KOD

```
class AvlDugum {
    int icerik, yukseklik;
    AvlDugum sol, sag;

    AvlDugum(int icerik) {
        this.icerik = icerik;
        yukseklik = 1;
    }
}

class AVLTree {

    AvlDugum kok;
    //agacın yüksekligini döndürür
    int yukseklikBul(AvlDugum N) {
        if (N == null)
            return 0;

        return N.yukseklik;
    }


    int max(int a, int b) {
        return (a > b) ? a : b;
    }


    AvlDugum sagaDondur(AvlDugum y) {
        AvlDugum x = y.sol;
        AvlDugum T2 = x.sag;


        x.sag = y;
        y.sol = T2;


        y.yukseklik = max(yukseklikBul(y.sol), yukseklikBul(y.sag)) + 1;
        x.yukseklik = max(yukseklikBul(x.sol), yukseklikBul(x.sag)) + 1;


        return x;
    }


    AvlDugum solaDondur(AvlDugum x) {
        AvlDugum y = x.sag;
        AvlDugum T2 = y.sol;

        y.sol = x;
        x.sag = T2;

        x.yukseklik = max(yukseklikBul(x.sol), yukseklikBul(x.sag)) + 1;
        y.yukseklik = max(yukseklikBul(y.sol), yukseklikBul(y.sag)) + 1;
```

```
        return y;
    }


    int getBalance(AvlDugum N) {
        if (N == null)
            return 0;

        return yukseklikBul(N.sol) - yukseklikBul(N.sag);
    }

    AvlDugum ekle(AvlDugum node, int icerik) {


        if (node == null)
            return (new AvlDugum(icerik));

        if (icerik < node.icerik)
            node.sol = ekle(node.sol, icerik);
        else if (icerik > node.icerik)
            node.sag = ekle(node.sag, icerik);
        else
            return node;


        node.yukseklik = 1 + max(yukseklikBul(node.sol),
                yukseklikBul(node.sag));


        int balance = getBalance(node);


        if (balance > 1 && icerik < node.sol.icerik) {
            System.out.println("tek rotasyon gerekti (sol sol durumu)");
            return sagaDondur(node);
        }

        if (balance < -1 && icerik > node.sag.icerik) {
            System.out.println("tek rotasyon gerekti (sağ sağ durumu)");
            return solaDondur(node);
        }

        if (balance > 1 && icerik > node.sol.icerik) {
            System.out.println("cift rotasyon gerekti (sol sağ durumu)");
            node.sol = solaDondur(node.sol);
            return sagaDondur(node);
        }

        if (balance < -1 && icerik < node.sag.icerik) {
            System.out.println("çift rotasyon gerekti (sağ sol durumu)");
            node.sag = sagaDondur(node.sag);
            return solaDondur(node);
        }


        return node;
    }
```

```
    void preOrder(AvlDugum node) {
        if (node != null) {
            System.out.print(node.icerik + " ");
            preOrder(node.sol);
            preOrder(node.sag);
        }
    }

    public static void main(String[] args) {
        AVLTree tree = new AVLTree();
        tree.kok = tree.ekle(tree.kok, 1881);
        tree.kok = tree.ekle(tree.kok, 1905);
        tree.kok = tree.ekle(tree.kok, 1920);
        tree.kok = tree.ekle(tree.kok, 1923);
        tree.kok = tree.ekle(tree.kok, 1938);
        tree.kok = tree.ekle(tree.kok, 1915);

        System.out.println("Ağacın elemanları");
        tree.preOrder(tree.kok);
    }
}
```

EKRAN ÇIKTISI

```
"C:\Program Files\Java\jdk-13.0.2\bin\java.exe" "-j
tek rotasyon gerekti (sağ sağ durumu)
tek rotasyon gerekti (sağ sağ durumu)
çift rotasyon gerekti (sağ sol durumu)
Ağacın elemanları
1920 1905 1881 1915 1923 1938
Process finished with exit code 0
```