E17 – Esercizi sugli Array in Java

Esercizio 1 (esercizio 9.1 del libro di testo). Implementare un metodo statico di nome `stringaCorta` che prende come parametro un array di oggetti `String` e che restituisce la stringa dell'array avente lunghezza minima. Nel caso vi fossero due o più stringhe con la lunghezza minima, il metodo ne deve restituire una arbitrariamente.

Esercizio 2 (esercizio 9.2 del libro di testo). Implementare un metodo statico di nome `filtra` che prende come parametro un array `a` di numeri interi e un intero positivo `k`, e che restituisce un nuovo array `b` di numeri interi contenente i soli elementi di `a` che sono divisibili per `k`. Si faccia in modo che l'array `b` abbia una dimensione uguale al numero di elementi che deve contenere (non devono cioè esservi posizioni inutilizzate).

Esercizio 3 (esercizio 9.3 del libro di testo). Un oggetto della classe `AgendaTelefonica` permette di gestire una semplice agenda (rubrica) telefonica. Ogni utenza dell'agenda ha un nome, un cognome e un numero di telefono. L'agenda può gestire un numero limitato di utenze. Tale numero deve essere scelto all'atto della creazione di un oggetto della classe `AgendaTelefonica`. La classe ha il seguente scheletro:

```
public class AgendaTelefonica{
      // nome[i], cognome[i] e numTel[i] memorizzano
      // nome, cognome e telefono della i-esima utenza
      private String[] nome;
      private String[] cognome;
      private String[] numTel;

      /* costruttore: crea un'agenda che gestisce
         al massimo dim utenze
      */
      public AgendaTelefonica(int dim)

      /* se l'agenda non è piena, inserisce in agenda
         una nuova utenza con nome n, cognome c
         e numero di telefono t; poi restituisce true.
         Se l'agenda è piena restituisce false
      */
      public boolean inserisci(String n, String c, String t)

      /* restituisce il numero di telefono di un'utenza
         con nome n e cognome c, se una tale utenza è
         presente in agenda; altrimenti restituisce null
      */
      public String trova (String n, String c)

      /* visualizza sullo standard output la lista completa
         delle utenze in agenda
      */
      public void listaUtenze()
}
```

Scrivere il corpo dei metodi della classe `AgendaTelefonica`. Scrivere inoltre una classe di test che:

- Permette all'utente di creare un'agenda di dimensione a sua scelta;

- Mostra ripetutamente all'utente un menù dal quale sia possibile scegliere una tra le seguenti opzioni:
    1. Inserimento di una nuova utenza nell'agenda;
    2. Ricerca di una nuova utenza nell'agenda;
    3. Visualizzazione di tutte le utenze dell'agenda sullo standard output;
    4. Uscita dal menù.

Ogni volta che tale menù viene presentato all'utente, la classe di test dovrà effettuare sull'agenda l'azione corrispondente a quella che l'utente selezionerà, chiedendo ove necessario l'inserimento di ulteriori dati (ad esempio nel caso l'utente selezioni l'opzione 1 o 2). Se l'utente seleziona l'opzione 4, il test deve terminare.

Esercizio 4 (esercizio 9.6 del libro di testo). Scrivere un programma che prende sulla linea di comando (cioè all'atto del suo avvio) una sequenza di numeri reali e che stampa a video le seguenti informazioni: (i) la somma di tutti i numeri della sequenza; (ii) la media aritmetica esatta dei numeri della sequenza; (iii) il numero massimo e il numero minimo della sequenza.

Esercizio 5 (esercizio 9.8 del libro di testo). Scrivere un programma che fa inserire all'utente una matrice quadrata di stringhe a sua scelta, e che stampa a video la stringa della diagonale principale che risulta lessicograficamente più piccola. Si ricordi che la diagonale principale di una matrice quadrata di dimensione *n*, è l'insieme delle celle (0, 0), (1, 1), …, (*n*-1, *n*-1), cioè l'insieme delle celle (*i*, *j*) tali che *i* = *j*.

