Casos de Teste para o DOJO Árvore B+:

```
public class ArvoreBMaisTest {

   ArvoreBMais instance = null;

   Metadados tabMetadados = null;
   Metadados tabMetadadosSaida = null;

   List<NoInterno> tabIndice = null;
   List<NoInterno> tabIndiceSaida = null;

   List<NoFolha> tabDados = null;
   List<NoFolha> tabDadosSaida = null;

   private static final String NOME_ARQUIVO_INDICE = "indice.dat";
   private static final String NOME_ARQUIVO_DADOS = "clientes.dat";

   public ArvoreBMaisTest() {
   }

   @Before
   public void setUp() {
      instance = new ArvoreBMais();
      tabMetadados = new Metadados();
      tabMetadadosSaida = new Metadados();
      tabIndice = new ArrayList<NoInterno>();
      tabIndiceSaida = new ArrayList<NoInterno>();
      tabDados = new ArrayList<NoFolha>();
      tabDadosSaida = new ArrayList<NoFolha>();

      //deleta o arquivo da rodada anterior
      Arquivos.deletaArquivo(NOME_ARQUIVO_DADOS);
      Arquivos.deletaArquivo(NOME_ARQUIVO_INDICE);
   }

   /*
    * Arvore B+ de altura H = 1, com apenas 1 nó (raiz é uma folha) que tem 2 registros
    * Ver arquivo ArvoreTesteDOJO.pdf para o desenho da árvore
    */
   private void montaArvoreH1() throws FileNotFoundException, IOException {
      tabMetadados = new Metadados(0,true);
      Arquivos.salva(NOME_ARQUIVO_INDICE, tabMetadados);
      List<Cliente> clientes = new ArrayList<Cliente>();
      clientes.add(new Cliente(10,"JOAO      "));
      clientes.add(new Cliente(13,"ANA MARIA "));
      tabDados.add(new NoFolha(2,-1,-1,clientes));
      Arquivos.salva(NOME_ARQUIVO_DADOS, 0, tabDados);
   }
```

```
/*
   * Arvore B+ de altura H = 1, com apenas 1 nó (raiz é uma folha) que tem 4 registros
   * Ver arquivo ArvoreTesteDOJO.pdf para o desenho da árvore
   */
  private void montaArvoreH1Cheia() throws FileNotFoundException, IOException {
     tabMetadados = new Metadados(0,true);
     Arquivos.salva(NOME_ARQUIVO_INDICE, tabMetadados);
     List<Cliente> clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(10,"JOAO      "));
     clientes.add(new Cliente(13,"ANA MARIA "));
     clientes.add(new Cliente(15,"BIANCA    "));
     clientes.add(new Cliente(26,"CLARA     "));
     tabDados.add(new NoFolha(4,-1,-1,clientes));
     Arquivos.salva(NOME_ARQUIVO_DADOS, 0, tabDados);
  }

 /*
   * Arvore B+ com altura H = 2, com raiz e mais 3 folhas
   * Ordem de insercao das chaves na árvore: 10, 13, 15, 20, 25, 35, 37
   * Ver arquivo ArvoreTesteDOJO.pdf para o desenho da árvore
   */
  private void montaArvoreH2() throws FileNotFoundException, IOException {
     tabMetadados = new Metadados(Metadados.TAMANHO,false);
     Arquivos.salva(NOME_ARQUIVO_INDICE, tabMetadados);
     //Estrutura do No Interno: m, apontaFolha, pontPai, Lista de ponteiros, Lista de Chaves
     //ATENCAO: os ponteiros são absolutos
     tabIndice.add(new NoInterno(2,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO,2*NoFolha.TAMANHO),
Arrays.asList(15,25)));

     Arquivos.salva(NOME_ARQUIVO_INDICE, Metadados.TAMANHO, tabIndice);

     List<Cliente> clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(10,"JOAO      "));
     clientes.add(new Cliente(13,"ANA MARIA "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 1*NoFolha.TAMANHO, clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(15,"JOSE      "));
     clientes.add(new Cliente(20,"MARIANA   "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(25,"RONALDO   "));
     clientes.add(new Cliente(35,"MARCELA   "));
     clientes.add(new Cliente(37,"LEONARDO  "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

     Arquivos.salva(NOME_ARQUIVO_DADOS, 0, tabDados);
  }
```

```
  /*
   * Arvore B+ com altura H = 2, com raiz e mais 3 folhas, 1 delas cheia
   * Ordem de insercao das chaves na árvore: 10, 13, 15, 20, 25, 35, 37, 21, 23
   * Ver arquivo ArvoreTesteDOJO.pdf para o desenho da árvore
   */
  private void montaArvoreH2Cheia() throws FileNotFoundException, IOException {
    tabMetadados = new Metadados(Metadados.TAMANHO,false);
    Arquivos.salva(NOME_ARQUIVO_INDICE, tabMetadados);
    //Estrutura do No Interno: m, apontaFolha, pontPai, Lista de ponteiros, Lista de Chaves
    //ATENCAO: os ponteiros são absolutos
    tabIndice.add(new NoInterno(2,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO,2*NoFolha.TAMANHO),
Arrays.asList(15,25)));

    Arquivos.salva(NOME_ARQUIVO_INDICE, Metadados.TAMANHO, tabIndice);

    List<Cliente> clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(10,"JOAO      "));
    clientes.add(new Cliente(13,"ANA MARIA "));
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 1*NoFolha.TAMANHO, clientes));

    clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(15,"JOSE      "));
    clientes.add(new Cliente(20,"MARIANA   "));
    clientes.add(new Cliente(21,"BRUNA     "));
    clientes.add(new Cliente(23,"BRUNO     "));
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(4, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

    clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(25,"RONALDO   "));
    clientes.add(new Cliente(35,"MARCELA   "));
    clientes.add(new Cliente(37,"LEONARDO  "));
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

    Arquivos.salva(NOME_ARQUIVO_DADOS, 0, tabDados);
  }
```

```
 /*
   * Arvore B+ com altura H = 3, com raiz que aponta para 2 nós internos, e mais 6 folhas
   * Ordem de insercao das chaves na árvore: 10, 13, 15, 20, 25, 35, 37, 38, 39, 50, 55, 60, 70
   * Ver arquivo ArvoreTesteDOJO.pdf para o desenho da árvore
   */
  private void montaArvoreH3() throws FileNotFoundException, IOException {
     tabMetadados = new Metadados(Metadados.TAMANHO,false);
     Arquivos.salva(NOME_ARQUIVO_INDICE, tabMetadados);
     tabIndice.add(new NoInterno(1, false, -1, Arrays.asList(0, 1*NoInterno.TAMANHO), Arrays.asList(37)));
     tabIndice.add(new NoInterno(2, true, 0 + Metadados.TAMANHO, Arrays.asList(0, 1*NoFolha.TAMANHO,
2*NoFolha.TAMANHO), Arrays.asList(15, 25)));
     tabIndice.add(new NoInterno(2, true, 0 + Metadados.TAMANHO, Arrays.asList(3*NoFolha.TAMANHO,
4*NoFolha.TAMANHO, 5*NoFolha.TAMANHO), Arrays.asList(39, 55)));

     Arquivos.salva(NOME_ARQUIVO_INDICE, Metadados.TAMANHO, tabIndice);

     List<Cliente> clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(10,"JOAO      "));
     clientes.add(new Cliente(13,"ANA MARIA "));
     tabDados.add(new NoFolha(2,1 * NoInterno.TAMANHO + Metadados.TAMANHO,
1*NoFolha.TAMANHO,clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(15,"JOSE      "));
     clientes.add(new Cliente(20,"MARIANA   "));
     tabDados.add(new NoFolha(2, 1 * NoInterno.TAMANHO + Metadados.TAMANHO,
2*NoFolha.TAMANHO,clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(25,"RONALDO   "));
     clientes.add(new Cliente(35,"MARCELA   "));
     tabDados.add(new NoFolha(2, 1 * NoInterno.TAMANHO + Metadados.TAMANHO,
3*NoFolha.TAMANHO,clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(37,"LEONARDO  "));
     clientes.add(new Cliente(38,"KARLA     "));
     tabDados.add(new NoFolha(2, 2 * NoInterno.TAMANHO + Metadados.TAMANHO,
4*NoFolha.TAMANHO,clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(39,"MARIO     "));
     clientes.add(new Cliente(50,"RICARDO   "));
     tabDados.add(new NoFolha(2, 2 * NoInterno.TAMANHO + Metadados.TAMANHO,
5*NoFolha.TAMANHO,clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(55,"ROSA      "));
     clientes.add(new Cliente(60,"MICHELE   "));
     clientes.add(new Cliente(70,"RAFAEL    "));
```

```
    tabDados.add(new NoFolha(3, 2 * NoInterno.TAMANHO + Metadados.TAMANHO, -1,clientes));

    Arquivos.salva(NOME_ARQUIVO_DADOS, 0, tabDados);
  }

 /**
  * Testa o caso de Arvore B+ vazia
  */
 //@Test
 public void testaCriaArvoreBMaisVazia() throws FileNotFoundException, Exception {
    instance.criaArvoreBMais(NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    //Metadados
    tabMetadadosSaida = Arquivos.leMetadadosArvoreBMais(NOME_ARQUIVO_INDICE);
    assertEquals(0, tabMetadadosSaida.pontRaiz);
    assertEquals(true, tabMetadadosSaida.raizFolha);
    //NosInternos - vazio

    //Dados
    tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);
    tabDados.add(new NoFolha(0, -1, -1, null));
    assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
  }

 /**
  * Testa busca
  */
 //@Test
 public void testaBusca1() throws FileNotFoundException, Exception {
    montaArvoreH1();

    //Testa busca -- chave procurada está na raiz. Raiz é um nó folha
    ResultBusca result = instance.busca(13,NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(0, result.pontFolha);
    assertEquals(1, result.pos);
    assertEquals(true, result.encontrou);
  }

 //@Test
 public void testaBusca2() throws FileNotFoundException, Exception {
    montaArvoreH1();

    //Testa busca -- chave procurada não está na árvore. Raiz é um nó folha
    ResultBusca result = instance.busca(6,NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(0, result.pontFolha);
    assertEquals(0, result.pos);
    assertEquals(false, result.encontrou);
  }

 //@Test
 public void testaBusca3() throws FileNotFoundException, Exception {
    montaArvoreH2();

    //Testa busca -- chave está na árvore
```

```
