Cognome _____Nome _____
Matricola _____Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2015-16
18 Febbraio 2016

Testo

Il database del sistema di gestione delle chiamate di uno Smart-Phone è costituito da due vettori paralleli. Il primo è denominato “contacts” e contiene oggetti di tipo “Contatto” che rappresentano i contatti presenti all'interno della rubrica dello Smart-Phone. Il secondo vettore è denominato “calls” e contiene oggetti di tipo “Chiamata” che rappresentano le informazioni relative ad ogni chiamata effettuata ad un determinato contatto. Ad ogni contatto può corrispondere più di una chamata, in quel caso le informazioni del contatto saranno replicate.
Per ogni contatto presente nella posizione i-esima del vettore “contacts”, le informazioni relative ad una chiamata si troveranno nella corrispondente posizione del vettore “calls”. Nel caso in cui il contatto in posizione i-esima non abbia alcuna chiamata associata, nella posizione corrispondente nel vettore “calls” sarà presente un riferimento *null*. Entrambi i vettori hanno dimensione pari alla costante “MAX_ELEM” (inizializzata a 1024). Se il numero di contatti contenuti nell’archivio è inferiore a “MAX_ELEM”, i primi elementi del vettore conterranno gli oggetti di tipo “Contatto”, mentre gli altri conterranno riferimenti *null*. Tutti gli elementi *null* del vettore “contacts” si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Contatto contiene le informazioni relative ad un contatto:

```
public class Contatto {
        private static int idProgressivo = 0;
        private int id;
        public String nome;
        public String cognome;
        public String indirizzo;
        public String citta;
        public String cap;
        public String telefono;

        public Contatto(int myId, String name, String surname,
                                String address, String city,       String zipcode, String phone) {
                id = myId;        nome = name;      cognome = surname;         indirizzo = address;
                citta = city;     cap = zipcode;    telefono = phone;
        }

        public int getId() {
                return id;
        }

        public String toString() {
                return   "# " + id + ": " + telefono + " - " + nome + " " + cognome + " - " +
        indirizzo + " - " + citta + " - " + cap;
        }
}
```

La classe Chamata contiene le informazioni relative alle singole chiamate ad un contatto.

```
public class Chiamata {
        private static int numeroProgressivo = 0;
        private int numero;
        boolean ricevuta;
        public String data;
        public String ora;
        public String durata;

        public Chiamata(boolean incoming, String date, String hour, String duration) {
                numero = numeroProgressivo++;       ricevuta = incoming;
                data = date;      ora = hour;       durata = duration;
        }

        public int getNumero() {
                return numero;
        }

        public String toString() {
                return   ((ricevuta)? "<< ": ">> ") + data + " - " + ora + " - " + durata;
```

```
        }
}
```

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 "contacts" 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 contaChiamate(Contatto[] contatti, Chiamata[] chiamate, String begin, String end)
```

Il metodo deve contare il numero totale delle chiamate ricevute in un determinato periodo compreso tra una data di inizio ed una di fine, estremi inclusi. Le due date sono specificate per mezzo di 2 stringhe, begin ed end, che rispettano il seguente formato: "GG/MM/AAAA".

B. Scrivere il metodo statico:

```
public static void ordinaContatti(Contatto[] contatti, Chiamata[] chiamate)
```

Il metodo deve ordinare, nel vettore "contacts", gli elementi in maniera decrescente usando come criterio la data e l'ora della chiamata corrispondente ponendo prima le chiamate effettuate, poi quelle ricevute ed in fine le chiamate nulle.
Il metodo deve mantenere la corrispondenza iniziale tra contatti e chiamate.

C. Scrivere il metodo statico:

```
public static int trafficoResiduo(Contatto[] contatti, Chiamata[] chiamate, int mese, int anno)
```

Supponendo un contratto con una franchigia di 100 minuti di traffico voce mensili, il metodo deve calcolare i minuti di traffico voce residuo per un determinato mese di un determinato anno arrotondando, per ogni chiamata, i secondi maggiori o uguali a 30 al minuto successivo.

D. Scrivere il metodo statico:

```
public static boolean rimuoviChiamatePerse(Contatto[] contatti, Chiamata[] chiamate)
```

Il metodo deve eliminare dal database, specificato dai parametri "contatti" e "chiamate", tutte le chiamate "perse". Per chiamata persa si intende una chiamata ricevuta (attributo ricevuta = true, R/E = R nella tabella) con durata pari a "00:00:00". Il metodo deve restituire true o false a seconda del fatto che sia stato eliminata almeno una chiamata nel database e mantenere l'archivio in uno stato consistente rimuovendo i contatti duplicati senza chiamata associata.

E. Scrivere il metodo main che:

definisca ed inizializzi i vettori "contacts" e "calls" secondo i valori in tabella. La stampa dell'archivio consiste nello stampare le informazioni di ogni contatto e le chiamate associate (se ve ne sono). Si utilizzino correttamente i relativi metodi toString() implementati nelle due classi.

Id	**Nome e Cognome**	**Indirizzo**	**Città**	**Cap**	**Telefono**	**R/E**	**Data** (gg/mm/aaaa)	**Ora** (hh:mm)	**Durata** (hh:mm:ss)
0	Pietro Rossi	Via Roma, 31	Pisa	56127	050642687	R	01/01/2013	00:00	00:00:00
1	Giovanni Verdi	Via Pollione, 5	Chieti	66100	08712278				
2	Mario Ramarri	Via Lenin, 4	Roma	00149	0623476	E	03/02/2014	09:45	00:00:55
3	Mario Rossi	Piazza Cairoli, 3	Pisa	56124	050576904	E	02/04/2014	10:21	00:01:23
1	Giovanni Verdi	Via Pollione, 5	Chieti	66100	08712278	E	07/07/2014	15:32	01:33:45
0	Pietro Rossi	Via Roma, 31	Pisa	56127	050642687	R	09/10/2014	17:04	00:00:00

- Avvalendosi del metodo al punto A, si stampi a video il numero delle chiamate effettuate dal 01/01/2014 al 31/12/2014.
- 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 il traffico voce residuo in minuti relativo al mese di luglio 2014.
- Avvalendosi del metodo al punto D. Si rimuovano tutte le chiamate perse.

  Al termine dell'operazione si stampi a video l'archivio aggiornato se l'operazione è avvenuta con successo, altrimenti si stampi un messaggio di errore.

Soluzione

```
public class Appello_20160218 {

	public static final int MAX_ELEM = 1024;

	public static final int HOURS_START_IDX = 0;
	public static final int HOURS_END_IDX = 2;
	public static final int MINUTES_START_IDX = 3;
	public static final int MINUTES_END_IDX = 5;
	public static final int SECONDS_START_IDX = 6;
	public static final int YEAR_START_IDX = 6;

	public static int contaContatti(Contatto[] contatti){
		int count = 0;
		while (count < MAX_ELEM && contatti[count] != null) {
			count++;
		}
		return count;
	}

	/*
	 *  Metodo ausiliario per confrontare 2 date e che ritorna:
	 *  - valore = 0 se e solo se d1 è uguale a d2.
	 *  - valore < 0 se e solo se d1 è antecedente a d2.
	 *  - valore < 0 se e solo se d1 è successivo a d2.
	 */
	public static int confrontaData(String d1, String d2) {
		int ret = 0;
		int g1 = Integer.parseInt(d1.substring(0, 2));
		int m1 = Integer.parseInt(d1.substring(3, 5));
		int a1 = Integer.parseInt(d1.substring(6));
		int g2 = Integer.parseInt(d2.substring(0, 2));
		int m2 = Integer.parseInt(d2.substring(3, 5));
		int a2 = Integer.parseInt(d2.substring(6));

		if (a1 != a2) {
			ret = a1 - a2;
		}
		else if (m1 != m2) {
			ret = m1 - m2;
		}
		else {
			ret = g1 - g2;
		}

		return ret;
	}

	/*
	 * A.
	 * Il metodo deve contare il numero totale delle chiamate ricevute
in un determinato periodo compreso
	 * tra una data di inizio ed una di fine, estremi inclusi.
	 * Le due date sono specificate per mezzo di 2 stringhe, begin ed
end, che rispettano il seguente formato:
	 * "GG/MM/AAAA".
	 */
```

