```
/*
 * Soluzione dell'esercizio 1
 */
class CodaCircolare {
  private int head;
  private int tail;
  private int[] coda;
  private static final int MAXELEMENTI = 10;

  public CodaCircolare() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public static boolean isEmpty(CodaCircolare c) {
    return c.tail == c.head; //.............................aggiunto
  }

  public static boolean isFull(CodaCircolare c) {
    return (c.tail+1)%c.MAXELEMENTI == c.head ; //...........aggiunto
  }

  public static void inserisci(CodaCircolare c, int elemento) {
    if (!isFull(c)) {
      c.coda [c.tail] = elemento;
      c.tail = (c.tail+1) % c.MAXELEMENTI;
      System.out.println("elemento inserito");
    } else {
      System.out.println("coda piena");
    }
  }

  public static int estrai(CodaCircolare c) {
    if (!isEmpty(c)) {
      int elemento = c.coda[c.head];
      c.head = (c.head+1) % c.MAXELEMENTI;
      return elemento;
    } else {
      System.out.println("coda vuota");
      return 0;
    }
  }
}
```

```
/*
 * Soluzione dell'esercizio 2
 */
class UsaCodaCircolare {
  public static void main(String[] args) {
    CodaCircolare c = new CodaCircolare();
    while (!CodaCircolare.isFull(c)) {
      System.out.print("Inserisci un elemento nella coda: ");
      int e = Leggi.unInt();
      CodaCircolare.inserisci(c,e);
    }
    /*
     * Il codice che segue non fa parte dell'esercizio, ma
     * serve solo pre verificare ciò che è stato fatto.
     */
    int contatore = 1;
    while (!CodaCircolare.isEmpty(c)) {
      System.out.print(contatore+")");
      contatore++;
      System.out.println(CodaCircolare.estrai(c));
    }
  }
}
```

```
/*
 * Soluzione dell'esercizio 3
 */
class UsaCodaCircolare2 {
  public static void main(String[] args) {
    CodaCircolare c = new CodaCircolare(); //.............punto 1
    while (!CodaCircolare.isFull(c)) { //.................inizio punto 2
      System.out.print("Inserisci un elemento nella coda: ");
      // la riga qui sotto è superflua:
      //   int e = Leggi.unInt();
      // basta mettere direttamente Leggi.unInt() qui sotto
      CodaCircolare.inserisci(c,Leggi.unInt());
    } //...................................................fine punto 2
    CodaCircolare d = new CodaCircolare(); //.............punto 3
    for (int i = 1; i <= 3; i++) { //.....................inizio punto 4
      int e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(d,e);
    }
    /* Si puo' fare anche (meglio):
      for (int i = 1; i <= 3; i++)
          CodaCircolare.inserisci(d,CodaCircolare.estrai(c));
    */ //...................................................fine punto 4
  }
}
```

```
/*
 * Soluzione esercizio 4
 * (prima versione)
 */
class UsaCodaCircolare2 {
  public static void main(String[] args) {
    CodaCircolare c = new CodaCircolare();
    while (!CodaCircolare.isFull(c)) {
      System.out.print("Inserisci un elemento nella coda: ");
      CodaCircolare.inserisci(c,Leggi.unInt());
    }
    CodaCircolare d = new CodaCircolare();
    for (int i = 1; i <= 3; i++)
      CodaCircolare.inserisci(d,CodaCircolare.estrai(c));
    stampaCodaCircolare(c);
    stampaCodaCircolare(d);
  }

  static void stampaCodaCircolare(CodaCircolare c) {
    System.out.print("|");
    if (!CodaCircolare.isEmpty(c))
      System.out.print(CodaCircolare.estrai(c));
    while (!CodaCircolare.isEmpty(c))
      System.out.print(","+CodaCircolare.estrai(c));
    System.out.println("|");
  }
}
```

```
/*
 * Soluzione esercizio 4
 * (seconda versione)
 */
class UsaCodaCircolare2 {
  public static void main(String[] args) {
    CodaCircolare c = new CodaCircolare();
    while (!CodaCircolare.isFull(c)) {
      System.out.print("Inserisci un elemento nella coda: ");
      CodaCircolare.inserisci(c,Leggi.unInt());
    }
    CodaCircolare d = new CodaCircolare();
    for (int i = 1; i <= 3; i++)
      CodaCircolare.inserisci(d,CodaCircolare.estrai(c));
    stampaCodaCircolare(c);
    stampaCodaCircolare(d);
  }

  /* L'implementazione precedente aveva il grave effetto collaterale
   * di svuotare la coda durante la stampa.