Esercizio 6 (esercizio 9.10 del libro di testo). Scrivere un semplice programma che fa inserire all'utente una matrice quadrata di interi e che dice all'utente se la matrice è triangolare superiore oppure no. Si ricordi che una matrice quadrata è triangolare superiore se tutti gli elementi al di sotto della diagonale principale sono 0.

Esercizio 7 (esercizio 9.12 del libro di testo). Un oggetto della classe `TabellaDiNomi` rappresenta una tabella di nomi di persona (stringhe). Lo scheletro della classe è il seguente:

```
public class TabelleDiNomi{
      // variabile che memorizza la tabella di nomi
      private String[][] tab;

      /* costruttore: crea un oggetto che rappresenta
         l'insieme delle stringhe contenute
         nell'array di array tabella
      */
      public TabellaDiNomi(String[][] tabella)

      /* restituisce il nome di lunghezza minima
         nell'intera matrice
      */
      public String nomeCorto()


      /* restituisce un array di interi che memorizza
         in ogni sua cella di indice i la lunghezza totale
         dei nomi della riga di indice i nella tabella
```

```
      */
      public int[] lunghTotRiga()

      /* restituisce il nome di lunghezza massima
         sulla colonna k specificata
      */
      public String nomeLungoNellaColonna(int k)
}
```

Completare l'implementazione della classe `TabellaDiNomi`, scrivendo il corpo del costruttore e dei metodi. Scrivere inoltre una classe di test che verifica il corretto funzionamento della classe `TabellaDiNomi`.

Soluzioni

Esercizio 1 – svolgimento.

```
public static String stringaCorta(String[] seq){
      String corta = seq[0];
      for (int i=1; i<seq.length; i++)
            if (seq[i].length() < corta.length())
                  corta = seq[i];
      return corta;
}
```

Esercizio 2 – svolgimento.

```
public static int[] filtra(int[] a, int k){
      // conta gli elementi da inserire nell'array restituito
      // e ne memorizza il numero in numElementi
      int numElementi = 0;
      for (int i=0; i<a.length; i++)
            if (a[i]%k == 0)
                  numElementi++;

      // crea l'array b da restituire e vi memorizza gli elementi
      // dell'array a che sono divisibili per k
      int[] b = new int[numElementi];
      int j = 0; // la prossima posizione libera nell'array b
      for (int i=0; i<a.length; i++)
            if (a[i]%k == 0){
                  b[j] = a[i];
                  j++;
            }
      return b;
}
```

