Identificando um reajuste salarial

Desenvolva um programa que solicite ao usuário que informe o salário mínimo atual e os seguintes dados de um estagiário: o turno de trabalho, a função e o número de horas trabalhadas no mês. Com base no turno do trabalho do funcionário, deverá ser calculado um coeficiente, como segue:

- Matutino: 1% do salário mínimo;
- Vespertino: 2% do salário mínimo;
- Noturno: 3% do salário mínimo.

Tendo o valor do coeficiente calculado, com base nele obtenha o salário bruto do estagiário, que é o produto dele (coeficiente) e as horas trabalhadas. Sobre o salário bruto é preciso obter um imposto hipotético a ser descontado, como segue:

- Calouro: salário < 300,00 tem imposto de 1%;
- Salário >= 300,00 tem imposto de 2%;
- Veterano: salário < 400,00 tem imposto de 3%;
- Salário >= 400,00 tem imposto de 4%

Para alguns estagiários, será também pago um valor em forma de gratificação. Para os que preencherem todos os requisitos, ela será de R$ 50,00; caso contrário, de R$ 30,00.

Os requisitos são:

- Turno: noturno;
- Número de horas trabalhadas: superior a 80.

Também é oferecido o auxílio-alimentação. Se o estagiário preencher algum dos requisitos, seu auxílio-alimentação será de um terço do seu salário bruto; caso contrário, será de metade deste terço. Os requisitos são:

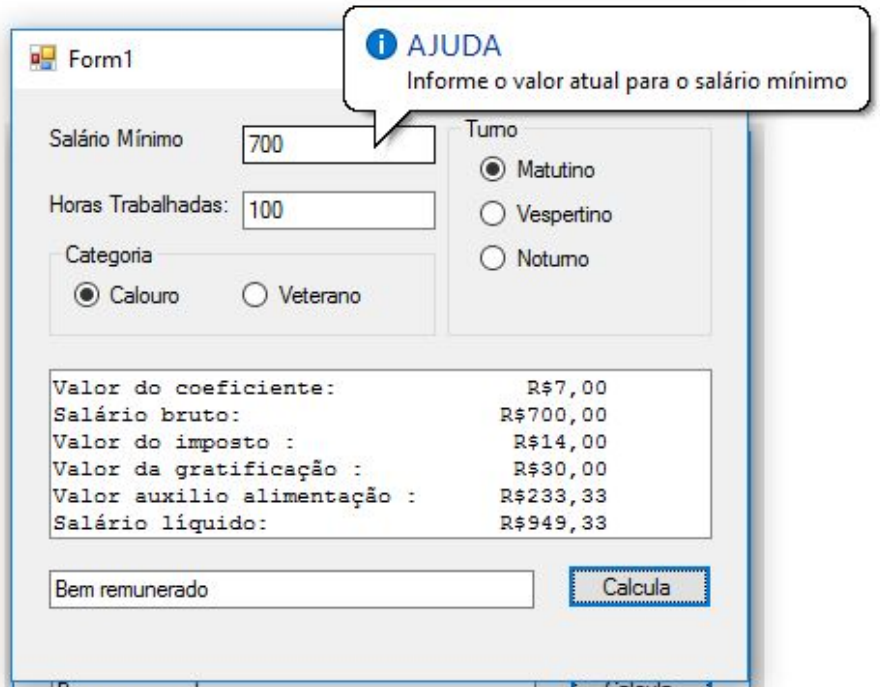
- Categoria: calouro;
- Salário bruto: menor que a metade do salário mínimo.

