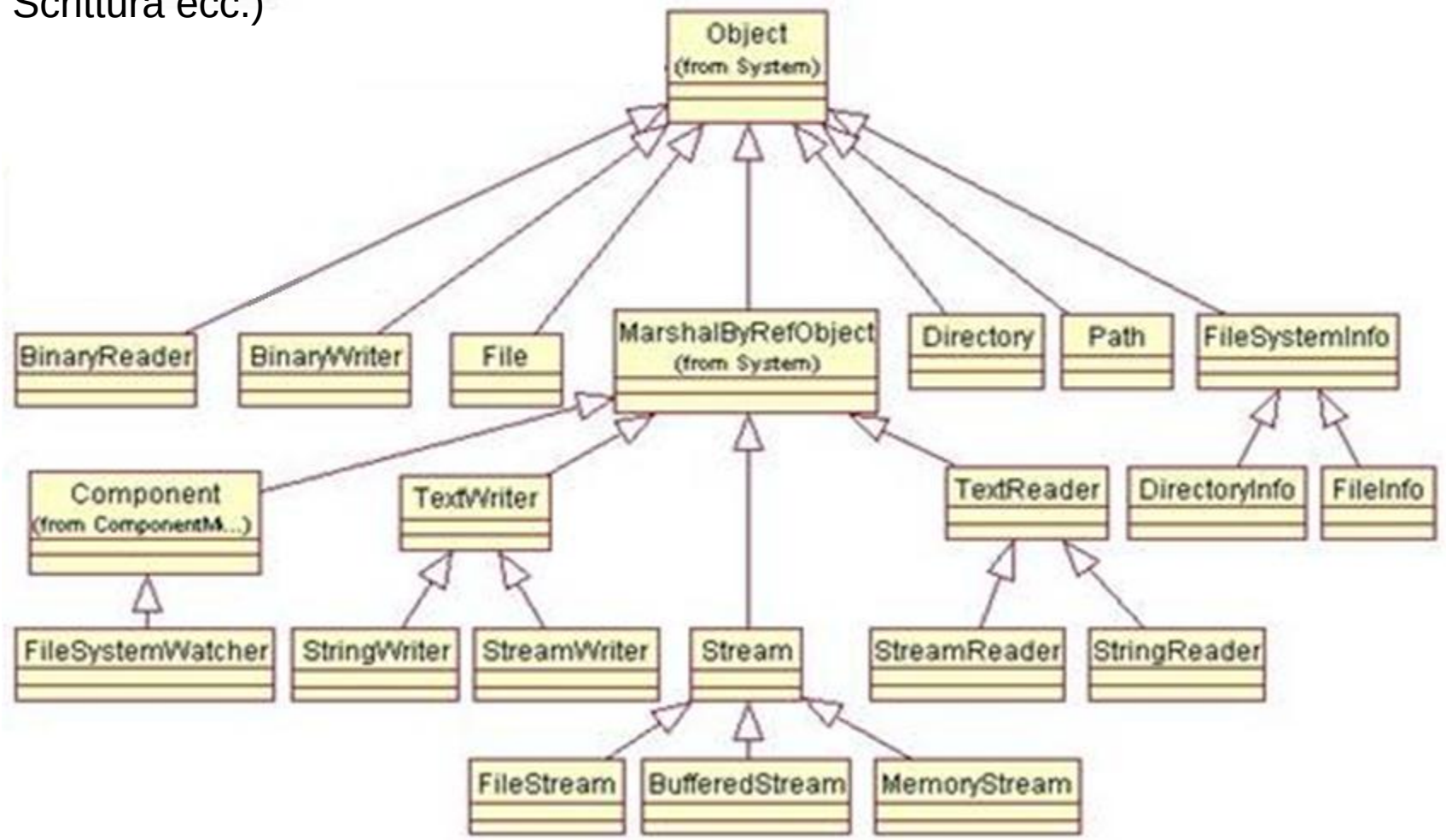
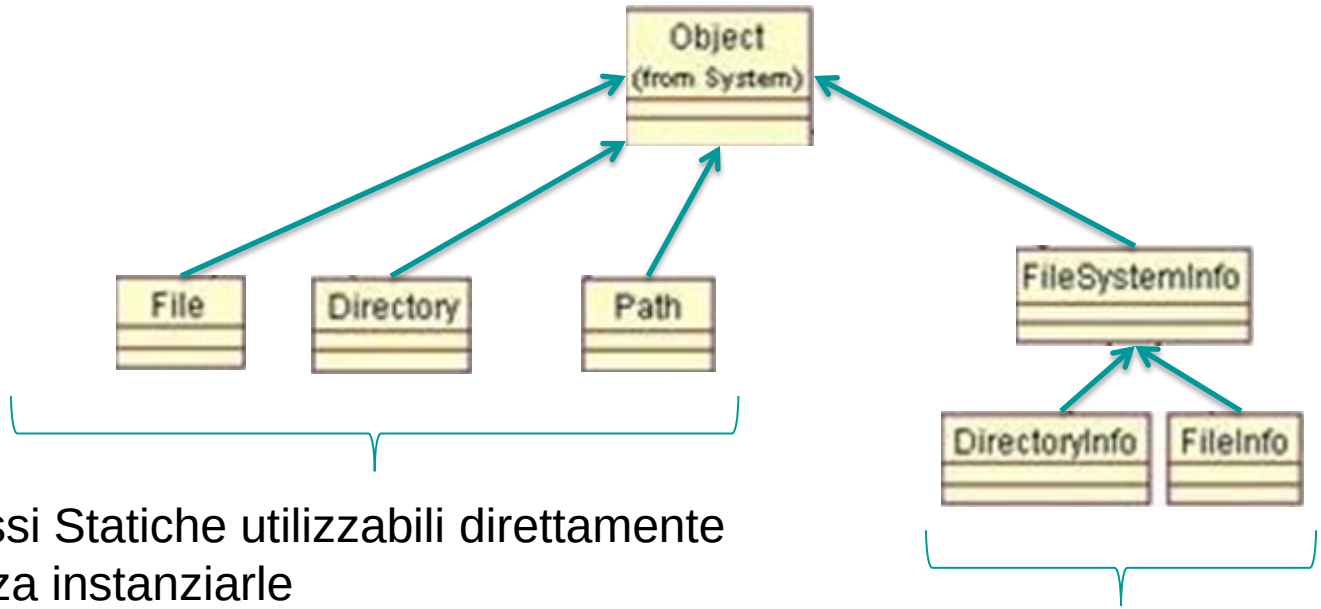
File e Stream

