**Gebze Technical University**

**Computer Engineering**

**CSE 222 - 2018 Spring**

**HOMEWORK 04 REPORT**

**DENİZ BABAT**

**131044013**

İçindekiler

[1 PART1 1](#_Toc511678695)

[1.1 GİRİŞ 1](#_Toc511678696)

[1.1.1 PROBLEM TANIMI: 1](#_Toc511678697)

[1.1.2 CLASS DİAGRAMI: 1](#_Toc511678698)

[1.1.3 PROBLEME YAKLAŞIM 2](#_Toc511678699)

[1.1.4 SONUÇ 2](#_Toc511678700)

[2 PART 2 2](#_Toc511678701)

[2.1 GİRİŞ 2](#_Toc511678702)

[2.1.1 PROBLEMİN TANIMI 2](#_Toc511678703)

[2.1.2 CLASS DİAGRAMI 3](#_Toc511678704)

[2.1.3 PROBLEME YAKLAŞIM 3](#_Toc511678705)

[2.1.4 SONUÇ 3](#_Toc511678706)

Şekil Tablosu

[Şekil 1: Part1 Class Diagramı 1](file:///C:\Users\deniz\IdeaProjects\HW04_131044013\Rapor\131044013_rapor.docx#_Toc511678768)

[Şekil 2: Part-1 sonuç 2](#_Toc511678769)

[Şekil 3: Part2 Class diagramı 3](file:///C:\Users\deniz\IdeaProjects\HW04_131044013\Rapor\131044013_rapor.docx#_Toc511678770)

[Şekil 4: Part-2 sonuç 3](#_Toc511678771)

# PART1

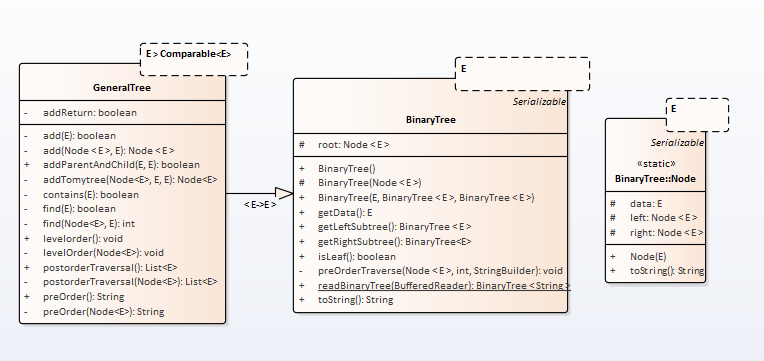
## GİRİŞ

### PROBLEM TANIMI:

Problem Binary Tree’ye iki ilişkili iki eleman eklemektir. Bu ilişki parent ve child ilişkisidir. Eklenecek parent ve child koşulu şöyledir; Eğer tree de parent varsa child eklenip return değeri true olarak dönecektir. Eğer parent yoksa return değeri false olarak dönecektir. İlişki bundan ibarettir. Bu problemin çözülmesi için GenerelTree adında bir class oluşturdum ve metod formatları ve açıklaması aşağıdaki gibidir.

* **public boolean addParentAndChild(E parent, E child):** bu metodun bir parent bir de child parametresi alarak child’ı ağaca ekler ya da eklemez. Parent’ı olmayan child’lar için dalse return eder ve ekrana basar. Metod parent’ı bulmak için tüm ağacı dolaşır ve parent’ı bulduğunda ise; alt ağacı recursive olan başka bir fonksiyona parametre olarak verip çocuğu ağaca ekler. Bu yüzden T(n) = O(nlog(n)) dir.
* **public List<E> postorderTraversal():** Bu metod mevcut ağaçtaki dataları postOrder sıralamaya göre sıralar. Tüm ağacı dolaştığı için T(n) = O(n)dir.
* **public void levelorder():** Bu metod general tree ye göre ağaçtaki dataları level order’a göre ekrana basar yani search yapar. Tüm ağacı dolaştığı için T(n) = O(n)dir.
* **public String preOrder():** Bu metod ağaçtaki dataları preorder’a göre ekrana bir string generate eder. Tüm ağacı dolaştığı için T(n) = O(n)dir.