   * Se il metodo si chiamasse stampaESvuotaCodaCircolare, non ci
   * sarebbe niente di male, ma visto che si chiama stampaCodaCircolare,
   * chi lo usa si aspetera' che la coda rimanga invariata.
   * L'implementazione deve essere coerente con questa aspettativa.
   */
  static void stampaCodaCircolare(CodaCircolare c) {
    CodaCircolare temp = new CodaCircolare();
    int e;
    String temp2 = "|";
    if (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += e;
    }
    while (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += "."+e;
    }
    temp2 += "]";
    System.out.println(temp2);
    c = temp;
  }
}
```

```
/*
 * Soluzione esercizio 5
 */
class CodaCircolare {
  private int head;
  private int tail;
  private int[] coda;
  private int MAXELEMENTI = 10; //.............................modificato

  public CodaCircolare() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num) {
    MAXELEMENTI = num; // qua si potrebbe mettere anche num+1
                       // dipende dalla semantica che si vuole
                       // dare a questo costruttore
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public static boolean isEmpty(CodaCircolare c) {
    return c.tail == c.head;
  }

  public static boolean isFull(CodaCircolare c) {
    return (c.tail+1)%c.MAXELEMENTI == c.head ;
  }

  public static void inserisci(CodaCircolare c, int elemento) {
    if (!isFull(c)) {
      c.coda [c.tail] = elemento;
      c.tail = (c.tail+1) % c.MAXELEMENTI;
      System.out.println("elemento inserito");
    } else {
      System.out.println("coda piena");
    }
  }

  public static int estrai(CodaCircolare c) {
    if (!isEmpty(c)) {
      int elemento = c.coda[c.head];
      c.head = (c.head+1) % c.MAXELEMENTI;
      return elemento;
    } else {
      System.out.println("coda vuota");
      return 0;
    }
  }
}
```

```
/*
 * Soluzione esercizio 6
 */
class CodaCircolare {
  private int head;
  private int tail;
  private int[] coda;
  private int MAXELEMENTI = 10;

  public CodaCircolare() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num, int[] elem) { //..............aggiunto
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    int i;
    for (i = 0; i < elem.length && i < MAXELEMENTI-1; i++)
      coda[i] = elem[i];
    tail = i;
  }

  public static boolean isEmpty(CodaCircolare c) {
    return c.tail == c.head;
  }

  public static boolean isFull(CodaCircolare c) {
    return (c.tail+1)%c.MAXELEMENTI == c.head ;
  }

  public static void inserisci(CodaCircolare c, int elemento) {
    if (!isFull(c)) {
      c.coda [c.tail] = elemento;
      c.tail = (c.tail+1) % c.MAXELEMENTI;
      System.out.println("elemento inserito");
    } else {
      System.out.println("coda piena");
    }
  }

  public static int estrai(CodaCircolare c) {
    if (!isEmpty(c)) {
      int elemento = c.coda[c.head];
      c.head = (c.head+1) % c.MAXELEMENTI;
      return elemento;
    } else {
      System.out.println("coda vuota");
      return 0;
    }
  }
}
```

```
/*
 * Soluzione esercizio 7
 */
class UsaCodaCircolare2 {
  public static void main(String[] args) {
    System.out.print("Inserisci la dimensione della coda: ");
    int n = Leggi.unInt();
