Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale

Esame di Informatica - a.a. 2012-13

04 febbraio 2014

Testo

Il database di una banca è costituito da due vettori paralleli. Il primo è denominato “clienti” e contiene oggetti di tipo “Correntista” che rappresentano i correntisti presenti nell’archivio della banca. Il secondo vettore è denominato “conti” e contiene oggetti di tipo “Conto_corrente” che rappresentano le informazioni di un conto corrente di un cliente. Ogni cliente può avere più di un conto corrente nella banca, in quel caso le informazioni del cliente saranno replicate.

Per ogni correntista presente nella posizione i-esima del vettore “clienti” si troveranno le informazioni relative al suo conto corrente nella corrispondente posizione del vettore “conti”. Nel caso che il correntista in posizione i-esima non abbia alcun conto aperto, nella sua posizione nel vettore “conti” sarà presente un riferimento *null*. Entrambi i vettori hanno dimensione pari alla costante “MAX_CLIENTI” (inizializzata a 1024). Se il numero di correntisti contenuti nell’archivio è inferiore a “MAX_CLIENTI”, i primi elementi del vettore conterranno gli oggetti di tipo “Correntista”, mentre gli altri conterranno riferimenti *null*. Tutti gli elementi *null* del vettore “clienti” si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Correntista contiene le informazioni relative ad un correntista ed un metodo per stampare queste informazioni:

```
public class Correntista {
	public int id;
	public String nome;
	public String cognome;
	public int annoNascita;
	public int meseNascita;
	public int giornoNascita;

	public Correntista(int id, String nome, String cognome, int giornoNascita,
			int meseNascita, int annoNascita) {
		this.id = id;
		this.nome = nome;
		this.cognome = cognome;
		this.giornoNascita = giornoNascita;
		this.meseNascita = meseNascita;
		this.annoNascita = annoNascita;
	}

	public String toString(){
		return "[" + this.id + "] " + this.nome + " " + this.cognome + " \t"
		    + this.giornoNascita + "/" + this.meseNascita + "/" + this.annoNascita;
	}
}
```

La classe Conto_corrente contiene le informazioni relative al conto corrente di un correntista.

```
public class Conto_corrente {
	static private int numeroProgressivo = 0;
	private int numeroConto;
	public double saldo;
	public int annoApertura;
	boolean mutuo;
	public String filiale;

	public Conto_corrente(double saldo, int annoApertura, String filiale, boolean mutuo) {
		this.numeroConto = numeroProgressivo++;
		this.saldo = saldo;
		this.annoApertura = annoApertura;
		this.filiale = filiale;
		this.mutuo = mutuo;
	}

	public String toString() {
		return Integer.toString(numeroConto);
	}
}
```

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 "clienti" che non contiene riferimenti "null".

A) **Scrivere il metodo statico:**

```
private static String riempi(String s, int MAX)
```

Il metodo deve restituire una stringa di dimensione MAX, riempiendo con spazi la stringa di partenza "s". Se la lunghezza di s è maggiore di MAX, si deve effettuare il troncamento di "s" e restituire la stringa troncata.

B) **Scrivere il metodo statico:**

```
public static void ordinaClienti(Correntista[] clients, Conto_corrente[] cc_db)
```

Il metodo deve ordinare nel vettore "clients" gli elementi in maniera decrescente, usando come criterio il numero delle consonanti che compongono il cognome del correntista. L'intero archivio deve essere lasciato in uno stato consistente.

C) **Scrivere il metodo statico:**

```
public static void stampaSaldi(Conto_corrente[] cc_db)
```

Il metodo deve stampare i saldi dei conti presenti nell'array "cc_db". Se una riga corrisponde ad un riferimento null non deve essere stampato alcunché. Se il saldo è negativo deve essere stampata la stringa <*value error*>; altrimenti deve essere stampato il saldo arrotondato alla seconda cifra decimale. L'arrotondamento deve essere effettuato sempre all'intero più vicino (quello inferiore se la parte decimale è minore di 0.5, quello superiore se è maggiore o uguale). Si suggerisce di utilizzare come supporto per lo svolgimento il metodo di libreria `double Math.floor(double d)` che restituisce l'intero inferiore di `d` espresso come double.

D) **Scrivere il metodo statico:**

```
public static void printClientiSenzaMutuo(Correntista[] clients, Conto_corrente[] cc_db)
```

Il metodo deve ricercare i clienti che non hanno nessun mutuo aperto presso nessuna filiale e stampare le informazioni relative al cliente come se fossero in una tabella, in cui ogni colonna ha lunghezza fissa pari a 12 caratteri. Le colonne da stampare sono tre: la prima contiene l'id, la seconda nome e cognome separati da spazio, la terza contine la data di nascita nella forma gg/mm/anno.

E) **Scrivere il metodo main che:**

- definisca ed inizializzi i vettori "clienti" e "conti" secondo i valori riportati in tabella e stampi a video l'archivio. La stampa dell'archivio consiste nel stampare le informazioni di ogni cliente (usando il metodo "toString" della classe "Correntista") e, se il correntista possiede un conto, devono essere indicati anche filiale e saldo arrotondato.

Id	Nome e Cognome	Data di Nascita	Saldo	Anno Apertura	Filiale	Mutuo
0	Maria Rossini	15/10/1954	10,523	1999	Roma1	No
1	Piero Ars	15/10/1958				
0	Maria Rossini	15/10/1954	12,2	2009	Roma1	Sì
2	Giovanni Verdi	22/08/1980	-10,3	2007	Milano1	No

- Stampi lo stato iniziale del'archivio.
- Utilizzando il metodo del punto B ordini i correntisti e stampi a video la lista ordinata.
- Utilizzando il metodo C, stampi i saldi di tutti i conti presenti.
- Utilizzando il metodo D, stampi i clienti che non hanno nessun mutuo presso nessuna filiale.