Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

EMİRHAN KARAGÖZOĞLU
151044052

Course Assistant: Mehmet Burak Koca

1 INTRODUCTION

1.1 Problem Definition

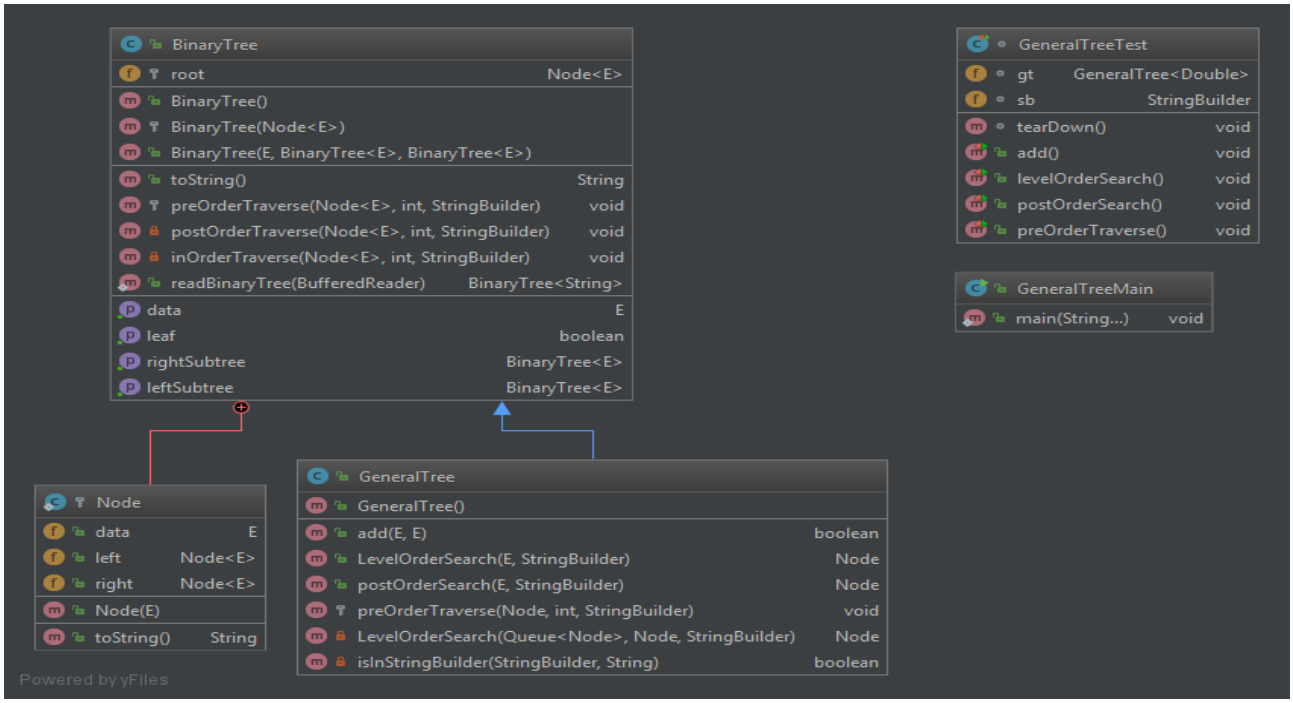
Birinci partta Binary Tree classından generic bir class extend etmemiz ve bu classın binary tree olarak general treeyi temsil etmesi istendi. Bu yapı üzerinde level order ve post order search yapan 2 method, pre order traverse yapan bir method ve bir de add methodu yazılması istendi.
İkinci partta multidimensional binary search tree generic classı yazılması istendi. Bu classın Binary Tree classını extend etmesi ve Search Tree Interface'ini impliment etmesi istendi.

1.2 System Requirements

Birinci partta istenilen bir tipte class objesi üzerinden add methodu çağırılması için o tipte bir parent ve bir child girilmesi gerekmektedir. Level order ve post order search methodlarının ararken nasıl gezindiklerini görmek için methodlara String Builder objesi gönderilmelidir.
İkinci partta MDST objesi oluşturmak için dimension gereklidir. MDST'nin tüm methodlarına parametre olarak göndermek için multidimension objesi gereklidir. Multidimension ojbesi oluşturmak için MDST objesi ile aynı tipte ve aynı dimensine sahip vector gereklidir.