    ResultBusca result = instance.busca(20,NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(1*NoFolha.TAMANHO, result.pontFolha);
    assertEquals(1, result.pos);
    assertEquals(true, result.encontrou);
}

//@Test
public void testaBusca4() throws FileNotFoundException, Exception {
    montaArvoreH2();

    //Testa busca -- chave não está na árvore
    ResultBusca result = instance.busca(16,NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(1*NoFolha.TAMANHO, result.pontFolha);
    assertEquals(1, result.pos);
    assertEquals(false, result.encontrou);
}

//@Test
public void testaBusca5() throws FileNotFoundException, Exception {
    montaArvoreH3();

    //Testa busca -- chave está na árvore
    ResultBusca result = instance.busca(20,NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(1*NoFolha.TAMANHO, result.pontFolha);
    assertEquals(1, result.pos);
    assertEquals(true, result.encontrou);
}

//@Test
public void testaBusca6() throws FileNotFoundException, Exception {
    montaArvoreH3();

    //Testa busca -- chave não está na árvore
    ResultBusca result = instance.busca(16,NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(1*NoFolha.TAMANHO, result.pontFolha);
    assertEquals(1, result.pos);
    assertEquals(false, result.encontrou);
}

//@Test
public void testaInsere1() throws FileNotFoundException, Exception {
    montaArvoreH1();

    int end = instance.insere(11,"VANESSA   ", NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(0, end);

    tabDados = new ArrayList<NoFolha>();
    List<Cliente> clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(10,"JOAO      "));
    clientes.add(new Cliente(11,"VANESSA   "));
    clientes.add(new Cliente(13,"ANA MARIA "));

    tabDados.add(new NoFolha(3,-1,-1,clientes));
```

```
        tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

        assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
    }


    //@Test
    public void testaInsere2() throws FileNotFoundException, Exception {
        montaArvoreH2();

        int end = instance.insere(11,"VANESSA   ", NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
        assertEquals(0, end);

        tabDados = new ArrayList<NoFolha>();
        tabIndice = new ArrayList<NoInterno>();

        tabIndice.add(new NoInterno(2,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO,2*NoFolha.TAMANHO),
Arrays.asList(15,25)));

        List<Cliente> clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente(10,"JOAO      "));
        clientes.add(new Cliente(11,"VANESSA   "));
        clientes.add(new Cliente(13,"ANA MARIA "));
        //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
        tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, 1*NoFolha.TAMANHO, clientes));

        clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente(15,"JOSE      "));
        clientes.add(new Cliente(20,"MARIANA   "));
        //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
        tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

        clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente(25,"RONALDO   "));
        clientes.add(new Cliente(35,"MARCELA   "));
        clientes.add(new Cliente(37,"LEONARDO  "));
        //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
        tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

        tabIndiceSaida = Arquivos.leNosInternos(NOME_ARQUIVO_INDICE);
        tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

        assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
        assertArrayEquals(tabIndice.toArray(), tabIndiceSaida.toArray());
    }
```

```
  /**
   * Testa inserção em árvore de altura H=2
   * Exige particionamento de uma página folha
   */
  //@Test
  public void testaInsere3() throws FileNotFoundException, Exception {
     //Árvore tem um dos nós folha cheio. E é neste nó que a inserção ocorrerá
     montaArvoreH2Cheia();

     int end = instance.insere(16,"VANESSA   ", NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
     assertEquals(1*NoFolha.TAMANHO, end);

     tabDados = new ArrayList<NoFolha>();
     tabIndice = new ArrayList<NoInterno>();

     tabIndice.add(new NoInterno(3,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO,3*NoFolha.TAMANHO,
2*NoFolha.TAMANHO), Arrays.asList(15,20,25)));

     List<Cliente> clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(10,"JOAO      "));
     clientes.add(new Cliente(13,"ANA MARIA "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 1*NoFolha.TAMANHO, clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(15,"JOSE      "));
