

**Gebze Technical University
Computer Engineering**

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

**CELAL CAN KAYA
161044014**

Course Assistant: Fatma Nur Esirci

1 Worst RedBlack Tree

1.1 Problem Solution Approach

Red-Black Tree Add Metodu

- 1- Eğer kök ile eleman eşitse, kökü return et.
- 2- Eğer eleman kökten küçük ise
 - a) Kökün sol node'u null ise elemanı yerleştir ve elemanı return et.
 - b) Değilse
 - a. Kökün iki child'ıda kırmızı ise renkleri swap et.
 - b. Metodu kökün sol child'ı ile rekürsif olarak çağır.
 - c. Kökün solu kırmızı ise
 - Kökün solunun soluda kırmızı ise
✓ Sağa rotate et.
 - Kökün solunun sağıda kırmızı ise
✓ Sağa rotate et.
 - d. Kökü return et
- 3- İlk 2 durumda doğru değilse,
 - a) Kökün sağ node'u null ise elemanı yerleştir ve elemanı return et.
 - b) Değilse
 - a. Kökün iki child'ıda kırmızı ise renkleri swap et.
 - b. Metodu kökün sağ child'ı ile rekürsif olarak çağır.
 - c. Kökün sağı kırmızı ise
 - Kökün sağının sağıda kırmızı ise
✓ Sola rotate et.
 - Kökün sağının soluda kırmızı ise
✓ Sola rotate et.
 - d. Kökü return et

Height'i 6 olan bir worst RedBlack Tree oluşturulması isteniyor.RedBlack Tree bir dengeli ağaç olduğu için worst case'i en az dengeli olduğu durumdur.Bunun içinde en az sayıda eleman ile 6 yüksekliğine ulaşmamız lazım.En az sayıda eleman ile ulaşmamız gerektiği için elemanları artan/azalan şekilde eklememiz lazım.Ayrıca kod her çalıştırıldığında farklı elemanlardan oluşturulan bir tree ekrana bastırmamız bekleniyor.Bu nedenle 0'dan 9999'a kadar 22 tane(6 Height için en az 22 eleman gerekli) farklı sayı üretiyorum.Daha sonra Java'nın sort'unu kullanarak bu elemanları büyükten küçüğe ve küçüktan büyüğe olacak şekilde sıralayıp döngü ile sırasıyla ağaca ekliyorum.

RedBlack Tree Class'ı kullanıldı.

RedBlack Tree Class'ı, BinarySearchWithRotate Class'ından extend edildiği için

BinarySearchWithRotate Class'ı kullanıldı.

BinarySearchWithRotate Class'ı, BinarySearchTree Class'ını extend ettiği için BinarySearchTree Class'ı kullanıldı.

BinarySearchTree Class'ı, BinarySearch Class'ını extend ettiği için BinarySearch Class'ı kullanıldı ve aynı zamanda SearchTree Interface'sini implement ettiği için SearchTree interface'si kullanıldı.

1.2 Test Cases

Worst bir RedBlack Tree İçin ulaşmak istediğimiz yüksekliğe en az sayıda eleman ekleyerek ulaşmamız gerekir. Bu nedenle elemanları sürekli artan veya sürekli azalan bir şekilde tree'ye eklememiz gerekiyor. İlk test case'imde elemanları artan bir şekilde, ikinci test case'imde ise elemanları azalan bir şekilde sıralayarak tree'ye ekledim. Kodu her çalıştırdığımızda 0'dan 9999'a kadar 22 tane farklı eleman üretiyor ve daha sonra bunları artan/azalan şeklinde sıralıyor. Sıralanan bu elemanları tree'ye sırayla ekliyorum.

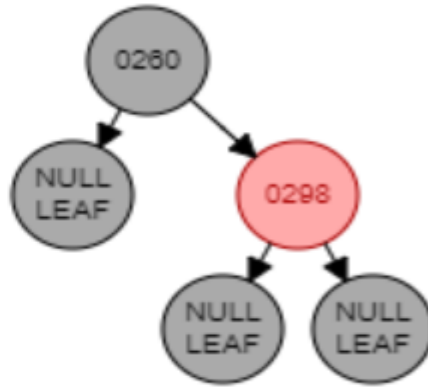
1.3 Running Commands and Results

Her eleman ekledikten sonra tree'yi ekrana bastırıyorum fakat her elemandan sonra tree'nin yeni halinin ekran görüntüsünü koyarsam çok uzayacağından sadece height'i arttıran ekran görüntülerini koydum. Her elemandan sonra tree'nin yeni halini görmek isterseniz kodu çalıştırıp test edebilirsiniz.

