a program az ikon színének feketére változtatja, megjelenítve az ikont.
2. Ezután a játékos választ egy másik rejtett ikont.
3. Ha az ikonok megegyeznek, akkor láthatóak maradnak. Ha nem, akkor mindkét ikon ismét el lesz rejtve.

Annak érdekében, hogy a program ilyen módon működjön, hozzá kell adni egy [Click](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.click) eseménykezelőt, amely megváltoztatja a kiválasztott címke színét.

## Klikk esemény hozzáadása az egyes címkékre

1. Nyisd meg az ablakot a **Windows Forms Designer alkalmazásban** . A **Solution Explorer alkalmazásban** válaszd az Form1.cs fájlt. A menüsoron válassza a **View (Nézet** > **Designer (Tervező) lehetőséget**.
2. Válaszd ki az első címkevezérlőt. Ezután tartsd lenyomva a **Ctrl** gombot, miközben a többi címkét kiválasztod. Ellenőrizd le, hogy minden címke ki van-e választva.
3. Válassza a **Tulajdonságok** ablak eszköztár **Events (Események)** gombot az **Események** oldal megjelenítéséhez a **Tulajdonságok** ablakban. Görgess le a **Click (Kattintás)** eseményig, és írd be a mezőbe **label\_Click** szöveget a következő képernyőképen látható módon.



1. Nyomd meg az **Enter** billentyűt. Az IDE hozzáad egy Click eseménykezelőt label\_Click() névvel, és összeköti az űrlap minden címkéjével.
2. Töltsd ki a kód többi részét az alábbiak szerint:

/// <summary>

/// Minden címke Klikk eseményét kezeli

/// </summary>

/// <param name="sender">A címke, amire kattintottak</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// Ha a címke színe fekete, akkor a játékos már rákattintott

// és az ikon is megjelent

// így hagyja figyelmen kívül az utasítást

if (clickedLabel.ForeColor == Color.Black)

return;

clickedLabel.ForeColor = Color.Black;

}

}

**Megjegyzés**

Ha a label\_Click() kódblokkot másolod és beilleszti ahelyett, hogy a kódot kézzel beírnád, akkor mindenképpen cseréld ki a meglévő label\_Click() kódot. Ellenkező esetben egy duplikált kódblokk lesz a végén.

**Megjegyzés**

Mivel a különböző címkevezérlő kattintási eseményeket egyetlen eseménykezelő metódusra lettek kapcsolva, így ugyanazt a metódust hívják meg, függetlenül attól, hogy melyik címkét választja a játékos. Az eseménykezelő módszerének tudnia kell, melyik címkét választotta, tehát a sender nevet használja címke azonosításához. A metódus első sora azt mondja a programnak, hogy nem csak általános objektum, hanem kifejezetten egy címkéről van szó, és a clickedLabel nevet használja címke tulajdonságainak és módszereinek eléréséhez.

A metódus először ellenőrzi, hogy clickedLabel az objektumot sikerült-e konvertálni (kasztolni) címke vezérlőelemmé. Sikertelen érték esetén null lesz az értéke, és nem akarja végrehajtani a kód többi részét a metódusban. Ezután a metódus ellenőrzi a kiválasztott címke színét a címke **ForeColor** tulajdonságának kiolvasásával. Ha a címke színe szövege fekete, akkor ez azt jelenti, hogy az ikont már kiválasztották, és a metódus befejeződött. (Ezt az return állítás hajtja végre: Azt mondja a programnak, hogy hagyja abba a metódus folyatását.) Ellenkező esetben az ikont nem választották ki, tehát a program megváltoztatja a címke szövegét fekete színre.

1. A menüsoron válaszd a **Fájl** > **Minden mentése menüpontot** az folyamat mentéséhez, majd a menüsorban válaszd a **Debug (Hibakeresés)** > **Start Debugging (Hibakeresés** **indítása)** lehetőséget a program futtatásához. Üres kék háttérrel rendelkező űrlapot kell látnod. Válaszd az ablak bármelyik celláját, és az egyik ikonnak láthatóvá kell válnia. Folytasd a különféle helyek kiválasztását a játékban. Amiket kiválasztottál ikonokat, azoknak kell megjelenniük.