```
	public static int contaChiamate(Contatto[] contatti, Chiamata[]
chiamate, String begin, String end) {
		int count = 0;
		int n = contaContatti(contatti);
		for (int i=0; i<n; i++) {
			if (
					chiamate[i] != null && !chiamate[i].ricevuta
&&
					!chiamate[i].durata.equals("00:00:00") &&
					(confrontaData(begin, chiamate[i].data) <= 0)
&&
					(confrontaData(chiamate[i].data, end) <= 0)
			) {
				count++;
			}
		}
		return count;
	}

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

	/* B.
	 * Il metodo deve ordinare, nel vettore “contacts”, gli elementi in
maniera decrescente,
	 * usando come criterio la data e l'ora della chiamata corrispon-
dente ponendo prima le
	 * chiamate effettuate, poi quelle ricevute ed in fine le chiamate
nulle.
	 * Il metodo deve mantenere la corrispondenza iniziale tra contatti
e chiamate.
	 */
	public static void ordinaContatti(Contatto[] contatti, Chiamata[]
chiamate) {
		int n = contaContatti(contatti);
		for (int i = 0; i < n-1; i++) {
			for (int j=i+1; j < n; j++) {
				if (
						chiamate[i] == null ||
						(
							chiamate[j] != null &&
							(
								(
									(!chiamate[j].ricevuta
&& chiamate[i].ricevuta) ||
```

```
                                                (chiamate[j].ricevuta
== chiamate[i].ricevuta) &&
                                                (
                                                      (confrontaDa-
ta(chiamate[i].data, chiamate[j].data) < 0) ||
                                                      (
                                                            chiama-
te[i].data.equals(chiamate[j].data) &&
                                                            (chia-
mate[i].ora.compareTo(chiamate[j].ora) < 0)
                                                      )
                                                )
                                          )
                                    )
                              )
                        ) {
                              scambiaContatti(contatti, i, j);
                              scambiaChiamate(chiamate, i, j);
                        }
                  }
            }
      }

      /*
       * Metodo ausiliaro che restituisce la durata in minuti arrotondan-
do i secondi
       * maggiori o uguali a 30 al minuto successivo.
       */
      public static int convertiDurata(String duration) {
            String ora = duration.substring(HOURS_START_IDX,
HOURS_END_IDX);
            String minuti = duration.substring(MINUTES_START_IDX,
MINUTES_END_IDX);
            String secondi = duration.substring(SECONDS_START_IDX);
            int ris = Integer.parseInt(ora)*60 + Integer.parseInt(minuti);
            if (Integer.parseInt(secondi) >= 30) {
                  ris++;
            }
            return ris;
      }

      /*
       * C.
       * Supponendo un contratto con una franchigia di 100 minuti di
traffico voce mensili, il metodo deve calcolare
       * i minuti di traffico voce residuo per un determinato mese di un
determinato anno arrotondando,
       * per ogni chiamata, i secondi maggiori o uguali a 30 al minuto
successivo.
       */
      public static int trafficoResiduo(Contatto[] contatti, Chiamata[]
chiamate, int mese, int anno) {
            int m = 100;
            int n = contaContatti(contatti);

            String begin = "01/";
            if (mese  < 9) {
                  begin += "0";
```

```
            }
            begin += mese + "/" + anno;

            String end = "01/";
            if (mese < 12) {
                  mese += 1;
            }
            else {
                  mese = 1;
                  anno += 1;
            }
            if (mese < 9) {
                  end += "0";
            }
            end += mese + "/" + anno;

            for (int i=0; i<n; i++) {
                  if (
                        chiamate[i] != null && !chiamate[i].ricevuta &&
                        (confrontaData(begin, chiamate[i].data) <= 0) &&
                        (confrontaData(chiamate[i].data, end) < 0)
                  ) {
                        m -= convertiDurata(chiamate[i].durata);
                  }
            }

            return (m);
      }

      /*
       * D.
       * Il metodo deve eliminare dal database, specificato dai parametri
"contatti" e "chiamate",
       * tutte le chiamate "perse". Per chiamata persa si intende una
chiamata ricevuta
       * (attributo ricevuta = true, R/E = R nella tabella) con durata
pari a "00:00:00".
       * Il metodo deve restituire true o false a seconda del fatto che
sia stato eliminata almeno una chiamata
       * nel database e mantenere l'archivio in uno stato consistente ri-
muovendo i contatti duplicati
       * senza chiamata associata.
       */
      public static boolean rimuoviChiamatePerse(Contatto[] contatti,
Chiamata[] chiamate) {
            int n = contaContatti(contatti);
            int callDel = 0;
            int contDel = 0;

