

Gebze Technical University

Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 04 REPORT

CELAL TEMİZ

101044070

Course Assistant : Mehmet Burak KOCA

1 INTRODUCTION

1.1 Problem Definition

Part 1 : Problem BinaryTree sınıfının extend edilerek bir general tree sınıfının oluşturulmasıdır. Bu tree yapısı oluşturulurken, node' un sol tarafına bir parent' a child' lar, diğer tarafına ise sibling' ler (parent' a ait kardeşler) eklenmelidir.

Ağaca eleman ekleme işleminin yapılması için bir methodun yazılması istenmiştir. Eğer ekleme işlemi başarılı ise true, eğer parent bu ağaç içerisinde yoksa false return edecektir.

Bu methoda ek olarak, levelOrderSearch() ve postOrderSearch() methodlarının implement edilmesi istenmektedir.

Level-Order Search : Bu method ağacı level-order olarak gezecektir. İlk root node 1.level, onun çocukları 2.level, çocukların çocukları da 3.level olarak sırayla devam edecektir.

Post-Order Search : Bu method ağacı post-order olarak gezecektir. Alt ağaçlardan köke doğru post-order şekilde gezmesi istenmektedir. Eğer aranan item ağaç içerisinde ise Node referansı, yoksa null değeri return edecektir.

Bunlara ek olarak preOrderTraverse() methodunun da override edilmesi istenmektedir.

Part 2 : Bu kısımda çok boyutlu item' lar için bir general search tree yapısının oluşturulması istenmektedir. Boyut sayısına göre karşılaştırma yapılan item değişecektir. Örneğin, 2 boyutlu bir yapıda 1. seviyede karşılaştırma x konumuna göre, 2. Seviyede karşılaştırma y konumuna göre, 3. Seviyede karşılaştırma tekrar x konumuna göre, 4. seviyede tekrar y konumuna göre olacak şekilde sırasıyla gerçekleşecektir.

Bu yapı oluşturulurken BinaryTree sınıfı extend edilecek, SearchTree interface' i implement edilecektir.

1.2 System Requirements

Geliştirme Ortamı :

intelliJ version = 2017.3.4, jdk= 8 – Virtual Machine – Debian OS

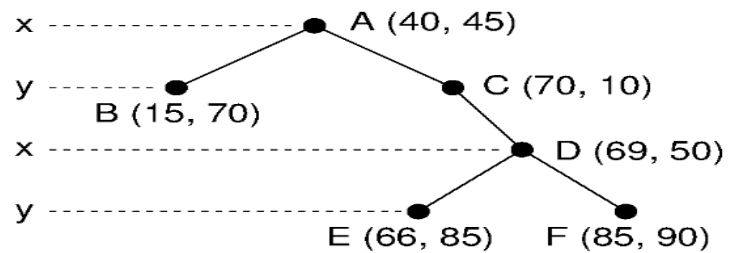
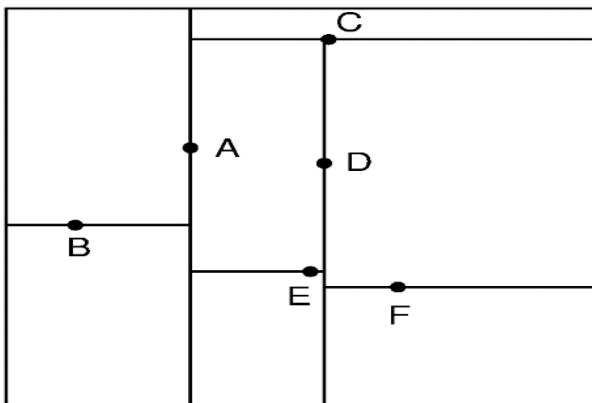
Veriler :

Part 1 için örnek veriler .txt dosyalarına kaydedilerek gerçekleştirilen işlemler bu dosyalardan alınan verilerle sağlanmaktadır. Bu değerler test için integer, character ve String olarak dosyalar içerisine eklenmiştir.

```
1 1
2 2,1
3 4,1
4 7,1
5 3,2
6 6,2
7 5,3
8
1 a
2 b,a
3 d,a
4 g,a
5 h,a
6 i,a
7 j,a
8 c,b
9 f,b
10 k,b
11 l,b
12 e,c
13
```