    CodaCircolare c = new CodaCircolare(n); //..............punto 1
    int[] temp = new int[n/2]; // si puo' scegliere anche n/2-1
    int j = 0;
    int e;
    while (!CodaCircolare.isFull(c)) { //...................punto 2
      System.out.print("Inserisci un elemento nella coda: ");
      e = Leggi.unInt();
      CodaCircolare.inserisci(c,e);
      if (j < temp.length) {
        temp[j] = e;
        j++;
      }
    }
    CodaCircolare d = new CodaCircolare(n*2,temp); //........punto 3
    stampaCodaCircolare(c); //..............................punto 4
    stampaCodaCircolare(d);
  }

  static void stampaCodaCircolare(CodaCircolare c) {
    CodaCircolare temp = new CodaCircolare();
    int e;
    String temp2 = "|";
    if (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += e;
    }
    while (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += ","+e;
    }
    temp2 += "|";
    System.out.println(temp2);
    c = temp;
  }
}
```

```
/*
 * Soluzione esercizio 8
 */
class CodaCircolare {
  private int head;
  private int tail;
  private int[] coda;
  private int MAXELEMENTI = 10;

  public CodaCircolare() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num, int[] elem) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    int i;
    for (i = 0; i < elem.length && i < MAXELEMENTI-1; i++)
      coda[i] = elem[i];
    tail = i;
  }

  public static boolean isEmpty(CodaCircolare c) {
    return c.tail == c.head;
  }

  public static boolean isFull(CodaCircolare c) {
    return (c.tail+1)%c.MAXELEMENTI == c.head ;
  }

  public static void inserisci(CodaCircolare c, int elemento) {
    if (!isFull(c)) {
      c.coda [c.tail] = elemento;
      c.tail = (c.tail+1) % c.MAXELEMENTI;
      System.out.println("elemento inserito");
    } else {
      System.out.println("coda piena");
    }
  }

  public static int estrai(CodaCircolare c) {
    if (!isEmpty(c)) {
      int elemento = c.coda[c.head];
      c.head = (c.head+1) % c.MAXELEMENTI;
      return elemento;
    } else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getTail(CodaCircolare c) { //.............aggiunto
    if (!isEmpty(c))
      return c.coda[c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1 ];
    else {
      System.out.println("coda vuota");
```

```
      return 0;
    }
  }

  public static int getHead(CodaCircolare c) { //.............aggiunto
    if (!isEmpty(c))
      return c.coda[c.head];
    else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getIndexTail(CodaCircolare c) { //.........aggiunto
    if (!isEmpty(c))
      return c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getIndexHead(CodaCircolare c) { //.........aggiunto
    if (!isEmpty(c))
      return c.head;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getSize(CodaCircolare c) { //.............aggiunto
    return (c.head <= c.tail ? 0 : c.MAXELEMENTI) + (c.tail-c.head);
  }

}
```

```
/*
 * Soluzione esercizio 9
 *
 * A causa dell'implementazione del metodo inserisci, che stampa un
 * messaggio ad ogni inserimento, l'output di questo programma
 * non è esattamente quello richiesto. Se si utilizza la versione
 * di stampaCodaCircolare di UsaCodaCircolare3.java invece
 * l'output è esattamente quello richiesto.
