Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica
a.a. 2010-11
22 luglio 2011

Testo

Il database di un videonoleggio è costituito da due vettori paralleli. Il primo è denominato "videoteca" e contiene oggetti di tipo "Film" che rappresentano i film presenti nell'archivio. Il secondo vettore è denominato "noleggi" e contiene oggetti di tipo "Noleggio" che rappresentano le informazioni di noleggio di un film. Le informazioni di noleggio del film presente in una determinata posizione nel vettore "videoteca", si trovano nella corrispondente posizione nel vettore "noleggi". Nel caso un dato film non sia stato noleggiato, nella sua posizione nel vettore "noleggi" sarà presente un riferimento "null". Entrambi i vettori hanno dimensione pari alla costante "MAX_FILM" (inizializzata a 256). Se il numero di film contenuti nell'archivio è inferiore a "MAX_FILM", i primi elementi del vettore conterranno gli oggetti di tipo "Film", mentre gli altri conterranno riferimenti "null". Tutti gli elementi "null" del vettore "videoteca" si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Film contiene le informazioni relative ad un film ed un metodo per la stampa a video di queste informazioni:

```
public class Film {

    private int codice;
    public String titolo;
    public String regia;
    public int anno;
    public String[] cast;

    public Film(int co, String t, String r, int a, String[] ca) {
        codice = co;
        titolo = t;
        regia = r;
        anno = a;
        cast = ca;
    }

    public int getCodice(){
        return codice;
    }

    public void stampaInfo() {

        System.out.println("Film n° " + codice + " / " + titolo + " / " + anno
                + " / " + regia);

        if (cast != null && cast.length > 0) {
            System.out.print("\tCast:");
            for (int i = 0; i < cast.length - 1; i++) {
                System.out.print(" " + cast[i] + ",");
            }
            System.out.println(" " + cast[cast.length - 1]);
        }
    }

}
```

La classe Noleggio contiene il nome dell'utente che ha noleggiato un determinato film e la data di noleggio:

```
public class Noleggio {

    public String utente;
    public int giorno;
    public int mese;
    public int anno;

    public Noleggio(String u, int g, int m, int a) {
        utente = u;
        giorno = g;
        mese = m;
        anno = a;
    }
}
```

Si consiglia di procedere nel seguente modo: implementare un metodo e successivamente scrivere la parte del main che utilizza tale metodo, in modo da poterne verificare immediatamente la correttezza.

Le varie operazioni devono essere eseguite sulla porzione significativa dell'archivio, cioè la porzione di "videoteca" che non contiene riferimenti "null".

a) Scrivere il metodo statico:

```
public static void ordinaData(Film[] filmdb, Noleggio[] noldb)
```

che prende in ingresso i vettori "filmdb" e "noldb" costituenti il database di un videonoleggio e ordina i vettori a seconda della data di noleggio, da quello noleggiato prima a quello noleggiato dopo. I film non noleggiati vanno spostati dopo quelli noleggiati.

b) Scrivere il metodo statico:

```
public static Film[] trovaFilm(Film[] filmdb, String attSi, String attNo)
```

che prende in ingresso il vettore "filmdb" che rappresenta l'archivio della videoteca e le stringhe "attSi" e "attNo". Il metodo deve ritornare un vettore di Film contenente soltanto i film che hanno nel cast l'attore specificato dalla stringa "attSi" ma non l'attore specificato dalla stringa "attNo".

c) Scrivere il metodo statico:

```
public static Film[] noleggiati(Film[] filmdb, Noleggio[] noldb)
```

che prende in ingresso i vettori "filmdb" e "noldb" costituenti il database di un videonoleggio e ritorna un vettore di Film contenente i film noleggiati. Se non ci sono film noleggiati, il metodo non deve creare nessun vettore e deve ritornare "null".

d) Scrivere il metodo statico:

```
public static boolean eliminaFilm(Film[] filmdb, Noleggio noldb[],
        int codice)
```

che prende in ingresso i vettori "filmdb" e "noldb" costituenti il database di un videonoleggio ed un intero "codice". Il metodo deve eliminare dall'archivio il film avente codice pari al parametro "codice", se presente. Se l'eliminazione ha successo, il metodo deve ritornare "true". Se, invece, nessun film con il codice appropriato viene trovato nell'archivio, il metodo deve ritornare "false". N.B. L'archivio, dopo l'eliminazione, deve essere ricompattato, altrimenti possono rimanere dei buchi che impedirebbero il corretto funzionamento dei metodi.

