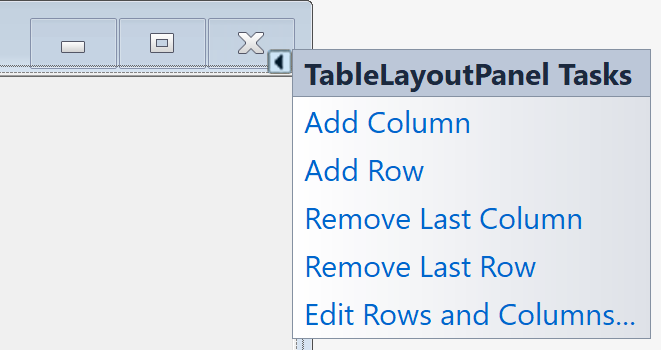
**Windows Form – Matching Game**

* Create new Windows Form project in Visual Studio called MatchingGame
* Change the application title bar name to ‘Matching Game’
  + Change the *Text* property to ‘Matching Game’
* Change the application window size
  + Change the *Size* property to 1100, 1100
* Add a *TableLayoutPanel* to our form
  + Set the *BackColor* to *CornflowerBlue*
    - Select the *Web* tab at the top when you press the drop-down
    - *CornflowerBlue* is towards the bottom of the list
  + Set the *Dock* property to *Fill*
* Set the *CellBorderStyle* to Inset (provides a visual border to the cells
* Add two rows and two columns to the *TableLayoutPanel*
  + Click the triangle on the *TableLayoutPanel* to show the *Add Column* and *Add Row*
  + Click each one twice to add two more columns and rows



* Select *Edit Rows and Columns…* and set the percentage to all columns and rows to 25%
* With the *TableLaoutPanel*selected add a *Label* control
  + Make sure the *BackColor* is *CornflowerBlue*
  + Set *AutoSize* to *FALSE*
  + Set *Dock* to *Fill*
  + Set the *TextAlign* to *MiddleCenter*
  + Set *Font* to *Webdings Bold 72*
    - Click the (…) button next to the current font setting
  + Set *Text* to ‘c’
* Copy (Ctrl+C) the label to the other cells (Ctrl+V)
  + The paste will be done 15 more times to fill all the cells
* In the *Solution Explorer* right-click on *Form1.cs* and select *View Code*
  + Can also use Ctrl+Alt+0
* Add the code to add a *Random* object, this will be used to retrieve a random number

public partial class Form1 : *Form*

{

*Random* random = new *Random*();

* Add the code to define the matching icons text for the game

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

{

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

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

};

* Add the code to set the Label controls text to the icons text

/// <summary>

/// Assign each icon from the list to a random square

/// </summary>

private void AssignIconsToSquares()

{

// iterate through all controls in the tableLayoutPanel1

foreach (*Control* control in tableLayoutPanel1.*Controls*)

{

*Label* iconLabel = control as *Label*;

//Label iconLabel = (Label)control;

if (iconLabel != null)

{

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

iconLabel.*Text* = icons[randomNumber];

iconLabel.*ForeColor* = iconLabel.*BackColor*;

icons.*RemoveAt*(randomNumber);

}

}

}

* Call the method to set the Label control text at the start of the game

public Form1()

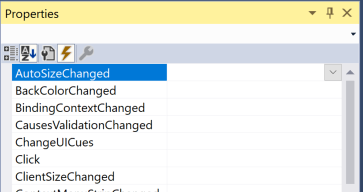
{

InitializeComponent();

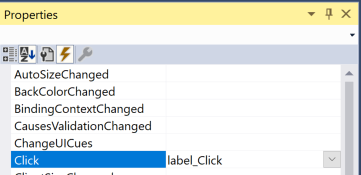
AssignIconsToSquares();

}

* Open the Form1.cs designer
  + right-click on *Form1.cs* in the *Solution Explorer* and select *View Code*
  + Can also use Shift+F7
* Select all the Label controls
  + Hold Ctrl and click on all the Labels
* Set the event for all the Label controls
  + Click the Lightning Bolt Icon in Properties to list all the events



* + Change the *Click* value to ‘label\_Click’
    - This will create a click event in code



* Add the code to handle the Click event

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

{

*Label* clickedLabel = sender as *Label*;

if (clickedLabel != null)

{

// Return if the label text/forecolor is black (already clicked)

if (clickedLabel.*ForeColor* == *Color*.*Black*)

return;

// Set the label forecolor to black

clickedLabel.*ForeColor* = *Color*.*Black*;

