

Gebze Teknik Üniversitesi  
Bilgisayar Mühendisliği

CSE 222- 2018 Bahar

ÖDEV 4 RAPORU

GULZADA IISAEVA  
131044085

# 1 GİRİŞ

## 1.1 Problem Tanımı

Ödev 2 bölümden oluşmaktadır

Not: Sadece Bölüm 1 implement edildi, Bölüm 2 nin başlangıç kısmı yapıldı

### BÖLÜM 1:

Kitabın BinaryTree sınıfını extend ederek General tree oluşturmamız gerekiyordu.

Ayrıca

- add(E parent, E child) : verilen parent noduna child ekler
- levelOrderSearch(node,element) : Ağacın düz bir düzende ilerleyerek verilen değerin olup olmadığını kontrol eder
- postOrderSearch(node,element) : Kök düğümünden başlayarak alt ağaçlardan verilen değeri bulur
- preOrderTraverse() metodun override edilmesi

metotları yazılacak

### Bölüm 2:

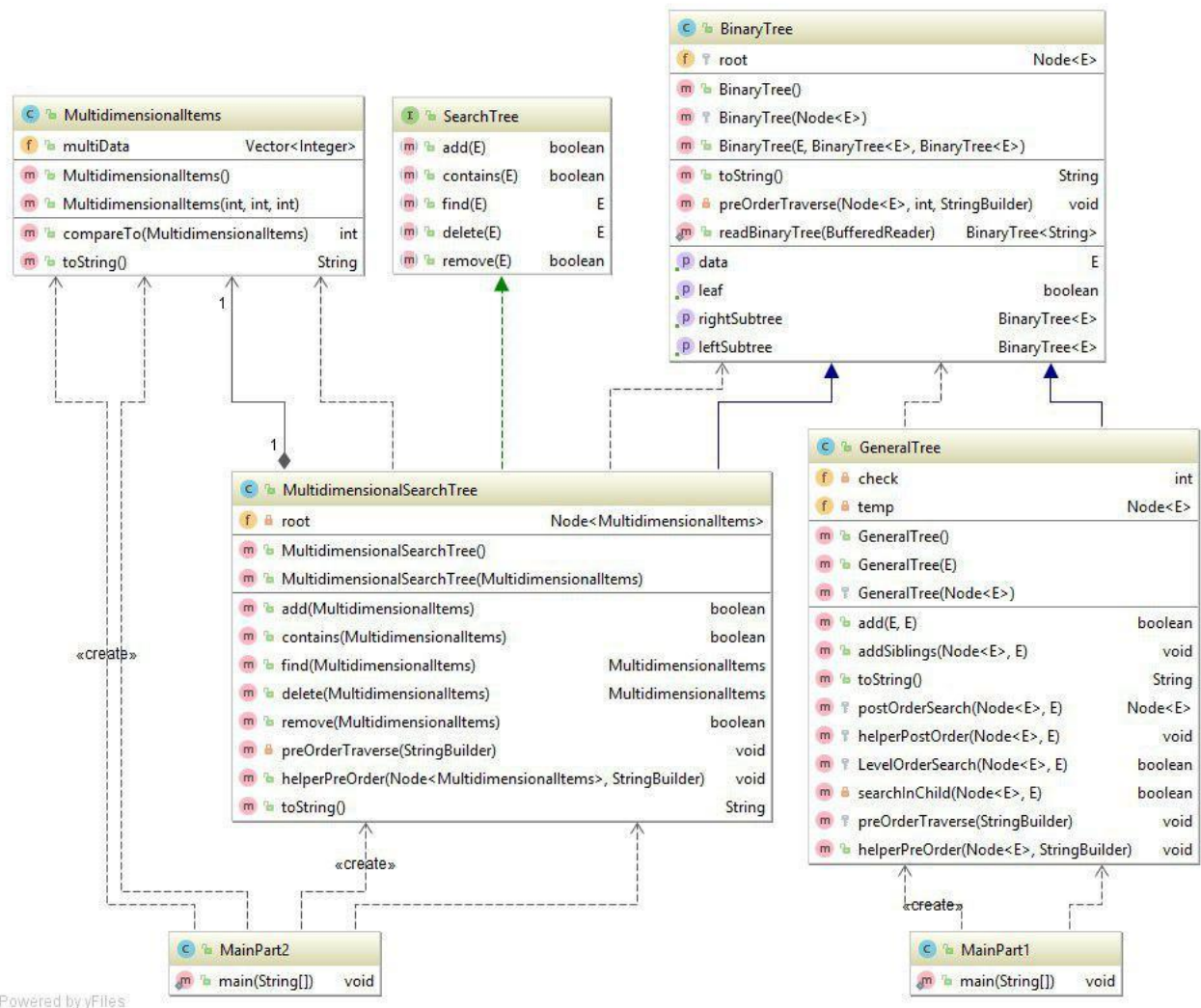
Bu bölümde benim yaptıklarım göre General ağacın düğümleri çok boyutlu olması gerekiyordu. Bir vector olmasını düşünüyordum ama vector comparable olmadığı için ayrı içinde vector objesini tutan ve Comparable sınıfını implement eden class yazılması düşünüldü.