2 METHOD

2.1 Class Diagrams



GeneralTree classı BinaryTree classından extend olmuştur ve BinaryTree classının tüm public protected method ve fieldlarına doğrudan erişim sağlayabilir. GenerelTree'nin add methodu parent ve child olarak generic tipte iki parametre alır. Verilen parenti current tree'de arar. Bulursa child'ı soluna ekler ve true döndürür, bulamazsa false döndürür. LevelOrderSearch methodu tree üzerinde general tree'ye göre level level gezerek aranan nodu bulmaya çalışır. Bulursa bulduğu nodu return eder bulamazsa null return eder. PostOrderSearch methodu general tree'ye göre post order sıra ile tree'yi gezerek aranan nodu bulmaya çalışır. Bulursa bulduğu nodu return eder bulamazsa null return eder. Binary Tree'nin preorder traverse methodu GeneralTree içerisinde override edildi. Bu method treeyi general tree'ye göre preorder şekilde gezip parametre olarak verilen string buildera doldurur.

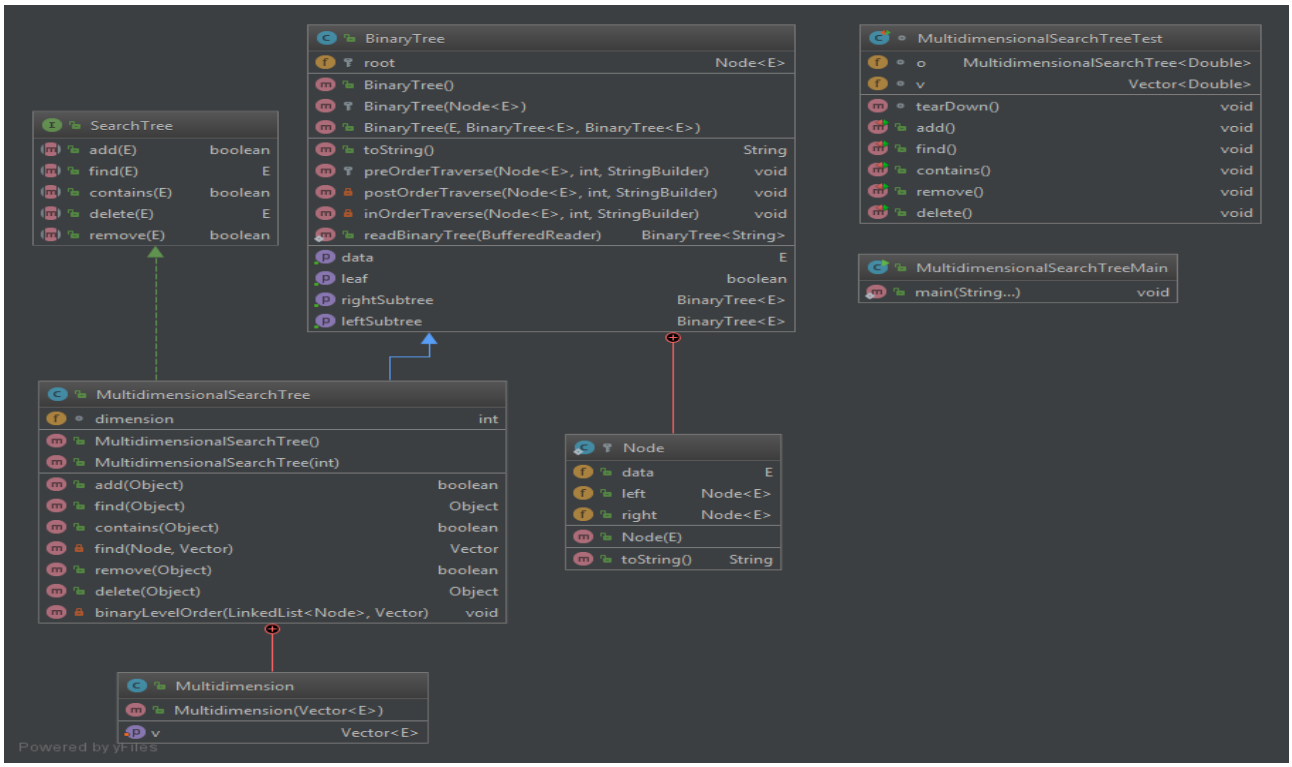
Time Complexity of Methods:

add(E) T(n) = O(n^3)