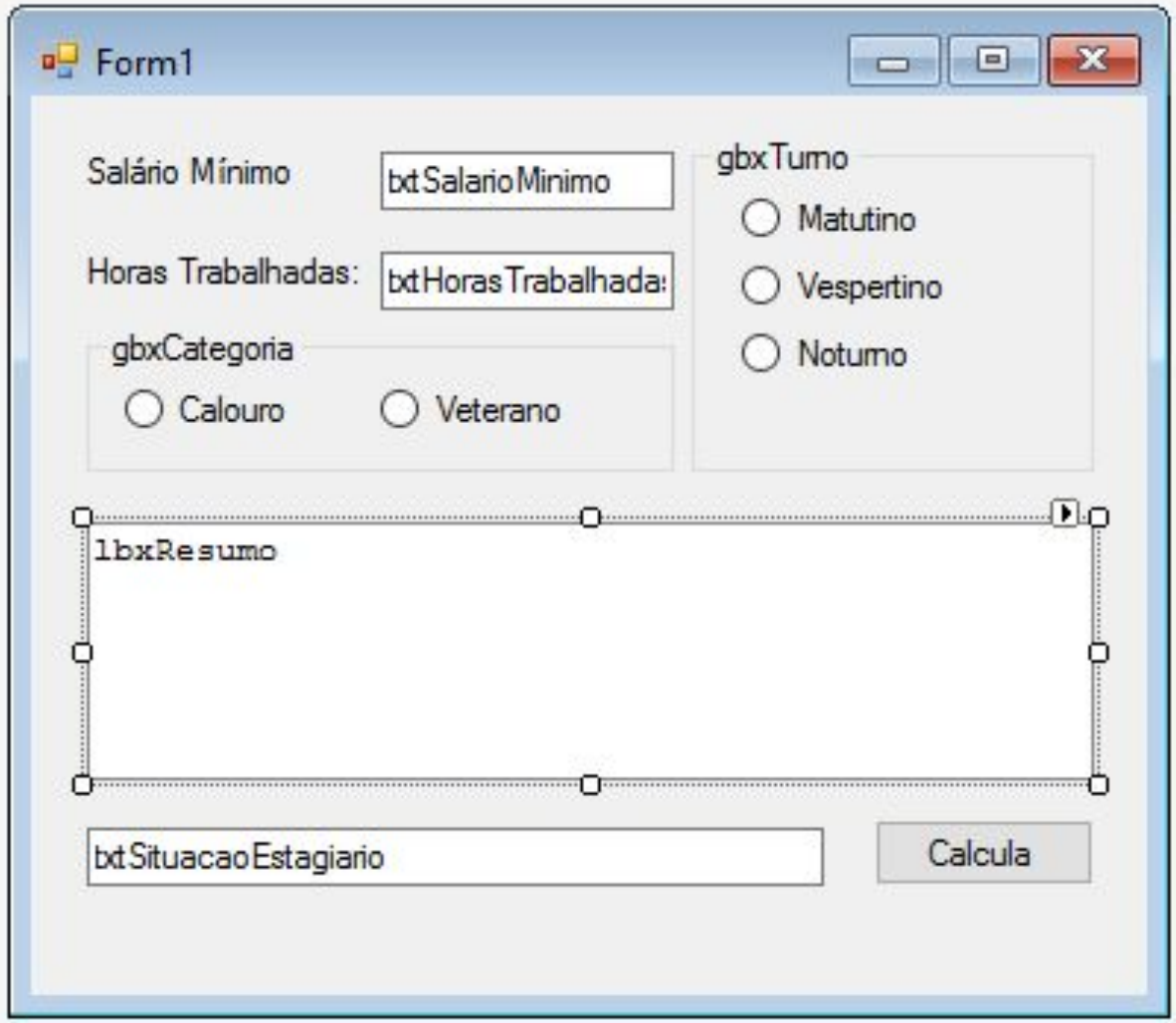
Uma classificação deve ser atribuída ao salário líquido: bruto menos imposto, mais gratificação, mais auxílio-alimentação. Essa classificação deve seguir os requisitos a seguir: • Menor que R$ 350,00: mal remunerado;

- Entre R$ 350,00 e R$ 600,00: normal;
- Maior que R$ 600,00: bem remunerado.

