Unità B.1 – Gestione eventi 1	MODULO L2

(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- Oggetto origine
- Oggetto destinatario
- Generatore di eventi
- Oggetto ascoltatore
- Registrazione di un ascoltatore
- Classi di ascolto
- **WindowListener**
- **WindowAdapter**
- **WindowFocusListener**
- **ActionListener**
- Classi **Adapter**

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. Qual è il meccanismo di funzionamento della *programmazione ad eventi*?
2. Quali sono gli oggetti coinvolti nella programmazione *event driven*?
3. Cosa è un *ascoltatore*?
4. Cosa contiene una *classe di ascolto*?
5. Come si sviluppa la *fase di gestione* degli eventi in un'applicazione?

B2) Competenza

1. Qual è la *classe di ascolto* per le finestre?
2. Qual è la *classe di ascolto* per i pulsanti?
3. Una volta scelto un tipo di evento, cosa indica *l'origine dell'evento*?
4. Cosa vuol dire *creare una classe di ascolto*?
5. Cosa vuol dire *implementare i metodi di una classe di ascolto*?
6. Cosa significa *registrare un evento*?

(C) ESERCIZI DI COMPRENSIONE

1. Per utilizzare gli eventi occorre utilizzare il *package* ……, che contiene le varie classi di ……..; queste sono ……, ossia classi ……. .. , caratterizzate dal fatto che i metodi non sono ……. . Una stessa classe di …… può gestire …… prodotti da diversi elementi grafici.
2. Una classe di ……. contiene tutti gli ……. riconoscibili da vari elementi grafici. Una volta implementato un metodo ……., questo deve essere …….. nell'applicazione per poter rispondere a determinati ……
3. Gli adattatori sono classi di ……. astratte che consentono di non ……. i metodi che non sono necessari. Il nome di queste classi è dato dall'oggetto origine seguito dal suffisso ……
4. Le classi di ascolto per le finestre sono la …….. , che gestisce gli eventi relativi alle operazioni sulle finestre, la ……. che gestisce il fatto che una finestra prenda o lasci il fuoco.
5. La classe di ascolto …… è in grado di gestire gli eventi prodotti da molti oggetti origine, quali …….., …… e …… Questa classe ha un unico metodo che ha nome …….. .
6. Nella programmazione ad eventi, detta in inglese …… …….. ……., l'utente compie una …….su un oggetto, detto ……dell'evento. L'oggetto ……manda un ……ad un generatore di eventi che, a sua volta, crea automaticamente un ……..evento; l'evento manda, a sua volta, un messaggio ad un altro oggetto, detto ……..; se il progettista ha implementato il ……..dell'ascoltatore e lo ha ……., l'applicazione eseguirà questo codice.
7. Scrivere l'istruzione per importare il *package* relativo agli eventi:
8. Scrivere i metodi dell'interfaccia **WindowListener**:

Evento	Metodo interfaccia WindowListener
Finestra attivata	
Finestra disattivata	
Finestra chiusa	
Riduci ad icona	
Finestra aperta	
Finestra ripristinata	
Chiudi finestra	

9. Scrivere le istruzioni per registrare in una finestra *f* ciascuno degli ascoltatori indicati nella seconda colonna della tabella seguente e creato per la classe di ascolto indicata nella prima colonna.

Classe di ascolto	Ascoltatore	Registrazione ascoltatore
WindowListener	gestoreW()	
WindowFocusListener	gestoreF()	
ActionListener	gestoreP()	

10. Indicare, per ciascuno degli ascoltatori riportati, la classe di ascolto corrispondente.

Ascoltatore	Classe corrispondente di ascolto
windowDeactivated()	
ActionPerformed()	
windowLostFocus()	
windowGainedFocus()	
windowClosed()	
windowClosing()	
windowIconified()	
windowDeiconified()	
windowOpened()	

11. Scrivere lo scopo di ciascuno dei seguenti passi della **fase di gestione** di un'interfaccia grafica.

Scelta del tipo di eventi	
Stabilire le origini degli eventi	
Creazione delle classi d'ascolto	
Implementazione dei metodi di ascolto	
Creazione ascoltatore	
Registrazione dell'ascoltatore	

