



# Projeto com Árvore Binária

**N**

---

este projeto, vamos trabalhar com os percursos em árvore binária de busca.

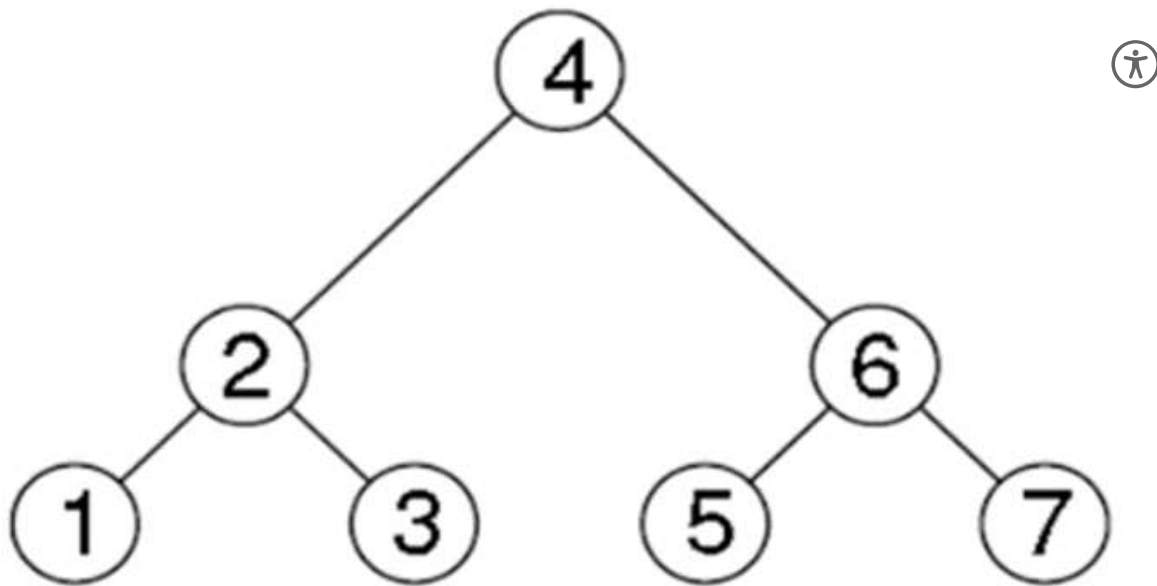
## DEFININDO AS APLICAÇÕES

Vamos desenvolver um programa que recebe um percurso do usuário, recebe uma árvore binária com elementos e mostra a lista na ordem solicitada pelo usuário.

Para isso, vamos montar um menu de opções, serão três opções:

- Em Ordem;
- Pré Ordem; e
- Pós Ordem.

Observando a imagem seguinte da árvore binária:



Vamos desenvolver um módulo para cada percurso e apresentar o percurso de acordo com a opção selecionada pelo usuário.

Realizando o percurso da árvore da imagem anterior, temos:

**Em Ordem: 1 2 3 4 5 6 7**

**Pré-Ordem: 4 2 1 3 6 5 7**

**Pós-Ordem: 1 3 2 5 7 6 4**



Desenvolva um algoritmo que recebe do usuário cinco números inteiros numa pilha com capacidade para cinco números e os mostra.

```
public void emOrdem (No ABB)
```

```
{
```

```
    if (ABB != nulo)
```

```
    {
```

```
        emOrdem(ABB.esquerda);
```

```
        visita(ABB);
```

```
        emOrdem(ABB.direita);
```

```
    }
```

```
}
```

```
public void preOrdem (No ABB)
```

```
{
```

```
    if (ABB != nulo)
```

```
    {
```

```
visita(ABB);
```



```
preOrdem(ABB.esquerda);
```

```
preOrdem(ABB.direita);
```

```
}
```

```
}
```

```
public void posOrdem (No ABB)
```

```
{
```

```
if (ABB != nulo)
```

```
{
```

```
posOrdem(ABB.esquerda);
```


```
posOrdem(ABB.direita);
```

```
visita(ABB);
```

```
}
```

```
}
```

**EXERCITANDO AS APLICAÇÕES**

Para que cada um dos métodos de percurso pré-ordem, pós-ordem e em ordem em uma árvore binária possa ser executado, temos que realizar sua chamada adequada em um método principal, no caso do Java, no void main. 

