```
//////////////////////////////////////////////////////////////////////
//                                        Tietorakenteet  K2001
//                                        Arto Wikla 9.2.2001
//                                        Muokannut Atte Tanskanen 2004
// Luokka public class AVLPuu
//
// AVL-hakupuun toteutus: alkiot Solmu-luokan ilmentymiä
//
// konstruktori: public AVLPuu()
//
// aksessorit:
//
//   public Solmu find(Solmu alkio) palauttaa viitteen löydettyyn
//                                  alkioon; null jos alkiota ei löydy
//
//   public void insert(Solmu alkio) lisää alkion puuhun sopivaan paikkaan,
//                                   so. puu säilyy hakupuuna; jos arvo jo
//                                   on puussa, mikään ei muutu
//
//   public Solmu remove(Solmu alkio) poistaa alkion puusta, so. puu
//                                    säilyy hakupuuna; jos arvoa ei
//                                    puussa, mikään ei muutu
//
//   public String toString() tuottaa puun merkkiesityksen
//                                  "hakemistorakenteena"
//
//////////////////////////////////////////////////////////////////////

public class AVLPuu  {

  public class AVLSolmu {  // salainen sisäluokka
    private Solmu alkio;
    private AVLSolmu vasen, oikea;
    private int korkeus;

    private AVLSolmu(Solmu alkio, AVLSolmu vasen, AVLSolmu oikea) {
        this.alkio = alkio;
        this.vasen = vasen;
        this.oikea = oikea;
        korkeus = 0;
    }
  }

  private AVLSolmu juuri;

  public AVLPuu() {
    juuri = null;
  }


  // -- insert apumetodeineen --

  public void insert(Solmu alkio) {
    juuri = insert(juuri, alkio);
  }

  private AVLSolmu insert(AVLSolmu t, Solmu alkio) {
    if (t == null)
      t = new AVLSolmu(alkio, null, null);

    else if (alkio.compareTo(t.alkio.getObject()) < 0) { // vasemmalle
      t.vasen = insert(t.vasen, alkio);

      if(korkeus(t.vasen) - korkeus(t.oikea) == 2 )
        if(alkio.compareTo(t.vasen.alkio.getObject()) < 0)
          t = kiertoOikealle(t);
      else
```

```
        t = kaksoisKiertoOikealle(t);
    }
    else if (alkio.compareTo(t.alkio.getObject()) > 0) { // oikealle
      t.oikea = insert(t.oikea, alkio);

      if(korkeus(t.oikea) - korkeus(t.vasen) == 2 )
        if(alkio.compareTo(t.oikea.alkio.getObject()) > 0)
          t = kiertoVasemmalle(t);
      else
        t = kaksoisKiertoVasemmalle(t);
    }
    else  // on jo
      ;

    t.korkeus = maksimi(korkeus(t.vasen), korkeus(t.oikea)) +1;
    return t;
  }


  // -- remove apumetodeineen --

  public Solmu remove(Solmu alkio) {
    Solmu poistettava = find(juuri, alkio);
    if (poistettava != null) {
      juuri = remove(juuri, alkio);
      return poistettava;
    }
    else
      return null;
  }

  private AVLSolmu remove(AVLSolmu t, Solmu alkio) {
    if (t == null)
      return t;

    // vasemmalle
    if (alkio.compareTo(t.alkio.getObject()) < 0) {
      t.vasen = remove(t.vasen, alkio);

      // tasapainoitetaan poiston jälkeen
      if (korkeus(t.oikea) - korkeus(t.vasen) == 2)
        t = kiertoVasemmalle(t);
      if (korkeus(t.vasen) - korkeus(t.oikea) == 2)
        t = kaksoisKiertoOikealle(t);
    } // oikealle
    else if (alkio.compareTo(t.alkio.getObject()) > 0) {
      t.oikea = remove(t.oikea, alkio);

      // tasapainoitetaan poiston jälkeen
      if (korkeus(t.vasen) - korkeus(t.oikea) == 2)
        t = kiertoOikealle(t);
      if (korkeus(t.oikea) - korkeus(t.vasen) == 2)
