Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 4 REPORT

Efkan DURAKLI
161044086

Course Assistant: Mehmet Burak KOCA

1. PART 1

1.1 INTRODUCTION

Bu bölümde genel ağaç yapısı ikili ağaç kullanılarak gerçekleştirildi. Genel ağaçta herhangi bir denge yoktur. İkili ağaçta bir nodun en fazla iki adet çocuğu olabilir. Genel ağaç ikili ağaç olarak ifade edilirken bir nodun ilk çocuğu o nodun soluna yazılırken diğerleri kardeşlerinin sağınna yazılır.

1.1.1 Problem Definition

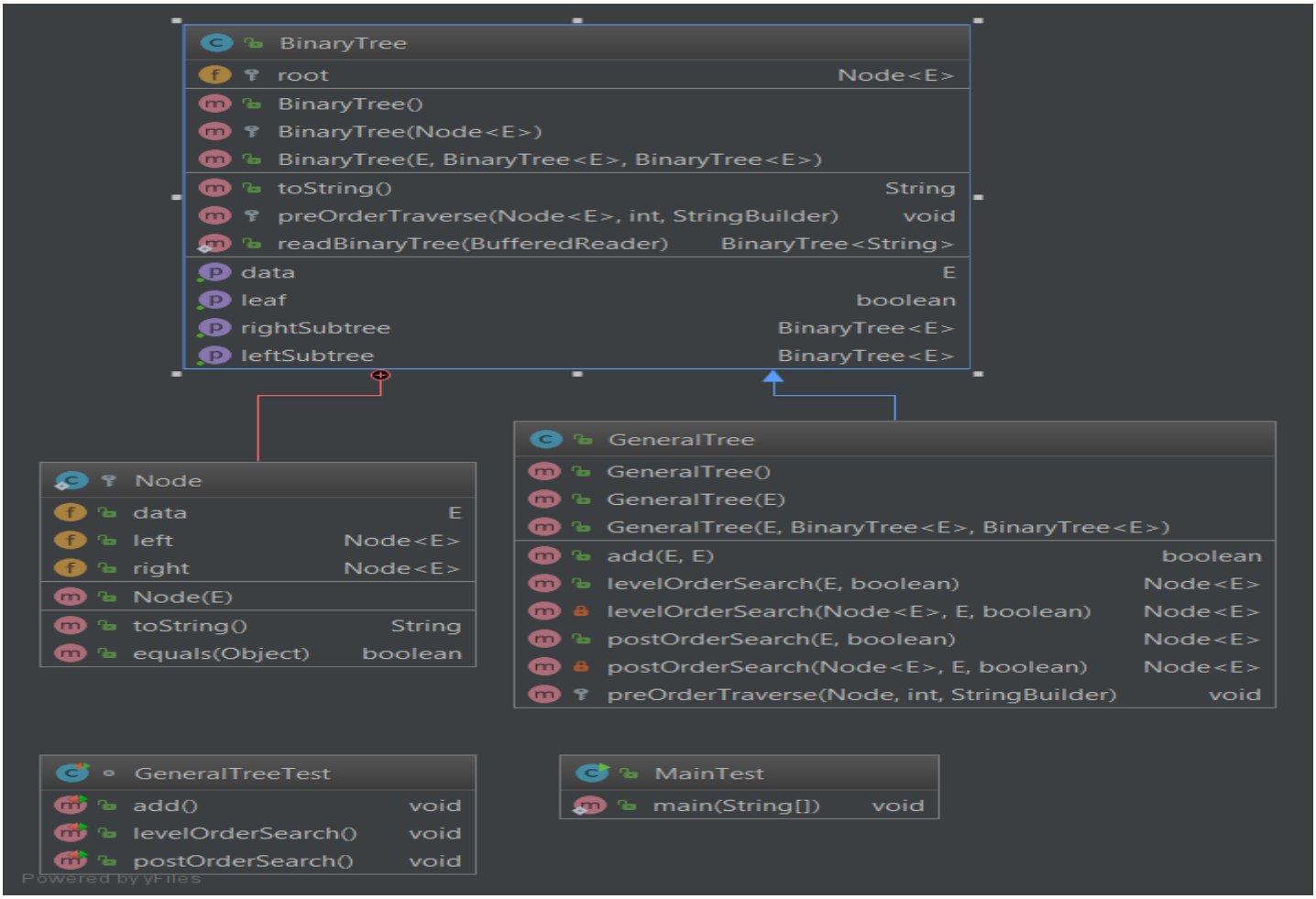
Bu yapının gerçekleştirilebilmesindeki problemler şunlardır.

