Laboratorio di Programmazione
Laurea in Bioinformatica

II Quadrimestre

20 febbraio 2008

1 JAVA: il tipo char e il costrutto switch

Esercizio Q2_1

La classe CharCfr contenga un programma che:

1. Dichiara due variabili c e d di tipo char e ne acquisisce i valori da tastiera;
2. A fronte di opportuni confronti stampa una o più frasi fra le seguenti:
   "c e d contengono lo stesso carattere <valore di c,d> (codice Unicode: <unicode di c,d>)"
   "c e d contengono caratteri differenti"
   "nella codifica Unicode il carattere <c> (<unicode di c>) precede <d> (<unicode di d>)"
   "nella codifica Unicode il carattere <c> (<unicode di c>) precede immediatamente <d> (<unicode di d>)"

```
import prog.io.*;

class CharCfr {
    public static void main(String[] args) {

    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    char c,d;
    c=in.readChar("Inserire il primo carattere: ");
    d=in.readChar("Inserire il secondo carattere: ");

    if(c==d)
        out.println("c e d contengono lo stesso carattere "+c+" (codice Unicode: "+(int)c+" )");
    else {
        out.println("c e d contengono caratteri differenti");
        if(c==d-1) {
            out.println("e in Unicode "+c+" ("+(int)(c)+") precede immediatamente "+d+" ("+(int)(d)+")");
            } else {
                if(c<d) out.println("e in Unicode "+c+" ("+(int)(c)+") precede "+d+" ("+(int)(d)+")");
            }
        }
    }
}
```

Esercizio Q2_2

Si scriva un'applicazione per la simulazione di una semplice calcolatrice. Il programma deve scrivere il risultato dopo aver ricevuto da tastiera due numeri e il simbolo dell'operazione desiderata.

```
import prog.io.*;

class Calcolatrice {
    public static void main(String[] args) {

    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    int a=in.readInt("Inserire il primo operando: ");
    int b=in.readInt("Inserire il secondo operando: ");

    double risultato;
    char c=in.readChar("Inserire il tipo di operazione (+,-,*,/,%): ");

    switch (c) {
        case '+':
            risultato = a+b;
            out.println("Risultato: "+risultato);
            break;
        case '*':
            risultato = a*b;
            out.println("Risultato: "+risultato);
            break;
        case '-':
            risultato = a-b;
            out.println("Risultato: "+risultato);
            break;
        case '/':
            risultato = (double)a/b;
            out.println("Risultato: "+risultato);
            break;
        case '%':
            risultato = a%b;
            out.println("Risultato: "+risultato);
            break;
        default:
            out.println("operazione non valida");
        }
    }
}
```

Esercizio Q2_3

Si scriva un'applicazione che stampa tutte le lettere minuscole dell'alfabeto facendo uso dell'operatore postfisso di incremento. Nel codice appena scritto si sostituisca all'operatore postfisso l'operatore prefisso e si osservi il risultato.

```
import prog.io.*;

class StampaAlfab {
    public static void main(String[] args) {

    ConsoleOutputManager out = new ConsoleOutputManager();
    char i='a';
    do {
        out.print(i++ +",");
    } while(i<='z');

    out.println("...");
    }
}
```

Esercizio Q2_4

Si scriva un'applicazione che permette di specificare somme di numeri interi inserendo da tastiera stringhe come "1+13+6+123". [sugg.: si ricorra ai metodi della classe String che permettono di individuare l'indice di un carattere (+ in questo caso) e di estrarre sottostringhe]

```
import prog.io.*;

class Calcolatrice2{
    public static void main(String[] args){

        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out = new ConsoleOutputManager();

        out.println("Somma di interi ");
        String st1;
        int fromIndex=0;
        st1 = in.readLine("Inserire degli interi separati dal +: ");
        int ind;
        String auxst;
        int somma=0;
        while((ind=st1.indexOf("+",fromIndex))>0){
            //out.println("trovato un più in posizione: "+ind);
            auxst=st1.substring(fromIndex,ind);
            somma+=Integer.parseInt(auxst);
            fromIndex=ind+1;
            }

        auxst=st1.substring(fromIndex,st1.length());
        somma+=Integer.parseInt(auxst);
        out.println("Somma: "+somma);
    }
}
```

2 JAVA: metodi e campi statici

Esercizio Q2_5

Si scriva un programma per la risoluzione delle equazioni di secondo grado
[sugg. si usi il metodo sqrt della class Math per il calcolo della radice quadrata]

```
import prog.io.*;
class EqSecondoGrado {
    public static void main(String[] args) {

    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();
    double a,b,c;

    a=in.readDouble("Inserisci il coeff. a: ");
    b=in.readDouble("Inserisci il coeff. b: ");
    c=in.readDouble("Inserisci il coeff. c: ");

    double x1=(-b+Math.sqrt(b*b-4.0*a*c))/(2.0*a);
    double x2=(-b-Math.sqrt(b*b-4.0*a*c))/(2.0*a);
    out.println("x1: "+x1);
    out.println("x2: "+x2);
    }
}
```

Esercizio Q2_6

Si consideri un cerchio di raggio r il cui centro coincide con l'origine di una coppia di assi cartesiani. Si scriva un programma che:

1. generando una sequenza di N punti a caso di coordinate x e y comprese fra 0 ed r calcoli la frequenza dei punti che cadono nel quarto di cerchio di circonferenza (cioè il rapporto fra i punti che cadono nel quarto di circonferenza e il numero totale di punti generati)
2. usando il metodo al punto precedente calcolare un'approssimazione di π sapendo che la frequenza dei punti che cadono nel quarto di cerchio di circonferenza costituisce un'approssimazione del rapporto fra l'area A_q del quarto di cerchio e l'area del quadrato di lato r. Si ricordi poi che e' $4 * Aq = \pi * r^2$.
3. si confronti il valore stimato di π con il valore della costante statica Math.PI .

```
import prog.io.*;
class AreaCerchioePI {
    public static void main(String[] args) {

    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    double randx, randy;
    double r=2.5;
    int contatore=0;
    double freq;
    double stimaPI;

    int N=in.readInt("Inserisci il n. di campioni: ");

    for(int i=0; i<N; i++){

        randx=r*Math.random();
        randy=r*Math.random();

        if((randx*randx+randy*randy)<r*r)
            contatore+=1;
    }
```

```
    freq = (double)contatore/N;
    out.println("Stima dell'area: "+((r*r)*freq*4.0));
    out.println("Area reale: "+(Math.PI*r*r));

    stimaPI = 4.0*freq;
    out.println("Stima di PI: "+stimaPI);
    out.println("PI in Math: "+Math.PI);
    }
}
```

3 JAVA: array e collezioni

Esercizio Q2_7

La classe OrdinaStringhe contenga un programma che:

1. inizializza un array di stringhe acquisite da tastiera
2. ordina l'array per lunghezza crescente delle stringhe, adoperando una variabile di tipo stringa come contenitore temporaneo
3. stampa l'array così ordinato

```
import prog.io.*;

class OrdinaStringhe {
    public static void main(String[] args) {

    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    final int MAX = 5;
    String[] arrStr = new String[MAX];
    String temp;

    for(int i=0; i<MAX; i++)
        arrStr[i] = in.readLine("Inserire la stringa n. " + (i+1) + ": ");

    for(int i=0; i<MAX; i++)
        for(int j=i+1; j<MAX; j++)
        if (arrStr[i].length() > arrStr[j].length()) {
            temp = arrStr[i];
            arrStr[i] = arrStr[j];
            arrStr[j] = temp;
        }

    int counter = 1;
    for (String s : arrStr)
        out.println("Locazione " + counter++ + ": " + s);
    }
}
```

Esercizio Q2_8

La classe OrdinaStringhe2 contenga un programma che:

1. inizializza un array di stringhe acquisite da tastiera
2. ordina gli elementi dell'array per lunghezza crescente delle stringhe in un nuovo array, senza adoperare alcun contenitore temporaneo
3. stampa il nuovo array

[SUGG.: copiare nel secondo array e poi escludere la stringa più corta del primo array via via fino a suotare il primo array]

```
import prog.io.*;

class OrdinaStringhe2 {
    public static void main(String[] args) {

    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();

    final int MAX = 5;
    String[] arrStr = new String[MAX];
    String[] ordStr = new String[MAX];
```

```
    for(int i=0; i<MAX; i++)
        arrStr[i] = in.readLine("Inserire la stringa n. " + (i+1) + ": ");

    int lunghMax;

    for(int i=0; i<MAX; i++) {

        lunghMax = i;            // assumo la stringa più corta inizialmente in posizione i

        for(int j=i; j<MAX; j++)       // scorro il vettore di partenza dalla posizione i
        if (arrStr[j].length() < arrStr[lunghMax].length())
            lunghMax = j;          // aggiorno la posizione della stringa più corta

        ordStr[i] = arrStr[lunghMax];  // copio il riferimento alla stringa più corta
        arrStr[lunghMax] = arrStr[i];  // sovrascrivo l'elemento copiato

        out.println("Locazione " + (i+1) + ": " + ordStr[i]);
    }
    }
}
```

4 JAVA: array di array

Esercizio Q2_9

La classe OpMatrici contenga un programma che:

1. crea due matrici 2x2, la prima con righe [1 2] e [3 4], la seconda con righe [5 6] e [7 8].
2. esegue la visita completa delle due matrici, prima per RIGA e poi per COLONNA, stampandone il risultato.
3. ne esegue il prodotto matriciale. Per verifica, si noti che il risultato corretto e' una matrice 2x2 con righe [19 22] e [43 50].

```
import prog.io.*;

class OpMatrici {
    public static void main(String[] args) {
        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out = new ConsoleOutputManager();

        int size1 = 2;
        int size2 = 2;
        int[][] Matr1 = {{1,2},{3,4}};
        int[][] Matr2 = {{5,6},{7,8}};

        out.println("=======Visita per righe=============");
        for(int i=0; i<size1; i++)
            for(int j=0; j<size2; j++)
                out.print(Matr1[i][j]+"\t");
        for(int i=0; i<size1; i++)
            for(int j=0; j<size2; j++)
                out.print(Matr2[i][j]+"\t");

        out.println("\n=======Visita per colonne=============");
        for(int i=0; i<size2; i++)
            for(int j=0; j<size1; j++)
                out.print(Matr1[j][i]+"\t");
        for(int i=0; i<size1; i++)
            for(int j=0; j<size2; j++)
                out.print(Matr2[j][i]+"\t");

        out.println("\n=======Prodotto vettoriale============");
        int[][] ProdMatr=new int[size1][size2];
        for(int i=0; i<size1; i++){
            for(int j=0; j<size2; j++){
                ProdMatr[i][j]=0;
                for(int k=0; k<size2; k++)
                    ProdMatr[i][j] += Matr1[i][k]*Matr2[k][j];
                out.print(ProdMatr[i][j]+"\t");
            }
            out.print("\n");
        }
    }
}
```

Esercizio Q2_10

La classe LetturaVerticale contenga un programma che:

1. legge e salva in un array di stringhe una sequenza di stringhe da input fintantoché non viene immessa la stringa "FINE"
2. colloca le stringhe in una matrice di simboli avente un numero di colonne pari alla lunghezza della stringa minore, in cui la riga n-esima contiene i simboli della stringa n-esima
3. presenta a video una nuova sequenza, formata leggendo la matrice colonna dopo colonna.

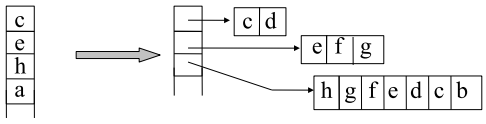
Si risolva il problema prima adoperando un array di array di caratteri, poi lavorando col solo array di stringhe iniziale (ovvero senza creare la matrice di caratteri)

```
import prog.io.*;
class LetturaVerticale {
    public static void main(String[] args) {
        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out = new ConsoleOutputManager();
        final int MAX = 1000;
        String[] aux = new String[MAX];
        String s;
        char c;
        int righe;

        for(righe=0; righe<MAX-1; righe++) {
            s = in.readLine("Immetti stringa n." + (righe+1) + " [FINE per terminare]: ");
            if(s.equals("FINE"))
                break;
            else
                aux[righe] = s;
        }

        String[] arrString = new String[righe];
        for(int i=0; i<righe; i++)
            arrString[i] = aux[i];
        int minimo = Integer.MAX_VALUE;

        for(String t : arrString)
            minimo = (t.length() < minimo ? t.length() : minimo);

        // Soluzione con array di array
        char[][] arrChar = new char[righe][minimo];
        for (int i=0; i<righe; i++)         // crea matrice
          for (int j=0; j<minimo; j++)
            arrChar[i][j] = arrString[i].charAt(j);

        for (int j=0; j<minimo; j++) {      // stampa matrice
            for (int i=0; i<righe; i++)
                out.print(arrChar[i][j]);
            out.println();
        }

        // Soluzione con array di stringhe
        for (int j=0; j<minimo; j++) {      // stampa array
            for (int i=0; i<righe; i++)
                out.print(arrString[i].charAt(j));
            out.println();
        }
    }
}
```

Esercizio Q2_11

La classe CaratteriNelMezzo contenga un programma che:

1. inizializza un array di 10 simboli UNICODE scelti a caso tra i codici decimali 1 e 150 adoperando il metodo statico random() di Math
2. sostituisce col carattere 'a' gli elementi nell'array che non sono alfabetici minuscoli
3. inserisce carattere 'z' nell'ultimo elemento dell'array, e infine stampa gli elementi dell'array risultante
4. per ogni elemento dell'array escluso l'ultimo, costruisce un nuovo array che contiene tutti i caratteri intermedi tra quello contenuto in quell'elemento (compreso) e quello contenuto nell'elemento di indice superiore (escluso), in avanti o all'indietro a seconda che l'elemento di indice superiore contenga un carattere rispettivamente posteriore o anteriore nell'ordinamento fissato dal codice UNICODE. Ad esempio:



5. stampa, riga dopo riga, ciascuna lista di caratteri contenuti nei nuovi array così definiti

```
import prog.io.*;
class CaratteriNelMezzo {
    public static void main(String[] args) {
        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out = new ConsoleOutputManager();

        final int MAX = 10;
        char[] arrChar = new char[MAX];
        int distanza;

        for(int i=0; i<MAX-1; i++) {
            arrChar[i] = (char)(1+ Math.random()*150);
            if(arrChar[i]<'a' || arrChar[i]>'z')
                arrChar[i] = 'a';
        }

        arrChar[MAX-1] = 'z';
        for (char c : arrChar)
            out.println(c);

        char[][] arrMezzo = new char[MAX-1][];

        for(int i=0; i<MAX-1; i++)
            if(arrChar[i+1]-arrChar[i] >0) {
                distanza = arrChar[i+1]-arrChar[i];
                arrMezzo[i] = new char[distanza];
                for(int j=0; j<distanza; j++)
                    arrMezzo[i][j] = (char)(arrChar[i] + j);
                }
                else {
                    distanza = arrChar[i]-arrChar[i+1];
                    arrMezzo[i] = new char[distanza];
                    for(int j=0; j<distanza; j++)
                        arrMezzo[i][j] = (char)(arrChar[i] - j);
                }

        for(int i=0; i<MAX-1; i++) {
           for (char c : arrMezzo[i])
                out.print(c);
           out.println();
        }
    }
}
```

5 JAVA: progettazione di classi

Esercizio Q2_12

Si progetti una classe di nome Rettangolo, la quale istanzia oggetti in grado di rappresentare rettangoli mediante i loro valori di base b e altezza h. La classe contenga:

1. un costruttore Rettangolo (double b, double h), il quale costruisce un rettangolo di base b e altezza h
2. un costruttore Rettangolo (double b), il quale costruisce un rettangolo di base e altezza b (ovvero un quadrato)
3. un metodo double area(), che restituisce l'area del rettangolo
4. un metodo double perimetro(), che restituisce il perimetro del rettangolo
5. un metodo double diagonale(), che restituisce la diagonale del rettangolo
6. un metodo double diagonale(), che restituisce la diagonale del rettangolo
7. un metodo String toString(), che restituisce una rappresentazione del rettangolo nel formato

```
 --------------------------
|                          |
|                          |
 --------------------------
```

   ben dimensionata limitatamente all'approssimazione ammessa (per semplicità si consideri l'altezza del prompt uguale al doppio della base)
8. un metodo main che, dato un valore di partenza del perimetro, individua il rettangolo che a parità di perimetro possiede area massima al variare di base e altezza e di conseguenza ne stampa i valori di base, altezza e area nonchè una sua rappresentazione. Per determinare l'area massima il metodo adopererà una procedura numerica che fa una ricerca di massimo partendo da 10000 valori scelti a caso della base compatibili col perimetro dato.

```
import prog.io.*;

class Rettangolo {
    private double base, altezza;

    public Rettangolo(double b, double h) {
        base = b;
        altezza = h;
    }

    public Rettangolo(double l) {
        this(l,l);
    }

    public double area() {
        return base*altezza;
    }

    public double perimetro() {
        return 2*(base+altezza);
    }

    public double diagonale() {
        return Math.sqrt(base*base + altezza*altezza);
    }

    public String toString() {
        final double H_OVER_B = 2;           // rapporto altezza/base del prompt
        String rigaOrizz, spazioOrizz, figuraIntera;

        rigaOrizz = " ";
        spazioOrizz = "|";

        for(int i=1; i<(int)base; i++) {
            rigaOrizz += "-";
            spazioOrizz += " ";
        }
```

```
        rigaOrizz += " \n";
        spazioOrizz += "|\n";

        figuraIntera = rigaOrizz;
        for(int i=1; i<(int)(altezza/H_OVER_B); i++)
            figuraIntera += spazioOrizz;
        figuraIntera += rigaOrizz;

        return figuraIntera;
    }

    public static void main(String[] args) {

    final int MAX_ITER = 10000;
    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out = new ConsoleOutputManager();
    double perimetro = in.readDouble("Perimetro: ");
    double b, h, area;
    double bMax=0, hMax=0, areaMax=0;
    Rettangolo r;
    Rettangolo rMax = new Rettangolo(0);

    for (int i=0; i<MAX_ITER; i++) {
        b = Math.random() * perimetro/2;   // massimo valore della base: perimetro/2
        h = perimetro/2 - b;
        r = new Rettangolo(b,h);
        area = r.area();

        if (area>areaMax) {
            bMax = b;
            hMax = h;
            areaMax = area;
            rMax = r;
        }
    }

    out.println("Base massima:" + bMax);
    out.println("Altezza massima:" + hMax);
    out.println("Area massima:" + areaMax);
    out.println("Rappresentazione del rettangolo:\n" + rMax.toString());
    }
}
```

Esercizio Q2_13

Si progetti una classe di nome Carte Francesi, i cui oggetti modellano il mazzo di 52 carte francesi. Per modellare il mazzo si consiglia di adoperare un array di interi carte[i] di lunghezza 52, in cui ciascun elemento contiene una carta rappresentata da un numero n che va da 1 a 52. La carta corrispondente a un numero n si ottiene ordinando i quattro semi nell'ordine: cuori, quadri, picche, fiori. Per esempio, i numeri da 1 a 13 corrispondono al seme di cuori, e così via.

La classe contenga:

1. un costruttore CarteFrancesi(), il quale istanzia un mazzo attribuendo un valore a ciascun elemento del campo carte[]
2. un metodo statico private static String toString(int n), utile alla classe per stampare la carta n in un formato esplicativo per l'utente (es.: la carta di valore 25 verrà codificata nella stringa regina di quadri)
3. un metodo public String vediTesta, che visualizza la carta in testa al mazzo
4. un metodo public void spostaTesta(), che sposta in fondo al mazzo la carta presente in testa al mazzo
5. un metodo public void mescola(int volte), che rimescola il mazzo un numero di volte come da parametro dato. Il rimescolamento avvenga nel modo seguente: un mazzetto contenente un numero casuale di carte viene estratto dalla testa del mazzo; il mazzetto è reinserito in coda al mazzo in modo da alternare, partendo dalla coda, una carta presa dal mazzetto con una carta esistente nel mazzo

La classe dovrà far parte di un package di nome myclasses accessibile da ambiente Java.