e) Scrivere il metodo main che:

- definisca ed inizializzi i vettori "videoteca" e "noleggi" secondo i valori riportati in tabella e stampi a video l'archivio. La stampa dell'archivio consiste nella stampa delle informazioni di ogni film (usando il metodo "stampaInfo" della classe "Film") e, se il film è noleggiato, deve essere indicato anche l'utente e la data di noleggio.

Codice	Titolo	Regia	Anno	Cast	Utente	Data
0	Berlinguer ti voglio bene	Giuseppe Bertolucci	1977	Roberto Benigni Carlo Monni		
1	Johnny Stecchino	Roberto Benigni	1991	Roberto Benigni Nicoletta Braschi	Gianni Verdi	20/7/2011
2	Il mostro	Roberto Benigni	1994	Roberto Benigni Nicoletta Braschi	Mario Rossi	6/7/2011
3	Non ci resta che piangere	Troisi e Benigni	1985	Massimo Troisi Roberto Benigni	Mario Rossi	22/6/2011

- ordini i vettori "videoteca" e "noleggi" utilizzando il metodo "ordinaData" e, poi, stampi nuovamente l'archivio.
- stampi l'elenco dei titoli dei film dove ha recitato "Roberto Benigni" ma non "Nicoletta Braschi", usando il metodo "trovaFilm".
- stampi il titolo dei film noleggiati, usando il metodo "noleggiati". Se non ci sono film noleggiati, va stampato un messaggio d'errore.
- elimini, utilizzando il metodo "eliminaFilm", dall'archivio il film avente come codice "2" e stampi l'esito del metodo. Se l'eliminazione avviene con successo, si stampi nuovamente l'archivio.

