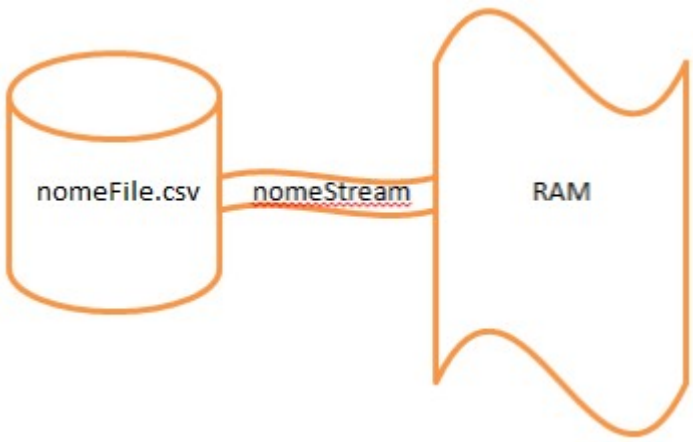
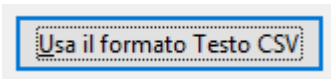
Accesso a file di testo: uso di stream o flusso di caratteri - Esempi

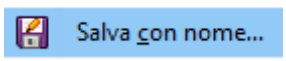


Lettura con gestione delle eccezioni

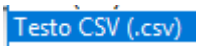
da tabella

	minuti	stanza
2	0	5
3	0	12
4	1	3
5	2	1
6	2	4
7	3	2
8	4	6

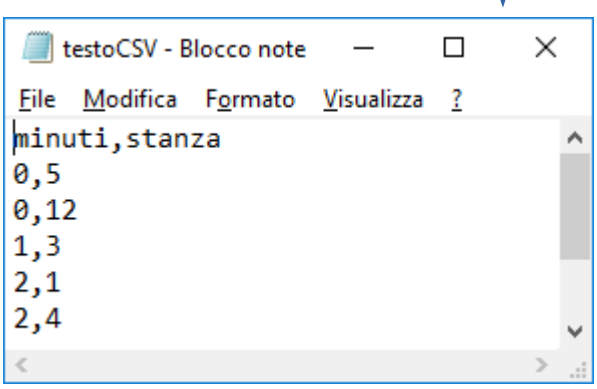


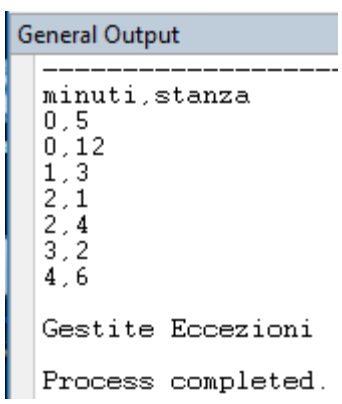


salvata come



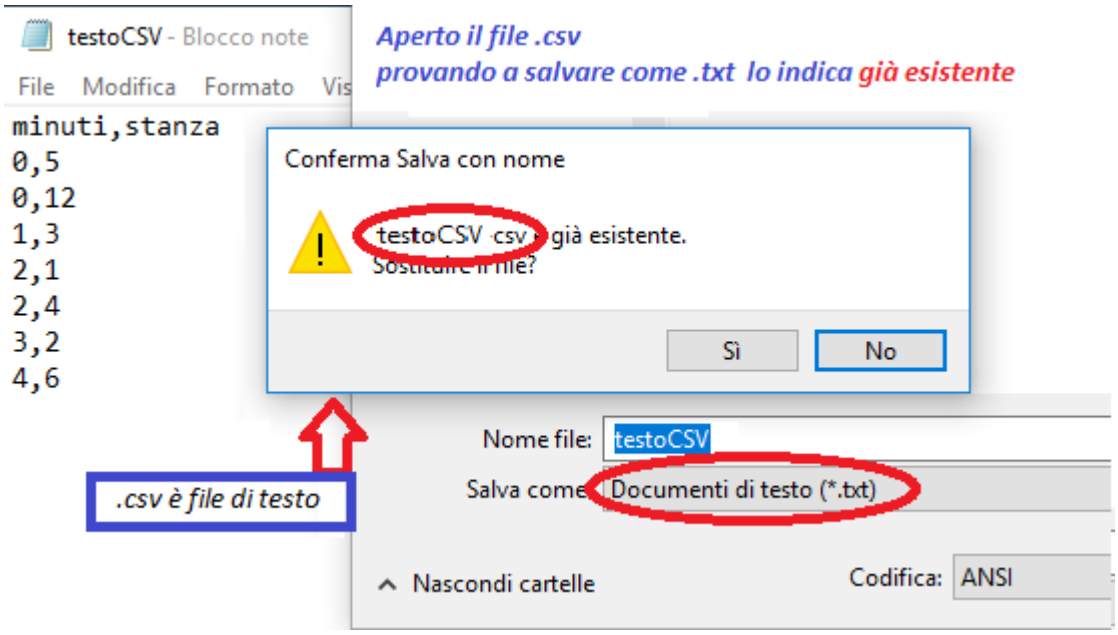

```
import java.io.*;
import java.util.Scanner;

public class LeggiFile {

        public LeggiFile(){
                try{
                        FileReader fr = new FileReader("testoCSV.csv");  // Comma-Separated Values
                        Scanner in = new Scanner(fr);
                        while (in.hasNextLine()) { // se l'input termina hasNextLine() ritorna falso
                            String line = in.nextLine(); // lettura di una riga alla volta
                            System.out.println(line);
                        }
                        fr.close();
                }
                catch (FileNotFoundException ne){ System.err.println (ne); }
                catch (IOException ioe){ System.out.println("Eccezione di I/O"); }
                finally {
                        // codice da eseguire prima dell'uscita dal try-catch-finally
                        System.out.println("\nGestite Eccezioni");
                }
        }// fine costruttore

        public static void main(String[] args){
                new LeggiFile();
        }// fine main
}// fine class
```