LevelOrderSearch(E,StringBuilder) T(n) = O(n^3)

PostOrderSearch(E,StringBuilder) T(n) = O(n^2)

PreOrderTraverse(Node,int,StringBuilder) T(n) = O(n)



MDST(Multidimension Search Tree) classı Binary Tree'den extend olmuş ve Search Tree interface'ini impliment etmiş bir classtır. Field olarak tree'nin dimension'ını tutar. Default demension 3 tür ve kullanıcının isteğine göre 0'dan büyük herhangi bir dimension'a sahip tree objesi oluşturulabilir. MDST SearchTree interface'inden aldığı 5 methodu impliment eder. Bu methodlar generic parametre alıp generic return(find,delete) etmektedir. Bu

generic type ise MDST'nin inner classı olan multidimension objesi olucak. Bu class sadece içerisinde vector tutmaktadır. Biz methodlara bu classın objesini yollayarak vector yollamış oluyoruz ve multidimension bilgi sağlıyoruz. Add methodu tree'nin her levelini dimension sayısına göre katmanlar (Örneğin dimension 2 ise katmanlar x,y,x,y.. şeklinde ilerler). Ekleyeceği elemanı tree'nin node'undan başlayıp leaf'e gelene kadar katmanın gerektirdiği index ile karşılaştırıp büyük ise sağa küçük ve eşit ise sola ilerler ve herhangi bir leaf'e ulaşınca eklemeyi gerçekleştirir. Find methodu aranan eleman treede varsa o elemanı return eder yoksa null return eder. Contains methodu aranan eleman treede varsa true yoksa false return eder. Remove methodu silinen node treede bulunuyor ise subtreelerini yeniden şekillendirip siler ve true döndürür, bulunmuyorsa false döndürür. Delete methodu silinen node treede bulunuyor ise subtreelerini yeniden şekillendirip siler ve sildiği node'u döndürür, bulunmuyorsa null döndürür.
Time Complexity of Methods:
add(E) T(n) = O(n)
find(E) T(n) = O(n)
contains(E) T(n) = O(n)
remove(E) T(n) = O(n)
delete(E) T(n) = O(n^2)

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

Birinci partta add işlemi yaparken kardeşler sağa çocuklar sola ekleniyor. Parent node tree'de bulunuyor ve child node null değilse ekleme işlemi gerçekleniyor. Level order search yaparken kardeşleri queue üzerinde tutup string builder'a dolduruyorum. String builder'a eklerken queue'dan çıkartıp çıkıralan node'un tüm çocukları queue'ya ekleniyor. Yani her recursive call'da bir level gezmiş (gezerkende aynı zamanda target aranıyor) ve bir alt level queue'ya eklenmiş oluyor ve bu queue recursive call'a parametre olarak iletiliyor.Aranan node bulunduğunda o node return ediliyor, bulunmazsa null return ediliyor.

PostOrderSearch'de root'tan başlayıp en sola kadar iniyor bu yolu stack'a atıyorum. Stackten pop yaparken pop edilen node'un sağı null olana kadar string buildera dolduyorum. Bu şekilde left-right-localroot şekilde post order gezip search etmiş oluyorum. Aranan node bulunduğunda o node return ediliyor, bulunmazsa null return ediliyor. PreOrderTraverse methodu Binary Tree classının pre order traverse methodu ile aynı mantıkta çalışıyor.
İkinci partta node'lar multidimension şekilde represent edilmektedir. Node'un dimension'ı ile tree'nin katmanları aynı şekilde oluşmakta (1,2,3 şeklinde ise nodelar dimension 3'tür ve ağaç rootdan leaf'e kadar x,y,z,x,y,z,.. şeklinde katmanlanır). Her node'a ekleme yaparken o node'un bulunduğu katmana göre karşılaştırma yapılır ve eklenecek node küçük ise sola büyük ise sağa eklenir. Find methodu recursive olarak treeyi gezer ve aranan elemanı bulursa onu return eder. Contains methodu da find methodunu kullanır. Delete yaparken tüm tree level order olarak bir linkedliste doldurulur (silincek node hariç). Daha sonra linked listten poll yaparak yeni bir tree oluşturulur.

3 RESULT

3.1 Test Cases

Part1'in JUnit testlerinde add,LevelOrderSearch, PostOrderSearch, PreOrderTraverse methodları test edildi.