 */
class UsaCodaCircolare2 {
  public static void main(String[] args) {
    System.out.print("Inserisci la dimensione della coda: ");
    int n = Leggi.unInt();
    CodaCircolare c = new CodaCircolare(n);
    int[] temp = new int[n/2];
    int j = 0;
    int e;
    while (!CodaCircolare.isFull(c)) {
      System.out.print("Inserisci un elemento nella coda: ");
      e = Leggi.unInt();
      CodaCircolare.inserisci(c,e);
      if (j < temp.length) {
        temp[j] = e;
        j++;
      }
    }
    CodaCircolare d = new CodaCircolare(n*2,temp);
    //.................................................2 righe aggiunte
    System.out.print("index tail: "+CodaCircolare.getIndexTail(c));
    System.out.println(" tail: "+CodaCircolare.getTail(c));
    stampaCodaCircolare(c);
    //.................................................2 righe aggiunte
    System.out.print("index tail: "+CodaCircolare.getIndexTail(d));
    System.out.println(" tail: "+CodaCircolare.getTail(d));
    stampaCodaCircolare(d);
  }

  static void stampaCodaCircolare(CodaCircolare c) {
    CodaCircolare temp = new CodaCircolare();
    int e;
    String temp2 = "[";
    if (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += e;
    }
    while (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += ","+e;
    }
    temp2 += "]";
    System.out.println(temp2);
    c = temp;
  }
}
```

```
/*
 * Soluzione esercizio 10
 */
class CodaCircolare {
  private int head;
  private int tail;
  private int[] coda;
  private int MAXELEMENTI = 10;

  public CodaCircolare() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num, int[] elem) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    int i;
    for (i = 0; i < elem.length && i < MAXELEMENTI-1; i++)
      coda[i] = elem[i];
    tail = i;
  }

  public static boolean isEmpty(CodaCircolare c) {
    return c.tail == c.head;
  }

  public static boolean isFull(CodaCircolare c) {
    return (c.tail+1)%c.MAXELEMENTI == c.head ;
  }

  public static void inserisci(CodaCircolare c, int elemento) {
    if (!isFull(c)) {
      c.coda [c.tail] = elemento;
      c.tail = (c.tail+1) % c.MAXELEMENTI;
      System.out.println("elemento inserito");
    } else {
      System.out.println("coda piena");
    }
  }

  public static int estrai(CodaCircolare c) {
    if (!isEmpty(c)) {
      int elemento = c.coda[c.head];
      c.head = (c.head+1) % c.MAXELEMENTI;
      return elemento;
    } else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getTail(CodaCircolare c) {
    if (!isEmpty(c))
      return c.coda[c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1 ];
    else {
      System.out.println("coda vuota");
```

```
      return 0;
    }
  }

  public static int getHead(CodaCircolare c) {
    if (!isEmpty(c))
      return c.coda[c.head];
    else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getIndexTail(CodaCircolare c) {
    if (!isEmpty(c))
      return c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getIndexHead(CodaCircolare c) {
    if (!isEmpty(c))
      return c.head;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getSize(CodaCircolare c) {
    return (c.head <= c.tail ? 0 : c.MAXELEMENTI) + (c.tail-c.head);
  }

  public static int[] getVettore(CodaCircolare c) { //.........aggiunto
    int[] temp = new int[getSize(c)];
    int j = getIndexHead(c);
    for (int i = 0; i <temp.length; i++) {
      temp[i] = c.coda[j];
      j = (j+1)%c.MAXELEMENTI;
    }
    return temp;
  }

}
```

```
/*
 * Soluzione prima parte esercizio 11
 */
class CodaCircolare {
  private int head;
  private int tail;
  private int[] coda;
  private int MAXELEMENTI = 10;

  public CodaCircolare() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
  }

  public CodaCircolare(int num, int[] elem) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    int i;
    for (i = 0; i < elem.length && i < MAXELEMENTI-1; i++)
      coda[i] = elem[i];
    tail = i;
  }

  public static boolean isEmpty(CodaCircolare c) {
    return c.tail == c.head;
  }

  public static boolean isFull(CodaCircolare c) {
    return (c.tail+1)%c.MAXELEMENTI == c.head ;
  }

