

Gebze Technical University

Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

Celal TEMİZ

101044070

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

1.1 Problem Solution Approach

Bu problemin çözümünde red black tree' nin worst bir durum oluşturmamasını sağlamak için **ağacın denge durumunu oluştururken hem left hem de right rotation yapacak şekilde** belirli aralıklarda random sayıların üretilmesi sağlanmıştır. Önce küçük bir değer üretilip köke konulmuş, ardından belirli aralıkta bu değerden büyük bir değer üretilmiş ve son olarak bu üretilen iki değer arasında olacak şekilde bir random sayı üretilerek ağacın bu son hamlede double rotation yapması sağlanmıştır. Kısaca bu yapının worst durumu için ağaca eklenen elemanların, ağacı double rotation yapmaya zorlayacak şekilde eklenmesi sağlanmıştır.

Bir sonraki adıma geçerken, üretilen bu değerlerden büyük bir sayı ve bu sayıdan küçük ama ilk adımda üretilen 3 farklı sayıdan büyük bir değer üretilerek 6 uzunluğunda bir worst red black tree tasarlanmıştır.

Örneğin sayıları üretilme şekli aşağıdaki gibidir :

100 – 500 – 300
700 – 600
900 – 800
1100 – 1000
1300 – 1200
1500 – 1400
1700 – 1600
1900 – 1800
2100 – 2000
2300 – 2200
2500 – 2500

Problemin çözümünde RedBlackTree adında generic bir sınıf implement edilmiştir. Ayrıca implementasyon için gerekli BinarySearchTree, BinarySearchTreeWithRotate, BinaryTree sınıfları ve SearchTree interface' i kullanılmıştır.

BinarySearchTreeWithRotate sınıfı içerisinde rotateLeft() methodunun implementasyonu, rotateRight() methodunun simetriği şeklinde implement edilmiştir.

RedBlackTree sınıfı içerisinde add() methodunda item > localRoot.data durumu, diğer duruma simetrik şekilde implement edilmiştir.

RedBlackTree sınıfı içerisinde moveBlackDown() methodunun implementasyonun Chapter 9 – Figure 9.26 – 9.27' de yer alan örnek görselinden faydalanarak implement edilmiştir.

1.2 Test Cases

Test 1 Elemanları :

333 – 107 – 91 – 40 – 102 – 145 – 125 – 220 – 961 – 735 – 512 – 925 – 1092 – 1059 – 983 – 1064 – 1116 – 1099 – 1339 – 1166 – 1410 – 1357 – 1460

Test 2 Elemanları :

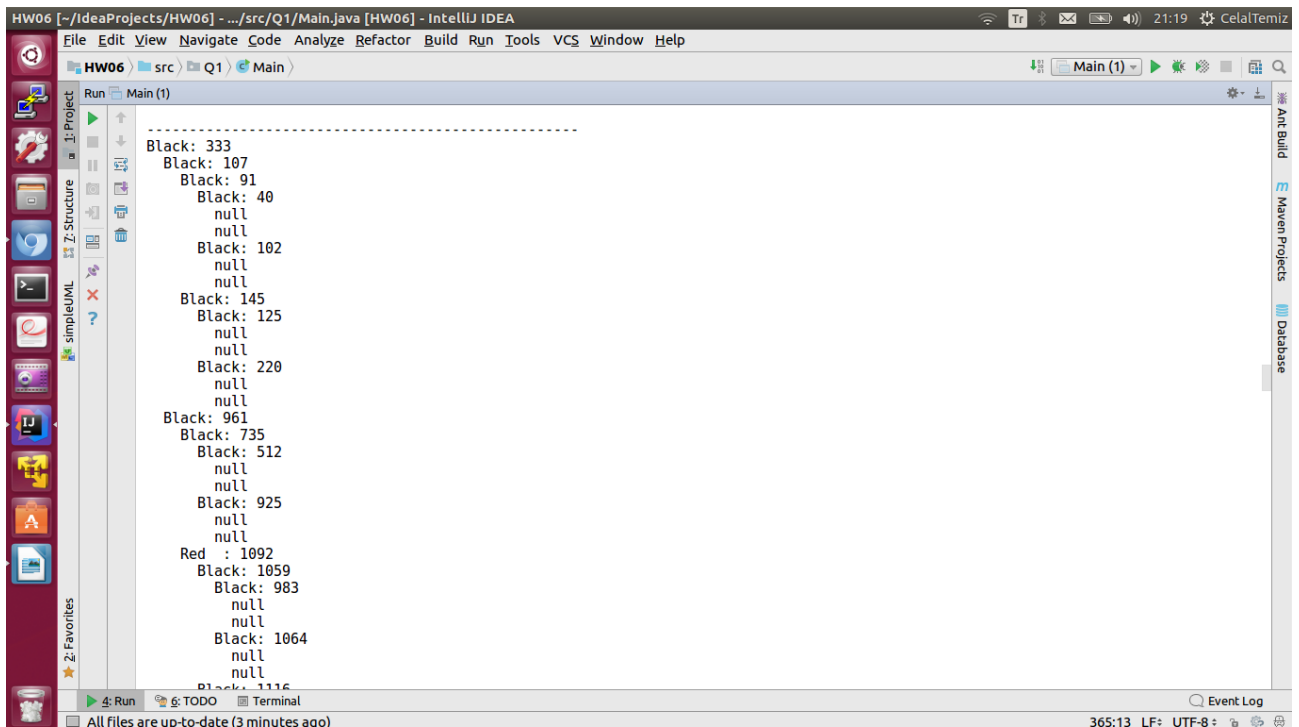
629 – 585 – 14 – 13 – 496 – 616 – 615 – 621 – 794 – 777 – 672 – 779 – 1148 – 1118 – 1062 – 1144 – 1209 – 1160 – 1407 – 1396 – 1468 – 1426 – 1481