# 5. lépés: Címke referenciák hozzáadása

A programnak nyomon kell követnie, hogy melyik Címke vezérlőt választotta ki a játékos. Jelenleg a program megmutatja a játékos által választott összes címkét. Ezt fogjuk most megváltoztatni. Az első címke kiválasztása után a programnak meg kell mutatnia a címke ikonját. A második címke kiválasztása után a programnak rövid ideig mindkét ikont meg kell jelenítenie, majd ismét el kell rejtenie mindkét ikont. A program nyomon fogja követni, melyik címkevezérlőt választja ki először, és melyiket másodjára a referencia változók felhasználásával.

## Címke referenciák hozzáadása

1. Címke referencia készítéséhez add hozzá a következő kódot.

public partial class Form1 : Form

{

// firstClicked az elsőnek kiválasztott címkére fog mutatni

// amire a játékos klikkelt, de ennek értéke addig null

// amíg a játékos rá nem kattintott egy címkére

Label firstClicked = null;

// secondClicked a második címkére fog hivatkozni

// amire a játékos kattint

Label secondClicked = null;

A referenciaváltozók hasonlóak azokhoz az objektumokhoz, amelyeket korábban találkozhattál (például [időzítő](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.timer) objektumok, [<T> lista](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1) objektumok és [véletlenszerű](https://docs.microsoft.com/en-us/dotnet/api/system.random) objektumok) az űrlapon. Ezek a változók azonban nem adnak hozzá két további címkevezérlőt az űrlapon, mert a new kulcsszó használata nélkül egyik sem jön létre. A new kulcsszó nélkül objektum nem kerül létrehozásra. Ezért firstClickedés secondClickedhivatkozási változóknak nevezzük: Csak nyomon követik (vagy hivatkoznak) Címkeobjektumok.

Ha egy változó nem követi nyomon az objektumot, akkor egy speciális fenntartott értékre állítja be: null. Így, amikor a program elindul, mind a firstClicked és a secondClicked kezdeti értéke null, ami azt jelenti, hogy a változókat nem követnek nyomon semmit.

1. Módosítsd a [Click](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.click) eseménykezelőt az új firstClicked referenciaváltozó felhasználásával. Töröld ki az label\_Click() eseménykezelő metódus utolsó sorát ( clickedLabel.ForeColor = Color.Black;) és cseréld le a következő if utasításra. (Ügyelj arra, hogy tartalmazza a kommenteket és a teljes if feltételt.)

/// <summary>

/// Minden címke Klikk eseményét kezeli

/// </summary>

/// <param name="sender">A címke, amire kattintva lett</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// Ha a címke színe fekete, akkor a játékos már rákattintott

// és az ikon is megjelent

// akkor hagyja figyelmen kívül az utasítást

if (clickedLabel.ForeColor == Color.Black)

return;

// Ha a firstClicked értéke null, akkor ez az első ikon

// a párok közül, amit a játékos választott ki,

// így a firstClicked a kiválasztott címkével lesz egyenlő

// a szöveg színe fekete lesz, majd visszatér

if (firstClicked == null)

{

firstClicked = clickedLabel;

firstClicked.ForeColor = Color.Black;

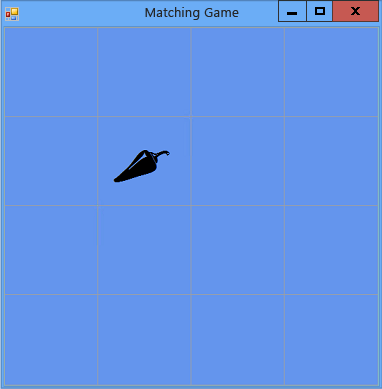
return;

}

}

}

1. Mentsd el és futtasd a programot. Válasz az egyik címkevezérlőből, majd megjelenik annak az ikonja.
2. Miután kiválasztod a következő ikont, észre vetted, hogy nem történik semmi. A program már nyomon követi az első címkét, amelyet a játékos választott, tehát firstClicked értéke már nem null. Mikor az if feltétel ellenőrzi firstClicked értékét, hogy egyenlő-e null-al, és megállapítja, hogy már nem, akkor a feltétel alatti utasításokat már nem hajtja végre. Csak az első kiválasztott ikon válik feketévé, a többi ikon pedig láthatatlan marad, amint az a következő képen látható.

