Cognome _____ Nome _____
Matricola _____ Postazione PC _____

Corso di Laurea in Ingegneria Gestionale
Esame di Informatica - a.a. 2015/2016
4 Luglio 2016

Testo

Il database del sistema di gestione delle tessere "fedeltà" di un supermercato è costituito da due vettori paralleli. Il primo è denominato "cards" e contiene oggetti di tipo "Tessera" che rappresentano le tessere presenti all'interno della banca dati del supermercato. Il secondo vettore è denominato "products" e contiene oggetti di tipo "Prodotto" che rappresentano le informazioni relative ad ogni prodotto acquistato con una determinata tessera. Ad ogni tessera può corrispondere più di un prodotto acquistato, in quel caso le informazioni della tessera saranno replicate.
Per ogni tessera presente nella posizione i-esima del vettore "cards", le informazioni relative ad un prodotto si troveranno nella corrispondente posizione del vettore "products". Nel caso in cui la tessera in posizione i-esima non abbia alcun prodotto associato, nella posizione corrispondente nel vettore "products" sarà presente un riferimento *null*. Entrambi i vettori hanno dimensione pari alla costante "MAX_ELEM" (inizializzata a 1024). Se il numero di tessere contenute nell'archivio è inferiore a "MAX_ELEM", i primi elementi del vettore conterranno gli oggetti di tipo "Tessera", mentre gli altri conterranno riferimenti *null*. Tutti gli elementi *null* del vettore "cards" si devono trovare alla fine del vettore e non possono trovarsi in mezzo agli elementi validi.

Le classe Tessera contiene le informazioni relative ad un tessera:

```
public class Tessera {

        private static int numeroProgressivo = 0;
        private int numero;                    public String cliente;
        public String iscrizione;       public String validita;

        public Tessera(String name, String signup, String validity) {
                numero = numeroProgressivo++;       cliente = name;
                iscrizione = signup;           validita = validity;
        }

        public Tessera(int number, String name, String signup, String validity) {
                numero = number;           cliente = name;
                iscrizione = signup; validita = validity;
        }

        public int getNum() {
                return numero;
        }

        public String toString() {
                return    ("#" + numero + ": " + cliente + " - " +
                          iscrizione + " ( " + validita + " )");
        }

}
```

La classe Prodot contiene le informazioni relative ai singoli prodotti acquistati con una determinata tessera.

```
public class Prodotto {

        private String codice;          public String data;
        public double prezzo;           public int quantita;

        public Prodotto(String code, String date, double price, int amount) {
                codice = code;          data = date;
                prezzo = price;         quantita = amount;
        }

        public String getCodice() {
```

```
            return codice;
        }

        public String toString() {
            return ( "[" + codice + "] " + data +
                    " - " + prezzo + " € (" + quantita + ")");
        }

}
```

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 "cards" 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 contaTesserePerProdotto(Tessera[] tessere,
                                          Prodotto[] prodotti, String code)
```

Il metodo deve contare il numero totale di tessere che abbiano acquistato almeno un prodotto con un determinato codice ("code"). Nel caso ci siano tessere che abbiano acquistato più prodotti in oggetto, esse vanno conteggiate una sola volta.

B. Scrivere il metodo statico:

```
public static void ordina(Tessera[] tessere, Prodotto[] prodotti, boolean crescente)
```

Il metodo deve ordinare, nel vettore "cards", gli elementi in maniera crescente o decrescente, in base al parametro 'crescente', usando come criterio la data di iscrizione della tessera.

C. Scrivere il metodo statico:

```
public static int puntiFedelta(Tessera[] tessere, Prodotto[] prodotti,
                                                          int numero)
```

Supponendo che il supermecato assegni dei punti fedeltà ad ogni tessera assegnando 1 punto per ogni euro speso per ogni singolo prodotto, arrotondato per difetto; il metodo deve calcolare il totale dei punti accumulati da una determinata tessera individuata dal suo numero. Per l'arrotondamento si può utilizzare il metodo *Math.floor(d)*.

D. Scrivere il metodo statico:

```
public static boolean creaDuplicatoTessera(Tessera[] tessere,
                                           Prodotto[] prodotti, int numero)
```

Il metodo deve duplicare una determinata tessera individuata dal suo numero assegnandogli sia un nuovo numero progressivo calcolato automaticamente sia tutti gli acquisti della tessera originale duplicandoli a sua volta. Il metodo inoltre deve restituire **true** o **false** a seconda del fatto che sia stata possibile la duplicazione completa.

E. Scrivere il metodo main che:

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

Numero	**Cliente**	**Iscrizione** (gg/mm/aaaa)	**Validità** (mm/aaaa)	**Codice**	**Data** (gg/mm/aaaa)	**Prezzo (€)**	**Quantità**
0	Pietro Rossi	03/02/2013	02/2018	A01	03/02/2013	1,80	1
1	Mario Rossi	15/03/2014	03/2019				
2	Mario Ramarri	02/04/2014	04/2019	A02	02/04/2014	3,50	5
3	Giovanni Verdi	07/07/2016	07/2021	C03	07/07/2016	10,80	10
3	Giovanni Verdi	07/07/2016	07/2021	F18	07/07/2016	4,99	3
0	Pietro Rossi	03/02/2013	02/2018	F18	09/10/2014	4,99	4
2	Mario Ramarri	02/04/2014	04/2019	C03	09/11/2014	10,80	2
2	Mario Ramarri	02/04/2014	04/2019	A01	25/12/2014	1,80	1

1. Avvalendosi del metodo al punto A stampi a video, il numero delle tessere che hanno acquistato almeno un prodotto con codice C03.
2. Ordini l'intero archivio utilizzando il metodo del punto B prima in maniera decrescente e poi in maniera crescente e stampi a video l'archivio prima e dopo i successivi ordinamenti.
3. Utilizzando il metodo C, stampi il totale dei punti della tessera numero 0.
4. Duplichi la tessera numero 2 utilizzando il metodo del punto D. 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 Appello_20160704 {

	public static final int MAX_ELEM = 1024;

	public static int contaTessere(Tessera[] tessere){
		int count = 0;
		while (count < MAX_ELEM && tessere[count] != null) {
			count++;
		}
		return count;
	}

	/*
	 * A.
	 * Il metodo deve contare il numero totale di tessere che abbiano acquistato almeno un pro-
dotto
	 * con un determinato codice ("code"). Nel caso ci siano tessere che abbiano acquistato
	 * più prodotti in oggetto, esse vanno conteggiate una sola volta.
	 */
	public static int contaTesserePerProdotto(Tessera[] tessere, Prodotto[] prodotti, String co-
de) {
		int count = 0;
		int n = contaTessere(tessere);
		for (int i=0; i<n; i++) {
			if ( prodotti[i] != null && code.equals(prodotti[i].getCodice()) ) {
				boolean conteggiato = false;
				for (int j=0; j<i && !conteggiato; j++) {
					conteggiato = (
							(tessere[i].getNum() == tessere[j].getNum()) &&
							(prodotti[j] != null) &&
							(code.equals(prodotti[j].getCodice()))
					);
				}
				if (!conteggiato) {
					count++;
				}
			}
		}
		return count;
	}

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

	private static String dataIscrizioneConfrontabile(String data) {
		return data.substring(6) + data.substring(3, 5) + data.substring(0, 2);
	}

	/* B.
	 * Il metodo deve ordinare, nel vettore "cards", gli elementi in maniera crescente o
```

```
	 * decrescente, in base al parametro 'crescente', usando come criterio la data di
	 * iscrizione della tessera.
	 */
	public static void ordinaTessere(Tessera[] tessere, Prodotto[] prodotti, boolean crescente){
		int n = contaTessere(tessere);
		for (int i = 0; i < n-1; i++) {
			for (int j = i + 1; j < n; j++) {
				String di = dataIscrizioneConfrontabile(tessere[i].iscrizione);
				String dj = dataIscrizioneConfrontabile(tessere[j].iscrizione);
				if (
						( crescente && di.compareTo(dj) > 0 ) ||
						(!crescente && dj.compareTo(di) > 0 )
				) {
					scambiaTessere(tessere, i, j);
					scambiaProdotti(prodotti, i, j);
				}
			}
		}
	}

	/*
	 * C.
	 * Supponendo che il supermecato assegni dei punti fedeltà ad ogni tessera assegnando
	 * 1 punto per ogni euro speso per ogni prodotto arrotondato per difetto; il metodo
	 * deve calcolare il totale dei punti accumulati da una determinata tessera individuata
	 * dal suo numero. Per l'arrotondamento si può utilizzare il metodo Math.floor(d).
	 */
	public static int puntiFedelta(Tessera[] tessere, Prodotto[] prodotti, int numero) {
		int n = contaTessere(tessere);
		int punti = 0;

		for (int i=0; i<n; i++) {
			if (tessere[i].getNum() == numero && prodotti[i] != null) {
				punti += Math.floor(prodotti[i].prezzo * prodotti[i].quantita);
			}
		}
		return punti;
	}

	private static int contaIstanzeTessera(Tessera[] tessere, int numero) {
		int n = contaTessere(tessere);
		int count = 0;