### CLASS DİAGRAMI:



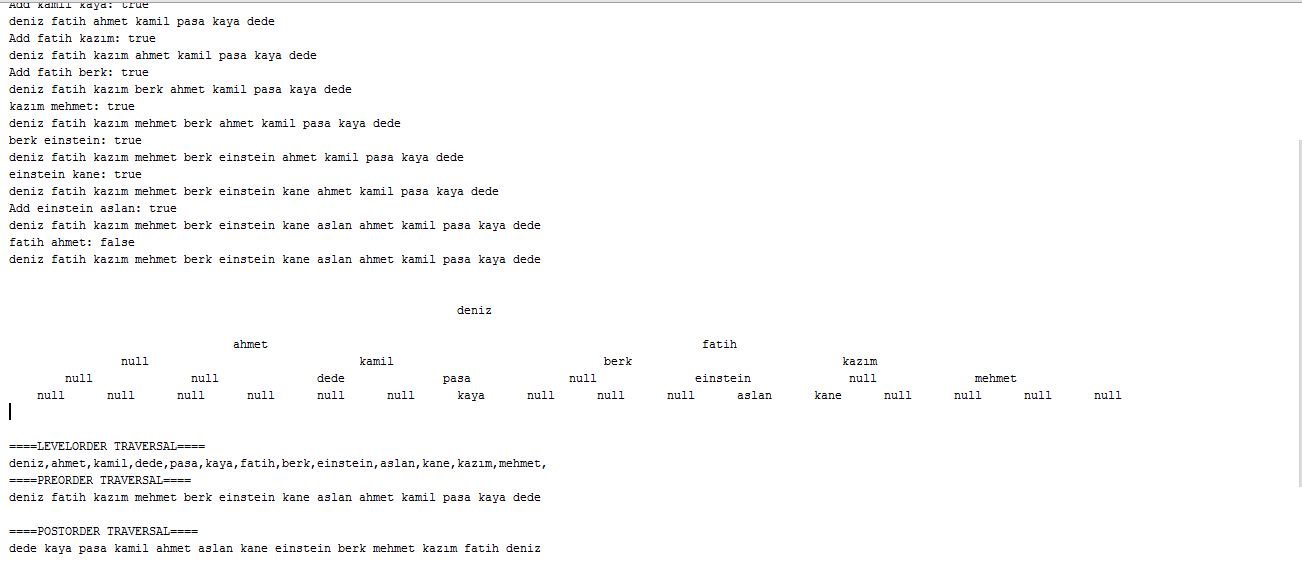
Şekil 1: Part1 Class Diagramı

Part 1 class diagramı olarak şekil 1 de gözükmektedir. Yukarıda açıklaması yapılmıştır.

### PROBLEME YAKLAŞIM

Problemi çözmek için ilk olarak null olan ağaıcın rootuna ilk gelen parentı ve child ı ekledim. Daha Sonra eklenecek child’ın parent’ı ağaçta var mı yok mu diye aratıyorum. Yoksa false return ediyorum. Eğer varsa; çocuğun parent’ını bulup altındaki ağaç’ın (parent’ında leaf’i olabilir, Leaf yoksa ağacın leafine ekler.) çocuğu ekliyorum.

### SONUÇ



Şekil 2: Part-1 sonuç

# PART 2

## GİRİŞ

### PROBLEMİN TANIMI

Hiper düzlem üzerindeki multidimensional noktalar düzlem eksenlerine göre ağaca eklenir. Root her zaman X eksini ile başlar ve ağaç y, z, k ... x, y … vs. olarak devam eder. Bu problem çözümü için MDS class’ı oluşturulmuştur. Metodları aşağıdaki gibidir.

* **public void add(ArrayList<Integer> points, int dimens):**  Bu metod iki parametre alır ilk parametre değerlerden oluşan koordinat düzlemi, ikinci parametre kaç boyutlu bir düzlem olduğu ile ilgilidir. Bu metod eksenleri karşılaştırdığı için her geçtiği düzlemi compare eder. Bu yüzden normal ağaca add methodu gibi çalışır. Bu yüzden T(N) = O(log(n)) dir.

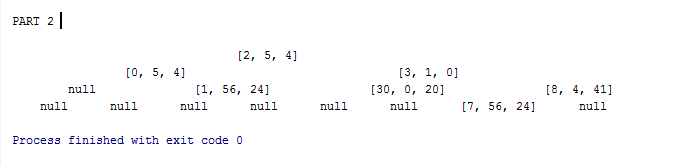
### CLASS DİAGRAMI

Şekil 3: Part2 Class diagramı

### PROBLEME YAKLAŞIM

Ağaca eklencek düzlemi ilk olarak kaç boyutta olaması için fonksiyona boyut girilmesi için parametre koydum. Bu paramtre yi her recursive kolda bir azaltıyorum. Böylece her düzlemi düzgün bir şekilde karşılaştırma yapabiliyorum. Eğer bu parametre 0 olursa tekrar parametre ilk halini recursive kola veriyorum, böylece düzlem noktasının ağacın leaf’ine eklenmesi için düzlemin eksenlerini tekrarlamasını sağlıyorum ve her ekseni kendi içinde karşılaştırıyorum.

### SONUÇ



Şekil 4: Part-2 sonuç