Esempi di uso di ZucLib

Lettura/scrittura semplificata di dati

Problema Hello

La classica applicazione che riproduce a video "Salve Gente" può assumere il seguente aspetto:

```
import static zuclib.Varie.*; // significa: voglio usare i comandi in zuclib.Varie

public class SalveGente {
    public static void main(String[] args) {
        writeLn("Salve Gente");
    }
}
```

Il comando writeLn sostituisce in questo caso il più lungo e complesso System.out.println che richiederebbe diverse spiegazioni.

Problema della Somma di due numeri

Il classico programma che letti due numeri da tastiera ne scriva la somma a video può essere risolto in Java in modo molto semplice con zuclib:

```
import static zuclib.Varie.*;

public class SommaDueNumeri {
    public static void main(String[] args) {
        int a = askInt("dammi il primo intero: ");
        int b = askInt("dammi il secondo intero: ");
        writeLn("la somma di ", a, " e ", b, " e` ", a + b);
    }
}
```

Proviamo ad eseguirlo:

```
dammi il primo intero: 100
dammi il secondo intero: -5.3
Intero non valido. RIPROVA
dammi il secondo intero: sbc
Intero non valido. RIPROVA
dammi il secondo intero: -5
la somma di 100 e -5 e` 95
```

Si osservi che la funzione askInt chiede la ripetizione dell'immissione del dato nel caso in cui il dato non sia valido. Si noti anche che la funzione writeLn prende un numero variabile di parametri.
Per leggere numeri reali si farà della funzione askDouble che restituisce il double letto da tastiera. Per le stringhe è invece disponibile la funzione askString.

Problema della Tabella Pitagorica

La scrittura a video della tabella pitagorica richiede solitamente un certo rispetto dell'incolonnamento dei numeri. Questo si può fare con:

```
import static zuclib.Varie.*;

public class TabellaPitagorica {

    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            for (int j = 1; j <= 10; j++) {
                write(intFormat(i*j,4));
            }
            writeLn();
        }
    }
}
```

L'output corrispondente è:

```
   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

Il corretto incolonnamento dei numeri si deve all'uso della funzione intFormat(int,int) di zuclib.Varie con l'ovvio significato.

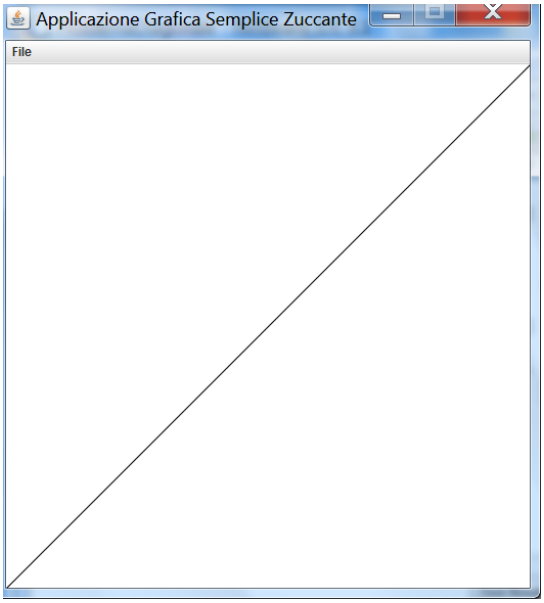
Grafica molto elementare

Problema della Diagonale

Disegnare una linea dal vertice in basso a sinistra al vertice in alto a destra di una finestra grafica. Con zuclib il programma può essere ridotto a:

```
import static zuclib.GraficaSemplice.*;

public class ProblemaDiagonale {

    public static void main(String[] args) {
        linea(0,0,1,1);
    }
}
```

L'effetto è il seguente:

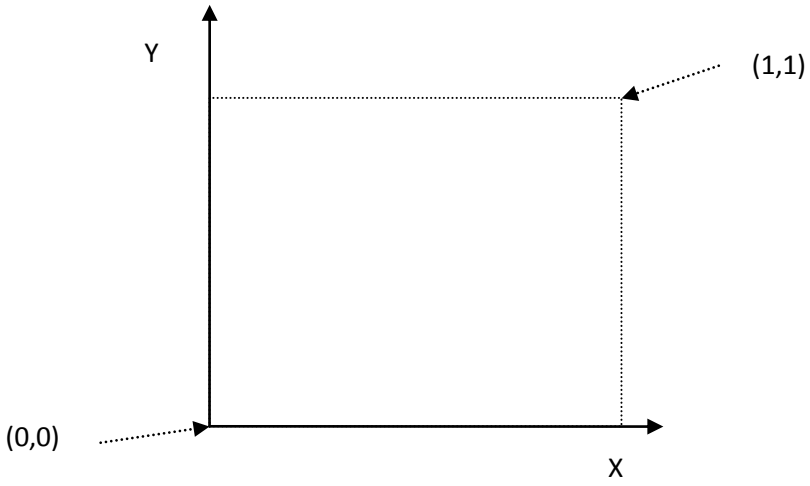


Utilizzando le librerie standard di Java il codice sarebbe molto più lungo e complesso.
L'area a disposizione è implicitamente di dimensioni 512 x 512 pixel.
Sia il titolo della finestra che l'area grafica a disposizione possono essere impostati specificati con il comando esplicito setFinestra.

Esempio:

```
setFinestra(320, 200, "Mio Titolo");
```

Il comando `linea` prende 4 parametri x0,y0,x1,y1 e traccia una linea con il colore corrente (in questo caso il nero) dal punto (x0,y0) al punto (x1,y1).
Il sistema grafico di riferimento prevede l'utilizzo di coordinate reali, che chiameremo **coordinate utente**, come rappresentato in figura in cui l'area tratteggiata rappresenta la parte disegnabile della finestra (canvas).



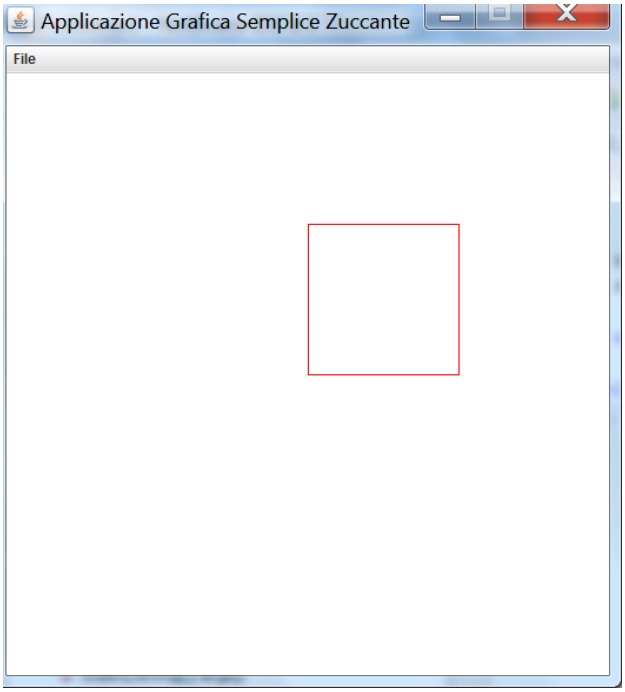
Il centro della finestra ha quindi coordinate (0.5, 0.5). Il vantaggio di questo sistema di riferimento è che il codice per la grafica è indipendente dal numero di pixel scelti per la larghezza e l'altezza della finestra.
Il menu **File** ha un'unica operazione *Salva Immagine* che consente di salvare l'immagine costruita (poco interessante in questo caso).

Problema del Quadrato

Porponiamoci di disegnare un quadrato di colore rosso con la grafica della tartaruga.
Le tartarughe sono la base degli sprite di Scratch. La soluzione in Java con la libreria zuclib potrebbe assumere questo aspetto in cui TARTA rappresenta approssimativamente il "gatto" di Scratch:

```
public class Quadrato {

    public static void main(String[] args) {
        TARTA.setColore(ROSSO);
        for(int i=0;i<4;i++){
            TARTA.avanti(0.25);
            TARTA.destra(90);
        }
    }
}
```

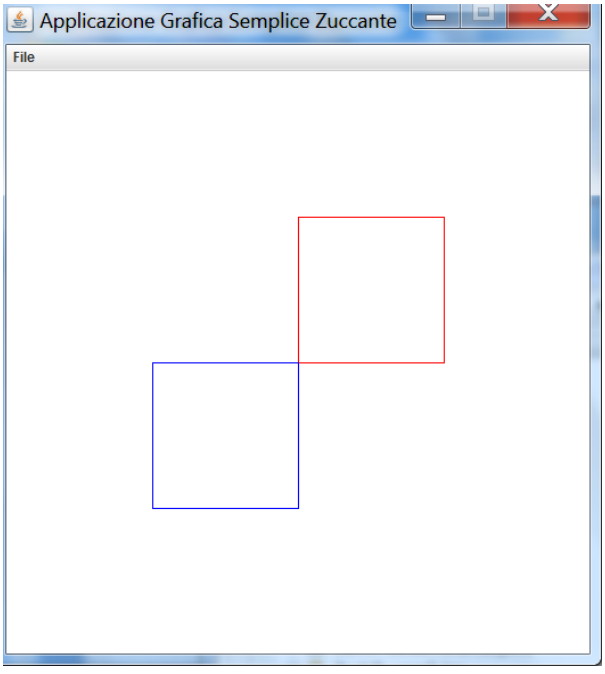
Eseguendo il programma otterremo la creazione della seguente finestra:



Il comando:

```
TARTA.avanti(0.25);
```

ha l'effetto di fa avanzare TARTA di una quantità pari al 25% dei pixel della larghezza (o altezza) utile della finestra.
E' possibile creare e utilizzare più tartarughe nello stesso programma. Ad esempio, per disegnare "contemporaneamente" due quadrati rispettivamente di colore rosso e di colore blu come in figura:



potremmo usare due tartarughe come segue:

```
import static zuclib.GraficaSemplice.*;
import zuclib.Tartaruga;

public class Quadrati {

    public static void main(String[] args) {
        Tartaruga tr = new Tartaruga(0.5,0.5,90,ROSSO);
        tr.ritardoMillisec(500);
        Tartaruga tb = new Tartaruga(0.5,0.5,-90,BLU);
        tb.ritardoMillisec(500);
        for(int i=0;i<4;i++){
            tr.avanti(0.25);
            tb.avanti(0.25);
            tr.destra(90);
            tb.destra(90);
        }

    }
}
```

In questo caso il ritardo introdotto nei movimenti avanti e destra con i comandi:

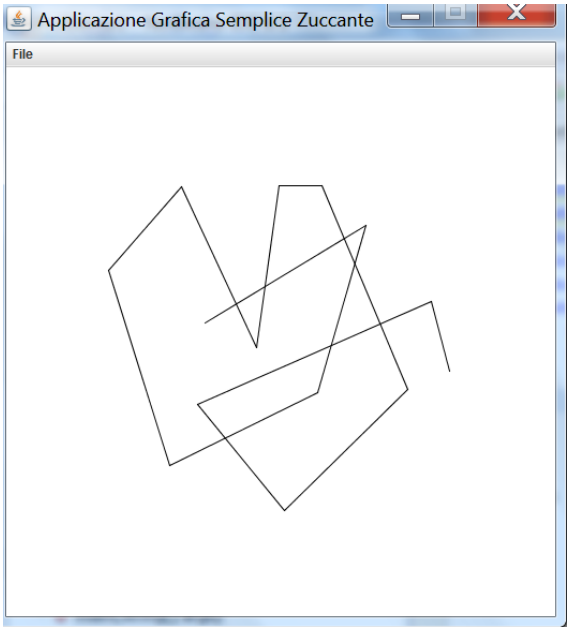
```
tr.ritardoMillisec(500)  e  tb.ritardoMillisec(500);
```

ha lo scopo di far vedere la successione di movimenti.

Problema del disegno di linee con il mouse

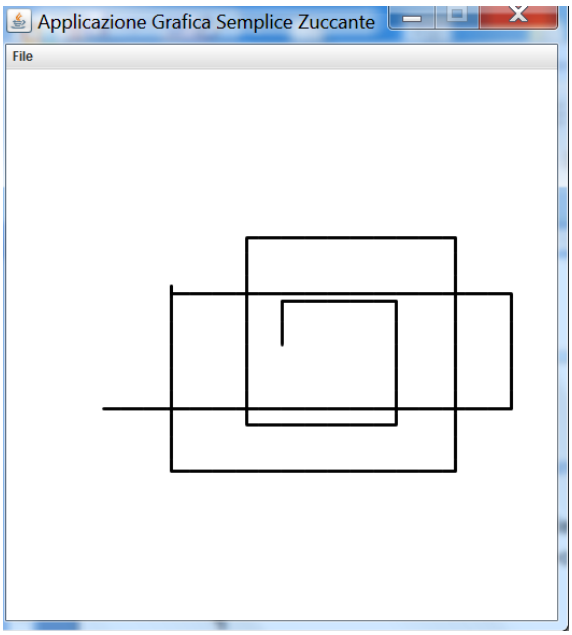
Vogliamo unire con delle linee i punti cliccati con il mouse: ogni punto è individuato da dove si trova il mouse quando viene rilasciato il pulsante.

