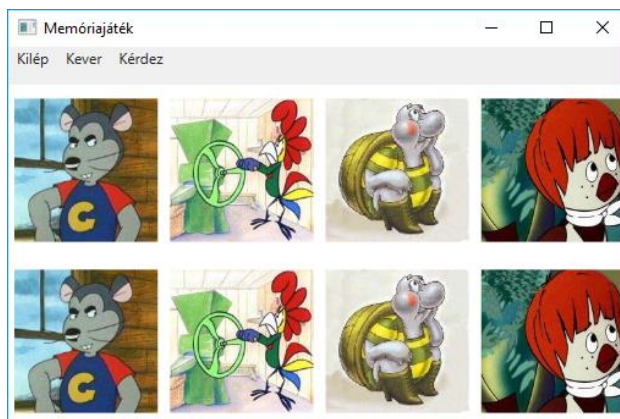
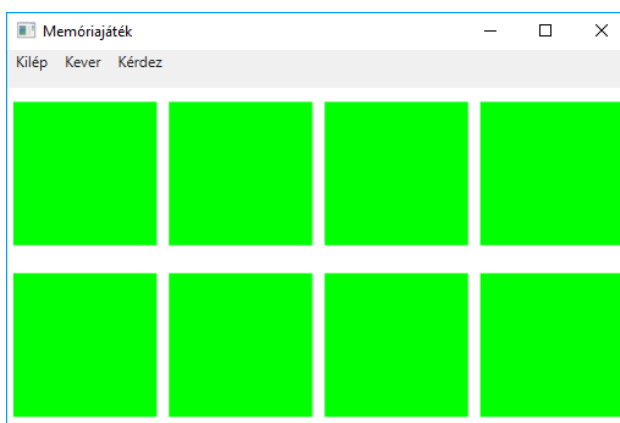


## Vizuális programozás gyakorlat

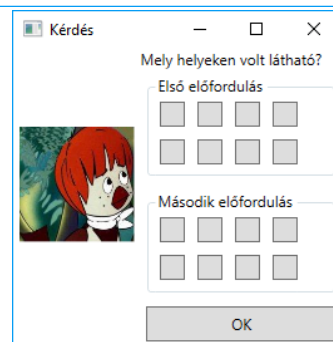
Készítsen egy Windows Presentation Foundation alkalmazást, ami a közismert, képeken alapuló memóriajáték egy egyszerű változatát valósítja meg. A program funkcionalitása a következő:



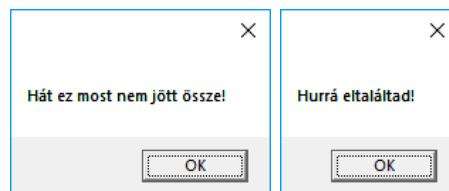
A **Kever** menüpont hatására az induláskor automatikusan betöltött négy pár képet véletlenszerű elrendezésben megjeleníti. Várakozik két másodpercet, majd a képek helyett zöldre festett téglalapokat jelenít meg (hatter.jpg).



A **Kérdez** menüpont hatására egy új ablakot jelenít meg *Kérdés* fejléccel. Az ablakban az előzőekben látható négy képtípus valamelyike jelenik meg véletlenszerűen. A felhasználó kiválasztja, hogy szerint mely helyeken fordult elő a főablakban a kép, majd kattint az OK gombon. Egy lehetséges megoldást mutat a jobb oldali ábra.



Az OK gombon történő kattintást követően eltűnik a Kérdés ablak, és a válasz helyességétől függően a mellékelt két üzenetablak egyike jelenik meg, majd újból láthatóvá válik a nyolc kép úgy, ahogy azt a legelső ábrán láthatjuk.



