

LABORATORIO DI PROGRAMMAZIONE

Corso di Laurea Ing.
Gestionale 21/22

Ing. Antonio Luca Alfeo

luca.alfeo@ing.unipi.com



APPELLO DEL 13/06/2014

TRACCIA

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 String orarioArrivo;

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

    public int getId(){
        return id;
    }

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

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 Ordine(String product, int quantity, double price, boolean served){  
        numeroOrdine = numeroProgressivo++;  
        prodotto = product;  
        numero = quantity;  
        prezzo = price;  
        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 int contaOrdini(Ordine[] ordini)
```

Il metodo deve contare il numero di ordini effettivamente presenti nel vettore ordini.

B) Scrivere il metodo statico:

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

Il metodo deve ordinare, nel vettore “tavoli”, gli elementi secondo l'ordine di arrivo, considerando per semplicità solo l'ora di arrivo (es. chi arriva alle 9 si trova prima di chi arriva alle 11). La stringa all'interno del campo `orarioArrivo` della classe `Tavolo` può trovarsi sia nel formato `h:mm` che nel formato `hh:mm`. 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.

C) Scrivere il metodo statico:

```
public static double conto(Tavolo[] tavoli, Ordine[] ordini, int id)
```

Il metodo deve restituire il conto totale relativo al tavolo il cui id è specificato come parametro. Nel computo si devono considerare solo gli ordini effettivamente serviti.

D) Scrivere il metodo statico:

```
public static boolean eliminaOrdineOre(Tavolo[] tavoli, Ordine[] ordini, int ora)
```

Il metodo deve eliminare dal database specificato dai parametri “tavoli” e “ordini”, tutti gli ordini relativi a tavoli i cui clienti sono arrivati prima dell'ora specificata come parametro. Anche i corrispondenti tavoli devono essere eliminati, l'archivio deve rimanere in uno stato consistente.

E) Scrivere il metodo main che:

- definisca ed inizializzi i vettori “tables” e “orders” secondo i valori riportati 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.

Id	Nome e Cognome	Posizione	ArrivoCliente	Prodotto	Quantità	Costo	Servito
0	Mario Rossi	Ristopub	12:30	Acqua	1	2	Sì
1	Pietro Rossi	Pizzeria	9:50				
0	Mario Rossi	Ristopub	12:30	Birra	2	3.50	Sì
2	Gio' Verdi	Veranda	11:25	Vino	1	10	Sì
2	Gio' Verdi	Veranda	11:25	Acqua	1	2	Sì

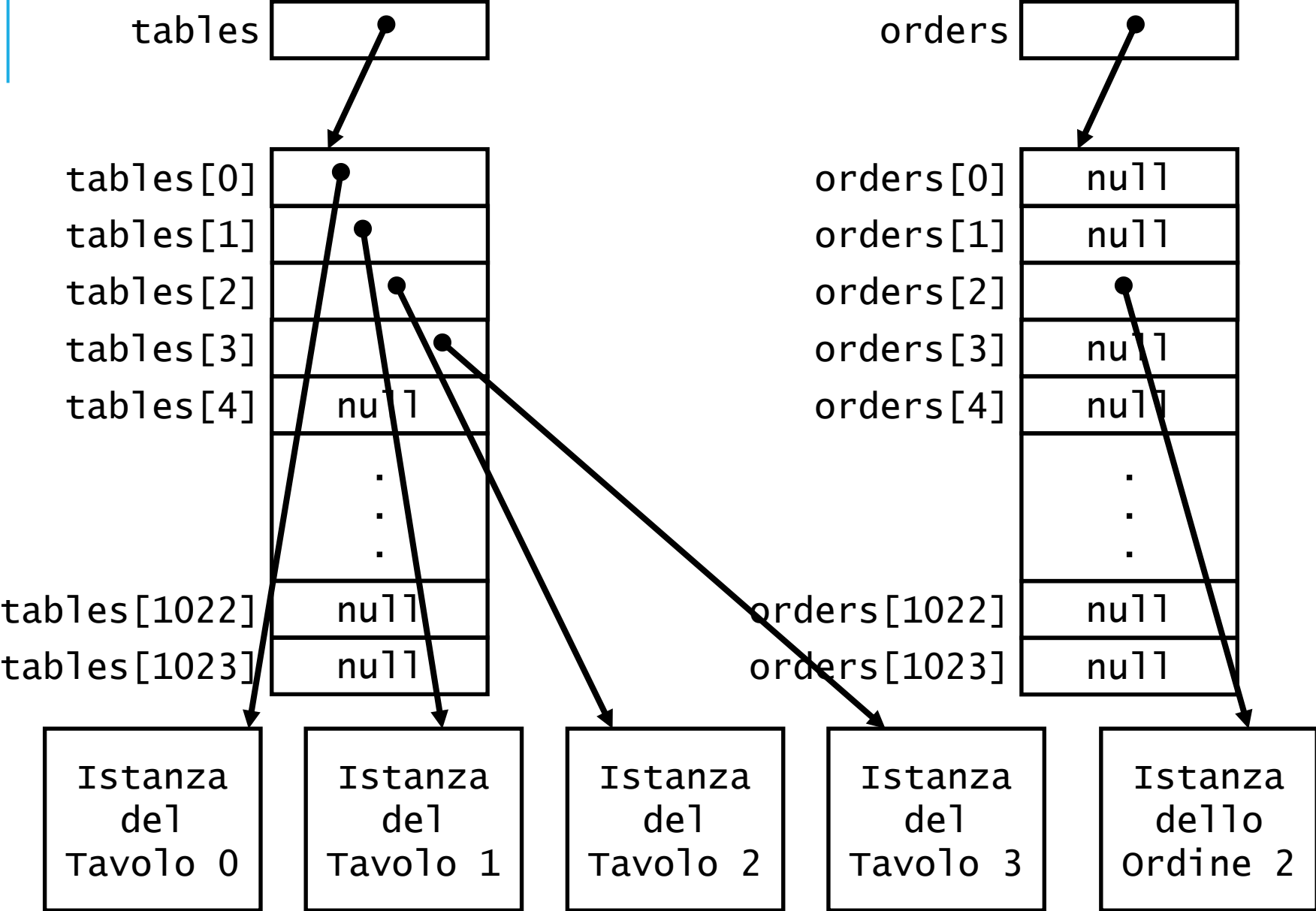
- Avvalendosi del metodo al punto A stampi il numero di ordini effettivamente presenti.
- Ordini l'intero archivio utilizzando il metodo del punto B e stampi a video l'archivio prima e dopo l'ordinamento.
- Utilizzando il metodo C, stampi le informazioni relative al Tavolo con id 0.
- Elimini gli ordini effettuati prima delle ore 12, utilizzando il metodo del punto D. Al termine dell'operazione si stampi l'archivio aggiornato se l'operazione è avvenuta con successo, altrimenti si stampi un messaggio di errore.



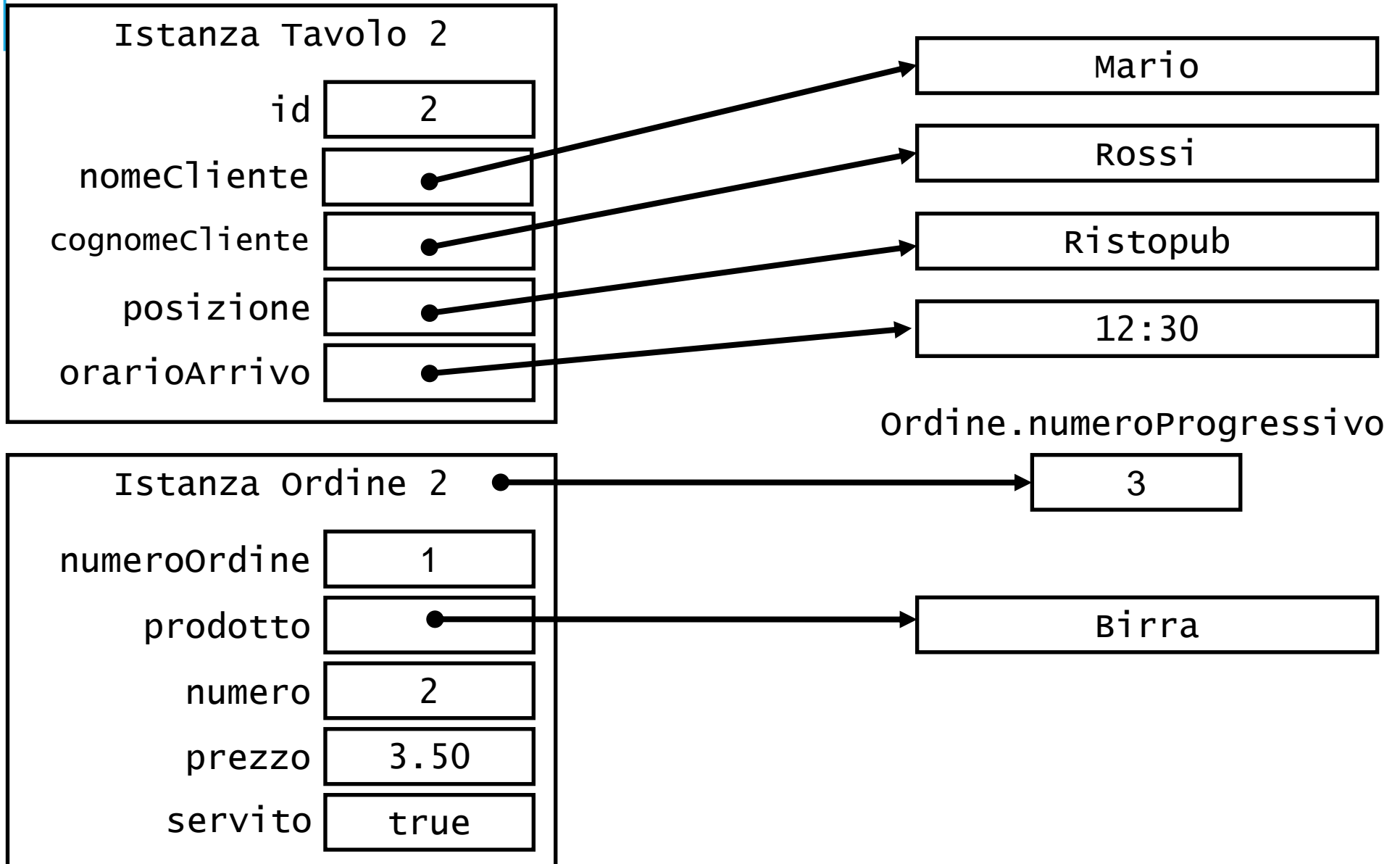
APPELLO DEL 13/06/2014

SOLUZIONE STEP BY STEP

STRUTTURA DATABASE



STRUTTURA ISTANZE DI TAVOLO ED ORDINE



INIZIALIZZAZIONE DATABASE (E)