## 1.2 Sistem Gereksinimleri

- Her 2 bölümün testi ayrı sınıflarda test edildi . “MainPart1”, “MaintPart2” sınıflarında gerçekleştirilmiştir. Buradaki değerlerle oynayarak farklı sonuçlar elde edilebilir.

## 2 METOD

### 2.1 Sınıf Diyagramları



## Diyagram açıklaması :

1. **BinaryTree sınıfı**: Kitaptan alındı.
2. **GeneralTree sınıfı** : General tree yapısı. BinaryTree sınıfından türeyor.
  - Metotlar:
    - add(E parent, E child) – verilen child ı parente ekler
    - addSiblings(E parent, E child) – child ı childların en sonuna ekler
    - postOrderSearch(node, element) : element değerine eşit olanı post order mantığına göre arar ve döndürür, yoksa null döndürür
    - helperPostOrder(node, element) : postOrderSearch(node, element) metotunun içinde kullanılan yardımcı metot
    - LevelOrderSearch(node, element) : element değerine eşit olanı level order mantığına göre arar ve true döndürür, yoksa false döndürür
    - searchInChild(node,element)- LevelOrderSearch(node, element) metotunun içinde kullanılan yardımcı metot
    - preOrderTraverse(StringBuilder sb) – ağacı pre order mantığına göre StringBuildere ekleyip döndürür.
    - helperPreOrder(node,sb)- preOrderTraverse(StringBuilder sb) metotunun içinde kullanılan yardımcı metot
    - toString() metodu ekrana basmak için
3. **MultiDimensional sınıfı**: Bölüm ikideki ağacın düğümleri çok boyutlu olması için yazılan sınıf. Comparable sınıfını implement eder.
  - Metotlar:
    - compareTo(MultiDimensional o)- verilen değer eşit mi değil mi kontrol eder
    - toString() metodu ekrana basmak için
4. **MultiDimensionalSearchTree sınıfı**: Çok boyutlu düğümleri olan ağaç yapısı. SearchTree interfaceini implement ediyor
  - İmplement edilen metotlar:
    - preOrderTraverse(StringBuilder sb) – ağacı pre order mantığına göre StringBuildere ekleyip döndürür.
    - helperPreOrder(node,sb)- preOrderTraverse(StringBuilder sb) metotunun içinde kullanılan yardımcı metot
    - toString() metodu ekrana basmak için

## 2.2 Problem Çözüm Yaklaşımı

### BÖLÜM 1:

Bu bölümde BinaryTree'yi extend eden General Tree sınıfı yazıldı.

