ESTRUTURAS DE DADOS – 2º Desenvolvimento de Sistemas – PROVA C – 29/09/2025

RA:_____ Nome: _____

Username: _____ Senha: _____ Computador usado: _____

- **Desligue** seu celular e o guarde desligado na mochila. Coloque a mochila no local indicado pelo professor.
- **Não são permitidos o uso de Internet, consultas ou qualquer tipo de comunicação, seja por que modo for**.
- Faça seu login na rede usando o username e senha fornecidos pelo professor.
- Mapeie a unidade de rede \\Venus\ProvaTP
- Copie a pasta ***prova***C dessa unidade de rede para a área de trabalho (desktop) de seu computador e renomeie a palavra ***prova*** da pasta copiada com os dígitos do seu RA antes da letra C. Por exemplo: ***24179***C.
- Desenvolva a prova usando os arquivos presentes nessa pasta.
- Utilize boas práticas de programação: nomes significativos, indentação, comentários.
- **É somente sua a responsabilidade de não perder seu trabalho durante sua prova,** portanto salve frequentemente seu projeto e faça cópias da pasta de trabalho na pasta c:\temp e/ou Documentos.
- Após terminar sua prova, compacte a pasta com sua solução (***seuRA***C) e chame o professor para fazer a entrega do arquivo compactado na atividade "Prova C" do Classroom.

Tema – Tratamento de Árvore Binária de Strings

Parte 1 · Métodos e propriedades

Na classe NoArvore<Dado>:

1. Codifique uma propriedade lógica na classe NoArvore<Dado> que retorne se o nó é ou não uma folha.

Na classe Arvore<Dado>:

Cada método da classe Arvore descrito abaixo deverá ser público, pois será chamado pela aplicação (formulário). Cada um desses métodos deverá chamar o método recursivo associado, que será privativo da classe Arvore.

2. **void Inserir(Dado novoDado, Dado dadoExistente, bool esquerda):** insere um nó na árvore binária de como descendente de um nó contendo o dadoExistente, à esquerda se o parâmetro esquerda valer **true** e à direita caso esse parâmetro valha **false**. Você pode usar o método Buscar(), previamente codificado, para auxiliar na busca do dadoExistente, que será o nó antecessor do novo nó a ser inserido. Não esqueça de ligar o novo nó inserido ao descendente anterior (esquerdo ou direito) do nó antecessor.
3. **int ContarNosInternos():** chama o método privado recursivo associado, que contará e retornará a quantidade de nós da árvore que não são folhas, usando a propriedade EhFolha de cada nó. Após o retorno do método recursivo para o método público, este retornará à aplicação o valor calculado.
4. **void CaminhoAte(Dado procurado, ref List<NoArvore<Dado>> lista):** cria uma lista com o caminho da raiz até o dado e a retorna como parâmetro (preenchida ou vazia, no caso de árvore vazia)**.** Para tanto, chamará o método privado recursivo associado, que fará a busca do dado procurado e, enquanto não o acha, adiciona na lista o endereço de cada nó visitado. Após o retorno do método recursivo para o método público, este retornará à aplicação a lista gerada.
5. **bool EstaBalanceada():** verifica se a árvore está balanceada. Uma árvore está balanceada se a diferença de altura entre o ramo esquerdo e o ramo direito é, no máximo, | 1 |, sendo que cada ramo e sub-ramo estão também balanceados. Para tanto, chamará o método privado recursivo associado, passando a raiz como primeiro parâmetro; esse método deverá calcular a altura de cada nó, à esquerda e a direita e retornar se essas alturas estão ou não balanceadas. Após o retorno do método recursivo para o método público, este retornará à aplicação o resultado lógico definido.
6. **int SomaDosDados()** – tratando os dados armazenados na árvore como valores inteiros, retorne a soma de todos os valores armazenados na árvore. Para converter a propriedade Info para inteiro, use o método Convert.ToInt32(informação desejada), que retorna "informação desejada" como um valor inteiro. Você pode usar try/catch para descobrir se a conversão para inteiro foi bem sucedida ou não. Caso o dado não seja inteiro, somar 0.

A classe Arvore possui alguns métodos pré-codificados e funcionais, como Altura(), AlturaRecursiva(), Buscar(), BuscarRecursivo(), Desenhar() e DesenharNo().

Parte 2 · Interface Gráfica (Windows Forms):

Usando o formulário previamente desenvolvido, observe que já existem declarados arvore, vetor e pbArvore_Paint() da classe do formulário. Esses atributos deverão ser usados nos tratadores de eventos descritos a seguir.

Sempre que houver uma operação, deve-se chamar o método pbArvore.Invalidate(), que disparará a execução do evento Paint do pbArvore que, por sua vez, já chama o método que desenha a árvore no pictureBox pbarvore.