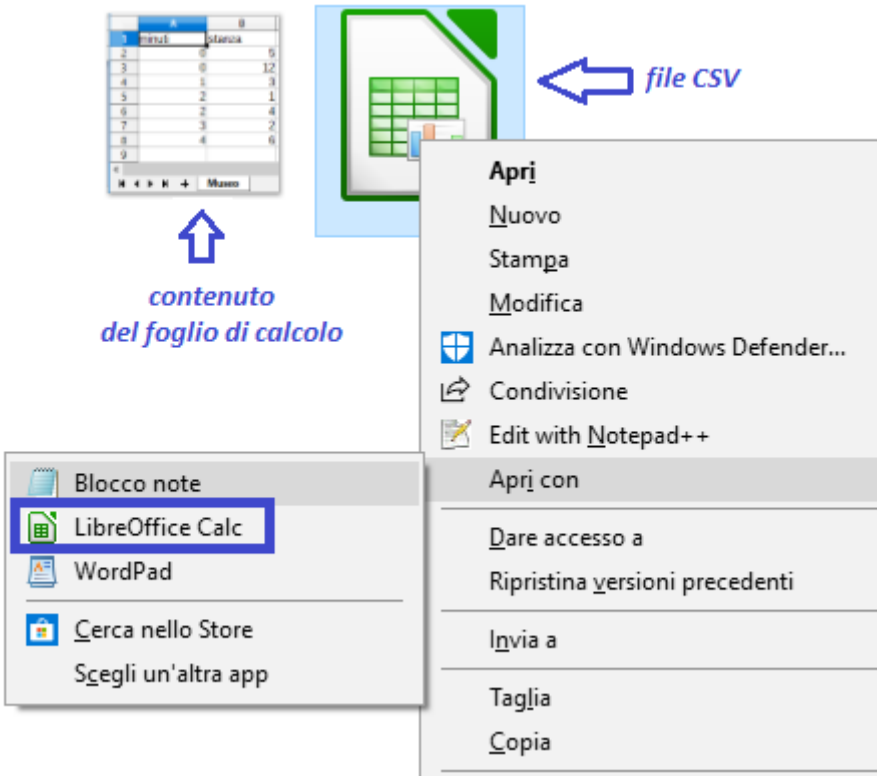
nb: *per manipolare documenti Microsoft (ad esempio creare un file Excel .xlsx) in Java, scaricabili API Apache POI*

Verificare che un file **.csv**
è un file di testo:



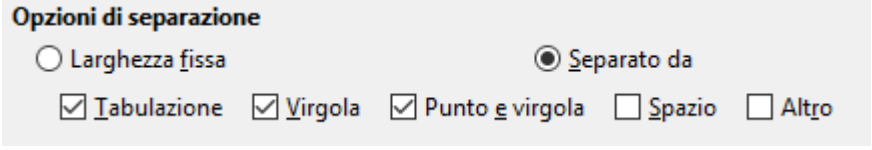
con **particolare formattazione**

Verificare che un file **.csv**
può essere importato in un foglio di calcolo



scegliendo di **aprilo**
con l'applicativo **Calc**

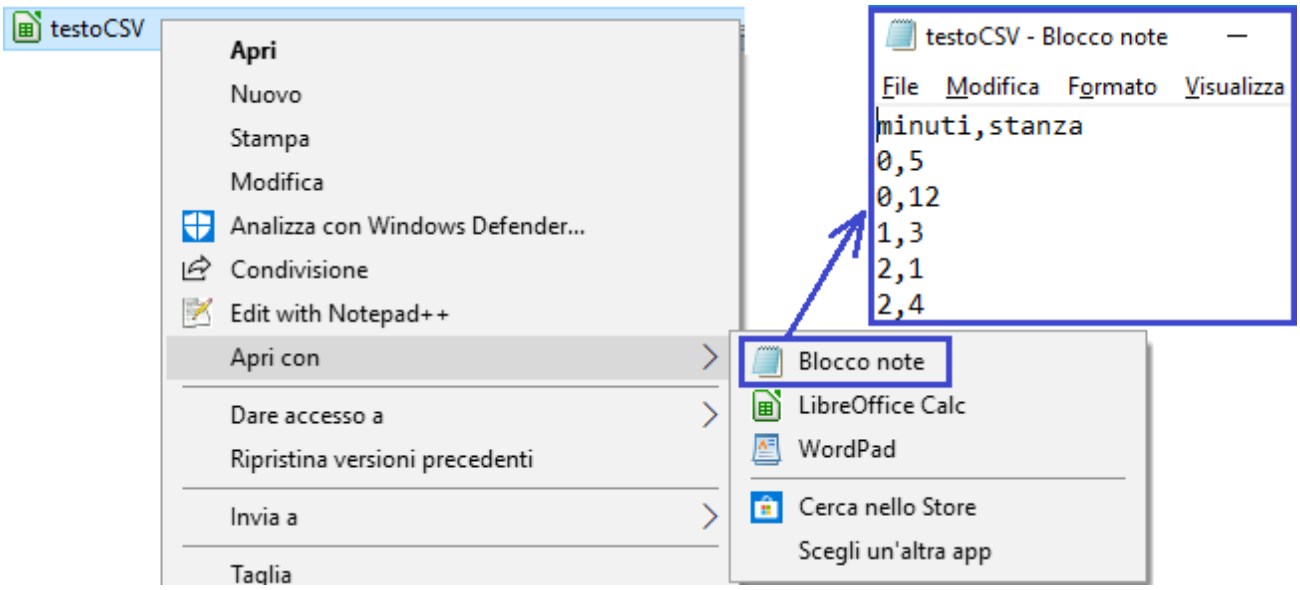
ed impostando le
opzioni di separazione



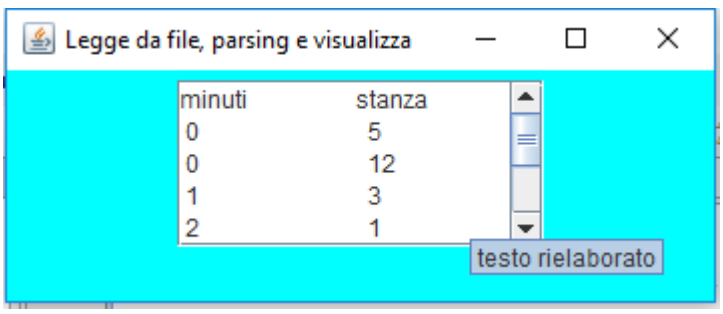
Lettura con parsing

*In informatica, il **parsing**, analisi sintattica o parsificazione è un processo che analizza un flusso continuo di dati in ingresso (input, letti per esempio da un file o una tastiera) in modo da determinare la sua struttura grazie ad una data grammatica formale.*

Input



Output desiderato



Specifiche:

- si gestiscano eventi di Mouse
- si memorizzi ogni linea letta da file in *collezione* di String usando **ArrayList**<String>
- si realizzi una stringa accodando gli elementi della *collezione* suddivisi con delimitatori **","**
- si distingua tra legenda e valori interi per formattare la stringa da visualizzare in un'area di testo ricordando le potenzialità di oggetti della classe Scanner[1]:

  Scanner interpreta come **delimitatori tra tokens** gli *spazi*, il *tabulatore* e l'*'a capo'*

  Volendo un delimitatore diverso da ***whitespace,*** rende disponibile il metodo seguente:

  ***oggettoScanner*.useDelimiter("\\s*");**

  soluzione

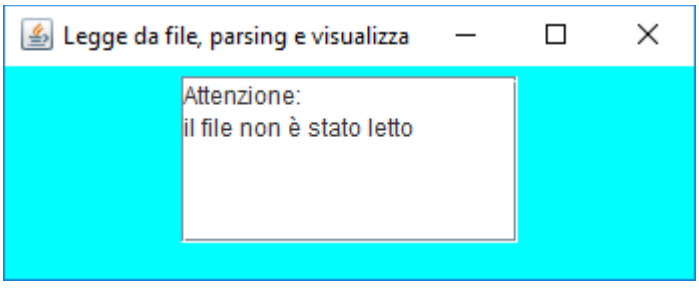
```
Scanner sc = new Scanner(s).useDelimiter("\\s*,\\s*");   // oggetto Scanner
                                                         // per leggere una stringa
                                                         // con delimitatori  ","
```