```
public class Appello_13062014 {  
    static final int MAX_ELEM = 1024;  
    public static void main(String[] args) {  
  
        Tavolo[] tables = new Tavolo[MAX_ELEM];  
        Ordine[] orders = new Ordine[MAX_ELEM];  
  
        tables[0] = new Tavolo(0, "Mario", "Rossi", "Ristopub", "12:30");  
        tables[1] = new Tavolo(1, "Pietro", "Rossi", "Pizzeria", "9:50");  
        tables[2] = new Tavolo(0, "Mario", "Rossi", "Ristopub", "12:30");  
        tables[3] = new Tavolo(2, "Gio'", "Verdi", "Veranda", "11:25");  
        tables[4] = new Tavolo(2, "Gio'", "Verdi", "Veranda", "11:25");  
        orders[0] = new ordine("Acqua", 1, 2, true);  
        orders[2] = new ordine("Birra", 2, 3.50, true);  
        orders[3] = new ordine("Vino", 1, 10, true);  
        orders[4] = new ordine("Acqua", 1, 2, true);  
  
    }  
}
```


STAMPA DATABASE (E)

```
public static Appello_13062014 {
```

```
...
```

```
public static int contaTavoli(Tavolo[] tavoli) {  
    int count = 0;  
    while (count < MAX_ELEM && tavoli[count] != null) {  
        count++;  
    }  
    return count;  
}
```

Evito di
ciclare su
tutti gli
elementi
a null



```
public static void stampaDB(Tavolo[] tavoli, Ordine[] ordini) {  
    int n = contaTavoli(tavoli);  
    for (int i = 0; i < n; i++) {  
        if (ordini[i] != null) {  
            System.out.println(tavoli[i] + " - " + ordini[i]);  
        }  
        else {  
            System.out.println(tavoli[i]);  
        }  
    }  
}
```

```
...
```

```
}
```

STAMPA ORDINI (A)

```
public static Appello_13062014 {  
    ...  
  
    public static int contaOrdini(Ordine[] ordini) {  
        int count = 0;  
        for (int i=0; i < MAX_ELEM; i++) {  
            if (ordini[i] != null) {  
                count++;  
            }  
        }  
        return count;  
    }  
    ...  
  
    public static void main(String[] args) {  
        ...  
  