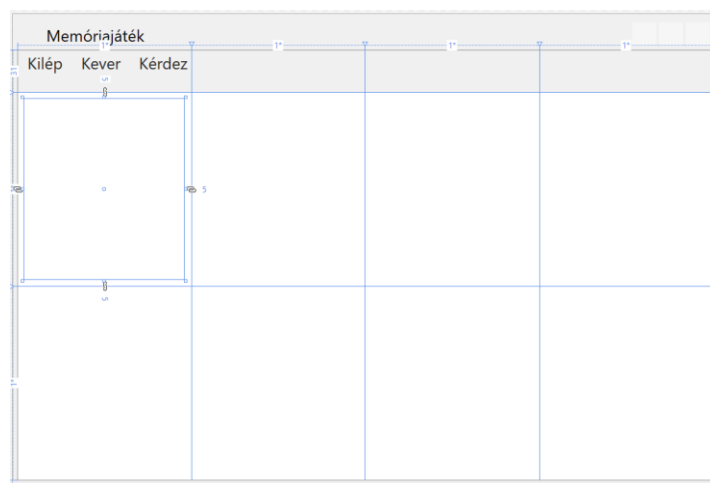
## Tanácsok a megvalósításhoz

A játékhoz szükséges négy kép és a háttérkép lemásolható a `t:\info\Johanyák Csaba\Rajzfilmfigurak\` címről.

Hozzunk létre egy WPF alkalmazást úgy, hogy a megoldás és a projekt neve legyen *Memoriajatek*.

### A felület kialakítása

Az ablak osztályát nevezzük át *wndFőablakra*, míg annak állományneve legyen *wndFoablak* alakú. Az ablak fejlécében a Memóriajáték felirat jelenjen meg. A menü és alatta képek egy három soros és négy oszlopos rácsban legyenek elhelyezve. A menü három pontot tartalmazzon az alábbi ábrának megfelelően.



Minden menüponthoz generáltassunk egy *Click* eseménykezelő vázat a Visual Studioval. A rács második és harmadik sorának minden cellájába egy *Image* komponens kerüljön úgy, hogy egy 5 pontos margó kihagyásával töltsse ki a cellát. Mivel mind a nyolc komponensnél ugyanazt a három tulajdonság értéket állítjuk be, ezért érdemes azt nem komponens szinten megtenni, hanem stílusdefiniáció segítségével az őket tartalmazó rács szintjén megadni. A felületet leíró XAML kód az alábbi.

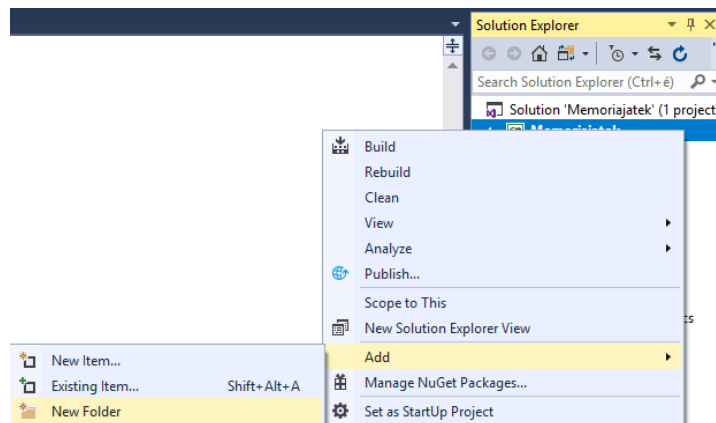
```
<Window x:Class="Memoriajatek.wndFőablak"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:Memoriajatek"
    mc:Ignorable="d"
    Title="Memóriajáték" Height="350" Width="525">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="31"/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
    </Grid>
```

```

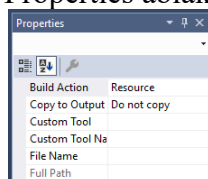
        </RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.Resources>
        <Style TargetType="{x:Type Image}">
            <Setter Property="HorizontalAlignment" Value="Stretch" />
            <Setter Property="VerticalAlignment" Value="Stretch" />
            <Setter Property="Margin" Value="5" />
        </Style>
    </Grid.Resources>
    <Menu Grid.ColumnSpan="4"
        Grid.Row="0" Grid.Column="0"
        HorizontalAlignment="Stretch" >
        <MenuItem Header="Kilép" x:Name="miKilép"
            Click="miKilép_Click"/>
        <MenuItem Header="Kever" x:Name="miKever"
            Click="miKever_Click"/>
        <MenuItem Header="Kérdez" x:Name="miKérdez"
            Click="miKérdez_Click"/>
    </Menu>
    <Image x:Name="im10" Grid.Row="1" Grid.Column="0" />
    <Image x:Name="im11" Grid.Row="1" Grid.Column="1" />
    <Image x:Name="im12" Grid.Row="1" Grid.Column="2" />
    <Image x:Name="im13" Grid.Row="1" Grid.Column="3" />
    <Image x:Name="im20" Grid.Row="2" Grid.Column="0" />
    <Image x:Name="im21" Grid.Row="2" Grid.Column="1" />
    <Image x:Name="im22" Grid.Row="2" Grid.Column="2" />
    <Image x:Name="im23" Grid.Row="2" Grid.Column="3" />
</Grid>
</Window>