  
***Memóriajáték***, egy ikon megjelenítése

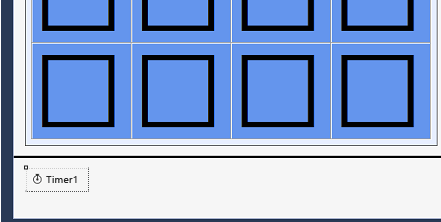
Ezt a problémát az következő lépésében tudod kijavítani egy **Timer (Stopper)**  hozzáadásával.

# 6. lépés: Stopper hozzáadása

Ebben a lépésben hozzáadunk egy [Stopper](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.timer) vezérlőt a memóriajátékhoz. Az időzítő megadott számú milliszekundumot vár, majd elindítja az eseményt, tick néven hivatkozva rá. A stopper segít egy művelet elindításához, vagy egy művelet rendszeres megismétléséhez. A stoppert, úgy fogjuk felhasználni, ha a játékos két ikont kiválaszt és ha az ikonok nem egyeznek meg, rövid idő elteltével ismét elrejti a két ikont.

## Stopper készítése

1. A **Windows Forms Designer** eszköztárában válaszd a **Timer (Stopper)** (az **Components (Összetevők)** kategóriában) lehetőséget, majd nyomd meg az **Enter** billentyűt, vagy kattintson duplán az stopperre a vezérlő hozzáadásához. Az stopper ikonjának, ami **Timer1-nek lett elnevezve**, az űrlap alatti **helyen** kell megjelennie, ahogy az a következő képen látható.

  
***Stopper***

**Megjegyzés**

Ha az eszköztár üres, akkor a ToolBox megnyitása előtt feltétlenül válassza ki az űrlaptervezőt, és nem az űrlap mögött lévő kódból add hozzá.

1. Válaszd ki a **Timer1** ikont az időzítő beállításához. A **Tulajdonságok** ablakban válts az események megtekintésére. Állítsd át a Stopper **Interval (Intervallum)** értékét **750-re**, és hagyd az **Enabled (Engedélyezve)** tulajdonságát **Hamis értéken**. Az **Interval** megmondja az stoppernek, hogy meddig kell várakozni a tick-ek között, vagy mikor indítsa el a [Tick](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.timer.tick) eseményt. A 750 érték azt jelzi, hogy az stoppernek 1 másodperc háromnegyedét (750 milliszekundumot) várnia kell, mielőtt elindítja a Tick eseményt. Ezután elindítja a [Start ()](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.timer.start#System_Windows_Forms_Timer_Start) metódust, miután a játékos kiválasztotta a második címkét.
2. Válaszd az stopper ikont a **Windows Forms Designer alkalmazásban**, majd nyomd meg az **Enter** billentyűt, vagy kattints duplán az stopperre, hogy hozzáadjon egy üres Tick eseménykezelőt.

/// <summary>

/// Az stopper elindul, ha a játékos

/// két ikonra klikkel, amik nem egyeznek meg,

/// addig elszámol háromnegyed másodpercig,

/// majd leáll és elrejti mindkét ikont

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void timer1\_Tick(object sender, EventArgs e)

{

// Megállítja a stoppert

timer1.Stop();

// Elrejti mindkét ikont

firstClicked.ForeColor = firstClicked.BackColor;

secondClicked.ForeColor = secondClicked.BackColor;

// Visszaállítja a firstClicked és secondClicked

// értékét, így legközelebb ha címkére kattint,

// akkor tudja majd a program, hogy ez lesz az első kattintás

firstClicked = null;

secondClicked = null;

}

A Tick eseménykezelő három dolgot végez: először, hogy a stopper biztos ne mérjen a [Stop ()](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.timer.stop#System_Windows_Forms_Timer_Stop) metódussal megállítja. Majd két referenciaváltozót használ, firstClicked-et és secondClicked-et annak érdekében, hogy a játékos által kiválasztott két címke ikonjai ismét láthatatlanná váljanak. Végül visszaállítja a firstClicked és a secondClicked változókat null értékre. Fontos ezt a lépést végrehajtani, mert így áll vissza a program alaphelyzetbe. Ezután már nem követi nyomon a [Label](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.label) vezérlőelemeit, és készen áll arra, hogy a játékos újra kiválasztja a címkéket.

