Prova Finale – Tecniche Avanzate di Programmazione

COGNOME E NOME: _____ *MATRICOLA:* _____

Tempo a disposizione: 5 ore

Visitor

Si consideri una struttura di dati generica i cui elementi sono di tipo diverso e implementano tutti la stessa interfaccia (esempio: la struttura ad albero corrispondente ai controlli contenuti in un Frame, in cui ciascun controllo contiene altri controlli, tutti gli elementi estendono Windows.Forms.Control, ma possono esserci controlli diversi, come bottoni, combo box, label, liste ecc.).

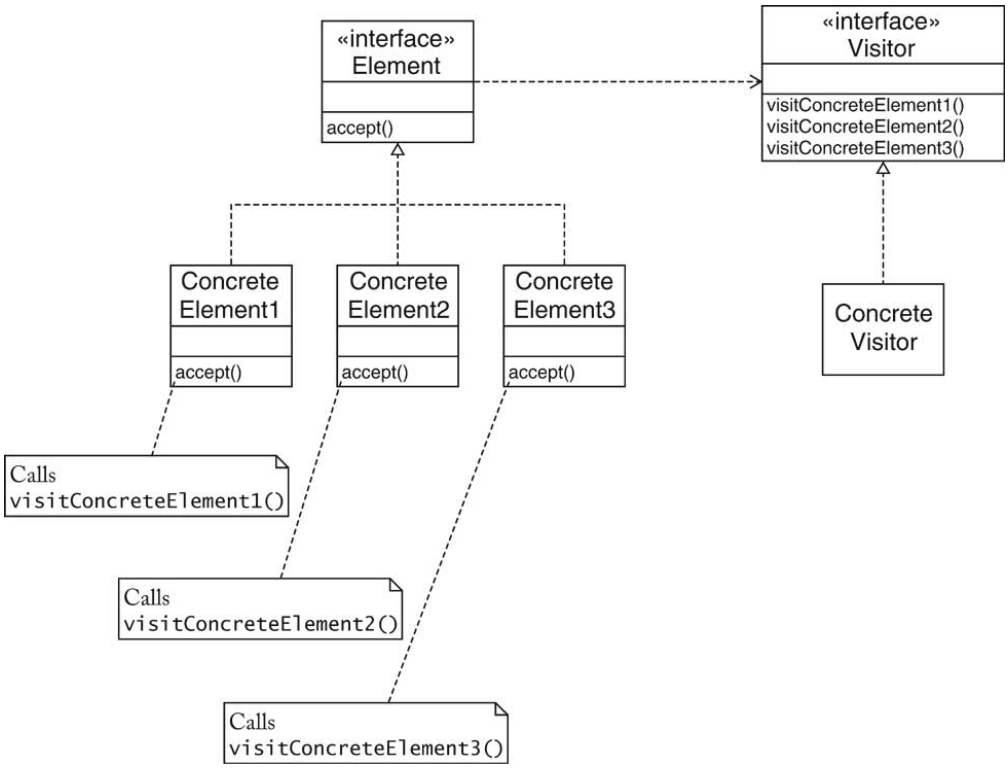
Si supponga che sia necessario implementare su questa struttura operazioni diverse (es: operazioni di disegno, operazioni di ridimensionamento, operazioni di acquisizione e perdita del fuoco). Una soluzione sarebbe definire tutte le operazioni come metodi di Control, e sovrascriverle nelle sottoclassi; questa soluzione, però, ha lo svantaggio che l'aggiunta di una nuova operazione richiede di modificare l'interfaccia di Control aggiungendo un nuovo metodo e di implementarlo in tutte le sottoclassi, e quindi di ricompilare l'intera gerarchia. Viceversa, l'obiettivo è quello di consentire di aggiungere nuove operazioni sulla gerarchia cambiando il meno possibile la gerarchia di componenti originali.

Il design pattern **Visitor** è una soluzione a questo problema. Il pattern prevede di implementare le operazioni sulla gerarchia in oggetti esterni detti per l'appunto visitor. Tutti i visitor implementano la stessa interfaccia *Visitor*, composta da un metodo per ogni tipologia di oggetto presente nella struttura di dati (es: i visitor per la gerarchia di controlli Windows.Forms avranno un metodo VisitButton(), un metodo VisitCheckBox(), un metodo VisitComboBox() ecc.).

Per eseguire le operazioni, ciascuna classe della gerarchia deve implementare un unico metodo *void Accept(Visitor v)*, attraverso il quale un oggetto "accetta" di essere visitato da un visitor. Ciascuna sottoclasse fornirà una implementazione diversa del metodo, specificando quale dei metodi del visitor applicare all'oggetto (es: in Button, il metodo sarà *void Accept(Visitor v) {v.VisitButton(this);}* in ComboBox, il metodo sarà *void Accept(Visitor v) {v.VisitComboBox(this);}* ecc.).

A questo punto, per aggiungere nuove operazioni sulla gerarchia sarà sufficiente aggiungere nuove implementazioni dell'interfaccia Visitor (es: PaintVisitor, ResizeVisitor, FocusVisitor ecc.) e utilizzare gli oggetti in questione per visitare la struttura di dati.

Il diagramma UML riportato di seguito riassume la struttura di classi del pattern:



Utilizzando il pattern Visitor, scrivere il modello di un'applicazione in linguaggio **C#** che effettua operazioni sul file system.

In particolare, dato il percorso di una cartella del file system, l'applicazione deve essere in grado di effettuare le seguenti operazioni:

- stampare la struttura delle sottocartelle e dei file contenuti, secondo il seguente formato indentato
  ```
  cartella: tmp
      prova.html
      dati.txt
      cartella: icons
          bottone.jpg
          cartella: varie
              prova.jpg
  ```

- calcolare il numero di sottocartelle, di file di tipo html (con estensione .html), file di tipo .jpg e file di altro tipo contenuti nella cartella in questione

Sviluppare inoltre un piano completo di test di regressione per il modello.

Suggerimenti:

- *per gli scopi della prova, considerare le porzioni del file system analizzate come alberi di oggetti di 4 tipi: (a) cartelle (b) file html (c) file jpg (d) file di altro tipo*
- *le classi delle API di .NET da utilizzare per ottenere informazioni su file e cartelle sono System.IO.FileInfo e System.IO.DirectoryInfo; è necessario però decorare gli oggetti in questione in modo da fornire le funzionalità richieste per l'utilizzo dei visitor*
- *per ciascuna delle due operazioni richieste è necessario creare una classe che implementa l'interfaccia Visitor (es: PrintVisitor e CountVisitor); per eseguire ciascuno dei due visitor è sufficiente chiamare sull'oggetto che rappresenta la cartella radice del sottoalbero da analizzare il metodo Accept(); es: detto radice il riferimento alla cartella radice della porzione del file system considerata, per ottenere una stampa utilizzando il PrintVisitor è sufficiente eseguire la chiamata radice.Accept(new PrintVisitor())*