TEST 1 (Sürekli Artan)

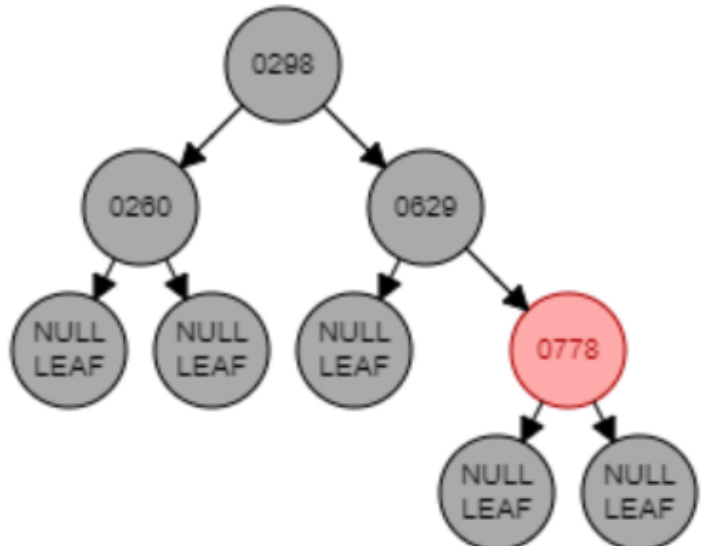
Height = 1 :

```
260 Elemanı Tree'ye Ekleniyor.  
  
Black: 260  
  null  
  null  
  
298 Elemanı Tree'ye Ekleniyor.  
  
Black: 260  
  null  
Red : 298  
  null  
  null
```



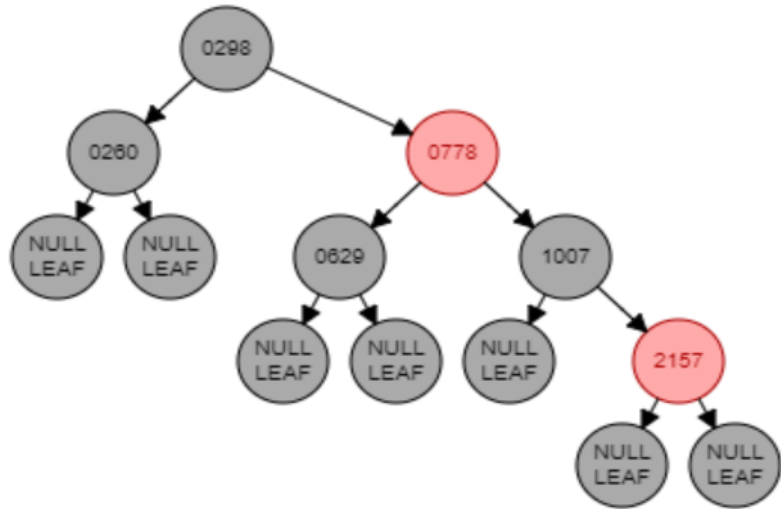
Height = 2 :

```
778 Elemanı Tree'ye Ekleniyor.  
  
Black: 298  
Black: 260  
  null  
  null  
Black: 629  
  null  
Red : 778  
  null  
  null
```



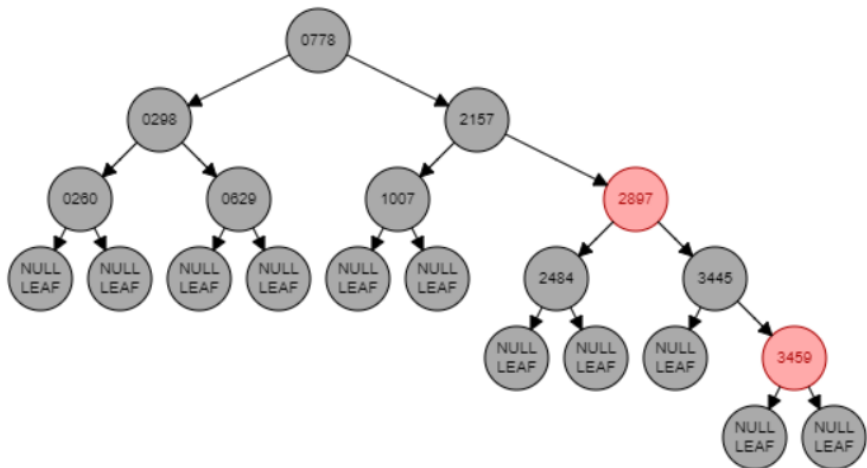
Height = 3 :

```
2157 Elemanı Tree'ye Ekleniyor.
Black: 298
  Black: 260
    null
    null
  Red : 778
    Black: 629
      null
      null
    Black: 1007
      null
      Red : 2157
        null
        null
```



