Simulatore di un miscelatore di acqua

In un miscelatore entrano acqua calda alla temperatura TC con acqua fredda alla temperatura TF ed esce acqua ad una temperatura intermedia TM. Se si indicano con FC ed FF i flussi di acqua calda ed acqua fredda in litri al secondo, la temperatura TM dell'acqua fornita dal miscelatore si può calcolare con la seguente formula:

$$TM = \frac{TC*FC + TF*FF}{FC + FF}$$

Ogni variazione di TC, FC, TF, FF si ripercuote immediatamente in una variazione di TM. L'acqua fornita dal miscelatore deve percorrere un tubo prima di essere disponibile in uscita. Le variazioni di TM si avvertono all'uscita del tubo solo dopo un tempo di ritardo RIT, misurato in secondi, che è uguale al tempo che l'acqua impiega a percorrere il tubo stesso. Generalmente quindi la temperatura T dell'acqua all'uscita del tubo in un momento qualsiasi sarà diversa da TM. Ad un certo istante, che chiameremo istante iniziale, si suppone che TM sia eguale a T e che tale sia anche il valore della temperatura dell'acqua in tutto il tubo. Indicheremo questo valore con TZERO. Supponiamo anche che sia noto il valore di FC in tale istante ed indichiamo tale valore con FZERO. A partire dall'istante iniziale, FF, TF, TC variano nel tempo t e sono dati i loro valori ad intervalli regolari di un secondo.

Nel tentativo di far rimanere la temperatura T dell'acqua che esce dal tubo al valore TZERO, è stato applicato un dispositivo che, ad intervalli regolari di un secondo, misura T e produce istantaneamente una variazione DFC del flusso di acqua calda FC proporzionale alla differenza fra T e TZERO secondo la formula: DFC = -K(T - TZERO).

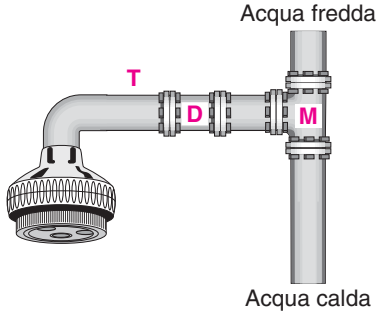
Il programma deve simulare il comportamento del sistema descritto, cioé calcolare il valore secondo per secondo delle variabili FC, T, TM per una durata di N secondi. I valori di RIT, K, TZERO, FZERO, N debbono essere introdotti dall'utente del programma all'inizio di ogni simulazione.

Il programma deve stampare i valori delle variabili FF, TF, FC, TC, T, TM secondo per secondo in modo tale che siano facilmente confrontabili tra di loro. Si suppone che le temperature varino nell'intervallo fra 0 e 99.9 gradi centigradi e i flussi nell'intervallo fra 0 e 50 litri al secondo e che per tutte le variabili reali sia significativa solo la prima cifra decimale. I valori di FF, TF, TC sono dati secondo per secondo e per un massimo di 50 secondi.

(Problema tratto dalla seconda prova scritta di informatica, maturità tecnica industriale, 1979)

TRACCIA PER LA SOLUZIONE

È opportuno schematizzare il problema fornendone una rappresentazione sistemica con l'individuazione delle parti, delle correlazioni e delle finalità.



- *Parti*: un miscelatore (M), un dispositivo (D) e un tubo (T)
- *Correlazioni*: il miscelatore riceve sia acqua calda che acqua fredda e le miscela fornendole ad un tubo. Il dispositivo controlla la temperatura all'uscita del tubo e regola il flusso di acqua calda al miscelatore.
- *Finalità*: mantenere costante la temperatura dell'acqua all'uscita del tubo.

Risulta molto utile schematizzare il significato di tutti i simboli usati dal problema costruendo la seguente tabella:

TC	temperatura acqua calda (gradi centigradi)
TF	temperatura acqua fredda (gradi centigradi)
TM	temperatura acqua miscelata (gradi centigradi)
FC	flusso acqua calda (l/s)
FF	flusso acqua fredda (l/s)
DFC	variazione del flusso di acqua calda
K	costante per il calcolo di DFC
RIT	tempo di ritardo
T	temperatura dell'acqua all'uscita del tubo
TZERO	temperatura all'istante iniziale in tutto il tubo
FZERO	flusso acqua calda iniziale
N	durata della simulazione (secondi)

Il sistema parte con uno stato iniziale (TZERO, FZERO) e con le impostazioni (RIT, K, N) definite dall'utente e poi evolve. Si suppone che i dati di input (FF, TF, TC) vengano inseriti ogni secondo dall'utente, tramite un'interfaccia grafica.

In questo problema vengono utilizzate le componenti grafiche del package AWT.