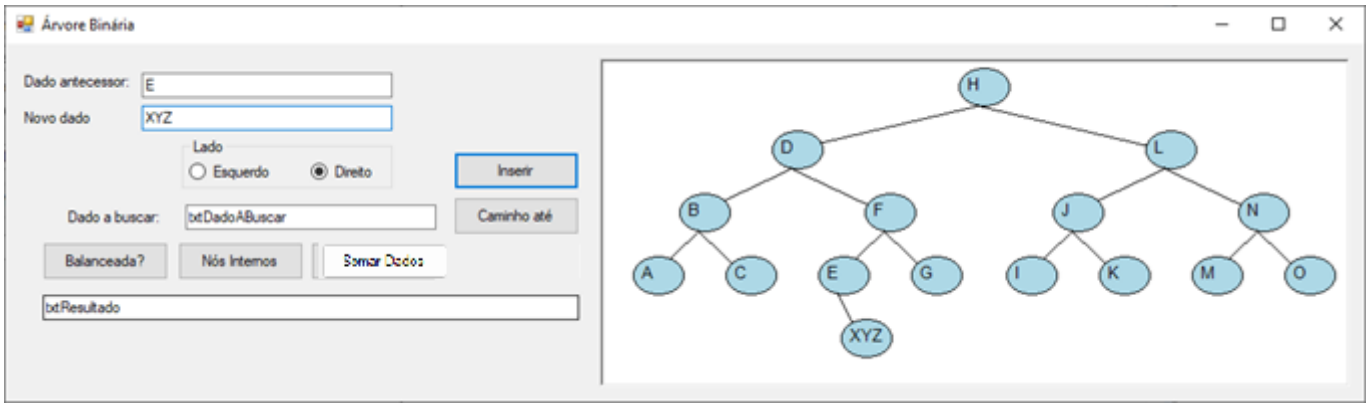
btnInserir.Click: chamar o método Inserir() da árvore, passando como parâmetros os valores dos TextBoxes txtDadoAntecessor e txtNovoDado, bem como o lado para o qual se escolheu inserir o nó do novo dado abaixo do nó do dado antecessor. Atualizar a exibição da árvore.

btnCaminhoAte.Click: chamar o método CaminhoAte(), passando como parâmetro a String digitada em txtDadoABuscar, e exibir, no txtResultado, o caminho retornado pelo método chamado.

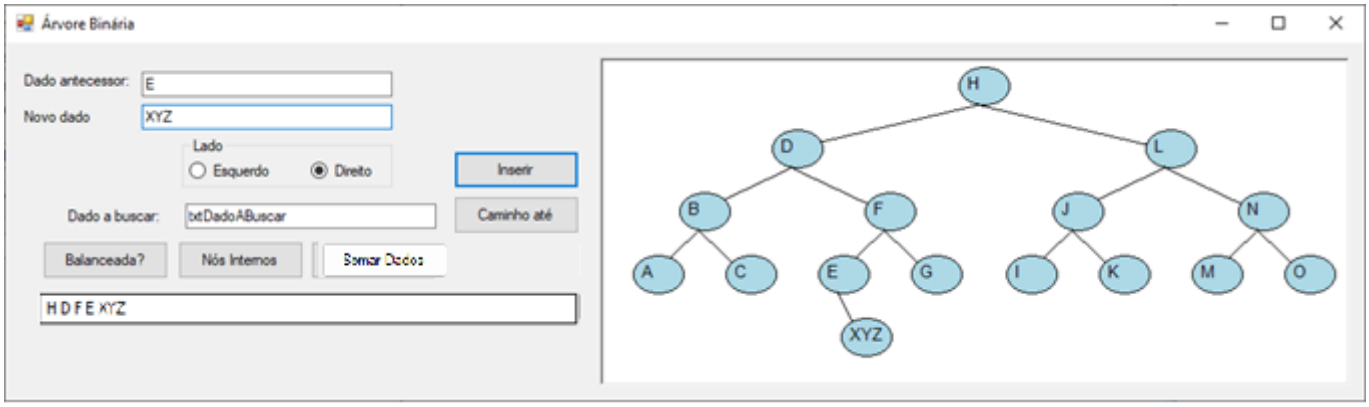
btnBalanceada.Click: chamar o método EstaBalanceada() da árvore, e escrever, no txtResultado, se a árvore está ou não balanceada.

btnNosInternos.Click: chamar o método ContarNosInternos() da árvore, e escrever, no txtResultado, o número de nós internos da árvore, resultante desse método.

btnSomarValores.Click: chamar o método SomaDosDados() da árvore e exiba o resultado em txtResultado.



Inserir



Caminho até