  public static void inserisci(CodaCircolare c, int elemento) {
    if (!isFull(c)) {
      c.coda [c.tail] = elemento;
      c.tail = (c.tail+1) % c.MAXELEMENTI;
      System.out.println("elemento inserito");
    } else {
      System.out.println("coda piena");
    }
  }

  public static int estrai(CodaCircolare c) {
    if (!isEmpty(c)) {
      int elemento = c.coda[c.head];
      c.head = (c.head+1) % c.MAXELEMENTI;
      return elemento;
    } else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getTail(CodaCircolare c) {
    if (!isEmpty(c))
      return c.coda[c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1 ];
    else {
      System.out.println("coda vuota");
```

```
      return 0;
    }
  }

  public static int getHead(CodaCircolare c) {
    if (!isEmpty(c))
      return c.coda[c.head];
    else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getIndexTail(CodaCircolare c) {
    if (!isEmpty(c))
      return c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getIndexHead(CodaCircolare c) {
    if (!isEmpty(c))
      return c.head;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getSize(CodaCircolare c) {
    return (c.head <= c.tail ? 0 : c.MAXELEMENTI) + (c.tail-c.head);
  }

  public static int[] getVettore(CodaCircolare c) {
    int[] temp = new int[getSize(c)];
    int j = getIndexHead(c);
    for (int i = 0; i <temp.length; i++) {
      temp[i] = c.coda[j];
      j = (j+1)%c.MAXELEMENTI;
    }
    return temp;
  }

  //.....................................................aggiunto
  public static CodaCircolare somma(CodaCircolare c, CodaCircolare d) {
    int[] elementiC = getVettore(c);
    int[] elementiD = getVettore(d);
    int dimensione = c.MAXELEMENTI+d.MAXELEMENTI;
    CodaCircolare codaRisultato = new CodaCircolare(dimensione,elementiC);
    /*
     * Le righe di codice sopra potevano essere scritte anche:
     * int[] elementiD = getVettore(d);
     * CodaCircolare codaRrisultato =
     *   new CodaCircolare(c.MAXELEMENTI+d.MAXELEMENTI,getVettore(c));
     */
    for (int i = 0; i<getSize(d); i++)
      inserisci(codaRisultato,elementiD[i]);
    return codaRisultato;
  }
}
```

```
/*
 * Soluzione seconda parte esercizio 11
 */
class UsaCodaCircolare3 {
  public static void main(String[] args) {
    int[] elementi1 = {1,2,3,4};
    int[] elementi2 = {-2,12};
    CodaCircolare c1 = new CodaCircolare(6,elementi1);
    CodaCircolare c2 = new CodaCircolare(23,elementi2);
    CodaCircolare c3 = CodaCircolare.somma(c1,c2);
    stampaCodaCircolare(c1);
    stampaCodaCircolare(c2);
    stampaCodaCircolare(c3);
  }

  static void stampaCodaCircolare(CodaCircolare c) {
    CodaCircolare temp = new CodaCircolare();
    int e;
    String temp2 = "|";
    if (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += e;
    }
    while (!CodaCircolare.isEmpty(c)) {
      e = CodaCircolare.estrai(c);
      CodaCircolare.inserisci(temp,e);
      temp2 += "."+e;
    }
    temp2 += "]";
    System.out.println(temp2);
    c = temp;
  }
}
```

```
/*
 * Soluzione esercizio 12
 */
class CodaCircolare {
  private int head;
  private int tail;
  private int[] coda;
  private int MAXELEMENTI = 10;
  private int numElementi; //.................................aggiunto

  public CodaCircolare() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
    numElementi = 0; //.....................................aggiunto
  }

  public CodaCircolare(int num) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
    numElementi = 0; //.....................................aggiunto
  }

  public CodaCircolare(int num, int[] elem) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    int i;
    for (i = 0; i < elem.length && i < MAXELEMENTI; i++) //..modificato
      coda[i] = elem[i];
    tail = i%MAXELEMENTI; //................................modificato
    numElementi = i; //.....................................aggiunto
  }

  public static boolean isEmpty(CodaCircolare c) {
    return c.numElementi == 0; //...........................modificato
  }

  public static boolean isFull(CodaCircolare c) {
    return c.MAXELEMENTI == c.numElementi; //...............modificato
  }