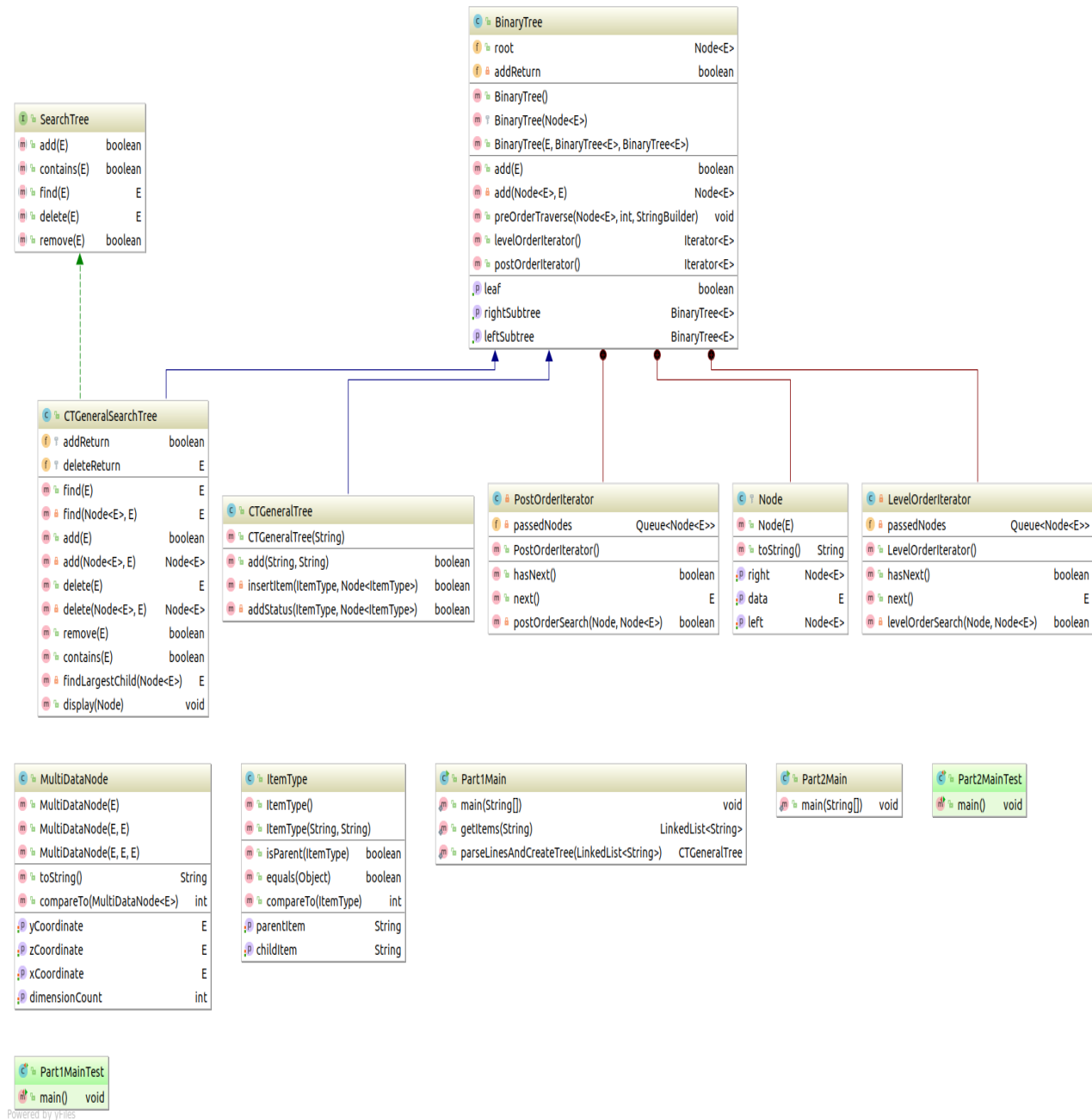
```
1 Celal
2 Ahmet,Celal
3 Ayşe,Celal
4 Selma,Celal
5 Hakan,Celal
6 Derya,Selma
7 Caner,Selma
8 Pelin,Derya
9 Onur,Ayşe
10 Atakan,Ahmet
11 Can,Ahmet
12 Esra,Atakan
```

Part2 için 2 boyutlu örnek veriler aşağıdaki gibidir.