Bizden istenilen **add(parent,child)** metotunda:İlk önce parent ve child ağaçta var mı diye

LevelOrderSearch() metodu ile kontrol edilir. Yoksa false döndürür. Sonra postOrderSearch() metodu ile verilen parentin düğümünü bulur. Eğer daha önce child yoksa soluna ekler, varsa addSibling() metodunu çağırarak childların en sonuna ekler.

addSibling(parent,child) - parentin solundaki ilk çocuğun sağındakileri null olana kadar gidip oraya ekler.

helperPostOrder(node, element) verilen element rootdaki datayı eşit mi bakıyor , değilse solunu ve sağını recursive olarak çağırıyor. Bulduğu anda private olarak tuttuğumuz temp düğüme atanır, private olarak tuttuğumuz check değişkenine -1 atanır. Bulamadıysa null döndürür.

postOrderSearch(node,element) helperPostOrder(node, element) metodunu çağırır. Eğer check değişkeni 0 ise null döndürür, değilse temp'i döndürür

LevelOrderSearch(node,element) metodu kökten başlayarak aramaya başlar, kökün değeri elemente eşit değilse , searchInChild(node,element) metodunu çağırarak child ağacından arar.

searchInChild(node,element) metodu child ağacı null olana kadar her düğümün datası elemente eşit mi kontrol eder. Eğer elementi bulmuşsa true, değilse false döndürür.

preOrderTraverse(Stringbuilder sb) ağacı ekrana basmak için yazılan metot. BinaryTree de bu metot private olarak tanımlandığı için General Tree de ayrı yazıldı. Ağaçtaki elemanların seviyesine göre yazıldı.

## BÖLÜM 2:

Bölüm ikide ağacın çok boyutlu düğümleri için MultidimensionalItems diye sınıf yazdım. İçinde vektor tipinde değişken tuttum. Integer x, y, z i alan constructoru var. MultidimensionalSearchTree de ise sadece constructor ve preOrderTraverse metodu yazıldı. Constructorda sadece rootu Multidimensional tipinde değer atadım. Yapılanlar bu kadar.

## 3 SONUÇ

### 3.1 Test Case

#### I. JUnit

GeneralTree sınıfın belli bir metodları JUnit'de test edildi.

Test edilen metodlar:

- 1) add() metodu ağaca eleman ekleyerek test ettim. Eğer verilen parent ağaçta var ise ve child eklenirse true döndürmesi lazım.