Para a resolução desse problema que possui vários pontos de avaliações, será feito uso de outra estrutura condicional (ou de seleção), a switch...case. Um controle de “auxílio” ao usuário, conhecido como ToolTip, e um controle que representa uma lista de valores exibidos em forma de linhas, conhecido como ListBox, serão introduzidos também nessa resolução.

A figura representa a interface desejada, que será implementada para o problema proposto nesta seção.




Form1
Salário Mínimo
txtSalarioMinimo
Horas Trabalhadas:
txtHorasTrabalhada
gbxCategoria
Calouro
Veterano
gbxTurno
Matutino
Vespertino
Noturno
lbxResumo
txtSituacaoEstagiario
Calcula

Na figura que representa a aplicação, é possível verificar um componente que exibe um resultado dos valores obtidos após o processamento dos dados informados. Esse controle é o ListBox. Também é visível um “balão” com texto que auxilia o usuário em relação a o que informar em um determinado campo do formulário, que é o ToolTip. A caixa em amarelo, que representa a situação do salário do estagiário, é um TextBox com formatação em seu estilo visual. Os demais controles são todos já conhecidos.

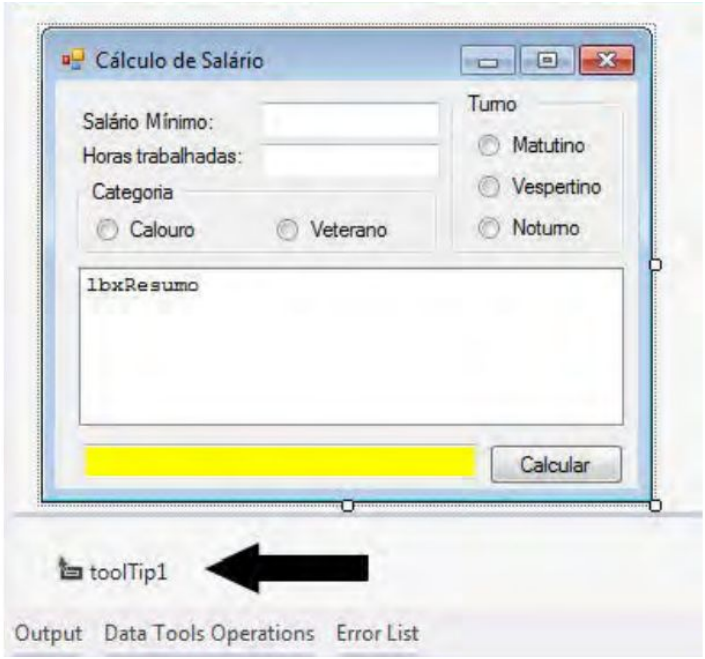
Um controle ListBox exibe uma lista de itens. Esses itens normalmente são strings que, uma vez selecionadas, podem disparar alguma ação na aplicação. Além disso, eles são armazenados na propriedade Items e podem ser exibidos em uma única coluna ou em múltiplas.

Para a exibição em múltiplas colunas, à propriedade MultiColumn deve-se atribuir o valor true. A seleção dos itens nesse controle pode ser configurada para permitir que mais de um item seja selecionado. Para isso, é preciso configurar a propriedade SelectionMode. Caso o tamanho visual do controle (com os itens exibidos) seja inferior à quantidade dos itens tanto em altura quanto comprimento, barras de rolagem são exibidas, quando configuradas para exibição automática.