Prof. Francesco Accarino
IIS Altiero Spinelli Sesto San Giovanni

C# mette a disposizione numerose classi per lavorare con il File System di Windows e poter svolgere tutte quelle operazioni che riguardano files e directorys (Ottenere Informazioni , Creazione, Cancellazione, Copia Lettura Scrittura ecc.)



Per ottenere Informazioni su File e Directory o fare delle operazioni tipo: Creazione cancellazione copia ecc.



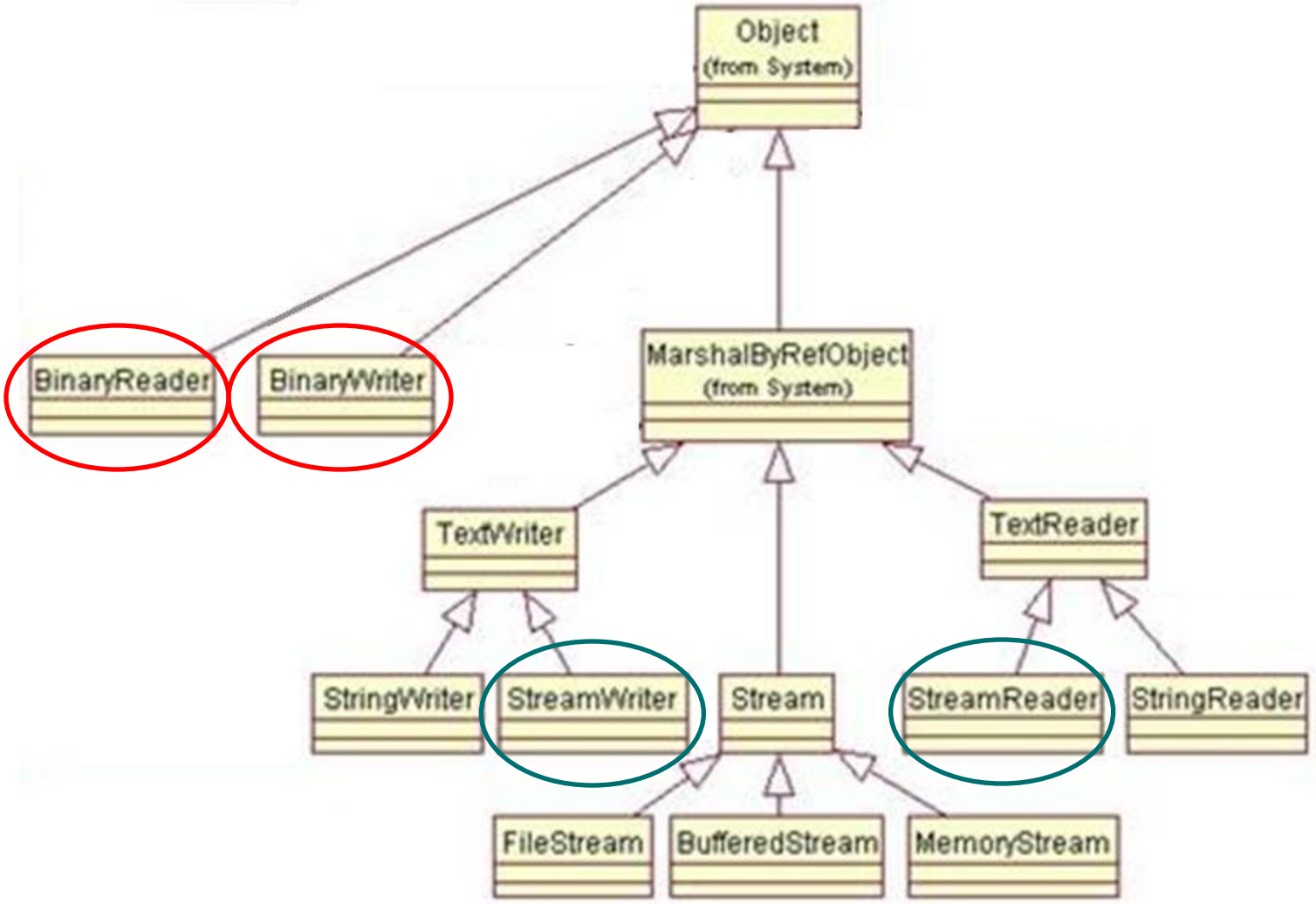
Classi Statiche utilizzabili direttamente senza instanziarle

