```
//////////////////////////////////////////////////////////////////////
//                                  Tietorakenteet K2000
//                                  Arto Wikla 5.2.2000
//                                  Muokannut Atte Tanskanen 2004
//
//Luokka public class Jono
//
// Jonon toteutus kaksisuuntaisena linkitettynä rengaslistana.
// Alkiot ovat tyyppiä Tiedosto.
//
// Vain viimeiseen solmuun ylläpidetään linkkiä (perä).
// Perän seuraaja on keula.
//
// Toteutuksessa solmut ovat luokan Solmu ilmentymiä.
//
// konstruktori:
//            public Jono() luo tyhjän jonon
//
// aksessorit:
//
//   public Solmu toQueue(Tiedosto alkio)
//                    laittaa "alkio"n jonoon; null ei kelpaa alkioksi;
//                    palauttaa viitteen uuteen alkioon
//
//   public Solmu fromQueue()
//                    palauttaa arvonaan jonon ensimmäisen alkion ja
//                    poistaa sen jonosta; jos jono on tyhjä,
//                    metodi palauttaa arvon null
//
//   public Solmu moveToEnd(Solmu siirrettävä)
//                    Siirtää "siirrettävä"n solmun jono perälle
//                    palauttaa viitteen siirrettyyn solmuun
//
//   public Solmu remove(Solmu poistettava)
//                    Poistaa "poistettavan"n solmun jonosta
//                    palauttaa viitteen poistettuun solmuun
//
//   public String toString()
//                    muodossa: keula: ( [ tiedosto1 ] [ tiedosto2 ] ) :perä
//
//////////////////////////////////////////////////////////////////////


public class Jono {

  private Solmu perä;   // perä.linkki on keula!
  private int lkm;

  public Jono() {
    perä = null;   // tyhjä jono
    lkm = 0;
  }

  public Solmu toQueue(Tiedosto alkio) {

    if (alkio == null) // null ei kelpaa
      return null;    // selvän teki

    if (perä == null) {
      perä = new Solmu();    // 1. solmu
      perä.tieto  = alkio;
      perä.edellinen = perä;    // linkitetään itseensä
      perä.seuraava = perä;    // linkitetään itseensä
      lkm = 1;
      return perä;           // valmista tuli
    }
```

```
    Solmu vanhaPerä = perä;
    perä = new Solmu();
    perä.tieto  = alkio;
    perä.seuraava = vanhaPerä.seuraava; // uusi perä osoittamaan keulaan
    perä.edellinen = vanhaPerä;        // uusi perä osoittamaan edelliseen
    vanhaPerä.seuraava.edellinen = perä;  // keula osoittamaan uuteen perään
    vanhaPerä.seuraava = perä;  // ja toiseksi viimeinen osoittamaan
    ++lkm;                     // uuteen viimeiseen

    return perä;
  }


  // poistetaan jonon keulilta eli poistetaan
  // Least Recently Used -tiedosto välimuistirakenteesta

  public Solmu fromQueue() {

    if (perä == null) // tyhjä
      return null;    // valmista tuli

    Solmu keula = perä.seuraava;

    if (perä == keula)    // 1 alkion jono: viimeinenkin hävitetään
      perä = null;
    else {                // pidempi jono
      perä.seuraava = keula.seuraava;  // perä osoittamaan uuteen keulaan
      keula.seuraava.edellinen = perä; // uusi keula osoittamaan perään
    }

    --lkm;
    return keula;
  }


  // siirretään solmu jonon perälle haettaessa
  // välimuistissa jo olevaa tiedostoa

  public Solmu moveToEnd(Solmu siirrettävä) {
    if (siirrettävä == null) // null ei käy
      return null;

    if (siirrettävä == perä) // siirrettävä on jo viimeinen
      return perä;


    // päivitellään siirrettävän solmun molemmilla
    // puolilla olevien ja perän ja keulan linkkejä

    siirrettävä.edellinen.seuraava = siirrettävä.seuraava;
    siirrettävä.seuraava.edellinen = siirrettävä.edellinen;
    siirrettävä.seuraava = perä.seuraava;
    siirrettävä.edellinen = perä;
    perä.seuraava.edellinen = siirrettävä;
    perä.seuraava = siirrettävä;
    perä = siirrettävä;

    return perä;
  }


  // poistetaan solmu kokonaan jonosta, jos levyllä oleva
  // päivitetty tiedosto on liian suuri välimuistiin

  public Solmu remove(Solmu poistettava) {
    if (poistettava == null) // null ei kelpaa
      return null;
```

```
    if (perä == perä.seuraava) // vain yksi solmu jonossa
      perä = null;
    else {
      if (poistettava == perä) // jos poistettava on perä
        perä = poistettava.edellinen;

      // päivitellään solmujen linkit
      poistettava.seuraava.edellinen = poistettava.edellinen;
      poistettava.edellinen.seuraava = poistettava.seuraava;
    }

    return poistettava; // palautetaan linkki poistettuun solmuun
  }


  // jonon merkkiesitys

  public String toString() {

    String mjono=" keula (vanhin): ( ";

    if (perä != null) {

      Solmu p = perä;
      do {
        mjono += p.seuraava.tieto + " ";
        p = p.seuraava;
      } while (p != perä);
    }
    return mjono+") :perä (uusin)";
  }
}
```