12. Per ciascuno degli esercizi seguenti, completare eventuali parti mancanti, correggere eventuali errori, determinare gli output prodotti e dare una breve descrizione dell'applicazione.

a.
```
import java.awt.*;
public class ButtonEvent extends Frame
{       Button btn = new Button ("Premi");
        TextField btn_txt = new TextField (50);
        ButtonEvent ()
        {       super ("Gestione pulsante");
                setLocation(300);
                setSize(300,100);
                setLayout(null);
                setBounds(10, 30, 60,30);
                btn_txt.setBounds(50, 70, 170, 120);
                …;
                …;
                btn.addActionListener (gestorePulsante(btn_txt));
                setVisible(true);
        }
        public static void main (String args[])
        {       new ButtonEvent();
        }
}
public class gestorePulsante implements ActionListener
{       TextField txt;
        public gestorePulsante(TextField tf)
        {  this.txt=tf;   }
        public void actionPerformed (e)
        {   txt.setText("Premuto pulsante" + getActionCommand());
        }
}
```

b.
```
import java.awt.*;
import java.awt.event.*;
public class gestoreF implements WindowListener
{       public void windowClosing(…)  { …   }
        public void windowIconified(…….  e) …
        public void windowDeiconified(WindowEvent e) …
        public void windowActivated(WindowEvent e) …
        public void windowDeactivated(WindowEvent e) …
        public void windowOpened(WindowEvent e) …
        public void windowClosed(WindowEvent e) …
 }
public class Window extends Frame
{   public Window()
```

```
        {
            setTitle("Gestione finestra");
            setLocation(200);
            setSize(200,200);
            setVisible(true);
            addWindowListener(…)
        }
      public static void main(String args[])
      {   Window w = Window();
      }
    }
c.  import java.awt.*;
    import java.awt.event.ActionListener;
    import java.awt.event.ActionEvent;
    public class ButtonEvent
    { ButtonEvent ()
      {     Frame f=new Frame("Gestione pulsanti");
            f.setLocation(200,200);
            Button B1 = new Button("Premi");
            Button B2 = new Button ("Esci");
            f.add(B1, BorderLayout.SOUTH);
            f.add(B2,BorderLayout.NORTH);
            B1.addActionListener(new …);
            B2.addActionListener(new …);
            f.setVisible(true);
            f.pack();
            f.setVisible(true);
      }
      public class gestoreB implements ActionListener
      { public gestoreB()
        {  System.out.println("Ascoltatore creato");   }
        public void actionPerformed(ActionEvent e)
        { String s=e.getActionCommand();
          if (s.equals("Premi"))  System.out.println("Premuto B1");
           else  {        System.out.println("Premuto B2");
                          System.exit(0);
        }
      }
    }
      public static void main (String args[])
      { new …;   }
    } // end class
d. import java.awt.*;
    import java.awt.event.ItemListener;
    import java.awt.event.ItemEvent;
    class TestCheckBox extends Frame
    {  Checkbox ac_chk = new Checkbox("Aria condizionata");
       Checkbox ta_chk = new Checkbox("Tetto apribile");
       Checkbox ss_chk = new Checkbox("Servosterzo");
       Checkbox ve_chk = new Checkbox("Vetri elettrici");
       Label status = new Label("Totale prezzo: e " + 20000);
       TestCheckBox()
    {  super("Checkbox Example");
         setSize(300,300);
         setLocation(200,300);
         Panel gridPanel = new Panel(new GridLayout(0, 1, 50,20));
         gridPanel.add(ac_chk);
         gridPanel.add(ta_chk);
         gridPanel.add(ss_chk);
         gridPanel.add(ve_chk);
         add(gridPanel, BorderLayout.CENTER);
         ac_chk.addItemListener(new gestoreChk());
         ta_chk.addItemListener(new gestoreChk());
         ss_chk.addItemListener(new gestoreChk());
         ve_chk.addItemListener(new gestoreChk());
         add(status, BorderLayout.SOUTH);
         pack();
         setVisible(true);
         addWindowListener(new gestoreF());
       }
       public class gestoreChk …….. ItemListener
       {  public void itemStateChanged(ItemEvent evt)
            {   computeTotal();   }
            void computeTotal()
            {  int total = 25000;
               if (ac_chk.getState()) total += 510;
               if (ta_chk.getState()) total += 822;
```

```
            if (ss_chk.getState()) total += 150;
            if (ve_chk.getState()) total += 320;
            status.setText("Totale prezzo: e " + total);
        }
    }
    static public void main(String[] args)
    {  new TestCheckBox();
     }
}  // end class
```