1 Si ricordino le dispense online introduttive al linguaggio (pg. 38).

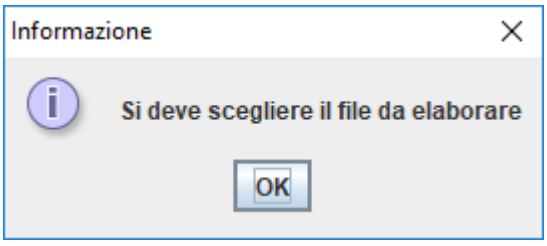
Possibile soluzione proposta da studente

```
import java.io.*; // per accesso a file
import java.util.*;   // per Scanner
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Parsing extends JFrame {

   private Container c;
   private JPanel p;
   private JTextArea tB;       // visualizza stringhe rielaborate
   private JScrollPane tabB;
   private String path = "testoCSV.csv";
   private ArrayList<String> val  = new ArrayList<String>();   // uso collezione di String

   public void ini(){

      c= getContentPane();
      setSize(360,145); // misure in pixel: larghezza, altezza
      setLocation(200,00); // (0,0) angolo sup. sin
      setTitle("Legge da file, parsing e visualizza");
      p = new JPanel();
      p.setBackground(Color.green); // sfondo del pannello colorato
      tB = new JTextArea (5,15);  // righe, colonne              // area di testo per visualizzare stringhe rielaborate
      tB.addMouseListener(new Adatta());// notifica evento all'ascoltatore
      tB.setEditable(false);
      tabB = new JScrollPane(tB);
      tB.setToolTipText("testo rielaborato");
      p.add(tabB);
      c.add(p); // aggiunge il pannello
      setVisible(true); // mostra il frame
      setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
   }
   public void leggiFile(){  // legge da file e memorizza nella collezione di String

            String linea ="";
            try{
                     FileReader fr = new FileReader(path);
                     Scanner in = new Scanner(fr);
                     while (in.hasNextLine()) { // se non ci sono più linee l'input ritorna falso
                              linea = in.nextLine();
                              val.add(linea);              // popola collezione: ogni linea un elemento
                      }
            fr.close();
            }
            catch (FileNotFoundException fe){ System.out.println("Non trovato il file: " + path);   }
            catch (IOException e){  System.err.println(e);   }
            catch (NoSuchElementException ne){  System.err.println(ne);  }
            }// fine metodo leggiFile

            public String elabora(  ){  // elabora collezione di String
             String s ="";
             for(int i = 0; i<val.size(); i++){   // si costruisce una stringa inserendo delimitatori ","

                     if (i == val.size()-1)
                          s = s + val.get(i);                          // ad eccezione dell'ultimo
                     else
                           s = s + val.get(i) + ",";                   // inserendo separatore
             }
```

```
                                                    // si prepara la stringa da visualizzare in tB
        String sB="";
        int token = -1;

        Scanner sc = new Scanner(s).useDelimiter("\\s*,\\s*");  // oggetto Scanner per leggere una stringa
                                                                // con delimitatori  ","
        String min = sc.next();
        String st = sc.next();
        sB = min + "\t" + st + "\n";  // legenda
        boolean flag = false;
        while (sc.hasNext()) {
                if (sc.hasNextInt()) {
                        token = sc.nextInt();
                        if(!flag){
                           sB = sB + " " + token + "\t ";
                           flag = true;
                        }
                        else{
                            sB =  sB + " " + token + "\n";  // a capo ogni coppia
                            flag = false;
                        }
                  }
        }
        return sB;
        }// fine metodo elabora

        private class Adatta extends MouseAdapter{

            public void mouseClicked(MouseEvent me){  // rielabora collezione di String e visualizza in tB

                        String sB = "Attenzione:\nil file non è stato letto";  // per controllo: se non trova il file
                        p.setBackground(Color.cyan); // per test ... con un click
                        leggiFile();
                        try{
                                sB = elabora();  // elabora e riscrive ogni volta in area di testo
                           }
                            catch(NoSuchElementException e) { }  // se non trova il file
                            tB.setText(sB);
                }

        } // fine classe Adatta

    public static void main(String[] args) {
            Parsing gui= new Parsing();
            gui.ini();
    }
} // fine GUI
```



Eventualmente si può prevedere di scegliere subito il file da elaborare ed usare un oggetto di tipo JFileChooser per cercarlo



Eventualmente si può visualizzare – in seconda area di testo - anche il contenuto del file prima della rielaborazione