Prática de Laboratório 7

Objetivos:

- Tipos Abstratos de Dados - TADs.
- Implementação de Árvore Binária de Pesquia.

1. Implemente a estrutura de dados Arvore Binária conforme abaixo:

Nodo

```
public class Nodo {
    private Integer valor;
    private Nodo esquerda;
    private Nodo direita;

    public Nodo(int valor) {
        this.valor = valor;
        esquerda = null;
        direita = null;
    }

    public void setValor(Integer valor) {
        this.valor = valor;
    }

    public int getValor() {
        return valor;
    }

    public Nodo getEsquerda() {
        return this.esquerda;
    }

    public void setEsquerda(Nodo esq) {
        this.esquerda = esq;
    }

    public Nodo getDireita() {
        return direita;
    }

    public void setDireita(Nodo direita) {
        this.direita = direita;
    }

}
```

Árvore Binária

```
3  public class ArvoreBinaria {
4
5      private Nodo raiz;
6
7      public ArvoreBinaria() {
8          this.raiz = null;
9      }
10
11     public boolean Vazia() {
12         return this.raiz == null;
13     }
14
15     public void insere(Integer reg) {
16         this.raiz = this.insere(reg, this.raiz);
17     }
18
19     private Nodo insere(Integer reg, Nodo p) {
20         if (p == null) {
21             p = new Nodo(reg);
22         } else if (reg < p.getValor())
23             p.setEsquerda(insere(reg, p.getEsquerda()));
24         else if (reg > p.getValor())
25             p.setDireita(insere(reg, p.getDireita()));
26         else
27             System.out.println("Erro: Registro ja existente");
28         return p;
29     }
30
31     public void EmOrdem(Nodo R) {
32         if (R == null) {
33             return;
34         }
35
36         if (R.getEsquerda() != null) {
37             EmOrdem(R.getEsquerda());
38         }
39
40         System.out.println("Valor: " + R.getValor());
41
42         if (R.getDireita() != null) {
43             EmOrdem(R.getDireita());
44         }
45     }
46
47     public Nodo getRaiz() {
48         return raiz;
49     }
```

(continuação da classe Árvore Binária)

```
51      private Nodo antecessor(Nodo q, Nodo r) {
52          if (r.getDireita() != null)
53              r.setDireita(antecessor(q, r.getDireita()));
54          else {
55              q.setValor(r.getValor());
56              r = r.getEsquerda();
57          }
58          return r;
59      }
60
61      private Nodo retira(Integer reg, Nodo p) {
62          if (p == null)
63              System.out.println("Erro: Registro nao encontrado");
64          else if (reg < p.getValor())
65              p.setEsquerda(retira(reg, p.getEsquerda()));
66          else if (reg > p.getValor())
67              p.setDireita(retira(reg, p.getDireita()));
68          else {
69              if (p.getDireita() == null)
70                  p = p.getEsquerda();
71              else if (p.getEsquerda() == null)
72                  p = p.getDireita();
73              else
74                  p.setEsquerda(antecessor(p, p.getEsquerda()));
75          }
76          return p;
77      }
78
79      public void retira(Integer reg) {
80          this.raiz = this.retira(reg, this.raiz);
81      }
82
83
84      public void imprimeEmOrdem() {
85          this.EmOrdem(this.raiz);
86      }
87
88
89  }
90
```

Exercícios:

1. Implemente funções que executem o caminhamento na árvore “Pós-Ordem” e “Pré-Ordem”.
2. Customize a árvore binária para incluir dados de uma classe Aluno.
    a. Classe aluno

    Aluno:

    Nome
    Matrícula
    Curso

    b. Considere os valores a serem considerados para inserção na árvore a matrícula. Lembre-se que não deve ser possível inserir dois alunos com a mesma matrícula.

Bom trabalho!