Si progetti infine la classe UsaCarte contenente un metodo main che, appoggiandosi alle risorse della classe CarteFrancesi contenuta nel package myclasses, simula il gioco delle due carte. Il giocatore che dà le carte rimescoli 20 volte il mazzo secondo la regola di mescolamento fornita dalla classe.

```
package myclasses;

public class CarteFrancesi {
    private static final int NUM_CARDS = 52;
    private int[] carte = new int[NUM_CARDS];

    public CarteFrancesi() {
        for(int i=0; i<NUM_CARDS; i++)
            carte[i] = i+1;
    }

    public String vediTesta() {                    // stampa la testa del mazzo
       return toString(carte[0]);
    }

    public void spostaTesta() {                    // sposta la prima carta in fondo
        int temp = carte[0];
        for(int i=0; i<NUM_CARDS-1; i++)
            carte[i] = carte[i+1];
        carte[NUM_CARDS-1] = temp;
    }

    public void mescola(int volte) {
        int spessore_mazzetto;
        int[] mazzetto;
        for (int i=0; i<volte; i++) {
            spessore_mazzetto = 1+(int)(NUM_CARDS/2*Math.random());  // sceglie lo spessore del mazzetto (max metà mazzo)
            mazzetto = new int[spessore_mazzetto];

            for(int j=0; j<spessore_mazzetto; j++)                // estrae il mazzetto dalla testa del mazzo
                mazzetto[j] = carte[j];

            for(int j=0; j<NUM_CARDS-spessore_mazzetto; j++)      // ridetermina la testa del mazzo
                carte[j] = carte[j+spessore_mazzetto];

            for(int j=1; j<=spessore_mazzetto; j++) {              // reinserisce il mazzetto in coda alternando le carte
                carte[NUM_CARDS-2*j+1] = mazzetto[spessore_mazzetto-j];
                carte[NUM_CARDS-2*j] = carte[NUM_CARDS-spessore_mazzetto-j];
            }
        }
    }
```

```
    private static String toString(int num) {
        String s;
        if (num == 0)
           s = "none";
        else {
            switch (num % (NUM_CARDS/4)) {
                case 1:
                    s = "asso di ";
                    break;
                case 11:
                    s = "jack di ";
                    break;
                case 12:
                    s = "regina di ";
                    break;
                case 0:
                    s = "re di ";
                    break;
                default:
                    s=(num % (NUM_CARDS/4)) + " di ";
            }

            switch ((num-1) / (NUM_CARDS/4)) {
                case 0:
                    s += "cuori";
                    break;
                case 1:
                    s += "quadri";
                    break;
                case 2:
                    s += "fiori";
                    break;
                case 3:
                    s += "picche";
            }
    }
    return s;
    }
}
```

```
import prog.io.ConsoleOutputManager;
import myclasses.CarteFrancesi;

class UsaCarte {
    public static void main(String[] args) {

        ConsoleOutputManager out = new ConsoleOutputManager();
        CarteFrancesi carte = new CarteFrancesi();

        carte.mescola(20);
        out.println("Il giocatore 1 estrae: " + carte.vediTesta());
        carte.spostaTesta();
        out.println("Il giocatore 2 estrae: " + carte.vediTesta());
        carte.spostaTesta();
    }
}
```

6 JAVA: ricorsione

Esercizio Q2_14

Si scriva una classe che dato un intero k calcoli, sia in versione iterativa che in versione ricorsiva, il k-mo elemento della seguente serie (serie di Fibonacci): 1,1,2,3,5,8,13,21,... La serie è tale che il primo e il secondo numero sono uguali a 1, mentre ciascun elemento successivo è uguale alla somma dei due numeri che lo precedono.

```
import prog.io.*;

public class Ric_Fibonacci {

    public static int fibonacciIter(int k){
        int n, temp1, temp2;
        n=1;
        if (k>1){
            temp1=temp2=1;
            for (int i=0; i<k-1; i++){
                n=temp2+temp1;
                temp2=temp1;
                temp1=n;
            }
        }
        return n;
    }

    public static int fibonacciRicor(int k){
        int n, temp1, temp2;
        n=1;
        if (k>1)
            n=fibonacciRicor(k-1)+fibonacciRicor(k-2);
        return n;
    }

    public static void main(String[] args) {
        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out= new ConsoleOutputManager();

        int k=in.readInt("Digita un intero positivo: ");

        out.println("Il corrispondente termine di Fibonacci(iterativo): "+fibonacciIter(k));
        out.println("Il corrispondente termine di Fibonacci(ricorsivo): "+fibonacciRicor(k));
    }
}
```

Esercizio Q2_15

Si scriva una classe che si serve di metodi ricorsivi per il calcolo del fattoriale di un numero intero n inserito dall'utente e per il calcolo della potenza m^n, dove m è un secondo numero intero fornito dall'utente.

```
import prog.io.*;
public class Ric_PotenzaFattoriale {

    public static int potenzaRicor(int m, int n){
        if (n==0)
            return 1;
    else
            return m*potenzaRicor(m,n-1);
    }

    public static int fattorialeRicor(int n){
        if (n<2)
            return 1;
```

```
    else
            return n*fattorialeRicor(n-1);
    }

    public static void main(String[] args) {
        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out= new ConsoleOutputManager();

        int n=in.readInt("Fornisci un valore n per il calcolo di n!: ");
        out.println("Il valore di n! e': "+fattorialeRicor(n));

        int m=in.readInt("Fornisci un valore m per il calcolo di m^n: ");
        out.println("Il valore di m^n e': "+potenzaRicor(m,n));
    }
}
```

Esercizio Q2_16

Si scriva una classe che si serve di un metodo ricorsivo per la stampa invertita di una stringa fornita dall'utente. Il metodo ricorsivo sia del tipo:

```
 public static void stampainvert(String s, int i, ConsoleOutputManager out)
```

dove i parametri passati sono la stringa da stampare, un indice che determina l'indice del carattere da cui iniziare a stampare, e un oggetto di tipo ConsoleOutputManager per la stampa.

```
import prog.io.*;
public class Ric_InvertiStringa {

    public static void stampainvert(String s, int i, ConsoleOutputManager out){
        int lunghezza=s.length();
        if (i<lunghezza-1)
            stampainvert(s,i+1,out);
        out.print(s.charAt(i));
    }

    public static void main(String[] args) {
        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out= new ConsoleOutputManager();

        String stringa = in.readLine("Immetti una stringa: ");

        stampainvert(stringa,0,out);
    }
}
```

Esercizio Q2_17

Si scriva una classe che si serve di un metodo ricorsivo per ordinare un array di interi in [0,9] fornito dall'utente. Il metodo ricorsivo implementi la seguente strategia in tre passi (algoritmo "mergesort"):

