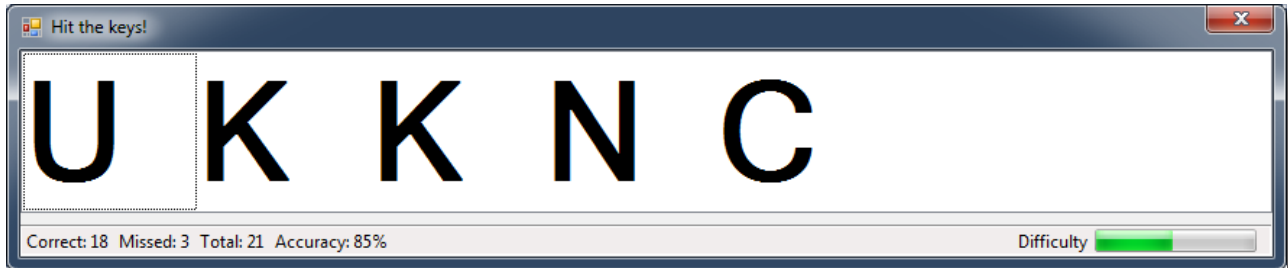


Build a typing game

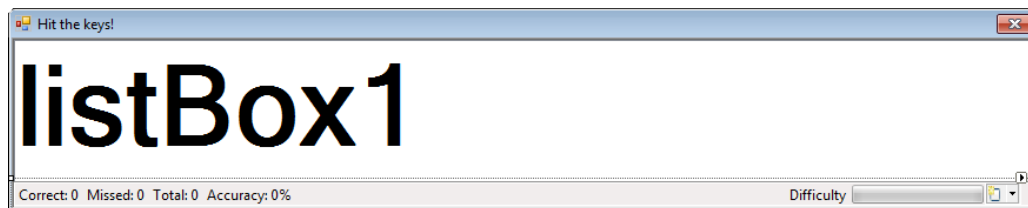
You've reached a milestone...you know enough to build a game! Here's how your game will work. The form will display random letters. If the player types one of them, it disappears and the accuracy rate goes up. If the player types an incorrect letter, the accuracy rate goes down. As the player keeps typing letters, the game goes faster and faster, getting more difficult with each correct letter. If the form fills up with letters, the game is over!



1

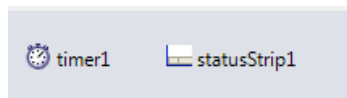
Build the form.

Here's what the form will look like in the form designer:



You'll need to:

- ★ Turn off the minimize box and maximize box. Then set the form's **FormBorderStyle** property to **Fixed3D**. That way, the player won't be able to accidentally drag and resize it. Then resize it so that it's much wider than it is tall (we set our form's size to 876, 174).
- ★ Drag a **Listbox** out of the Toolbox onto the form. Set its **Dock** property to **Fill**, and its **MultiColumn** property to **True**. Set its **Font** to 72 point bold.
- ★ In the Toolbox, expand the "All Windows Forms" group at the top. This will display many controls. Find the **Timer** control and double-click on it to add it to your form.
- ★ Find the **StatusStrip** in the "All Windows Forms" group in the Toolbox and double-click on it to add a status bar to your form. You should now see the **StatusStrip** and **Timer** icons in the gray area at the bottom of the form designer:



See how you can use a Timer to make your form do more than one thing at once? Take a minute and flip to #3 in the "Leftovers" appendix to learn about another way to do that.



You'll be using three new controls, but they're easy to work with!

Even though you haven't seen a `ListBox`, `StatusStrip`, or `Timer` before, you already know how to set their properties and work with them in your code. You'll learn a lot more about them in the next few chapters.

2 Set up the `StatusStrip` control.

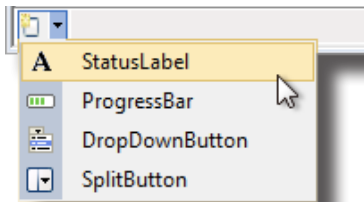
Take a closer look at the status bar at the bottom of the screenshot. On one side, it's got a series of labels:

Correct: 18 Missed: 3 Total: 21 Accuracy: 85%

And on the other side, it's got a label and a progress bar:

Difficulty

Add a `StatusLabel` to your `StatusStrip` by clicking its drop-down and selecting `StatusLabel`:



- ★ Set the `StatusStrip`'s `SizingGrip` property to `False`.
- ★ Use the Properties window to set its `(Name)` to `correctLabel` and its `Text` to "Correct: 0". Add three more `StatusLabel`s: `missedLabel`, `totalLabel`, and `accuracyLabel`.
- ★ Add one more `StatusLabel`. Set its `Spring` to `True`, `TextAlign` to `MiddleRight`, and `Text` to "Difficulty". Finally, add a `ProgressBar` and name it `difficultyProgressBar`.

