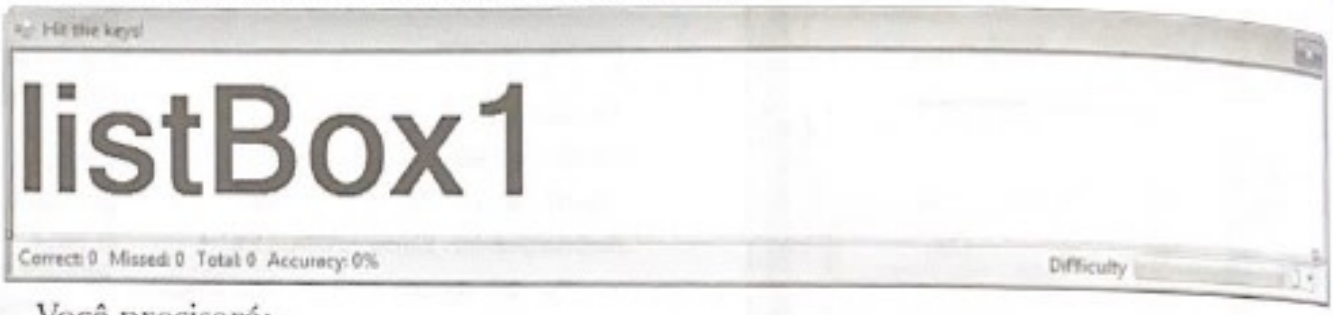
Desenvolva um jogo de digitação

Você atingiu uma quilometragem... sabe o suficiente para construir um jogo! Eis como seu jogo funcionará. O formulário exibirá letras aleatórias. Se o jogador digitar uma delas, ela desaparecerá e a taxa de precisão aumentará. Se o jogador digitar uma letra incorreta, a taxa de precisão diminuirá. Quando o jogador continua digitando letras, o jogo vai cada vez mais rapidamente, ficando mais difícil a cada letra correta. Se o formulário preencher as letras, a jogo terá acabado!

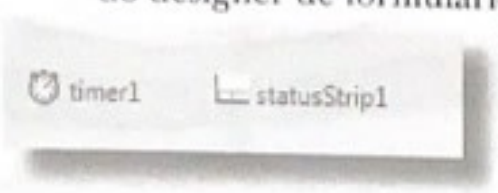
Faça isso!

1. **Desenvolva o formulário.**
Eis como ficará o formulário no formulário:



Você precisará:

- ★ Desativar as caixas minimizar e maximizar. Então, defina a propriedade **`FormBorderStyle`** do formulário para **`Fixed3D`**. Assim, o jogador não será capaz de arrastar e redimensionar sem querer. Então, redimensione-o para que fique muito mais largo do que alto (definimos o tamanho de nosso formulário para 876, 174).
- ★ Arrastar um **`ListBox`** de Toolbox (Caixa de Ferramentas) para o formulário. Defina sua propriedade **`Dock`** para Fill (Preencher) e sua propriedade **`MultiColumn`** para True. Defina sua **Font** para 72 pontos e negrito.
- ★ Em Toolbox, expanda o grupo "All Windows Forms" (Todos os Formulários Windows) no topo. Isto exibirá muitos controles. Encontre o controle **`Timer`** e clique duas vezes nele para adicioná-lo ao seu formulário.
- ★ Encontrar **`StatusStrip`** no grupo "All Windows Forms" em Toolbox e clicar duas vezes para adicionar uma barra de status ao seu formulário. Agora, você deverá ver os ícones **`StatusStrip`** e **`Timer`** na área cinza na parte inferior do designer de formulários:

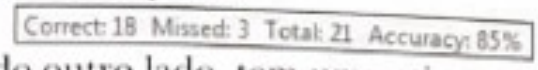


Veja como você pode usar Timer para que seu formulário faça mais de uma coisa por vez?

Você usará três controles novos, mas eles são fáceis de trabalhar!

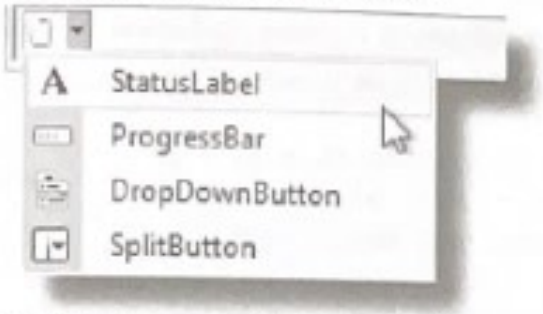
Mesmo que você não tenha visto um Listbox, StatusStrip ou Timer antes, já sabe como definir suas propriedades e trabalhar com eles em seu código. Você aprenderá muito mais sobre eles nos próximos capítulos.

2 **Configure o controle StatusStrip.**
Veja de perto a barra de status na parte inferior da tela. Em um lado, tem uma série de etiquetas:



E do outro lado, tem uma etiqueta e uma barra de progresso:

Adicione `StatusLabel` a `StatusStrip` clicando em sua lista suspensa e selecionando `StatusLabel`:



★ Defina a propriedade `SizingGrip` de `StatusStrip` para False.

★ Use a janela Properties (Propriedades) para definir (Name) para `correctLabel` e `Text` para "Correct: 0". Adicione mais três `StatusLabels`: `missedLabel`, `totalLabel` e `accuracyLabel`.

★ Adicione mais um `StatusLabel`. Defina `Spring` para `True`, `TextAlign` para `MiddleRight` e `Text` para "Difficulty" (Dificuldade). Finalmente, adicione ProgressBar e nomeie-a como `difficultyProgressBar`.