Classi che bisogna instanziare per poterle utilizzare (più riferimenti allo stesso oggetto)

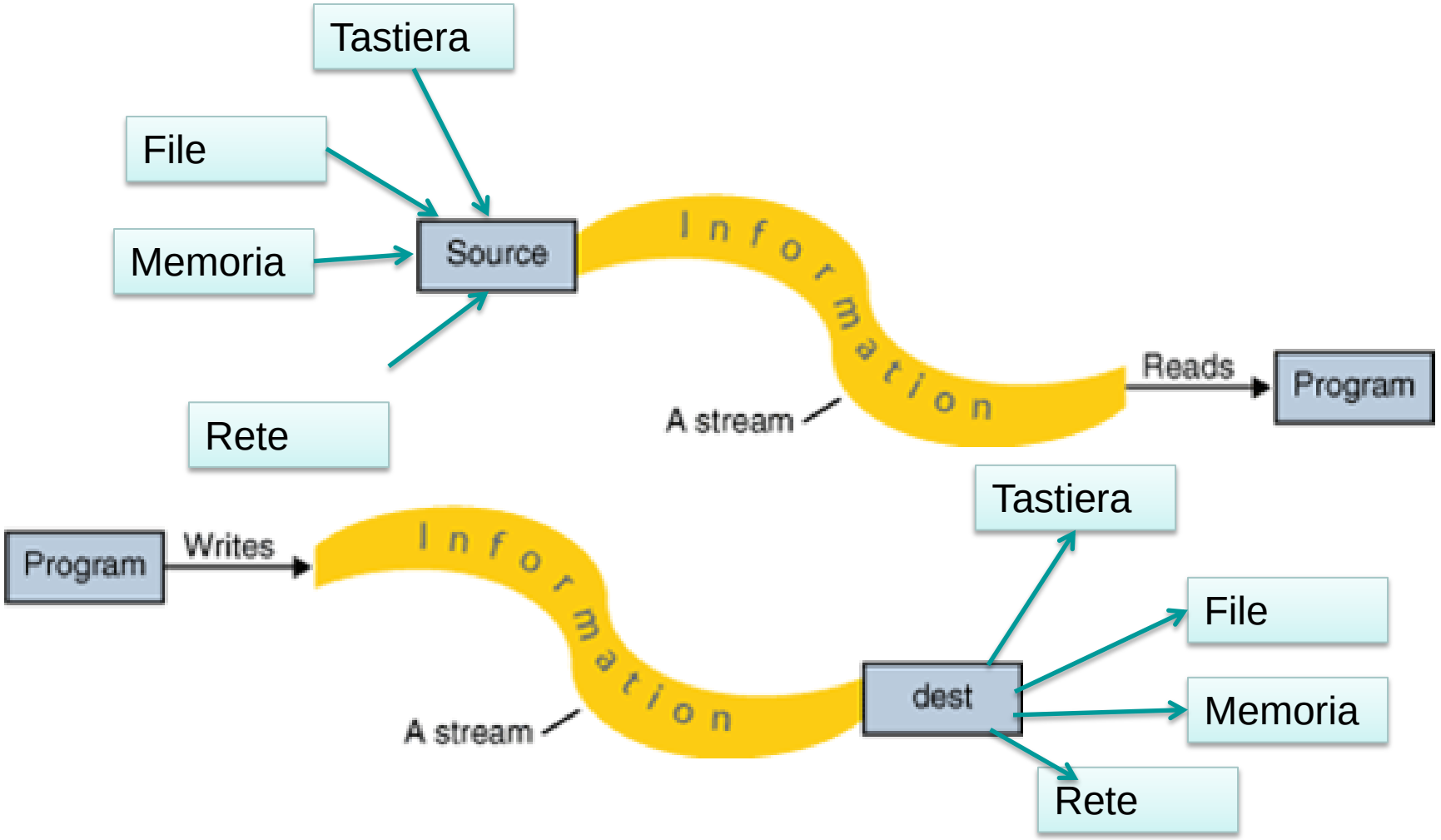
La classe statica File mette a disposizione svariati metodi per manipolare i file

Member	Description
Copy()	Copies the specified file to the specified target path.
Create()	Creates the specified file.
Delete()	Deletes the specified file.
Exists()	Returns Boolean value indicating whether the specified file exists.
GetAttributes()	Returns an object of type System.IO.FileAttributes which contain different information regarding file like whether its is hidden or not.
GetCreationTime()	Returns an object of type DateTime that represents the date time of the creation of this file.
GetLastAccessTime()	Returns an object of type DateTime that represents the date time of the last access to this file.
GetLastWriteTime()	Returns an object of type DateTime that represents the date time of the last write action to this file.
Move()	Moves the specified file to the specified path.
Open()	Opens the specified file and returns the System.IO.FileStream object for this file.
OpenRead()	Opens the specified file for reading purpose and returns a read only System.IO.FileStream object for this file.
OpenWrite()	Opens the specified file for reading purpose and returns a read/write System.IO.FileStream object for this file.
SetAttributes()	Accepts an object of type System.IO.FileAttributes which contain different information regarding file and set these attributes to the file.