        t = kaksoisKiertoVasemmalle(t);
    }
    else { // tässä tapahtuu varsinainen poisto

      // poistettavan molemmat lapset olemassa
      if (t.vasen != null && t.oikea != null) {
        // siirretään poistettavan alkion oikean alipuun pienimmän
        // alkion tiedot poistettavaan solmuun
        t.alkio = findMin(t.oikea).alkio;
        // poistetaan sitten samainen pienin alkio oikeasta alipuusta
        // näin saimme siirrettyä poistettavan tilalle
        // oikean alipuunsa pienimmän alkion
        t.oikea = remove(t.oikea, t.alkio);

        // tasapainoitetaan poiston jälkeen
```

```
        // jos solmun vasen alipuu liian korkea tehdään kiertoOikealle
        if (korkeus(t.vasen) - korkeus(t.oikea) == 2)
          t = kiertoOikealle(t);
        // jos kierron jälkeen oikea alipuu liian korkea, "siksak" tilanne
        // tehdään kaksoisKiertoVasemmalle
        if (korkeus(t.oikea) - korkeus(t.vasen) == 2)
          t = kaksoisKiertoVasemmalle(t);

      }
      else if (t.vasen != null) {
        // poistettavalla vain vasen alipuu
        // korvataan poistettava sillä
        t = t.vasen;
      }
      else {
        // poistettavalla vain oikea alipuu
        // korvataan poistettava sillä
        t = t.oikea;
      }
    }

    // päivitetään solmujen korkeus-kentät
    if (t != null)
      t.korkeus = maksimi(korkeus(t.vasen), korkeus(t.oikea)) + 1;
    return t;
  }



  private static int korkeus(AVLSolmu t) {
    return (t == null) ? -1 : t.korkeus;
  }

  private static int maksimi(int eka, int toka) {
    return (eka > toka) ? eka : toka;
  }


  private static AVLSolmu kiertoOikealle(AVLSolmu k2) {
    AVLSolmu k1 = k2.vasen;
    k2.vasen = k1.oikea;
    k1.oikea = k2;
    k2.korkeus = maksimi(korkeus(k2.vasen), korkeus(k2.oikea)) + 1;
    k1.korkeus = maksimi(korkeus(k1.vasen), k2.korkeus) + 1;
    return k1;
  }

  private static AVLSolmu kiertoVasemmalle(AVLSolmu k1) {
    AVLSolmu k2 = k1.oikea;
    k1.oikea = k2.vasen;
    k2.vasen = k1;
    k1.korkeus = maksimi(korkeus(k1.oikea), korkeus(k1.vasen)) + 1;
    k2.korkeus = maksimi(korkeus(k2.oikea), k1.korkeus) + 1;
    return k2;
  }

  private static AVLSolmu kaksoisKiertoOikealle(AVLSolmu k3) {
    k3.vasen = kiertoVasemmalle(k3.vasen);
    return kiertoOikealle(k3);
  }

  private static AVLSolmu kaksoisKiertoVasemmalle(AVLSolmu k3) {
    k3.oikea = kiertoOikealle(k3.oikea);
    return kiertoVasemmalle(k3);
  }
```

```
  // -- find apumetodeineen --

  public Solmu find(Solmu alkio) {
    return find(juuri, alkio);
  }

  private Solmu find(AVLSolmu t, Solmu alkio){

    if (t == null)
      return null;

    int erotus = alkio.compareTo(t.alkio.getObject());

    if (erotus < 0)  // Oikealle
      return find(t.vasen, alkio);
    if (erotus > 0)  // Vasemmalle
      return find(t.oikea, alkio);
    return t.alkio;  // löytyi
  }


  // -- findMin luokan sisäiseen käyttöön --

  private static AVLSolmu findMin(AVLSolmu t) {

    if (t == null)
      return null;

    while (t.vasen != null)
      t = t.vasen;
      return t;
  }


  // -- toString() apumetodeineen --

  public String toString() {
    return toString(juuri, "");
  }

  private static String toString(AVLSolmu juuri, String sisennys) {

    if (juuri==null)
      return "";
    else return
      toString(juuri.oikea, sisennys + "      |") +
      "\n" + sisennys + "--->" + juuri.alkio +
      toString(juuri.vasen, sisennys + "      |");

  }

}
```