- Ağaca eleman eklenirken nasıl ekleneceği.
- Ağaçta arama yapılırken ağacın nasıl gezileceği.

Bu problemleri çözme yaklaşımından Problem Solution Approach kısmında bahsedeceğim.

1.2 METHOD

1.2.1 Class Diagrams



1.2.2 Problem Solution Approach

Ağaca eleman eklemek için add metodu yazıldı. Bu metot eklenecek elemanın ebeveynini ve kendisini alır. Eleman eklemeden önce ebeveynin ağaçta olıp olmadığı kontrol edilir. Eğer ağaçta varsa sol çocuğu kontrol edilir. Eğer solunda çocuk yoksa sola eklenir, eğer çocuk varsa en son çocuğu bulana kadar gidilir ve en son çocuk olarak eklenir.

Ağaçta arama postOrderSearch ve levelOrderSearch olmak üzere iki türlü arama yapıldı. postOrderSeach'da ebeveyne bakmadan önce çocuklara bakılır ve sonra ebeveyne bakılır.levelOrderSearch'da arama 1. nesilden başlayarak son nesile kadar gidilerek yapılır. postOrderSearch recursive olarak yapıldı, levelOrderSearch için queue yapısı kullanıldı.

1.2.3 Algorithm Analysis

Bu bölümde yazılan üç ana metodun algoritma analizi yapılacak.

- **postOrderSearch():**

Bu metot arama yaparken gelen parametreni ağaçta olmaması durumunda bütün ağacı gezer. Dolayısıyla en kötü durumda bu ağacın çalışma süresi n'dir.

Dolayısıyla T(n) = O(n) diyebiliriz.

- **levelOrderSearch():**

Bu metot arama yaparken gelen parametreni ağaçta olmaması durumunda bütün ağacı gezer. Dolayısıyla en kötü durumda bu ağacın çalışma süresi n'dir.

Dolayısıyla T(n) = O(n) diyebiliriz.

- **add() metodu:**

Bu metot verilen ebeveyni ağaçta aramak için postOrderSearch metodunu çağırır. Bu metodun çalışma zamanı O(n)'dir. Bulunan elemanın son çocuğuna kadar gidebilmek için bir döngü yapılır. Bu döngünün çalışma süresi maksimum n olabileceği için Bu işlemin de çalışma zamanı O(n)'dir diyebiliriz.

Dolayısıyla T(n) = O(n) + O(n) = O(n) diyebiliriz.

1.3 RESULT

1.3.1 Test Cases

Main Test

Bu ağacı test edebilmek için İngiliz Kraliyet ailesinin üyeleri ağaca eklenedi. Bu ağaç üzerinde postOrder ve levelOrder gezildi ve sonuçlar ekrana yazıldı.

