Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2017/2018
2 Luglio 2018

Testo

Il database del sistema di gestione delle tratte autostradali di una compagnia di gestione delle autostrade è costituito da due vettori paralleli. Il primo è denominato “sections” e contiene oggetti di tipo “Tratta” che rappresentano le tratte autostradali presenti all'interno della banca dati della compagnia. Il secondo vettore è denominato “passes” e contiene oggetti di tipo “Passaggio” che rappresentano le informazioni relative ad ogni passaggio effettuato da un veicolo in una determinata tratta autostradale. Ad ogni tratta autostradale può corrispondere più di un passaggio veicolare, in quel caso le informazioni della tratta autostradale saranno replicate.
Per ogni tratta autostradale presente nella posizione i-esima del vettore “sections”, le informazioni relative ad un passaggio veicolare si troveranno nella corrispondente posizione del vettore “passes”. Nel caso in cui la tratta autostradale in posizione i-esima non abbia alcun passaggio veicolare associato, nella posizione corrispondente nel vettore “passess” sarà presente un riferimento *null*. Entrambi i vettori hanno dimensione pari alla costante “MAX_ELEM” (inizializzata a 1024). Se il numero delle tratte contenute nell’archivio è inferiore a “MAX_ELEM”, i primi elementi del vettore conterranno gli oggetti di tipo “Tratta”, mentre gli altri conterranno riferimenti *null*. Tutti gli elementi *null* del vettore “sections” si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Tratta contiene le informazioni relative ad una tratta autostradale:

```
public class Tratta {

	private String codice;	public String ingresso;	public String uscita;
	public double distanza;	public double costo;	public String autostrada;

	public Tratta(String code, String entrance, String exit, double distance, double cost, String highway) {
		codice = code;		ingresso = entrance;		uscita = exit;
		distanza = distance;	costo = cost;		autostrada = highway;
	}

	public String getCodice() {
		return codice;
	}

	public String toString() {
		return	("#" + codice + " [" + autostrada + "]: " + ingresso + " - " + uscita +
				" (Km " + distanza + ", € " + costo + ")");
	}

}
```

La classe Passaggio contiene le informazioni relative ai passaggi veicolari relativi ad una tratta autostradale.

```
public class Passaggio {

	private static int numeroProgressivo = 0;
	private int numero;		public String targa;		public boolean titoloViaggio;
	public String dataEntrata;	public String dataUscita;

	public Passaggio(String licenseNumber, char ticketType, String entranceDate, String exitDate) {
		targa = licenseNumber;		titoloViaggio = tickeType;
		dataEntrata = entranceDate;	dataUscita = exitDate;
		numero = ++numeroProgressivo;
	}

	public Passaggio(int number, String license, char ticketType, String entranceDate, String exitDate) {
		targa = license;		titoloViaggio = ticketType;
		dataEntrata = entranceDate;	dataUscita = exitDate;
		numero = number;
	}

	public int getNum() {
		return numero;
	}

	public String toString() {
		return ( "[" + numero + "] " + targa + ": " + dataEntrata + " - " + dataUscita +
				" (" + titoloViaggio + ")");
	}

}
```

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 "sections" 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 contaVeicoli(Tratta[] tratte, Passaggio[] passaggi)
```

Il metodo deve contare il numero di veicoli distinti transitati in tutte le tratte autostradali.

B. Scrivere il metodo statico:

```
public static void ordinaDB(Tratta[] tratte, Passaggio[] passaggi)
```

Il metodo deve ordinare, nell'array parallelo, gli elementi in maniera crescente usando come criterio il codice della tratta ed eventualmente la targa del veicolo che ha percorso la tratta stessa. Le tratte senza passaggi associati vanno poste in fondo all'array.

C. Scrivere il metodo statico:

```
public static double spesaVeicolo(Tratta[] tratte, Passaggio[] passaggi,
                                                        String targa)
```

Il metodo deve calcolare la spesa totale di un veicolo identificato dalla sua targa.

D. Scrivere il metodo statico:

```
public static boolean rimuoviVeicoli(Tratta[] tratte, Passaggio[] passaggi)
```

Il metodo deve eliminare dal database tutti i passaggi il cui veicolo abbia un titolo di viaggio; restituire **true** o **false** a seconda del fatto che sia stato eliminato almeno un passaggio e mantenere l'archivio in uno stato consistente evitando di eliminare definitivamente le tratte senza passaggi associati.

E. Scrivere il metodo main che:

definisca ed inizializzi i vettori "sections" e "passes" 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 tratta autostradale e dei relativi passaggi veicolari (se ve ne sono). Si utilizzino correttamente i relativi metodi *toString()* implementati nelle due classi.

Codice	Autostrada	Ingresso	Uscita	Distanza (Km)	Costo (€)	Nr.	Targa	Data Entrata (gg/mm/aaaa hh:mm)	Data Uscita (gg/mm/aaaa hh:mm)	Titolo Viaggio
PNVA12	A12	Pisa Nord	Viareggio	11,7	2,00	2	PG345ZQ	01/05/2018 20:10	01/05/2018 20:15	(T)elepass
GEVA12	A12	Genova Est	Viareggio	131,9	15,00					
FNBCA1	A1	Firenze Nord	Bologna Casalecchio	85,4	8,10	6	QR987LK	11/06/2018 00:05	11/06/2018 01:23	(T)elepass
PNFNA11	A11	Pisa Nord	Firenze Nord	76,4	6,1	5	HJ234FR	01/06/2018 09:36	01/06/2018 09:05	(N)essuno
PNFNA11	A11	Pisa Nord	Firenze Nord	76,4	61	7	QR987LK	10/06/2018 22:45	11/06/2018 00:05	(T)elepass
PNVA12	A12	Pisa Nord	Viareggio	11,7	2,00	1	AJ632RR	28/04/2018 07:02	28/04/2018 07:15	(T)elepass
FNBCA1	A1	Firenze Nord	Bologna Casalecchio	85,4	8,10	3	AS768ED	18/05/2018 08:36	18/05/2018 09:45	(B)iglietto
FNBCA1	A1	Firenze Nord	Bologna Casalecchio	85,4	8,10	4	HJ234FR	01/06/2018 09:05	01/06/2018 08:22	(N)essuno

- Avvalendosi del metodo al punto A stampi a video il numero di veicoli distinti transitati in tutte le tratte autostradali.
- Ordini l'intero archivio utilizzando il metodo del punto B e stampi a video l'archivio prima e dopo l'ordinamento.
- Avvalendosi del metodo al punto C stampi a video la spesa totale del veicolo identitificato dalla targa AJ632RR.
- Utilizzando il metodo al punto D, effettui l'eliminazione di tutti i passaggi il cui veicolo abbia un titolo di viaggio. Al termine dell'operazione si stampi l'archivio aggiornato se l'operazione è avvenuta con successo, altrimenti si stampi un messaggio di errore.

Soluzione

```
public class Esame {
```

```
	public static final int MAX_ELEM = 1024;

	public static int contaTratte(Tratta[] tratte){
		int count = 0;
		while (count < MAX_ELEM && tratte[count] != null) {
			count++;
		}
		return count;
	}

	/*
	 * A.
	 * Il metodo deve contare il numero di veicoli distinti transitati in tutte le tratte autostradali.
	 */
	public static int contaVeicoli(Tratta[] tratte, Passaggio[] passaggi) {
		int count = 0;
		int n = contaTratte(tratte);
		for (int i = 0; i < n; i++) {
			if ( passaggi[i] != null ) {
				boolean conteggiato = false;
				for (int j = 0; j < i && !conteggiato; j++) {
					conteggiato = ( passaggi[j] != null ) &&
							( passaggi[i].targa.equals(passaggi[j].targa) );
				}
				if (!conteggiato) {
					count++;
				}
			}
		}
		return count;
	}

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

	/*
	 * B.
	 * Il metodo deve ordinare, nell'array parallelo, gli elementi in maniera crescente usando come
	 * criterio il codice della tratta ed eventualmente la targa del veicolo che ha percorso la tratta stessa.
```

```
      * Le tratte senza passaggi associati vanno poste in fondo all'ar-
ray.
      */
     public static void ordinaDB(Tratta[] tratte, Passaggio[] passaggi)
{
          int n = contaTratte(tratte);
          for (int i = 0; i < n-1; i++) {
               for (int j = i + 1; j < n; j++) {
                    if (
                         passaggi[i] == null || (
                              passaggi[j] != null && (
                                   tratte[i].getCodice().compare-
ToIgnoreCase(tratte[j].getCodice()) > 0 ||
                                   (
                                        tratte[i].getCodice().compa-
reToIgnoreCase(tratte[j].getCodice()) == 0 &&
                                        passaggi[i].targa.compare-
ToIgnoreCase(passaggi[j].targa) > 0
                                   )
                              )
                         )
                    ) {
                         scambiaTratte(tratte, i, j);
                         scambiaPassaggi(passaggi, i, j);
                    }
               }
          }
     }

     /*
      * C.
      * Il metodo deve calcolare la spesa totale di un veicolo identifi-
cato dalla sua targa.
      */
     public static double spesaVeicolo(Tratta[] tratte, Passaggio[] pas-
saggi, String targa) {
          double spesa = 0.0;
          int n = contaTratte(tratte);
          for (int i = 0; i < n; i++) {
               if (passaggi[i] != null && pas-
saggi[i].targa.equals(targa)) {
                    spesa += tratte[i].costo;
               }
          }
          return spesa;
     }