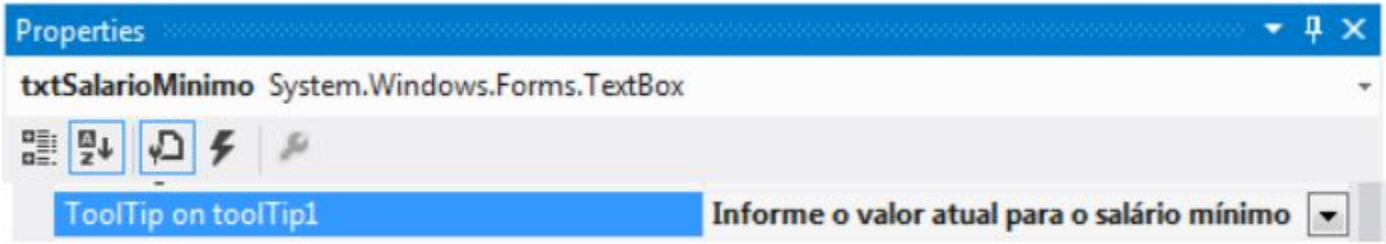
O ToolTip é um componente que oferece um ótimo recurso visual para exibição de mensagens auxiliadoras para os usuários. Para que o texto possa ser exibido, o usuário precisa apontar o cursor do mouse para o controle que possua uma configuração para esse recurso.

Cabe ressaltar que esse componente não é representado graficamente no formulário em tempo de implementação. Ele fica alocado em uma área específica da área de desenho de formulários, conforme retratado na figura à seguir, e as configurações são realizadas nele. Cada controle que fará uso desse recurso recebe uma nova propriedade quando o ToolTip é arrastado. Esta propriedade tem o nome ToolTip on tooltip1, sendo tooltip1 o nome dado ao componente.

Todos os controles necessários estão na categoria Common Controls da ToolBox; arraste-os. Ao arrastar o ToolTip, solte-o sobre o formulário ou sobre algum controle dele. Verifique, por meio da figura abaixo, que ele fica em uma área não visual na área de desenho de formulários, como já comentado. Configure os demais controles, tendo a imagem à seguir como base.



Com esse controle no formulário, uma nova propriedade é inserida para os controles visuais. Como ele permite configurações visuais que podem ser diferentes para grupos distintos de controles, é possível inserir quantos ToolTips forem necessários e, para cada um, uma nova propriedade é inserida.



Com exceção do Button, os controles terão seus valores manipulados e verificados por meio de código. Dessa maneira, nomeie-os de acordo com a sua semântica. De igual forma, configure a propriedade Text de cada um e configure o estilo do TextBox ao lado do Button da maneira que preferir. Para o ListBox, configure a fonte para Courier. As propriedades que precisarão de uma configuração específica são para o ToolTip e são apresentadas na sequência

Sempre que precisar que todas as letras de um texto possuam o mesmo tamanho, procure escolher uma letra monoespaçada, como a Courier.

Propriedades para o ToolTip

- IsBallon : True

Quando essa propriedade recebe o valor True, o texto auxiliar aparece em um balão de diálogo em vez de em uma caixa retangular.

- ToolTipIcon : Info

Nas caixas de auxílio (ou balões) exibidos, é possível adicionar um ícone, o que dá uma boa visão semântica do tipo de auxílio que está sendo exibido. O controle oferece três possíveis opções: Info, Warning e Error.

- ToolTipTitle : Ajuda

Essa propriedade coloca em destaque um título para a mensagem de auxílio ao usuário

O problema proposto neste exemplo emprega diversas verificações, com diversos processamentos, inclusive com dependências entre eles. Aliada a essa nova situação, toda a resolução proposta aqui será delegada a um método específico, que, por sua vez, terá a responsabilidade de também delegar para métodos ainda mais específicos. Ou seja, não é o método que recebe a delegação do evento que tem a responsabilidade de resolver o problema. Ele precisa saber quem resolve e, então, encaminhar para o responsável o que a interface com o usuário oferece, como parâmetros. A seguir, apresento o método para o evento Click do Button.

evento Click para o Button

```
private void btnCalcula_Click(object sender, EventArgs e)
{
    RadioButton rbnTurno = gbxTurno.Controls.OfType <RadioButton>().SingleOrDefault(r => r.Checked);
    RadioButton rbnCategoria = gbxCategoria.Controls.OfType<RadioButton>().SingleOrDefault(r => r.Checked);
    RealizarProcessamento(rbnTurno, rbnCategoria, Convert.ToDouble(txtHorasTrabalhadas.Text), Convert.ToDouble(txtSalarioMinimo.Text));
}
```

método RealizarProcessamento()

