Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2018/2019
10 Giugno 2019

Testo

Il database del sistema di gestione delle officine di una compagnia di autoriparazioni è costituito da due vettori paralleli. Il primo è denominato “garages” e contiene oggetti di tipo “Officina” che rappresentano le officine presenti all'interno della banca dati della compagnia. Il secondo vettore è denominato “repairs” e contiene oggetti di tipo “Riparazione” che rappresentano le informazioni relative ad ogni riparazione effettuata o da effettuare in una determinata officina. Ad ogni officina può corrispondere più di una riparazione, in quel caso le informazioni dell’officina saranno replicate.
Per ogni officina presente nella posizione i-esima del vettore “garages”, le informazioni relative ad una riparazione si troveranno nella corrispondente posizione del vettore “repairs”. Nel caso in cui l’officina in posizione i-esima non abbia alcuna riparazione associata, nella posizione corrispondente nel vettore “repairs” sarà presente un riferimento *null*. Entrambi i vettori hanno dimensione pari alla costante “MAX_ELEM” (inizializzata a 1024). Se il numero delle officine contenute nell’archivio è inferiore a “MAX_ELEM”, i primi elementi del vettore conterranno gli oggetti di tipo “Officina”, mentre gli altri conterranno riferimenti *null*. Tutti gli elementi *null* del vettore “garages” si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Officina contiene le informazioni relative ad una officina:

```
public class Officina {

	private String codice;	public String citta;
	public String indirizzo;	public String cap;

	public Officina(String code, String city, String address, String zip) {
		codice = code;		citta = city;
		indirizzo = address;	cap = zip;
	}

	public String getCodice() {
		return codice;
	}

	public String toString() {
		return	("#" + codice + ": " + indirizzo + " - " + citta +  " (" + cap + ")");
	}

}
```

La classe Riparazione contiene le informazioni relative alle riparazioni relative ad una officina.

```
public class Riparazione {

	private static int numeroProgressivo = 0;
	private int numero;		private String descrizione;	public double prezzo;
	public String dataEntrata;	public String dataUscita;

	public Riparazione(String description, double price, String entranceDate, String exitDate) {
		descrizione = description;	prezzo = price;
		dataEntrata = entranceDate;	dataUscita = exitDate;
		numero = ++numeroProgressivo;
	}

	public Riparazione(int number, String description, double price, String entranceDate, String exitDate) {
		descrizione = description;	prezzo = price;
		dataEntrata = entranceDate;	dataUscita = exitDate;
		numero = number;
	}

	public int getNum() {
		return numero;
	}
	public String getDescrizione() {
		return descrizione;
	}

	public String toString() {
		String s = "[" + numero + "] " + descrizione + " (€ " + prezzo + ")" + ": " + dataEntrata;
		if (dataUscita != null) { s += " - " + dataUscita; }
		return s;

	}

}
```

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 "garages" che non contiene riferimenti *null*. Se si ha la necessità di convertire una stringa in un *long* oppure in un *int*, si possono utilizzare le funzioni di libreria *Long.parseLong(s)* ed *Integer.parseInt(s)* che convertono la stringa s rispettivamente in un *long* oppure in un *int* restituiti come risultato.

A. Scrivere il metodo statico:

```
public static double incassoTotale(Officina[] officine, Riparazione[] riparazioni, String descrizione)
```

Il metodo deve ritornare l'incasso totale di tutte le officine per la riparazione di descrizione passata come parametro.

B. Scrivere il metodo statico:

```
public static Riparazione [] genRiparazioni(Officina[] officine, Riparazione[] riparazioni)
```

Il metodo deve restituire un array contente le riparazioni ancora da effettuare in tutte le officine. Una riparazione è ancora da effettuare se non ha la data di uscita di un autoveicolo.

C. Scrivere il metodo statico:

```
public static void ordinaDB(Officina[] officine, Riparazione[] riparazioni)
```

Il metodo deve ordinare, nell'array parallelo, gli elementi in maniera crescente usando come criterio la data di ingresso di un autoveicolo. Le officine senza riparazioni associate vanno poste in fondo all'array.

D. Scrivere il metodo statico:

```
public static boolean rimuoviOfficina(Officina[] officine, Riparazione[] riparazioni, String citta)
```

Il metodo deve eliminare dal database tutte le officine di una determinata città passata come parametro; restituire **true** o **false** a seconda del fatto che sia stato eliminata almeno una officina e mantenere l'archivio in uno stato consistente.