Soluzione

```
public class Esame {

    public static final int MAX_FILM = 256;

    /*
     * Questo metodo conta i film presenti nell'archivio. Non è richiesto dal
     * testo, ma semplifica la scrittura degli altri metodi.
     */
    public static int contaFilm(Film[] filmdb) {
        int nFilm = 0;
        for (int i = 0; i < MAX_FILM; i++) {
            if (filmdb[i] != null) {
                nFilm++;
            }
        }
        /*
         * Per evitare di scorrere l'intero array, il conteggio può essere
         * realizzato anche nel seguente modo (considerando che non ci devono
         * essere elementi nulli in mezzo agli elementi validi):
         */
        /*
        int nFilm = 0;
        while (nFilm < MAX_FILM && filmdb[nFilm] != null) {
            nFilm++;
        }
        */
        return nFilm;
    }


    /*
     * Questo metodo stampa le informazioni relative a tutti i film presenti
     * nell'archivio e, per i film noleggiati, indica l'utente che li ha
     * noleggiati e la data di noleggio. Non è richiesto dal testo, ma
     * semplifica la scrittura del main
     */
    public static void stampaArchivio(Film[] filmdb, Noleggio[] noldb) {
        /* Conteggio dei film presenti nell'archivio */
        int nFilm = contaFilm(filmdb);

        for (int i = 0; i < nFilm; i++) {
            filmdb[i].stampaInfo();
            if (noldb[i] != null) {
                System.out.println("\tNoleggiato a " + noldb[i].utente + " il "
                        + noldb[i].giorno + "/" + noldb[i].mese + "/"
                        + noldb[i].anno);
            }
        }
    }

    /*
     * Questo metodo confronta due date, ciascuna espressa da tre interi che
     * indicano giorno, mese ed anno. Ritorna un valore:
     * positivo se la prima data è maggiore della seconda;
     * negativo se la prima data è minore della seconda;
     * zero se le date sono uguali.
     * Non è richiesto dal testo, ma semplifica la scrittura dei metodi.
     */
    public static int confrontaData(int g1, int m1, int a1, int g2, int m2,
            int a2) {
        /*
         * Prima si confrontano gli anni. La differenza tra gli anni (qualora
         * sia diversa da 0) delle due date costituisce anche il valore di
         * ritorno, perché se è positiva allora la prima data è maggiore,
         * mentre se negativa la prima data è minore.
         */
        int ret = a1 - a2;
        /*
         * Nel caso in cui l'anno sia lo stesso, allora il confronto avviene
```

```
         * sul mese in maniera simile a quanto già visto.
         */
        if (ret == 0) {
            ret = m1 - m2;
            /*
             * Nel caso in cui anche il mese sia lo stesso, allora il confronto
             * avviene sul giorno.
             */
            if (ret == 0) {
                ret = g1 - g2;
            }
        }
        return ret;
    }

    public static void ordinaData(Film[] filmdb, Noleggio[] noldb) {
        /* Conteggio dei film presenti nell'archivio */
        int nFilm = contaFilm(filmdb);
        /* Ordinamento */
        for (int i = 0; i < nFilm - 1; i++) {
            for (int j = i + 1; j < nFilm; j++) {
                /*
                 * Gli elementi vanno scambiati in due casi:
                 * 1) se filmdb[i] non è stato noleggiato mentre filmdb[j] sì;
                 * 2) se filmdb[i] è stato noleggiato dopo filmdb[j].
                 * Il primo caso serve a spostare gli elementi non noleggiati
                 * dopo quelli noleggiati. Si può esprimere nel seguente modo:
                 * noldb[i] == null && noldb[j] != null
                 * Il secondo caso serve ad ordinare per data gli elementi
                 * noleggiati. Il confronto viene effettuato controllando il
                 * valore di ritorno del metodo confrontaData.
                 * In questo caso, però, prima di poter confrontare le due date
                 * bisogna assicurarsi che entrambi i film siano noleggiati,
                 * altrimenti si può incorrere in un errore tentando di
                 * accedere ad un oggetto inesistente (riferimento nullo).
                 */
                if ((noldb[i] == null && noldb[j] != null)
                        || (noldb[i] != null && noldb[j] != null &&
                                confrontaData(noldb[i].giorno, noldb[i].mese,
                                        noldb[i].anno, noldb[j].giorno,
                                        noldb[j].mese, noldb[j].anno) > 0)) {
                    /* Scambio i due film nelle posizioni i e j */
                    Film tmp = filmdb[i];
                    filmdb[i] = filmdb[j];
                    filmdb[j] = tmp;
                    /*
                     * Per mantenere la consistenza dei dati, vanno scambiati
                     * anche i corrispondenti elementi del vettore parallelo
                     */
                    Noleggio tmp2 = noldb[i];
                    noldb[i] = noldb[j];
                    noldb[j] = tmp2;