            for( int i = 0; i < n; i++ ) {

                  if ( (chiamate[i] != null) && chiamate[i].ricevuta &&
chiamate[i].durata.equals("00:00:00") ) {
                        chiamate[i] = null;
                        callDel++;
                  }

                  if (chiamate[i] == null) {
```

```
                        int j = 0;
                        while (
                              ( j < n ) &&
                              ( contatti[j] == null || i == j || contat-
ti[i].getId() != contatti[j].getId() )
                        ) {
                              j++;
                        }
                        if (j < n) {
                              contatti[i] = null;
                              contDel++;
                        }
                  }
            }

            if(contDel > 0) {
                  // Nota: si è fatta almeno una elminazione percui biso-
gna fare una compattazione
                  for(int i=0; i<n-1; i++) {
                        for(int j=i+1; j<n; j++) {
                              if(contatti[i] == null && contatti[j]!=
null){
                                    scambiaContatti(contatti, i, j);
                                    scambiaChiamate(chiamate, i, j);
                              }
                        }
                  }
            }
            return (callDel > 0);
      }

      public static void stampaDB(Contatto[] contatti, Chiamata[] chiama-
te){
            int n = contaContatti(contatti);
            for (int i=0; i < n; i++){
                  if (chiamate[i] != null){
                        System.out.println(contatti[i] + " " + chiama-
te[i]);
                  }
                  else {
                        System.out.println(contatti[i]);
                  }
            }
      }

      /*
       * E.
       */
      public static void main(String[] args) {
            Contatto[] contacts = new Contatto[MAX_ELEM];
            Chiamata[] calls = new Chiamata[MAX_ELEM];

            contacts[0] = new Contatto(0, "Pietro", "Rossi", "Via Roma,
31", "Pisa", "56127", "050642687");
            contacts[1] = new Contatto(1, "Giovanni", "Verdi", "Via Pol-
lione, 4", "Chieti", "66100", "08712278");
            contacts[2] = new Contatto(2, "Mario", "Ramarri", "Via Lenin,
4", "Roma", "00149", "0623476");
```

```
            contacts[3] = new Contatto(3, "Mario", "Rossi", "Piazza Cairo-
li, 3", "Pisa", "56124", "050576904");
            contacts[4] = new Contatto(1, "Giovanni", "Verdi", "Via Pol-
lione, 4", "Chieti", "66100", "08712278");
            contacts[5] = new Contatto(0, "Pietro", "Rossi", "Via Roma,
31", "Pisa", "56127", "050642687");

            calls[0] = new Chiamata(true, "01/01/2013", "00:00",
"00:00:00");
            calls[2] = new Chiamata(false, "03/02/2014", "09:45",
"00:00:55");
            calls[3] = new Chiamata(false, "02/04/2014", "10:21",
"00:01:23");
            calls[4] = new Chiamata(false, "07/07/2014", "15:32",
"01:33:45");
            calls[5] = new Chiamata(true, "09/10/2014", "17:04",
"00:00:00");

            System.out.println("\nA.");
            System.out.println("Il numero totale delle chiamate effettuate
dal 01/01/2014 al 31/12/2014 sono: ");
            System.out.println(contaChiamate(contacts, calls,
"01/01/2014", "31/12/2014"));

            System.out.println("\nB.");
            System.out.println("\nDatabase PRIMA dell'ordinamento:");
            stampaDB(contacts, calls);
            ordinaContatti(contacts, calls);
            System.out.println("\nDatabase DOPO l'ordinamento:");
            stampaDB(contacts, calls);

            System.out.println("\nC.");
            System.out.println("Il traffico voce residuo per il mese di
luglio dell'anno 2014 è di:");
            System.out.print(trafficoResiduo(contacts, calls, 7, 2014));
            System.out.println(" Minuti.");

            System.out.println("\nD.");
            if (rimuoviChiamatePerse(contacts, calls)) {
                  System.out.println("Le nuove chiamate sono:");
                  stampaDB(contacts, calls);
            }
            else{
                  System.out.println("Nessun inserimento effettuato.");
            }
      }

}
```