```

A felhasznált képek számára hozzunk létre egy mappát a Solution Explorerben a projekten belül.



A mappa neve legyen Képek. A Képek mappába másoljuk be az öt képet (pl. Windows Fájlkészeltővel), majd vegyük fel őket a projekt Képek mappájába Solution Explorerben is. Jelöljük ki a képeket, és nézzük meg a Properties ablakban a hozzájuk tartozó beállításokat.



Ez azt jelenti, hogy erőforrásként lettek megjelölve, és a fordító be fogja fordítani őket a szerelvénybe.

Következő lépésként létrehozuk az ablak osztályában (*wndFőablak*) a programhoz szükséges adatokat. Elsőként a hátlapkép nevét tároló változót készítjük el.

```

/// <summary>
/// A hátlapként alkalmazott kép állomány neve.
/// </summary>
string HátlapNév = "hatter.jpg";

```

Hozzunk létre egy adattagot a négy rajfilmfigura képnevének tárolására.

```

/// <summary>
/// A játékban felhasznált mesefigurákat tartalmazó
/// kép állományok nevei.
/// </summary>
string[] KépNevek = { "grabowski.jpg", "kukori.jpg",
                      "kuldonc.jpg", "vili.jpg"
};

```

Hozzunk létre egy adattagot a hátlapkép objektum referenciájának tárolásához.

```

/// <summary>
/// A hátlapkép referenciáját tároló változó.
/// </summary>
BitmapImage biHátlapKép;

```

Hozzunk létre egy adattagot, aminek segítségével egy tömbben tudjuk tárolni a nyolc kép referenciáit.

```

/// <summary>
/// Tömb a nyolc kép objektum referenciáinak tárolására.
/// </summary>
BitmapImage[] biKépek = new BitmapImage[8];

```

Hozzunk létre egy adattagot a mesefigurák megjelenítésére használt Image komponensek referenciáinak tárolására.

```

/// <summary>
/// Tömb a mesefigurák megjelenítésére használt Image
/// komponensek referenciáinak tárolására.
/// </summary>
Image[] imKépHelyek;

```

Hozzunk létre egy adattagot a véletlenszám generáló objektum számára.

```

/// <summary>
/// Véletlenszámok előállítására szolgáló objektum.
/// </summary>
Random Véletlen = new Random();

```

A játék lényege, hogy a képeket egy rövid ideig megmutatjuk a felhasználónak, majd a hátlapképeket jelenítjük meg. Ehhez egy olyan időzítőre lesz szükségünk, ami lehetővé teszi, hogy a felületet kezelő számban hajtsuk végre a képcserét.

```

/// <summary>
/// Időzítő a képek memorizálására adott idő követéséhez.
/// </summary>
private DispatcherTimer dtIdőzítő;