**Megjegyzés**

Az Stopper objektumnak van egy Start() metódusa, amely elindítja az időmérést, és egy Stop()módszer, amely leállítja azt. Ha a stopper **Enabled (Engedélyezve)** értéke **True (Igaz)** a **Tulajdonságok** ablakban, az időmérés rögtön elindul, amint a program megnyílik. Ha viszont **False (Hamis)** az értéke , addig nem kezdi el az időmérést, amíg a Start() metódus nincs meghívva. Általában az időzítő újra és újra elindítja a Tick eseményét, az **Interval** tulajdonság által meghatározva, hogy hány milliszekundumot kell várni a Tick események között. Láthattad, hogy a Stop()metódus meg lett hívva a Tick eseményen belül. Ez a stoppert egyszeri indítási üzemmódba helyezi, ami azt jelenti, hogy amikor a Start() metódust meghívják, megvárja a megadott intervallumot, egyetlen Tick eseményt indít el, majd leáll.

1. Az új időzítő működésének megtekintéséhez lépj a kódszerkesztőbe, és a label\_Click() metódus tetejéhez és aljára add hozzá a következő kódot . (Egy if feltételt kell hozzáadni tetejéhez, és három utasítást az aljára; a kód többi része változatlan marad.)

/// <summary>

/// Minden címke Klikk eseményét kezeli

/// </summary>

/// <param name="sender">The label that was clicked</param>

/// <param name="e"></param>

private void label\_Click(object sender, EventArgs e)

{

// The stopper csak akkor működik, ha két nem egyező

// ikont jelenített meg a játékosnak,

// addig minden klikkelést figyelmen kívül hagy

if (timer1.Enabled == true)

return;

Label clickedLabel = sender as Label;

if (clickedLabel != null)

{

// Ha a címke színe fekete, akkor a játékos már rákattintott

// és az ikon is megjelent

// akkor figyelmen kívül az utasítást

if (clickedLabel.ForeColor == Color.Black)

return;

// Ha a firstClicked értéke null, akkor ez az első ikon

// a párok közül, amit a játékos választott ki,

// így a firstClicked a kiválasztott címkével lesz egyenlő

// a szöveg színe fekete lesz, majd visszatér

if (firstClicked == null)

{

firstClicked = clickedLabel;

firstClicked.ForeColor = Color.Black;

return;

}

// Ha a stopper még nem indult el

// és a firstClicked értéke nem null,

// így a második ikonnak, amire a játékos klikkelt

// a színe fekete lesz

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

// Ha a játékos két különböző ikonra kattint,

// akkor indítsa el a Stopper időmérését

// (várni fog háromnegyed másodpercet

// azután elrejti az ikonokat)

timer1.Start();

}

}

A metódus tetején lévő kód ellenőrzi az **Engedélyezve** tulajdonság értékét, hogy az időzítő elindult-e. Ha a játékos kiválasztja az első és a második címkét, és az időzítő elindul, a harmadik címke kiválasztása esetén semmit sem fog tenni.

A metódus alján látható kód beállítja a secondClicked referenciaváltozót a játékos által választott második címkevezérlő nyomon követésére, majd a címke ikon színét feketere állítja be, hogy láthatóvá váljon. Ezután elindítja az időzítőt egy lövés üzemmódban, így 750 milliszekundumot vár, majd egyetlen Tick eseményt indít. Az időzítő Tick eseménykezelője elrejti a két ikont, és visszaállítja a firstClicked és a secondClicked referencia változókat, így az űrlap készen áll arra, hogy a játékos válasszon egy másik ikonpárt.

1. Mentsd el és futtasd a programot. Válasz egy ikont, és az láthatóvá válik.
2. Válasz egy másik ikont. Röviden megjelenik, majd mindkét ikon eltűnik. Ismételd meg ezt többször. Az űrlap nyomon követi az általad választott első és második ikont, és az időzítő segítségével késlelteti az ikonok elrejtését.

