Tópicos Avançados em Estrutura de Dados

Atividade 10

Bruna Galastri Guedes	18.00189-0
Daniel Ughini Xavier	18.00022-3
Rodolfo Cochi Bezerra	18.00202-0
Vítor Martin Simoni	18.00050-9
Leonardo Cury Haddad	18.00442-3
Leonardo de Barros Rodrigues	18.02401-7

01/08/2020

Programa principal:

```
package br.maua;

import br.maua.arvore.Arvore;
import br.maua.arvore.ArvoreUtils;
import br.maua.arvore.No;

public class Main {

    /*
    Grupo
    Bruna G.Guede - 18.00189-0
    Daniel U. Xavier - 18.00022-3
    Rodolfo C. Bezerra - 18.00202-0
    Vitor M. Simoni - 18.00050-9
    Leonardo Cury Haddad - 18.00442-3
    Leonardo de Barros Rodrigues - 18.02401-7
     */

    public static void main(String[] args) {

        //Cria os nós da árvore do exercicio
        No no5 = new No(5);
        No no0 = new No(0);
        No no1 = new No(1);
        No no2 = new No(2);
        No no3 = new No(3);
        No no4 = new No(4);
        No no6 = new No(6);
        No no7 = new No(7);
        No no8 = new No(8);
        No no21 = new No(21);
        No no12 = new No(12);

        //Cria a árvore
        Arvore arvore = new Arvore(no5);

        //Monta a árvore
        arvore.adicionarNo(no5,no4);
        arvore.adicionarNo(no5,no3);
        arvore.adicionarNo(no5,no7);
        arvore.adicionarNo(no4,no1);
        arvore.adicionarNo(no4,no2);
        arvore.adicionarNo(no3,no8);
        arvore.adicionarNo(no7,no0);
        arvore.adicionarNo(no7,no6);
        arvore.adicionarNo(no8,no21);
        arvore.adicionarNo(no8,no12);

        //Roda alguns dos métodos
        System.out.println("---- Preorder ----");
        ArvoreUtils.preorder(no5);
        System.out.println("---- Posorder ----");
        ArvoreUtils.posorder(no5);
```

```
        System.out.println("----------------------");
        System.out.println("---- Filhos do 5 ----");
        ArvoreUtils.imprimeFilhos(no5);
        System.out.println("---- Pai do 8 ----");
        ArvoreUtils.imprimePai(no8);

        ArvoreUtils.dobraFilhos(no5);
        System.out.println("---- Filhos do 8 dobrados ----");
        System.out.println("-------   Filhos do 8   ------");
        ArvoreUtils.imprimeFilhos(no5);

        ArvoreUtils.dobraPai(no2);
        System.out.println("---- Pai do 2 dobrado ----");
        System.out.println("------   Pai do 2   ------");
        ArvoreUtils.imprimePai(no2);
        System.out.println("*Repare que o valor já havia sido dobrado anteriormente");

    }
}
```

Classe árvore:

```
package br.maua.arvore;

public class Arvore {

    /*
    A classe Arvore força uma raiz
     */

    public Arvore(No raiz) {
        this.raiz = raiz;
    }

    private No raiz; //Raiz da arvore

    public void adicionarNo(No pai, No filho){
        pai.adicionarFilho(filho);
    }

}
```

Classe ÁrvoreUtils:

```
package br.maua.arvore;


public class ArvoreUtils {

    //As funções/métodos pedidos no exercício estão definidas nessa classe

    public static void imprimeFilhos(No no){
        no.pegarFilhos().forEach(System.out::println);
    }
```

```
    public static No pai(No no){
        return no.pegarPai();
    }

    public static void imprimePai(No no){
        System.out.println(pai(no));
    }

    public static boolean ehInterno(No no){
        return !(no.pegarFilhos().isEmpty());
    }

    public static void imprimeFilhosFolhas(No no){
        no.pegarFilhos().forEach(filho->{
            if(!ehInterno(filho))System.out.println(filho);
        });
    }


    public static void dobraFilhos(No no ){
        no.pegarFilhos().forEach(filho->{
            filho.dado = filho.dado*2;
        });
    }

    public static void dobraPai(No no ){
        no.pegarPai().dado = no.pegarPai().dado*2;
    }

    public static void preorder(No no ){
        System.out.println(no);
        no.pegarFilhos().forEach(ArvoreUtils::preorder);
    }

    public static void posorder(No no ){
        imprimeFilhosFolhas(no);
        no.pegarFilhos().forEach(ArvoreUtils::posorder);
        if (ehInterno(no)) System.out.println(no);
    }
}
```

Classe Nó:

```
package br.maua.arvore;

import java.util.ArrayList;

public class No {

    public int dado; //Dado guardado no nó

    private No pai; //Pai

    private ArrayList<No> filhos = new ArrayList<>(); //Lista contendo os filhos

    //Construtor (define o dado)
```

```
    public No(int dado) {
        this.dado = dado;
    }

    //Getter para o pai
    public No pegarPai(){
        return this.pai;
    }

    //Getter p/ filhos
    public ArrayList<No> pegarFilhos() {
        return filhos;
    }

    //Adiciona um filho e define o pai do filho como esse nó
    protected void adicionarFilho(No filho){
        this.filhos.add(filho);
        filho.definirPai(this);
    }

    //Método privado, só é acessado pelo método adicionarFilho
    private void definirPai(No pai){
        this.pai = pai;
    }

    //Define a impressão do objeto nó como o dado guardado
    @Override
    public String toString() {
        return Integer.toString(dado);
    }

}
```

Questão 1

a)

```
    public static void imprimeFilhos(No no){
        no.pegarFilhos().forEach(System.out::println);
    }
```

b)

```
     public static No pai(No no){
        return no.pegarPai();
    }
```

c)

```
        public static void imprimePai(No no){
        System.out.println(pai(no));
    }
```

d)

```
    public static boolean ehInterno(No no){
    return !(no.pegarFilhos().isEmpty());
}
```

e)

```
    public static void imprimeFilhosFolhas(No no){
    no.pegarFilhos().forEach(filho->{
        if(!ehInterno(filho))System.out.println(filho);
    });
}
```

f)

```
    public static void preorder(No no ){
    System.out.println(no);
    no.pegarFilhos().forEach(ArvoreUtils::preorder);
}
```

g)

```
public static void posorder(No no ){
    imprimeFilhosFolhas(no);
    no.pegarFilhos().forEach(ArvoreUtils::posorder);
    if (ehInterno(no)) System.out.println(no);
}
```

Questão 2

a)

```
---------------------
---- Filhos do 5 ----
4
3
7
```

b)

```
    ---- Pai do 8 ----
3
```

c)

```
    ---- Filhos do 8 dobrados ----
-------   Filhos do 8   ------
8
6
14
```

d)

```
    ---- Pai do 2 dobrado ----
------   Pai do 2   ------
16
```