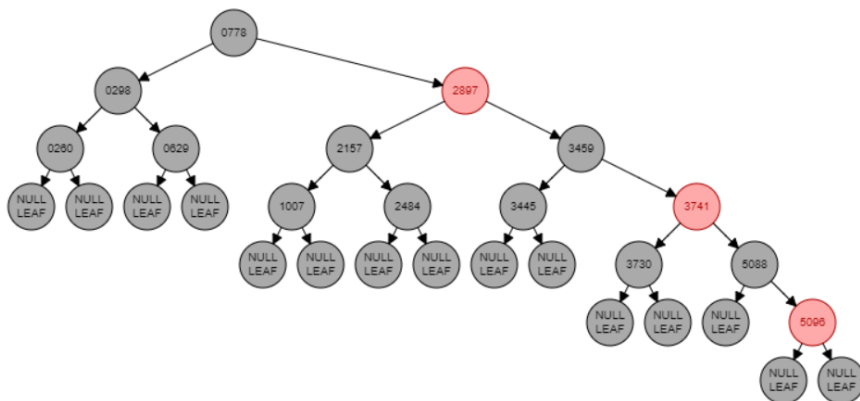
Height = 4 :

```
3459 Elemanı Tree'ye Ekleniyor.
Black: 778
  Black: 298
    Black: 260
      null
      null
    Black: 629
      null
      null
  Black: 2157
    Black: 1007
      null
      null
    Red : 2897
      Black: 2484
        null
        null
      Black: 3445
        null
        Red : 3459
          null
          null
```



Height = 5 :

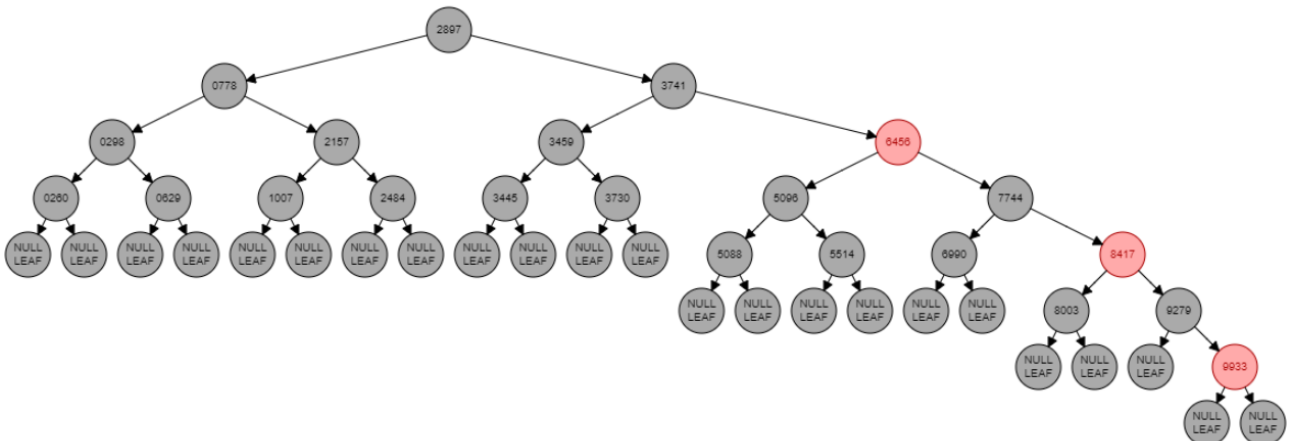
```
5096 Elemanı Tree'ye Ekleniyor.
Black: 778
  Black: 298
    Black: 260
      null
      null
    Black: 629
      null
      null
  Red : 2897
    Black: 2157
      Black: 1007
        null
        null
      Black: 2484
        null
        null
    Black: 3459
      Black: 3445
        null
        null
      Red : 3741
        Black: 3730
          null
          null
        Black: 5088
          null
          Red : 5096
            null
            null
```



Height = 6 :

9933 Elemanı Tree'ye Ekleniyor.

```
Black: 2897
  Black: 778
    Black: 298
      Black: 260
        null
        null
      Black: 629
        null
        null
    Black: 2157
      Black: 1007
        null
        null
      Black: 2484
        null
        null
  Black: 3741
    Black: 3459
      Black: 3445
        null
        null
      Black: 3730
        null
        null
    Red : 6456
      Black: 5096
        Black: 5088
          null
          null
        Black: 5514
          null
          null
      Black: 7744
        Black: 6990
          null
          null
        Red : 8417
          Black: 8003
            null
            null
          Black: 9279
            null
            Red : 9933
              null
              null
```