1. Divide l'array non ordinato in due sotto-array di uguale lunghezza (a meno di uno per lunghezze dispari)
2. Ordina ricorsivamente ognuno dei due sotto-array fino fino a ottenere il caso di lunghezza 1 (in questo caso viene restituito l'array stesso)
3. Ricompone, attraverso un metodo merge di servizio, i due sotto-array ordinati in un unico array ordina

```
import prog.io.*;
public class Ric_MergeSort {

    public static int[] mergesort(int[] m){
```

```
    int lunghezza=m.length;
    if (lunghezza<2)
        return m;
    else
    {
        int mezzo=lunghezza/2;
        int[] sx=new int[mezzo];
        for (int i=0; i<mezzo; i++)
            sx[i]=m[i];
        int[] dx=new int[lunghezza-mezzo];
        for (int i=mezzo; i<lunghezza; i++)
            dx[i-mezzo]=m[i];

        sx=mergesort(sx);
        dx=mergesort(dx);

        return merge(sx,dx);
    }
}

public static int[] merge(int[] a, int[] b){
    int lun_a=a.length;
    int lun_b=b.length;
    int[] v=new int[lun_a+lun_b];
    int i_a=0;
    int i_b=0;
    int j=0;

    while(i_a<lun_a && i_b<lun_b){
        if(a[i_a]<=b[i_b]){
            v[j]=a[i_a];
            i_a+=1;
        }
        else{
            v[j]=b[i_b];
            i_b+=1;
        }
        j+=1;
    }

    while(i_a<lun_a){
        v[j]=a[i_a];
        i_a++;
        j++;
    }

    while(i_b<lun_b){
        v[j]=b[i_b];
        i_b++;
        j++;
    }

    return v;
}

public static void main(String[] args) {
    ConsoleInputManager in = new ConsoleInputManager();
    ConsoleOutputManager out= new ConsoleOutputManager();

    String stringa = in.readLine("Immetti una sequenza di interi in [0,9] terminata da Invio: ");

   int lun=stringa.length();
   int seqint[]=new int[lun];
   for (int i=0;i<lun; i++)
        seqint[i]=(int)stringa.charAt(i)-'0';
```

```
        int[] seqord=new int[lun];
        seqord=mergesort(seqint);

        for (int i=0;i<lun; i++)
             out.print(seqord[i]);
    }
}
```

7 PERL: variabili numeriche e stringhe

Esercizio Q2_18

Scrivere un programma in PERL cha calcola l'area di un triangolo di base 3 e altezza 7, e stampa il risultato a schermo.

```
$base = 3;
$altezza = 7;

print "Calcolo dell'area di un triangolo\n";
print "di base $base e altezza $altezza\n";

$area = $base*$altezza/2;

print "Area = $area\n";
```

Esercizio Q2_19

Si modifichi l'esercizio precedente in modo che venga chiesto all'utente di specificare la base e l'altezza del triangolo.

```
print "Inserisci la base del triangolo: ";
$base = <STDIN>;
chomp($base);

print "Inserisci l'altezza del triangolo: ";
$altezza = <STDIN>;
chomp($altezza);

print "Calcolo dell'area di un triangolo\n";
print "di base ".$base." e altezza ".$altezza."\n";

$area = $base*$altezza/2;
print "Area = ".$area."\n";
```

Esercizio Q2_20

Scrivere un programma in PERL che, dopo aver chiesto all'utente di inserire una stringa alfanumerica e un intero n., stampa a video la stringa secondo lo schema seguente:

```
stringastringastringa ... (n volte)

stringa stringa stringa ... (n volte)

stringa
stringa (n volte)
stringa
...


print "Inserisci una stringa: ";
$stringa = <STDIN>;
chomp($stringa);

print "Inserisci il numero di ripetizioni: ";
$ripet = <STDIN>;
chomp($ripet);

print $stringa x $ripet."\n";
print "$stringa\t" x $ripet."\n";
print "$stringa\n" x $ripet;
```

Esercizio Q2_21

Scrivere un programma in PERL che, dopo aver chiesto all'utente di inserire due stringhe alfanumeriche, determina se sono uguali (e stampa un opportuno messaggio a video) o quale delle due stringhe precede lessicograficamente l'altra se sono diverse. In quest'ultimo caso, stampa a video le due stringhe in ordine lessicografico.

Si risolva l'esercizio prima facendo uso del costrutto "if", poi cercando una soluzione alternativa che ne eviti l'uso.

```
print "Inserisci la prima stringa: ";
$stringa1 = <STDIN>;
chomp($stringa1);

print "Inserisci la seconda stringa: ";
$stringa2 = <STDIN>;
chomp($stringa2);

if($stringa1 eq $stringa2) print "Le due stringhe sono uguali \n";
    elseif($stringa1 lt $stringa2) print "$stringa1 precede $stringa2";
        else print "$stringa2 precede $stringa1";

print "\n**********************\n";

($stringa1 eq $stringa2) and print "Le due stringhe sono uguali \n";
($stringa1 lt $stringa2) and print "$stringa1 precede $stringa2";
($stringa1 gt $stringa2) and print "$stringa2 precede $stringa1";
```

Esercizio Q2_22

Scrivere un programma in PERL che converte una cifra fornita dall'utente da euro a lire (sapendo che 1 euro è pari a 1936.27 lire)

```
$euro_lire = 1936.27; #lire
print "Inserisci la cifra in Euro: ";
$euro = <STDIN>;
chomp($euro);

print "$euro euro corrispondono a ".$euro*$euro_lire." lire";
```

Esercizio Q2_23

Dopo aver chiesto all'utente di inserire un carattere da tastiera, si stampi un messaggio che indica se il carattere è un numero fra 0 e 9, una lettera minuscola, una lettera maiuscola, o un tipo diverso di carattere.

```
print "Inserisci un carattere: ";
$char = <STDIN>;
chomp($char);

if($char ge "0" and $char le "9"){print "Il carattere e' un numero intero in [0,9] \n";}
    elsif( $char ge "a" and $char le "z" ){print "Il carattere e' una lettera minuscola\n";}
        elsif( $char ge "A" and $char le "Z" ){print "Il carattere e' una lettera maiuscola\n";}
            else{print "carattere sconosciuto.\n";}
```

8 PERL: array e costrutti di controllo per l'iterazione

Esercizio Q2_24

Si realizzino in Perl le seguenti operazioni su array:

1. Si inizializzi un array, di nome @array1 e di lunghezza 5, con i seguenti valori: aa, zeta, mio, cicala, fff.
2. Si stampino a schermo i valori dell'array su una riga e separati da uno spazio
3. Si stampino i valori dell'array in seconda, terza e quarta posizione.
4. Si eseguano nell'ordine le seguenti operazioni sull'array (usare gli operatori pop, push, etc.): eliminazione dalla coda dell'array degli ultimi tre elementi; inserimento in coda dell'elemento "tuo".
5. Si stampi la nuova lunghezza dell'array e l'array stesso in ordine sia crescente che decrescente.

```
my @array = ("aa","zeta","mio","cicala","fff");

#print $array[0],"\t",$array[1],"\t",$array[2],"\t",$array[3],"\t",$array[4],"\n";
print "Array originario:  @array \n";

print "Termini in posizione 2,3,4:  @array[1..3] \n";

print "Manipolo l'array ... \n";
pop @array;
pop @array;
pop @array;
push @array, "tuo";

my $len = scalar @array;
print ("Ora la lunghezza dell'array e': $len \n");

my @array_crescente = sort @array;
print "Array in ordine crescente: @array_crescente \n";

my @array_decrescente = sort{$b cmp $a} @array;
print "Array in ordine decrescente: @array_decrescente \n";
```