Per Leggere o scrivere informazioni sui file si utilizzano delle astrazioni molto potenti: i cosiddetti: “Stream”; e si differenzia in file binari e file di testo

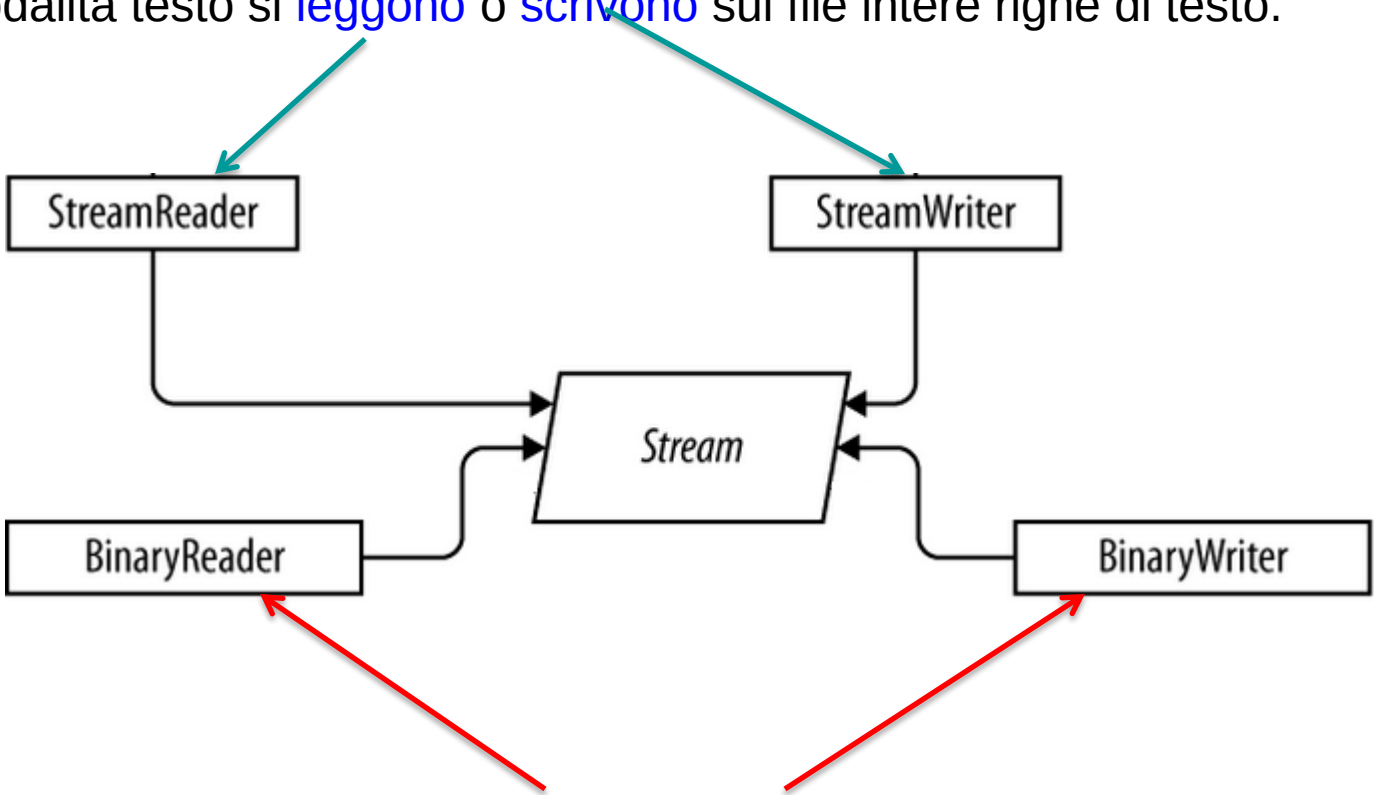


Gli Stream sono un'astrazione messa a disposizione dai moderni Sistemi Operativi che permettono alle applicazioni di prelevare informazioni da varie sorgenti o inviare informazioni a varie destinazioni.