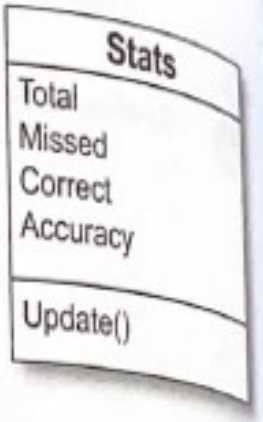
3 **Configure o controle Timer.**
Você notou como seu controle Timer não apareceu em seu formulário? É porque Timer é um *controle não visual*. Realmente não muda a aparência do formulário. Faz exatamente uma coisa: **chama um método sempre de novo**. Defina a propriedade **`Interval`** do controle Timer para 800, para que ele chame seu método a cada 800 milessegundos. Então, **clique duas vezes no ícone *timer1*** no designer. O IDE fará o que sempre faz quando você clica duas vezes em um controle: adicionará um método ao seu formulário. Desta vez, adicionará um chamado **`timer1_Tick`**. Eis o código para ele:

```
private void timer1_Tick(object sender, EventArgs e)
{
    // Adicione uma tecla aleatória a Listbox
    listBox1.Items.Add((Keys)random.Next(65, 90));
    if (listBox1.Items.Count > 7)
    {
        listBox1.Items.Clear();
        listBox1.Items.Add("Game over");
        timer1.Stop();
```

Você adicionará um campo chamado "random" daqui a pouco. Você pode adivinhar qual será seu tipo?

4 **Adicione uma classe para controlar as estatísticas do jogador.**

Se o formulário for exibir o número total de teclas pressionadas pelo jogador, o número das que faltam, o número das corretas e a precisão do jogador, então precisaremos de um modo de controlar esses dados. Parece um serviço para uma nova classe! Adicione uma classe chamada `Stats` ao seu projeto. Ela terá quatro campos int chamados `Total`, `Missed`, `Correct` e `Accuracy`, e um método chamado `Update` com um parâmetro `bool`: `true` se o jogador digitou uma letra correta que estava em ListBox ou `false` se o jogador perdeu uma




```
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);
    }
}
```

Sempre que o método Update() é chamado, ele calcula de novo a % correta e coloca-a no campo Accuracy.

5 **Adicione campos ao seu formulário para manter um objeto Stats e um objeto Random.**

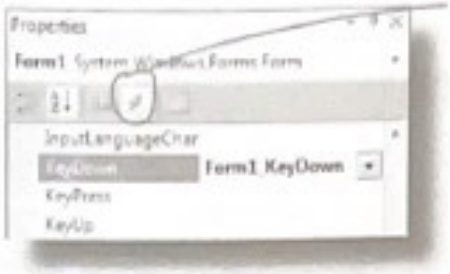
Você precisará de uma instância de sua nova classe **`Stats`** para armazenar realmente as informações, portanto, adicione um campo chamado **`stats`** para armazená-las. E você já viu que precisará de um campo chamado **`random`** – ele contará um objeto **`Random`**. Adicione os dois campos ao topo de seu formulário:

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

6 Lide com as teclas.

Há uma última coisa que seu jogo precisa fazer: sempre que o jogador pressiona uma tecla, ele precisa verificar se essa tecla está correta (e remover a letra de ListBox se estiver) e atualizar as estatísticas em `StatusStrip`.

Volte para o designer de formulários e selecione o formulário. Então, vá para a janela Properties e clique no botão com raio. Vá para a linha **KeyDown** e **clique duas vezes nela**. Isto informa ao IDE para adicionar um método chamado `Form1_KeyDown()` que é chamado sempre que o usuário pressiona uma tecla. Eis o código para o método:



Clique neste botão para muda a exibição da janela Properties. O botão à esquerda muda a janela Properties de volta para mostrar as propriedades.

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    // Se o usuário pressionou uma tecla que está em ListBox, remova-a
    // então, torne o jogo um pouco mais rápido
    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;

        // O usuário pressionou uma tecla correta, portanto atualize o objeto Stats
        // chamando seu método Update() com o argumento true
        stats.Update(true);
    }
    else
    {
        // O usuário pressionou uma tecla incorreta, portanto atualize o objeto
        // Stats chamando seu método Update( ) com o argumento false
        stats.Update(false);
    }

    // Atualize as etiquetas em StatusStrip
    correctLabel.Text = "Correct: " + stats.Correct;
    missedLabel.Text = "Missed: " + stats.Missed;
    totalLabel.Text = "Total: " + stats.Total;
    accuracyLabel.Text = "Accuracy: " + stats.Accuracy + "%";
}
```

São chamados de eventos e você aprenderá muito mais sobre eles depois.

Esta instrução `if` verifica ListBox para saber se ele contém a tecla que o jogador pressionou. Se contiver, então a tecla será removida de ListBox e a dificuldade do jogo será aumentada.

Esta é a parte que aumenta a dificuldade quando o jogador tem mais teclas corretas. Você pode tornar o jogo mais fácil reduzindo as quantidades que são subtraídas de `timer1.Interval` ou mais difícil aumentando-as.

Quando o jogador pressiona uma tecla, o método `Form1_KeyDown()` chama o método `Update()` do objeto `Stats` para atualizar as estatísticas do jogador, então, exibe-as em `StatusStrip`

7 Execute seu jogo.

Seu jogo terminou! Experimente e veja como você se saiu. Você pode precisar ajustar o tamanho da fonte de ListBox e assegurar que ele mantém 7 letras, e pode mudar a dificuldade ajustando os valores que são subtraídos de `timer1.Interval` no método `Form1_KeyDown()`.