Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 04 REPORT

EMIRE KORKMAZ
141044043

M. BURAK KOCA

1 INTRODUCTION

1.1 Problem Definition

General Tree yapısını BinaryTree class'ını extend ederek implement etmemiz ve bazı traverse methodları yazmamız isteniyor.

1.2 System Requirements

```
public void preOrderTraverse(MyNode<E> myNode);
```

Bu method traverse etmeye başlayacağı MyNode tipinde bir parametre alıyor. MyNode tipindeki objede bulunan children isimli vector yapısındaki listede objenin çocuklarını tuttum. Bu method objeyi ve daha sonra çocuklarını traverse ediyor. Bunu da her çocuğuna geldiğinde methodu o andaki çocuğu parametre olarak vererek bir daha çağırarak sağlıyor. Objeye her gelişinde de objenin data'sını ekrana basıyor.

Complexity : O(n)

```
public boolean postOrderSearch(MyNode<E> myNode, E target);
```

Bu method parametre olarak verilen objeyi kullanarak target bir değer alıp tree yapısında arıyor. Bulursa true, bulmazsa false return ediyor. Current adında geçici bir oje oluşturup root'u current'a atadım. current'ı kullanarak tree üzerinde dolaştım. Bir döngü içerisinde current'ın her bir çocuğunda yine postOrderSearch methodunu çağırdım ve karşılaştırma yaptım aranan değer bulunduğunda true döndürdün. Döngüden sonra artık hiçbir şey bulunmamış olacağı için false döndürdüm.

Complexity : O(n)

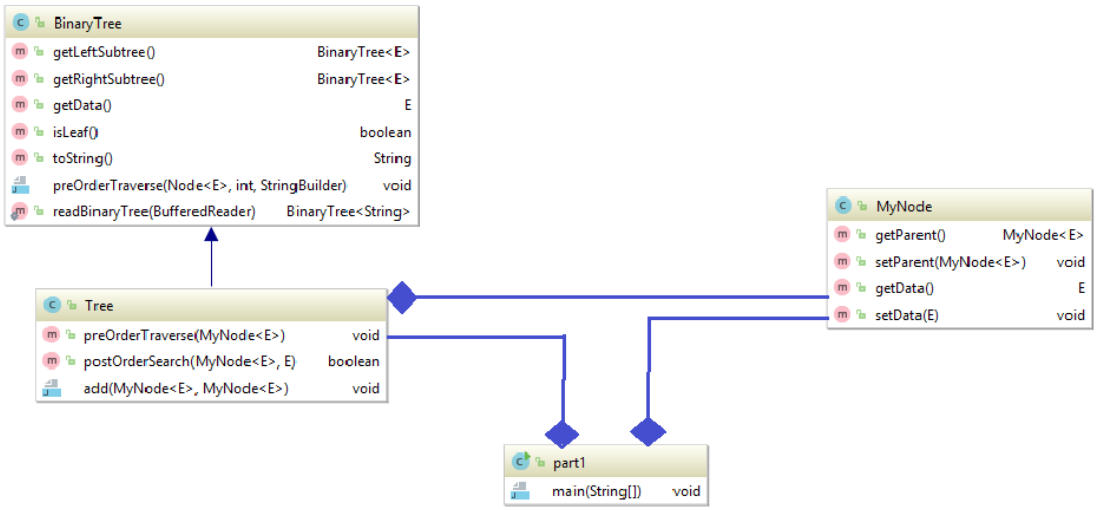
```
public void add(MyNode<E> p, MyNode<E> c);
```

Bu method, bir parent ve bir child değeri alarak tree'ye ekliyor. Eğer root null'sa yani tree'de başka bir node yoksa root'a parent'ı atıyor ve child'ı da root'un çocuğu olark set ediyor. Eğer root null değilse yani tree boş değilse verilen parent'ın tree'de olup olmadığını postOrderSearch methoduyla kontrol ediyor. Eğer tree'de yer alıyorsa verilen parent'ın çocucğu olarak child.'ı set ediyor. Child'ın da parent'ı olarak verilen parent'ı set ediyor.

Complexity : tetha(n)

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

Add use case diagrams if required.

2.3 Other Diagrams (optional)

Add other diagrams if required.

2.4 Problem Solution Approach

Tree'deki her bir node'u tutmak için MyNode adındaki kendi node yapımı oluşturdum. Bu yapıda yine MyNode tipindeki parent ve yine MyNode tipindeki children isimli bir vector tanımladım. Children bir node'un bütün çocuklarını tutuyor. PreOrderTraverse methodu bütün tree'yi gezerek önce sol tarafı sonra sağ tarafı daha sonra da root'u gezip ekrana basıyor. PostOrderSearch methodu verilen E tipindeki bir değeri tree'de arıyor. Bulursa true, diğer durumlarda false döndürüyor. Add methodu bir parent ve bir child alıp child'ı parent'ın çoçuğu olarak tree'ye ekliyor.

3 RESULT

3.1 Test Cases

Give test cases.

3.2 Running Results

```
public class part1 {
    public static void main(String[] args) {
        MyNode<String> node = new MyNode<>( data: "William");
        MyNode<String> node2 = new MyNode<>( data: "Marshall");
        MyNode<String> node3 = new MyNode<>( data: "Ted");
        MyNode<String> node4 = new MyNode<>( data: "Barney");
        MyNode<String> node5 = new MyNode<>( data: "Lily");
        MyNode<String> node6 = new MyNode<>( data: "Robin");
        MyNode<String> node7 = new MyNode<>( data: "Ranjit");

        Tree<String> tree = new Tree<~>();
        tree.add(node4, node2);
        tree.add(node2, node3);
        tree.add(node2, node);
        tree.add(node2, node5);
        tree.add(node5, node6);
        tree.add(node5, node7);

        tree.preOrderTraverse(node4);
```

part1 › main()

part1

```
"C:\Program Files\Java\jdk1.8.0_161\bin\java" ...
Barney Marshall Ted William Lily Robin Ranjit
Process finished with exit code 0
```