class FIFOListe<T> implements Iterable<T> {

  private Node hode;

  private Node hale;

  private class Node {

    Node neste;

    T ting;

    Node(T ting) {

      this.ting = ting;

    }

  }

  public int storrelse() {

    int teller = 0;

    for (Node tmp = hode; tmp != null; tmp = tmp.neste) {

      teller++;

    }

    return teller;

  }

  public boolean erTom() {

    return hode == null;

  }

  public void leggTil(T t) {

    Node nyNode = new Node(t);

    if (erTom()) {

      hode = nyNode;

    }

    else {

      hale.neste = nyNode;

    }

    hale = nyNode;

  }

  public void fjernAlt() {

    hode = null;

    hale = null;

  }

  public T pop() {

    Node returnode = hode;

    hode = hode.neste;

    return returnode.ting;

  }

  public void skrivUtListe() {

    Node tmp = hode;

    System.out.println("[");

    while (tmp != null) {

      System.out.println("\t" + tmp.ting);

      tmp = tmp.neste;

    }

    System.out.println("]");

  }

  public Iterator<T> iterator() {

    return new FIFOIterator();

  }

  private class FIFOIterator implements Iterator<T> {

    Node denne = hode;

    @Override

    public boolean hasNext() {

      return denne != null;

    }

    @Override

    public T next() {

      if (denne == null) throw new NoSuchElementException("next");

      Node tmp = denne;

      denne = denne.neste;

      return tmp.ting;

    }

  }

}