     /*
      * D.
      * Il metodo deve eliminare dal database tutti i passaggi il cui
veicolo abbia un titolo di viaggio;
      * restituire true o false a seconda del fatto che sia stato elimi-
nato almeno un passaggio e mantenere
      * l'archivio in uno stato consistente evitando di eliminare defi-
nitivamente le tratte senza passaggi associati.
      */
     public static boolean rimuoviVeicoli(Tratta[] tratte, Passaggio[]
passaggi) {
```

```
            int del = 0;
            int n = contaTratte(tratte);
            for (int i = 0; i < n; i++) {
                  if ( passaggi[i] != null && passaggi[i].titoloViaggio !=
'N' ) {
                        passaggi[i] = null;
                        boolean daEliminare = false;
                        for (int j = 0; j < n && !daEliminare; j++) {
                              daEliminare = (i != j) && (tratte[i] != null)
&& (tratte[j] != null) &&
                                                      (tratte[i].getCo-
dice().equals(tratte[j].getCodice()));
                        }
                        if (daEliminare) {
                              tratte[i] = null;
                        }
                        del++;
                  }
            }
            if (del > 0) {
                  for (int i = 0; i < n - 1; i++) {
                        for (int j = i; j < n; j++) {
                              if (tratte[i] == null && tratte[j] != null) {
                                    scambiaTratte(tratte, i, j);
                                    scambiaPassaggi(passaggi, i, j);
                              }
                        }
                  }
            }
            return (del > 0);
      }

      public static void stampaDB(Tratta[] tratte, Passaggio[] passaggi){
            int n = contaTratte(tratte);
            for (int i=0; i < n; i++){
                  if (passaggi[i] != null){
                        System.out.println(tratte[i] + " => " + pas-
saggi[i]);
                  }
                  else {
                        System.out.println(tratte[i]);
                  }
            }
      }

      public static void main(String[] args) {

            Tratta[] sections = new Tratta[MAX_ELEM];
            Passaggio[] passes = new Passaggio[MAX_ELEM];

            sections[0] = new Tratta("PNVA12", "Pisa Nord", "Viareggio",
11.7, 2.0, "A12");
            sections[1] = new Tratta("GEVA12", "Genova Est", "Viareggio",
131.9, 15.0, "A12");
            sections[2] = new Tratta("FNBCA1", "Firenze Nord", "Bologna
Casalecchio", 85.4, 8.1, "A1");
            sections[3] = new Tratta("PNFNA11", "Pisa Nord", "Firenze
Nord", 76.4, 6.1, "A11");
```

```
            sections[4] = sections[3];
            sections[5] = sections[0];
            sections[6] = sections[2];
            sections[7] = sections[2];

            passes[5] = new Passaggio("AJ632RR", 'T', "28/04/2018 07:02",
"28/04/2018 07:15");
            passes[0] = new Passaggio("PG345ZQ", 'T', "01/05/2018 20:10",
"01/05/2018 20:15");
            passes[6] = new Passaggio("AS768ED", 'B', "18/05/2018 08:36",
"18/05/2018 09:45");
            passes[7] = new Passaggio("HJ234FR", 'N', "01/06/2018 09:05",
"01/06/2018 08:22");
            passes[3] = new Passaggio("HJ234FR", 'N', "01/06/2018 09:36",
"01/06/2018 09:05");
            passes[4] = new Passaggio("QR987LK", 'T', "10/06/2018 22:45",
"11/06/2018 00:05");
            passes[2] = new Passaggio("QR987LK", 'T', "11/06/2018 00:05",
"11/06/2018 01:23");

            System.out.println("\nA.");
            System.out.println("Il numero di veicoli distinti è: " + con-
taVeicoli(sections, passes));

            System.out.println("\nB.");
            System.out.println("Database PRIMA dell'ordinamento:");
            stampaDB(sections, passes);
            ordinaDB(sections, passes);
            System.out.println("\nDatabase DOPO l'ordinamento:");
            stampaDB(sections, passes);

            System.out.println("\nC.");
            System.out.println("La spesa del veicolo targato AJ632RR è di:
" +
                        spesaVeicolo(sections, passes, "AJ632RR") + " €");

            System.out.println("\nD.");
            if (rimuoviVeicoli(sections, passes)) {
                  System.out.println("Operazione effettuata, nuovo data-
base:");
                  stampaDB(sections, passes);
            }
            else {
                  System.out.println("Operazione non effettuata!");
            }
      }

}
```