Yukarıdaki değerler main içerisinde oluşturulan RedBlackTree objesine eleman olarak eklenmektedir.

JUNIT main test proje test klasörüne eklenmiştir.

1.3 Running Commands and Results

Test 1 Sonuçları :

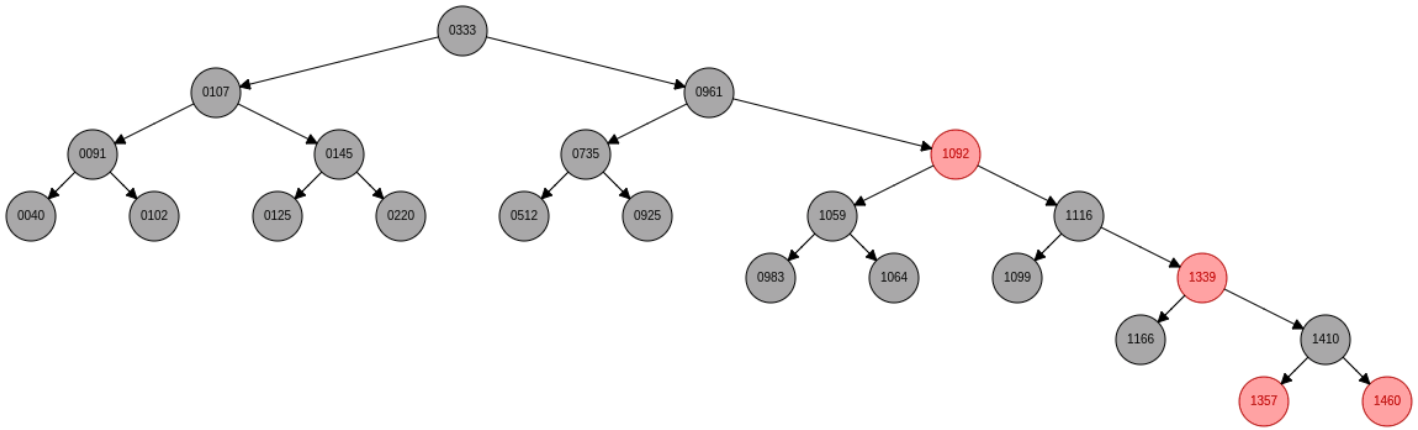


Şekil 1.3.1

```
HW06 [-/IdeaProjects/HW06] - .../src/Q1/Main.java [HW06] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HW06 > src > Q1 > Main >
Run Main (1)
null
Black: 961
Black: 735
Black: 512
null
null
Black: 925
null
null
Red : 1092
Black: 1059
Black: 983
null
null
Black: 1064
null
null
Black: 1116
Black: 1099
null
null
Red : 1357
Black: 1166
null
null
Black: 1410
Red : 1399
null
null
Red : 1460
null
null
----- TEST 2 -----
4: Run 6: TODO Terminal
All files are up-to-date (4 minutes ago)
365:13 LF: UTF-8
```