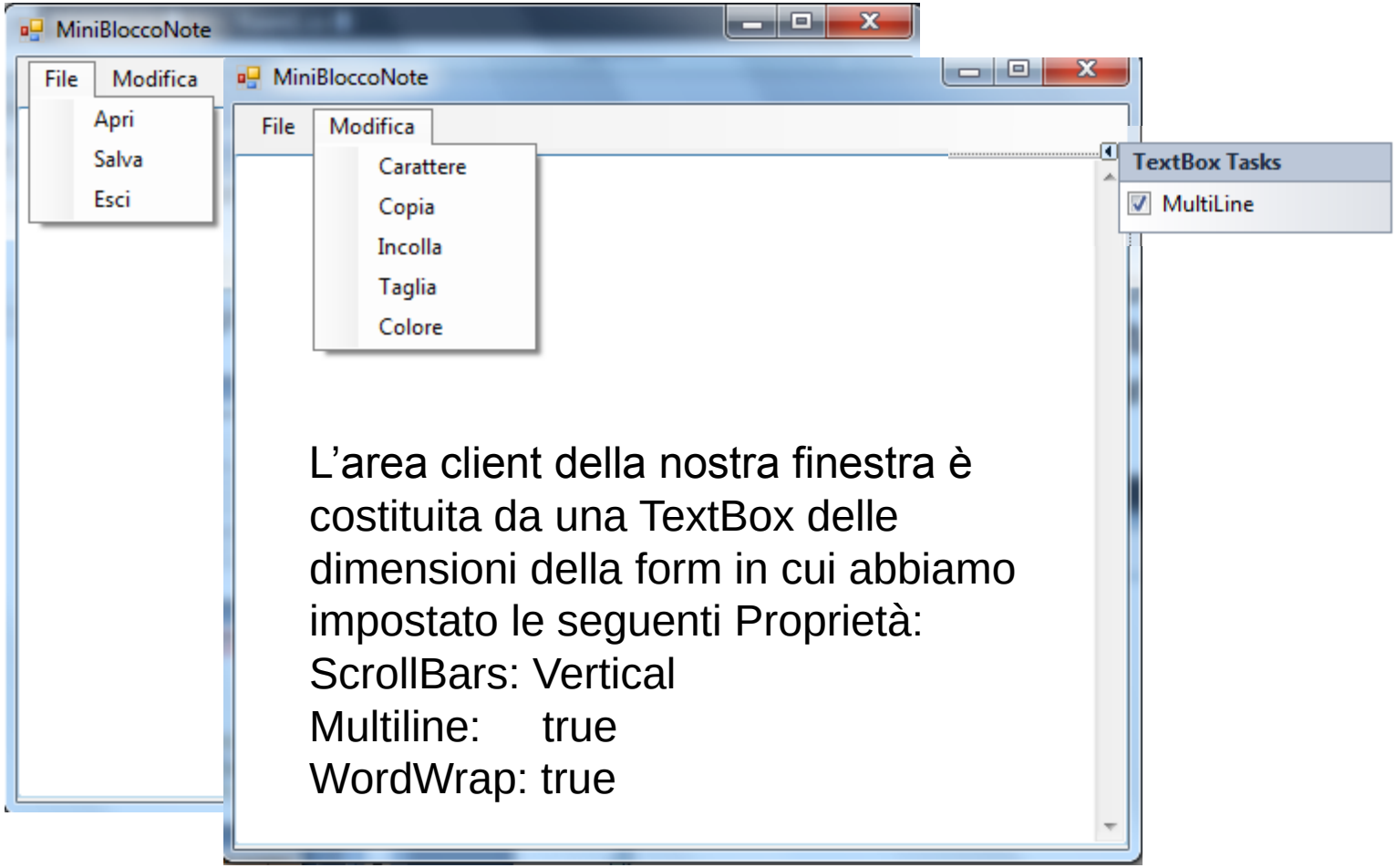
Riassumendo Possiamo quindi dire che una volta ottenuto uno Stream cioè un canale di comunicazione con il file (ottenibile con l'operazione di apertura del file stesso in cui viene anche specificato la modalità di utilizzzo attraverso vari parametri che vedremo in seguito) è possibile utilizzare degli oggetti per poter leggere e scrivere sullo Stream utilizzando la modalità testo o la modalità binaria.

Con la modalità testo si leggono o scrivono sul file intere righe di testo.



Con la modalità binaria è possibile leggere o scrivere dati in formato primitivo

Uso di File di testo: Esercitazione Semplice uso di file di testo (creazione, lettura e scrittura). Lo scopo dell'esercitazione è imparare ad usare i file di testo e nello stesso tempo imparare ad usare I Menu e i dialogBox Incorporati.



L'area client della nostra finestra è costituita da una TextBox delle dimensioni della form in cui abbiamo impostato le seguenti Proprietà:
ScrollBars: Vertical
Multiline: true
WordWrap: true

Suggerimenti:

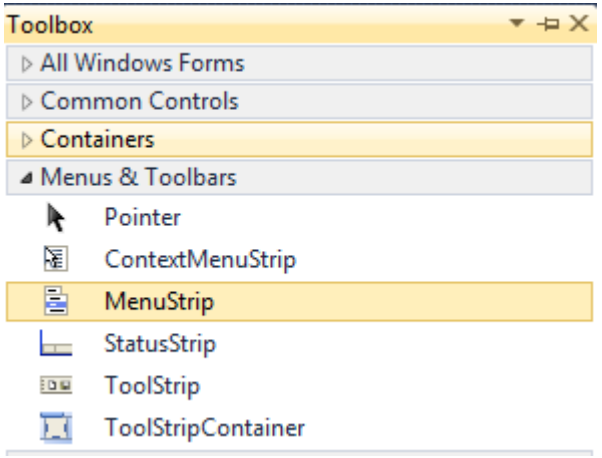
Per fare in modo che la TexBox si ridimensioni esattamente alle dimensioni della nostra finestra aggiungiamo un ascoltatore dell'evento SizeChanged alla nostra finestra con la seguente istruzione:

```
this.SizeChanged += new System.EventHandler(this.Form1_SizeChanged);
```