}

}

* Add the code to track the Label clicked
  + The first and second Label clicked

// Track the two labels being clicked

*Label* firstClick = null;

*Label* secondClick = null;

* Update the code to track the firsl Label clicked

if (clickedLabel.*ForeColor* == *Color*.*Black*)

return;

if (firstClick == null)

{

firstClick = clickedLabel;

// Set the label forecolor to black

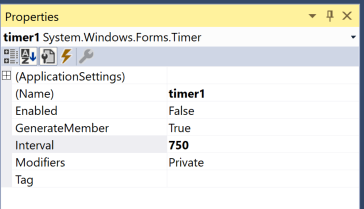
firstClick.*ForeColor* = *Color*.*Black*;

}

* Add a *Timer* component to the Application
  + Components show up at the bottom of the Designer window



* + Change the *Interval* property to 750
    - Properties may still be set to Events (Lightning Bolt) and may need to be set back to the property display
      * It’s the button with the wrench and page image



* Select the *Timer* component on the Designer window and press enter to create the event code
* Add the code to hide the icons and reset the click labels

/// <summary>

/// The timers starts when the player clicks two non-matching icons

/// Once the interval has expired the icons are hidden

/// </summary>

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

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

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

{

// Stop the timer

timer1.*Stop*();

// Hide both icons

firstClick.*ForeColor* = firstClick.*BackColor*;

secondClick.*ForeColor* = secondClick.*BackColor*;

// Reset firstClick and secondClick

firstClick = null;

secondClick = null;

}

* Update the code for the click event to handle both clicks and the timer

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

{

// If the timer is running then don't allow any more clicks until it is reset

if (timer1.*Enabled* == true)

return;

*Label* clickedLabel = sender as *Label*;

if (clickedLabel != null)

{

// Return if the label text/forecolor is black (already clicked)

if (clickedLabel.*ForeColor* == *Color*.*Black*)

return;

// Set the first click label if not set yet

if (firstClick == null)

{

firstClick = clickedLabel;

// Set the label forecolor to black to 'reveal' the icon

firstClick.*ForeColor* = *Color*.*Black*;

return;

}

// Set the second click label

secondClick = clickedLabel;

secondClick.*ForeColor* = *Color*.*Black*;

// Start the timer

timer1.*Start*();

}

}

* Add the code to check for a matching pair

// Set the second click label

secondClick = clickedLabel;

secondClick.*ForeColor* = *Color*.*Black*;

// Check for a match, compare the text of the two labels

if (firstClick.*Text* == secondClick.*Text*)

{

firstClick = null;

secondClick = null;

return;

}

* Add the code to check for a winner

/// Check all the labels to see if there forecolor != backcolor

/// If the forecolor doesn't equal the back color then the card has been revealed

/// If all labels don't equal the backcolor then the player is a winner

/// </summary>

private void CheckForWinner()

{

// Iterate through all controls in the tableLayoutPanel

foreach (*Control* control in tableLayoutPanel1.*Controls*)

{

// Check if the control is a Label

*Label* iconLabel = control as *Label*;

if (iconLabel != null)

{

// Check if any labels have the forecolor set to it's backcolor

// If so then the game is not over

if (iconLabel.*ForeColor* == iconLabel.*BackColor*)

return;

}

}

// If all the labels are flipped, the player has won

// Show winner message box

*MessageBox*.*Show*("You matched all the icons!", "Congratulations");

// Close the application

*Close*();

}

* Check for the winner in the click event

// Set the second click label

secondClick = clickedLabel;

secondClick.*ForeColor* = *Color*.*Black*;

CheckForWinner();

* **The application should now be complete. Run it and play the game.**

**Bonus**

* Replace the icons and colors with ones you choose.
* Add a game timer that limits how long the player has to complete the game
  + Add another Timer component
    - Have it start when the game starts
      * In code or setting the Enabled property
    - Add an event to the game timer that will display a game over message box and close the application
* Add sounds to the game
  + Selection Match
  + Selection Non-Match
  + Win Game
  + Short video on playing sounds: <https://www.youtube.com/watch?v=qOh4ooHg1UU&feature=youtu.be>
  + Here’s some of the code that might be required

using *System*.*Media*;

*SoundPlayer* soundPlayer = null;

soundPlayer = new *SoundPlayer*("*name.wav*");

soundPlayer.*Play*();

soundPlayer.*Stop*();

* Make the game board bigger by adding more rows and columns