O processamento para obter os valores desejados é delegado a um método chamado RealizarProcessamento(). Para cumprir seu objetivo, ele precisa saber quais opções estão ativas – ou qual RadioButton possui a propriedade Checked igual a true – nos containers do tipo GroupBox. Dessa maneira, esses valores são obtidos e enviados como argumento para esse método, que tem seu código apresentado na sequência.

```
private void RealizarProcessamento(RadioButton rbnTurno, RadioButton rbnCategoria, double horasTrabalhadas,double valorSalarioMinimo)
{
    double valorCoeficiente = GetCoeficiente(rbnTurno);
    double valorGratificacao = GetGratificacao(rbnTurno, horasTrabalhadas);
    double valorSalarioBruto = horasTrabalhadas * valorCoeficiente;
    double valorImposto = GetValorImposto(rbnCategoria, valorSalarioBruto);
    double valorAuxilioAlimentacao = GetValorAuxilioAlimentacao(rbnCategoria, valorSalarioBruto, valorSalarioMinimo);
    double valorSalarioLiquido = (valorSalarioBruto + (valorGratificacao + valorAuxilioAlimentacao)) - valorImposto;
    ApresentarResultados(valorCoeficiente, valorSalarioBruto, valorImposto, valorGratificacao, valorAuxilioAlimentacao,
      valorSalarioLiquido);
}
```

Observe na implementação do método RealizarProcessamento() que alguns valores são obtidos por meio de invocação a outros métodos. Esta técnica é importante, pois não traz para um método responsabilidades diferentes das que ele efetivamente tem. Após a realização de todos os cálculos, os valores obtidos são apresentados também por um método específico. A seguir, trago a implementação para o método GetCoeficiente().

método GetCoeficiente()

```
private double GetCoeficiente(RadioButton rbnTurno)
{
    double valorCoeficiente = 0;
    switch (rbnTurno.Text)
    {
        case "Matutino":
            valorCoeficiente = Convert.ToDouble(
            txtSalarioMinimo.Text) * 0.01;
            break;
        case "Vespertino":
            valorCoeficiente = Convert.ToDouble(txtSalarioMinimo.Text) * 0.02;
            break;
        case "Noturno":
            valorCoeficiente = Convert.ToDouble(txtSalarioMinimo.Text) * 0.03;
            break;
    }
    return valorCoeficiente;
}
```

Todo o processamento realizado nesse método é de fácil leitura. Entretanto, a estrutura condicional adotada, switch(), aparece pela primeira vez. Essa estrutura trabalha sempre em relação a um valor avaliado, no caso o rbnTurno.Text. O valor contido nessa variável será comparado com as constantes de cada case. A verificação que for verdadeira terá todo o código abaixo dela executado.

Devido a essa situação, cada conjunto de instruções atribuído a um case possui uma instrução break. Para exemplificar, no caso do valor de rbnTurno.Text ser “Matutino”, e a não existência do break, todas as instruções abaixo seriam executadas, até que um break fosse encontrado ou o switch() finalizado.

A documentação de referência define a estrutura condicional switch() como uma instrução de controle que lida com várias seleções, passando o controle para uma das demonstrações de casos dentro de seu corpo. A execução é transferida para o case, cuja expressão constante combina com o valor recebido como argumento para o switch(). Ela pode incluir qualquer número de case, mas dois cases não podem ter o mesmo valor constante. A execução do bloco de instruções começa na primeira instrução do case selecionado e continua até que a instrução break seja encontrada. Essa instrução de salto (break) é necessária para cada case, inclusive o último (MSDN).

método GetGratificacao()

No método GetGratificacao() são realizadas verificações e alguns processamentos, algo já trabalhado anteriormente, portanto não exige maiores explicações.

```
private double GetGratificacao(RadioButton rbnTurno, double horasTrabalhadas)
{
    double valorGratificacao = 30;
    if (rbnTurno.Text.Equals("Noturno") &&
    horasTrabalhadas > 80)
        valorGratificacao = 50;
    return valorGratificacao;
}
```

método GetValorImposto()