Esercizio Q2_25

Si modifichi lo script precedente in modo che l'utente costruisca l'array elemento per elemento.
[Sugg: si utilizzi un construtto while con istruzione di interruzione ("last") in caso di valore inserito pari ad es. a 0]

```
my @array = ();
print "Inserisci un nuovo elemento (0 per finire): ";
while(<STDIN>){
    chomp;
    last unless $_;
    print "Elemento inserito: $_ \n";
    @array=(@array,$_);
    print "Inserisci un nuovo elemento (0 per finire): ";
}

print "Array originario:  @array \n";

print "Termini in posizione 2,3,4:  @array[1..3] \n";

...
```

Esercizio Q2_26

Si scriva uno script che realizza le seguenti operazioni:

1. Ricava il numero di lettere maiuscole (dimensione dell'intervallo [A,Z]) e lo comunica all'utente. Stampa inoltre su una riga tutte le lettere maiuscole.
2. Chiede all'utente di inserire un numero intero k compreso tra 2 e il numero di lettere maiuscole ricavato (itera la richiesta fino a quando l'utente non inserisce un intero nel range richiesto)
3. Stampa su una riga le prime k lettere maiuscole
4. Crea un nuovo array che viene usato per contenere le prime k lettere maiuscole raggruppate in stringhe di lunghezza crescente. Ad es. per k=6, st2[0]=A, st2[1]=BC, st2[2]=DEF.
5. Stampa il nuovo array.

```
#conta le lettere maiuscole
my @maiuscole=("A".."Z");
print "@maiuscole\n";
my $lenmaiuscole= scalar @maiuscole;
print "Le lettere maiuscole sono $lenmaiuscole.\n";

my $ind=0;
while($ind<2 or $ind>$lenmaiuscole){
print "Inserisci un numero intero tra 2 e $lenmaiuscole: ";
$ind=<STDIN>;
chomp($ind);
}

print "Le prime $ind lettere maiuscole sono: @maiuscole[0..($ind-1)] \n";

my $ind1=0;
my $ind2=0;
my $incr=1;
my @stringArray=();
while($ind2<$ind){
    @stringArray=(@stringArray,(@maiuscole[$ind1..$ind2],"\n"));
    $ind1=$ind2+1;
    $ind2+=++$incr;
}
@stringArray=(@stringArray,(@maiuscole[($ind1)..($ind-1)],"\n"));

print "\nNuovo array con elementi raggruppati: \n";
print " @stringArray \n"
```

Esercizio Q2_27

Si realizzi un programma che permette di gestire un array in modo interattivo, fornendo all'utente il seguente menu di scelte:

```
che operazione si vuole fare? i (inserisci), c (cancella), s (sort) 0 (termina):
```

A seguito di una data scelta, il programma esegua le operazioni necessarie e stampi il nuovo array. Nel caso "i" si gestisca interattivamente l'inserimento di un nuovo elemento.

```
my @array = ();
my $el;

print "Cosa vuoi fare? i (inserisci), c (cancella), s (sort) 0 (termina) :";
$scelta=<STDIN>;
chomp $scelta;
while(!($scelta eq "0")){

    for($scelta){ #switch
        if ($_ eq "i"){ print "hai scelto i. Che elemento vuoi accodare? ";
                  $el=<STDIN>;
                  chomp $el;
                  push @array, $el;
```

```
                                print "Il nuovo array: @array \n";
                }
        elsif($_ eq "c"){ pop @array;
                   print "hai cancellato l'ultimo elemento\n";
                   print "Il nuovo array: @array \n";
                }
        elsif($_ eq "s"){ @array = sort @array;
                   print "hai riordinato l'array\n";
                   print "Il nuovo array: @array \n";
                }
        else {print "scelta non valida\n"};
    }
    print "Cosa vuoi fare? i (inserisci), c (cancella), s (sort), 0 (termina) :";
    $scelta=<STDIN>;
    chomp $scelta;
}
```

Esercizio Q2_28

Si cheda all'utente di immettere una stringa di lunghezza arbitraria costituita dalle sole quattro lettere A,C,G e T (lettere diverse sono comunque ammesse). Si mutino poi tutte le lettere T in U e tutte le eventuali lettere diverse da A,C,G e T, in N. Si stampi infine la stringa risultante.
[Sugg.: si usi la funzione substr ($stringa,$i,$j) per estrarre da $stringa la sottostringa di lunghezza $j che inizia dalla posizione $i]

```
print "Inserisci una stringa composta dalle lettere A, C, G e T: ";
$stringa = <STDIN>;
chomp($stringa);

my $newstring="";

my $i=0;
while($i < length $stringa){
    $char=substr ($stringa,$i,1);
      if ($char eq "T"){
        $newstring=$newstring."U";
      }
      elsif(!($char eq "A") and !($char eq "C") and !($char eq "G")){
        $newstring=$newstring."N";
    }
    else{
        $newstring=$newstring.$char;
    }
    $i++
}

print "Stringa trascritta: $newstring";
```

Esercizio Q2_29

Si costruisca un vettore @basi contenente i caratteri A, T, G, e C. Si costruisca poi un vettore di lunghezza 10 formato da una combinazione scelta a caso degli elementi di @basi Si iteri il passo precedente un numero prefissato di volte concatenando tutti i vettori generati in una variabile stringa Si stampino i singoli vettori uno per riga e infine la variabile stringa contenente la concatenazione dei vettori.
[Sugg: per generare un valore intero scelto a caso nell'intervallo [0,$len] si usi l'istruzione "int rand($len)"].

```
my @basi = ("A","T","G","C");
my @array1 = ();

my $dim1=4;
my $dim2=10;

$string="";
```

```
my $i=0;
foreach $i(0..$dim1-1){
    foreach $j(0..$dim2-1){
        $array1[$j]=$basi[int rand(4)];
    }
    $string=$string." @array1";
    print "array1 $i: @array1 \n";
}

print "string: $string";
```

9 Java: esercizi di ricapitolazione

Esercizio Q2_30

Si progetti una classe di nome Treno, la quale istanzia oggetti in grado di rappresentare la potenza di traino della locomotiva, il numero di vagoni e il peso di un treno. Si assuma che l'oggetto abbia senso solo se il peso del treno non supera la potenza e se il numero di vagoni è inferiore a 10: quest'ultimo campo dovrà essere pubblicamente accessibile a tutte le classi che intendano adoperare le risorse fornite dalla classe Treno. La classe contenga:

1. un costruttore Treno(int p), il quale costruisce un oggetto che modella un treno formato dalla sola locomotiva di potenza p
2. un costruttore Treno(int p, int n, int m), il quale se possibile costruisce un oggetto che modella un treno formato dalla locomotiva di potenza p più n vagoni ciascuno pesante m, altrimenti restituisce la sola locomotiva di potenza p
3. un metodo boolean accodaVagone(int m), che se possibile aggiunge al treno modellato un vagone di massa m, e in tal caso restituisce vero; altrimenti restituisce falso
4. un metodo int sganciaVagone(), che se possibile elimina il vagone di coda del treno modellato e in tal caso restituisce la sua massa; altrimenti restituisce 0
5. un metodo int quantoPesa(), che restituisce il peso del treno modellato
6. un metodo int quantoResta(), che restituisce il peso che può essere ancora aggiunto al treno
7. un metodo int quantiVagoni(), che restituisce il numero dei vagoni che attualmente formano il treno
8. un metodo String toString(), che restituisce il modello del treno nel formato "<p>:m1:m2:...", in cui p è la potenza della locomotiva e ciascun elemento tra caratteri due punti contiene la massa m del vagone i-esimo.