Vamos observar o método main com a escolha da opção e a chamada de cada percurso.

```
public static void main (String entrada[]) {
```

```
    ArvoreBinaria ABB;
```

```
    ABB = new ArvoreBinaria();
```

```
    do {
```

```
        op = menu();
```

```
        vi = LerNum();
```

```
        switch (op) {
```

```
            case 1 : emOrdem(ABB);
```

```
                break;
```

```
            case 2 : preordem(ABB);
```

```
                break;
```

```
            case 3 : posOrdem(ABB);
```

```
                break;
```

```
}
```



```
} while (op<1 && op >3);
```

```
System.exit(0);
```

```
}
```

## APLICAÇÕES NO JAVA

O método void visita é um método que mostra as informações de determinado nó da árvore binária de busca. Segue o método void visita na linguagem Java.

```
public static void visita (No ABB)
```

```
{
```

```
System.out.println(ABB.num + " ");
```

```
}
```

Temos também o método int Menu que apresenta uma mensagem de opções de percurso para o usuário poder escolher. Segue o Código em Java.

```
public static int menu() {
```

```
String msg = "";
```



```
int op;
```

```
msg = msg + "Digite 1 para Em Ordem\n";
```

```
msg = msg + "Digite 2 para Pré Ordem\n";
```

```
msg = msg + "Digite 3 para Pós Ordem\n";
```

```
msg = msg + "Digite 0 para sair do sistema\n";
```

```
op = Integer.parseInt(JOptionPane.showInputDialog(msg));
```

```
return op;
```

```
}
```

### **Atividade extra**

### **Indicação de leitura:**

Você pode utilizar o livro Estrutura de Dados: algoritmos, análise da complexidade e implementações em Java e C/C++, da Ana Fernanda Gomes Ascencio e Graziela Santos de Araújo, no capítulo 7 com os projetos e exercícios sobre Árvore Binária.



## Referência Bibliográfica

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos Teoria e Prática**. Editora Cammpus. 3a Edição. 2012.

GOODRICH, M. T.; TAMASSIA, R. **Estruturas de Dados & Algoritmos em Java**. Editora Grupo A: Bookman, 5a Edição. 2013.

## Atividade Prática 16 – Projeto com árvore binária

Título da Prática: Aplicações as Árvores binárias em Java

Objetivos: Entender como utilizar o netbeans para desenvolver programas em Java para manipular e aplicar os recursos em árvores binárias

Materiais, Métodos e Ferramentas: Computador, netbeans, Java.

## Atividade Prática

No desenvolvimento de um algoritmo que recebe do usuário cinco números inteiros numa árvore binária, de acordo com a regra (recursiva): “*todo elemento à*



*esquerda é menor que a raiz, todo elemento à direita é maior ou igual à raiz”, e*



mostra esses números, temos o algoritmo seguinte.

Desenvolva o programa em Java deste algoritmo no NetBeans.

## Algoritmo BIntNo

### início\_algoritmo

// definição do tipo registro BIntNo com os campos abaixo

**tipo** BIntNo = **registro** // o tipo registro chama-se BIntNo

valor **numérico\_inteiro**; // campos inteiros

esq, dir **BIntNo**; // campo vetor de capacidade

**fimregistro;**

**Fim\_algoritmo.**

## Algoritmo ArvoreBinaria

### início\_algoritmo

**Declarar**

Raiz BIntNo;

BlntNo inserir (arvore BlntNo, novoNo **numérico\_inteiro**)



**início\_módulo**

**se** (arvore = nulo)

**então**

**retornar novo** BlntNo (novoNo);

**senão**

**se** (novoNo < arvore.valor)

**então**

arvore.esq ← inserir (arvore.esq, novoNo);

**senão**

arvore.dir ← inserir (arvore.dir, novoNo);

**fimse;**

**fimse;**

**retornar** arvore;

**fim\_módulo;**

inserirNo (novoValor **numérico\_inteiro**)

**início\_módulo**

Raiz ← inserir(Raiz, novoValor);

**fim\_módulo;**



exibirEsquerdo (arv BIntNo)

**início\_módulo**

**se** (arv <> nulo)

**então**

exibirEsquerdo (arv.esq);

escrever(arv.valor);

**fimse;**

**fim\_módulo;**

exibirNoEsq()

**início\_módulo**

exibirEsquerdo(Raiz);

**fim\_módulo;**

exibirDireito (arv BIntNo)

**início\_módulo**

**se** (arv <> nulo)

**então**



exibirDireito(arv.dir);

escrever (arv.valor);

**fimse;**

**fim\_módulo;**

exibirNoDir()

**início\_módulo**

exibirDireito(Raiz);

**fim\_módulo;**

**início\_módulo**

exibirNoEsq( );

exibirRaiz( );

exibirNoDir( );

**fim\_módulo;**

exibirRaiz()

**início\_módulo**

escrever("raiz ", Raiz.valor);