  public static void inserisci(CodaCircolare c, int elemento) {
    if (!isFull(c)) {
      c.coda [c.tail] = elemento;
      c.tail = (c.tail+1) % c.MAXELEMENTI;
      c.numElementi++; //..................................aggiunto
      System.out.println("elemento inserito");
    } else {
      System.out.println("coda piena");
    }
  }

  public static int estrai(CodaCircolare c) {
    if (!isEmpty(c)) {
      int elemento = c.coda[c.head];
      c.head = (c.head+1) % c.MAXELEMENTI;
      c.numElementi--; //..................................aggiunto
      return elemento;
    } else {
      System.out.println("coda vuota");
      return 0;
    }
  }
```

```
  public static int getTail(CodaCircolare c) {
    if (!isEmpty(c))
      return c.coda[c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1 ];
    else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getHead(CodaCircolare c) {
    if (!isEmpty(c))
      return c.coda[c.head];
    else {
      System.out.println("coda vuota");
      return 0;
    }
  }

  public static int getIndexTail(CodaCircolare c) {
    if (!isEmpty(c))
      return c.tail == 0 ? c.MAXELEMENTI-1 : c.tail-1;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getIndexHead(CodaCircolare c) {
    if (!isEmpty(c))
      return c.head;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public static int getSize(CodaCircolare c) {
    return c.numElementi; //.............................modificato
  }

  public static int[] getVettore(CodaCircolare c) {
    int[] temp = new int[getSize(c)];
    int j = getIndexHead(c);
    for (int i = 0; i <temp.length; i++) {
      temp[i] = c.coda[j];
      j = (j+1)%c.MAXELEMENTI;
    }
    return temp;
  }

  public static CodaCircolare somma(CodaCircolare c, CodaCircolare d) {
    int[] elementiD = getVettore(d);
    CodaCircolare codaRisultato =
      new CodaCircolare(c.MAXELEMENTI+d.MAXELEMENTI,getVettore(c));
    for (int i = 0; i<getSize(d); i++)
      inserisci(codaRisultato,elementiD[i]);
    return codaRisultato;
  }
}
```

```
class CodaCircolareOO {
  private int head;
  private int tail;
  private int[] coda;
  private int MAXELEMENTI = 10;
  private int numElementi;

  public CodaCircolareOO() {
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
    numElementi = 0;
  }

  public CodaCircolareOO(int num) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    tail=head;
    numElementi = 0;
  }

  public CodaCircolareOO(int num, int[] elem) {
    MAXELEMENTI = num;
    coda = new int[MAXELEMENTI];
    head = 0;
    int i;
    for (i = 0; i < elem.length && i < MAXELEMENTI; i++)
      coda[i] = elem[i];
    tail = i%MAXELEMENTI;
    numElementi = i;
  }

  public boolean isEmpty() {
    return numElementi == 0;
  }

  public boolean isFull() {
    return MAXELEMENTI == numElementi;
  }

  public void inserisci(int elemento) {
    if (!isFull()) {
      coda [tail] = elemento;
      tail = (tail+1) % MAXELEMENTI;
      numElementi++;
    } else
      System.out.println("coda piena");
  }

  public int estrai() {
    if (!isEmpty()) {
      int elemento = coda[head];
      head = (head+1) % MAXELEMENTI;
      numElementi--;
      return elemento;
    } else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public int getTail() {
    if (!isEmpty())
      return coda[tail == 0 ? MAXELEMENTI-1 : tail-1 ];
    else {
```

```
      System.out.println("coda vuota");
      return -1;
    }
  }

  public int getHead() {
    if (!isEmpty())
      return coda[head];
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public int getIndexTail() {
    if (!isEmpty())
      return tail == 0 ? MAXELEMENTI-1 : tail-1;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public int getIndexHead() {
    if (!isEmpty())
      return head;
    else {
      System.out.println("coda vuota");
      return -1;
    }
  }

  public int getSize() {
    return numElementi;
  }

  public int[] getVettore() {
    int[] temp = new int[getSize()];
    int j = getIndexHead();
    for (int i = 0; i <temp.length; i++) {
      temp[i] = coda[j];
      j = (j+1)%MAXELEMENTI;
    }
    return temp;
  }

  /*
   * Attenzione. Questo metodo può rimanere static.
