**FATEC SÃO CAETANO DO SUL – ANTONIO RUSSO**

**Filipe Pereira Bezerra  
Pedro Zamboni Branzani**

**ESTRUTURAS DE DADOS  
INSTRUMENTO AVALIATIVO DO TERCEIRO BIMESTRE  
26 DE SETEMBRO DE 2023**   
**PROFESSOR CARLOS HENRIQUE VERÍSSIMO PEREIRA**

**São Paulo**  
**2023**

**SUMÁRIO**

**RESUMO1**

**ARGUMENTAÇÃO TEÓRICA2**

MODELO DE PESQUISA

RECURSIVIDADE

ÁRVORE BINÁRIA

**RESULTADOS OBTIDOS3**

EXECUÇÃO DO PROGRAMA

**CONCLUSÃO** **4**

1. **RESUMO**

Este texto visa elucidar o conceito da recursão na estrutura da Árvore Binária. Recentemente, durante uma aula, foi abordada a construção de uma Árvore Binária, que, em essência, é uma representação de uma hierarquia não convencional. Ela é composta por nós, cuja criação e organização são conduzidas de maneira recursiva.

Cada nó da Árvore Binária é constituído por um valor inteiro e duas referências aos seus filhos, um à esquerda e outro à direita. A recursividade desempenha um papel crucial na criação e na travessia dessa estrutura. À medida que os nós são construídos, os números são ordenados e estabelecem relações de parentesco, distinguindo pais de filhos.

A lógica da recursão é a seguinte: se o número raiz possuir sub-árvores, que são os ramos filhos da raiz, o nó é classificado como interno. No entanto, caso o nó não possua filhos, ele é considerado externo ou uma folha.

Em outras palavras, ao inserir um número na árvore, a comparação é feita com o nó raiz. Se o número for maior, ele é encaminhado para a direita na árvore; caso contrário, segue para a esquerda. Isso é repetido até que não haja mais filhos, chegando-se aos nós externos.

Como resultado, a organização da Árvore Binária exibe uma ordenação natural dos números, onde os valores maiores se encontram à direita e os menores à esquerda, refletindo uma estrutura hierárquica intrincada.

1. **Argumentação Teórica**
   1. **Modelo de pesquisa**

A busca em ordem é usada no algoritmo principalmente por duas razões fundamentais:

A busca da Ordenação dos Elementos garante que os elementos de busca sejam visitados e impressos em ordem crescente. Isso é uma característica essencial da busca, onde os valores menores estão sempre à esquerda e os valores maiores estão sempre à direita. A Recursão Natural busca aproveita essa propriedade, permitindo que os valores sejam processados em ordem.

A Recursão Natural se alinha naturalmente com a recursão. A recursão é uma técnica eficaz para percorrer estruturas de dados hierárquicas como árvores. Quando você está percorrendo uma árvore binária de busca, a busca em ordem permite que você processe os nós da árvore em uma sequência lógica: primeiro o nó mais à esquerda, depois o nó atual e, por fim, o nó mais à direita. Essa sequência é facilmente expressa de forma recursiva, pois você pode aplicar a mesma lógica a cada subárvore.

Portanto, usar a busca em ordem é uma escolha natural quando você deseja percorrer uma árvore binária de busca para imprimir seus elementos em ordem crescente. Ela garante que os elementos sejam processados de acordo com a estrutura da árvore e permite que você obtenha uma lista ordenada dos valores contidos na árvore de maneira eficiente, sem a necessidade de ordenação adicional. Isso é especialmente valioso quando se lida com grandes conjuntos de dados ou quando se deseja manter a ordenação em tempo real à medida que novos elementos são inseridos na árvore.