E. Scrivere il metodo main che:

definisca ed inizializzi i vettori "garages" e "repairs" secondo i valori in tabella inizializzando a *null* o a *zero* i campi non presenti. La stampa dell'archivio consiste nello stampare le informazioni di ogni officina e delle relativeriparazioni (se ve ne sono). Si utilizzino correttamente i relativi costruttori e metodi *toString()* implementati nelle due classi.

Codice Officina	**Città**	**Indirizzo**	**CAP**	**Numero Riparazione**	**Descrizione**	**Prezzo** (€)	**Data Entrata** (gg/mm/aaaa)	**Data Uscita** (gg/mm/aaaa)
PICA95	Pisa	Via Cattaneo, 95	56125	2	Marmitta	180,00	01/05/2018	10/05/2018
GEMO9	Genova	Via Morin, 9	16129					
FIBA3	Firenze	Via dei Bardi, 3	50125	6	Frizione	340,00	10/06/2018	18/06/2018
PIDI2	Pisa	Via Diotisalvi, 2	56122	5	Motore	710,00	05/06/2018	
PIDI2	Pisa	Via Diotisalvi, 2	56122	7	Frizione	280,00	11/06/2018	21/06/2018
PICA95	Pisa	Via Cattaneo, 95	56125	1	Semiasse	543,00	28/04/2018	31/06/2018
FIBA3	Firenze	Via dei Bardi, 3	50125	3	Trasmissione	311,00	18/05/2018	28/06/2018
FIBA3	Firenze	Via dei Bardi, 3	50125	4	Motore	60,00	01/06/2018	

- Avvalendosi del metodo al punto A, stampi a video gli incassi totale di tutte le officine per le riparazioni di frizione e motore.
- Avvalendosi del metodo al punto B stampi a video le riparazioni da effettuare in tutte le officine. Se non ve ne sono, stampi un opportuno messaggio.
- Ordini l'intero archivio utilizzando il metodo al punto C e stampi a video l'archivio prima e dopo l'ordinamento.
- Avvalendosi del metodo al punto D, elimini dall'archivio tutte le officine della città di Pisa. 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 Esame {
	public static final int MAX_ELEM = 1024;

	public static int contaOfficine(Officina[] officine){
		int count = 0;
		while (count < MAX_ELEM && officine[count] != null) {
			count++;
		}
		return count;
	}

	/*
	 * A.
	 * Il metodo deve ritornare l'incasso totale di tutte le officine
	 * per la riparazione passata come parametro.
	 */


	public static double incassoTotale(Officina[] officine, Riparazione[] riparazioni,
String descrizione) {
		double incasso = 0.0;
		int n = contaOfficine(officine);
		for (int i = 0; i < n; i++) {
			if (riparazioni[i] != null && riparazioni[i].getDescri-
zione().equals(descrizione) ) {
				incasso += riparazioni[i].prezzo;
			}
		}
		return incasso;
	}


	/*
	 * B.
	 * Il metodo deve restituire un array contente le riparazioni ancora da effettuare
in tutte le officine.
	 * Una riparazione è ancora da effettuare se non ha la data di uscita di un auto-
veicolo.
	 */
	public static Riparazione[] genRiparazioni(Officina[] officine, Riparazione[] ri-
parazioni) {
		Riparazione[] risu = null;
		int count = 0;
		int n = contaOfficine(officine);
		for (int i = 0; i < n; i++) {
			if (riparazioni[i] != null && riparazioni[i].dataUscita == null) {
				count++;
			}
		}

		if (count>0) {
			risu = new Riparazione[count];
			int k=0;
			for (int i = 0; i < n; i++) {
				if (riparazioni[i] != null && riparazioni[i].dataUscita ==
null) {
					risu[k]=riparazioni[i];
					k++;
				}
			}
		}

		return risu;
	}


	// Trasforma la data in formato long per essere confrontata
	private static long dataConfrontabile(String data)
	{
		return Long.parseLong(data.substring(6, 10) + data.substring(3,5) +
data.substring(0,2));
```

```
	}

	// scambia l'elemento in posizione i-esima con quello in posizione j-esima
	private static void scambiaOfficine(Officina[] t, int i, int j){
		Officina tmp = t[i];
		t[i] = t[j];
		t[j] = tmp;
	}