(D) ESERCIZI DI APPLICAZIONE

1. Creare un'applicazione *ConvertiMisure*, che converta da centimetri a pollici e viceversa, prevedendo tutti i controlli che si ritengano opportuni.
2. Creare un'applicazione *Contatore* che visualizzi un valore intero, che può essere incrementato, decrementato o azzerato a scelta dell'utente, mediante opportuni controlli.
3. .Creare un'applicazione *Anagrafica* che consenta di immettere i dati personali, prevedendo opportuni controlli per la registrazione e l'annullamento.
4. Creare un'applicazione *Cruscotto* che simuli i controlli presenti nel cruscotto di un veicolo.
5. Disegnare un'interfaccia grafica che simuli l'operazione di pagamento di un bollettino postale on line:
   a. introdurre i controlli che si ritengono opportuni, una casella di testo **Totale** (per l'importo totale) e una casella di spunta **Inviato** (per segnalare l'avvenuto invio del pagamento);
   b. creare l'ascoltatore per il pulsante di **Invio** che spedisce il pagamento del bollettino. Il pulsante deve:
      - calcolare il totale da pagare (bollettino e tassa)
      - visualizzare il totale nella casella **Totale**
      - contrassegnare la casella di spunta **Inviato**, per indicare l'avvenuto invio.
   c. Scrivere nel riquadro seguente lo schema dell'ascoltatore

Oggetto origine	
Interfaccia usata	
Ascoltatore	
Gestore di evento	
Registrazione	

   d. Scrivere il codice Java del gestore di evento.
6. Disegnare un'interfaccia grafica che simuli il pannello di controllo di un'autoradio:
   a. Introdurre i controlli che si ritengono opportuni; supponendo che siano memorizzabili 6 canali, prevedere una casella di testo **Canale** (numero del canale corrente) e una casella di testo **Emittente** (nome dell'emittente);
   b. creare l'ascoltatore per il pulsante di **Avanti** che passa al canale successivo. Il pulsante deve:
      - passare al canale successivo
      - se il canale corrente è l'ultimo, ritornare al primo
      - visualizzare il canale corrente nella casella **Canale**
      - visualizzare nella casella **Emittente** l'emittente selezionata.
   c. Scrivere nel riquadro seguente lo schema dell'ascoltatore

Oggetto origine	
Interfaccia usata	
Ascoltatore	
Gestore di evento	
Registrazione	

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 1

Problema: Realizzare un'applicazione che consenta di convertire le misure di temperatura da gradi Celsius a gradi Fahreneight. La formula per la conversione è la seguente:

°F =32 + °C / 100 * 180;

Obiettivi: utilizzo di contenitori e componenti, posizionamento dei controlli, costruzione di ascoltatori

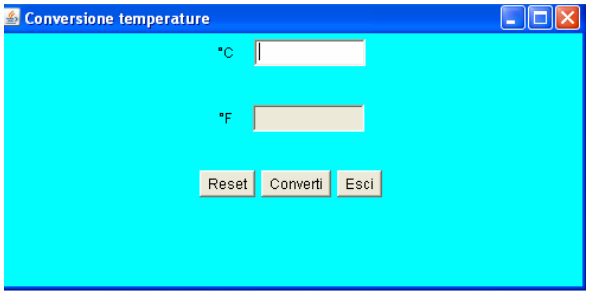
Costruzione della classe

1) Attivare l'ambiente di sviluppo (TextPad, Eclipse, ecc)
2) Creare un nuovo file e salvarlo come *ConvertiTemperature.java*. In questo file mettere gli attributi seguenti, entrambi di tipo **double**:
   a. gradi_c,
   b. gradi_f;
3) Implementare i seguenti metodi:
   1. costruttore senza parametri;
   2. costruttore con parametri;
   3. metodo modificatore **public void** set_c (**double** gc)
   4. metodo accessore **public double** get_f ()
4) Scrivere, nello stesso file o in un file separato, il metodo **main()** per testare la classe creata. Il **main()** deve leggere da input (da finestra di comando o finestra grafica) un valore di temperatura in gradi centigradi e deve stampare il corrispondente valore convertito in gradi Fahreneight. A tale scopo, verificare la seguente corrispondenza:

°C	°F
15	59
35	95
85	185
120	248

Costruzione interfaccia grafica

Occorre, ora realizzare una interfaccia grafica per la classe *ConvertiTemperature*, come quella riportata nella figura fianco.



