Fondamenti di informatica 1 — A.A. 2000-2001
Appello del 16 luglio 2001
Tempo a disposizione: 60 minuti

RigheUguali — Analisi di un array di array

Cognome:_____ Nome: _____ Matricola:_____ Calcolatore: _____

Specifiche:

Nell'ambito della classe Java **RigheUguali**, definire i seguenti metodi di classe:

- un metodo **uguali** che, ricevendo come parametro una coppia **a** e **b** di array di interi (non nulli), verifica se gli array **a** e **b** sono tra loro uguali, e restituisce l'esito di tale verifica. (Due array sono uguali se hanno la stessa lunghezza e gli elementi in posizioni corrispondenti sono uguali.)
- un metodo **righeUguali** che, ricevendo come parametro una array di array di interi **a** (non nullo, e tale che ogni riga di **a** è un array non nullo) verifica se le righe di **a** sono tra di loro tutte uguali. (Un array di array vuoto è da considerarsi tale che le sue righe sono tra di loro tutte uguali.)

Esempi:

- Siano **a** e **b** gli array **{ 1, 2, 3 }** e **{ 1, 2, 3 }**; l'invocazione **uguali(a,b)** deve restituire il valore **true**
- siano **c** e **d** gli array **{ 1, 2, 3 }** e **{ 1, 2 }**; l'invocazione **uguali(c,d)** deve restituire il valore **false**
- siano **e** ed **f** gli array **{ 1, 2, 3 }** e **{ 1, 2, 4 }**; l'invocazione **uguali(e,f)** deve restituire il valore **false**
- sia **x** l'array di array **{ { 1, 2, 3 }, { 1, 2, 3 }, { 1, 2, 3 } }**; l'invocazione **righeUguali(x)** deve restituire il valore **true** — infatti, le tre righe di **x** sono tra di loro uguali
- sia **y** l'array di array **{ { 1, 2, 3 }, { 1, 2 }, { 1, 2, 3 } }**; l'invocazione **righeUguali(y)** deve restituire il valore **false** — infatti, sebbene la prima e la terza riga di **y** siano tra di loro uguali, la seconda riga di **y** è diversa sia dalla prima che dalla terza riga
- sia **v** l'array di array **{ { 1, 2, 3 } }**; l'invocazione **righeUguali(v)** deve restituire il valore **true**
- sia **w** l'array di array **{ { }, { } }** e **z** l'array di array **{ }**; le invocazioni **righeUguali(w)** e **righeUguali(z)** devono restituire il valore **true**

Che cosa deve essere fatto:

Sulla base delle specifiche dell'esercizio, definire la classe **RigheUguali** come segue:

- definire il metodo di classe **boolean uguali(int[] a, int[] b)** — si assumano **a** e **b** array non nulli;
- definire il metodo di classe **boolean righeUguali(int[][] a)** — si assuma **a** un array di array non nullo, in cui tutte le righe di **a** sono non nulle;
- definire una applicazione di test (definendo il metodo di classe **main**);
- definire, se necessario, altri metodi privati.

Verifica:

La correttezza del codice scritto deve essere verificata mediante compilazione ed esecuzione, utilizzando degli opportuni insiemi di dati di ingresso.

Ciascuno degli insiemi di dati di ingresso usato per la verifica deve essere documentato da una invocazione del metodo (visualizzando il risultato dell'invocazione del metodo insieme al valore atteso) nonché da un commento nel codice che descrive l'insieme di dati di ingresso e ne motiva la scelta. Ad esempio:

```
/* array di array composto da una sola riga */
System.out.println( righeUguali(new int[][] { { 1, 2, 3 } }) + " (true)" );
```

Fondamenti di informatica 1 — A.A. 2000-2001
Appello del 16 luglio 2001
Tempo a disposizione: 60 minuti

Ripetizioni — Analisi di una stringa

Cognome:_____ Nome: _____ Matricola:_____ Calcolatore: _____

Specifiche:

Nell'ambito della classe Java **Ripetizioni**, definire il seguente metodo di classe:

– un metodo **ripetizioni** che, ricevendo come parametri una stringa **s** (non nulla), un carattere **car** e un numero naturale **k**, verifica se **s** contiene almeno **k** occorrenze consecutive del carattere **car**.

Esempi:

Si consideri la stringa **s** = **"aabbcccaaa"**

– l'invocazione **ripetizioni(s, 'b', 2)** deve restituire **true** — infatti, **s** contiene due occorrenze consecutive del carattere **'b'**, a partire dalla posizione 2

– l'invocazione **ripetizioni(s, 'c', 2)** deve restituire **true** — infatti, **s** contiene almeno due occorrenze consecutive del carattere **'c'**, a partire dalla posizione 4 oppure dalla posizione 5

– l'invocazione **ripetizioni(s, 'c', 4)** deve restituire **false** — infatti, **s** contiene al massimo tre occorrenze consecutive del carattere **'c'**

– l'invocazione **ripetizioni(s, 'a', 4)** deve restituire **false** — infatti **s**, pur contenendo cinque occorrenze del carattere **'a'**, non ne contiene quattro consecutive

Che cosa deve essere fatto:

Sulla base delle specifiche dell'esercizio, definire la classe **Ripetizioni** come segue:

– definire il metodo di classe pubblico **boolean ripetizioni(String s, char car, int k)** — si assuma **s** non nulla e **k** maggiore o uguale a zero;

– definire una applicazione di test (definendo il metodo di classe **main**);

– definire, se necessario, altri metodi privati.

Suggerimento:

Scrivere anche un metodo che, ricevendo come parametri una stringa **s**, un carattere **car**, un naturale **k** e un indice **index**, verifica se la stringa **s** contiene **k** occorrenze consecutive del carattere **car** a partire dalla posizione **index**.

Che cosa è stato già fatto:

Della classe **String** è possibile usare SOLO i seguenti metodi d'istanza: **int length()** (che calcola la lunghezza della stringa) e **char charAt(int index)** (che restituisce il carattere di posizione **index** nella stringa).

Verifica:

La correttezza del codice scritto deve essere verificata mediante compilazione ed esecuzione, utilizzando degli opportuni insiemi di dati di ingresso.

Ciascuno degli insiemi di dati di ingresso usato per la verifica deve essere documentato da una invocazione del metodo (visualizzando il risultato dell'invocazione del metodo insieme al valore atteso) nonché da un commento nel codice che descrive l'insieme di dati di ingresso e ne motiva la scelta. Ad esempio:

```
/* solo gli ultimi k elementi di s sono uguali a car */
System.out.println( ripetizioni("abccc", 'c', 3) + " (true)" );
```