* 1. **Recursividade**

A estrutura do código e seus princípios podem ser resumidos da seguinte forma:

1. A entidade denominada "Nó" encapsula um valor inteiro e duas referências aos filhos, designados como "esquerdo" e "direito". Esta abstração serve para criar e percorrer os nós que compõem a árvore.
2. A classe principal, conhecida como "Árvore Binária", abriga um campo privado chamado "raiz". Essa variável é uma referência fundamental, apontando para o nó raiz da árvore.
3. Dentro dessa estrutura, existe o método privado "Inserir". Este método desempenha um papel crucial na inserção de novos valores na árvore. A recursão é aplicada para navegar na árvore e determinar a localização apropriada para inserir o novo valor. O método avalia se o valor a ser inserido é maior ou menor que o valor no nó atual e, com base nessa comparação, direciona a inserção para a subárvore esquerda ou direita, invocando repetidamente o próprio método "Inserir".
4. Adicionalmente, encontramos o método privado "Mostrar". Esse método executa uma travessia em ordem da árvore binária, ou seja, apresenta os valores dos nós em ordem crescente. Novamente, a recursão é a técnica empregada para percorrer a árvore. O processo começa pela exploração do filho esquerdo, seguido pela exibição do valor no nó atual e, finalmente, a exploração do filho direito.
5. Por fim, o método "Main" atua como ponto de partida para a execução do programa. Ele instância a classe, lê valores inteiros fornecidos pelo usuário e os insere na árvore binária utilizando o método "Inserir". A recursão é um elemento fundamental durante a inserção e também na exibição dos valores da árvore, realizada pelo método "Mostrar".
   1. **Árvore binária**

A Árvore Binária é uma estrutura hierárquica que se assemelha a uma lista organizada. Nessa representação, cada nó contém um valor inteiro e possui duas referências, uma para o nó à esquerda e outra para o nó à direita. A capacidade de construir essa estrutura se baseia na aplicação do conceito de recursividade, que desempenha um papel crucial na criação e na exploração dos nós, estabelecendo relações hierárquicas.

O primeiro número inserido na árvore é conhecido como a "raiz", e os números subsequentes são organizados em ordem, representando as descendências da raiz. Quando um novo número é adicionado à árvore, a comparação é realizada em relação ao valor da raiz. Se o valor a ser inserido for maior que o valor da raiz, ele é direcionado para o ramo direito da árvore. Por outro lado, se o valor for menor, ele segue para o ramo esquerdo. Esse processo é repetido até que não haja mais filhos, sendo estes últimos chamados de "folhas".

O resultado final dessa organização revela uma ordem natural dos números, onde os valores maiores são posicionados à direita e os menores à esquerda. Essa estrutura reflete uma hierarquia intrínseca, demonstrando a versatilidade e utilidade da Árvore Binária como uma ferramenta para organização e análise de dados.

1. **Resultados Obtidos**



“NÓ”

Este código representa uma classe chamada Nó, que é usada para criar nós em uma árvore binária, como parte de uma estrutura de dados. Aqui está o que cada parte faz:

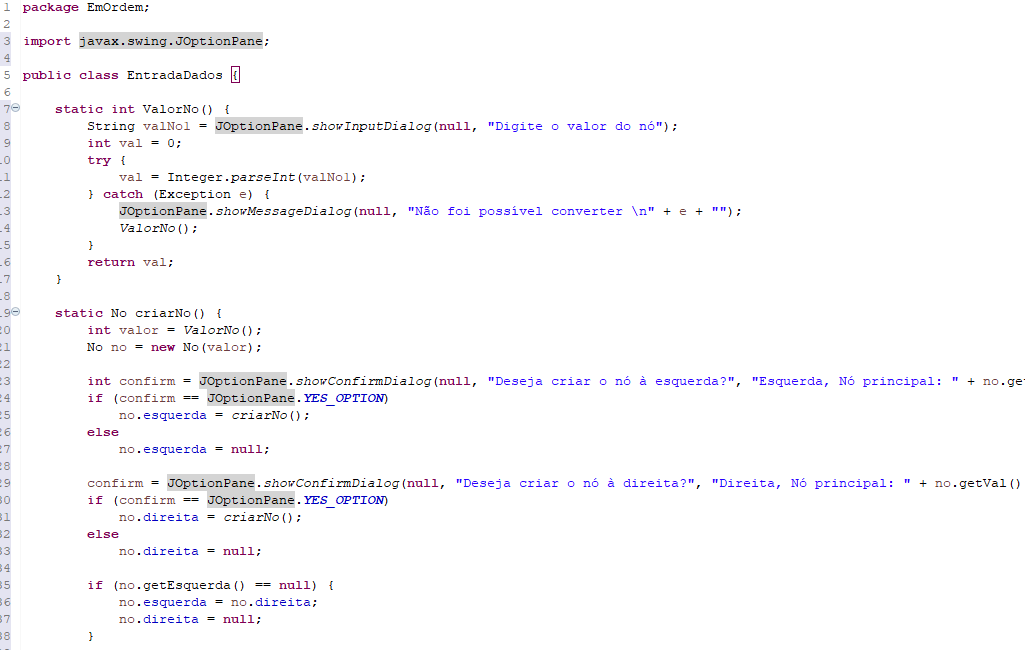
private int val: Esta variável armazena um número inteiro associado a um nó na árvore. Cada nó pode conter um número diferente.