2 METHOD

2.1 Class Diagrams



2.2 Problem Solution Approach

Part 1' de problemin çözümü için öncelikle generic bir yapı oluşturmak için ItemType adında bir sınıf oluşturdum. Bu sınıf içerisine ağaca eklenecek olan parent ve child' a ait verileri bu sınıf içerisine ekledim.

Daha sonra bir BinaryTree sınıfı implementasyonu, içerisine 2 farklı private inner iterator sınıfları ekleyerek levelOrderSearch() ve preOrderSearch() davranışlarını istendiği gibi implement etmeye çalıştım. Bu sınıfları implement etmemin nedeni, iterator' un oluşturulan tree üzerinde gezerken takip ettiği sırayı değiştirebilmektir.

Bu işlemten sonra bir general tree yapısı oluşturmak için CTGeneralTree sınıfı yazdım. Bu sınıf içerisine ilk olarak bir tane constructor koydum ve parametre olarak verdiğim ilk parent olmasını sağladım. Daha sonra ödevde istendiği gibi bir childItem ve bir parentItem alan add() methodu yazdım. Bu parentItem ve childItem değerlerini aldıktan sonra ağaca eklemek için bir ItemType ikilisi oluşturdum. Bu ikilinin daha önceden ağaç içerisine eklenip eklenmediğini anlamak için bir addStatus() methodu yazdım. Methodun döndürdüğü değere göre ekleme işlemini gerçekleştirdim. Bu ekleme işlemini yaparken ödevde istendiği gibi çocuklar sol tarafa, kardeşlerde sağ tarafa olacak şekilde yaptım.

Part 2' de problemin çözümü için öncelikle MultiDataNode adında generic bir sınıf oluşturdum. Bu sınıfın içerisinde default olarak 3 tane koordinat bilgisi tutmasını sağladım. Bu sınıfa ait 1 parametrelili, 2 parametrelili ve 3 parametrelili constructor yazdım. Bu sınıfa ait objeler oluşturulurken 1 parametrelili objeler oluşturulduğunda diğer parametreler null olarak ilklendirilecektir ve işlemler ona göre yapılacaktır. Burada general tree yapısını oluştururken kitabın kaynak kodlarından faydalandım.

3 RESULT

3.1 Test Cases

3.1.1 Main Test :

Part1Main ve Part2Main sınıflarını yazarak yaptığım işlemlerin sonuçlarını test ettim.

Part1 için oluşturduğum .txt dosyalarındaki verileri okuyarak işlemleri gerçekleştirdim.

Part2 için sınıf içerisinde örnek objeler oluşturarak işlemleri gerçekleştirdim.