O método GetValorImposto() é representado a seguir. Nessa implementação, há o uso das duas estruturas condicionais em conjunto.

```
private static double GetValorImposto(RadioButton rbnCategoria, double valorSalarioBruto)
{
    double valorImposto = 0;
    switch (rbnCategoria.Text)
    {
        case "Calouro":
            if (valorSalarioBruto < 300)
                valorImposto = valorSalarioBruto * 0.01;
            else
                valorImposto = valorSalarioBruto * 0.02;
            break;
        case "Veterano":
            if (valorSalarioBruto < 400)
                valorImposto = valorSalarioBruto * 0.03;
            else
                valorImposto = valorSalarioBruto * 0.04;
            break;
    }
    return valorImposto;
}
```

método GetValorAuxilioAlimentacao()

Para obtenção do valor do auxílio-alimentação, é invocado o método GetAuxilioAlimentacao(), apresentado na sequência.

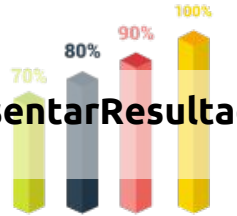
```
private void ApresentarResultados(double valorCoeficiente, double valorSalarioBruto, double valorImposto, double valorGratificacao,
  double valorAuxilioAlimentacao, double valorSalarioLiquido)
{
    txtSituacaoEstagiario.Text = GetSituacaoEstagiario(valorSalarioLiquido);
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}", "Valor do coeficiente:", valorCoeficiente));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}", "Salário bruto:", valorSalarioBruto));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}", "Valor do imposto :", valorImposto));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}", "Valor da gratificação :", valorGratificacao));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}", "Valor auxilio alimentação :", valorAuxilioAlimentacao));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}", "Salário líquido:", valorSalarioLiquido));
}
```

método ApresentarResultados()

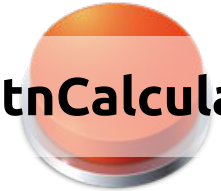
Os dados processados precisam ser apresentados ao usuário final. Para isso, devem ser responsabilidade de um método específico. Neste sentido, o método ApresentarResultados() formata e apresenta os resultados no controle ListBox, conforme apresentado na sequência.

```
private void ApresentarResultados(double valorCoeficiente, double valorSalarioBruto, double valorImposto, double valorGratificacao, double
  valorAuxilioAlimentacao, double valorSalarioLiquido)
{
    txtSituacaoEstagiario.Text = GetSituacaoEstagiario(
    valorSalarioLiquido);
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}",
    "Valor do coeficiente:", valorCoeficiente));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}",
    "Salário bruto:", valorSalarioBruto));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}",
    "Valor do imposto :", valorImposto));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}",
    "Valor da gratificação :", valorGratificacao));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}",
    "Valor auxilio alimentação :", valorAuxilioAlimentacao));
    lbxResumo.Items.Add(String.Format("{0,-29}{1,12:C}",
    "Salário líquido:", valorSalarioLiquido));
}
```

RealizarProcessamento()



ApresentarResultados()



btnCalcular

GetCoeficiente()

GetGratificacao()

GetValorAuxilioAlimentacao()

GetSituacaoEstagiario()

GetValorImposto()

Assinatura dos métodos

GetSituacaoEstagiario(

ApresentarResultados(

GetValorAuxilioAlimentacao(

GetValorImposto(

GetGratificacao(

GetCoeficiente(

RealizarProcessamento(

método GetSituacaoEstagiario()

O método a ser apresentado é o responsável pela identificação da situação do qualificador do salário do estagiário.

```
private string GetSituacaoEstagiario(double valorSalarioLiquido)
{
    if (valorSalarioLiquido < 350)
        return "Mal remunerado";
    else if (valorSalarioLiquido < 600)
        return "Normal";
    else
        return "Bem remunerado";
}
```