

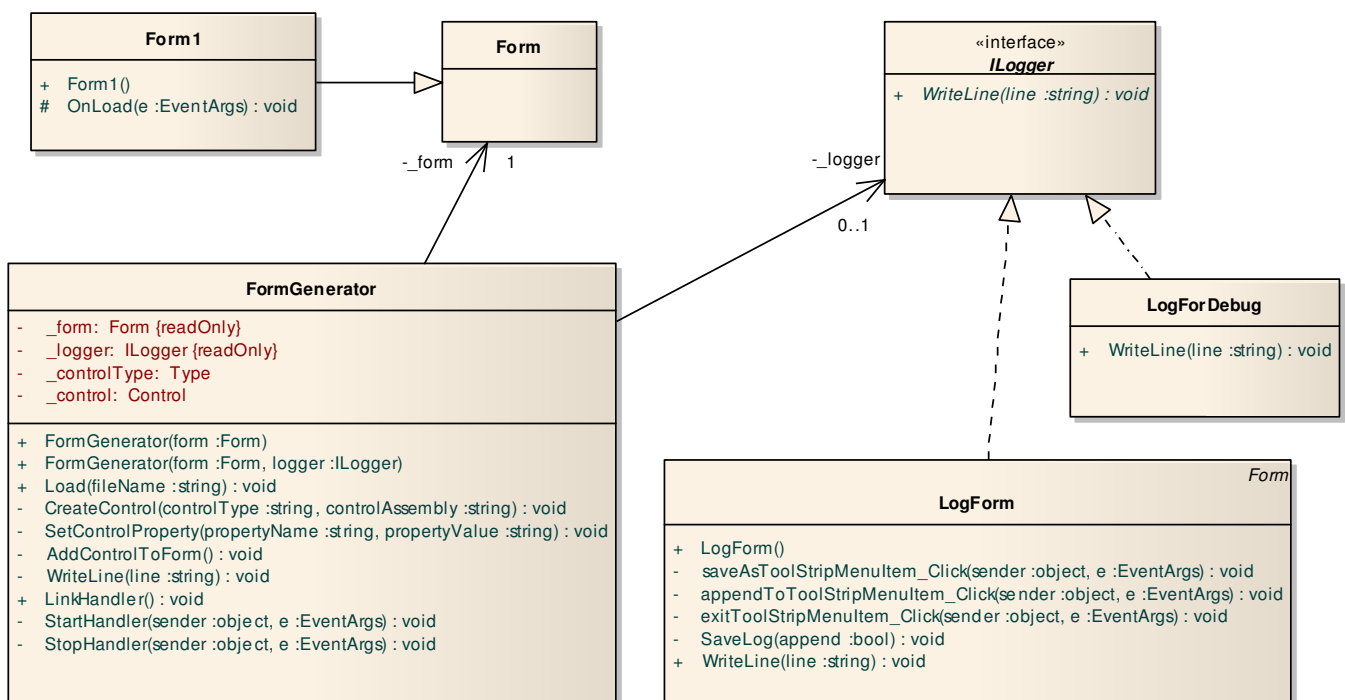
Prova 2

AVVERTENZE PER LA CONSEGNA

1. All'inizio di ogni file sorgente inserire in un commento i propri dati: cognome, nome e numero di matricola.
2. Al termine della prova, **consegnare** (inviare) **l'intero direttorio** contenente la solution **dopo averlo compresso**.

Passo 0 – Creare un progetto di tipo **WindowsFormsApplication** di nome **Prova2** e fare il *build* in modo che venga creato l'eseguibile. Scaricare il file **Prova2Start.zip** contenente: i file **Controls.xml** e **LogFile.txt**, da inserire nel direttorio che contiene i sorgenti di **Prova2**, e i file **GridComponent.dll** e **Prova2Controls.dll**, da inserire nel direttorio che contiene l'eseguibile di **Prova2**. Aggiungere al progetto **Prova2** la classe **FormGenerator** e la *windows form* **LogForm** (come da schema UML). Responsabilità della classe **FormGenerator** sono:

1. aggiungere dinamicamente ad una generica *form* (nel nostro caso un'istanza di **Form1**) una serie di controlli le cui caratteristiche sono memorizzate in un file XML (nel nostro caso, il file **Controls.xml**) – ved. passo 1;
2. elencare su una seconda *form* di tipo **LogForm** sia le operazioni effettuate con successo, sia quelle fallite – ved. passo 2;
3. collegare l'evento Click di alcuni dei controlli inseriti a gestori di eventi di altri controlli – ved. passo 3.