Esercizio 3 – svolgimento. Si riporta di seguito il codice della classe `AgendaTelefonica` e di una classe di test.

```
public class AgendaTelefonica{
   // nome[i], cognome[i] e numTel[i] memorizzano
   // nome, cognome e telefono della i-esima utenza
   private String[] nome;
   private String[] cognome;
   private String[] numTel;

   /* costruttore: crea un'agenda che gestisce
      al massimo dim utenze
   */
   public AgendaTelefonica (int dim){
         this.nome = new String[dim];
         this.cognome = new String[dim];
         this.numTel = new String[dim];
      }


   /* se l'agenda non è piena, inserisce in agenda
```

```
      una nuova utenza con nome n, cognome c
      e numero di telefono t; poi restituisce true.
      Se l'agenda è piena restituisce false
   */
   public boolean inserisci (String n, String c, String t){
         boolean inserito = false;
         int i=0;
         while (!inserito && i<this.nome.length)
               if (this.nome[i]==null){
                        // trovata posizione vuota
                        this.nome[i]=n;
                        this.cognome[i]=c;
                        this.numTel[i]=t;
                        inserito = true;
               }
               else
                        i++;
         return inserito;
   }


   /* restituisce il numero di telefono di un'utenza
      con nome n e cognome c, se una tale utenza è
      presente in agenda; altrimenti restituisce null
   */
   public String trova (String n, String c){
         String numero = null;
         int i=0;
         while (numero==null && i<this.nome.length && this.nome[i]!=null)
               if (this.nome[i].equals(n) && this.cognome[i].equals(c))
                        numero = this.numTel[i];
               else
                        i++;
         return numero;
   }

   /* visualizza sullo standard output la lista completa
      delle utenze in agenda
   */
   public void listaUtenze (){
         int i=0;
         System.out.println ("LISTA COMPLETA DELLE UTENZE");
         System.out.println ();
         while (i<this.nome.length && this.nome[i]!=null){
               System.out.println ("Nome = " + this.nome[i]);
               System.out.println ("Cognome = " + this.cognome[i]);
               System.out.println ("Telefono = " + this.numTel[i]);
               System.out.println ();
               i++;
         }
   }
}

import fond.io.*;

public class ProvaAgendaTelefonica{

      public static void main (String[] args){
            InputWindow in = new InputWindow();
            int dim = in.readInt ("Creazione agenda - Inserisci dim. massima");
            AgendaTelefonica agenda = new AgendaTelefonica (dim);
            int opzione;
```

```
            do{
                  System.out.println ("\n1 - Inserisci utenza");
                  System.out.println ("2 - Ricerca utenza");
                  System.out.println ("3 - Visualizza lista utenze");
                  System.out.println ("4 - Termina\n");

                  opzione = in.readInt ("Scegliere opzione (1--4)");
                  if (opzione == 1){
                      String n = in.readString ("Inserisci nome");
                      String c = in.readString ("Inserisci cognome");
                      String t = in.readString ("Inserisci numero telefonico");
                      if (agenda.inserisci (n,c,t))
                           System.out.println ("Utenza inserita");
                      else
                           System.out.println ("Agenda piena");
                  }
                  else if (opzione == 2){
                        String n = in.readString ("Inserisci nome");
                        String c = in.readString ("Inserisci cognome");
                        String num = agenda.trova (n,c);
                        if (num != null)
                             System.out.println ("Numero di Telefono = " + num);
                        else
                             System.out.println ("Utenza inesistente");
                  }
                  else if (opzione == 3){
                        agenda.listaUtenze();
                  }
                  else if (opzione == 4){
                        System.out.println ("Applicazione terminata");
                  }
                  else
                       System.out.println ("Opzione non valida, ripeti scelta");
            }while (opzione != 4);
      }
}
```

Esercizio 4 – svolgimento.

```
public class Esercizio4{
      public static void main(String[] args){
            double[] seq = new double[args.length];
            for (int i=0; i<seq.length; i++)
                  seq[i] = Double.parseDouble(args[i]);
            double somma, media, min, max;
            min = max = somma = seq[0];
            for (int i=1; i<seq.length; i++){
                  somma += seq[i];
                  if (seq[i] < min)
                        min = seq[i];
                  if (seq[i] > max)
                        max = seq[i];
            }
            media = somma/seq.length;
            System.out.println("Somma = " + somma);
            System.out.println("Media = " + media);
            System.out.println("Min   = " + min);
            System.out.println("Max   = " + max);
      }
}
```

Esercizio 5 – svolgimento.

```
import fond.io.*;

public class Esercizio5{
      public static void main(String[] args){
            InputWindow in = new InputWindow();
            int n = in.readInt("Dimensione della matrice?");
            String[][] mat = new String[n][n];

            // inserimento delle stringhe della matrice
            for (int i=0; i<mat.length; i++)
                  for (int j=0; j<mat[i].length; j++)
                        mat[i][j] = in.readString("Stringa in posizione
                                                ("+i+","+j+")?");

            // calcolo della stringa lessicograficamente
            // più piccola sulla diagonale principale
            String piccola = mat[0][0]; // inizializzazione
            for (int i=1; i<mat.length; i++)
                  if (piccola.compareTo(mat[i][i]) > 0)
                        piccola = mat[i][i];
            System.out.println("Stringa lessicograf. minore sulla diagonale = "
                                                            + piccola);
      }
}
```

