# 节点树<T>说明

问 9 年零 5 个月前　活跃 6 年零 9 个月前　浏览了 23,000 次

5

3

不知道根据网站的规则我是否可以这样做……但我会抓住机会……请容忍我，我只是一个学生……:-)

我有一个大学作业……我很难理解课程应该做什么……我已经在三个不同的场合去找过我的老师，我从他那里得到的答案根本没有帮助。无论如何，分配细节如下...

创建一个称为 `Tree` 节点的容器的类。树类应支持以下方法。

**public void add(Node parent, Node child){}** -- 添加一个新的子节点到父节点

**public void removeChild(Nodeparent, Node child){}** -- 从父节点中删除子节点。

**public Node getRootNode(){}** -- 返回树的根

**public void setRoot(Node root){}** -- 设置树的根节点

**public boolean contains(T data){}** -- 在树中搜索给定类型

**public void dfs(Node child){}** -- 对树执行深度优先搜索并输出每个节点（缩进）

**public void bfs(Node child){}** -- 对树执行广度优先搜索并输出每个节点（缩进）

1. 树类应该被参数化以处理泛型类型 T，允许创建字符串、文件等的树，例如 `Tree<String> tree = new Tree<String>()`
2. 树类应使用邻接表实现树结构，并按以下方式定义： `Map<Node<T>, List<Node<T>>> tree = new HashMap<Node<T>, List<Node<T>>();`

节点类也应该被参数化以处理泛型类型 T 并公开几个方法……

现在我已经编写了我的 Node 类，它工作正常……老实说，我确信我已经编写了一个创建树的 Node 类。但在阅读了 Tree 类的描述后，我很困惑。我应该在树图中存储什么。我很难想象整个事情。

也许有人可以解释老师想要什么，让我朝着正确的方向前进。我**不是**在寻找代码本身……只是想了解我应该做什么。

我的节点类

```
public class Node<T>
{
    private Node<T> root; // a T type variable to store the root of the list
    private Node<T> parent; // a T type variable to store the parent of the list
    private T child;
```

**加入 Stack Overflow**学习、分享知识并建立您的职业生涯。 Sign up ×

```
public Node(T child)
{
    setParent(null);
    setRoot(null);
    setItem(child);
}

// constructor overloading to set the parent
public Node(Node<T> parent)
{
    this.setParent(parent);
    //this.addChild(parent);
}

// constructor overloading to set the parent of the list
public Node(Node<T> parent, Node<T> child)
{
    this(parent);
    this.children.add(child);
}

/**
* This method doesn't return anything and takes a parameter of
* the object type you are trying to store in the node
*
* @param  Obj  an object
* @param
**/
public void addChild(Node<T> child)
{
    child.root = null;
    child.setParent((Node<T>)this);
    this.children.add(child); // add this child to the list
}

public void removeChild(Node<T> child)
{
    this.children.remove(child); // remove this child from the list
}

public Node<T> getRoot() {
    return root;
}

public boolean isRoot()
{
    // check to see if the root is null if yes then return true else return false
    return this.root != null;
}

public void setRoot(Node<T> root) {
    this.root = root;
}

public Node<T> getParent() {
    return parent;
}

public void setParent(Node<T> parent) {
    this.parent = parent;
}

public T getItem() {
    return child;
}
```

**加入 Stack Overflow**学习、分享知识并建立您的职业生涯。

Sign up

×

```
    public boolean hasChildren()
    {
        return this.children.size()>0;
    }

    @SuppressWarnings("unchecked")
    public Node<T>[] children()
    {
        return (Node<T>[]) children.toArray(new Node[children.size()]);
    }

    @SuppressWarnings({ "unchecked"})
    public Node<T>[] getSiblings()
    {
        if(this.isRoot()!=false && parent==null)
        {
            System.out.println("this is root or there are no siblings");
            return null;
        }
        else{
            List<Node<T>> siblings = new ArrayList<Node<T>>((Collection<? extends
Node<T>>) Arrays.asList(new Node[this.parent.children.size()]));
            Collections.copy(siblings, this.parent.children);
            siblings.remove(this);
            return siblings.toArray(new Node[siblings.size()]);
        }
    }
}
```