3 Set up the `Timer` control.

Did you notice how your `Timer` control didn't show up on your form? That's because the `Timer` is a *non-visual control*. It doesn't actually change the look and feel of the form. It does exactly one thing: it **calls a method over and over again**. Set the `Timer` control's `Interval` property to 800, so that it calls its method every 800 milliseconds. Then **double-click on the `timer1` icon** in the designer. The IDE will do what it always does when you double-click on a control: it will add a method to your form. This time, it'll add one called `timer1_Tick`. Here's the code for it:

```
private void timer1_Tick(object sender, EventArgs e)
{
    // Add a random key to the ListBox
    listBox1.Items.Add((Keys)random.Next(65, 90));
    if (listBox1.Items.Count > 7)
    {
        listBox1.Items.Clear();
        listBox1.Items.Add("Game over");
        timer1.Stop();
    }
}
```

You'll add a field called "random" in just a minute. Can you guess what its type will be?



4 Add a class to keep track of the player stats.

If the form is going to display the total number of keys the player pressed, the number that were missed and the number that were correct, and the player's accuracy, then we'll need a way to keep track of all that data. Sounds like a job for a new class! Add a class called `Stats` to your project. It'll have four `int` fields called `Total`, `Missed`, `Correct`, and `Accuracy`, and a method called `Update` with one `bool` parameter: `true` if the player typed a correct letter that was in the `ListBox`, or `false` if the player missed one.

Stats
Total
Missed
Correct
Accuracy
Update()

```
class Stats
{
    public int Total = 0;
    public int Missed = 0;
    public int Correct = 0;
    public int Accuracy = 0;

    public void Update(bool correctKey)
    {
        Total++;

        if (!correctKey)
        {
            Missed++;
        }
        else
        {
            Correct++;
        }

        Accuracy = 100 * Correct / (Missed + Correct);
    }
}
```

Every time the `Update()` method is called, it recalculates the % correct and puts it in the `Accuracy` field.

5 Add fields to your form to hold a `Stats` object and a `Random` object.

You'll need an instance of your new `Stats` class to actually store the information, so add a field called `stats` to store it. And you already saw that you'll need a field called `random`—it'll contain a `Random` object.

Add the two fields to the top of your form:

```
public partial class Form1 : Form
{
    Random random = new Random();
    Stats stats = new Stats();
    ...
}
```



Before you go on, there are three properties you need to set. Set the `Timer` control's `Enabled` property to `True`, the `ProgressBar` control's `Maximum` property to 701, and the `Form`'s `KeyPreview` property to `True`. Take a minute and figure out why you need those properties. What happens if you don't set them?

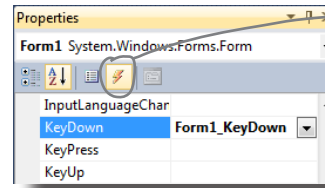




6 Handle the keystrokes.

There's one last thing your game needs to do: any time the player hits a key, it needs to check if that key is correct (and remove the letter from the ListBox if it is), and update the stats on the StatusStrip.

Go back to the form designer and select the form. Then go to the Properties window and click on the lightning bolt button. Scroll to the **KeyDown** row and **double-click on it**. This tells the IDE to add a method called `Form1_KeyDown()` that gets called every time the user presses a key. Here's the code for the method:



Click this button to change the Properties window's view. The button to the left of it switches the Properties window back to showing you properties.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    // If the user pressed a key that's in the ListBox, remove it
    // and then make the game a little faster
    if (listBox1.Items.Contains(e.KeyCode))
    {
        listBox1.Items.Remove(e.KeyCode);
        listBox1.Refresh();
        if (timer1.Interval > 400)
            timer1.Interval -= 10;
        if (timer1.Interval > 250)
            timer1.Interval -= 7;
        if (timer1.Interval > 100)
            timer1.Interval -= 2;
        difficultyProgressBar.Value = 800 - timer1.Interval;

        // The user pressed a correct key, so update the Stats object
        // by calling its Update() method with the argument true
        stats.Update(true);
    }
    else
    {
        // The user pressed an incorrect key, so update the Stats object
        // by calling its Update() method with the argument false
        stats.Update(false);
    }

    // Update the labels on the StatusStrip
    correctLabel.Text = "Correct: " + stats.Correct;
    missedLabel.Text = "Missed: " + stats.Missed;
    totalLabel.Text = "Total: " + stats.Total;
    accuracyLabel.Text = "Accuracy: " + stats.Accuracy + "%";
}
```

This if statement checks the ListBox to see if it contains the key the player pressed. If it does, then the key gets removed from the ListBox and the game difficulty is increased.

This is the part that increases the difficulty as the player gets more keys right. You can make the game easier by reducing the amounts that are subtracted from `timer1.Interval`, or make it harder by increasing them.

These are called events, and you'll learn a lot more about them later on.

When the player presses a key, the `Form1_KeyDown()` method calls the `Stats` object's `Update()` method to update the player stats, and then it displays them in the `StatusStrip`.

7 Run your game.

Your game's done! Give it a shot and see how well you do. You may need to adjust the font size of the ListBox to make sure it holds exactly 7 letters, and you can change the difficulty by adjusting the values that are subtracted from `timer1.Interval` in the `Form1_KeyDown()` method.

