

Studente (Cognome Nome): \_\_\_\_\_

Matricola: \_\_\_\_\_

Corso di Informatica  
Corso di Laurea in Ingegneria Gestionale  
a.a. 2006-07  
Secondo Compitino – 21 Dicembre 2006

*Si noti che le soluzioni ai quesiti saranno considerate valide solo se il materiale consegnato includerà anche lo svolgimento. Tale foglio deve essere consegnato insieme allo svolgimento.*

## Quesito 1

Una libreria è rappresentata da un array di libri, identificati dal titolo e dalla quantità di libri disponibili per ciascun titolo. Un ordine eseguito da un cliente è rappresentato da un array di libri, identificati dal titolo e dalla quantità ordinata.

**a) Scrivere un metodo Java che verifichi che un ordine sia valido.** Il metodo deve avere il seguente prototipo:

```
public static boolean is_valid(String [] lib_titolo, int []  
lib_quantità, String [] ordine_titolo, int [] ordine_quantità)
```

e deve restituire true se:

- 1) tutti i titoli dell'ordine sono presenti in libreria
- 2) la quantità per ciascun elemento dell'ordine è  $\leq$  della disponibilità in libreria.

Prerequisiti del metodo: gli array non sono null e le quantità sono  $\geq 0$ .

**b) Scrivere un metodo Java che aggiorna l'elenco dei libri presenti in libreria in relazione all'ordine ricevuto, decrementando il numero delle copie disponibili della quantità richiesta.** Il metodo deve avere il seguente prototipo:

```
public static void aggiorna(String [] lib_titolo, int [] lib_quantità,  
String [] ordine_titolo, int [] ordine_quantità)
```

Prerequisiti del metodo: gli array non sono null e l'ordine è valido.

**c) Scrivere un metodo Java che calcola il numero di elementi non validi all'interno di un ordine.** Il metodo deve avere il seguente prototipo:

```
public static int non_validi(String [] lib_titolo, int []  
lib_quantità, String [] ordine_titolo, int [] ordine_quantità)
```

Prerequisiti del metodo: gli array non sono null.

**d) Scrivere un programma main di test che**

- Inizializza la libreria con i seguenti valori:

Titolo	Disponibilità
Analisi I	4
C++	2
Visual Basic	7
Visio	1

- Crea un ordine non valido.

- Se l'ordine non è valido, crei da questo, un ordine valido eliminando gli elementi che lo rendono non valido. Suggerimento: Quanti sono gli elementi presenti nel nuovo ordine (che deve essere valido)?

Esempio: se l'ordine è il seguente:

Titolo	Quantità
C#	4
Visual Basic	1
C++	6

L'ordine valido derivato da questo è

Titolo	Quantità
Visual Basic	1

- Aggiorna l'elenco dei libri presenti in libreria
- Stampa il nuovo elenco con un formato analogo alla tabella di sopra

Note:

- E' possibile risolvere l'esercizio anche usando le classi. In tal caso, modificare in modo coerente l'intestazione delle funzioni.
- E' possibile scrivere funzioni aggiuntive per risolvere dei compiti richiesti in più punti del programma.

## Quesito 2

Si rappresenti il diagramma di flusso relativo al seguente metodo Java:

```
public static void f(int [] v) {
    v = new int [4];
    for(int i=0;i<v.length;i++){
        v[i]=i*i;
        if (v[i]>=4)
            i++;
    }
    System.out.println(v[2]+" "+v[3]);
}
```

## Quesito 3

Si determini l'output prodotto dall'esecuzione del seguente programma Java, in cui il corpo del metodo f e' stato omesso per brevità, essendo identico al metodo del quesito precedente.

```
public static void main(String[] args) {
    int [] a = new int [4];
    System.out.println(a[2]+" "+a[3]);
    f(a);
    System.out.println(a[2]+" "+a[3]);
}
```