Şekil 1.3.2

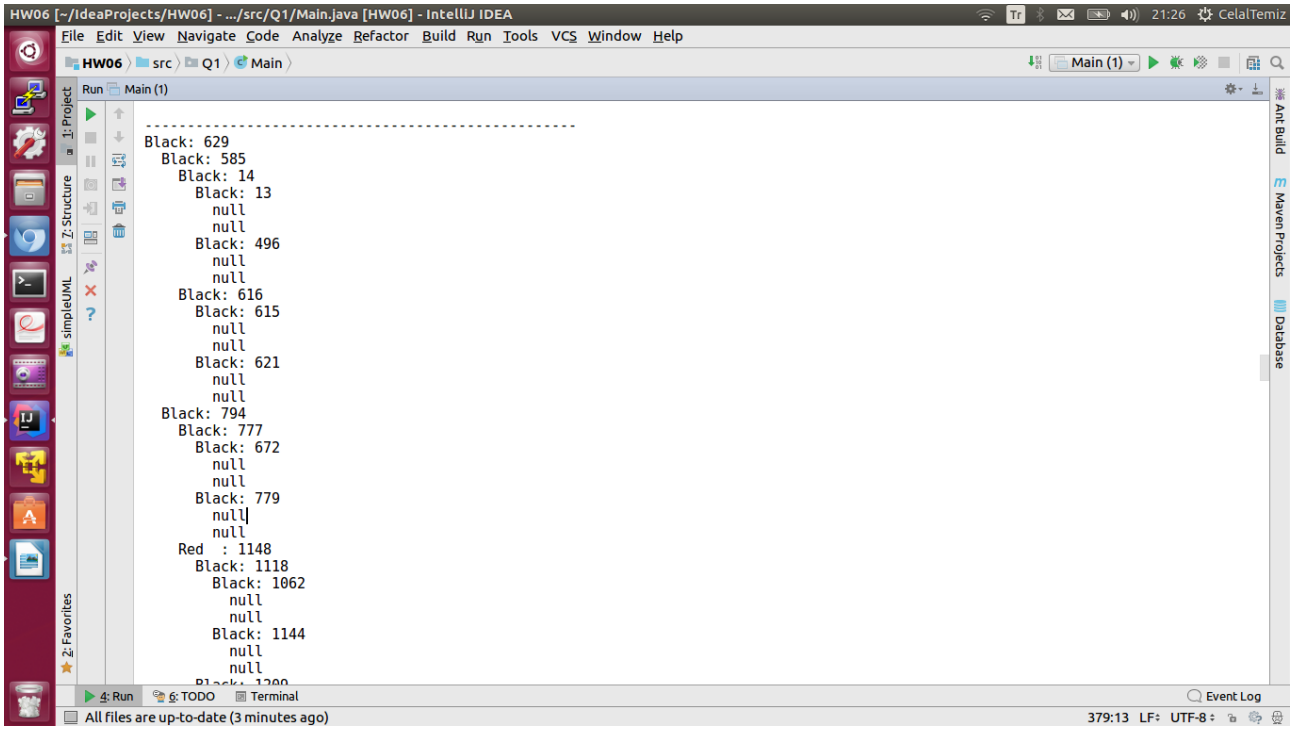
Oluşan Red Black Tree simülasyonu :