                }
            }
        }
    }

    public static Film[] trovaFilm(Film[] filmdb, String attSi, String attNo) {
        /* Conteggio dei film presenti nell'archivio */
        int nFilm = contaFilm(filmdb);
        int nRis = 0;
        /* Conteggio degli elementi del vettore risultato */
        /* Il primo for scorre tutti i film dell'archivio */
        for (int i = 0; i < nFilm; i++) {
            /*
             * Per ogni film, si cercano nel cast attSi ed attNo. Se nel cast
             * di filmdb[i] è stato trovato l'attore attSi, la variabile
             * booleana trovatoAttSi viene messa a true. Se nel cast viene
             * trovato l'attore attNo, allora si mette a true la variabile
             * booleana trovatoAttNo. In questo modo al termine del ciclo
             * si può sapere se attSi ed attNo erano presenti nel cast.
```

```
             */
            boolean trovatoAttSi = false;
            boolean trovatoAttNo = false;
            for (int j = 0; j < filmdb[i].cast.length; j++) {
                if (filmdb[i].cast[j].equals(attSi)) {
                    trovatoAttSi = true;
                }
                if (filmdb[i].cast[j].equals(attNo)) {
                    trovatoAttNo = true;
                }
            }
            /*
             * Dopo la ricerca nel cast si controlla se è stato trovato attSi e
             * non è stato trovato attNo. In tal caso, è stato trovato un film
             * da inserire nel vettore dei risultati, quindi si incrementa il
             * contatore nRis.
             */
            if(trovatoAttSi && !trovatoAttNo){
                nRis++;
            }
        }
        /* Creazione del vettore risultato */
        Film[] ris = new Film[nRis];
        /* Popolamento del vettore risultato */
        int k = 0;
        /* Il primo for scorre tutti i film dell'archivio */
        for (int i = 0; i < nFilm; i++) {
            /*
             * Come prima, si cercano gli attori attSi ed attNo nel cast,
             * settando le variabili trovatoAttSi e trovatoAttNo in maniera
             * adeguata.
             */
            boolean trovatoAttSi = false;
            boolean trovatoAttNo = false;
            for (int j = 0; j < filmdb[i].cast.length; j++) {
                if (filmdb[i].cast[j].equals(attSi)) {
                    trovatoAttSi = true;
                }
                if (filmdb[i].cast[j].equals(attNo)) {
                    trovatoAttNo = true;
                }
            }
            /*
             * Se è stato trovato attSi e non attNo, si aggiunge il film al
             * vettore di output e viene aggiornato l'indice k degli elementi
             * del vettore di output.
             */
            if(trovatoAttSi && !trovatoAttNo){
                ris[k] = filmdb[i];
                k++;
            }
        }
        return ris;
    }

    public static Film[] noleggiati(Film[] filmdb, Noleggio[] noldb){
        /* Conteggio dei film presenti nell'archivio */
        int nFilm = contaFilm(filmdb);
        int nRis = 0;
        /* Conteggio degli elementi del vettore risultato */
        for (int i = 0; i < nFilm; i++) {
            if (noldb[i] != null) {
                nRis++;
            }
        }
        /*
         * Inizalizzo il valore di ritorno a null. Il vettore sarà creato solo
         * se ci sono elementi, come richiesto dal testo.
         */
        Film[] ris = null;
        if (nRis > 0) {
```

```
            /* Creazione del vettore risultato */
            ris = new Film[nRis];
            /* Popolamento del vettore risultato */
            int k = 0;
            for (int i = 0; i < nFilm; i++) {
                if (noldb[i] != null) {
                    ris[k] = filmdb[i];
                    k++;
                }
            }
        }
        return ris;
    }

    public static boolean eliminaFilm(Film[] filmdb, Noleggio noldb[],
            int codice) {
        /* Conteggio dei film presenti nell'archivio */
        int nFilm = contaFilm(filmdb);
        /*
         * Ricerca dell'indice del film da eliminare. Il valore sarà
         * memorizzato in idx, che viene inizializzato a -1, un valore non
         * valido.
         */
        int idx = -1;
        /*
         * Il ciclo for viene interrotto non appena l'indice viene trovato.
         * Modificando la variabile idx con un valore valido, infatti, la
         * condizione (idx == -1) viene invalidata causando la terminazione del
         * ciclo. Se non viene trovato nessun film con il codice richiesto, al
         * termine del ciclo (una volta esaminati tutti i film) la variabile
         * idx avrà ancora assegnato il valore -1.
         */
        for (int i = 0; i < nFilm && idx == -1; i++) {
            if (filmdb[i].getCodice() == codice) {
                idx = i;
            }
        }
        /* Variabile per il valore di ritorno */
        boolean ret = false;
        /*
         * Se idx vale -1, il codice non è stato trovato nell'archivio. Il