Passo 1 – Inserire nella classe **FormGenerator** tutti i *field* e i metodi (vuoti) elencati nello schema UML. In **Form1**, ridefinire il metodo **OnLoad** in modo da creare una nuova istanza di **FormGenerator** e quindi invocare opportunamente i metodi **Load** e **LinkHandler** di tale istanza. Il metodo **Load** deve scandire il contenuto del file XML (mediante **XmlReader** o **XmlDocument**) e invocare in modo opportuno i metodi **CreateControl**, **SetControlProperty** e **AddControlToForm** – **Attenzione**: il file XML può contenere dei commenti che devono essere gestiti correttamente dal metodo **Load**. Il metodo **CreateControl** accetta come argomenti due stringhe contenenti il nome del tipo di controllo da creare e il nome dell'*assembly*

Prova 2

che contiene la definizione del controllo. Poiché il controllo da creare potrebbe appartenere a un *assembly* non ancora caricato in memoria, come prima operazione il metodo **CreateControl** deve invocare **Assembly.Load**. Successivamente, **CreateControl** deve ottenere il tipo di controllo mediante **Type.GetType** a cui è necessario passare il nome completo della classe `<nomeTipoControllo,nomeAssembly>`. Infine, deve creare il controllo. **Attenzione:** il metodo **CreateControl** deve inizializzare correttamente i due field `_controlType` e `_control` e non deve mai generare eccezioni; se per una qualsiasi ragione non riesce a creare il controllo, deve assegnare a `_controlType` e `_control` il valore **null**. Il metodo **SetControlProperty** deve assegnare un valore a una proprietà del controllo e non deve mai generare eccezioni. Infine, il metodo **AddControlToForm** deve aggiungere il controllo alla *form*.

Al fine di elencare tutte le operazioni effettuate con successo o con fallimento, nei metodi **CreateControl**, **SetControlProperty** e **AddControlToForm** invocare opportunamente il metodo **WriteLine** della classe **FormGenerator**. Tale metodo, a sua volta, deve invocare il metodo **WriteLine** del `_logger`, se presente. Per testare l'applicazione, utilizzare un'istanza di **LogForDebug** che deve visualizzare le informazioni di *log* su *console* (in *debug*, sulla finestra di *output*).

Passo 2 – Aggiungere alla **LogForm** un menù “**File**” con le tre voci “**Save As...**”, “**Append To...**” e “**Exit**” e una **TextBox** multi linea e con *scroll bar* che riempia tutta la *form*. Il metodo **WriteLine** della **LogForm** deve visualizzare le informazioni di *log* nella **TextBox**. L'informazione visibile nella *form* dopo il caricamento del file **Controls.xml** deve essere identica a quella riportata nel file **LogFile.txt**. Il menù “**Save As...**” deve permettere il salvataggio su file testo delle informazioni di *log* contenute correntemente nella *form*, il menù “**Append To...**” deve permettere di effettuare la stessa operazione, andando però in *append* su file. Poiché le due operazioni sono quasi identiche, utilizzare il metodo **SaveLog** per non ripetere inutilmente il codice.

Passo 3 – Il metodo **LinkHandler** della classe **FormGenerator** deve scandire tutti i controlli contenuti nella *form* su cui si è operato, alla ricerca di controlli di tipo **Button** con la proprietà **Text** uguale a “**Start**” o “**Stop**”. All'evento **Click** dei bottoni “**Start**” deve associare il gestore **StartHandler**, mentre all'evento **Click** dei bottoni “**Stop**” deve associare il gestore **StopHandler**.

Il metodo **StartHandler** deve scandire tutti i controlli contenuti nella *form* e per ogni controllo che implementa il metodo (pubblico) “**void Start()**”, deve invocare tale metodo utilizzando il metodo **Invoke** della classe **MethodInfo**. In modo del tutto analogo, il metodo **StopHandler** deve scandire tutti i controlli contenuti nella *form* e per ogni controllo che implementa il metodo (pubblico) “**void Stop()**”, deve invocare tale metodo utilizzando il metodo **Invoke** della classe **MethodInfo**. Si noti che, anche in questo caso, esiste del codice ripetuto che potrebbe essere eliminato...