```

A főablak konstruktorában hozzuk létre a képeket megjelenítő komponensek tömbjét, és töltjük be a képeket a memóriába, majd jelenítsük meg őket 3 másodperc időtartamra. Ezt követően állítsuk be, hogy alapból minden komponens a hátlapképet mutassa.

```
/// <summary>
/// A főablak konstruktora. Létrehozza a képeket megjelenítő
/// komponensek tömbjét, és betölti a képeket a memóriába.
/// </summary>
public wndFőablak()
{
    // Komponensek létrehozása és inicializálása.
    InitializeComponent();
    // Képeket megjelenítő komponensek tömbjének létrehozása.
    imKéphelyek = new Image[]
    {
        im10,im11,im12,im13,
        im20,im21,im22,im23
    };
    // Időzítő objektum létrehozása, az időzítés beállítása 3 másodpercre
    dtIdőzítő = new DispatcherTimer
    {
        Interval = new TimeSpan(0, 0, 0, 0, 3000),
        IsEnabled = false
    };
    dtIdőzítő.Tick += dtIdőzítő_Tick;
    // Képek betöltése a memóriába.
    KépeketBetölt();
    // Képek láthatóvá tétele.
    KépeketMegmutat();
    // Minden komponens a hátlapképet mutatja, ha letelt a beállított
    // időintervallum.
    dtIdőzítő.Start();
}
```

Az időzítő eseménykezelője (dtIdőzítő\_Tick()), a KépeketBetölt() és a KépeketMegmutat() metódusok még nincsenek készen, ezért először készítsünk hozzájuk egy üres vázat, majd folytassuk a munkát a KépeketBetölt() metódus törzsének megírásával.

```
/// <summary>
/// Az időzítés leteltével a hátlapképet helyezi el mind a nyolc kép komponensre,
/// majd leállítja az időzítőt.
/// </summary>
private void dtIdőzítő_Tick(object sender, EventArgs e)
{
    HátlapképpelKitölt();
    dtIdőzítő.Stop();
}
```

A képek betöltéséhez ún. Pack Uri (ld. <http://msdn.microsoft.com/en-us/library/aa970069.aspx>) típusú erőforrás elérés használata szükséges. A relatív útvonalaknál elegendő a projekt gyökérkönyvtárhoz viszonyítva megadni az erőforrás elérési útvonalát. elsőként a hátlapképet, majd ezt követően egy ciklusban a rajzfilmfigurák képeit fogjuk betölteni. A később szükséges keverés könnyebb megoldása érdekében minden betöltött rajzfilmfigura kép referenciáját két tömb elemében tároljuk, így a négy képhez egy nyolcelemű tömböt használunk.

```

/// <summary>
/// Betölti memóriába a játékhoz használt képeket. A képek a
/// projekt gyökérkönyvtárában levő Kepek alkönyvtárában kell legyenek.
/// </summary>
private void KépeketBetölt()
{
    try
    {
        // Hátlapkép betöltése.
        biHátlapKép = new BitmapImage(new Uri(@"Kepek/" + HátlapNév, UriKind.Relative));
        // A négy mesefigura kép betöltése.
        for (int i = 0; i < 4; i++)
        {
            biKépek[i] = new BitmapImage(new Uri(@"Kepek/" + KépNevek[i],
                UriKind.Relative));
            // A másodpéldányokat beazonosító referenciák.
            biKépek[i + 4] = biKépek[i];
        }
    }
    catch (Exception)
    {
        MessageBox.Show("A képek nem találhatók a megadott útvonalon!",
            "Hiba", MessageBoxButton.OK);
    }
}

```

A hátlapképeket megmutató metódus tartalmát alakítsuk ki az alábbiak szerint.

```

/// <summary>
/// Mind a nyolc helyen a hátlapképet jeleníti meg.
/// </summary>
private void HátlapképpelKitölt()
{
    for (int i = 0; i < 8; i++)
    {
        imKépHelyek[i].Source = biHátlapKép;
    }
}