Animation Completed

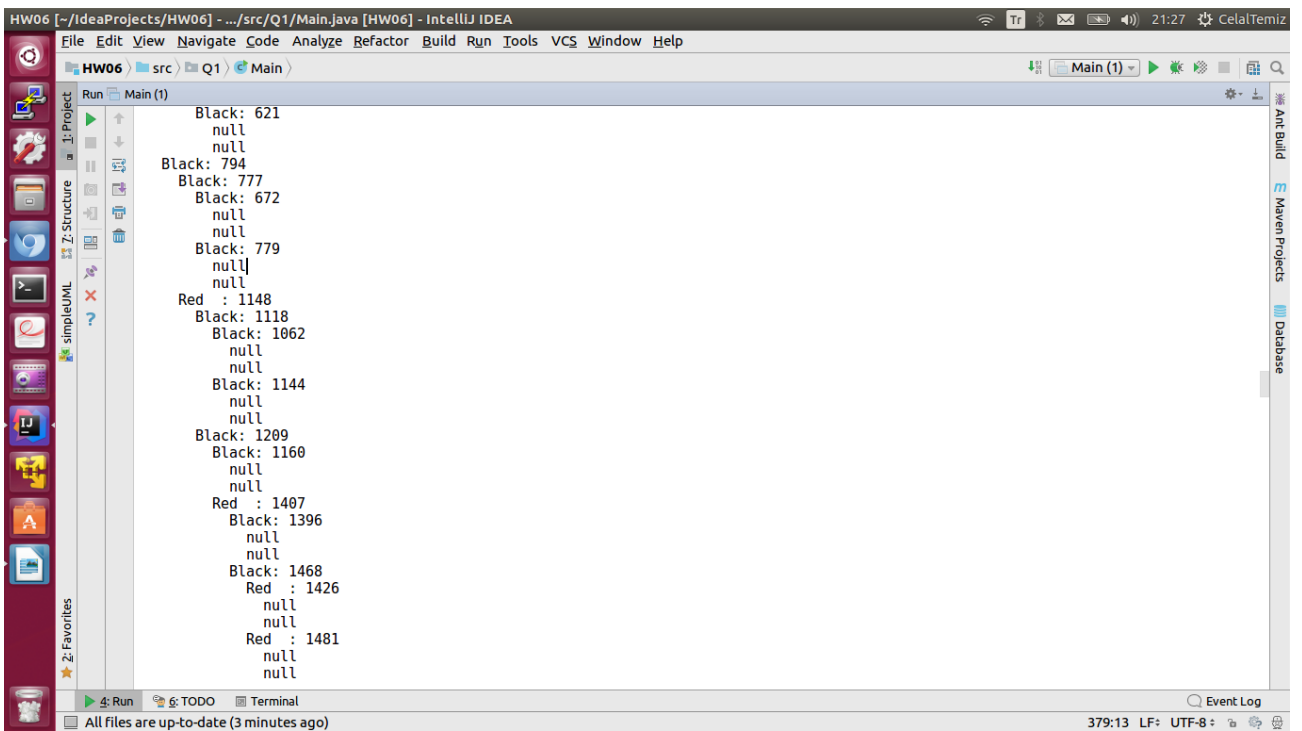
Şekil 1.3.3

Test 2 Sonuçları



```
Black: 629
Black: 585
Black: 14
Black: 13
null
null
Black: 496
null
null
Black: 616
Black: 615
null
null
Black: 621
null
null
Black: 794
Black: 777
Black: 672
null
null
Black: 779
null
null
Red : 1148
Black: 1118
Black: 1062
null
null
Black: 1144
null
null
Black: 1209
```

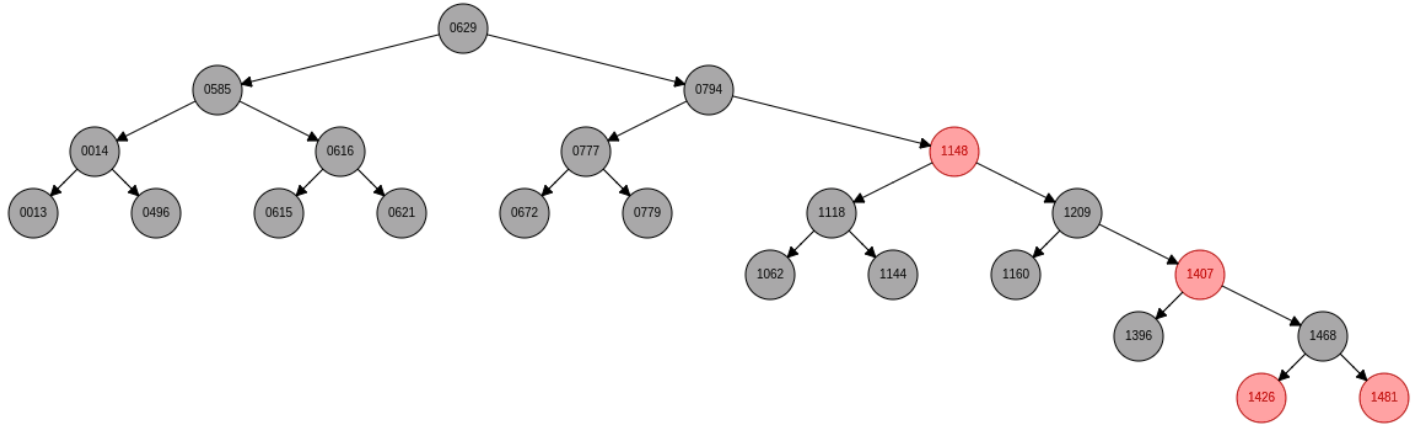
Şekil 1.3.4



```
Black: 621
null
null
Black: 794
Black: 777
Black: 672
null
null
Black: 779
null
null
Red : 1148
Black: 1118
Black: 1062
null
null
Black: 1144
null
null
Black: 1209
Black: 1160
null
null
Red : 1407
Black: 1396
null
null
Black: 1468
Red : 1426
null
null
Red : 1481
null
null
```

Şekil 1.3.5

Oluşan Red Black Tree simulasyonu :



Animation Completed

Şekil 1.3.6

2 binarySearch method

2.1 Problem Solution Approach

Bu problemin çözümünde Btree' nin BinarySearch methodu implement edilmiştir. Bu method parametre olarak sırasıyla aranan eleman, node' un elemanlarını bulunduran dizi, dizinin başlangıç indexi ve dizinin boyunutu almaktadır.

Alınan dizi iki parçaya bölünür ve ortadaki elemanın aranan eleman olup olmadığına bakılır. Eğer aranan eleman bu indexte ise bu index değeri, elemanın eklenmesi için return edilir. Eğer aranan eleman burada yer alan elemandan küçükse dizinin sol tarafına, büyükse sağ tarafına bakılır. Aynı şekilde dizi tekrar iki parçaya bölünerek ortadaki elemanın konumuna göre aranan elemanın indexi bulunur.