**fim\_módulo;**

No (item **numérico\_inteiro**)

**início\_módulo**

tempNo, pai, filho, temp BIntNo;

tempNo  $\leftarrow$  Raiz;

pai  $\leftarrow$  null;

filho  $\leftarrow$  Raiz;

**enquanto** (tempNo  $\neq$  nulo e tempNo.valor  $\neq$  item) **faça**

pai  $\leftarrow$  tempNo;

**se** (item < tempNo.valor)

**então**

tempNo  $\leftarrow$  tempNo.esq;

**senão**

tempNo  $\leftarrow$  tempNo.dir;

**fimse;**

**se** (tempNo = nulo)



**então**

escrever("item não localizado!");

**fimse;**

**se** (pai = nulo)

**então**

**se** (tempNo.dir = nulo)

**então**

Raiz  $\leftarrow$  tempNo.esq;

**senão**

**se** (tempNo.esq = nulo)

**então**

Raiz  $\leftarrow$  tempNo.dir;

**senão**

**para** temp  $\leftarrow$  tempNo **e** filho  $\leftarrow$  tempNo.esq **até** filho.dir  $\neq$  null

**passo** temp  $\leftarrow$  filho **e** filho  $\leftarrow$

filho.dir **faça**

**fimpara;**

**se** (filho  $\neq$  tempNo.esq)

**então**



temp.dir  $\leftarrow$  filho.esq;

filho.esq  $\leftarrow$  Raiz.esq;

**fimse;**

filho.dir  $\leftarrow$  Raiz.dir;

Raiz  $\leftarrow$  filho;

**fimse;**

**fimse;**

**senão**

**se** (tempNo.dir = nulo)

**então**

**se** (pai.esq = tempNo)

**então**

pai.esq  $\leftarrow$  tempNo.esq;

**senão**

pai.dir  $\leftarrow$  tempNo.esq;

**fimse;**

**senão**

**se** (tempNo = nulo)

**então**



**se** (pai.esq = tempNo)

**então**

    pai.esq  $\leftarrow$  tempNo.dir;

**senão**

    pai.dir  $\leftarrow$  tempNo.dir;

**fimse;**

**senão**

**para** temp  $\leftarrow$  tempNo **e** filho  $\leftarrow$  tempNo.esq **até** filho.dir  $\neq$  nulo

**passo** temp  $\leftarrow$  filho **e** filho  $\leftarrow$  filho.dir

**fimpara;**

**se** (filho  $\neq$  tempNo.esq)

**então**

    temp.dir  $\leftarrow$  filho.esq;

    filho.esq  $\leftarrow$  tempNo.esq;

**fimse;**

    filho.dir  $\leftarrow$  tempNo.dir;

**se** (pai.esq = tempNo)

**então**



pai.esq  $\leftarrow$  filho;



**senão**

pai.dir  $\leftarrow$  filho;

**fimse;**

**fimse;**

**fimse;**

**fimse;**

**fimenquanto;**

**fimmódulo;**

**fim\_algoritmo.**

**Algoritmo teste**

**início\_algoritmo**

**Declarar**

num **numérico\_inteiro;**

ArvoreBinaria arv  $\leftarrow$  novo ArvoreBinaria();

ler(num);

```
arv.inserirNo(num);
```



```
ler(num);
```

```
arv.inserirNo(num);
```

```
ler(num);
```

```
arv.inserirNo(num);
```

```
ler(num);
```

```
arv.inserirNo(num);
```

```
ler(num);
```

```
arv.inserirNo(num);
```

```
arv.exibirNo();
```