Riprendendo le parti del sistema individuate precedentemente, si passa agli oggetti, descrivendo il loro stato e i loro comportamenti.

Il diagramma della classe *Miscelatore* è mostrato nella figura:

Miscelatore
TC TF FC FF tub
TM miscela

Gli attributi pubblici TC,TF,FC,FF corrispondono ai valori dell'acqua di ingresso del miscelatore. L'attributo privato *tub* è un riferimento al tubo che è collegato al miscelatore; è usato dal miscelatore per sapere a chi deve fornire l'acqua.

Il metodo TM calcola, in base agli attributi, la temperatura dell'acqua miscelata.

Il metodo *miscela* aggiunge l'acqua miscelata al tubo.

Il diagramma della classe *Dispositivo* è mostrato nella figura:

Dispositivo
K tub misc
regolaFC

L'attributo privato K è la costante usata nel calcolo di DFC.
L'attributo privato *tub* è un riferimento al tubo che è collegato al dispositivo e serve per misurare la temperatura di uscita.
L'attributo privato *misc* è un riferimento al miscelatore e serve per impostare il valore di FC.
Il metodo pubblico *regolaFC* calcola il valore di DFC e imposta il flusso FC del miscelatore.

Con la classe *Tubo* si vuole descrivere la seguente situazione: il tubo riceve l'acqua (con temperatura TM) dal miscelatore al tempo *t*; quest'acqua resta nel tubo per un certo numero di secondi indicati da RIT e poi compare all'uscita. Ad ogni secondo viene fornita, in ingresso al tubo, acqua con temperature TM diverse. Questa non compare subito all'uscita del tubo, ma vi compare dopo un periodo di ritardo RIT. Per modellare questa situazione si può usare un vettore con RIT posizioni, ad ognuna delle quali corrisponde una certa temperatura dell'acqua: le nuove temperature vengono aggiunte da un lato e, dopo aver percorso tutto il vettore, escono dal tubo.

Tubo
temp TZERO
misura Tadd

L'attributo privato *temp* rappresenta il vettore contenente i valori della temperatura.
L'attributo pubblico TZERO è la temperatura iniziale.
Il metodo pubblico *misuraT* misura la temperatura dell'acqua all'uscita del tubo.
Il metodo pubblico *add* è usato dal miscelatore per aggiungere una nuova TM all'inizio del vettore. Questo causa l'eliminazione di una temperatura alla fine del vettore.
Oltre alle tre classi descritte precedentemente, ci sono quelle che si occupano di gestire l'interfaccia grafica e l'interazione con l'utente.

DESCRIZIONE DELL'INTERFACCIA

L'interfaccia di questo programma è sia a caratteri che grafica. Le impostazioni e i valori iniziali vengono letti al *Prompt dei comandi*, successivamente viene aperta una finestra per gestire l'input dell'utente e per visualizzare la situazione secondo per secondo. All'interno della finestra, si posiziona nella parte sinistra un pannello in cui l'utente può inserire tre valori (FF, TF, TC). Nella parte centrale c'è un'area di testo che mostra in forma tabellare tutti i valori richiesti (FF, TF, FC, TC, T, TM).

IMPLEMENTAZIONE DELLA CLASSE (*Miscelatore.java*)

```
class Miscelatore
{
   public float TC, TF, FC, FF;
   private Tubo tub;

   // costruttore
   public Miscelatore(float FZERO, Tubo tub)
   {
      FC = FZERO;
      this.tub = tub;
   }
```

```
    // calcola la temperatura dell'acqua miscelata
    public float TM()
    {
      return (float) ((TC*FC) + (TF*FF)) / (FC+FF);
    }

    // inserisce nel tubo il valore di TM
    public void miscela()
    {
      tub.add(TM());
    }
  }
```

Il costruttore di questa classe riceve come parametri il valore iniziale di FC e il tubo a cui è collegata. Quest'ultimo oggetto viene usato nel metodo *miscela* per aggiungere una nuova temperatura al tubo. La classe contiene quattro variabili pubbliche che verranno modificate dalle altre classi.

IMPLEMENTAZIONE DELLA CLASSE (*Dispositivo.java*)

```
class Dispositivo
{
  private float K;
  private Tubo tub;
  private Miscelatore misc;

  // costruttore
  public Dispositivo(float K, Tubo tub, Miscelatore misc)
  {
    this.K = K;
    this.tub = tub;
    this.misc = misc;
  }

  // calcola il valore di DFC e imposta il flusso FC del miscelatore
  public void regolaFC()
  {
    float DFC = -K*(tub.misuraT() - tub.TZERO);
    misc.FC += DFC;
  }
}
```

L'unico metodo è *regolaFC* che viene richiamato ogni volta che l'utente inserisce nuovi valori per FF, TF e TC.

