

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

**CELAL CAN KAYA
161044014**

Course Assistant: Mehmet Burak KOCA

1 INTRODUCTION

1.1 Problem Definition

Part 1’de bir General Tree’nin sol kolda çocuk(child), sağ kolda kardeş(sibling) olan bir Binary Tree şeklinde tutulması istenmektedir. Bunu yaparken BinaryTree sınıfının extend edilmesi ve add(), levelOrderSearch() ve postOrderSearch(), metodlarını eklememiz ve preOrderTraverse() metodunu override etmemiz istendi.

1.2 System Requirements

Part 1) Part1 sınıfı tipinde bir obje oluşturulmalı ve bu obje add() metodu çağırılarak doldurulmalıdır.add() metoduna 2 parametre gönderilmeli, ilk parametre parentItem, ikinci parametre childItem olmalıdır.add() metodu içerisinde parentItem’in var olup olmadığı kontrol edilirken levelOrder veya postOrder search kullanılmalıdır.Eklenen kodda levelOrderSearch kullanılarak parentItem aranmıştır.levelOrderSearch kısmı parametreler değiştirilmeden postOrderSearch’a çevirilerek postOrderSearch kullanılarakta parentItem aratılabilir.levelOrder ve postOrderSearch’ta ilk parametreye root Node’u gönderilmeli, ikinci parametreye aranan eleman gönderilmelidir.

Part 1 sınıfı, BinaryTree sınıfından extend edilmiştir. Part 2 sınıfı SearchTree sınıfını implement

edip aynı zamanda BinaryTree sınıfından extend edilmiştir.

2.2 Use Case Diagrams

Use Case diyagram gerekmediği için eklemedim.

2.3 Other Diagrams (optional)

Add other diagrams if required.

2.4 Problem Solution Approach

Part 1’de add() metodu için öncelikle Root Node’unun dolu yada boş olduğu kontrol ettim.Eğer boşsa verilen childItem’i Root Node’una attım.Root Node’unun dolu olduğu zamanlarda parentItem’in ağaç içerisinde yer alıp almadığını yazdığım search metodlarından biriyle kontrol ettim.**add Metodu içerisinde çağırılan levelOrderSearch kısmı postOrderSearch yapılarak postOrderSearch her ekleme metodu çağırıldığında tekrardan kontrol edilebilir.**Eğer parentItem ağaç içerisinde yoksa eleman eklenemeyeceğinden “false” return ettim.ParentItem ağaç içerisinde varsa gerekli Node’a gidip childItem’i ekledim.

LevelOrderSearch ve PostOrderSearch metodu için recursive şekilde ağacı taradım ve elemanı bulduktan sonraki diğer elemanları kontrol etmeden bulunan Node’u return ettim. Ayrıca

LevelOrderSearch’ın düzgün çalışması için Queue yapısını kullandım.

PreOrderTraverse metodunda derinlik konusunda ufak bir sıkıntı olduğu için BinaryTree sınıfının preOrderTraverse’sinde ufak bir değişiklik yaparak kullandım.

Zaman yetişmediğinden dolayı Part 2’de sadece add() metodunu implement edebildim.

2.5 Algorithm Analysis

PART 1)

levelOrderSearch() : İlk elemanda direk bulabileceği için $\Omega(1)$ veya hiç bulamayacağı durumda $O(n)$ olacaktır.

postOrderSearch() : İlk elemanda direk bulabileceği için $\Omega(1)$ veya hiç bulamayacağı durumda $O(n)$ olacaktır.

Add() : Fonksiyon içerisinde levelOrder veya postOrderSearch çağırılacağı için $O(n)$ olacaktır.

preOrderTraverse() : Tek elemanlı bir ağaç olabileceği için $\Omega(1)$, 1’den fazla elemanı olan ağaçlar için $O(n)$ olacaktır.