**Fim\_algoritmo.**

---

—

**Gabarito Atividade Prática**

```
import javax.swing.*;
```



```
class BIntNo
```

```
{
```

```
    int valor;
```

```
    BIntNo esq, dir;
```

```
    BIntNo(int novoValor)
```

```
{
```

```
        valor = novoValor;
```

```
}
```

```
}
```

```
class ArvoreBinaria
```

```
{
```

```
    private BIntNo Raiz;
```

```
    private BIntNo inserir (BIntNo arvore, int novoNo)
```

```
{
```

```
        if (arvore == null)
```

```
{
```



```
    return new BIntNo (novoNo);
```

```
}
```

```
else
```

```
{
```

```
    if (novoNo < arvore.valor)
```

```
    {
```

```
        arvore.esq = inserir (arvore.esq, novoNo);
```

```
    }
```

```
else
```

```
{
```

```
    arvore.dir = inserir (arvore.dir, novoNo);
```

```
}
```

```
}
```

```
return arvore;
```

```
}
```

```
public void inserirNo (int novoValor)
```

```
{
```

```
Raiz = inserir(Raiz, novoValor);
```



```
}
```

```
private void exibirEsquerdo (BIntNo arv)
```

```
{
```

```
if (arv != null)
```

```
{
```

```
    exibirEsquerdo (arv.esq);
```

```
    System.out.println(arv.valor);
```

```
}
```

```
}
```

```
private void exibirDireito (BIntNo arv)
```

```
{
```

```
if (arv != null)
```

```
{
```

```
    exibirDireito(arv.dir);
```

```
    System.out.println (arv.valor);
```

```
}
```

```
}
```



```
public void exhibirRaiz()
```

```
{
```

```
    System.out.println("raiz " + Raiz.valor);
```

```
}
```

```
public void exhibirNoEsq()
```

```
{
```

```
    exhibirEsquerdo(Raiz);
```

```
}
```

```
public void exhibirNoDir()
```

```
{
```

```
    exhibirDireito(Raiz);
```

```
}
```

```
public void exhibirNo()
```

```
{
```

```
exibirNoEsq();
```



```
exibirRaiz();
```

```
exibirNoDir();
```

```
}
```

```
public void excluirNo (int item)
```

```
{
```

```
try
```

```
{
```

```
    BIntNo tempNo, pai, filho, temp;
```

```
    tempNo = Raiz;
```

```
    pai = null;
```

```
    filho = Raiz;
```

```
    while (tempNo != null && tempNo.valor != item)
```

```
    {
```

```
        pai = tempNo;
```

```
        if (item < tempNo.valor)
```

```
{
```



```
    tempNo = tempNo.esq;
```

```
}
```

```
else
```

```
{
```

```
    tempNo = tempNo.dir;
```

```
}
```

```
}
```

```
if (tempNo == null)
```

```
{
```

```
    System.out.println("item não localizado!");
```

```
}
```

```
if (pai == null)
```

```
{
```

```
    if (tempNo.dir == null)
```

```
{
```

```
        Raiz = tempNo.esq;
```

```
}
```

```
else
```



```
{
```



```
    if (tempNo.esq == null)
```

```
    {
```

```
        Raiz = tempNo.dir;
```

```
    }
```

```
    else
```

```
    {
```

```
        for (temp = tempNo, filho = tempNo.esq ; filho.dir != null ; temp = filho,  
filho = filho.dir);
```

```
        if (filho != tempNo.esq)
```

```
        {
```

```
            temp.dir = filho.esq;
```

```
            filho.esq = Raiz.esq;
```

```
        }
```

```
        filho.dir = Raiz.dir;
```

```
        Raiz = filho;
```

```
    }
```

```
}
```

```
}
```

else



{

if (tempNo.dir == null)

{

if (pai.esq == tempNo)

{

    pai.esq = tempNo.esq;

}

else

{

    pai.dir = tempNo.esq;

}

}

else

{

if (tempNo == null)

{

    if (pai.esq == tempNo)

{

```
    pai.esq = tempNo.dir;
```



```
    }
```

```
else
```

```
{
```

```
    pai.dir = tempNo.dir;
```

```
}
```

```
}
```

```
else
```

```
{
```

```
    for (temp = tempNo, filho = tempNo.esq ; filho.dir != null ; temp = filho,  
    filho = filho.dir);
```

```
    if (filho != tempNo.esq)
```

```
    {
```

```
        temp.dir = filho.esq;
```

```
        filho.esq = tempNo.esq;
```

```
    }
```

```
    filho.dir = tempNo.dir;
```

```
    if (pai.esq == tempNo)
```

```
    {
```

```
    pai.esq = filho;
```

```
    }
```

```
else
```

```
{
```

```
    pai.dir = filho;
```

```
    }
```

```
}
```

```
}
```

```
}
```

```
}
```

```
catch(NullPointerException erro)
```

```
{
```

```
    // item não encontrado
```

```
}
```

```
}
```

```
}
```

```
class teste
```

```
{
```



```
public static void main (String args [])
```



```
{
```

```
    ArvoreBinaria arv = new ArvoreBinaria();
```

```
        arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog("Digite um  
número inteiro"))));
```

```
        arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog("Digite um  
número inteiro"))));
```

```
        arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog("Digite um  
número inteiro"))));
```

```
        arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog("Digite um  
número inteiro"))));
```

```
        arv.inserirNo(Integer.parseInt( JOptionPane.showInputDialog("Digite um  
número inteiro"))));
```

```
    arv.exibirNo();
```

```
    System.exit(0);
```

```
}
```

```
}
```

**Ir para exercício**