# 7. lépés: Párok láthatóvá tétele

A játék jól működik, mindaddig, amíg a játékos csak azokat az ikonpárokat választja, amelyek nem egyeznek meg. Gondoljuk át, hogy mi történjen, ha a játékos kiválaszt egy párt. Ahelyett, hogy az ikonokat az időzítő bekapcsolásával eltűntetnénk), a játéknak vissza kell állítania magát, hogy a továbbiakban ne kövesse a firstClicked és a secondClicked változókat címkéket , anélkül, hogy visszaállítaná a két választott címke színét.

## Párok láthatóságának fenntartása

1. Add hozzá a következő if állítást a label\_Click() metódushoz, a kód végénél, közvetlenül annak az utasítás fölé, ahol elindítja az időzítőt.

// Amikor a játékos ide elér, a stopper nem számol

// és a firstClicked értéke nem null,

// tehát a második ikonra klikkelt a játékos

// Beállítja a színét feketére

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

// Ha a játékos két egyező ikonra klikkel, akkor feketén maradnak

// és visszaállítja a firstClicked és a secondClicked értékeit,

// hogy egy másik ikonra kattinthasson

if (firstClicked.Text == secondClicked.Text)

{

firstClicked = null;

secondClicked = null;

return;

}

// Ha a játékos ide elér,

// akkor két különböző ikonra kattintott

// a stopper időzítő elindul

// majd elrejti az ikonokat

timer1.Start();

}

}

Az if feltétel első ellenőrzi, hogy a játékos által választott első címkén szereplő ikon megegyezik-e a második címkén szereplő ikonnal. Ha az ikonok azonosak, a program végrehajtja a következő három utasítást. Az első két állítás visszaállítja a firstClicked és a secondClicked referenciaváltozókat, így már nem követik nyomon a címkék egyikét sem. (Ezt a két utasítást felismerheted az időzítő [Tick](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.timer.tick) eseménykezelője alapján.) A harmadik utasítás egy return, amely azt mondja a programnak, hogy fejezze be a metódust többi utasítás végrehajtása nélkül.

Megfigyelheted, hogy a kód egy része egyetlen egyenlő jelet ( =) használ, míg más utasítások két egyenlő jelet ( ==) használnak. Nézzük meg, hogy miért.

Ez egy jó példa, amely megmutatja a különbséget. Az if feltétel zárójeleiben lévő kód:

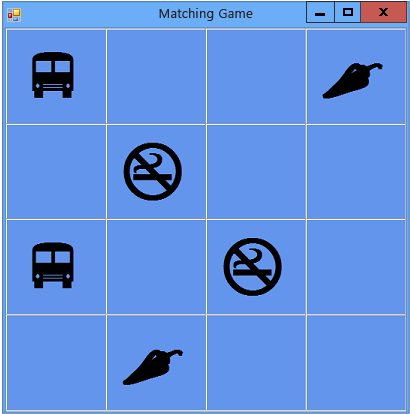
firstClicked.Text == secondClicked.Text

Az első mondat az if utasítás után:

firstClicked = null;

A két állítás közül az első ellenőrzi, hogy a két ikon azonos-e. Mivel két értéket hasonlítanak össze, a C# az == egyenlőség operátort használja. A második utasítás ténylegesen megváltoztatja az értéket (úgynevezett hozzárendelés), és a firstClicked referenciaváltozót értékét nulla-ra állítja a visszaállításhoz. Ezért használja  = a hozzárendelési operátort.

1. Mentsd el és futtasd a programot, majd kezdj el kiválasztani az űrlapon lévő ikonokat. Ha olyan párt választasz, amely nem egyezik meg, akkor az stopper Tick esemény aktiválódik és mindkét ikon eltűnik. Ha egyező párt választasz, akkor az új if utasítás lefut és a visszatérési utasítás miatt a módszer kihagyja az stoppert indító kódot, így az ikonok láthatóak maradnak, amint a következő képen látható.

  
***Memóriajáték*** látható ikonpárokkal

# 8. lépés: Metódus hozzáadása annak ellenőrzésére, hogy a játékos nyert-e

Készítettünk egy szórakoztató játékot, de még egy további elemre szükség lesz. A játéknak akkor kell befejeződnie, amikor a játékos nyer, ezért hozzá kell adni egy CheckForWinner() metódust annak ellenőrzésére, hogy a játékos végzett-e.