爪哇 树

分享 改进这个问题 跟随

20 年 6 月 20 日 9:12编辑

社区 机器人

1 1

问2012 年 4 月 24 日 1:36

吉拉尼

405 1 5 19

8 “裸”在你身边？我一直穿着衣服——这是“忍受我”。 —— 达菲莫 2012 年 4 月 24 日 1:38

一个有用的区别：你是在处理这种树，还是这种树？我已经猜到了前者，但使用邻接表的要求暗示了后者。 —— 泰蒙 2012 年 4 月 24 日 1:47

我认为您提供的第一个链接看起来与我们所讲授的内容非常接近..... —— 吉拉尼 2012 年 4 月 24 日 2:01

## 3 个回答

积极的 | 最老的 | 投票

8

您将地图用于以下用途：

hashmap 的键是给定的节点，hashmap 的值是给定节点的子节点。

```
public class Tree<T> {
    private Node<T> rootNode;
    private HashMap<Node<T>, List<Node<T>> tree;
```

加入 Stack Overflow学习、分享知识并建立您的职业生涯。 Sign up

```
                System.out.println(node);
                return;
            }
            for(Node<T> n : tree.get(node)) {
                System.out.println(node);
                expandNode(n);
            }
        }
    }
```

我能说清楚这棵树是如何工作的吗？？

分享 改进这个答案 跟随

15 年 1 月 2 日 6:32编辑



斯宾塞·维佐雷克
**20.1k** 7 39 50

2012 年4 月 24 日 1:41回答

胡安·阿尔贝托·洛佩斯·卡瓦洛蒂
**4,536** 3 21 42

嗯，在我的 Node 类中，我创建了一个方法，该方法返回数组中的子项.....所以我应该怎么做，将 node<t> 作为键，将节点的子项作为数组..... —— 吉拉尼 2012 年 4 月 24 日 1:45

1 这样想，哈希映射是整个树，您可以为自己保留对根节点的引用，然后它的工作原理是这样的，具有给定的节点，您调用哈希映射的 get 方法来获取其子节点该节点，因此您可以遍历子节点并使用相同的映射继续检索节点的子节点，当 get 方法返回 null 时，您就有了叶节点。
—— 胡安·阿尔贝托·洛佩斯·卡瓦洛蒂 2012 年 4 月 24 日 1:48

还有另一个细节，要使其工作，您应该在 Node 类中实现 equals 和 hashcode 方法。
—— 胡安·阿尔贝托·洛佩斯·卡瓦洛蒂 2012 年 4 月 24 日 1:49

嗯...好吧，我想我会按照你给我的想法工作...并将在我的节点类中实现哈希码和 equals 方法...但我不知道我是什么类型的对象将存储在节点中.....我可以根据什么生成我的哈希码？和equals方法？ #
—— 吉拉尼 2012 年 4 月 24 日 2:06

您可以在托管对象上调用 hashCode，因为这是从 Object 类继承的方法
—— 胡安·阿尔贝托·洛佩斯·卡瓦洛蒂 2012 年 4 月 24 日 2:08

0

查看列表中的 2 点，我猜想第 1 点对您来说是最令人困惑的。

树本身可以参数化。树类上的类型参数可以在用于创建节点的内部类中使用。也就是说，就您的赋值而言，节点类可能应该在 Tree 类内部，以便它可以使用赋予 Tree 类的 T 类型参数。

这样，树就可以存储任何东西（字符串、文件、双打等）。Node 类只是作为存储任何类型 T 对象的好方法。

分享 改进这个答案 跟随

2012 年4 月 24 日 1:44回答

凯文628
**3,350** 3 24 43

你对混淆部分是对的..... :-) 在描述中，我们被要求为节点编写一个单独的类.....我已经写了 —— 吉拉尼 2012 年 4 月 24 日 1:51

**加入 Stack Overflow**学习、分享知识并建立您的职业生涯。 Sign up ×

0

由于看起来您不应该这样做，因此您可以创建 `Tree<T>` 一个包含对单个 `Node<T>` 对象的引用的包装类。您可以将所需方法的逻辑放在任一类中。

代码的开头可能是这样的：

```
public class Tree<T> {

    private Node<T> root;

    public Tree(Node<T> root) {
        this.root = root;
    }

}
```

分享 改进这个答案 跟随

2012 年4 月 24 日 2:08编辑

2012 年4 月 24 日 1:49回答

泰蒙

**22.8k** 8 58 81

描述清楚地表明 Tree 类应该支持我在我原来的帖子中写的方法……但是你能给我一个 Tree 类的初始化语句的例子……？ 拜托……只是想弄清楚你说的话……另一位胡安先生也很有道理…… —— 吉拉尼 2012 年 4 月 24 日 1:56

这正是我开始编写 Tree 类的方式，但后来对整个 Map 问题感到困惑…… —— 吉拉尼 2012 年 4 月 24 日 2:11

啊，没关系，我一直在想这一切都是错误的。我假设边缘应该是节点状态的一部分。他们不是。 —— 泰蒙 2012 年 4 月 24 日 2:15

**加入 Stack Overflow**学习、分享知识并建立您的职业生涯。

Sign up

×