TEST 2 (Sürekli Azalan)

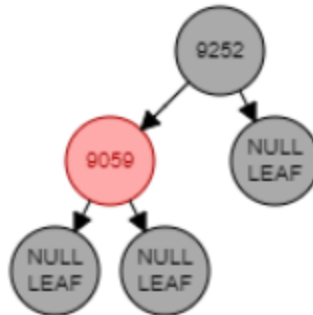
Height = 1 :

```
9252 Elemanı Tree'ye Ekleniyor.
```

```
Black: 9252  
  null  
  null
```

```
9059 Elemanı Tree'ye Ekleniyor.
```

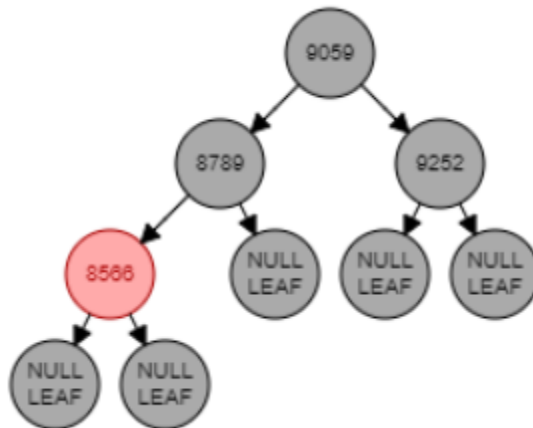
```
Black: 9252  
  Red : 9059  
    null  
    null  
    null
```



Height = 2 :

```
8566 Elemanı Tree'ye Ekleniyor.
```

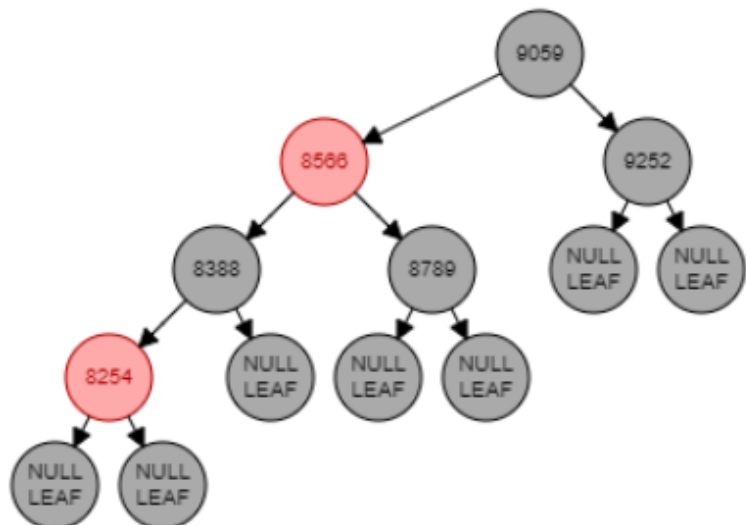
```
Black: 9059  
  Black: 8789  
    Red : 8566  
      null  
      null  
      null  
Black: 9252  
  null  
  null
```



Height = 3 :

```
8254 Elemanı Tree'ye Ekleniyor.
```

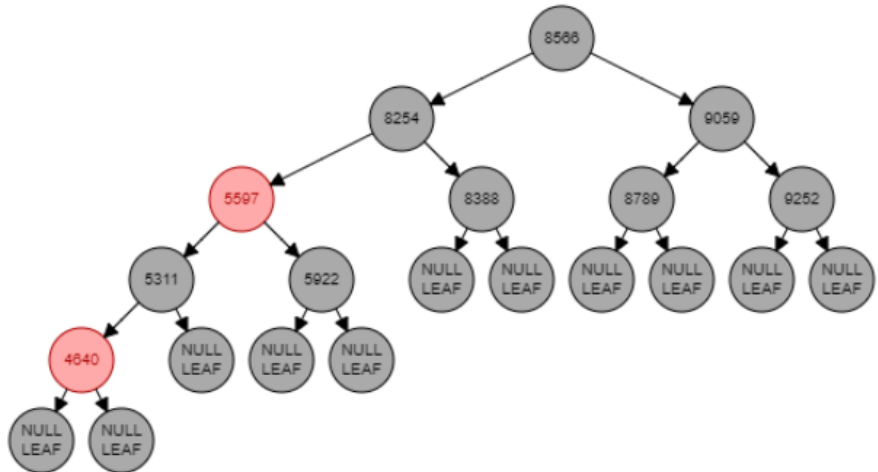
```
Black: 9059  
  Red : 8566  
    Black: 8388  
      Red : 8254  
        null  
        null  
        null  
    Black: 8789  
      null  
      null  
      null  
Black: 9252  
  null  
  null
```