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

	/*
	 * C.
	 * Il metodo deve ordinare, nell'array parallelo, gli elementi in maniera
	 * crescente usando come criterio la data di ingresso di un autoveicolo.
	 * Le officine senza riparazioni associate vanno poste in fondo all'array.
	 */
	public static void ordinaDB(Officina[] officine, Riparazione[] riparazioni) {
		int n = contaOfficine(officine);
		for (int i = 0; i < n-1; i++) {
			for (int j = i + 1; j < n; j++) {
				if (
					riparazioni[i] == null || (
						riparazioni[j] != null && (
							dataConfrontabile(riparazioni[j].dataEn-
trata) <
							dataConfrontabile(riparazioni[i].dataEn-
trata)
						)
					)
				) {
					scambiaOfficine(officine, i, j);
					scambiaRiparazioni(riparazioni, i, j);
				}
			}
		}
	}



	/*
	 * D.
	 * Il metodo deve eliminare dal database tutte le officine di una determinata
città passata come parametro;
	 * restituire true o false a seconda del fatto che sia stato eliminato almeno una
officina e mantenere
	 * l'archivio in uno stato consistente.
	 */
	public static boolean rimuoviOfficine(Officina[] officine, Riparazione[] ripara-
zioni, String citta) {
		int del = 0;
		int n = contaOfficine(officine);
		for (int i = 0; i < n; i++) {
			if (officine[i].citta.equals(citta)) {
				officine[i] = null;
				riparazioni[i] = null;
				del++;
			}
		}
		if (del > 0) {
			for (int i = 0; i < n - 1; i++) {
				for (int j = i; j < n; j++) {
					if (officine[i] == null && officine[j] != null) {
						scambiaOfficine(officine, i, j);
						scambiaRiparazioni(riparazioni, i, j);
					}
```

```
                        }
                    }
                }
                return (del > 0);
        }

        public static void stampaDB(Officina[] officine, Riparazione[] riparazioni){
                int n = contaOfficine(officine);
                for (int i=0; i < n; i++){
                        if (riparazioni[i] != null){
                                System.out.println(officine[i] + " => " + riparazioni[i]);
                        }
                        else {
                                System.out.println(officine[i]);
                        }
                }
        }

        public static void main(String[] args) {

                Officina[]          garages = new Officina[MAX_ELEM];
                Riparazione[] repairs = new Riparazione[MAX_ELEM];

                garages[0] = new Officina("PICA95", "Pisa", "Via Cattaneo, 95", "56125");
                garages[1] = new Officina("GEMO9", "Genova", "Via Morin, 9", "16129");
                garages[2] = new Officina("FIBA3", "Firenze", "Via dei Bardi, 3", "50125");
                garages[3] = new Officina("PIDI2", "Pisa", "Via Diotisalvi, 2", "56122
        ");
                garages[4] = garages[3];
                garages[5] = garages[0];
                garages[6] = garages[2];
                garages[7] = garages[2];

                repairs[5] = new Riparazione("Semiasse", 543.00, "28/04/2018",
"28/04/2018");
                repairs[0] = new Riparazione("Marmitta", 180.00, "01/05/2018",
"01/05/2018");
                repairs[6] = new Riparazione("Trasmissione", 311.00, "18/05/2018",
"18/05/2018");
                repairs[7] = new Riparazione("Motore", 60.00, "01/06/2018", null);
                repairs[3] = new Riparazione("Motore", 710.00, "05/06/2018", null);
                repairs[2] = new Riparazione("Frizione", 280.00, "10/06/2018",
"11/06/2018");
                repairs[4] = new Riparazione("Frizione", 340.00, "11/06/2018",
"11/06/2018");


                System.out.println("\nA.");
                System.out.println("L'incasso totale per i motori vale "+incassoTotale(ga-
rages, repairs,"Motore"));
                System.out.println("L'incasso totale per le frizioni vale "+incassoTo-
tale(garages, repairs,"Frizione"));

                System.out.println("\nB.");
                System.out.println("Riparazioni da effettuare: ");
                Riparazione[] dariparare = genRiparazioni(garages, repairs);
                if (dariparare==null || dariparare.length==0)
                        System.out.println("Nessuna");
                else for (int i=0; i<dariparare.length;i++)
                                System.out.println(dariparare[i]);



                System.out.println("\nC.");
                System.out.println("Database PRIMA dell'ordinamento:");
                stampaDB(garages, repairs);
                ordinaDB(garages, repairs);
                System.out.println("\nDatabase DOPO l'ordinamento::");
                stampaDB(garages, repairs);
```

```
		System.out.println("\nD.");
		if (rimuoviOfficine(garages, repairs, "Pisa")) {
			System.out.println("Archivio aggiornato:");
			stampaDB(garages, repairs);
		}
		else {
			System.out.println("Nessuna Eliminazione!");
		}

	}

}
```