```

A képek láthatóvá tétele (KépeketMegmutat() metódus) azt jelenti, hogy a biKépek tömbben hivatkozott sorrendben rendeljük a képeket az egyes Image komponensekhez. A metódus működése hasonló a HátlapképpelKitölt() metóduséhoz.

```

/// <summary>
/// Láthatóvá teszi a nyolc képet.
/// </summary>
private void KépeketMegmutat()
{
    for (int i = 0; i < 8; i++)
    {
        imKépHelyek[i].Source = biKépek[i];
    }
}

```

Próbáljuk ki a programot úgy, hogy egyszer a rajzfilmfigurákat és egyszer a hátlapképek megmutatását kapcsoljuk be a konstruktorban. Ha minden jól megy, akkor továbbléphetünk a a Kever menüpont eseménykezelőjének elkészítéséhez. Ebben az induláskor automatikusan betöltött négy pár képet véletlenszerű elrendezésben jelenítjük meg, várakozunk három másodpercet, majd a képek helyett zöldre festett téglalapokat jelenítünk meg (hátlap).

```

/// <summary>
/// A Kever menüpont hatására az induláskor automatikusan
/// betöltött négy pár képet véletlenszerű elrendezésben
/// megjeleníti. Várakozik, majd a képek helyett zöldre
/// festett téglalapokat jelenít meg (hatter.jpg).
/// </summary>
/// <param name="sender">A menüpont objektum.</param>
/// <param name="e"></param>
private void miKever_Click(object sender, RoutedEventArgs e)
{
    // Véletlen sorrend meghatározása.
    VéletlenSorrendbeRak();
    // Képek láthatóvá tétele.
    KépekMegmutat();
    // Várakozás az időzítőben beállított ideig, majd a hátlapképek megjelenítése.
    dtIdőzítő.Start();
}

```

A képek elhelyezkedését meghatározó VéletlenSorrendbeRak() metódusban létrehozunk egy generikus lista objektumot, amibe betesszük az összes képet. Innen véletlenszerűen kihúзва képeket alakítjuk ki a biKépek tömb tartalmát.

```

/// <summary>
/// Meghatározza a képek véletlenszerű sorrendjét.
/// </summary>
private void VéletlenSorrendbeRak()
{
    // Létrehozunk egy lista objektumot a képek referenciáinak tárolására.
    List<BitmapImage> KépLista = new List<BitmapImage>();
    // Mind a 8 kép referenciáját elhelyezzük a listában.
    KépLista.AddRange(biKépek);
    // "Húzás a kalapból" véletlenszerűen kivesszük a nyolc referenciát.
    for (int i = 0; i < 8; i++)
    {
        // A maradék listából véletlenszerűen kiválasztunk egy elemet.
        int Sorszám = Véletlen.Next(0, KépLista.Count);
        // Betesszük a tömb i. helyére
        biKépek[i] = KépLista[Sorszám];
        // Eltávolítjuk a listáról.
        KépLista.RemoveAt(Sorszám);
    }
}

```

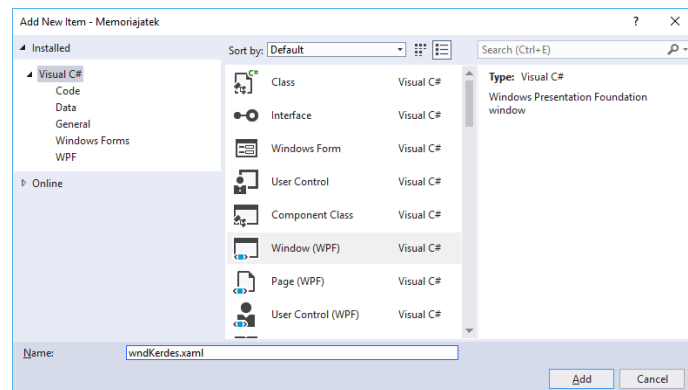
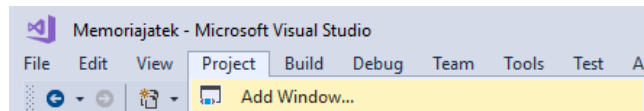
Készítsük el a **Kilép** menüpont eseménykezelőjének kódját. A korábban legenerált vázba csak egyetlen utasítást kell beírni.

```

/// <summary>
/// Kilép az alkalmazásból.
/// </summary>
/// <param name="sender">A Kilép menüpont objektum.</param>
/// <param name="e"></param>
private void miKilép_Click(object sender, RoutedEventArgs e)
{
    Application.Current.Shutdown();
}

