Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2014
02 Febbraio 2015

Testo

Il database di un bar è costituito da due vettori paralleli. Il primo è denominato “tables” e contiene oggetti di tipo “Tavolo” che rappresentano i tavoli presenti all'interno di un locale. Il secondo vettore è denominato “orders” e contiene oggetti di tipo “Ordine” che rappresentano le informazioni di ogni singolo ordine di un tavolo. Ad ogni tavolo può corrispondere più di un ordine, in quel caso le informazioni del tavolo saranno replicate.
Per ogni tavolo presente nella posizione i-esima del vettore “tables” si troveranno le informazioni relative ad un ordine nella corrispondente posizione del vettore “orders”. Nel caso che il tavolo in posizione i-esima non abbia alcun ordine associato, nella posizione corrispondente nel vettore “orders” sarà presente un riferimento *null*. Entrambi i vettori hanno dimensione pari alla costante “MAX_ELEM” (inizializzata a 1024). Se il numero di tavoli contenuti nell’archivio è inferiore a “MAX_ELEM”, i primi elementi del vettore conterranno gli oggetti di tipo “Tavolo”, mentre gli altri conterranno riferimenti *null*. Tutti gli elementi *null* del vettore “tables” si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Tavolo contiene le informazioni relative ad un tavolo:

```
public class Tavolo {
	private int id;
	public String nomeCliente;
	public String cognomeCliente;
	public String posizione;

	public Tavolo(int myId, String nome, String cognome, String posizione){
		this.id = myId;
		this.nomeCliente = nome;
		this.cognomeCliente = cognome;
		this.posizione = posizione;
	}

	public int getId(){
		return id;
	}

	public String toString(){
		return "Tav.#" + id + " (" + posizione + ") "
			+ nomeCliente + " " + cognomeCliente;
	}
}
```

La classe Ordine contiene le informazioni relative ai singoli ordini di un tavolo.

```
public class Ordine {

	static private int numeroProgressivo = 0;
	private int numeroOrdine;

	public String prodotto;
	public int numero;
	public double prezzo;
	public boolean servito;
	public String orario;

	public Ordine(String product, int quantity, double price, String time, boolean served){
		numeroOrdine = numeroProgressivo++;
		prodotto = product;
		numero = quantity;
		prezzo = price;
		orario =  time;
		servito = served;
	}
```

```
        public String toString(){
                return "[ord." + numeroOrdine + "] "
                        + "num. " + numero + " " + prodotto;
        }
}
```

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 "tables" che non contiene riferimenti "null".

A) Scrivere il metodo statico:

```
public static void modificaOrario(Ordine[] ordini)
```

Il metodo interpreta la stringa orario degli ordini come il numero di secondi trascorsi a partire dalla mezzanotte. Tale orario deve essere convertito nel formato hh:mm:ss (interpretato nelle 24h), andando conseguentemente ad aggiornare il campo orario degli ordini.

B) Scrivere il metodo statico:

```
public static void ordinaTavoli(Tavolo[] tavoli, Ordine[] ordini)
```

Il metodo deve ordinare, nel vettore "tables", gli elementi in maniera crescente usando come criterio criterio il numero di minuti individuati dalla parte mm dell'ordine, interpretata come numero intero. Ad esempio se l'ora è 15:14:55 (hh:mm:ss), il numero da considerare per l'ordinamento è 14. I tavoli corrispondenti ad ordini nulli si troveranno nella parte alta del vettore. Una stringa può essere convertita in intero tramite la funzione di libreria *int Integer.parseInt(String tmp).*

C) Scrivere il metodo statico:

```
public static void arrotonda(Tavolo[] tables, Ordine[] ordini)
```

Il metodo deve arrotondare il prezzo unitario per ogni ordine associato ai tavoli. L'arrotondamento deve essere effettuato sempre alla decina più vicina (es. 2.35 diventa 0.00, 6.35 diventa 10.00; ossia si arrotonda alla decina inferiore se la parte unitaria è minore di 5, a quello superiore se è maggiore o uguale).Per lo svolgimento può essere utile la funzione di libreria *double Math.floor(double d)* che restituisce l'intero inferiore di d espresso come double.

D) Scrivere il metodo statico:

```
public static boolean delete(Tavolo[] tables, Ordine[] ordini, char c, int m)
```

Il metodo deve eliminare dal db specificato da "tables" e "ordini" tutti i tavoli e gli ordini dei clienti che contengono nel cognome almeno *m* caratteri *c*, con c ed m passati come parametro. Se c non è una lettera minuscola, il metodo non deve effettuare eliminazioni. Nel conteggio dei caratteri, invece, non si deve fare distinzione tra minuscole e maiuscole (es. con c='p' ed m=3, "Pappini" deve essere eliminato). Il metodo restituisce true se è stata effettuata almeno una eliminazione, false altrimenti. Si lasci l'archivio in uno stato consistente.

E) Scrivere il metodo main che:

- definisca ed inizializzi i vettori "tables" e "orders" secondo i valori in tabella. La stampa dell'archivio consiste nello stampare le informazioni di ogni tavolo e gli ordini associati (se ve ne sono). Si utilizzino correttamente i relativi metodi toString() implementati nelle due classi. Per ogni ordine si stampi anche orario e prezzo.

Id	Nome e Cognome	Posizione	OrarioOrdine	Prodotto	Quantità	Prezzo	Servito
0	Mario Rossi	Ristopub	75366	Acqua	1	2.36	No
1	Pietro Rossi	Pizzeria					
2	Mario Ramarri	Ristopub	75900	Birra	2	3.50	Sì
3	Giovanni Verdi	Veranda	82500	Vino	2	15.62	Sì

- Avvalendosi del metodo al punto A si modifichi l'archivio e lo si stampi aggiornato.
- 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, si aggiorni l'archivio e si stampi l'archivio aggiornato.
- Utilizzando il metodo del punto D, elimini tavoli e ordini passando come parametri c='r' e m=3. Al termine di ogni operazione, se essa è avvenuta con successo si stampi l'archivio aggiornato o viceversa si stampi un messaggio di errore.