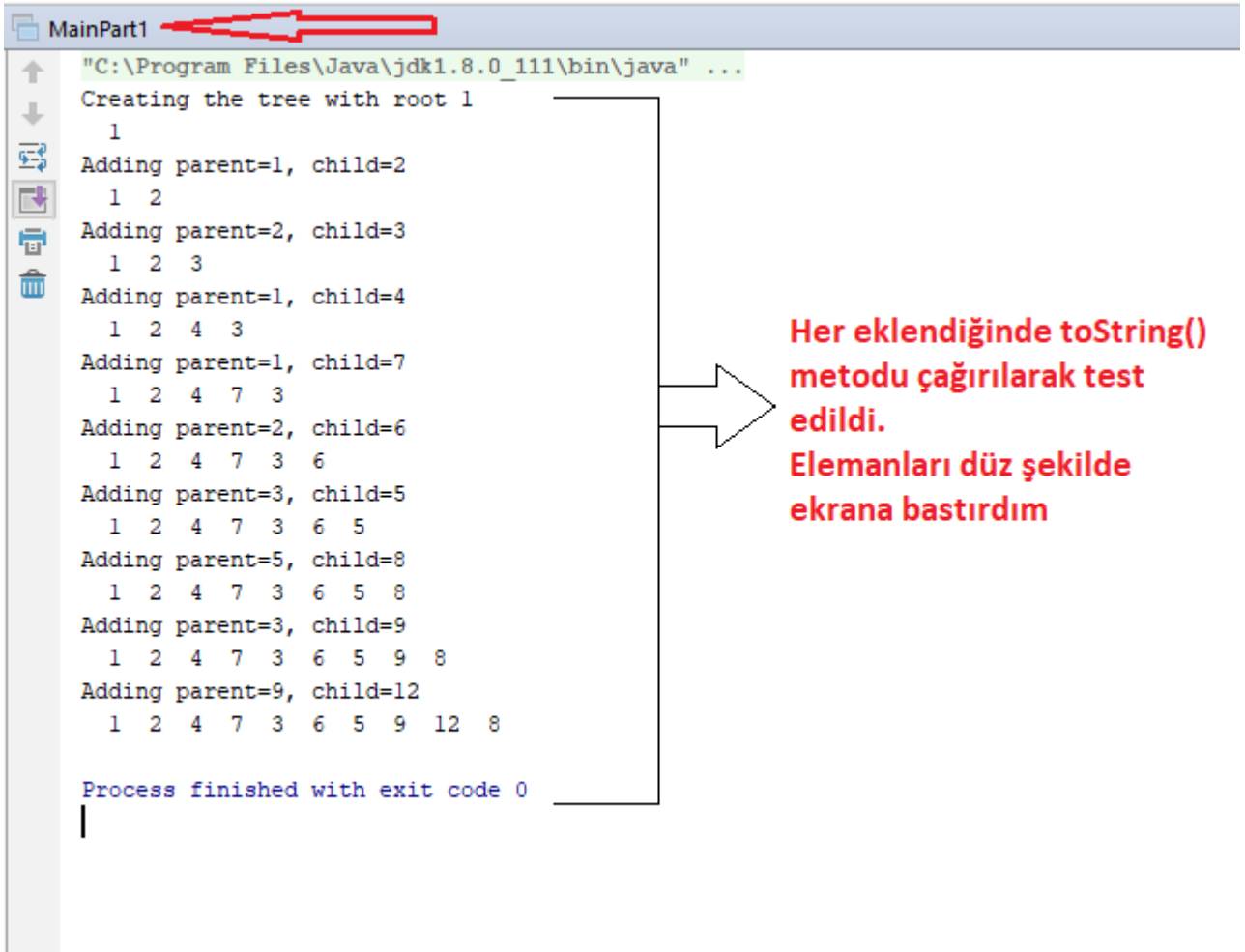
- a. helperPostOrder() metodu add metotunda çağırıldığı için otomatik test edilmiş oldu
- 2) postOrderSearch () metodu: verilen değerdeki element ağaçta var mı test edildi. Var olmayan elemanı arattırarak null mu kontrol edildi
- 3) levelOrderSearch () metodu: verilen değerdeki element ağaçta var mı test edildi

## II. Main Test

Her bölümün çalışması için "Mainpart1.java" , "Mainpart2.java" sınıfı yazıldı. Amaç her bölüm için yazılan sınıfların her metodunu test etmek . Bu sınıflardaki inputları değiştirerek farklı sonuçlar elde edilebilir. Bizden istenilen tüm metodlar test edildi. Çalıştırma sonuçları kısmındaki ekran görüntüsü resimlerden detaylı görebiliriz.

## 3.2 Çalıştırma sonuçları

Bu test Mainpart1.java" , "Mainpart2.java" sınıflarının test edilmesidir. Detaylar resimlerde açıklanmıştır



```
MainPart1
"C:\Program Files\Java\jdk1.8.0_111\bin\java" ...
Creating the tree with root 1
1
Adding parent=1, child=2
1 2
Adding parent=2, child=3
1 2 3
Adding parent=1, child=4
1 2 4 3
Adding parent=1, child=7
1 2 4 7 3
Adding parent=2, child=6
1 2 4 7 3 6
Adding parent=3, child=5
1 2 4 7 3 6 5
Adding parent=5, child=8
1 2 4 7 3 6 5 8
Adding parent=3, child=9
1 2 4 7 3 6 5 9 8
Adding parent=9, child=12
1 2 4 7 3 6 5 9 12 8
Process finished with exit code 0
```

Her eklendiğinde toString() metodu çağırılarak test edildi. Elemanları düz şekilde ekrana bastırdım