         * metodo non deve eliminare nulla e deve ritornare false. Se, invece,
         * idx ha un valore valido, allora si procede con l'eliminazione.
         */
        if (idx != -1) {
            ret = true;
            /*
             * Per effettuare correttamente l'eliminazione, bisogna assicurarsi
             * che non rimangano "buchi" nell'archivio. Quindi va fatta una
             * operazione di compattazione dell'archivio. L'eliminazione
             * avviene automaticamente assieme alla compattazione, infatti, per
             * compattare l'archivio si spostano tutti i riferimenti ai film
             * successivi a quello da eliminare indietro di una posizione. Così
             * facendo, il rifermento nella posizione idx + 1 sarà copiato
             * sopra quello da eliminare (in posizione idx).
             */
            for (int i = idx + 1; i < nFilm; i++) {
                filmdb[i - 1] = filmdb[i];
                noldb[i - 1] = noldb[i];
            }
            /*
             * Alla fine dell'operazione, si avranno due copie del riferimento
             * all'ultimo film, una su filmdb[nFilm - 2] ed una su
             * filmdb[nFilm - 1]. Si risolve il problema cancellando l'ultima
             * copia del riferimento.
             */
            filmdb[nFilm - 1] = null;
            noldb[nFilm - 1] = null;
            /*
             * L'operazione di compattazione sopra descritta permette di
             * preservare l'ordinamento del vettore. Tuttavia, questo non era
```

```
             * un requisito richiesto dal testo. Quindi sarebbe stato possibile
             * effettuare un'operazione di eliminazione e compattazione molto
             * più semplice, che però non lascia l'archivio ordinato.
             * Questa operazione consiste nella copia dell'ultimo riferimento a
             * film direttamente nella posizione del film che si vuole
             * eliminare e nella successiva eliminazione dell'ultimo elemento
             * valido del vettore:
             */
            /*
            filmdb[idx] = filmdb[nFilm - 1];
            noldb[idx] = noldb[nFilm - 1];
            filmdb[nFilm - 1] = null;
            noldb[nFilm - 1] = null;
            */
        }
        return ret;
    }

    public static void main(String[] args) {
        Film videoteca[] = new Film[MAX_FILM];
        Noleggio noleggi[] = new Noleggio[MAX_FILM];

        String[] tmp = new String[] { "Roberto Benigni", "Carlo Monni" };
        videoteca[0] = new Film(0, "Berlinguer ti voglio bene",
                "Giuseppe Bertolucci", 1977, tmp);
        /*
         * Si può evitare di usare la variabile tmp usando la new direttamente
         * nei parametri del costruttore:
         */
        /*
        videoteca[0] = new Film(3, "Berlinguer ti voglio bene",
                "Giuseppe Bertolucci", 1977,
                new String[] { "Roberto Benigni", "Carlo Monni" });
        */

        tmp = new String[] { "Roberto Benigni", "Nicoletta Braschi" };
        videoteca[1] = new Film(1, "Johnny Stecchino", "Roberto Benigni", 1991,
                tmp);

        tmp = new String[] { "Roberto Benigni", "Nicoletta Braschi" };
        videoteca[2] = new Film(2, "Il mostro", "Roberto Benigni", 1994, tmp);

        tmp = new String[] { "Massimo Troisi", "Roberto Benigni" };
        videoteca[3] = new Film(3, "Non ci resta che piangere",
                "Troisi e Benigni", 1985, tmp);

        noleggi[1] = new Noleggio("Gianni Verdi", 20, 7, 2011);
        noleggi[2] = new Noleggio("Mario Rossi", 6, 7, 2011);
        noleggi[3] = new Noleggio("Mario Rossi", 22, 6, 2011);

        System.out.println("Archivio film:");
        stampaArchivio(videoteca, noleggi);

        ordinaData(videoteca, noleggi);
        System.out.println();
        System.out.println("Archivio dopo l'ordinamento:");
        stampaArchivio(videoteca, noleggi);

        Film[] ris = trovaFilm(videoteca, "Roberto Benigni",
                "Nicoletta Braschi");
        System.out.println();
        System.out.println("Film dove ha recitato Roberto Benigni"
                + " ma non Nicoletta Braschi:");
        for (int i = 0; i < ris.length; i++) {
            System.out.println(ris[i].titolo);
        }

        Film[] ris2 = noleggiati(videoteca, noleggi);
        System.out.println();
        if (ris2 == null) {
            System.out.println("Non ci sono film noleggiati.");
```

```
        } else {
            System.out.println("Film attualmente noleggiati:");
            for (int i = 0; i < ris2.length; i++) {
                System.out.println(ris2[i].titolo);
            }
        }

        boolean ret = eliminaFilm(videoteca, noleggi, 2);
        System.out.println();
        if (!ret) {
            System.out.println("Il codice non è stato trovato.");
        } else {
            System.out.println("Film eliminato con successo.");
            System.out.println("Archivio dopo l'eliminazione:");
            stampaArchivio(videoteca, noleggi);
        }
    }

}
```