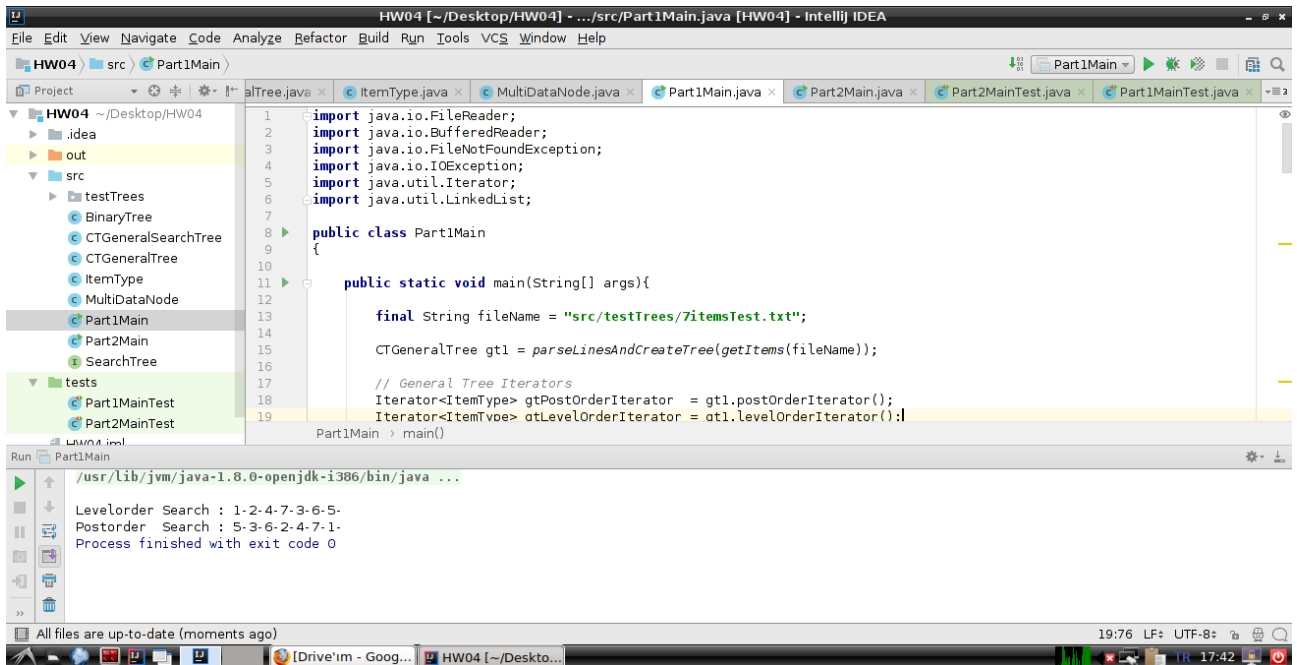
3.1.2 Unit Test :

Part1MainTest ve Part2MainTest ile Unit testlerini gerçekleştirdim.

3.2 Running Results

Part1Main.java

Bu sınıf ders asistanımız tarafından örnek olarak verilen bir ağaç yapısını, bir .txt dosyasına kaydettikten sonra dosyadan okuma, onları parse etme ve istenen levelOrderSearch() ve postOrderSearch() işlemlerini gerçekleştirmektedir.



```
HW04 [~/Desktop/HW04] - .../src/Part1Main.java [HW04] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

HW04 [~/Desktop/HW04]
  src
    testTrees
      BinaryTree
      CTGeneralSearchTree
      CTGeneralTree
      ItemType
      MultiDataNode
      Part1Main
      Part2Main
      SearchTree
    tests
      Part1MainTest
      Part2MainTest

1  import java.io.FileReader;
2  import java.io.BufferedReader;
3  import java.io.FileNotFoundException;
