Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2016/2017
3 Luglio 2017

Testo

Il database del sistema di gestione delle prenotazioni dei voli di una compagnia aerea è costituito da due vettori paralleli. Il primo è denominato “flights” e contiene oggetti di tipo “Volo” che rappresentano i voli presenti all'interno della banca dati della compagnia. Il secondo vettore è denominato “reservations” e contiene oggetti di tipo “Prenotazione” che rappresentano le informazioni relative ad ogni prenotazione effettuata relativa ad un determinato volo. Ad ogni volo può corrispondere più di una prenotazione, in quel caso le informazioni del volo saranno replicate.
Per ogni volo presente nella posizione i-esima del vettore “flights”, le informazioni relative ad una prenotazione si troveranno nella corrispondente posizione del vettore “reservations”. Nel caso in cui il volo in posizione i-esima non abbia alcuna prenotazione associata, nella posizione corrispondente nel vettore “reservations” sarà presente un riferimento *null*. Entrambi i vettori hanno dimensione pari alla costante “MAX_ELEM” (inizializzata a 1024). Se il numero dei voli contenuti nell’archivio è inferiore a “MAX_ELEM”, i primi elementi del vettore conterranno gli oggetti di tipo “Volo”, mentre gli altri conterranno riferimenti *null*. Tutti gli elementi *null* del vettore “flights” si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Volo contiene le informazioni relative ad un volo:

```
public class Volo {

        public int capienza;          public String data;
        public String partenza;       public String arrivo;

        public Volo(int capacity, String date, String departure, String arrival) {
                capacita = capacity;   data = date;
                partenza = departure;  arrivo = arrival;
        }

        public String getCodice() {
                return "" + partenza.charAt(0) + arrivo.charAt(0) +
                        data.substring(8) + data.substring(3, 5) + data.substring(0, 2);
        }

        public String toString() {
                return  ("#" + getCodice() + ": " + data + " - " +
                        partenza + " - " + arrivo + " ( " + capienza + " )");
        }
}
```

La classe Prenotazione contiene le informazioni relative alle singole prenotazioni relative ad un determinato volo.

```
public class Prenotazione {

        private static int numeroProgressivo = 0;
        private int numero;           public String data;    public String passeggero;
        public double prezzo;         public int posto;

        public Prenotazione(String date, String passenger, double price, int sit) {
                data = date;          passeggero = passenger;
                prezzo = price;       posto = sit;           numero = ++numeroProgressivo;
        }

        public Prenotazione(int number, String date, String passenger, double price, int sit) {
                data = date;          passeggero = passenger;
                prezzo = price;       posto = sit;           numero = number;
        }

        public int getNum() {  return numero; }

        public void print() {  System.out.println("(" + data + ") " + passeggero);  }

        public String toString() {
                return ( "[" + numero + "] " + data +
                        " - " + prezzo + " € (" + posto + ")");
        }

}
```

Si consiglia di procedere implementando un metodo e successivamente la parte del main che utilizza tale metodo. Le varie operazioni devono essere eseguite sulla porzione significativa dell'archivio, cioè la porzione di "flights" che non contiene riferimenti *null*. Se si ha la necessità di convertire una stringa in intero, si può utilizzare la funzione di libreria *Integer.parseInt(s)* che converte la stringa s in un intero restituito come risultato.

A. Scrivere il metodo statico:

```
public static int contaPostiDisponibili(
                        Volo[] voli, Prenotazione[] prenotazioni, String codice)
```

Il metodo deve contare il numero totale dei posti disponibili su un determinato volo identificato dal codice passato come parametro.

B. Scrivere il metodo statico:

```
public static void ordinaPrenotazioni(Volo[] voli, Prenotazione[] prenotazioni)
```

Il metodo deve ordinare, nel vettore "reservations", gli elementi in maniera decrescente usando come criterio il numero di prenotazione del volo. I voli senza prenotazione vanno posti in cima all'array parallelo.

C. Scrivere il metodo statico:

```
public static Prenotazione[] getPrenotazioni(
                        Volo[] voli, Prenotazione[] prenotazioni, String codice)
```

Il metodo deve tornare un nuovo array contenente la copia di tutte le prenotazioni relative ad un determinato volo identificato dal codice passato come parametro. Se il volo non esiste, il metodo deve tornare *null* mentre se il volo non ha nessuna prenotazione, il metodo deve tornare un array di lunghezza 0.

D. Scrivere il metodo statico:

```
public static boolean rimuoviVoli(
                        Volo[] voli, Prenotazione[] prenotazioni, String data)
```

Il metodo deve eliminare dal database, specificato dai parametri "voli" e "prenotazioni", tutti i voli, e le prenotazioni associate, effettuati in una determinata data passata come parametro; restituire **true** o **false** a seconda del fatto che sia stato eliminato almeno un volo e mantenere l'archivio in uno stato consistente.

E. Scrivere il metodo main che:

definisca ed inizializzi i vettori "flights" e "reservations" secondo i valori in tabella inizializzando a *null* o a *zero* i campi non presenti. La stampa dell'archivio consiste nello stampare le informazioni di ogni volo e delle relative prenotazioni (se ve ne sono). Si utilizzino correttamente i relativi metodi *toString()* implementati nelle due classi.

Codice	**Data** (gg/mm/aaaa)	**Partenza**	**Arrivo**	**Capacità**	**Nr.**	**Data** (gg/mm/aaaa)	**Prezzo (€)**	**Posto**
PB160101	01/01/2016	Pisa	Barcellona	200	2	31/12/2015	59,99	75
PR170110	10/01/2017	Pisa	Roma	100				
PM170320	20/03/2017	Pisa	Milano	150	5	15/02/2017	199,99	40
MA170612	12/06/2017	Milano	Amsterdam	300	6	07/03/2017	70,5	150
MA170612	12/06/2017	Milano	Amsterdam	300	7	11/06/2017	141,00	23
PB160101	01/01/2016	Pisa	Barcellona	200	1	09/09/2015	59,99	200
PM170320	20/03/2017	Pisa	Milano	150	3	20/03/2016	19,00	100
PM170320	20/03/2017	Pisa	Milano	150	4	20/03/2016	19,00	99

- Avvalendosi del metodo al punto A stampi a video, il numero totale dei posti disponibili sul volo PB160101.
- Ordini l'intero archivio utilizzando il metodo del punto B e stampi a video l'archivio prima e dopo l'ordinamento.
- Utilizzando il metodo C, stampi a video tutte le prenotazioni del volo PB160101, se ce ne sono, oppure un messaggio di errore se il volo non dovesse esistere.
- Utilizzando il metodo al punto D, rimuova tutti i voli effettuati in data in data 01/01/2016. Al termine dell'operazione si stampi l'archivio aggiornato se l'operazione è avvenuta con successo, altrimenti si stampi un messaggio di errore.