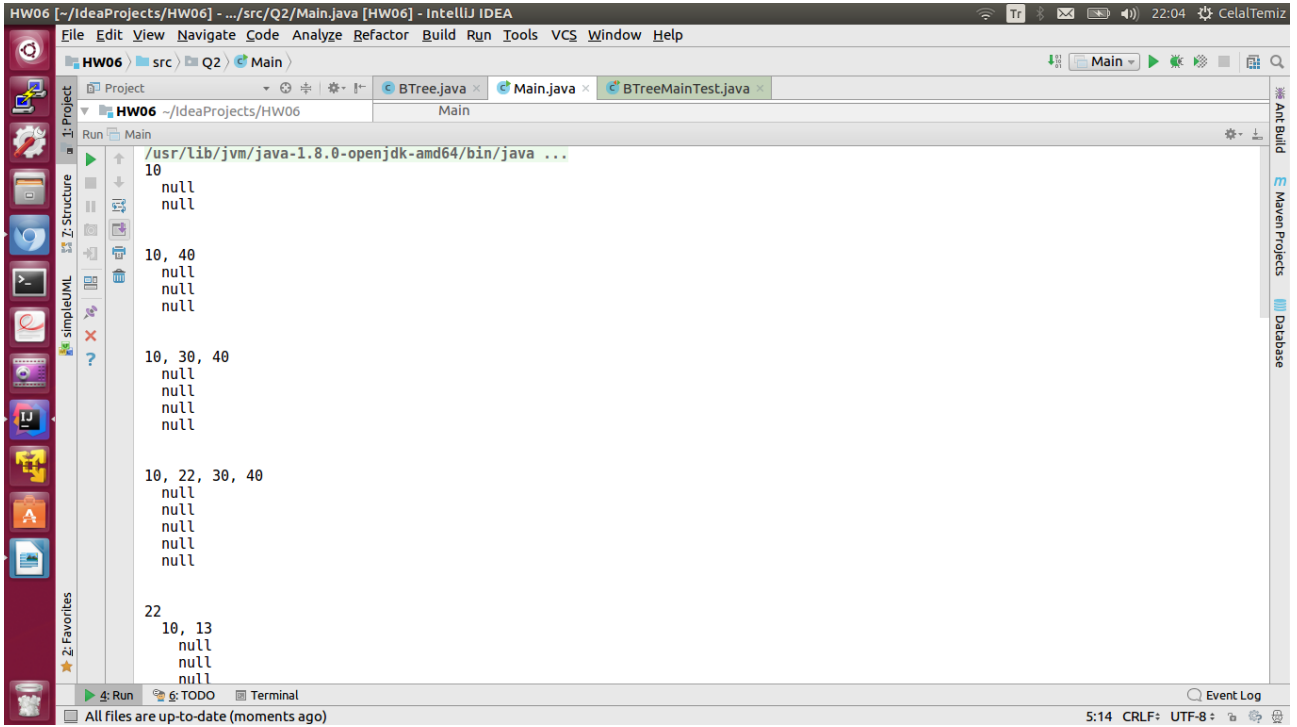
Problemin çözümü için Btree generic sınıfı implement edilmektedir. Ayrıca Comparable sınıfı extend edilmiş, SearchTree interface' i implement edilmiştir.

2.2 Test Cases

Order değeri 5 olan bir Btree' ye sırasıyla 10, 40, 30, 22, 13, 20, 18, 15 elemanları eklenmiştir. BinarySearch() methodu add() methodu içerisinde çağırılarak dönen index değerine göre ekleme yapmaktadır.

JUNIT main test proje test klasörüne eklenmiştir.

2.3 Running Commands and Results



```
HW06 [~/IdeaProjects/HW06] - .../src/Q2/Main.java [HW06] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HW06 [src] Q2 Main
Project HW06 ~/IdeaProjects/HW06 Main
Run Main
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
10
null
null

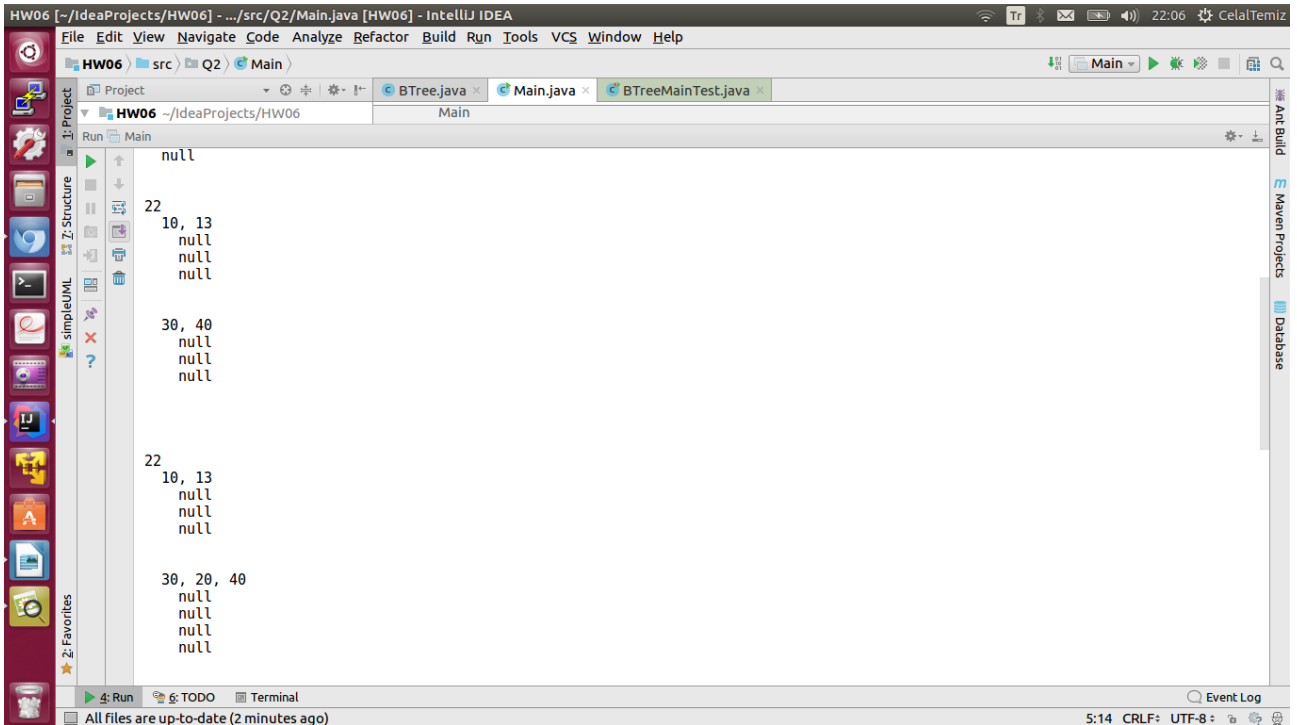
10, 40
null
null
null

10, 30, 40
null
null
null
null

10, 22, 30, 40
null
null
null
null
null

22
10, 13
null
null
null
```