4  import java.io.IOException;
5  import java.util.Iterator;
6  import java.util.LinkedList;
7
8  public class Part1Main
9  {
10
11     public static void main(String[] args){
12
13         final String fileName = "src/testTrees/7itemsTest.txt";
14
15         CTGeneralTree gtl = parseLinesAndCreateTree(getItems(fileName));
16
17         // General Tree Iterators
18         Iterator<ItemType> gtPostOrderIterator = gtl.postOrderIterator();
19         Iterator<ItemType> gtLevelOrderIterator = gtl.levelOrderIterator();

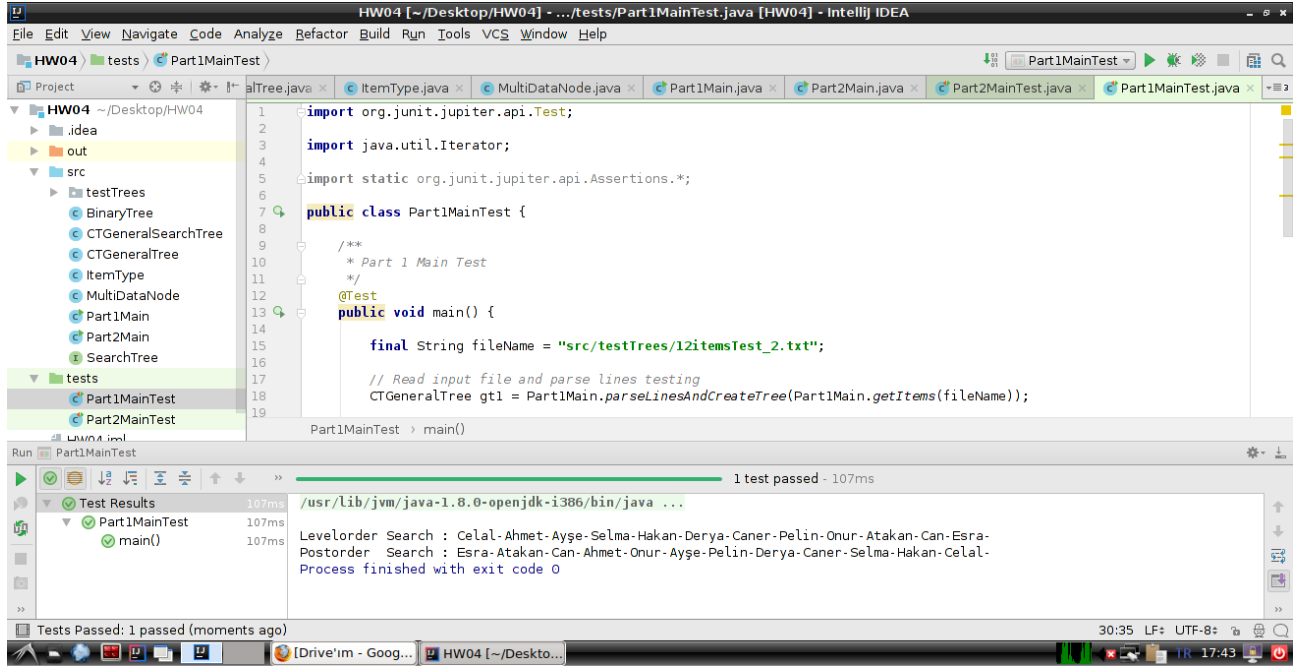
Part1Main > main()

Run: Part1Main
/usr/lib/jvm/java-1.8.0-openjdk-1386/bin/java ...
Levelorder Search : 1-2-4-7-3-6-5-
Postorder Search : 5-3-6-2-4-7-1-
Process finished with exit code 0

All files are up-to-date (moments ago)
19:76 LF+ UTF-8+
HW04 [~/Desktop/...
```