1) Creare un file di nome *Interfaccia.java* nella stessa cartella della classe realizzata. La classe eredita da **Frame** e crea una finestra con i seguenti attributi:

Attributi	Valore
Titolo	"Conversione temperature"
Posizione	(200, 200)
Dimensioni	500, 250
Colore sfondo	**Color.cyan**

2) Successivamente creare 4 pannelli di nome *p1*, *p2*, *p3* e *p4* che rappresentano le righe su cui vengono posti i controlli (la quarta riga non si vede nella schermata di esempio sopra riportata, poiché si attiva quando l'utente chiede la conversione senza aver immesso un valore di °C).
3) creare gli oggetti grafici indicati nella tabella seguente:

Oggetto	Nome	Attributi	Significato
TextField	gradic_txt	10	Casella di testo per le temperature in °C
TextField	gradif_txt	10	Casella di testo per le temperature in °F
Label	gradic_lbl	Sinistra	Etichetta per gradic_txt
Label	gradif_lbl	Sinistra	Etichetta per gradif_txt
Label	err_lbl	Sinistra	Etichetta di errore
Button	reset_btn	"Reset"	Pulsante per il reset
Button	conv_btn	"Converti"	Pulsante per la conversione
Button	exit_btn	"Esci"	Pulsante per l'uscita

4) Aggiungere a *p1* gli oggetti *gradic_lbl* e *gradic_txt*.
5) Aggiungere a *p2* gli oggetti *gradif_lbl* e *gradif_txt*.
6) Aggiungere al pannello *p3* i 3 pulsanti creati.
7) Aggiungere a *p4* l'oggetto *err_lbl*.
8) Impostare un layout a griglia con parametri (4, 2, 10, 10)
9) Aggiungere i pannelli alla finestra.
10) Rendere visibile la finestra
11) Scrivere il metodo **main()** per la classe *Interfaccia* per testare la corretta rappresentazione dell'interfaccia grafica creata.

Costruzione ascoltatori

L'applicazione necessita di due ascoltatori:

- un ascoltatore di classe **WindowListener** per consentire la chiusura della finestra;
- un ascoltatore **ActionListener** per attivare i 3 pulsanti.

1) Nella classe *Conversione.java* aggiungere il seguente metodo:

```
public class gestoreFinestra extends WindowAdapter
{   public void windowClosing(WindowEvent e)
    {   System.exit(0);}
}
```

Usando la classe **WindowAdapter** invece di **WindowListener** possiamo evitare di scrivere le firme dei metodi non utilizzati. In caso contrario, occorre scrivere tutte le firme e lasciare per esse le parentesi graffe vuote.

2) Provare l'ascoltatore creato e verificare la chiusura della finestra.
3) Sempre nella classe *Conversione.java* aggiungere il metodo indicato nella figura seguente. Questo metodo:
   a. tramite una **switch** sull'oggetto e.**getActionCommand**() stabilisce quale è il pulsante che ha richiesto l'evento;
   b. attiva l'evento in base al pulsante (effettua il reset, la conversione o l'uscita);
   c. utilizza, nel caso della conversione, un oggetto di classe *ClasseConversione* classe creata in precedenza e già testata.

```
public class gestorePulsanti implements ActionListener
{           public void actionPerformed(ActionEvent e)
            {  String s=e.getActionCommand(),
                       valore;
               double c, f;
               if (s.equals("Reset"))
               {          gradic_txt.setText("");          gradif_txt.setText("");
                          err_lbl.setText(" ");
               }
               else
               if (s.equals("Converti"))
               {          Conversione conv = new Conversione();
                          try
                          {  conv.set_c(Double.parseDouble(gradic_txt.getText()));
                             gradif_txt.setText("" + conv.get_f());
                          }
                          catch (NumberFormatException ex)
                       {  err_lbl.setText("Immettere un valore!");
                          }
               }
               else   System.exit(0);
            }
} // end gestorePulsanti()
```