## Quesito 1: soluzione.

Note: il quesito può essere risolto in svariati modi. Qui ne viene proposto uno. Per una soluzione che fa un uso estensivo dei metodi, vedi il testo e la soluzione del quesito C, dove cambiano solo i nomi dei parametri e l'inizializzazione, ma gli algoritmi sono identici.

```
public class testoA {

    public static boolean is_valid(
        String[] lib_titolo,
        int[] lib_quantita,
        String[] ordine_titolo,
        int[] ordine_quantita) {
        // prima soluzione: si contano i numeri degli elementi dell'ordine
        // che sono validi.
        // se sono uguali alla lunghezza dell'ordine, l'ordine è valido
        boolean valido = false; // l'inizializzazione non è imp.
        int num_validi = 0; // numero di ordini validi
        int i; // indice per ciclare sugli ordini
        int j; // indice per ciclare sui libri
        for (i = 0; i < ordine_titolo.length; i++) {
            for (j = 0; j < lib_titolo.length; j++) {
                // cerca un titolo dell'ordine
                // presente in libreria
                if (ordine_titolo[i].equals(lib_titolo[j]))
                    // condizione di validità
                    // nota: non testo il caso di lib_quantita < 0
                    // perchè è dato come prerequisito
                    if (ordine_quantita[i] <= lib_quantita[j])
                        num_validi++; // la riga dell'ordine è valida
            } // end_for_j
        } // end_for_i

        // test di validità:
        // il numero di righe valide deve
        // essere uguale alla lunghezza dell'ordine
        if (num_validi == ordine_titolo.length)
            valido = true;
        else
            valido = false;

        return (valido);
    } //end_is_valid

    // seconda soluzione, quella suggerita durante l'esame:
    // un ordine è valido
    // se il numero di entry non valide è pari a 0
    public static boolean is_valid1(
        String[] lib_titolo,
        int[] lib_quantita,
        String[] ordine_titolo,
        int[] ordine_quantita) {

        boolean valido;
        if (non_validi(lib_titolo,
            lib_quantita, ordine_titolo, ordine_quantita) == 0)
            valido = true;
        else
            valido = false;
    }
}
```

```

        return valido;
    }

    public static void aggiorna(
        String[] lib_titolo,
        int[] lib_quantita,
        String[] ordine_titolo,
        int[] ordine_quantita) {
        // per ogni titolo ordinato,
        // lo cerco in libreria
        // ed aggiornno le relative quantità
        //nota:non occorrono controlli
        // perchè l'ordine è valido
        int i; // contatore sugli ordini
        int j; // contatore sui libri
        for (i = 0; i < ordine_titolo.length; i++) {
            for (j = 0; j < lib_titolo.length; j++) {
                if (ordine_titolo[i].equals(lib_titolo[j]))
                    // aggiornno la quantità
                    lib_quantita[j] -= ordine_quantita[i];
            } // end_for_j
        } // end_for_i
    } //end_aggiorna

    public static int non_validi(
        String[] lib_titolo,
        int[] lib_quantita,
        String[] ordine_titolo,
        int[] ordine_quantita) {
        // si contano i numeri degli elementi dell'ordine
        // che sono validi.
        // il numero di elementi non validi è la lunghezza dell'array
        // degli ordini - il numero di entry non valide
        int non_validi = 0; // numero di righe non valide
        int num_validi = 0; // numero di ordini validi
        int i; // indice per ciclare sugli ordini
        int j; // indice per ciclare sui libri
        for (i = 0; i < ordine_titolo.length; i++) {
            for (j = 0; j < lib_titolo.length; j++) {
                // cerca un titolo dell'ordine
                // presente in libreria
                if (ordine_titolo[i].equals(lib_titolo[j]))
                    // condizione di validità
                    if ((ordine_quantita[i] >= 0)
                        && (ordine_quantita[i] <= lib_quantita[j]))
                        num_validi++; //la riga dell'ordine è valida

            } // end_for_j
        } // end_for_i

        non_validi = ordine_titolo.length - num_validi;

        return (non_validi);
    } //end_non_validi

    public static void main(String[] args) {

        // inizializzazione libreria
        String[] lib_titolo =
            new String[] { "Analisi I", "C++", "Visual Basic", "Visio" };
        int[] lib_quantita = new int[] { 4, 2, 7, 1 };
    }

