Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2015
9 Settembre 2015

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 “orders”. 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, int anno)
```

Il metodo deve contare il numero totale delle chiamate effettuate in un determinato anno specificato da un intero. Una chiamata è "effettuata" quando ha l'attributo ricevuta pari a **false** (R/E = R nella tabella) e la sua durata diversa da "00:00:00".

B. Scrivere il metodo statico:

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

Il metodo deve ordinare, nel vettore "contacts", gli elementi in maniera crescente, usando come criterio la durata per elemento 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 double creditoResiduo(Contatto[] contatti, Chiamata[] chiamate,
                                                          double credito)
```

Supponendo un piano tariffario di spesa pari a € 0.5 al minuto con scatto alla risposta pari ad € 0.1, per le chiamate in uscita; ed un piano tariffario di autoricarica pari a € 0.2 al minuto, senza scatto alla risposta, per le chiamate in ingresso; partendo da un credito iniziale pari al parametro "credito", il metodo deve calcolare il credito residuo in base a tutte le chiamate: sia effettuate (attributo ricevuta = false, R/E = E nella tabella), arrotondando i secondi non nulli al minuto successivo; sia ricevute (attributo ricevuta = true, R/E = R nella tabella) arrotondando i secondi al minuto successivo se e solo se sono maggiori di 30.

D. Scrivere il metodo statico:

```
public static boolean inserisciChiamata(Contatto[] contatti, Chiamata[] chiamate,
       boolean received, String tel, String date, String hour, Sgring duration)
```

Il metodo deve inserire database, specificato dai parametri "contatti" e "chiamate", una chiamata di una determinata durata, effettuata o ricevuta ad una determinata data ed ora verso il primo contatto trovato, individuato per mezzo del numero di telefono, che non abbia una chiamata associata oppure duplicando il contatto stesso in caso contrario. Il metodo deve restituire **true** o **false** a seconda del fatto che sia stato trovato almeno un contatto nel database. Se non esiste nessun contatto associato al numero di telefono, l'inserimento non deve essere effettuato. L'inserimento deve mantenere l'archivio ordinato come nel punto B e in uno stato consistente.