3 RESULT

3.1 Test Cases

```
@Test
public void add() {
    Part1<Integer> a = new Part1();
    a.add( parentitem: 1, childitem: 1);
    Assertions.assertEquals(a.add( parentitem: 1, childitem: 2), actual: true);
    Assertions.assertEquals(a.add( parentitem: 5, childitem: 4), actual: false);
}

Part1Test > postOrderSearch()
Part1Test
All 3 tests passed - 31ms
Test Results 31ms
Part1Test 31ms
  postOrderSearch() 31ms
  levelOrderSearch()
  add()
```

Integer Tipinde bir ağaç oluşturdum. Bu ağaca 1 elemanını ekledim. Daha sonra 1 elemanının çocuğu olarak 2 elemanını eklemeyi denedim. 1 Elemanı ağaç içerisinde olduğu için **true** return etti. Daha sonra 5 elemanının çocuğu olarak 4 elemanını eklemeyi denedim. 5 Elemanı ağaç içerisinde olmadığı için 4 elemanını ekleyemedi ve **false** return etti.

Böylece testlerden başarıyla geçmiş oldu.

```
@Test
public void levelOrderSearch() {
    Part1<Integer> a = new Part1();
    a.add( parentitem: 1, childitem: 1);
    a.add( parentitem: 1, childitem: 2);
    a.add( parentitem: 1, childitem: 3);
    a.add( parentitem: 2, childitem: 4);
    a.add( parentitem: 2, childitem: 5);
    a.add( parentitem: 4, childitem: 6);

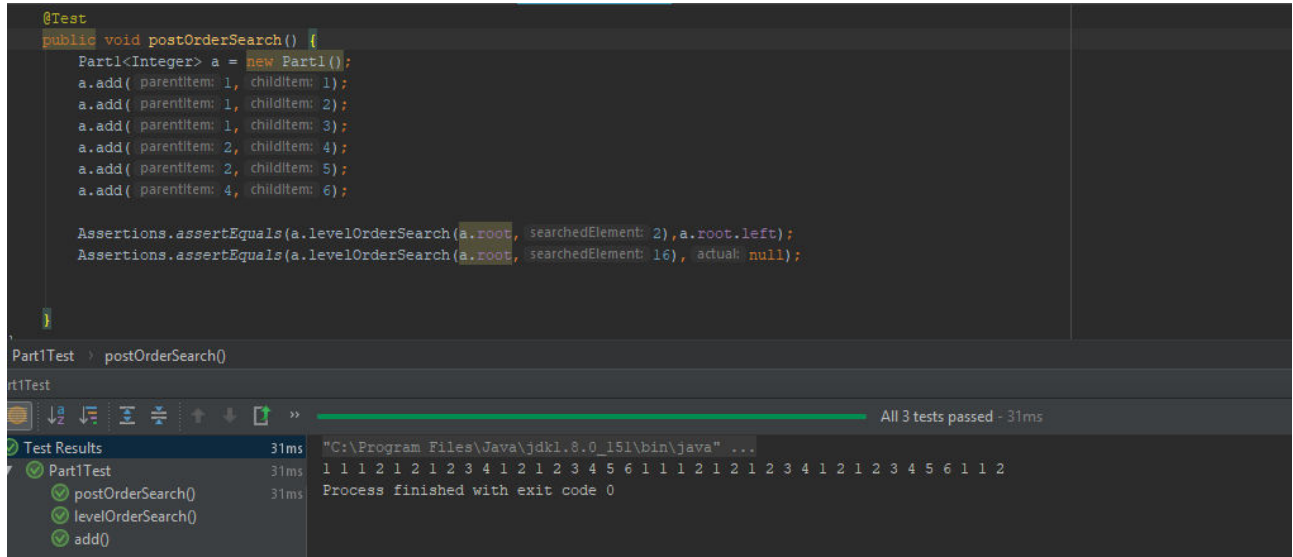
    Assertions.assertEquals(a.levelOrderSearch(a.root, searchedElement: 2), a.root.left);
    Assertions.assertEquals(a.levelOrderSearch(a.root, searchedElement: 16), actual: null);
}

Part1Test > postOrderSearch()
Part1Test
All 3 tests passed - 31ms
Test Results 31ms
Part1Test 31ms
  postOrderSearch() 31ms
  levelOrderSearch()
  add()
```