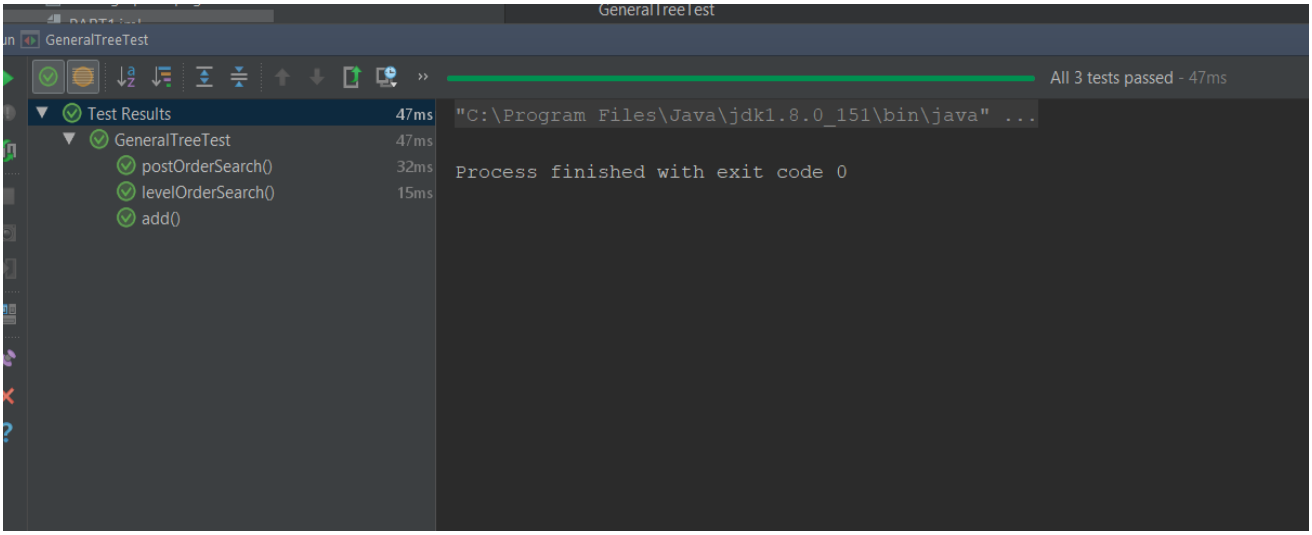
Bu ağacı test etmek için integelardan oluşan elemanlar ağaca eklendi ve postOrder ve levelOrder arama sonuçları ekrana yazdırıldı.

Unit Tests

Add, postOrderSearch ve levelOrderSearch metotlarının birim testleri gerçekleştirildi ve bütün testler başarılı bir şekilde sonuçlandı.

1.3.2 Running Results

Unit Test Results



Main Test Results

Main Test sonuçlarını proje dosyasındaki MainTest sınıfının main metodunu çalıştırarak görebilirsiniz.

2. PART 2

2.1 INTRODUCTION

Bu bölümde çok boyutlu ikili arama ağacı gerçekleştirildi. Çok boyutlu ikili ağacın normal ikili ağaçtan farkı çok boyutlu ağaca eleman eklenirken karşılaştırmalar her level için o levela karşılık gelen boyutla yapılır. Bu yapı gerçeklenirken kitaptaki BinaryTree sınıfı extend edildi ve kitaptaki SearchTree interface'i implement edildi.