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	Mario Rossi	Piazza Cairoli, 3	Pisa	56124	050576904	R	01/01/2013	00:00	00:00:00
1	Pietro Rossi	Via Roma, 31	Pisa	56127	050642687				
2	Mario Ramarri	Via Lenin, 4	Roma	00149	0623476	R	03/02/2014	09:45	00:00:55
3	Giovanni Verdi	Via Pollione, 5	Chieti	66100	08712278	R	02/04/2014	10:21	00:01:23
3	Giovanni Verdi	Via Pollione, 5	Chieti	66100	08712278	E	07/07/2014	15:32	01:33:45
1	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 nell' anno 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 il credito residuo partendo da un credito iniziale di € 50,00.
- Avvalendosi del metodo al punto D. Si giunga la chiamata ricevuta dal numero "050642687", di durata pari a "00:01:05" in data "23/06/2015" alle "23:43". Al termine dell'operazione si stampi l'archivio aggiornato se l'operazione è avvenuta con successo, altrimenti si stampi un messaggio di errore.

```
public class Appello_20150909 {

	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;
	}

	/*
	 * A.
	 * Il metodo deve contare il numero totale delle chiamate effettuate in un determinato anno specificato da
un intero.
	 * Una chiamata è "effettuata" quando ha l'attributo ricevuta pari a true (R/E = R nella tabella) e la sua
durata è
	 * diversa da "00:00:00".
	 */
	public static int contaChiamate(Contatto[] contatti, Chiamata[] chiamate, int anno) {
		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") &&
				Integer.parseInt(chiamate[i].data.substring(YEAR_START_IDX)) == anno
			) {
				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 crescente,
	 * usando come criterio la durata per elemento 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.
	 */
	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)
&&
								(chiama-
te[j].durata.compareToIgnoreCase(chiamate[i].durata) < 0)
						)
					)
				) {
					scambiaContatti(contatti, i, j);
					scambiaChiamate(chiamate, i, j);
				}
			}
		}
	}

	public static int convertiDurata(String duration, int soglia) {
		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) > soglia)
                {
                        ris++;
                }
                return ris;
        }

        /*
         * C.
         * Supponendo un piano tariffario di spesa pari a € 0.5 al minuto con scatto alla risposta pari ad € 0.1,
         * per le chiamate in uscita; ed un piano tariffario di autoricarica pari a € 0.2 al minuto,
         * senza scatto alla risposta, per le chiamate in ingresso; partendo da un credito iniziale pari
         * al parametro "credito", il metodo deve calcolare il credito residuo in base a tutte le chiamate:
         * sia effettuate (attributo ricevuta = false, R/E = E nella tabella), arrotondando i secondi non nulli
         * al minuto successivo; sia ricevute (attributo ricevuta = true, R/E = R nella tabella) arrotondando
         * i secondi al minuto successivo se e solo se sono maggiori di 30.
         */
        public static double creditoResiduo(Contatto[] contatti, Chiamata[] chiamate, double credito) {
                int n = contaContatti(contatti);
                for (int i=0; i<n; i++) {
                        if (chiamate[i] != null && !chiamate[i].durata.equals("00:00:00")) {
                                if (chiamate[i].ricevuta) {
                                        credito += 0.2 * convertiDurata(chiamate[i].durata, 30);
                                }
                                else {
                                        credito -= (0.1 + 0.5 * convertiDurata(chiamate[i].durata, 0));
                                }
                        }
                }
                return credito;
        }

        /*
         * D.
         * Il metodo deve inserire database, specificato dai parametri “contatti” e “chiamate”,
         * una chiamata di una determinata durata effettuata o ricevuta ad una determinata data ed ora
         * verso il primo contatto trovato, individuato per mezzo del numero di telefono, che non abbia
         * una chiamata associata oppure duplicando il contatto stesso in caso contrario.
         * Il metodo deve restituire true o false a seconda del fatto che sia stato trovato almeno un contatto nel
database.
         * Se non esiste nessun contatto associato al numero di telefono, l'inserimento non deve essere effettua-
to.
         * L'inserimento deve mantenere l'archivio ordinato come nel punto precedente e in uno stato consistente.
         */
        public static boolean inserisciChiamata(
                Contatto[] contatti, Chiamata[] chiamate, boolean received, String tel, String date, String hour,
String duration
        ) {
                boolean ret = false;
                int n = contaContatti(contatti);
                if ( n < MAX_ELEM ) {
                        Chiamata c = new Chiamata(received, date, hour, duration);
                        int insertIdx = n;
                        int contactIdx = n;
                        for (int i = 0; i < n; i++) {
                                /* Si cerca l'indice dell'ultimo contatto. */
                                if (contatti[i].telefono.equals(tel)) {
                                        contactIdx = i;
                                }
                                /* Si cerca l'indice dell'inserimento */
                                if (
                                        chiamate[i] != null && chiamate[i].ricevuta == received &&
                                        (chiamate[i].durata.compareToIgnoreCase(duration) > 0)
                                ) {
                                                insertIdx = i;
                                }
                        }

                        /* Contatto presente? */
                        if (contactIdx != n) {

                                /* Inserimento */
                                if (chiamate[contactIdx] != null) {
                                        contatti[n] = new Contatto(
                                                        contatti[contactIdx].getId(),
                                                        contatti[contactIdx].nome,
                                                        contatti[contactIdx].cognome,
                                                        contatti[contactIdx].indirizzo,
                                                        contatti[contactIdx].citta,
                                                        contatti[contactIdx].cap,
                                                        contatti[contactIdx].telefono
                                        );
                                        chiamate[n] = c;
                                        contactIdx = n;
                                }
                                else {
                                        chiamate[contactIdx] = c;
                                }
```

```
                                /*
                                 * I contatti e le rispettive chiamate dalla posizione insertIdx alla posizione
contactIdx
                                 * devono essere spostati in avanti di una posizione, prima di poter effettuare
l'inserimento.
                                 * È importante notare che si parte dal fondo, in modo tale da evitare di sovra-
scrivere
                                 * i contatti successivi con le rispettive chiamate.
                                 *
                                 * E' possibile utilizzare anche il metodo ordinaContatti().
                                 */
                                Contatto co = contatti[contactIdx];
                                Chiamata ch = chiamate[contactIdx];
                                for(int i = contactIdx; i > insertIdx; i--) {
                                        contatti[i] = contatti[i - 1];
                                        chiamate[i] = chiamate[i - 1];
                                }
                                contatti[insertIdx] = co;
                                chiamate[insertIdx] = ch;
                                ret = true;
                        }
                }
                return ret;
        }

        public static void stampaDB(Contatto[] contatti, Chiamata[] chiamate){
                int n = contaContatti(contatti);
                for (int i=0; i < n; i++){
                        if (chiamate[i] != null){
                                System.out.println(contatti[i] + " " + chiamate[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, "Mario", "Rossi", "Piazza Cairoli, 3", "Pisa", "56124",
"050576904");
                contacts[1] = new Contatto(1, "Pietro", "Rossi", "Via Roma, 31", "Pisa", "56127", "050642687");
                contacts[2] = new Contatto(2, "Mario", "Ramarri", "Via Lenin, 4", "Roma", "00149", "0623476");
                contacts[3] = new Contatto(3, "Giovanni", "Verdi", "Via Pollione, 4", "Chieti", "66100",
"08712278");
                contacts[4] = new Contatto(3, "Giovanni", "Verdi", "Via Pollione, 4", "Chieti", "66100",
"08712278");
                contacts[5] = new Contatto(1, "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(true, "03/02/2014", "09:45", "00:00:55");
                calls[3] = new Chiamata(true, "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.print("Il numero totale delle chiamate nell'anno 2014 è: ");
                System.out.println(contaChiamate(contacts, calls, 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("Credito Iniziale: € " + 50.0);
                System.out.print("Credito Residuo: € ");
                System.out.println(creditoResiduo(contacts, calls, 50.0));

                System.out.println("\nD.");
                if (inserisciChiamata(contacts, calls, true, "050642687", "23/06/2015", "23:43", "00:01:05")) {
                        System.out.println("Le nuove chiamate sono:");
                        stampaDB(contacts, calls);
                }
                else{
                        System.out.println("Nessun inserimento effettuato.");
                }
        }

}
```