Height = 4 :

4640 Elemanı Tree'ye Ekleniyor.

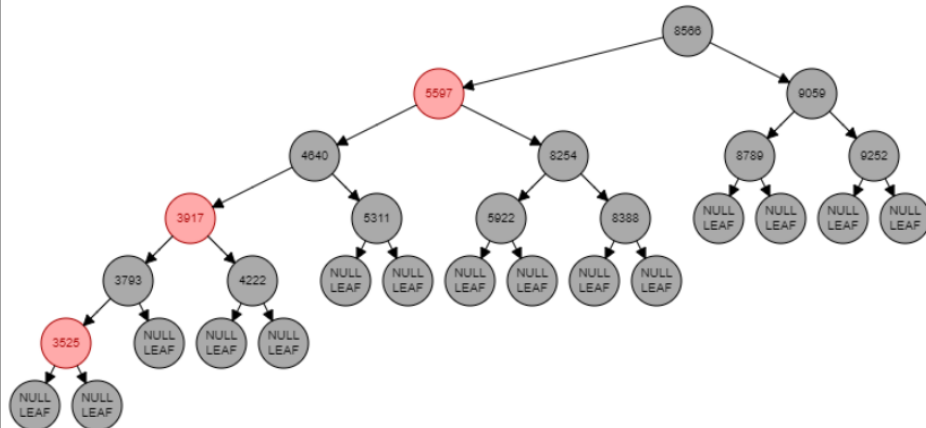
```
Black: 8566
Black: 8254
Red : 5597
Black: 5311
Red : 4640
null
null
null
Black: 5922
null
null
Black: 8388
null
null
Black: 9059
Black: 8789
null
null
Black: 9252
null
null
```



Height = 5 :

3525 Elemanı Tree'ye Ekleniyor.

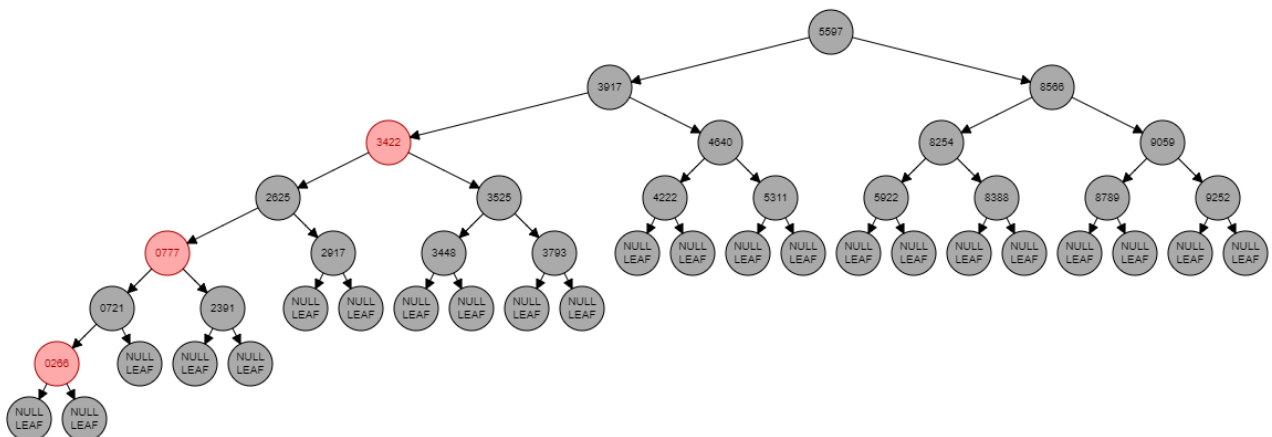
```
Black: 8566
Red : 5597
Black: 4640
Red : 3917
Black: 3793
Red : 3525
null
null
null
Black: 4222
null
null
Black: 5311
null
null
Black: 8254
Black: 5922
null
null
Black: 8388
null
null
Black: 9059
Black: 8789
null
null
Black: 9252
null
null
```



Height = 6 :

266 Elemanı Tree'ye Ekleniyor.

```
Black: 5597
  Black: 3917
    Red : 3422
      Black: 2625
        Red : 777
          Black: 721
            Red : 266
              null
              null
              null
            Black: 2391
              null
              null
          Black: 2917
            null
            null
        Black: 3525
          Black: 3448
            null
            null
          Black: 3793
            null
            null
      Black: 4640
        Black: 4222
          null
          null
        Black: 5311
          null
          null
    Black: 8566
      Black: 8254
        Black: 5922
          null
          null
        Black: 8388
          null
          null
      Black: 9059
        Black: 8789
          null
          null
        Black: 9252
          null
          null
```