## Metódus hozzáadása annak ellenőrzéséhez, hogy a játékos nyert-e

1. Adj hozzá egy CheckForWinner()metódust a kódk aljára, a timer1\_Tick() eseménykezelő alá, a következő minta szerint.

/// <summary>

/// Ellenőrzi, hogy meg lett e az összes ikon párja úgy,

/// hogy összehasonlítja az előtér színét a háttérszínnel.

/// Ha mindegyik ikonnak megegyezik, akkor a játékos nyert

/// </summary>

private void CheckForWinner()

{

// Végighalad mindegyik címkén a TableLayoutPanel-en,

// ellenőrzi, hogy mindegyik ikon megegyezik-e

foreach (Control control in tableLayoutPanel1.Controls)

{

Label iconLabel = control as Label;

if (iconLabel != null)

{

if (iconLabel.ForeColor == iconLabel.BackColor)

return;

}

}

// Ha a ciklus nem tér vissza, akkor nem talált

// több párosítatlan ikont

// Ami azt jelenti, hogy a játékos nyert. Megmutatja az üzenetet és bezárja az űrlapot

MessageBox.Show("Sikerült megtalálni az összes párt!", "Gratulálok");

Close();

}

A metódus egy másik foreach ciklus használ, hogy végighaladjon a [TableLayoutPanel](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.tablelayoutpanel) egyes címkéin. Az egyenlőség operátort használja az egyes címkék ikonjának színének ellenőrzéséhez, hogy ellenőrizze, hogy megegyezik-e a háttér színével. Ha a színek megegyeznek, az ikon láthatatlan marad, és a játékos még nem párosította az összes fennmaradó ikont. Ebben az esetben a program egy return utasítást használ a metódus többi részének átlépésére. Ha a ciklus az összes címkén végigmegy a return kijelentés végrehajtása nélkül, az azt jelenti, hogy az űrlap összes ikonjának meg lett a párja. A program megjelenít egy MessageBox-ot (Ablaküzenet-et), amelyben gratulál a játékosnak a győzelemhez, majd meghívja az űrlap Close() (Bezárás) utasítást a játék befejezéséhez.

1. Majd helyezzük el a címke [Click](https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.control.click) eseménykezelőjében az új CheckForWinner() metódust. Ellenőrizd, hogy a program azonnal megvizsgálja a nyerést, miután megjelenik a játékos által választott második ikon. Keresd meg azt a sort, amelyen beállítja a második kiválasztott ikon színét, majd hívd meg a CheckForWinner()metódust, a következő kód szerint.

// Amikor a játékos ide elér, a stopper nem számol

// és a firstClicked értéke nem null,

// tehát a második ikonra klikkelt a játékos

// Beállítja a színét feketére

secondClicked = clickedLabel;

secondClicked.ForeColor = Color.Black;

// Ellenőrzi, hogy a játékos nyert e

CheckForWinner();

// Ha a játékos két egyező ikonra klikkel, akkor feketén maradnak

// és visszaállítja a firstClicked és a secondClicked értékeit,

// hogy egy másik ikonra kattinthasson

if (firstClicked.Text == secondClicked.Text)

{

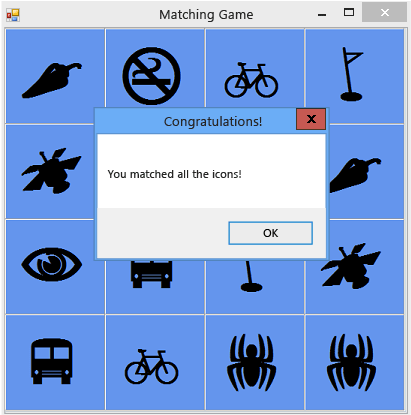
firstClicked = null;

secondClicked = null;

return;

}

1. Mentsd el és futtasd a programot. Játszd a végig a játékot és találd meg az összes ikont. Amikor nyersz, a program megjelenít egy gratuláló **MessageBox-ot** (ahogy a következő képernyőképen látható), majd bezárja az ablakot.

  
***Memóriajáték*** a ***MessageBox-***al