Esercizio 6– svolgimento. Per verificare che se la matrice è triangolare superiore, dobbiamo controllare, per ogni riga *i*, gli elementi in posizione (*i, j*) tali che $0 < j < i$, cioè quelli che sono sotto alla diagonale principale. Se anche solo uno di questi elementi è diverso da zero, possiamo concludere che la matrice non è triangolare superiore, ed arrestare il controllo dei rimanenti elementi. Ciò è fatto tramite due cicli annidati, la cui condizione di uscita è una condizione composta.

```
import fond.io.*;

public class Esercizio6{
      public static void main(String[] args){
            InputWindow in = new InputWindow();
            int n = in.readInt("Dimensione matrice?");
            int[][] mat = new int[n][n];

            // inserimento elementi nella matrice
            for (int i=0; i<mat.length; i++)
                  for (int j=0; j<mat[i].length; j++)
                        mat[i][j]=in.readInt("Elemento in posizione
                                                "+i+","+j+")?");
            // verifica se la matrice è triangolare superiore
            boolean triangolare = true;
            for (int i=0; triangolare && i<mat.length; i++)
                  for (int j=0; triangolare && j<i; j++)
                        if (mat[i][j]!=0)
                              triangolare = false;
            if (triangolare)
                  System.out.println("La matrice e' triangolare superiore");
            else
                  System.out.println("La matrice non e' triangolare superiore");
      }
}
```

Esercizio 7– svolgimento. Viene di seguito riportata una implementazione per la classe `TabellaDiNomi` e il codice di una possibile classe di test.

```
public class TabellaDiNomi{
	// variabile che memorizza la tabella di nomi
	private String[][] tab;

	/* costruttore: crea un oggetto che rappresenta
	 l'insieme delle stringhe contenute
	 nell'array di array tabella
	*/
	public TabellaDiNomi(String[][] tabella){
		int m = tabella.length;
		int n = tabella[0].length;
		this.tab = new String[m][n];
		for (int i=0; i<m; i++)
			for (int j=0; j<n; j++)
				this.tab[i][j] = tabella[i][j];
	}

	/* restituisce il nome di lunghezza minima
	   nell'intera matrice
	*/
	public String nomeCorto(){
		String corto = this.tab[0][0];
		for (int i=0; i<this.tab.length; i++)
			for (int j=0; j<this.tab[i].length; j++)
				if (this.tab[i][j].length() < corto.length())
					corto = this.tab[i][j];
		return corto;
	}


	/* restituisce un array di interi che memorizza
	 in ogni sua cella di indice i la lunghezza totale
	 dei nomi della riga di indice i nella tabella
	*/
	public int[] lunghTotRiga(){
		int[] lung = new int[this.tab.length];
		for (int i=0; i<this.tab.length; i++){
			lung[i]=0;
			for (int j=0; j<this.tab[i].length; j++)
				lung[i] += this.tab[i][j].length();
		}
		return lung;
	}

	/* restituisce il nome di lunghezza massima
	   sulla colonna k specificata
	*/
	public String nomeLungoNellaColonna(int k){
		String lungo = this.tab[0][k];
		for (int i=0; i<this.tab.length; i++)
			if (lungo.length() < this.tab[i][k].length())
				lungo = this.tab[i][k];
		return lungo;
	}
}
```

```
import fond.io.*;

public class TestTabellaDiNomi{
      public static void main(String[] args){
            InputWindow in = new InputWindow();
            int m = in.readInt("Numero di righe?");
            int n = in.readInt("Numero di colonne?");
            String[][] mat = new String[m][n];

            // inserimento dei nomi in una matrice
            for (int i=0; i<mat.length; i++)
                  for (int j=0; j<mat[i].length; j++)
                        mat[i][j]=in.readString("Nome in posizione
                                                      "+i+","+j+")?");

            // creazione di un oggetto TabellaDiNomi
            TabellaDiNomi t = new TabellaDiNomi(mat);

            OutputWindow out = new OutputWindow(); // per stampare l'output

            // test del metodo nomeCorto()
            out.writeln("Nome più corto: " + t.nomeCorto());

            // test del metodo lunghTotRiga()
            int[] lung = t.lunghTotRiga();
            for (int i=0; i<lung.length; i++)
                  out.writeln("Numero caratteri riga "+i+"="+lung[i]);

            // test del metodo nomeLungoNellaColonna(int k)
            // il test esegue il metodo per ogni possibile colonna,
            // ricordando che le dimensioni della matrice rappresentata
            // dall'oggetto t sono le stesse di mat
            for (int j=0; j<mat[0].length; j++){
                  out.write("Nome più lungo nella colonna "+j+": ");
                  out.writeln(t.nomeLungoNellaColonna(j));
            }
      }
}
```