2 binarySearch method

2.1 Problem Solution Approach

Pseudocode

1. Ortadaki indexi bul.
2. Eğer, üst sınır, alt sınırdan küçükse orta indexi return et.
3. Eğer üst sınır, alt sınıra eşitse orta indexi return et.
4. Eğer aranılan eleman ile orta indexteki elemanla eşitse orta indexi return et.
5. Eğer aranılan eleman, orta indexteki elemandan büyükse
 - a. Alt sınırı, orta index + 1 yaparak metodu recursive olarak çağır.
6. Eğer aranılan eleman, orta indexteki elemandan küçükse
 - a. Üst sınırı, orta index - 1 yaparak metodu recursive olarak çağır
7. Eğer üst sınır, orta indexe eşitse
 - a. Eğer aranılan eleman, orta indexteki elemandan büyükse
 - i. Orta index değeri + 1 indexindeki child ile recursive olarak çağır
 - b. Eğer aranılan eleman, orta indexteki elemandan küçükse
 - i. Orta index değeri + 1 indexindeki child ile recursive olarak çağır

BTree class'ı kullanıldı.

BTree Class'ı, SearchTree interface'sini implement ettiği için SearchTree interface'si kullanıldı.

2.2 Test Cases

BinarySearch metodu kullanılmadan zaten add metodu düzgün çalışmayacağı için BinarySearch metodunu ayrı olarak çağırmadım. Add metodunun sonucunda ortaya çıkan Tree outputlarını koydum.

Test 1 için Order'ı 6 olan bir tree oluşturdum.

Test 2 için Order'ı 4 olan bir tree oluşturdum.

2.3 Running Commands and Results

TEST 1

Çok fazla eleman eklediğim için her elemandan sonraki Tree yerine belli bölümlerdeki outputları ekledim.

5 Eleman Eklendikten Sonra :

```
5, 7, 8, 12, 15
null
null
null
null
null
null
```

0005	0007	0008	0012	0015
------	------	------	------	------

Kod kısmında 7 elemanından 4 kere ekliyorum fakat BinarySearch'te bulunduğu için sadece tree'ye eklenmiyor. Sadece ilk eklenen 7 tree'ye ekleniyor.

6 Eleman Eklendikten Sonra (Order Sınırına Ulaşıldı, Tree Bölünecek) :

```
8
 5, 7
  null
  null
  null

12, 15, 17
  null
  null
  null
  null
```



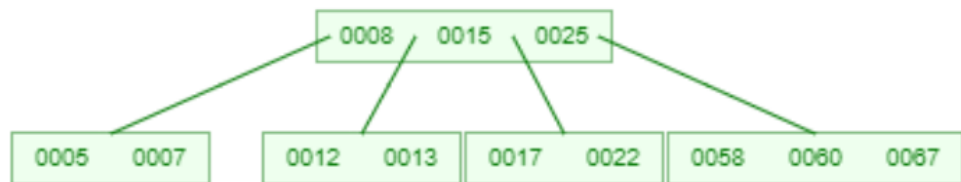
12 Eleman Eklendikten Sonra :

```
8, 15, 25
 5, 7
  null
  null
  null

12, 13
  null
  null
  null

17, 22
  null
  null
  null

58, 60, 67
  null
  null
  null
  null
```