```

A Kérdez menüpont eseménykezelőjének elkészítése előtt készítünk egy új ablakot a projektünkben wndKérdés néven (az állománynév wndKerdes legyen).

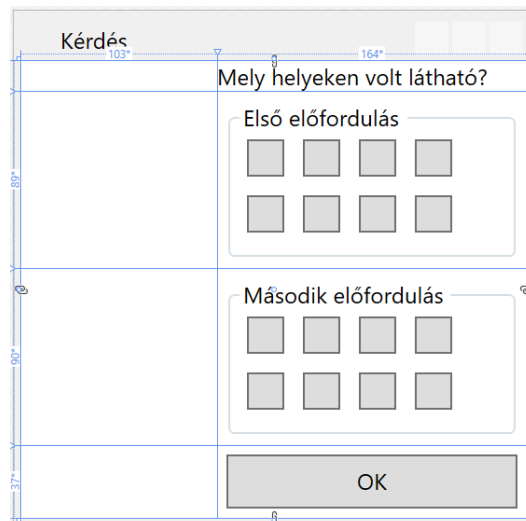


Nevezzük át az ablak osztályát wndKérdés-re és a fejlécben helyezzük el a Kérdés feliratot

```
<Window x:Class="Memoriajatek.wndKérdés"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:Memoriajatek"
        mc:Ignorable="d"
        Title="Kérdés" Height="276.5" Width="279">
    <Grid>

    </Grid>
</Window>
```

Az ablak felépítése az alábbi ábra szerinti lesz.



A rácsban kialakítunk két oszlopot és négy sort. Az első oszlop középső két cellájába elhelyezünk egy Image komponenst. A csoportablakokban egy-egy WrapPanel fogja tárolni a kiválasztáshoz szükséges komponenseket. A WrapPanel szélességének megfelelő megválasztásával érjük el, hogy ezek két sorban jelenjenek meg az ábra szerint. Mivel mindkét esetben a nyolc komponensből egyszerre csak egy lehet kiválasztva, ezért nem nyomógombot, hanem RadioButton komponenst használunk úgy, hogy megjelenése a ToggleButton (kétállapotú gomb) komponenshez legyen hasonló. A gépelési munka