```

```

// creazione ordine non valido
// (quello suggerito nel testo)
String[] ordine_titolo = new String[] { "C#", "Visual Basic", "C++"
};

int[] ordine_quantita = new int[] { 4, 1, 6 };

boolean valido =
    is_valid(lib_titolo, lib_quantita, ordine_titolo,
ordine_quantita);

// test di validità e azioni conseguenti.
if (!valido) {
    // creo un ordine temporaneo fatto da due array paralleli.
    // la dimensione è pari alla dimensione del vecchio
    // ordine - il numero di entry non valide
    int elem_non_validi =
        non_validi(
            lib_titolo,
            lib_quantita,
            ordine_titolo,
            ordine_quantita);
    int lunghezza = ordine_titolo.length - elem_non_validi;
    String[] tmp_ord_titolo = new String[lunghezza];
    int[] tmp_ord_quantita = new int[lunghezza];

    // tutti gli elementi validi li ricopio negli array fittizi
    int i, j; // due indici per scorrere i libri e gli ordini
    int k = 0; // indice dell'elemento inserito nell'array
temporaneo
    for (i = 0; i < ordine_titolo.length; i++)
        for (j = 0; j < lib_titolo.length; j++) {
            // condizioni di validità di una entry
            if (ordine_titolo[i].equals(lib_titolo[j])) {
                if ((ordine_quantita[i] >= 0)
                    && (ordine_quantita[i] <=
lib_quantita[j])) {
                    // l'entry è valida, la ricopio
nell'ordine
                    // temporaneo
                    tmp_ord_titolo[k] = ordine_titolo[i];
                    tmp_ord_quantita[k] =
ordine_quantita[i];
                    k++;
                } // end_if ordine_quantita
            } // end_if ordine_titolo
        } // end_for_j

    // ora butto via il vecchio ordine e prendo l'ordine valido
    ordine_titolo = tmp_ord_titolo;
    ordine_quantita = tmp_ord_quantita;

} // end_if_not_valido

// aggiorno il catalogo
aggiorna(lib_titolo, lib_quantita, ordine_titolo, ordine_quantita);

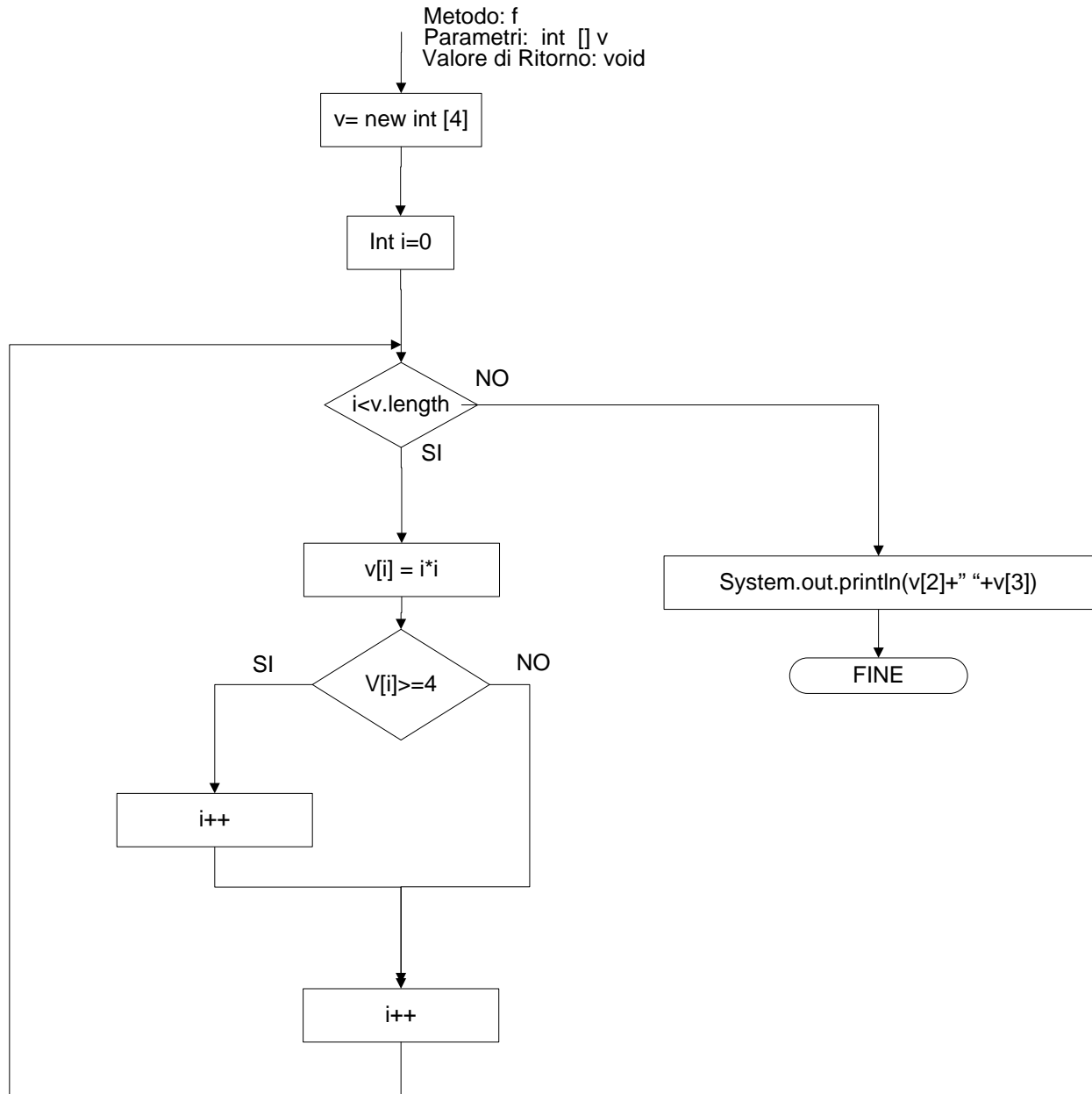
// stampo il catalogo
System.out.println(" Titolo\tQuantità");
for (int i = 0; i < lib_titolo.length; i++) {
    System.out.println(" " + lib_titolo[i] + "\t" +
lib_quantita[i]);
}

} // end_main

```

```
} // end_class
```

### Quesito 2: soluzione.



### Quesito 3: soluzione.

L'output del programma è il seguente:

```
0 0  
4 0  
0 0
```

Per la giustificazione, si ricorre alla evoluzione dello stato del programma, identificando i contenuti dell'area locale ed area globale e seguendo la loro evoluzione.