Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2016/2017
12 Giugno 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 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 contaPostiAdiacenti(Volo[] voli, Prenotazione[] prenotazioni)
```

Il metodo deve contare il numero totale delle prenotazioni che abbiano un posto adiacente prenotato sullo stesso volo.

B. Scrivere il metodo statico:

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

Il metodo deve ordinare, nel vettore "reservations", gli elementi in maniera crescente usando come criterio prima la data di prenotazione del volo e poi il posto prenotato. I voli senza prenotazione vanno posti in fondo all'array parallelo.

C. Scrivere il metodo statico:

```
public static double prezzoMedio(Volo[] voli, Prenotazione[] prenotazioni)
```

Il metodo deve calcolare il prezzo medio di un biglietto.

D. Scrivere il metodo statico:

```
public static boolean effettuaPrenotazione(
     Volo[] voli, Prenotazione[] prenotazioni, String codice, String data,
     String passeggero, double prezzo, int posto)
```

Il metodo deve inserire nel database specificato dai parametri "voli" e "prenotazioni" una nuova prenotazione (individuata dalla data, prezzo e posto) associata ad un volo che non abbia una prenotazione associata oppure duplicando il volo stesso in caso contrario. Il metodo deve restituire true o false a seconda del fatto che sia stato trovato almeno un volo associato al codice passato come parametro nel database e che l'inserimento sia possibile supponendo che vi siano posti disponibili e che il posto non sia già stato occupato. Se non esiste alcun volo associato al codice passato come parametro, l'inserimento non deve essere effettuato. L'inserimento deve mantenere l'archivio ordinato come nel punto precedente e 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 delle prenotazioni con posti adiacenti prenotati.
- 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 il prezzo medio di un biglietto.

- Utilizzando il metodo al ponto D, inserisca la prenotazione per il volo PR170110 effettuata in data 05/01/2017 per il posto 10 e pagato € 100.00. Al termine dell'operazione si stampi l'archivio aggiornato se l'operazione è avvenuta con successo, altrimenti si stampi un messaggio di errore.

Soluzione:

```
public class Esame {

	public static final int MAX_ELEM = 1024;

	public static int contaVoli(Volo[] voli){
	 int count = 0;
	 while (count < MAX_ELEM && voli[count] != null) {
		count++;
	 }
	 return count;
	}

	/*
	 * A.
	 * Il metodo deve contare il numero totale delle prenotazioni che abbiano un posto
	 * adiacente prenotato sullo stesso volo.
	 */
	public static int contaPostiAdiacenti(Volo[] voli, Prenotazione[] prenotazioni){
	 int count = 0;
	 int n = contaVoli(voli);
	 for (int i = 0; i < n; i++) {
		if (prenotazioni[i] != null) {
			boolean vicino = false;
			for (int j = 0; (j < n) && !vicino; j++) {
				vicino = (prenotazioni[j] != null) && (i != j) &&
						(voli[i].getCodice().equals(voli[j].getCodice())) &&
						(
							( prenotazioni[i].posto == (prenotazioni[j].posto
- 1)) ||
							( prenotazioni[i].posto == (prenotazioni[j].posto
+ 1))
						);
			}
			if (vicino) {
				count++;
			}
		}
```

```
 }
 return count;
}

// scambia l'elemento in posizione i-esima con quello in posizione j-esima
private static void scambiaVoli(Volo[] v, int i, int j){
 Volo tmp = v[i];
 v[i] = v[j];
 v[j] = tmp;
}

// scambia l'elemento in posizione i-esima con quello in posizione j-esima
private static void scambiaPrenotazioni(Prenotazione[] v, int i, int j){
 Prenotazione tmp = v[i];
 v[i] = v[j];
 v[j] = tmp;
}

// Converte una data in un formato intero confrontabile.
static public int dataConfrontabile(String data) {
 return Integer.parseInt(data.substring(6) + data.substring(3,5) + data.substring(0,2));
}

/* B.
 * Il metodo deve ordinare, nel vettore “reservations”, gli elementi in maniera crescente
 * usando come criterio prima la data di prenotazione del volo e poi il posto prenotato.
 * I voli senza prenotazione vanno posti in fondo all'array parallelo.
 */
public static void ordinaPrenotazioni(Volo[] voli, Prenotazione[] prenotazioni) {
 int n = contaVoli(voli);
 for (int i = 0; i < n-1; i++) {
            for (int j = i + 1; j < n; j++) {
                        if (
                                    prenotazioni[i] == null || (
                                                prenotazioni[j] != null && (
                                                            dataConfrontabile(prenotazioni[i].data) >
                                                            dataConfrontabile(prenotazioni[j].data) ||
                                                            (
                                                                        dataConfrontabile(prenotazioni[i].data) ==
                                                                        dataConfrontabile(prenotazioni[j].data) &&
                                                                        prenotazioni[i].posto >  prenotazioni[j].posto
```

```
                                        )
                                    )
                                )
                            ) {
                                scambiaVoli(voli, i, j);
                                scambiaPrenotazioni(prenotazioni, i, j);
                            }
                    }
            }
    }