csökkentése és a kód átláthatóságának növelése érdekében a 16 RadioButton közös tulajdonságait stílusbeállítások segítségével adjuk meg a rács szintjén.

```
<Window x:Class="Memoriajáték.wndKérdés"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:Memoriajáték"
    mc:Ignorable="d"
    Title="Kérdés" Height="276.5" Width="279">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="15*" />
        <RowDefinition Height="89*" />
        <RowDefinition Height="90*" />
        <RowDefinition Height="37*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="103*" />
        <ColumnDefinition Width="164*" />
    </Grid.ColumnDefinitions>
    <Grid.Resources>
        <Style BasedOn="{StaticResource {x:Type ToggleButton}}"
            TargetType="RadioButton">
            <Setter Property="Width" Value="20" />
            <Setter Property="Height" Value="20" />
            <Setter Property="Margin" Value="5" />
        </Style>
    </Grid.Resources>
    <Image Grid.Row="1" Grid.Column="0" Grid.RowSpan="2" Margin="5"
        VerticalAlignment="Stretch" HorizontalAlignment="Stretch"
        x:Name="imKép" />
    <TextBlock Grid.Row="0" Grid.Column="1">
        Mely helyeken volt látható?
    </TextBlock>
    <GroupBox Grid.Row="1" Grid.Column="1"
        x:Name="gbElső" Header="Első előfordulás" Margin="5">
        <WrapPanel Orientation="Horizontal" x:Name="wpElső" >
            <RadioButton x:Name="rbElső00" />
            <RadioButton x:Name="rbElső01" />
            <RadioButton x:Name="rbElső02" />
            <RadioButton x:Name="rbElső03" />
            <RadioButton x:Name="rbElső10" />
            <RadioButton x:Name="rbElső11" />
            <RadioButton x:Name="rbElső12" />
            <RadioButton x:Name="rbElső13" />
        </WrapPanel>
    </GroupBox>
    <GroupBox Grid.Row="2" Grid.Column="1"
        x:Name="gbMásodik" Header="Második előfordulás" Margin="5">
        <WrapPanel Orientation="Horizontal" x:Name="wpMásodik">
            <RadioButton x:Name="rbMásodik00" />
            <RadioButton x:Name="rbbMásodik01" />
            <RadioButton x:Name="rbbMásodik02" />
            <RadioButton x:Name="rbbMásodik03" />
            <RadioButton x:Name="rbbMásodik10" />
            <RadioButton x:Name="rbbMásodik11" />
            <RadioButton x:Name="rbbMásodik12" />
            <RadioButton x:Name="rbbMásodik13" />
        </WrapPanel>
    </GroupBox>
</Grid>
```

```

        <Button x:Name="btOK" Content="OK" Grid.Row="3"
                Grid.Column="1" Click="btOK_Click" Margin="5" />
    </Grid>
</Window>

```

Térjünk vissza a főablakhoz, és készítsük el a Kérdez menüpont eseménykezelőjének kódját. A Kérdez menüpont hatására a program egy új ablakot jelenít meg Kérdés fejléccel. Az ablakban az előzőekben látható négy képtípus valamelyike jelenik meg véletlenszerűen. A felhasználó kiválasztja, hogy szerint mely helyeken fordult elő a főablakban a kép, majd kattint az OK gombon. Ekkor eltűnik a Kérdez ablak, és a válasz helyességétől függően az alábbi két üzenetablak egyike jelenik meg, majd újból láthatóvá válik a nyolc kép úgy, ahogy azt a legelső ábrán láthatjuk.

```

/// <summary>
/// A Kérdez menüpont hatására egy új ablakot jelenít meg
/// Kérdés fejléccel. Az ablakban az előzőekben látható
/// négy képtípus valamelyike jelenik meg véletlenszerűen.
/// A felhasználó kiválasztja, hogy szerint mely helyeken fordult elő a
/// főablakban a kép, majd kattint az OK gombon.
/// Ekkor eltűnik a Kérdez ablak, és a válasz helyességétől
/// függően az alábbi két üzenetablak egyike jelenik meg,
/// majd újból láthatóvá válik a nyolc kép úgy, ahogy azt a
/// legelső ábrán láthatjuk.
/// </summary>
/// <param name="sender">A Kérdez menüpont objektum.</param>
/// <param name="e"></param>
private void miKérdez_Click(object sender, RoutedEventArgs e)
{
    // Ablak objektum létrehozása.
    wndKérdés wndKérdés = new wndKérdés();
    // Kép sorszám kiválasztása véletlenszerűen.
    int Sorszám = Véletlen.Next(0, 8);
    // Keresett kép kiválasztása.
    BitmapImage KeresettKép = biKépek[Sorszám];
    // Kép komponenshez rendelése.
    wndKérdés.biKép = KeresettKép;
    // Megjeleníti a párbeszédablakot.
    if (wndKérdés.ShowDialog()==true)
    {
        // Lekérdezi, hogy mely pozíciókat választotta ki a
        // felhasználó, majd előállítja az adott pozícióban található
        // képek referenciáit.
        BitmapImage biElső = biKépek[wndKérdés.ElsőSorszám];
        BitmapImage biMásodik = biKépek[wndKérdés.MásodikSorszám];
        // Megvizsgálja, hogy a keresett kép azonos-e a két megjelölt
        // képpel.
        if (biElső == KeresettKép && biMásodik == KeresettKép)
            MessageBox.Show("Hurrá eltaláltad!");
        else
            MessageBox.Show("Hát ez most nem jött össze!");
        // Újból láthatóvá teszi a képeket.
        KépekMegmutat();
    }
}