IMPLEMENTAZIONE DELLA CLASSE (*Tubo.java*)

```
import java.util.*;

class Tubo
{
   private Vector temp = new Vector();
   public final float TZERO;

   // costruttore
   public Tubo(int RIT, float TZERO)
   {
      // imposta il vettore al valore iniziale
      for(int i=0; i<RIT; i++)
      {
         temp.addElement(new Float(TZERO));
      }
      this.TZERO = TZERO;
   }

   // misura la temperatura dell'acqua all'uscita del tubo
   public float misuraT()
   {
      Float f = (Float) temp.elementAt(0);
      return f.floatValue();
   }

   // aggiunge una nuova TM eliminando quella all'uscita
   public void add(float TM)
   {
      temp.removeElementAt(0);
      temp.addElement(new Float(TM));
   }
}
```

Questa classe importa il package *java.util* perché contiene la classe *Vector*, necessaria per gestisce il vettore che registra la temperatura dell'acqua nel tubo.
Il costruttore crea un vettore di dimensione RIT e imposta il valore iniziale di tutte le temperature a TZERO. Nel vettore non si possono aggiungere i tipi di dati semplici come i *float*, quindi vengono creati degli oggetti di classe *Float* che rappresentano lo stesso numero. L'uscita del tubo corrisponde all'indice 0 del vettore, per questo la misurazione della temperatura T avviene leggendo l'elemento di indice 0. L'aggiunta di nuove temperature viene fatta in fondo al vettore con il metodo *add*. Questo metodo oltre ad aggiungere una nuova temperatura, elimina anche quella all'uscita del tubo; in questo modo la dimensione del vettore resta costante.

Mancano ancora due classi per completare il programma. La prima è la classe principale del programma, quella che contiene il metodo *main*, mentre la seconda è la classe che gestisce l'input dell'utente e l'evoluzione del sistema.

PROGRAMMA JAVA (*Sistema.java*)

```
import java.awt.*;
import java.io.*;

class Sistema
{
   public static void main (String args[])
   {
      InputStreamReader input = new InputStreamReader(System.in);
      BufferedReader tastiera = new BufferedReader(input);

      // valori iniziali inseriti da tastiera
      int RIT, N;
      float K, TZERO, FZERO;
      // lettura dei valori iniziali
      . . .
```

I due package importati servono per usare le classi che gestiscono l'input da tastiera e per la creazione della finestra.
Il primo compito del *main* è quello di leggere i valori che descrivono lo stato iniziale e le impostazioni dell'utente. La lettura da tastiera viene effettuata con le istruzioni viste nel capitolo 3 e, se va a buon fine, vengono impostati i valori di RIT, N, K, TZERO, FZERO.
Sfruttando questi valori, si creano le istanze delle tre classi definite in precedenza.

```
      // crea gli oggetti che gestiscono il sistema
      Tubo tub = new Tubo(RIT, TZERO);
      Miscelatore misc = new Miscelatore(FZERO, tub);
      Dispositivo disp = new Dispositivo(K, tub, misc);
```

Questi tre oggetti devono essere creati indicando i giusti parametri, che corrispondono a quelli definiti dal costruttore di ogni classe. Si crea per primo l'oggetto *tub*, perché deve essere passato come parametro ai successivi due oggetti. In questo modo gli oggetti *misc* e *disp* possono usare l'oggetto *tub* e i suoi metodi.

La terza parte della classe *Sistema* ha il compito di creare l'interfaccia grafica, aprendo una finestra e posizionando le componenti.

```
      // crea e apre la finestra
      Frame f = new Frame("Sistema");
      TextArea a = new TextArea(50, 15);
      PannelloComandi pc = new PannelloComandi(N, disp, misc, tub, a);

      a.setText("sec.\tFF\tTF\tFC\tTC\tT\tTM\n");
      f.add(pc, "West");
      f.add(a, "Center");
      f.addWindowListener(new GestoreFinestra());
      f.setSize (500, 200);
      f.setVisible (true);
   }
}
```

Le due componenti sono un'area di testo e un pannello dei comandi. L'area di testo viene inizializzata con il metodo *setText* (i caratteri \t indicano la tabulazione). Il pannello dei comandi è definito dalla classe che vedremo successivamente e rappresenta una componente *Panel*. Il pannello viene inserito nella parte sinistra della finestra, mentre l'area di testo nella parte centrale. Alla finestra viene associato un gestore che è definito dalla classe *GestoreFinestra*, identica alla classe che è stata dichiarata nel'inserto dopo il capitolo 5.