- Add testinde methodun olası return değerleri(true/false) ve eklemelerin child yoksa sola varsa o child(s) 'ın sağına yapıldığı gösterildi.
- LevelOrderSearch testinde tree'ye 4 node eklendi ve search yapıldı. Search yaparken izlenen yol ile beklenen yol karşılaştırıldı ve eşitlik gösterildi.
- PostOrderSearch testinde LevelOrderSearch testinde izlenen yol izlendi.
- PreOrderTraverse testinde tree'ye 4 node eklendi ve preorder traverse methodunun string builder'ı ile beklenen string karşılaştırıldı ve eşitlik gösterildi.

Part2'in JUnit testlerinde add, find, contains, delete, remove methodları test edildi.

- Add testinde methodun olası return değerleri(true/false) test edildi. Ayrıca eklenecek node büyükse sağa küçükse sola eklenmesi gerektiği test edildi.
- Find testinde treede olan bir node arandı ve döndürmesi gereken node ile karşılaştırıldı. Bir de tree'de olmayan bir node arandı ve null döndürdüğü test edildi.
- Contains testinde treede olan bir node arandı ve true döndürdüğü gösterildi. Bir de tree'de olmayan bir node arandı ve false döndürdüğü gösterildi.

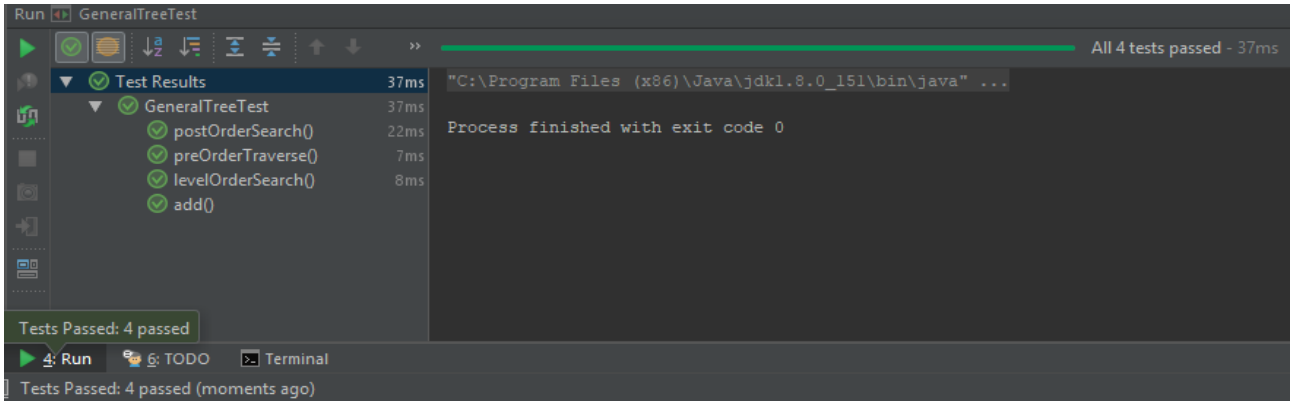
- Delete testinde treede olan bir node delete edildi ve döndürmesi gereken node ile karşılaştırıldı. Bir de tree'de olmayan bir node delete edildi ve null döndürdüğü test edildi.
- Remove testinde treede olan bir node remove edildi ve true döndürdüğü gösterildi. Bir de tree'de olmayan bir node remove edildi ve false döndürdüğü gösterildi.

Part1 main testinde şu şekil bir senaryo izlendi;
Tüm olasılıkları içeren geniş bir tree oluşturuldu ve bu tree üzerinde level order search ve pre order traverse methodları denendi. Çıktılar ekrana bastırıldı. Daha başka bir integer treesi oluşturulup bunun üzerinde post order search ve pre order traverse methodları denendi. Çıktılar ekrana bastırıldı.