     clientes.add(new Cliente(16,"VANESSA   "));

     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 3*NoFolha.TAMANHO, clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(25,"RONALDO   "));
     clientes.add(new Cliente(35,"MARCELA   "));
     clientes.add(new Cliente(37,"LEONARDO  "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

     //Nova página gerada pela operação do particionamento
     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(20,"MARIANA   "));
     clientes.add(new Cliente(21,"BRUNA     "));
     clientes.add(new Cliente(23,"BRUNO     "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));
```

```
        tabIndiceSaida = Arquivos.leNosInternos(NOME_ARQUIVO_INDICE);
        tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

        assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
        assertArrayEquals(tabIndice.toArray(), tabIndiceSaida.toArray());
    }


    /**
     * Testa inserção em árvore de altura H=2, chave do registro ja existe -- nao inserir
     */
    //@Test
    public void testaInsere4() throws FileNotFoundException, Exception {
        montaArvoreH2();

        int end = instance.insere(13,"MARIANA   ", NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
        assertEquals(-1, end);

        tabDados = new ArrayList<NoFolha>();
        tabIndice = new ArrayList<NoInterno>();

        tabIndice.add(new NoInterno(2,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO,2*NoFolha.TAMANHO),
Arrays.asList(15,25)));

        List<Cliente> clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente(10,"JOAO      "));
        clientes.add(new Cliente(13,"ANA MARIA "));
        //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
        tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 1*NoFolha.TAMANHO, clientes));

        clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente(15,"JOSE      "));
        clientes.add(new Cliente(20,"MARIANA   "));
        //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
        tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

        clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente(25,"RONALDO   "));
        clientes.add(new Cliente(35,"MARCELA   "));
        clientes.add(new Cliente(37,"LEONARDO  "));
        //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
        tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

        tabIndiceSaida = Arquivos.leNosInternos(NOME_ARQUIVO_INDICE);
        tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

        assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
        assertArrayEquals(tabIndice.toArray(), tabIndiceSaida.toArray());
    }

/**
     * Testa inserção em árvore de altura H=1 cheia, que causa aumento na altura da árvore
     */
```

```
  @Test
  public void testaInsere5() throws FileNotFoundException, Exception {
     montaArvoreH1Cheia();

     int end = instance.insere(11,"VANESSA   ", NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
     assertEquals(0, end);

     tabDados = new ArrayList<NoFolha>();
     tabIndice = new ArrayList<NoInterno>();

     tabIndice.add(new NoInterno(1,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO), Arrays.asList(13)));

     List<Cliente> clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(10,"JOAO      "));
     clientes.add(new Cliente(11,"VANESSA   "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 1*NoFolha.TAMANHO, clientes));

     clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(13,"ANA MARIA "));
     clientes.add(new Cliente(15,"BIANCA    "));
     clientes.add(new Cliente(26,"CLARA     "));
     //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
     tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

     tabIndiceSaida = Arquivos.leNosInternos(NOME_ARQUIVO_INDICE);
     tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

     assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
     assertArrayEquals(tabIndice.toArray(), tabIndiceSaida.toArray());
  }

 /**
   * Testa exclusão em árvore de altura H=2
   * Não é necessário concatenação
   */
  //@Test
  public void testaExclui1() throws FileNotFoundException, Exception {
     //Árvore tem um dos nós folha cheio. E é neste nó que a exclusão ocorrerá
     montaArvoreH2Cheia();

     int end = instance.exclui(20, NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
     assertEquals(1*NoFolha.TAMANHO, end);

     tabDados = new ArrayList<NoFolha>();
     tabIndice = new ArrayList<NoInterno>();

     tabIndice.add(new NoInterno(2,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO,2*NoFolha.TAMANHO),