IMPLEMENTAZIONE DELLA CLASSE (*PannelloComandi.java*)

```
import java.awt.*;
import java.awt.event.*;

class PannelloComandi extends Panel implements ActionListener
{
   private int N;
   private Dispositivo disp;
   private Miscelatore misc;
   private Tubo tub;
   private TextArea a;
   private int tempo;
   private Label titolo;
   private Button succ;
   private TextField valFF, valTF, valTC;
```

In questa parte vengono definiti tutti gli attributi della classe: i primi 5 sono dei riferimenti agli oggetti definiti nella classe *Sistema* e che vengono ricevuti come parametri dal costruttore; l'attributo *tempo* viene usato per scandire il passaggio del tempo e viene incrementato ogni volta che l'utente inserisce dei nuovi valori; gli ultimi sono le componenti dell'interfaccia grafica.
Vengono importati i package per poter usare le componenti e per gestire l'evento di pressione del mouse. L'evento viene gestito da questa classe, come indica la parola chiave *implements* seguita dall'interfaccia *ActionListener*.
I compiti del costruttore di questa classe sono due: impostare i riferimenti ricevuti come parametri e disporre le componenti grafiche nel pannello.

```
   // costruttore
   public PannelloComandi(int N,
                          Dispositivo disp,
                          Miscelatore misc,
                          Tubo tub,
                          TextArea a)
   {
      // imposta i riferimenti
      this.N = N;
      this.disp = disp;
      this.misc = misc;
      this.tub = tub;
      this.a = a;
      tempo = 1;
```

```
        // dispone le componenti
        titolo = new Label("TEMPO "+tempo);
        valFF = new TextField(6);
        valTF = new TextField(6);
        valTC = new TextField(6);
        succ = new Button("Successivo");
        succ.addActionListener(this);
        setLayout(new GridLayout(8,1));
        add(titolo);
        add(new Label("Valore FF:", Label.CENTER));
        add(valFF);
        add(new Label("Valore TF:", Label.CENTER));
        add(valTF);
        add(new Label("Valore TC:", Label.CENTER));
        add(valTC);
        add(succ);
    }
```

Il costruttore indica che il gestore del bottone *succ* è questa classe. Si deve quindi ridefinire il metodo *actionPerformed*, che viene richiamato ogni volta che l'utente preme il pulsante. Questo metodo è mostrato di seguito e rappresenta il cuore di tutto il programma.

```
    // gestore del bottone
    public void actionPerformed(ActionEvent e)
    {
        String bottone = e.getActionCommand();
        float FF,TF,TC;

        if (bottone.equals("Successivo"))
        {
            // legge i valori
            try
            {
                String val;
                val = valFF.getText();
                FF = Float.valueOf(val).floatValue();
                val = valTF.getText();
                TF = Float.valueOf(val).floatValue();
                val = valTC.getText();
                TC = Float.valueOf(val).floatValue();
            }
            catch(Exception exc)
            {
                // non modifica i valori
                return;
            }
```

Se è stato premuto il pulsante, vengono letti i valori che l'utente ha inserito. Se si è verificato qualche errore, il metodo viene interrotto e i valori inseriti non provocano effetti; altrimenti si procede impostando i nuovi valori del miscelatore.

```
        // imposta i valori
        misc.FF = FF;
        misc.TF = TF;
        misc.TC = TC;
```

A questo punto viene chiesto al Dispositivo di calcolare ed impostare il nuovo valore di FC, successivamente viene richiamato il metodo *miscela* in modo che il Miscelatore possa aggiungere al tubo il nuovo valore di TM. Infine vengono mostrati tutti i valori nell'area di testo.

```
        disp.regolaFC();
        misc.miscela();
        mostraValori();
```

Le ultime operazioni compiute da questo metodo incrementano il tempo per consentire l'input dei successivi valori dell'utente. Quando il tempo supera il valore *N*, viene bloccato il pulsante per negare l'inserimento di ulteriori valori.

```
        tempo += 1;
        titolo.setText("TEMPO "+tempo);
        if (tempo > N)
        {
           succ.setEnabled(false);
           titolo.setText("FINE.");
        }
     }
  }
```

I metodi che completano la classe *PannelloComandi* sono usati per formattare i valori e per mostrarli nell'area di testo in forma tabellare.

```
  // visualizza tutti i valori nell'area di testo
  private void mostraValori()
  {
     a.append(tempo+"\t");
     a.append(arrotonda(misc.FF)+"\t");
     a.append(arrotonda(misc.TF)+"\t");
     a.append(arrotonda(misc.FC)+"\t");
     a.append(arrotonda(misc.TC)+"\t");
     a.append(arrotonda(tub.misuraT())+"\t");
     a.append(arrotonda(misc.TM())+"\n");
  }

  // arrotonda alla prima cifra decimale
  private float arrotonda(float f)
  {
     return (float) (Math.round(f*10)) / 10;
  }
}
```