Per modellare i vagoni si consiglia di adoperare un array di interi mass[i], in cui ciascun elemento contiene la massa m del vagone i-esimo. La classe dovrà far parte di un package di nome myclasses accessibile da ambiente Java.

Si scriva infine una classe di nome TestTreno che contenga delle semplici istruzioni per la verifica del corretto funzionamento dei metodi della classe Treno.

```
package myclasses;

public class Treno {
    public static final int MAX_VAG = 10;
    private int power, mTot, lunghTreno;
    private int[] mass = new int[MAX_VAG];

    public Treno(int p, int n, int m) {
        if (n>10 || n*m>p) {
            power = p;
            for (int i=0; i<MAX_VAG; i++)
                mass[i]=0;
        } else {
            power = p;
            mTot = n*m;
            lunghTreno = n;
            for (int i=0; i<n; i++)
                mass[i]=m;
            for (int i=n; i<MAX_VAG; i++)
                mass[i]=0;
        }
    }

    public Treno(int p) {
        this(p,0,0);
    }

    public boolean accodaVagone(int m) {
        if (lunghTreno==MAX_VAG || m+mTot>power)
            return false;
        else {
            mass[lunghTreno++] = m;
            mTot += m;
            return true;
        }
    }
```

```
    public int sganciaVagone() {
        int massa;
            if (lunghTreno==0)
                massa = 0;
            else {
                massa = mass[--lunghTreno];
                mTot -= massa;
                mass[lunghTreno] = 0;
            }
        return massa;
    }

    public int quantoPesa() {
        return mTot;
    }

    public int quantoResta() {
        return power-mTot;
    }

    public int quantiVagoni() {
        return lunghTreno;
    }

    public String toString() {
        String s = "<" + power + ">";
        for(int i=0; i<lunghTreno; i++)
            s += ":" + mass[i];
        return s + ":";
    }
}
```

Esercizio Q2_31

Si scriva una classe di nome UsaTreno contenente un metodo main che, appoggiandosi alle risorse della classe Treno contenuta nel package myclasses, simula la creazione di due treni dotati di locomotive di potenza di traino uguale a 50, a cui sono accodati al più 10 vagoni di peso casuale variabile tra 1 e 10. Il metodo successivamente scambia i vagoni in modo da bilanciare quanto più possibile il peso dei due treni secondo una strategia "greedy", ovvero sganciando tutti i vagoni dai due treni e poi assegnando via via il vagone più pesante al treno in quel momento più leggero.

```
import prog.io.ConsoleOutputManager;
import myclasses.Treno;

class UsaTreno {

    public static void main(String[] args) {
        final int MAX_VAG = Treno.MAX_VAG;
        ConsoleOutputManager out = new ConsoleOutputManager();
        int potenza = 50;
        Treno t1 = new Treno(potenza);
        Treno t2 = new Treno(potenza);

        do
            t1.accodaVagone(1+(int)(10*Math.random()));
        while(t1.quantiVagoni()<MAX_VAG && t1.quantoResta()>0 );

        do
            t2.accodaVagone(1+(int)(10*Math.random()));
        while(t2.quantiVagoni()<MAX_VAG && t2.quantoResta()>0 );

        out.println("Treni creati casualmente:" );
        out.println("T1: " + t1.toString());
        out.println("T2: " + t2.toString());
```

```
        out.println("Peso treni creati casualmente:" );
        out.println("Peso T1: " + t1.quantoPesa());
        out.println("Peso T2: " + t2.quantoPesa());

        // mettiamo le masse di tutti i vagoni dentro a un vettore

        int[] vagoni = new int[t1.quantiVagoni()+t2.quantiVagoni()];
        int temp = 0;

        while (t1.quantiVagoni()>0)
            vagoni[temp++] = t1.sganciaVagone();

        while (t2.quantiVagoni()>0)
            vagoni[temp++] = t2.sganciaVagone();

        // eliminiamo via via l'elemento più pesante dal vettore e assegnamolo al treno più leggero

        int massimo, max_ind = 0;

        for (int i=0; i<vagoni.length; i++) {
           massimo = 0;

            for (int j=0; j<vagoni.length; j++) {
                if (vagoni[j]>massimo) {
                    massimo = vagoni[j];
                    max_ind = j;
                }
            }

            if (t1.quantiVagoni()<MAX_VAG && t2.quantiVagoni()<MAX_VAG) // entrambi i treni non saturi
                if (t1.quantoPesa()<t2.quantoPesa())
                    t1.accodaVagone(vagoni[max_ind]);
            else
                t2.accodaVagone(vagoni[max_ind]);
            else if (t1.quantiVagoni()<MAX_VAG)     // solo t1 non saturo
                t1.accodaVagone(vagoni[max_ind]);
            else                                        // solo t2 non saturo
            t2.accodaVagone(vagoni[max_ind]);
            vagoni[max_ind] = 0;
        }

    out.println("Treni riordinati:" );
    out.println("T1: " + t1.toString());
    out.println("T2: " + t2.toString());

    out.println("Peso treni riordinati:" );
    out.println("Peso T1: " + t1.quantoPesa());
    out.println("Peso T2: " + t2.quantoPesa());
    }
}
```

Esercizio Q2_32

Si progetti una classe di nome ZeroUno, la quale istanzia oggetti che modellano successioni formate da zero e uno. La classe mantiene la successione in un campo stringa privato (già dato nel testo) e fornisce:

1. un costruttore che, invocato senza parametri, costruisce un oggetto che rappresenta una successione vuota
2. un costruttore che, ricevuto un parametro s di tipo String, costruisce un oggetto che rappresenta la più grande successione di zero e uno contenuta in s nell'ordinamento dato dalla stringa. Ad es., se s è "0er10rt0rt1ghf10h", il campo zerouno dell'oggetto varrà "0100110"
3. un costruttore che, ricevuto un parametro n di tipo int, costruisce un oggetto che rappresenta una successione lunga n di zero e uno casualmente distribuiti
4. un metodo toString() che restituisce la stringa che rappresenta la successione di zero e uno in cui ogni termine è seguito da virgola: se, ad esempio, la successione è 0100 allora la stringa che la rappresenta sarà "0,1,0,0,"