   * Si potrebbe aggiungere un metodo non static
   *   public CodaCircolareOO sommaCon(CodaCircolareOO c)
   * con il significato implicito di sommare l'oggetto
   * corrente (this) con quello passato come parametro.
   */
  public static CodaCircolareOO somma(CodaCircolareOO c, CodaCircolareOO d) {
    int[] elementiD = d.getVettore();
    CodaCircolareOO codaRisultato =
      new CodaCircolareOO(c.MAXELEMENTI+d.MAXELEMENTI,c.getVettore());
    for (int i = 0; i<d.getSize(); i++)
      codaRisultato.inserisci(elementiD[i]);
    return codaRisultato;
  }
}
```

```
class UsaCodaCircolareOO {
  public static void main(String[] args) {
    CodaCircolareOO c1 = new CodaCircolareOO();
    while (!c1.isFull()) {
      System.out.print("Inserisci un elemento nella coda c1: ");
      c1.inserisci(Leggi.unInt());
    }
    CodaCircolareOO d1 = new CodaCircolareOO();
    for (int i = 1; i <= 3; i++)
      d1.inserisci(c1.estrai());
    System.out.println("Coda c1 e d1 dopo aver inserito i primi 3 "+
                       "elementi di c1 in d1:");
    stampaCodaCircolareOO(c1);
    stampaCodaCircolareOO(d1);
    System.out.print("Inserisci la dimensione della coda c2: ");
    int n = Leggi.unInt();
    CodaCircolareOO c2 = new CodaCircolareOO(n);
    int[] temp = new int[n/2];
    int j = 0;
    int e;
    while (!c2.isFull()) {
      System.out.print("Inserisci un elemento nella coda c2: ");
      e = Leggi.unInt();
      c2.inserisci(e);
      if (j < temp.length) {
        temp[j] = e;
        j++;
      }
    }
    CodaCircolareOO d2 = new CodaCircolareOO(n*2,temp);
    System.out.println("Coda c2");
    System.out.print("index tail: "+c2.getIndexTail());
    System.out.println(" tail: "+c2.getTail());
    stampaCodaCircolareOO(c2);
    System.out.println("Coda d2");
    System.out.print("index tail: "+d2.getIndexTail());
    System.out.println(" tail: "+d2.getTail());
    stampaCodaCircolareOO(d2);
    int[] elementi1 = {1,2,3,4};
    int[] elementi2 = {-2,12};
    CodaCircolareOO c3 = new CodaCircolareOO(6,elementi1);
    CodaCircolareOO c4 = new CodaCircolareOO(23,elementi2);
    CodaCircolareOO c5 = CodaCircolareOO.somma(c3,c4);
    CodaCircolareOO c6 = c3.somma(c3,c4);
    CodaCircolareOO c7 = c4.somma(c3,c4);
    System.out.println("Coda c3");
    stampaCodaCircolareOO(c3);
    System.out.println("Coda c4");
    stampaCodaCircolareOO(c4);
    System.out.println("Coda c5 ottenuta con invocazione "+
                       "CodaCircolareOO.somma(c3,c4)");
    stampaCodaCircolareOO(c5);
    System.out.println("Coda c6 ottenuta con invocazione "+
                       "c3.somma(c3,c4)");
    stampaCodaCircolareOO(c6);
    System.out.println("Coda c7 ottenuta con invocazione "+
                       "c4.somma(c3,c4)");
    stampaCodaCircolareOO(c7);
  }

  static void stampaCodaCircolareOO(CodaCircolareOO c) {
    CodaCircolareOO temp = new CodaCircolareOO();
    int e;
    String temp2 = "[";
    if (!c.isEmpty()) {
      e = c.estrai();
```

```
      temp.inserisci(e);
      temp2 += e;
    }
    while (!c.isEmpty()) {
      e = c.estrai();
      temp.inserisci(e);
      temp2 += ","+e;
    }
    temp2 += "]";
    System.out.println(temp2);
    c = temp;
  }
}
```