        System.out.println("\nA.");  
        System.out.println("Il numero di ordini effettivi è " +  
        contaOrdini(orders));  
    }  
}
```

ORDINAMENTO TAVOLI (B)

```
public static Appello_13062014 {  
    ...  
  
    public static int estraiOra(String orario) {  
        int idx = orario.indexOf(':', 0);  
        String ora = orario.substring(0, idx);  
        return Integer.parseInt(ora);  
    }  
  
    ...  
  
    public static void scambiaOrdine(Ordine[] v, int i, int j) {  
        Ordine tmp = v[i];  
        v[i] = v[j];  
        v[j] = tmp;  
    }  
  
    ...  
  
    public static void scambiaTavolo(Tavolo[] v, int i, int j) {  
        Tavolo tmp = v[i];  
        v[i] = v[j];  
        v[j] = tmp;  
    }  
  
    ...  
  
}
```

ORDINAMENTO TAVOLI (B)

```
public static Appello_13062014 {  
  
    ...  
  
    public static void ordinaArrivo(Tavolo[] tavoli, Ordine[] ordini) {  
        int n = contaTavoli(tavoli);  
        for (int i=0; i < n-1; i++) {  
            int min = i;  
            for (int j=i+1; j < n; j++) {  
                if (estraiOra(tavoli[j].orarioArrivo) <  
estraiOra(tavoli[min].orarioArrivo) {  
                    min = j;  
                }  
            }  
            scambiaTavolo(tavoli, i, min);  
            scambiaOrdine(ordini, i, min);  
        }  
    }  
  
    ...  
  
    public static void main(String[] args) {  
        ...  
        System.out.println("\nB.");  
        System.out.println("\nPrima dell'ordinamento:");  
        stampaDB(tables, orders);  
        System.out.println("\nDopo l'ordinamento:");  
        ordinaArrivo(tables, orders);  
        stampaDB(tables, orders);  
    }  
}
```

CONTO TAVOLO 0 (C)

```
public static Appello_13062014 {  
    ...  
  
    public static double conto(Tavolo[] tavoli, Ordine[] ordini, int id) {  
        int n = contaTavoli(tavoli);  
        double tot = 0;  
        for (int i=0; i < n; i++) {  
            if (tavoli[i].getId() == id) {  
                if (ordini[i] != null && ordini[i].servito) {  
                    tot += ordini[i].prezzo * ordini[i].numero;  
                }  
            }  
        }  
        return tot;  
    }  
}  
  
    ...  
  
    public static void main(String[] args) {  
        ...  
  
        System.out.println("\nC.");  
        double totTav0 = conto(tables, orders, 0);  
        System.out.println("\nIl Conto del Tavolo #0 è " + totTav0 + "€");  
    }  
}
```

ELIMINAZIONE ORDINI (D)

```
public static Appello_13062014 {
    ...
    public static boolean eliminaOrdineOra(Tavolo[] tavoli,
                                           Ordine[] ordini, int ora) {
        int n = contaTavoli(tavoli);
        int ordDel = 0;
        for (int i = 0; i < n; i++) {
            if (estraiOra(tavoli[i].orarioArrivo) < ora) {
                tavoli[i] = null;
                ordini[i] = null;
                ordDel++;
            }
        }
        if (ordDel > 0) {
            // bisogna compattare!!!
            for (int i=0; i < n-1; i++) {
                for (int j=i+1; j < n; j++) {
                    if (tavoli[i] == null && tavoli[j] != null) {
                        scambiaTavolo(tavoli, i, j);
                        scambiaOrdine(ordini, i, j);
                    }
                }
            }
            return true;
        }
        else
            return false;
    }
    ...
}
```


ELIMINAZIONE ORDINI (D)

```
public static Appello_13062014 {  
  
    ...  
  
    public static void main(String[] args) {  
  
        ...  
  
        System.out.println("\nD.");  
        if (eliminaOrdineOre(tables, orders, 12)) {  
            System.out.println("\nGli ordini rimanenti sono:");  
            stampaDB(tables, orders);  
        }  
        else {  
            System.out.println("\nNessuna eliminazione.");  
        }  
    }  
}
```