Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale

Esame di Informatica - a.a. 2012-13

04 luglio 2013

## *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:**

```
public static int numeroConti(Conto_corrente[] cc_db, String fil)
```

Il metodo deve restituire, per il vettore "cc_db", il numero di conti presenti nella filiale specificata dal parametro "fil".

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 secondo l'iniziale del cognome del correntista. Il vettore "cc_db" deve inoltre rimanere consistente.

C) **Scrivere il metodo statico:**

```
public static Correntista[] correntistiFiliale(Correntista[] clients,
                                               Conto_corrente[] cc_db, String fil)
```

Il metodo deve restituire un nuovo array contenente solo i correntisti che hanno un conto presso la filiale specificata dal parametro "fil". Se un correntista ha più conti, deve comparire una volta sola. Se nel vettore "clients" non vi sono correntisti che soddisfano il criterio, il metodo deve restituire un riferimento a "null".

D) **Scrivere il metodo statico:**

```
public static int eliminaContiSaldoNullo(Correntista[] clients, Conto_corrente[] cc_db)
```

Il metodo deve eliminare dal database specificato dai parametri "clients" e "cc_db" tutti i conti a saldo nullo. Anche i rispettivi correntisti devono essere eliminati (lasciando l'archivio in uno stato consistente). Il metodo deve inoltre restituire il numero di eliminazioni effettuate.

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, deve essere indicato anche il numero di conto (solo quello).

| Id | Nome e Cognome | Data di Nascita | Saldo | Anno Apertura | Filiale | Mutuo |
|---|---|---|---|---|---|---|
| 0 | Mario Rossi | 15/10/1954 | 0 | 1999 | Roma1 | No |
| 1 | Piero Rossi | 15/10/1958 | | | | |
| 0 | Mario Rossi | 15/10/1954 | 2500 | 2009 | Roma1 | Sì |
| 2 | Giovanni Verdi | 22/08/1980 | 500 | 2007 | Milano1 | Sì |

- Stampi un messaggio che indichi il numero di conti della filiale "Roma1", utilizzando il metodo del punto A.
- 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 nome e cognome dei correntisti che hanno un conto presso la filiale "Milano1". Se nessun correntista dovesse soddisfare il criterio, si deve stampare un messaggio di errore.
- Elimini i conti corrente come indicato nel punto D e stampi un messaggio che indichi il numero di conti eliminati. Se almeno uno è stato eliminato viene inoltre stampato l'archivio aggiornato.