Şekil 2.3.1



```
HW06 [~/IdeaProjects/HW06] - .../src/Q2/Main.java [HW06] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HW06 [src] Q2 Main
Project HW06 ~/IdeaProjects/HW06 Main
Run Main
null

22
10, 13
null
null
null

30, 40
null
null
null

22
10, 13
null
null
null

30, 20, 40
null
null
null
null
```

Şekil 2.3.2

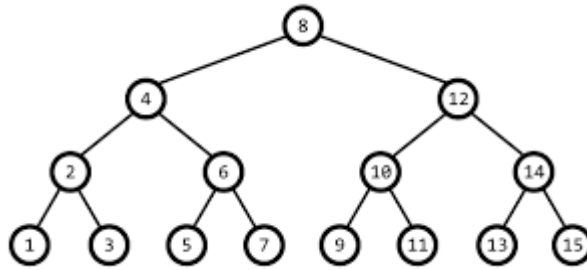
3 Project 9.5 in book

3.1 Problem Solution Approach

Bu problemin çözümüne bir binary tree alıp, onun AVL Tree olup olmadığını kontrol eden bir constructor eklenmiştir. Buna ek olarak rebalanceRight(), incrementBalance(), add() methodu içerisinde item > data olma durumu gibi istenen methodların implemantasyonu gerçekleştirilmiştir. Yapı bir binary tree olduğu için sağ ve sol birbirinin simetriği gibi görünüp, işlemler benzer şekilde gerçekleşmektedir.

3.2 Test Cases

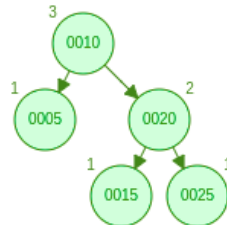
Örnek bir binary tree oluşturulmuş ve bunun AVL tree olup olmadığı test edilmiştir. Örnek binary tree aşağıdaki gibidir.



Şekil 3.2.1

Bu yapıdan 13, 14, 15 elemanları silindiğinde AVL tree yapısının bozulduğu JUNIT maintest içerisinde gözlenmektedir.