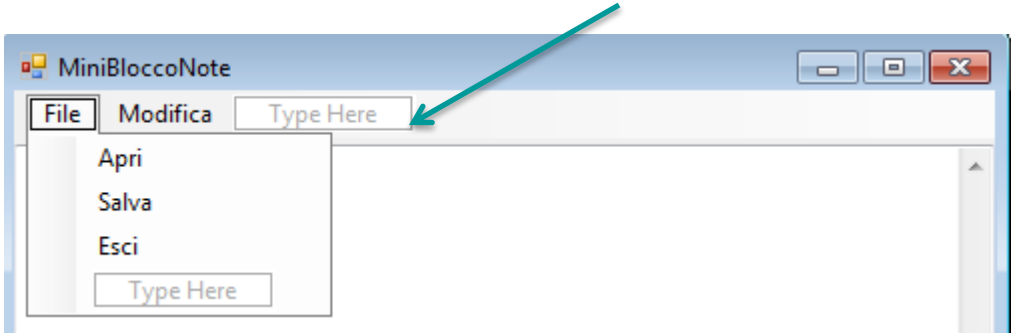
E la funzione Form1_SizeChanged ha il codice seguente:

```
private void Form1_SizeChanged(object sender, EventArgs e)
        {
            textBox1.SetBounds(0, 25, this.Width-10, this.Height-60);
        }
```

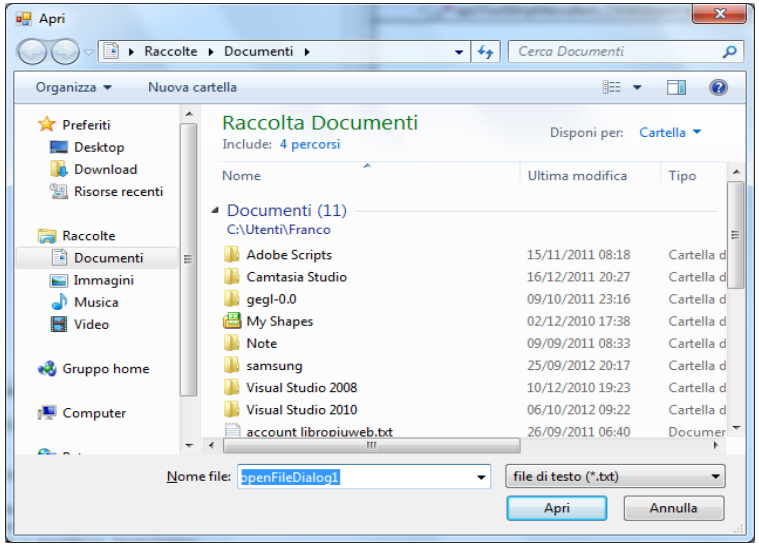
Per inserire il menu nella finestra basta cliccare sulla voce MenuStrip della ToolBox

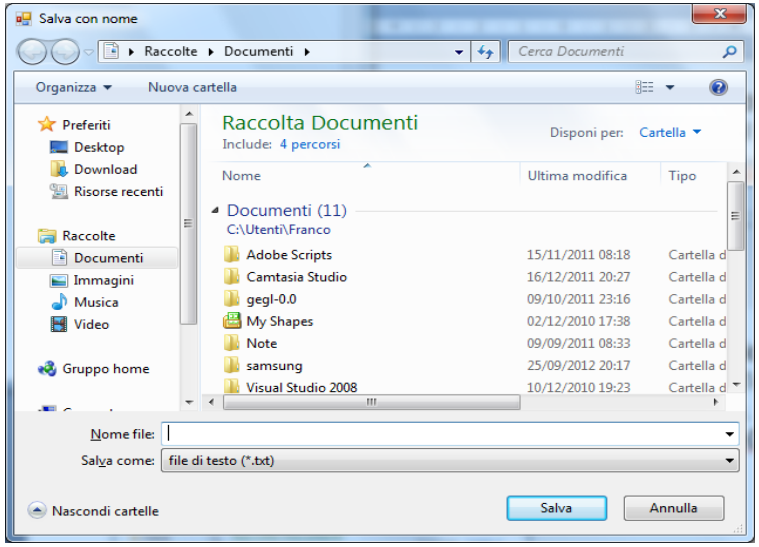


Avremo il così il nostro menu pronto all'uso nel quale andremo ad inserire le voci che vogliamo facendo click nelle voci vuote e scrivendo il testo:



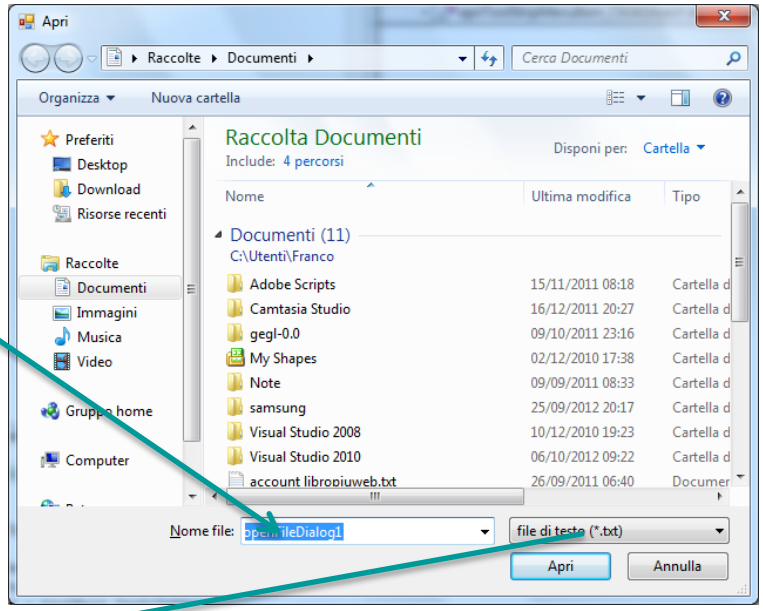
Una volta completato il nostro menu basta fare doppio clic sulle voci per aggiungere automaticamente ad esse un ascoltatore dell'evento clic e generare automaticamente la funzione associata all'evento stesso. Con le voci Apri e Salva utilizzeremo i Dialog incorporati openFileDialog e saveFileDialog