## Soluzione

```
public class AppelloEsame {
	static final int MAX_CLIENTI = 1024;

	/* A. Il metodo deve restituire, per il vettore “cc_db”, il numero di conti
	 * presenti nella filiale specificata dal parametro “fil”.
	 */
	public static int numeroConti(Conto_corrente[] cc_db, String fil){
		int counter = 0;
		for (int i=0; i < MAX_CLIENTI; i++){
			if (cc_db[i] != null){
				if (cc_db[i].filiale.equals(fil)){
					counter++;
				}
			}
		}
		return counter;
	}

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

	private static void scambiaClienti(Correntista[] v, int i, int j){
		Correntista tmp = v[i];
		v[i] = v[j];
		v[j] = tmp;
	}

	private static int countCorrentisti(Correntista[] clients){
		int counter = 0;
		while (counter < MAX_CLIENTI && clients[counter]!=null){
			counter++;
		}
		return counter;
	}

	/* B. Il metodo deve ordinare, nel vettore “clients”, gli elementi in maniera
	 * decrescente secondo l’iniziale del cognome del correntista. Il vettore “cc_db”
deve inoltre rimanere consistente.
	 */
	public static void ordinaClienti(Correntista[] clients, Conto_corrente[] cc_db){
	     int n = countCorrentisti(clients);
	     for (int i = 0; i < n-1; i++){
	        int max = i;
	        for (int j = i+1; j < n; j++){
	             if (clients[j].cognome.charAt(0) > clients[max].cognome.charAt(0)) {
	                    max = j;
	             }
	        }
	      scambiaClienti(clients, i, max);
	      scambiaConti(cc_db, i, max);
	      }
	}
```

```
	public static void ordinaClienti2(Correntista[] clients, Conto_corrente[] cc_db){
		int n = countCorrentisti(clients);
		for (int i = 0; i < n-1; i++){
		   for (int j = i+1; j < n; j++){
		      if (clients[j].cognome.charAt(0) > clients[i].cognome.charAt(0)) {
		         scambiaClienti(clients, i, j);
		         scambiaConti(cc_db, i, j);
		      }
		  }

		}
	}


	private static int contaDuplicati(Correntista[] c){
		// per costruzione assumo che non ci siano riferimenti null
		int num_dup = 0;
		int n = c.length;
		for (int i = 0; i < n-1; i++){
			boolean duplicato=false;
			for (int j = i+1; j < n && !duplicato ; j++){
				if (c[i].id == c[j].id){
					num_dup++;
					duplicato=true;
				}
			}
		}
		return num_dup;
	}

	/* C. Il metodo deve restituire un nuovo array contenente solo i correntisti
che hanno un conto presso la filiale
	 * specificata dal parametro “fil”. Se un correntista ha più conti, deve comparire
una volta sola. Se nel vettore
	 * “clients” non vi sono correntisti che soddisfano il criterio, il metodo deve
restituire un riferimento a “null”.
	 */

	public static Correntista[] correntistiFiliale(Correntista[] clients,
							Conto_corrente[] cc_db, String fil){
		Correntista[] ris_dup = null;
		Correntista[] ris = null;
		int num_clienti = countCorrentisti(clients);
		int num = numeroConti(cc_db, fil);
		if (num != 0){
			// costruisco il vettore con i duplicati
			int k = 0;
			ris_dup = new Correntista[num];
			for (int i=0; i < num_clienti; i++){
				if (cc_db[i] != null){
					if (cc_db[i].filiale.equals(fil)){
						ris_dup[k] = clients[i];
						k++;
					}
				}
			}

			/* costruisco il vettore senza duplicati a partire
			   da quello con i duplicati */
			int num_dup = contaDuplicati(ris_dup);
			int size = ris_dup.length - num_dup;
			ris = new Correntista[size];
			k = 0;
			for (int i=0; i<ris_dup.length; i++){
				boolean trovato = false;
				for (int j=0; j < ris.length; j++){
```

```
                                if (ris[j] != null && ris_dup[i].id == ris[j].id)
                                        trovato = true;
                        }
                        if (!trovato) {
                                ris[k] = ris_dup[i];
                                k++;
                        }
                }
        }
        return ris;
}

/*
 * D. Il metodo deve eliminare dal database specificato dai parametri “clients” e
“cc_db” tutti i conti a saldo nullo.
 * Anche i rispettivi correntisti devono essere eliminati (lasciando l'archivio in
uno stato consistente). Il metodo deve
 * inoltre restituire il numero di eliminazioni effettuate.
 */
public static int eliminaContoSaldoNullo(Correntista[] clients,
                                         Conto_corrente[] cc_db){
        int n = countCorrentisti(clients);
        int elim = 0;

        for (int i=0; i < n; i++){
                if (cc_db[i]== null || cc_db[i].saldo == 0.0){
                        elim++;
                        cc_db[i] = null;
                        clients[i] = null;
                }
        }

        for (int i=0; i < n-1; i++){
                for (int j=i+1; j< n; j++){
                        if (clients[i]== null && clients[j]!= null){
                                scambiaClienti(clients, i, j);
                                scambiaConti(cc_db, i, j);
                        }
                }
        }
        return elim;
}

public static void stampaDB(Correntista[] clients, Conto_corrente[] cc_db){
        int n = countCorrentisti(clients);
        for (int i=0; i < n; i++){
                if (cc_db[i] != null){
                        System.out.println(clients[i] + " - cc: " + cc_db[i]);
                }
                else {
                        System.out.println(clients[i]);
                }
        }
}

public static void main(String[] args) {
        Correntista[] clienti = new Correntista[MAX_CLIENTI];
        Conto_corrente[] conti = new Conto_corrente[MAX_CLIENTI];

        clienti[0] = new Correntista(0, "Mario", "Rossi", 15, 10, 1954);
        clienti[1] = new Correntista(1, "Piero", "Rossi", 15, 10, 1958);
        clienti[2] = new Correntista(0, "Mario", "Rossi", 15, 10, 1954);
        clienti[3] = new Correntista(2, "Giovanni", "Verdi", 22, 8, 1980);

        conti[0] = new Conto_corrente(0, 1999, "Roma1", false);
        conti[2] = new Conto_corrente(2500, 2009, "Roma1", true);
        conti[3] = new Conto_corrente(500, 2007, "Milano1", true);
```

```
            String filiale = "Roma1";
            System.out.println("\nA.");
            System.out.println("Il numero dei conti per la filiale " + filiale + " è "
                               + numeroConti(conti, filiale));

            System.out.println("\nB.");
            System.out.println("L'archivio prima dell'ordinamento");
            stampaDB(clienti, conti);
            ordinaClienti2(clienti, conti);
            System.out.println("\nL'archivio dopo l'ordinamento");
            stampaDB(clienti, conti);

            System.out.println("\nC.");
            filiale = "Milano1";
            Correntista[] c = correntistiFiliale(clienti, conti, filiale);
            if (c == null){
                 System.out.println("Errore, nessun correntista trovato per la filiale
                                    " + filiale);
            }
            else {
                   for (int i =0; i< c.length; i++)
                         System.out.println(c[i].nome + " " + c[i].cognome);
            }

            System.out.println("\nD.");
            int num = eliminaContoSaldoNullo(clienti, conti);
            System.out.println("Il numero di eliminazioni è stato: " + num);
            if (num > 1){
                   stampaDB(clienti, conti);
            }
      }

}
```