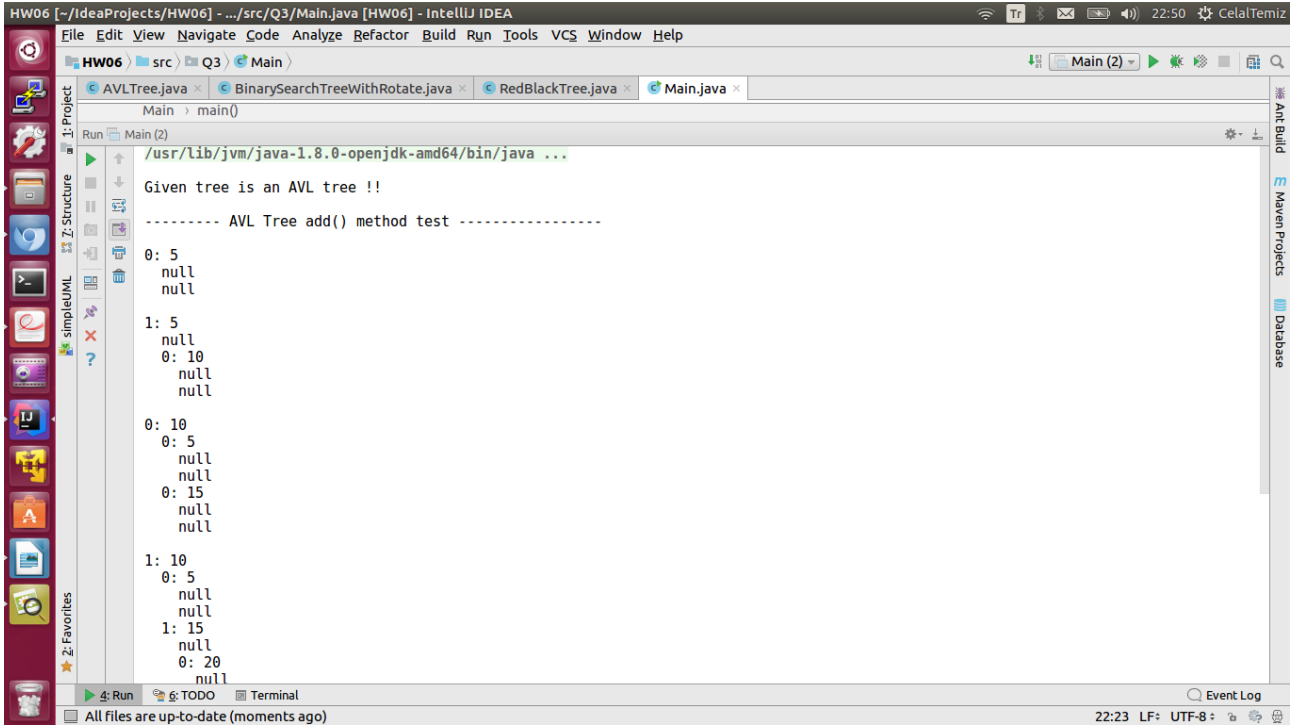
AVL Tree sınıfı içerisine yazılan ve düzenlenen methodlardan sonra örnek bir AVL Tree' nin oluşturulması sağlanmış ve test edilmiştir.



Şekil 3.2.2

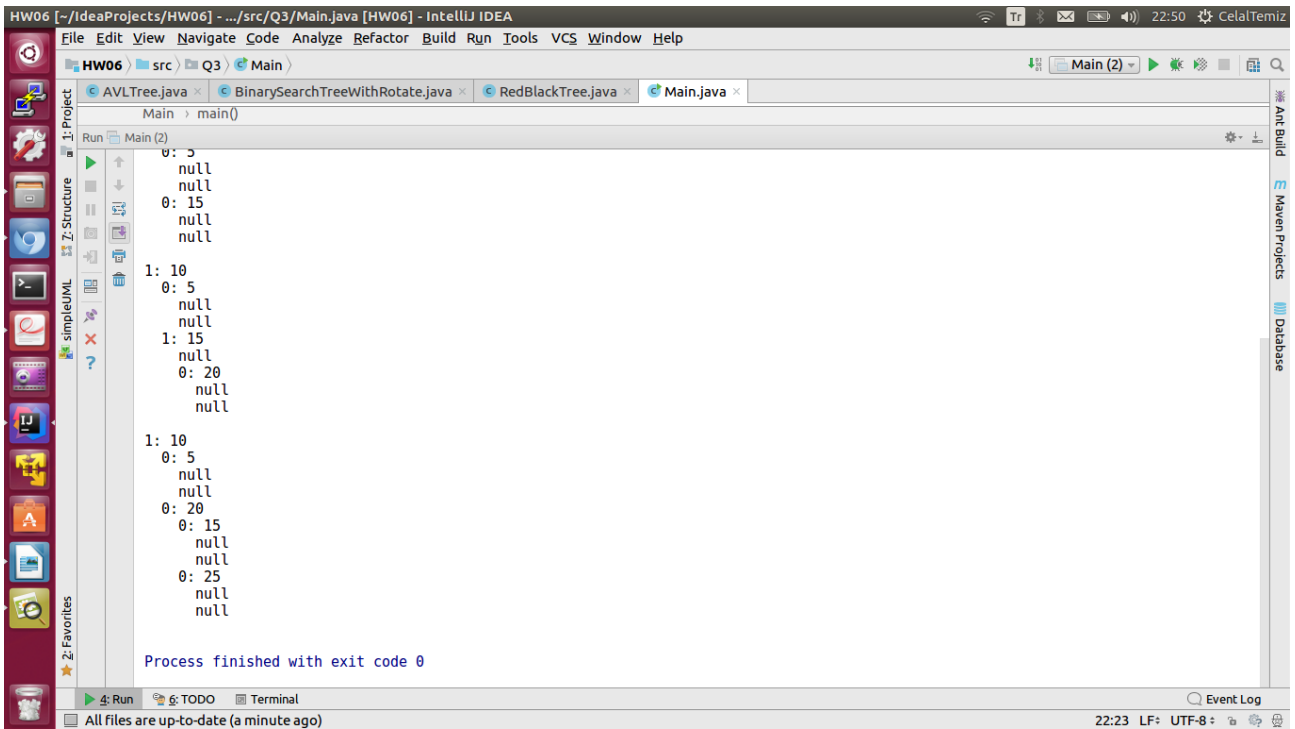
JUNIT main test proje test klasörüne eklenmiştir.