```

A metódusban a megjelenített kép megadásához (wndKérdés.biKép) és a felhasználó által kiválasztott pozíciók lekérdezéséhez (wndKérdés.ElsőSorszám, wndKérdés.MásodikSorszám) olyan tulajdonságokat használtunk, amelyek még nem léteznek a wndKérdés osztályban, ezért

következő lépésként ezeket létre kell hoznunk. A biKép esetében csak írható tulajdonság elegendő.

```
/// <summary>
/// Lehetővé teszi a párbeszédablakban megjelenített kép beállítását.
/// </summary>
public BitmapImage biKép
{
    set { imKép.Source = value; }
}
```

A másik két esetben csak olvasható tulajdonságokra van szükség, amelyek -1 -es értéket adnak vissza, amennyiben a felhasználó nem választott egyetlen RadioButton-t se az adott csoportból, egyéb esetben pedig visszaadják a kiválasztott RadioButton sorszámát (0..7). A feladatot úgy oldjuk meg, hogy végiglépkedünk a WrapPanelben található komponenseken minden gyermek komponenst RadioButtonként értelmezve, és lekérdezzük az állapotukat.

```
/// <summary>
/// -1 -es értéket ad vissza, amennyiben a felhasználó nem választott egyetlen
/// RadioButton-t se az első csoportból, egyéb esetben pedig visszaadja a
/// kiválasztott RadioButton sorszámát (0..7).
/// </summary>
public int ElsőSorszám
{
    get
    {
        int ssz = -1;
        for (int i = 0; i < wpElső.Children.Count; i++)
        {
            if ((wpElső.Children[i] as RadioButton)?.IsChecked==true)
            {
                ssz = i;
            }
        }
        return ssz;
    }
}

/// <summary>
/// -1 -es értéket ad vissza, amennyiben a felhasználó nem választott egyetlen
/// RadioButton-t se a második csoportból, egyéb esetben pedig visszaadja a
/// kiválasztott RadioButton sorszámát (0..7).
/// </summary>
public int MásodikSorszám
{
    get
    {
        int ssz = -1;
        for (int i = 0; i < wpMásodik.Children.Count; i++)
        {
            if ((wpMásodik.Children[i] as RadioButton)?.IsChecked == true)
            {
                ssz = i;
            }
        }
        return ssz;
    }
}
```

Utolsó lépésként elkészítjük az OK gomb eseménykezelőjét. itt az a feladat, hogy ellenőrizzük, hogy a felhasználó választott-e mindkét csoportból. A választás hiányában nem engedjük bezárulni a párbeszédablakot, és hibaüzenetet jelenítünk meg.

```
/// <summary>
/// OK gombon történő kattintás esetén megvizsgálja, hogy van-e kiválasztott.
/// RadioButton Ha nincs, akkor hibaüzenettel választásra készíti a
/// felhasználót, ha van, akkor engedélyezi a párbeszédablak lezárását.
/// </summary>
private void btOK_Click(object sender, RoutedEventArgs e)
{
    if (ElsőSorszám != -1 && MásodikSorszám != -1)
        DialogResult = true;
    else
        MessageBox.Show("Először válaszd ki a két pozíciót!");
}
```

Amennyiben a felhasználó mégsem akar választ adni, akkor a párbeszédablakot a jobb felső sarokban levő bezáró gombra kattintva zárhatja. Ilyenkor az ablakot megjelenítő ShowDialog() metódus visszatérési értéke false lesz.