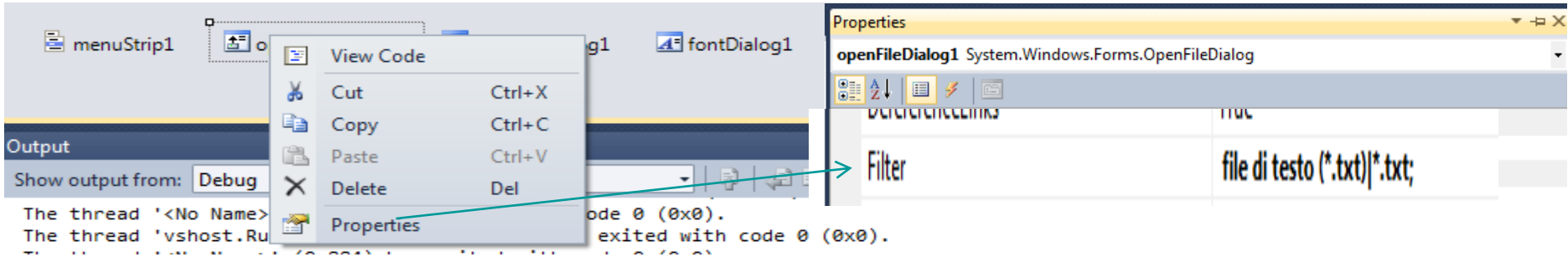


Gestione evento clic su apri:

```
private void apriToolStripMenuItem_Click(object sender, EventArgs e)
        {
            StreamReader sr;
            string letta;
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                sr = new StreamReader(openFileDialog1.FileName);


                while (!sr.EndOfStream)
                {
                    letta = sr.ReadLine();
                    letta = letta + "\r\n";
                    textBox1.Text = textBox1.Text+letta;
                }


                sr.Close();
            }
        }
```



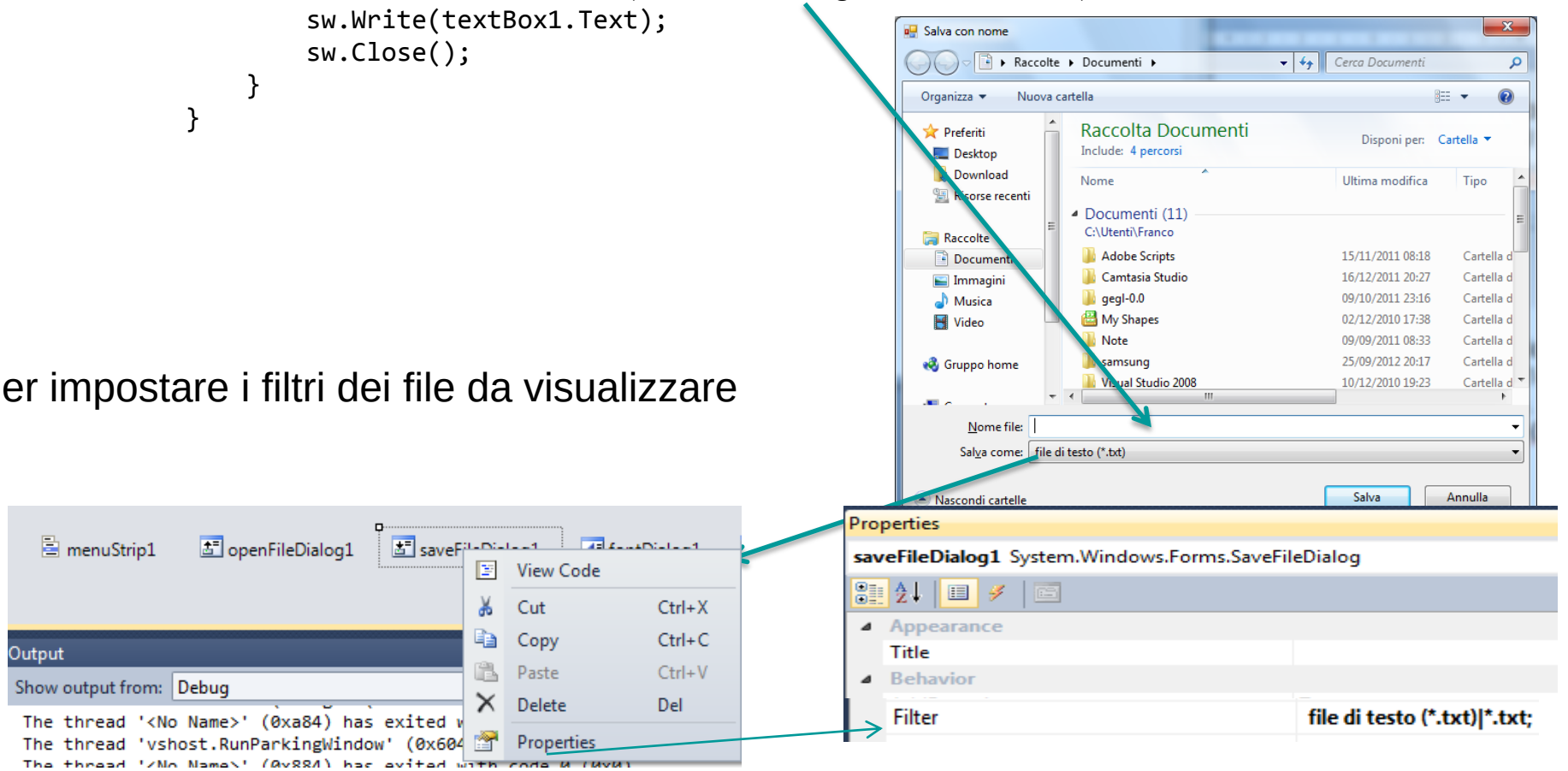
Per impostare i filtri dei file da visualizzare