Part1MainTest.java

Bu sınıf Unit test amacıyla yazıldı. İstenen işlemlerin sonuçları görüntülendi.



The screenshot shows the IntelliJ IDEA interface with the `Part1MainTest.java` file open. The code defines a `Part1MainTest` class with a `main` method that tests the `CTGeneralTree` class. The test results window shows that the test passed successfully.

```
import org.junit.jupiter.api.Test;
import java.util.Iterator;
import static org.junit.jupiter.api.Assertions.*;

public class Part1MainTest {
    /**
     * Part 1 Main Test
     */
    @Test
    public void main() {
        final String fileName = "src/testTrees/12itemsTest_2.txt";

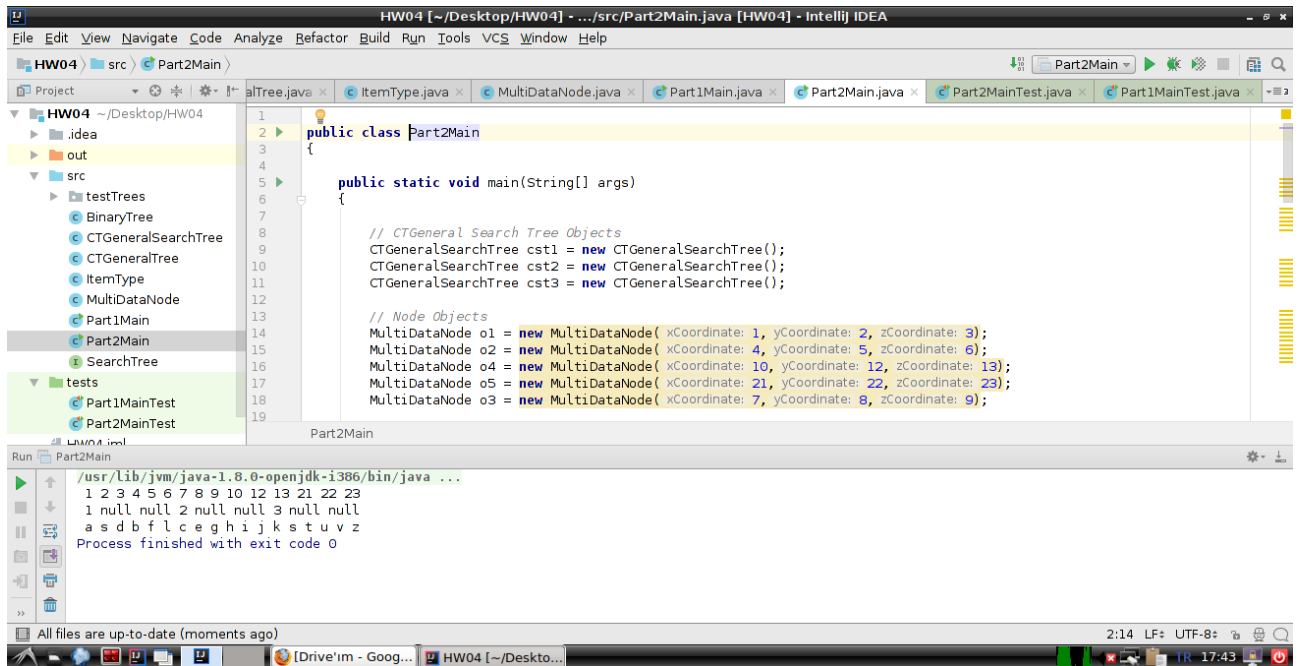
        // Read input file and parse lines testing
        CTGeneralTree gt1 = Part1Main.parseLinesAndCreateTree(Part1Main.getItems(fileName));
    }
}
```