3.3 Running Commands and Results



```
HW06 [~/IdeaProjects/HW06] - .../src/Q3/Main.java [HW06] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HW06 [src] Q3 Main
Main main()
Run Main (2)
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
Given tree is an AVL tree !!
----- AVL Tree add() method test -----
0: 5
  null
  null
1: 5
  null
  0: 10
    null
    null
0: 10
  0: 5
    null
    null
  0: 15
    null
    null
1: 10
  0: 5
    null
    null
  1: 15
    null
    0: 20
      null
      null
4: Run 6: TODO Terminal
All files are up-to-date (moments ago) 22:23 LF+ UTF-8
```

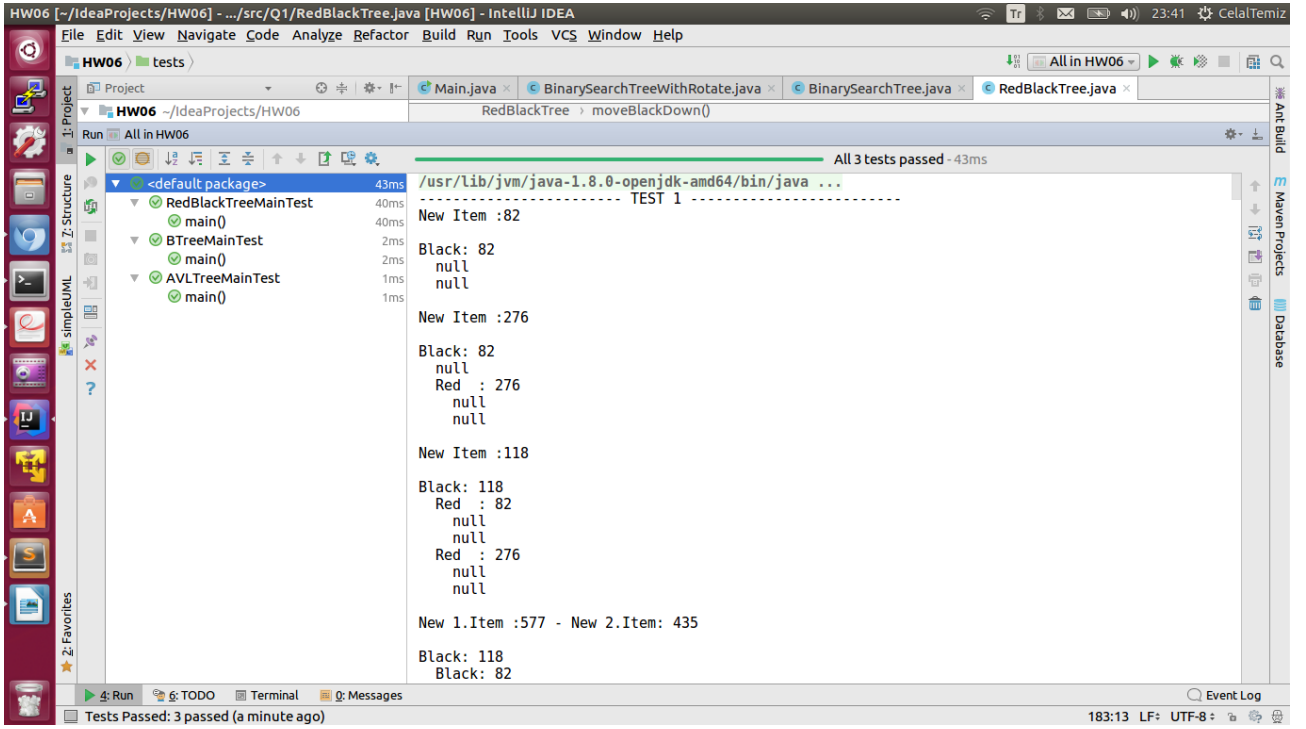
Şekil 3.3.1



```
HW06 [~/IdeaProjects/HW06] - .../src/Q3/Main.java [HW06] - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
HW06 [src] Q3 Main
Main main()
Run Main (2)
0: 5
  null
  null
0: 15
  null
  null
1: 10
  0: 5
    null
    null
  1: 15
    null
    0: 20
      null
      null
1: 10
  0: 5
    null
    null
  0: 20
    0: 15
      null
      null
    0: 25
      null
      null
Process finished with exit code 0
4: Run 6: TODO Terminal
All files are up-to-date (a minute ago) 22:23 LF+ UTF-8
```

Şekil 3.3.2

JUNIT MAIN TEST SONUÇLARI :



Şekil 3.3.3