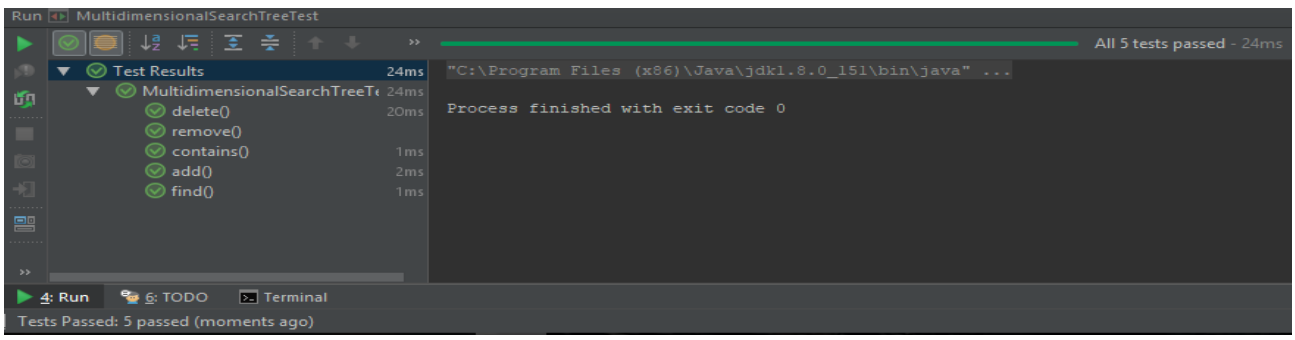
Part2 main testinde şu şekil bir senaryo izlendi;
Ödev pdf'indeki tree add methodu ile oluşturuldu. Sonra bu tree üzerinde find, contains, delete ve remove methodları denendi ve çıktılar ekrana bastırıldı. Daha sonra string 3 dimension bir tree oluşturuldu ve bunun üzerinde de tüm methodlar denenip çıktılar ekrana bastırıldıç

3.2 Running Results

- Part1 Junit Tests



- Part2 Junit Tests



- Part1 Main Test

```
William1
Robert William2 Adela Henry1
William Stephan William3 Matilda
Henry2
Henry Richard1 Geoffrey
John was found!

William1
Robert William2 Adela Henry1
William Stephan William3 Matilda
Henry2
Henry Richard1 Geoffrey John
Arthur Henry3 Richard
Edward1 Edmund
Edward2 Thomas Edmund1
Edward3
Emir wasn't found!

William1
Robert William2 Adela Henry1
William Stephan William3 Matilda
Henry2
Henry Richard1 Geoffrey John
Arthur Henry3 Richard
Edward1 Edmund
Edward2 Thomas Edmund1

Edward3 was found!

PreOrderTraverse

William1
   Robert
      William
         null
      null
   William2
      null
   Adela
      Stephan
         null
      null
   Henry1
      William3
         null
      Matilda
         Henry2
            Henry
               null
            Richard1
               null
            Geoffrey
               Arthur
                  null
               null
            John
               Henry3
                  Edward1
                     Edward2
                        Edward3
                           null
                        null
                     Thomas
                        null
                     Edmund1
                        null
                     null
                  Edmund
                     null
                  null
               Richard
                  null
               null
            null
         null
      null
   null
null

 --------------------------------
```

```
5 3 6 2 4
7 was found!

5 3 6 2 4 7 1
12 wasn't found!

PreOrderTraverse

1
  2
    3
      5
        null
      null
    6
      null
    null
  4
    null
  7
    null
  null
null


Process finished with exit code 0
```

- Part2 Main Test

```
-----------------------------------
[40, 45]
  [15, 70]
    null
    null
  [70, 10]
    null
    [69, 50]
      [66, 85]
        null
        null
      [85, 90]
        null
        null

[85, 90] is in MDS Tree
[69, 50] deleted from MDS Tree.

[40, 45]
  [15, 70]
    null
    null
  [70, 10]
    null
    [66, 85]
      null
      [85, 90]
        null
        null

[40, 45] deleted from MDS Tree.
```

```
[15, 70]
  null
  [70, 10]
    null
    [66, 85]
      null
      [85, 90]
        null
        null

--------------------------------
[lay, rat, house]
  [cow, jack, with]
    [fox, bear, and]
      null
      [is, deep, in]
        null
        null
    null
  [that, short, married]
    [malt, man, cat]
      null
      null
    null

[is, deep, in] is in MDS Tree
[malt, man, cat] deleted from MDS Tree.

[lay, rat, house]
  [cow, jack, with]
    [fox, bear, and]
      null
      [is, deep, in]
        null
        null
    null
  [that, short, married]
    null
    null

[cow, jack, with] deleted from MDS Tree.

[lay, rat, house]
  [fox, bear, and]
    null
    [is, deep, in]
      null
      null
  [that, short, married]
    null
    null

--------------------------------

Process finished with exit code 0
```