Tüm Elemanlar Eklendikten Sonra :

```
25
 8, 15
  5, 7
   null
   null
   null

11, 12, 13
  null
  null
  null
  null

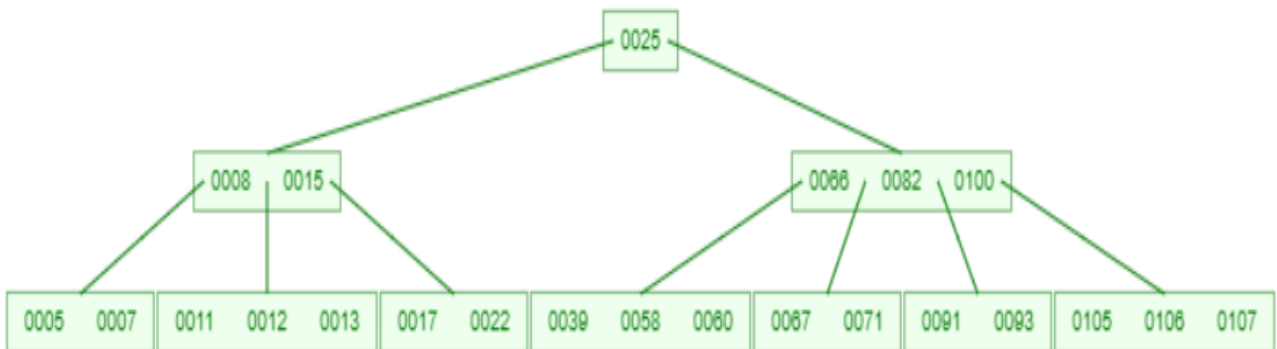
17, 22
  null
  null
  null

66, 82, 100
39, 58, 60
  null
  null
  null
  null

67, 71
  null
  null
  null

91, 93
  null
  null
  null

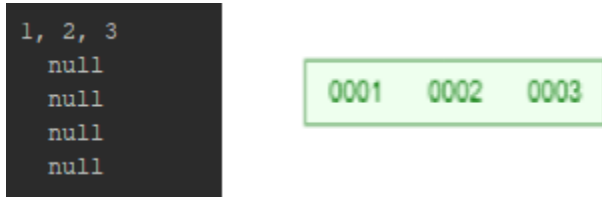
105, 106, 107
  null
  null
  null
  null
```



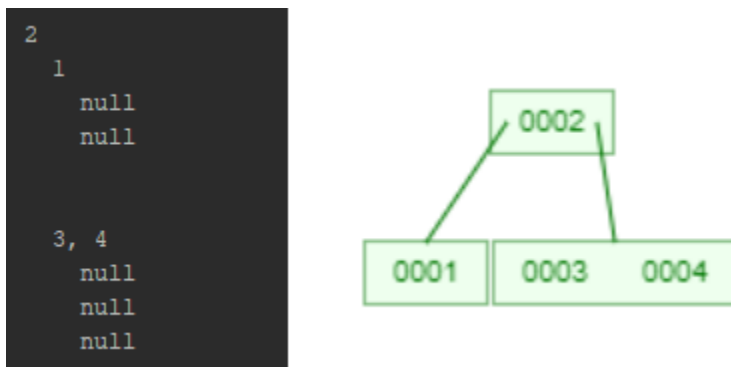
TEST 2

Order'ı 4 olan BTree'ye sıralı şekilde 1'den 15'e kadar olan elemanları ekledim.

3 Eleman Eklendikten Sonra :



4 Eleman Eklendikten Sonra :



Tüm Elemanlar Eklendikten Sonra :

```
4, 8
 2
  1
    null
    null

 3
    null
    null

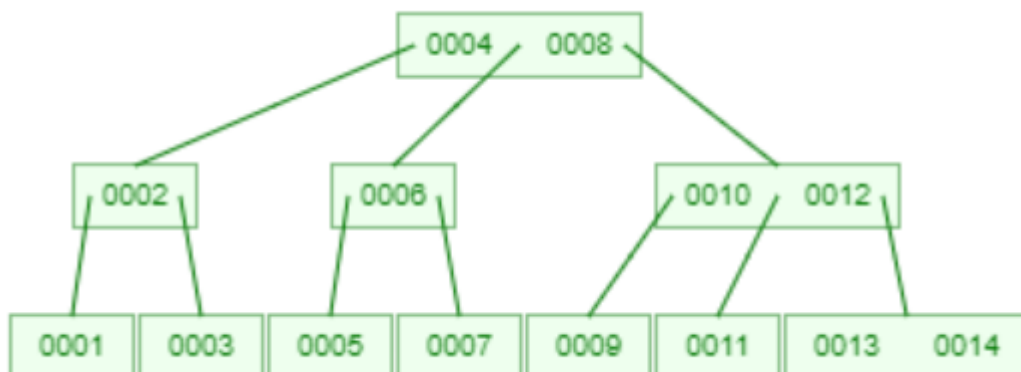
 6
  5
    null
    null

 7
    null
    null

10, 12
 9
    null
    null

11
    null
    null

13, 14
    null
    null
    null
```



3 Project 9.5 in book

3.1 Problem Solution Approach

Constructor Pseudocode

- BinaryTree'nin BST olup olmadığını kontrol et
- Değilse, AVLTree'de değildir.
- BinaryTree'nin Balanced olup olmadığını kontrol et.
- Değilse, AVLTree'de değildir.

AVLTree Class'ı, BinarySearchWithRotate Class'ından extend edildiği için BinarySearchWithRotate Class'ı kullanıldı.

BinarySearchWithRotate Class'ı, BinarySearchTree Class'ını extend ettiği için BinarySearchTree Class'ı kullanıldı.