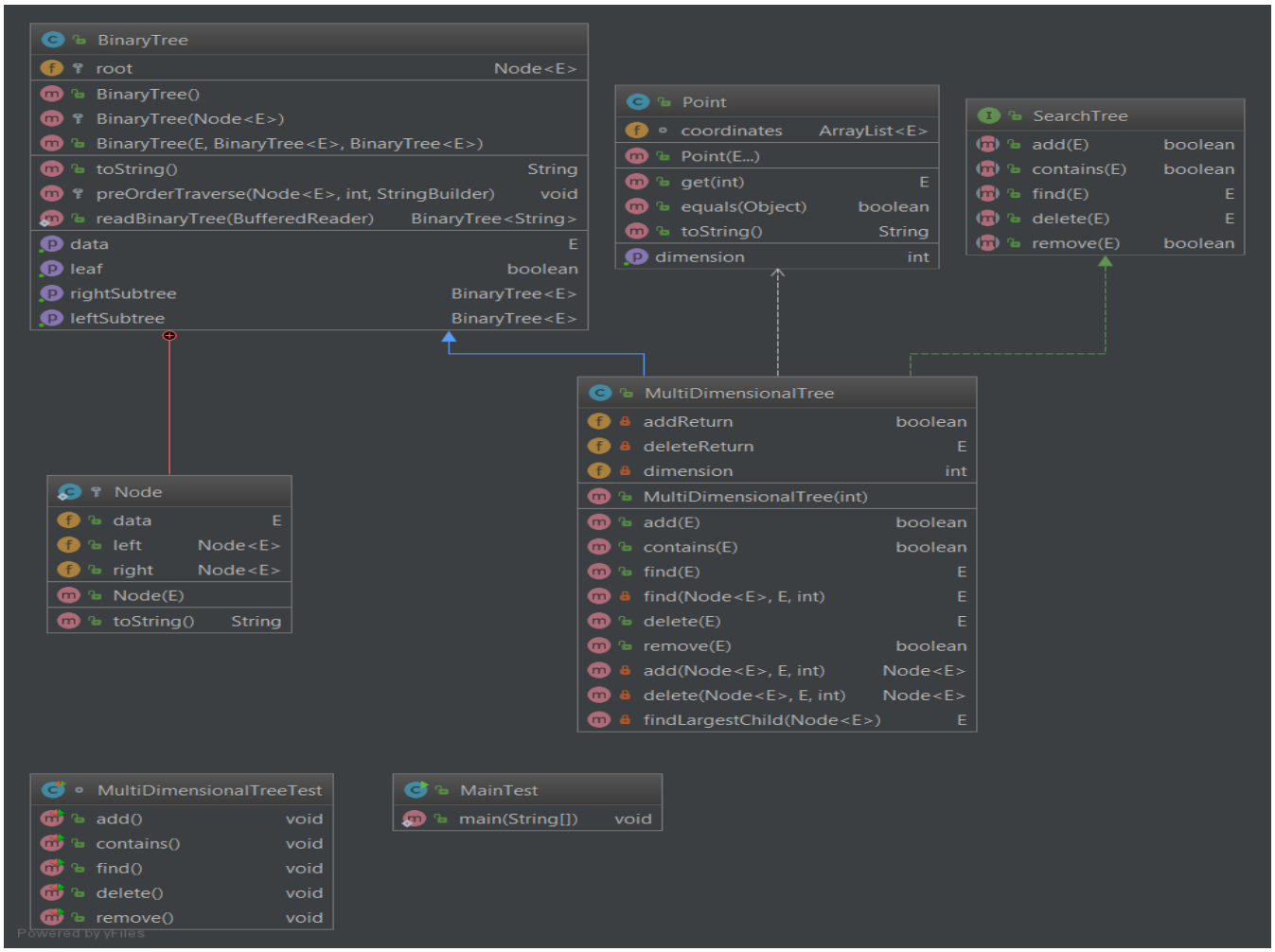
2.1.1 Problem Definition

Bu bölüm yapılırken karşılaşılan problemleri tanımlayacak olursak;

- Ağaca eklenecek elemanların boyutları 1 den fazla olabileceği için bu elemanların nasıl ifade edileceği.
- Eklenecek elemanların her bir boyutundaki elemanın karşılaştırılabilir olması gerekiyor.
- SearchTree interface'inin metotlarının implementasyonu yapılırken karşılaşılan problemler.

2.2 METHOD

2.2.1 Class Diagrams



2.2.2 Problem Solution Approach

Problem Definition kısmında bahsettiğim problemleri çözme yaklaşımından bahsedeceğim.

- Ağaca eklenebilecek elemanlar için bir Point adında bir generic sınıf yazıldı ve bu sınıfın dataları karşılaştırılabilir olması için dataların Comparable interfaci'ni implement eden bir sınıf olması zorunluluğu getirildi.
- SearchTree interface'indeki metotları implement ederken bu metotlar için wrapper fonksiyonlar yazıldı ve bu metotlara hangi levelda olduğunu söyleyen bir integer parametre eklendi.

2.2.3 Algorithm Analysis

● add()

Bu metot ekleme yaparken her seferinde ağaç ikiye bölünerek küçüldüğü için bu fonksiyonun çalışma zamanı Q(logn)'dir.

● find()

Bu metot arama yaparken her seferinde ağaç ikiye bölünerek küçüldüğü için bu fonksiyonun çalışma zamanı Q(logn)'dir.

● contains()

Bu metot arama yaparken find() metodunu çağırdığı için bu metodun çalışma zamanı find() metoduyla aynıdır.

● delete()

Bu metot silme yaparken her seferinde ağaç ikiye bölünerek küçüldüğü için bu fonksiyonun çalışma zamanı Q(logn)'dir.

● remove()

Bu metot silmeyaparken delete() metodunu çağırdığı için bu metodun çalışma zamanı delete() metoduyla aynıdır.

2.3 RESULT

2.3.1 Test Cases

Main Test

Bütün metotların test edildiği bir main test yazılmıştır.Bu testin sonuçlarını görebilmek için proje içerisindeki MainTest sınıfını çalıştırabilirsiniz.

UnitTest

Bütün metotlar için ünit test yapılmıştır. Testleirn sonucu aşağıdaki gibidir.