/* C.
 * Il metodo deve calcolare il prezzo medio di un biglietto.
 */
public static double prezzoMedio(Volo[] voli, Prenotazione[] prenotazioni) {
 int n = contaVoli(voli);
 int biglietti = 0;
 double prezzo = 0;
 for (int i=0; i < n; i++){
            if ( prenotazioni[i] != null ) {
                        prezzo += prenotazioni[i].prezzo;
                        biglietti++;
            }
 }
 if ( biglietti > 0) {
            prezzo = prezzo / biglietti;
 }
 return prezzo;
}

/* D.
 * Il metodo deve inserire nel database specificato dai parametri “voli” e “prenotazioni”
 * una nuova prenotazione (individuata dalla data, prezzo e posto) associata ad un volo
 * che non abbia una prenotazione associata oppure duplicando il volo stesso in caso contrario.
 * Il metodo deve restituire true o false a seconda del fatto che sia stato trovato almeno un
 * volo associato al codice passato come parametro nel database. Se non esiste alcun volo
 * associato al codice passato come parametro, l'inserimento non deve essere effettuato.
 * L'inserimento deve mantenere l'archivio ordinato come nel punto precedente e in uno stato
 * consistente.
 */
public static boolean effettuaPrenotazione(Volo[] voli, Prenotazione[] prenotazioni,
```

```
String codice, String data, String passeggero, double prezzo, int posto)
{
boolean ret = false;
int n = contaVoli(voli);
if ( n < MAX_ELEM ) {
	Prenotazione p = new Prenotazione(data, passeggero, prezzo, posto);
	int flightIdx = n;
	int i = 0;

	/* Si cerca l'indice della primo volo. */
	while ( ( i < n ) && ( flightIdx == n ) ) {
		if ( voli[i].getCodice().equals(codice) ) {
			flightIdx = i;
		}
		else {
			i++;
		}
	}

	/* Volo presente? */
	if (flightIdx != n) {

		/* Inserimento */
		if (prenotazioni[flightIdx] != null) {
			voli[n] = voli[flightIdx];
			prenotazioni[n] = p;
			flightIdx = n;
		}
		else {
			prenotazioni[flightIdx] = p;
		}

		/*
		 * E' possibile utilizzare anche il metodo ordinaPrenotazioni().
		 */
		ordinaPrenotazioni(voli, prenotazioni);

		ret = true;
	}
}
return ret;
```

```
}

public static void stampaDB(Volo[] voli, Prenotazione[] prenotazioni){
 int n = contaVoli(voli);
 for (int i=0; i < n; i++){
        if (prenotazioni[i] != null){
                System.out.println(voli[i] + " -> " + prenotazioni[i]);
        }
        else {
                System.out.println(voli[i]);
        }
 }
}

public static void main(String[] args) {

 Volo[] flights = new Volo[MAX_ELEM];
 Prenotazione[] reservations = new Prenotazione[MAX_ELEM];

 flights[0] = new Volo(200, "01/01/2016", "Pisa", "Barcellona");
 flights[1] = new Volo(100, "10/01/2017", "Pisa", "Roma");
 flights[2] = new Volo(150, "20/03/2017", "Pisa", "Milano");
 flights[3] = new Volo(300, "12/06/2017", "Milano", "Amsterdam");
 flights[4] = flights[3];
 flights[5] = flights[0];
 flights[6] = flights[2];
 flights[7] = flights[2];

 reservations[5] = new Prenotazione("09/09/2015", null, 59.99, 200);
 reservations[0] = new Prenotazione("31/12/2015", null, 59.99, 75);
 reservations[6] = new Prenotazione("20/03/2016", null, 19.00, 100);
 reservations[7] = new Prenotazione("20/03/2016", null, 19.00, 99);
 reservations[2] = new Prenotazione("15/02/2017", null, 199.99, 40);
 reservations[3] = new Prenotazione("07/03/2017", null, 70.50, 150);
 reservations[4] = new Prenotazione("11/06/2017", null, 141.00, 23);

 System.out.println("\nA.");
 System.out.println("Il numero di prenotazioni con posti adiacenti è: " +
        contaPostiAdiacenti(flights, reservations));

 System.out.println("\nB.");
```

```
        System.out.println("Database PRIMA dell'ordinamento:");
        stampaDB(flights, reservations);
        ordinaPrenotazioni(flights, reservations);
        System.out.println("\nDatabase DOPO dell'ordinamento:");
        stampaDB(flights, reservations);

        System.out.println("\nC.");
        System.out.println("tl prezzo medio di un biglietto è di €: " +
                prezzoMedio(flights, reservations));

        System.out.println("\nD.");
        if (effettuaPrenotazione(flights, reservations, "PR170110", "05/01/2017", null, 100.00, 10)) {
                System.out.println("Le nuove prenotazioni sono:");
                stampaDB(flights, reservations);
        }
        else{
                System.out.println("Nessuna prenotazione effettuata.");
        }
        }

}
```