Arrays.asList(15,25)));

     List<Cliente> clientes = new ArrayList<Cliente>();
     clientes.add(new Cliente(10,"JOAO      "));
     clientes.add(new Cliente(13,"ANA MARIA "));
```

```
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 1*NoFolha.TAMANHO, clientes));

    clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(15,"JOSE      "));
    clientes.add(new Cliente(21,"BRUNA     "));
    clientes.add(new Cliente(23,"BRUNO     "));
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

    clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(25,"RONALDO   "));
    clientes.add(new Cliente(35,"MARCELA   "));
    clientes.add(new Cliente(37,"LEONARDO  "));
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

    tabIndiceSaida = Arquivos.leNosInternos(NOME_ARQUIVO_INDICE);
    tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

    assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
    assertArrayEquals(tabIndice.toArray(), tabIndiceSaida.toArray());
}
```

```
/**
 * Testa exclusão em árvore de altura H=2
 * É necessário concatenação
 */
//@Test
public void testaExclui2() throws FileNotFoundException, Exception {
    montaArvoreH2();

    int end = instance.exclui(20, NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
    assertEquals(1*NoFolha.TAMANHO, end);

    tabDados = new ArrayList<NoFolha>();
    tabIndice = new ArrayList<NoInterno>();

    tabIndice.add(new NoInterno(1,true,-1,Arrays.asList(0,2*NoFolha.TAMANHO), Arrays.asList(25)));

    List<Cliente> clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(10,"JOAO      "));
    clientes.add(new Cliente(13,"ANA MARIA "));
    clientes.add(new Cliente(15,"JOSE      "));
```

```
      //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
      tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

      //Este nó agora é lixo. Dependendo da implementação, tem que mudar este código de teste.
      clientes = new ArrayList<Cliente>();
      clientes.add(new Cliente(15,"JOSE      "));
      clientes.add(new Cliente(20,"MARIANA   "));
      //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
      tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

      clientes = new ArrayList<Cliente>();
      clientes.add(new Cliente(25,"RONALDO   "));
      clientes.add(new Cliente(35,"MARCELA   "));
      clientes.add(new Cliente(37,"LEONARDO  "));
      //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
      tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

      tabIndiceSaida = Arquivos.leNosInternos(NOME_ARQUIVO_INDICE);
      tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

      assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
      assertArrayEquals(tabIndice.toArray(), tabIndiceSaida.toArray());
   }
}

/**
    * Testa exclusão em árvore de altura H=2
    * É necessário redistribuição
    */
   //@Test
   public void testaExclui3() throws FileNotFoundException, Exception {
      montaArvoreH2Cheia();

      int end = instance.exclui(13, NOME_ARQUIVO_INDICE, NOME_ARQUIVO_DADOS);
      assertEquals(0, end);

      tabDados = new ArrayList<NoFolha>();
      tabIndice = new ArrayList<NoInterno>();

      tabIndice.add(new NoInterno(2,true,-1,Arrays.asList(0,1*NoFolha.TAMANHO,2*NoFolha.TAMANHO),
Arrays.asList(20,25)));

      List<Cliente> clientes = new ArrayList<Cliente>();
      clientes.add(new Cliente(10,"JOAO      "));
      clientes.add(new Cliente(15,"JOSE      "));
      //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
      tabDados.add(new NoFolha(2, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

      //Este nó agora é lixo. Dependendo da implementação, tem que mudar este código de teste.
      clientes = new ArrayList<Cliente>();
      clientes.add(new Cliente(20,"MARIANA   "));
      clientes.add(new Cliente(21,"BRUNA     "));
      clientes.add(new Cliente(23,"BRUNO     "));
```

```
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, 2*NoFolha.TAMANHO, clientes));

    clientes = new ArrayList<Cliente>();
    clientes.add(new Cliente(25,"RONALDO   "));
    clientes.add(new Cliente(35,"MARCELA   "));
    clientes.add(new Cliente(37,"LEONARDO  "));
    //Estrutura do nó folha: m, pontPai, pontProx, registros de clientes
    tabDados.add(new NoFolha(3, 0+Metadados.TAMANHO, -1,clientes));

    tabIndiceSaida = Arquivos.leNosInternos(NOME_ARQUIVO_INDICE);
    tabDadosSaida = Arquivos.leNosFolha(NOME_ARQUIVO_DADOS);

    assertArrayEquals(tabDados.toArray(), tabDadosSaida.toArray());
    assertArrayEquals(tabIndice.toArray(), tabIndiceSaida.toArray());
}
```