Gestione evento clic su salva:

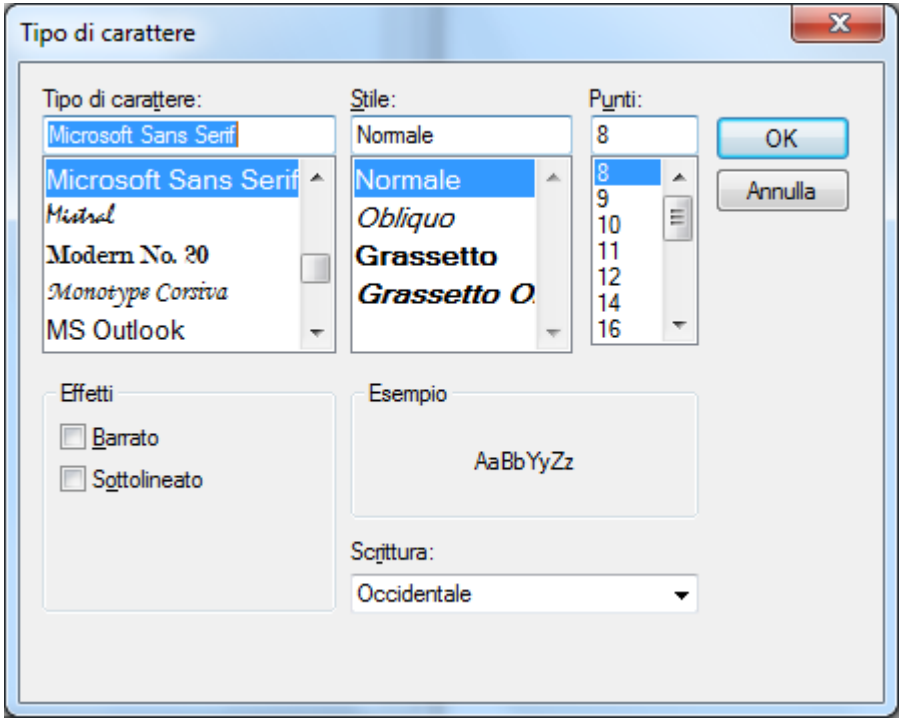
```
private void salvaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            StreamWriter sw;
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                sw = new StreamWriter(saveFileDialog1.FileName,true);
                sw.Write(textBox1.Text);
                sw.Close();
            }
        }
```

Per impostare i filtri dei file da visualizzare



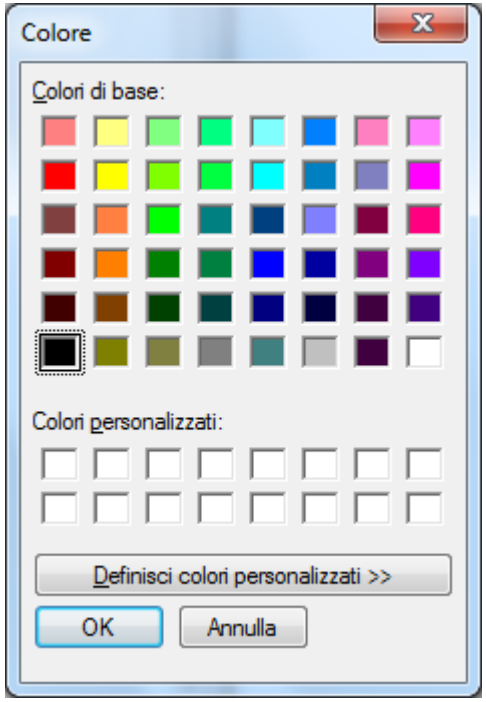
Gestione evento clic su carattere: uso di FontDialog

```
private void carattereToolStripMenuItem_Click(object sender, EventArgs e)
    {

            if (fontDialog1.ShowDialog() == DialogResult.OK)
      {
         textBox1.Font = fontDialog1.Font;
      }

    }
```



Gestione evento clic su carattere: uso di ColorDialog

```
private void coloreToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (colorDialog1.ShowDialog() == DialogResult.OK)
            {
                textBox1.ForeColor = colorDialog1.Color;
            }
        }
```



Gestione evento clic su copia taglia incolla:

```
private void copiaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textBox1.Copy();
        }

        private void incollaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textBox1.Paste();
        }

        private void tagliaToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textBox1.Cut();
        }
```

Stimolo: aggiungere all'applicazione la funzionalità di richiedere all'utente quando si esce dall'applicazione se vuole salvare il file se esso non è stato salvato o se è stato modificato.