5. un metodo toArray() che restituisce un array di boolean in cui l'elemento di indice i è true o false a seconda che il termine i-esimo della successione (numerata partendo da zero) sia rispettivamente uguale a uno o zero
6. un metodo somma() che RICORSIVAMENTE somma i termini della successione e restituisce un intero contenente la somma. Se ad esempio la successione è 001100010 allora somma() restituirà il valore 3

La classe inoltre contenga un metodo main() il quale, costruiti due oggetti ZeroUno rispettivamente adoperando una stringa e un intero come parametri per la creazione delle successioni a essi associate, dà successivamente la rappresentazione in stringa e la somma di entrambe le successioni.

```
import prog.io.ConsoleInputManager;
import prog.io.ConsoleOutputManager;

public class ZeroUno {

    private String zerouno;

    public ZeroUno () {
        zerouno = "";
    }

    public ZeroUno (String s) {
        this();
        for (int i=0; i<s.length(); i++)
            if (s.charAt(i)=='0' || s.charAt(i)=='1')
        zerouno += s.charAt(i);
    }

    public ZeroUno (int n) {
    this();
    for (int i=0; i<n; i++)
        zerouno += (Math.random()<0.5 ? 0 : 1);
    }

    public String toString() {
    String s = "";
    for (int i=0; i<zerouno.length(); i++)
        s += zerouno.charAt(i) + ",";
    return s;
    }

    public boolean[] toArray() {
    boolean[] b = new boolean[zerouno.length()];
    for (int i=0; i<zerouno.length(); i++)
        b[i] = (zerouno.charAt(i)=='0' ? false : true);
    return b;
    }

    public int somma() {
    if (zerouno.equals(""))
        return 0;
    else if (zerouno.equals("0"))
        return 0;
    else if (zerouno.equals("1"))
        return 1;
    else
        return (toArray()[0] ? 1:0) + (new ZeroUno(zerouno.substring(1))).somma();
    }

    ////////////////////////////// main()

    public static void main(String[] args) {
    ConsoleInputManager tastiera = new ConsoleInputManager();
    ConsoleOutputManager schermo = new ConsoleOutputManager();

    String s = tastiera.readLine("Stringa: ");
```

```
    ZeroUno z1 = new ZeroUno(s);
    ZeroUno z2 = new ZeroUno(10);

    schermo.println("Oggetto 1: " + z1.toString());
    schermo.println("Somma: " + z1.somma());

    schermo.println("Oggetto 2: " + z2.toString());
    schermo.println("Somma: " + z2.somma());
    }
}
```

Esercizio Q2_33

Si progetti una classe di nome GrandiIntPositivi, la quale istanzia oggetti che modellano numeri interi positivi di 20 cifre. La classe mantiene cifre comprese tra 0 e 9 in un campo array privato int[] cifre di dimensione costante MAX posta a 20. La classe fornisce:

1. un costruttore che, invocato senza parametri, costruisce un oggetto che rappresenta lo zero, ovvero inizializza l'array ponendo tutti gli elementi a zero;
2. un costruttore che, ricevuto un parametro di tipo String, costruisce un oggetto che rappresenta il numero contenuto nella stringa. Per semplicita' si assuma che la stringa contenga solo cifre decimali. Inoltre, lo stesso costruttore sia eventualmente in grado di scartare la coda di stringhe lunghe piu' di 20 cifre;
3. un metodo GrandiIntPositivi piu(GrandiIntPositivi g) che restituisce, in forma di un nuovo oggetto, la somma dell'oggetto che esegue il metodo con quello passato come parametro. Se la stessa somma supera le 20 cifre allora restituisce l'oggetto che rappresenta il piu' grande intero positivo di 20 cifre;
4. un metodo GrandiIntPositivi perInt(int a) che restituisce il prodotto dell'oggetto che esegue il metodo per l'intero positivo passato come parametro. Il metodo deve essere implementato in forma RICORSIVA, facendo uso del metodo "piu".
5. un metodo toString() che stampa in forma di stringa lunga 20 caratteri il numero rappresentato dall'oggetto che utilizza il metodo;
6. un metodo boolean equals(GrandiIntPositivi g) che verifica l'uguaglianza tra due oggetti in base al numero rappresentato.

La classe contenga infine un metodo main con semplici istruzioni di test della classe (somma fra due oggetti, prodotto per un intero positivo e stampa).

```
import prog.io.ConsoleInputManager;
import prog.io.ConsoleOutputManager;

public class GrandiIntPositivi {

    ////////////////////////////
    private static int MAX = 10;
    private int[] cifre;
    ////////////////////////////

    public GrandiIntPositivi() {
        cifre = new int[MAX];           // cifre tutte uguali a zero
    }
    ////////////////////////////

    public GrandiIntPositivi(String s) {
        this();
        int l = (s.length()>MAX ? MAX : s.length()) ;     // al massimo MAX cifre
        int n = MAX;
        while (l>0)
            cifre[--n] = s.charAt(--l) - '0';  // cifra meno significativa in coda all'array
    }
    ////////////////////////////

    public GrandiIntPositivi piu(GrandiIntPositivi g) {
        GrandiIntPositivi h = new GrandiIntPositivi();
        int somma_cifre;
```

```
        int riporto = 0;

        for (int i=MAX-1; i>=0; i--) {                  // iniziamo dalla cifra meno significativa
            somma_cifre = cifre[i]+g.cifre[i]+riporto;

            if (somma_cifre < 10) {
                h.cifre[i] = somma_cifre;
                riporto = 0;
            } else {
                h.cifre[i] = somma_cifre-10;
                riporto = 1;
            }
        }

        if (riporto == 1) {
            for (int i=0; i<MAX; i++)
            h.cifre[i] = 9;
        }

        return h;
    }
    ////////////////////////////

    public GrandiIntPositivi perInt(int a) {
        if (a>0)
            return this.piu(this.perInt(a-1));
        else{
            GrandiIntPositivi h = new GrandiIntPositivi();
            return h;
        }
    }
    ////////////////////////////

    public String toString() {
        String s = "";
        for (int i : cifre)
            s += i;
        return s;
    }
    ////////////////////////////

    public boolean equals(GrandiIntPositivi g) {
        return this.toString().equals(g.toString());
    }
    ////////////////////////////

    public static void main(String[] args) {
        ConsoleInputManager tastiera = new ConsoleInputManager();
        ConsoleOutputManager schermo = new ConsoleOutputManager();

        String s;
        GrandiIntPositivi g1,g2,g3,g4;

        s = tastiera.readLine("Stringa 1: ");
        g1 = new GrandiIntPositivi(s);
        s = tastiera.readLine("Stringa 2: ");
        g2 = new GrandiIntPositivi(s);
        g3 = g1.piu(g2);
        schermo.println(g1 + " +\n" + g2 + " =\n" + g3);
        int a=3;
        g4 = g1.perInt(a);
        schermo.println(g1 + " * " + a + " = " + g4);
        schermo.println("Uguali? " + (g1.equals(g2) ? "SI":"NO"));
    }
}
```