		for (int i=0; i<n; i++) {
			if (tessere[i].getNum() == numero) {
				count++;
			}
		}

		return count;
	}

	/*
	 * D.
	 * Il metodo deve duplicare una determinata tessera individuata dal suo numero
	 * assegnandogli sia un nuovo numero progressivo calcolato automaticamente sia
	 * tutti gli acquisti della tessera originale duplicandoli a sua volta. Il metodo
	 * inoltre deve restituire true o false a seconda del fatto che sia stata possibile
	 * la duplicazione completa.
	 */
	public static boolean creaDuplicatoTessera(Tessera[] tessere, Prodotto[] prodotti, int nume-
ro) {
		int n = contaTessere(tessere);
		int m = contaIstanzeTessera(tessere, numero);
		boolean ris = ((m > 0) && (MAX_ELEM >= (n + m)));

		if (ris) {
			int duplicati = 0;
			for(int i=0; i < n; i++) {
				if (tessere[i].getNum() == numero) {
					if (duplicati == 0) {
						tessere[n] = new Tessera(
							tessere[i].cliente, tessere[i].iscrizione,
							tessere[i].validita
						);
					}
					else {
						tessere[n+duplicati] = new Tessera(
							tessere[n].getNum(), tessere[n].cliente,
							tessere[n].iscrizione, tessere[n].validita
```

```
                                        );
                                }
                                if (prodotti[i] != null) {
                                        prodotti[n+duplicati] = new Prodotto(
                                                prodotti[i].getCodice(), prodotti[i].data,
                                                prodotti[i].prezzo, prodotti[i].quantita
                                        );
                                }
                                duplicati++;
                        }
                }
        }
        return (ris);
    }

    public static void stampaDB(Tessera[] tessere, Prodotto[] prodotti){
        int n = contaTessere(tessere);
        for (int i=0; i < n; i++){
            if (prodotti[i] != null){
                System.out.println(tessere[i] + " -> " + prodotti[i]);
            }
            else {
                System.out.println(tessere[i]);
            }
        }
    }

    public static void main(String[] args) {
        Tessera[] cards = new Tessera[MAX_ELEM];
        Prodotto[] products = new Prodotto[MAX_ELEM];

        cards[0] = new Tessera("Pietro Rossi", "03/02/2013", "02/2018");
        cards[1] = new Tessera("Mario Rossi", "15/03/2014", "03/2019");
        cards[2] = new Tessera("Mario Ramarri", "02/04/2014", "04/2019");
        cards[3] = new Tessera("Giovanni Verdi", "07/07/2016", "07/2021");
        cards[4] = new Tessera(cards[3].getNum(), "Giovanni Verdi", "07/07/2016", "07/2021");
        cards[5] = new Tessera(cards[0].getNum(), "Pietro Rossi", "03/02/2013", "02/2018");
        cards[6] = new Tessera(cards[2].getNum(), "Mario Ramarri", "02/04/2014", "04/2019");
        cards[7] = new Tessera(cards[2].getNum(), "Mario Ramarri", "02/04/2014", "04/2019");

        products[0] = new Prodotto("A01", "03/02/2013", 1.8, 1);
        products[2] = new Prodotto("A02", "02/04/2014", 3.5, 5);
        products[3] = new Prodotto("C03", "07/07/2016", 10.8, 10);
        products[4] = new Prodotto("F18", "07/07/2016", 4.99, 3);
        products[5] = new Prodotto("F18", "09/10/2014", 4.99, 4);
        products[6] = new Prodotto("C03", "09/11/2014", 10.8, 2);
        products[7] = new Prodotto("A01", "25/12/2014", 1.8, 1);

        System.out.println("\nA.");
        System.out.print("Il numero totale di tessere che hanno acquistato il prodotto con
codice C03 è: ");
        System.out.println(contaTesserePerProdotto(cards, products, "C03"));

        System.out.println("\nB.");
        System.out.println("\nDatabase PRIMA dell'ordinamento:");
        stampaDB(cards, products);
        ordinaTessere(cards, products, false);
        System.out.println("\nDatabase DOPO dell'ordinamento decrescente:");
        stampaDB(cards, products);
        ordinaTessere(cards, products, true);
        System.out.println("\nDatabase DOPO dell'ordinamento crescente:");
        stampaDB(cards, products);


        System.out.println("\nC.");
        System.out.print("Il totale dei punti della Tessera N.0 è: ");
        System.out.println(puntiFedelta(cards, products, 0));

        System.out.println("\nD.");
        if (creaDuplicatoTessera(cards, products, 2)) {
            System.out.println("Le nuove tessere sono:");
            stampaDB(cards, products);
        }
        else{
            System.out.println("Nessuna duplicazione.");
        }
    }

}
```