İkili Arama Ağaçları: Preorder, Inorder, Postorder Dolaşımları

İkili arama ağacını oluşturmak için gerekli düğüm yapısı ve işaretçi tanımlanmıştır. Her düğümün bir içeriği, sağ ve solundaki çocuklarını gösteren sag ve sol işaretçileri vardır. Ayrıca ağacın kök elemanını gösteren kok işaretçisi bulunmaktadır.

```
class Dugum {
    int icerik;
    Dugum sag;
    Dugum sol;
    Dugum(int icerik){
        this.icerik=icerik;
        sag=null;
        sol=null;
    }
}
    public class İkiliAramaAgaci {
    Dugum kok = null;
```

Ağacın preorder dolaşımı(Kök-Sol-Sağ) yinelemesiz olarak gerçekleştirilmiştir. Dolaşım için yığıt kullanılmıştır. İlk olarak boş bir yığıt oluşturulmuştur. Bir d düğümü oluşturularak kök düğümü ile başlatılır. d düğümü yığıta eklenir. Yığıt boşalana kadar çalışan bir döngü oluşturulur. Bu döngü içerisinde yığıttan eleman çıkarılır, yazdırılır, varsa önce sağ çocuğu sonra sol çocuğu yığıta eklenir. Sağ çocuğun sol çocuktan önce yığıta eklenmesi ile sol alt ağacın daha önce dolaşılması sağlanır. Bu işlemler döngü içerisinde tekrarlanır ve yığıt boşaldığında dolaşım tamamlanır.

```
    public void Gezinti_preOrder(){
    Dugum d = kok;
    Stack<Dugum> s= new Stack<Dugum>();
    if(d == null)
        return;
    s.push(d);
    while(!s.isEmpty()){
        d = s.pop();
        System.out.print(d.icerik+" ");
        if(d.sag!=null){
            s.push(d.sag);
        }
        if(d.sol!=null){
            s.push(d.sol);
        }
    }
        System.out.println("");
}
```

Ağacın inorder dolaşımı(Sol-Kök-Sağ) yinelemesiz olarak gerçekleştirilmiştir. Dolaşım için yığıt kullanılmıştır. İlk olarak boş bir yığıt oluşturulmuştur. Bir d düğümü oluşturularak kök düğümü ile başlatılır. d düğümü yığıta eklenir ve sol düğümü gösterecek şekilde d düğümü değiştirilir. d null ve yığıt boş değilse çalışan bir döngü oluşturulur. Bu döngü içerisinde yığıtın en üstündeki eleman çıkarılır, yazdırılır, d düğümü sağındaki eleman olacak şekilde değiştirilir. Tekrar yığıta ekleme işlemine geri dönülür ve aynı işlemler tekrarlanır. Eğer d düğümü null ve yığıt boş ise dolaşım tamamlanır.

```
    public void Gezinti_inOrder(){
    Dugum d = kok;
    Stack<Dugum> s= new Stack<Dugum>();
    if(d == null)
        return;
    while(d!=null || !s.isEmpty()){
        while(d!=null){
        s.push(d);
        d = d.sol;
    }
        d = s.pop();
        System.out.print(d.icerik+" ");
        d = d.sag;
    }
        System.out.println("");
}
```

Ağacın postorder dolaşımı(Sol-Sağ-Kök) yinelemesiz olarak gerçekleştirilmiştir. Dolaşım için iki tane yığıt kullanılmıştır. Birinci yığıt elemanları düğümlerin doğru sırasını elde etmek için, ikinci yığıt ise postorder dolaşımın sırasını elde etmek için kullanılır. İlk olarak boş iki yığıt oluşturulmuştur. Bir d düğümü oluşturularak kök düğümü ile başlatılır. Birinci yığıt boşalana kadar çalışan bir döngü oluşturulur. Bu döngü içerisinde birinci yığıttan bir düğüm çıkarılır ve ikinci yığıta eklenir. Bu düğümün varsa önce sol düğümü sonra sağ düğümü birinci yığıta eklenir. Bu işlemlerden sonra birinci yığıt boşaldığında dolaşım tamamlanmış olur ve ikinci yığıtta postorder gezinti sırası elde edilmiş olur. İkinci yığıttan elemanlar tek tek çıkarılıp yazdırılarak postorder dolaşımı tamamlanır.

```java
public void Gezinti_postOrder(){
Dugum d = kok;
Stack<Dugum> s1= new Stack<Dugum>();
Stack<Dugum> s2= new Stack<Dugum>();
if(d == null)
    return;
s1.push(d);
while(!s1.isEmpty()){
    d = s1.pop();
    s2.push(d);
    if(d.sol!=null)
        s1.push(d.sol);
    if(d.sag!=null)
        s1.push(d.sag);
}
while(!s2.isEmpty()){
    System.out.print(s2.pop().icerik+" ");
}
        System.out.println("");
    }
```