Integer Tipinde bir ağaç oluşturdum. Bu ağaca 6 farklı eleman ekledim. 2 Elemanını levelOrderSearch ile ağaç içerisinde aradım ve 2 elemanı ağaçta olduğu için 2 elemanını olduğu node'u return etti. Daha sonra 16 Elemanını levelOrderSearch ile ağaç içerisinde

aradım ve 16 elemanı ağaçta olmadığı için null return etti. Verilen ve beklenen değerler 2 testte uyuştu.

Böylece testlerden başarıyla geçmiş oldu.



```
@Test
public void postOrderSearch() {
    Part1<Integer> a = new Part1();
    a.add( parentItem: 1, childItem: 1);
    a.add( parentItem: 1, childItem: 2);
    a.add( parentItem: 1, childItem: 3);
    a.add( parentItem: 2, childItem: 4);
    a.add( parentItem: 2, childItem: 5);
    a.add( parentItem: 4, childItem: 6);

    Assertions.assertEquals(a.levelOrderSearch(a.root, searchedElement: 2), a.root.left);
    Assertions.assertEquals(a.levelOrderSearch(a.root, searchedElement: 16), actual: null);
}
```

Part1Test > postOrderSearch()

Test Results

Test Name	Duration	Result
Part1Test	31ms	Passed
postOrderSearch()	31ms	Passed
levelOrderSearch()	31ms	Passed
add()	31ms	Passed

All 3 tests passed - 31ms

Process finished with exit code 0

Integer Tipinde bir ağaç oluşturdum. Bu ağaca 6 farklı eleman ekledim. 2 Elemanını postOrderSearch ile ağaç içerisinde aradım ve 2 elemanı ağaçta olduğu için 2 elemanını olduğu node'u return etti. Daha sonra 16 Elemanını postOrderSearch ile ağaç içerisinde aradım ve 16 elemanı ağaçta olmadığı için null return etti. Verilen ve beklenen değerler 2 testte uyuştu.

Böylece testlerden başarıyla geçmiş oldu.

3.2 Running Results

```
Parent:13 Eklenen Child:20 - LevelOrder: 1 2 4 7 3 6 8 9 5 10 11 12 13

Tree'nin Son Durumu;

1
 2
  3
    5
      null
      null
    6
      null
      null
  4
    null
  7
    8
      10
      13
        20
          null
          null
          null
        11
          null
        12
          null
          null
      9
        null
        null
        null
null

POST ORDER TEST : 5 3 6 2 4 20 13 10 11 12 8 9 7 1
```

```
Parent:13 Eklenen Child:14 - LevelOrder: 8 3 4 9 5 3 4 9 2 10 1 2 10 1 5 7 5 7 13

Tree'nin Son Durumu;

8
 3
  2
    5
      13
      14
        null
        null
      9
        null
      7
        6
          5
            null
            null
          11
            null
            null
            null
        null
      4
        10
          null
          null
      9
        1
          null
          null
          null
null

POST ORDER TEST : 14 13 5 5 6 11 7 2 3 10 4 1 9 8
```

Main Test'in çıktısı çok uzun olduğu için 2 Farklı ağacın en son aldığı halin outputunu ekledim.Çıktılarda istenen tüm fonksiyonların çıktısı gözükmemektedir. Part 2'nin çıktısında sadece add metodu çalıştığı için sadece eklediğim elemanları gösterdim.Kod çalıştırıldığı zaman en alt

kısımda eklenen elemanları ekrana bastırdım.

- Main titles -> 16pt , 2 line break
- Subtitles -> 14pt, 1.5 line break
- Paragraph -> 12pt, 1.5 line break