# Homework 06-Rapor

## PART I:

Part1 de öncelikle bir tane wrapper metot yaptım public String encode(String message) isimli.Daha sonra bu wrapper metot içerisinde bir for döngüsü yardımıyla parametre olarak gelen mesajı tek tek harflerine ayırarak asıl işi yapmayı sağlayan private boolean encode(char ch,StringBuilder result,BinaryTree<HuffData> currentTree) metodunu çağırıyorum.Parametre olarak aldığım StringBuilder ı 0-1 lerle doldurup gelen mesajın kodu olarak return ettiriyorum.Aynı zamanda parametre olarak gelen currentTree belli bir düzene göre oluşturulmuş ve bu ağaçta,gelen karakterlerimi tek tek aratıyorum.

Base case olarak currentTree.isLeaf() ile ağacımın kökü null mı veya sağ ve sol çocuğu var mı diye bakıyorum.Eğer sağ ve sol çocuğu yoksa en sona gelmişim demektir.Yani artık çocuğu yok onun yerine orada data sı var.Bu durumda parametre olarak gelen karakterle karşılaştırma yapıyorum.Aynı karakterse true (buldum diye) return ediyorum.

Eğer base case'e girmezse if ile parametre olarak gelen ağacın datasının sembolüne bakıyorum null sa agac devam ediyor anlamına geliyor.Bu durumda recursive olarak metodumu tekrar çağırıyorum parametrelerin hepsi aynı sadece currentTree yerine currentTree. getLeftSubtree() olarak ağacı küçültüp öyle çağırıyorum.Bu metodun return değerine bakiyorum eğer true return ediyorsa sol taraftan geldiğini bildiğim için recursion'lar bittikten sonra 0 ekliyorum.Eğer false return ederse solda bulamadığını anlayıp recursive olarak metodumu tekrar çağırıyorum parametrelerin hepsi aynı sadece currentTree yerine currentTree. getRightSubtree() olarak ağacı küçültüp öyle çağırıyorum.Aynı şekilde bununda return değerine bakıyorum eğer true return ediyorsa sağ taraftan geldiğini bildiğim için recursion'lar bittikten sonra 1 ekliyorum.Eğer false return de artık if ten çıkıyorum.Zaten bu noktaya kadar return değerlerim false sa bulamamıştır deyip return false yapıyorum.

Son olarak tekrar wrapper metoda geliyor.Ancak kodlar recursion dan dolayı ters bir şekilde geldiği için ekleme yaptığım StringBuilder ı bir for içinde sondan başlatıp farklı bir StringBuilder a append edip return ediyorum.

## PART II:

Öncelikle BinarySearchTreeIterator<E> implements Iterator<E> sınıfı oluşturuyorum.Iterator implemets ettiğimin içinde next() ve hasNext() metotlarını override etmem gerektiği için içlerini dolduruyorum.

Yazdığım sınıfta 2 tane data tanımladım.Bunlar boş bir stack ve ağacın root'unu değiştirmek istemediğimden dolayı bir tanede Node tipinde temp_root tuttum.Bu temp_root a başta ağacın asıl root'unu atadım.

Constructor'ı içinde Stack i olusturup temp_root null olmadığı sürece stack'e push() ettim ve temp_root u temp_root.left olarak değiştirdim. hasNext() metodunda ise return stack.isEmpty() return ettim.

next() metodunda ise hasNext() ile stack 'in dolu olduğundan emin olup doluyken stack'in en üstteki elemanını pop()'la çekip return için res değişkeninde tutuyorum,bu pop() la çektiği yeri'de(adresi) bir Node da tutuyorum.Eğer bu Node(adres) un sağ taraftaki çocuğu null değilse stack'e sağ taraftaki çocuğu da push() la ekliyorum.Ve çocukta kendisinin sol çocuğunu gösteriyor artik.En sondada tuttuğum res değişkenini return ediyorum.

main'de kendi yaptığım inner classı iterator olarak kullanmak içinde
public Iterator<E> iter(){return new BinarySearchTreeIterator<>();}
ile oluşturuyorum.Test için ise :

```
Iterator iterator=Btree.iter();
while(iterator.hasNext())
        System.out.println(iterator.next()); ile test ediyorum.
```

Her iki part içinde JUnitTedtler yapılmıştır.

**********

https://en.wikipedia.org/wiki/Tree_traversal#In-order_2

3 Implementations
- 3.1Depth-first search
  - 3.1.2In-order

**NOT**➔Algoritma için bu kısımlardan yararlandım.

---

**Muhammet Tayyip Çankaya**
**131044054**