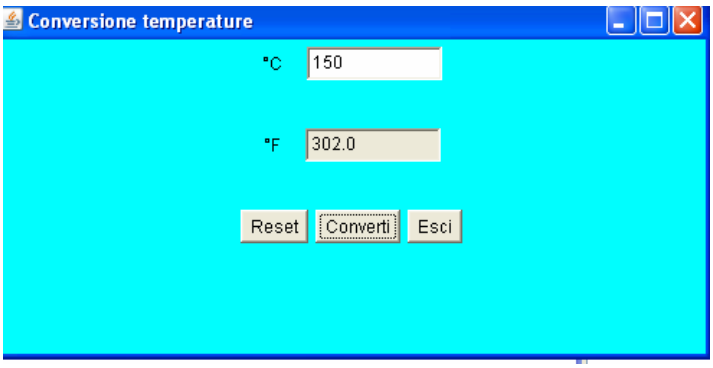


Fig. 1 - Un esempio di istanza dell'applicazione Conversione

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 2

Problema: Realizzazione di un'applicazione con interfaccia grafica per il calcolo del codice fiscale.
Obiettivi: elaborazione di stringhe e tabelle, costruzione di un'interfaccia grafica, uso e posizionamento di controlli, realizzazione di ascoltatori

Sviluppo. Il Codice Fiscale (CF) è un "codice" (lo dice la parola stessa), che identifica in modo univoco le persone che sono iscritte nei registri dell'anagrafe tributaria cioè i dati che servono poi per fare funzionare tutto il sistema tributario (il "fisco", da qui "fiscale"), quindi pagamento tasse, imposte, e così via. Il CF è un codice alfanumerico (composto da lettere e numeri) di 16 caratteri.
I primi 15 sono relativi ai dati personali (nome, cognome, sesso, data di nascita e luogo di nascita) mentre l'ultimo è un carattere di controllo che viene calcolato con delle formule applicate ai precedenti 15 caratteri.

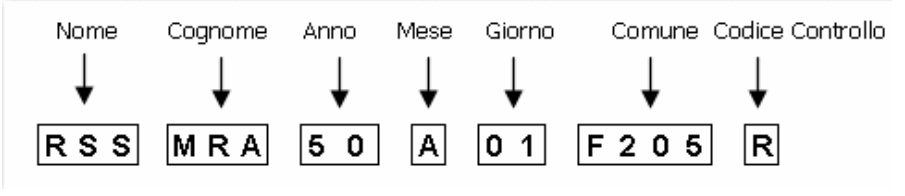


Fig. 2 - Un esempio di codice fiscale

si può notare (v. Fig. 2), è composto dai seguenti blocchi:
- 3 lettere per il cognome;
- 3 lettere per il nome;
- l'anno di nascita (numero);
- il mese della data di nascita (lettera);

Cognome
Sono necessari, come detto prima, 3 caratteri per rappresentare il cognome, in particolare la prima la seconda e la terza consonante del cognome.
È possibile che le consonanti siano meno di tre: in questo caso occorre aggiungere le vocali nell'ordine in cui compaiono nel cognome.
Per cognomi più corti di 3 caratteri è possibile sostituire il carattere mancante con la lettera X.
Chiaramente, se ci sono cognomi con più parti è necessario rimuovere gli spazi e considerare tutto come un cognome unico.
Esempi:
(Normale) Cognome : "ROSSI" - Codice Cognome : "RSS"
(Solo due consonanti) Cognome : "RIVA" - Codice Cognome : "RVI"
(Cognome minore di 3 car.) Cognome : "RE" - Codice Cognome : "REX"
(Cognome composto) Cognome : "DE CRESCENZO" - Codice Cognome : "DCR"
Nome
Per il nome il discorso è simile: qui abbiamo bisogno della prima, la terza e la quarta consonante. Anche qui potremmo trovarci nella situazione di un numero di consonanti minore di 3, nel qual caso si aggiungono le vocali.
Ripetiamo, anche qui, che se il nome è più corto di 3 lettere è possibile sostituire i caratteri mancanti con delle X.
Anche in questo caso, se fosse composto da più nomi, bisogna considerarlo tutto assieme.
- Se il nome il giorno della data di nascita (numero);
- il codice del comune di nascita;
- il carattere di controllo