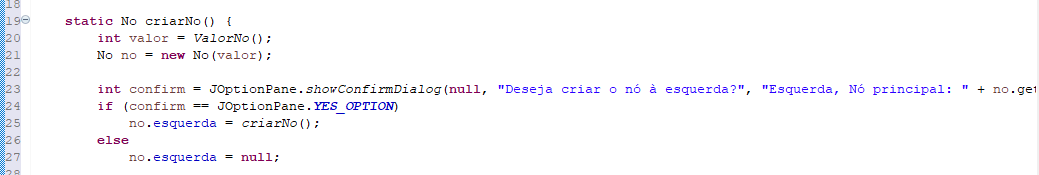
No direita e No esquerda: Estas são referências aos nós filhos direito e esquerdo do nó atual. Quando um novo nó é criado, essas referências são inicializadas como null, o que significa que inicialmente o nó não tem filhos.

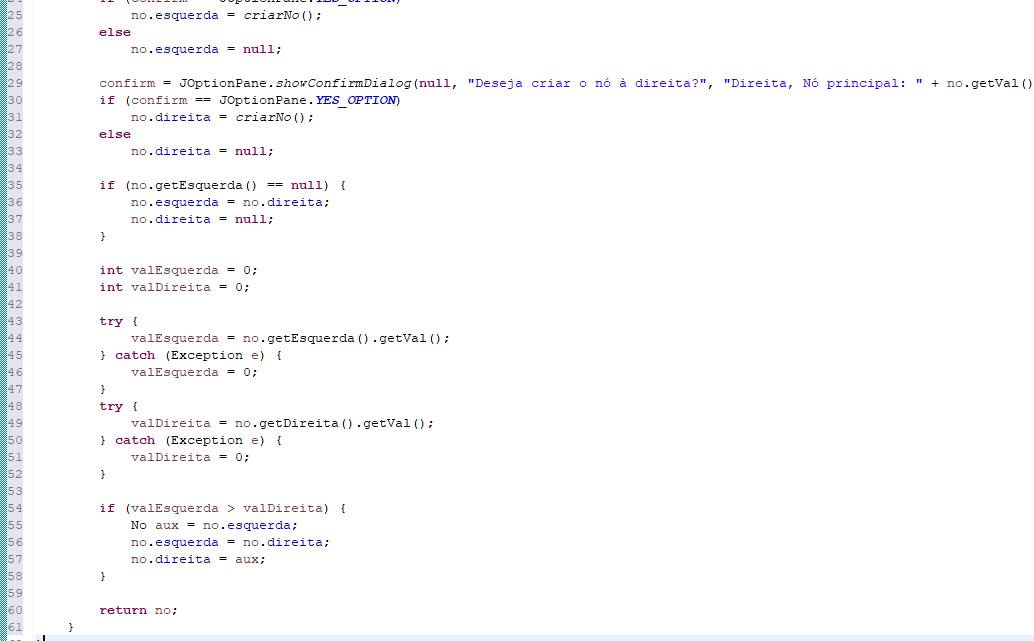
O construtor public No(int val): Quando um novo nó é criado, ele precisa de um valor inteiro como argumento. Este valor é atribuído ao val do nó, e as referências direita e esquerda são definidas como null.