Test Results: 1 test passed - 107ms

Levelorder Search : Celal-Ahmet-Ayşe-Selma-Hakan-Derya-Caner-Pelin-Onur-Atakan-Can-Esra-Postorder Search : Esra-Atakan-Can-Ahmet-Onur-Ayşe-Pelin-Derya-Caner-Selma-Hakan-Celal-Process finished with exit code 0

Part2Main.java

Bu sınıf Part2 probleminin çözümüne yönelik işlemler içermektedir. Örnekler objeler oluşturularak ağaca ekleme işlemleri yapılmaktadır. Girilmeyen boyutlara null değeri atanmaktadır.



The screenshot shows the IntelliJ IDEA interface with the `Part2Main.java` file open. The code defines a `Part2Main` class with a `main` method that creates and prints a search tree. The execution results window shows the output of the program.

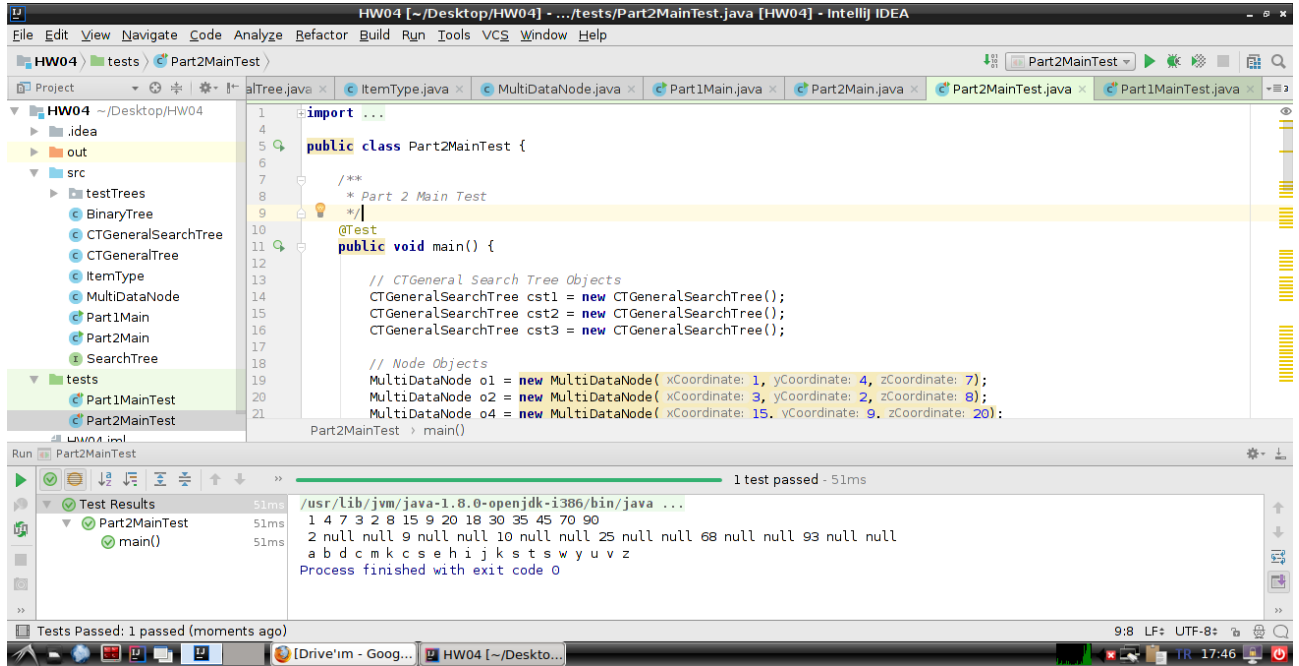
```
public class Part2Main {
    public static void main(String[] args) {
        // CTGeneral Search Tree Objects
        CTGeneralSearchTree cst1 = new CTGeneralSearchTree();
        CTGeneralSearchTree cst2 = new CTGeneralSearchTree();
        CTGeneralSearchTree cst3 = new CTGeneralSearchTree();

        // Node Objects
        MultiDataNode o1 = new MultiDataNode( xCoordinate: 1, yCoordinate: 2, zCoordinate: 3);
        MultiDataNode o2 = new MultiDataNode( xCoordinate: 4, yCoordinate: 5, zCoordinate: 6);
        MultiDataNode o4 = new MultiDataNode( xCoordinate: 10, yCoordinate: 12, zCoordinate: 13);
        MultiDataNode o5 = new MultiDataNode( xCoordinate: 21, yCoordinate: 22, zCoordinate: 23);
        MultiDataNode o3 = new MultiDataNode( xCoordinate: 7, yCoordinate: 8, zCoordinate: 9);
    }
}
```

1 2 3 4 5 6 7 8 9 10 12 13 21 22 23
1 null null 2 null null 3 null null
a s d b f l c e g h i j k s t u v z
Process finished with exit code 0

Part2MainTest.java

Bu sınıf Unit test amacıyla yazıldı. İstenen işlemlerin sonuçları görüntülendi.



The screenshot shows the IntelliJ IDEA IDE with the `Part2MainTest.java` file open. The code defines a `Part2MainTest` class with a `main` method. The `main` method initializes three `CTGeneralSearchTree` objects and three `MultiDataNode` objects. It then prints the results of a search operation. The output shows a sequence of numbers and null values, followed by a string of letters.

```
1 import ...
2
3
4
5 public class Part2MainTest {
6
7     /**
8      * Part 2 Main Test
9      */
10    @Test
11    public void main() {
12
13        // CTGeneral Search Tree Objects
14        CTGeneralSearchTree cst1 = new CTGeneralSearchTree();
15        CTGeneralSearchTree cst2 = new CTGeneralSearchTree();
16        CTGeneralSearchTree cst3 = new CTGeneralSearchTree();
17
18        // Node Objects
19        MultiDataNode o1 = new MultiDataNode( xCoordinate: 1, yCoordinate: 4, zCoordinate: 7);
20        MultiDataNode o2 = new MultiDataNode( xCoordinate: 3, yCoordinate: 2, zCoordinate: 8);
21        MultiDataNode o4 = new MultiDataNode( xCoordinate: 15, yCoordinate: 9, zCoordinate: 20);
22    }
```

The Run window shows the test results for `Part2MainTest`. The `main` method passed the test. The output of the test is as follows:

```
1 4 7 3 2 8 15 9 20 18 30 35 45 70 90
2 null null 9 null null 10 null null 25 null null 68 null null 93 null null
a b d c m k c s e h i j k s t s w y u v z
Process finished with exit code 0
```

4 Time Complexity

Ağaca eleman ekleme işlemi ve ağaç içerisinde eleman arama işlemi recursive olarak yaptım. Bu işlemlerin karmaşıklığı $O(\log n)$ olmaktadır.