BinarySearchTree Class'ı, BinarySearch Class'ını extend ettiği için BinarySearch Class'ı kullanıldı ve aynı zamanda SearchTree Interface'sini implement ettiği için SearchTree interface'si kullanıldı.

3.2 Test Cases

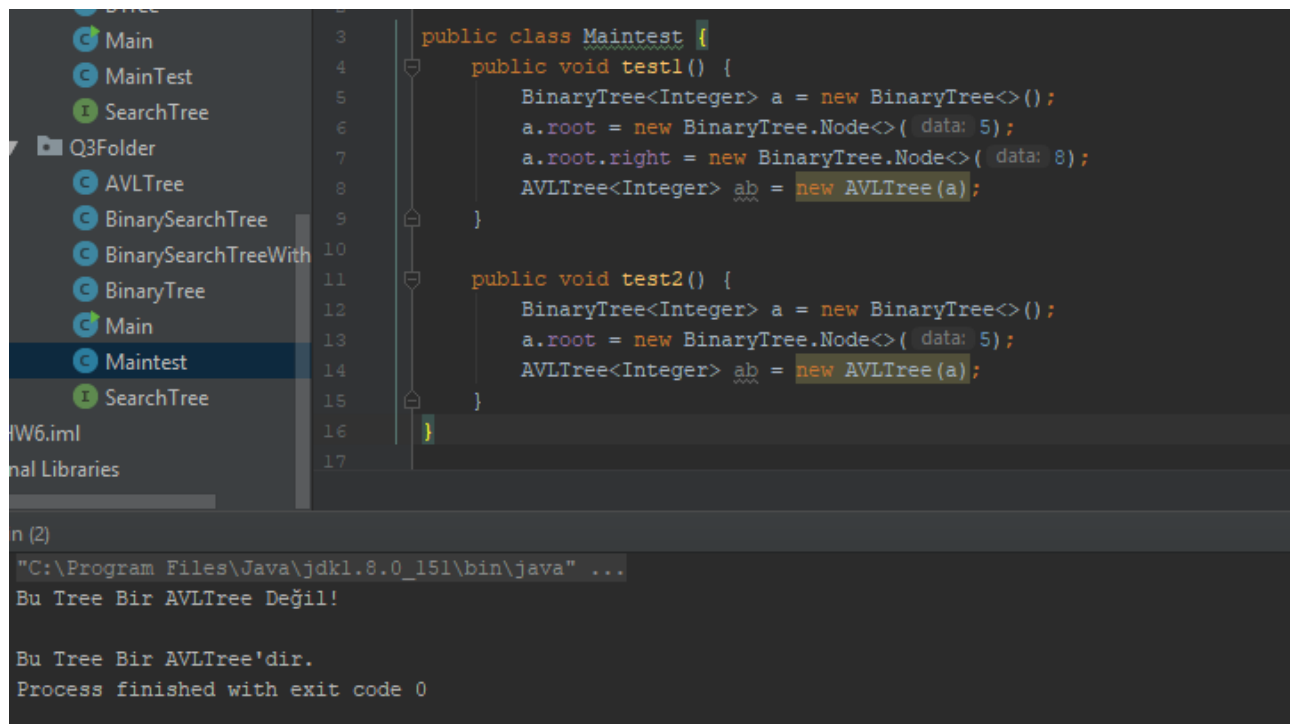
TEST 1

Constructor için ilk başta root'a 5, sağ child'ına 8 ekledim.

TEST 2

Constructor için root'a 5 ekledim.

3.3 Running Commands and Results



The screenshot shows an IDE with a project explorer on the left, a code editor in the center, and a console at the bottom. The project explorer lists several classes: BinaryTree, Main, MainTest, SearchTree, Q3Folder, AVLTree, BinarySearchTree, BinarySearchTreeWith, BinaryTree, Main, Maintest, and SearchTree. The code editor displays the following Java code:

```
3 public class Maintest {
4     public void test1() {
5         BinaryTree<Integer> a = new BinaryTree<>();
6         a.root = new BinaryTree.Node<> ( data: 5);
7         a.root.right = new BinaryTree.Node<> ( data: 8);
8         AVLTree<Integer> ab = new AVLTree(a);
9     }
10
11     public void test2() {
12         BinaryTree<Integer> a = new BinaryTree<>();
13         a.root = new BinaryTree.Node<> ( data: 5);
14         AVLTree<Integer> ab = new AVLTree(a);
15     }
16 }
17
```

The console at the bottom shows the following output:

```
n (2)
"C:\Program Files\Java\jdk1.8.0_151\bin\java" ...
Bu Tree Bir AVLTree Değil!

Bu Tree Bir AVLTree'dir.
Process finished with exit code 0
```