Os métodos getVal(), getDireita(), e getEsquerda(): Esses métodos permitem acessar os valores e as referências dos nós de fora da classe. getVal() retorna o valor do nó, getDireita() retorna a referência para o nó filho direito e getEsquerda() retorna a referência para o nó filho esquerdo.

Em resumo, esta classe No é usada para criar e manipular nós em uma árvore binária. Cada nó contém um valor e pode ter filhos à direita e à esquerda, formando a estrutura hierárquica da árvore. Essa estrutura de dados é amplamente utilizada em programação para resolver diversos tipos de problemas que envolvem organização e pesquisa de dados.







“ENTRADA DE DADOS”

Este código Java é um programa interativo que permite aos usuários criar uma árvore binária. Vou explicar as partes principais:

Método ValorNo(): Este método pede ao usuário para inserir um valor para um nó da árvore através de uma caixa de diálogo. Ele tenta transformar a entrada do usuário em um número inteiro. Se a conversão for bem-sucedida, o valor inserido é retornado. Caso contrário, se ocorrer algum erro na conversão, uma mensagem de erro é exibida, e o usuário é solicitado a inserir o valor novamente.

Método criarNo(): Este método é responsável por criar um novo nó na árvore. Primeiro, ele chama o método ValorNo() para obter um valor a ser atribuído ao nó. Em seguida, ele pergunta ao usuário se deseja criar um nó à esquerda e à direita do nó atual, usando caixas de diálogo. Dependendo das respostas do usuário, ele cria os nós correspondentes à esquerda e à direita. Após a criação, ele ajusta a ordem dos nós, colocando o nó de menor valor à esquerda e o de maior valor à direita. O nó final, com seus filhos e a estrutura organizada, é retornado como resultado.

Este código é uma maneira interativa de construir uma árvore binária, onde cada nó pode ter até dois filhos (um à esquerda e outro à direita), e os nós são organizados com base nos valores inseridos pelo usuário. É uma implementação simples de uma estrutura de árvore binária usando entrada de dados interativa.



“SAIDA DE DADOS”

Este código Java faz parte de um programa que lida com uma estrutura de árvore binária e fornece uma funcionalidade para exibir os valores dos nós em diferentes ordens.

Método saidaDadosEmOrdem(): Este método é usado para percorrer a árvore binária e coletar os valores dos nós em ordem. Ele recebe um nó como entrada e uma string result que é usada para acumular os valores dos nós. O método percorre a árvore recursivamente, indo para a esquerda, coletando o valor do nó atual e, em seguida, indo para a direita. Ele retorna uma string contendo os valores dos nós em ordem.

Método selectOrder(): Este método solicita ao usuário que escolha uma ordem para exibir os valores dos nós da árvore. Ele exibe uma caixa de diálogo que permite ao usuário inserir "1" para exibir os valores em ordem ou "0" para sair. Ele converte a entrada do usuário em um número inteiro e toma a ação correspondente. Se o usuário escolher "1" (em ordem), ele chama o método saidaDadosEmOrdem() para obter os valores em ordem e exibe o resultado em uma caixa de diálogo. Se o usuário escolher "0" (sair), o programa encerra a execução. Se a entrada do usuário não for válida, o método solicita novamente a escolha da ordem.

Em resumo, este código Java permite ao usuário escolher se deseja ver os valores dos nós de uma árvore binária em ordem e fornece essa funcionalidade. Ele utiliza métodos recursivos para percorrer a árvore e coletar os valores de acordo com a ordem selecionada pelo usuário. É uma aplicação interativa que demonstra como percorrer e exibir dados de uma árvore binária em ordem.

1. **CONCLUSÃO**

A incorporação da recursão na construção de uma Árvore Binária em um projeto Java trouxe notáveis vantagens para o processo de aprendizado. Esse aprimoramento resultou em uma hierarquia não linear mais estruturada e clara, que inclui valores inteiros e estabelece duas referências para os filhos, um à esquerda e outro à direita. Além disso, proporcionou uma visão mais aprofundada e elucidativa da Árvore em si. Essa implementação ressalta o potencial significativo da recursão ao criar e explorar os nós da árvore, enfatizando a relação de parentesco entre eles.