Vediamo con quali algoritmi si possono ricavare i diversi blocchi.
Esempi:
(Normale) Nome: "MARTA" - Codice Nome: "MRT"
(Solo due consonanti) Nome : "SALA" - Codice Nome : "SLA"
(Nome minore di 3 car.) Nome: "AL" - Codice Nome : "LAX"
(Nome composto) Nome : "MARIA PIA" - Codice Nome : "MRP"
Anno di nascita
Per l'anno vengono prese semplicemente le ultime due cifre.
Esempio:
Anno : 1970 - Codice Anno: 70
Mese
Per quanto riguarda il mese ci si serve della tabella di conversione mostrata a fianco, che associa ad ogni mese una lettera maiuscola.

Lettera	Mese
A	Gennaio
B	Febbraio
C	Marzo
D	Aprile
E	Maggio
H	Giugno
L	Luglio
M	Agosto
P	Settembre
R	Ottobre
S	Novembre
T	Dicembre

Giorno
In questo caso è sufficiente riportare il numero del giorno, tenendo conto che per le donne questo numero dev'essere aumentato di 40.
Esempi:
Uomo nato il : 22/10/1980 - Codice Giorno: 22
Donna nata il : 22/10/1980 - Codice Giorno: 62
Comune di nascita
È composto da quattro caratteri alfanumerici e rappresenta il codice ufficiale dei comuni italiani.; è contenuto in appositi database che contengono tutti i comuni d'Italia con relativi codici.

Esempi:
Comune : Brescia - Codice Comune :B157

Codice di controllo

Rimane l'ultimo carattere: il codice di controllo, che è forse la procedura più complessa. Consideriamo la seguente tabella (Fig. 3).

Si comincia con il prendere i caratteri del codice fiscale fin qui calcolato che sono 15, si prendono quelli in posizione pari e si convertono con i numeri corrispondenti della prima tabella. Tutti questi numeri vengono sommati.

Allo stesso modo, si trattano i caratteri in posizione dispari che devono essere convertiti però utilizzando la seconda tabella e che vengono sommati.

I valori ottenuti vengono a loro volta sommati e il totale viene diviso per 26. Il resto della divisione dev'essere convertito usando l'ultima tabella. Il carattere corrispondente è il codice di controllo!

Caratteri posizione pari

0=0	1=1	2=2	3=3	4=4	5=5	6=6	7=7	8=8	9=9	A=0	B=1	C=2
D=3	E=4	F=5	G=6	H=7	I=8	J=9	K=10	L=11	M=12	N=13	O=14	P=15
Q=16	R=17	S=18	T=19	U=20	V=21	W=22	X=23	Y=24	Z=25			

Caratteri posizione dispari

0=1	1=0	2=5	3=7	4=9	5=13	6=15	7=17	8=19	9=21	A=1	B=0	C=5
D=7	E=9	F=13	G=15	H=17	I=19	J=21	K=2	L=4	M=18	N=20	O=11	P=3
Q=6	R=8	S=12	T=14	U=16	V=10	W=22	X=25	Y=24	Z=23			

Carattere di controllo

0=A	1=B	2=C	3=D	4=E	5=F	6=G	7=H	8=I	9=J	10=K	11=L	12=M
13=N	14=O	15=P	16=Q	17=R	18=S	19=T	20=U	21=V	22=W	23=X	24=Y	25=Z

Fig. 3 - Tabella per la conversione dei caratteri

1) Lanciare l'ambiente di sviluppo (Texpad, Eclipse, ecc).
2) Creare la classe *CodiceFiscale.java*.
3) Scrivere il codice Java creando apposite funzioni che implementano le operazioni descritte in precedenza.
4) Testare l'applicazione in modalità testo (usare la console)
5) Se si vuole dotare l'applicazione di una interfaccia grafica, creare l'applicazione relativa e aggiungere ad essa la creazione di un oggetto *CodiceFiscale.java.*
6) Testare il funzionamento completo dell'applicazione.
7) Redigere la documentazione di progetto, contenente l'analisi del problema affrontato, le tabelle di impostazione delle proprietà dei controlli, il codice implementato e l'immagine dell'interfaccia creata.