```
import static zuclib.GraficaSemplice.*;

public class DisegnaLinee {

    public static void main(String[] args) {

        while (!mousePremuto()); //attendi prima pressione
        while (mousePremuto()); //attendi primo rilascio
        double x0 = mouseX();   //memorizza coordinate punto iniziale
        double y0 = mouseY();
        double x1, y1;
        while (true) {  // all'infinito
            while (!mousePremuto()); //attendi pressione
            while (mousePremuto()); //attendi rilascio
            x1 = mouseX();
            y1 = mouseY();
            linea(x0, y0, x1, y1);
            x0 = x1; // il nuovo punto di partenza è l'ultimo punto di arrivo
            y0 = y1;
        }
    }
}
```



Problema del disegno con i tasti freccia:

Proponiamoci ora di disegnare delle linee servendoci dei tasti freccia che si trovano in ogni tastiera.
I disegni che vogliamo fare siano come quello indicato in figura:



Il codice che illustra il modo di leggere e riconoscere i tasti freccia potrebbe essere:

```
import static zuclib.GraficaSemplice.*;

public class LetturaTasti {

    public static void main(String[] args) {
        TARTA.setGrossezza(0.004);
        double passo = 0.001;
        while (true) {
            if (premutoTasto(FRECCIA_SU)) {
                TARTA.setAngolo(90);
                TARTA.avanti(passo);
            } else if (premutoTasto(FRECCIA_GIU)) {
                TARTA.setAngolo(-90);
                TARTA.avanti(passo);
            } else if (premutoTasto(FRECCIA_SX)) {
                TARTA.setAngolo(180);
                TARTA.avanti(passo);
            } else if (premutoTasto(FRECCIA_DX)) {
                TARTA.setAngolo(0);
                TARTA.avanti(passo);
            }
        }
    }
}
```

Problema dei cerchi colorati

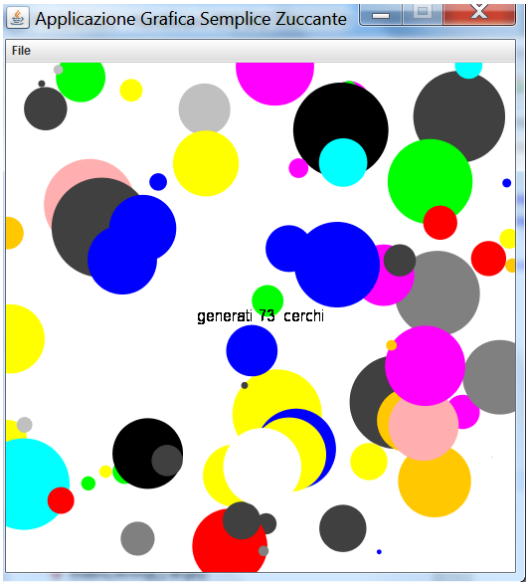
Questo problema ha lo scopo di illustrare la semplicità di generazione di numeri o colori casuali.

Disegnare N cerchi di raggio casuale (al massimo 0,1) disposti a caso nella finestra , con N scelto casualmente tra 2 e 100, riempiti con un colore casuale.

Il codice potrebbe essere:

```
public class CerchiColorati {

    public static void main(String[] args) {
        int n = random(2,100);
        for (int i = 0; i < n; i++) {
            cerchioPieno(Math.random(), Math.random(),Math.random()*0.1, coloreACaso());
            testo(0.5,0.5,"generati "+n+"  cerchi");
        }
    }
}
```

Una sua esecuzione potrebbe ad esempio generare la seguente finestra:



Si osservi che si sfruttano tre operazioni che hanno a che fare con il caso:

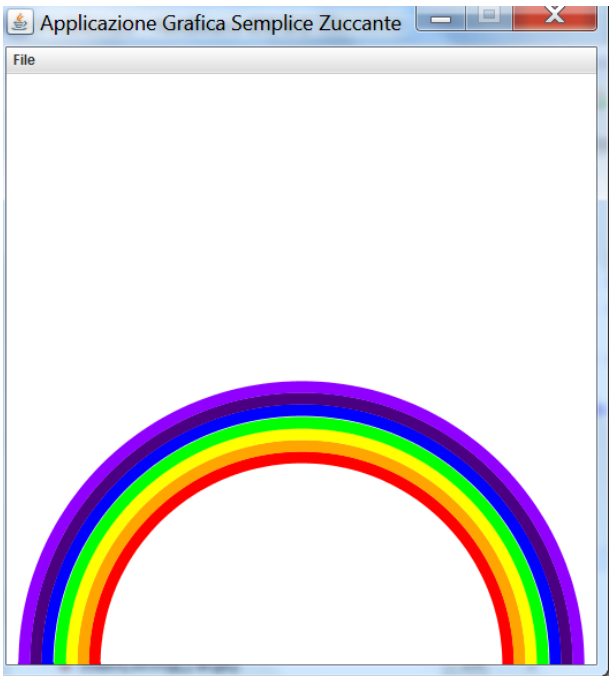
Math.random() : genera un double compres0 tra 0 e 1 (escluso)
--questa è una funzione della libreria Math, predefinita in Java

random(int a, int b): genera un intero casuale compre tra a e b (estremi inclusi)
--questa è una funzione della libreria zuclib contenuta nella classe Varie

coloreACaso() : genera un colore) tra un insieme di una ventina di colori tra i più usati.
--anche questa è una funzione della libreria zuclib contenuta nella classe Varie.

Problema dell'Arcobaleno

La grafica di zuclib è abbastanza ricca: dispone di operazioni per disegnare cerchi punti, linee, archi, rettangoli, ellissi, poligoni qualsiasi, anche riempiti con un dato colore.
Disegnare un arcobaleno come il seguente, facendo uso di archi:



Il codice può essere:

```
import java.awt.Color;
import static zuclib.GraficaSemplice.*;
import static zuclib.Varie.*;

public class Arcobaleno {

    public static void main(String[] args) {
        Color[] coloreArcobaleno= {ROSSO,ARANCIONE,GIALLO,VERDE,BLU,INDACO,VIOLA};
        setGrossezza(getGrossezza()*10);
        for (int i=0;i< coloreArcobaleno.length;i++){
            setColore(coloreArcobaleno[i]);
            arco(0.5,0,0.4+0.02*i,-180,180);
        }
    }
}
```

Problema dell'Arcobaleno

Progettare un programma di Paint in cui si possano scegliere i colori e anche la grossezza del pennello.

```
import java.awt.Color;
import static zuclib.GraficaSemplice.*;
public class Paint {
    public static final double maxLarghezza = 1.0;
    public static final double maxAltezza = 1;
    public static final double maxAltezzaTavolozza = maxAltezza / 20.0;
    public static double dimensionePennello = 0.005;
    public static final Color colori[] = {BIANCO,NERO, GRIGIO_CHIARO, GRIGIO,
        GRIGIO_SCURO, CELESTE, BLU, CIANO, VERDE_CHIARO, VERDE, VERDE_SCURO,
        GIALLO, ROSA, ROSSO, MAGENTA, ARANCIONE, VIOLA, INDACO, MARRONE_CHIARO,
        MARRONE, MARRONE_SCURO};
    static Color coloreAttuale = NERO;
    static int numColori = colori.length;
    static double larg = maxLarghezza / (numColori+1);

    public static void disegnaTavolozza() {
        for (int i = 0; i < numColori; i++) {
            double posX = i * larg, posY = maxAltezza - maxAltezzaTavolozza/2;
            rettangoloPieno(posX, posY, larg, maxAltezzaTavolozza, colori[i]);
        }
        mostraColoreGrossezza();
    }
    public static void mostraColoreGrossezza() {
        double posX= numColori*larg , posY = maxAltezza - maxAltezzaTavolozza/2;
        rettangoloPieno(posX,posY,larg,maxAltezzaTavolozza,BIANCO);
        cerchioPieno(posX,posY,dimensionePennello/2,coloreAttuale);

    }
    public static void disegnaPaint() {
        setFinestra(800, 800, "Paint V.04 - provare i tasti 1, 2, 3, R  ");
        disegnaTavolozza();
        coloreAttuale = NERO;
        double limite = maxAltezza - maxAltezzaTavolozza, posX, posY;
        while (true) {
            if (mouseY() >= limite) {
                //bisogna comunque cliccare  altrimenti non prende mouseY
                coloreAttuale = new Color(getMouseRGBcode());
                mostraColoreGrossezza();
            } else if (premutoTasto('1')) {
                dimensionePennello = 0.005;
                mostraColoreGrossezza();
            } else if (premutoTasto('2')) {
                dimensionePennello = 0.01;
                mostraColoreGrossezza();
            } else if (premutoTasto('3')) {
                dimensionePennello = 0.02;
                mostraColoreGrossezza();
            } else if (premutoTasto('R')) { // premutoTasto('r')
                pulisci();
                disegnaTavolozza();
                coloreAttuale = NERO;
                dimensionePennello = 0.005;
            } else {
                posY = mouseY();
                posX = mouseX();
                do {
                    double x = mouseX();
                    double y = mouseY();
                    if ( mousePremuto() && posY < limite - dimensionePennello )
                     linea(posX, posY, x, y, coloreAttuale, dimensionePennello);
                    posX = x;
                    posY = y;
                }
                while ( mousePremuto() && posY < limite - dimensionePennello );
            }
        }
    }
    public static void main(String[] args) {
        disegnaPaint();
    }
}
```