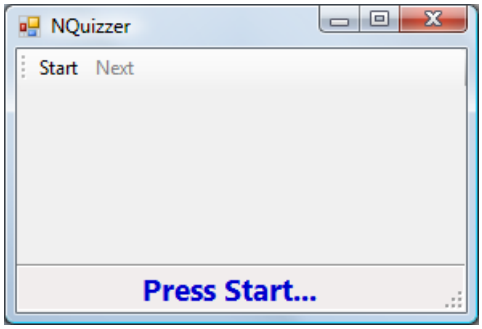
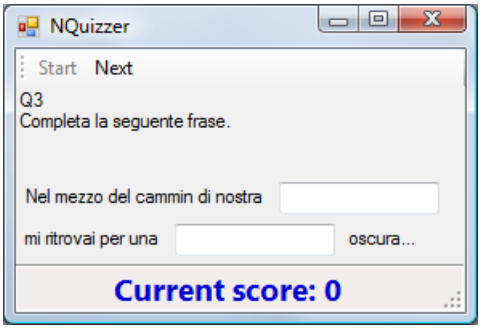
Prova d'esame del 22 giugno 2010

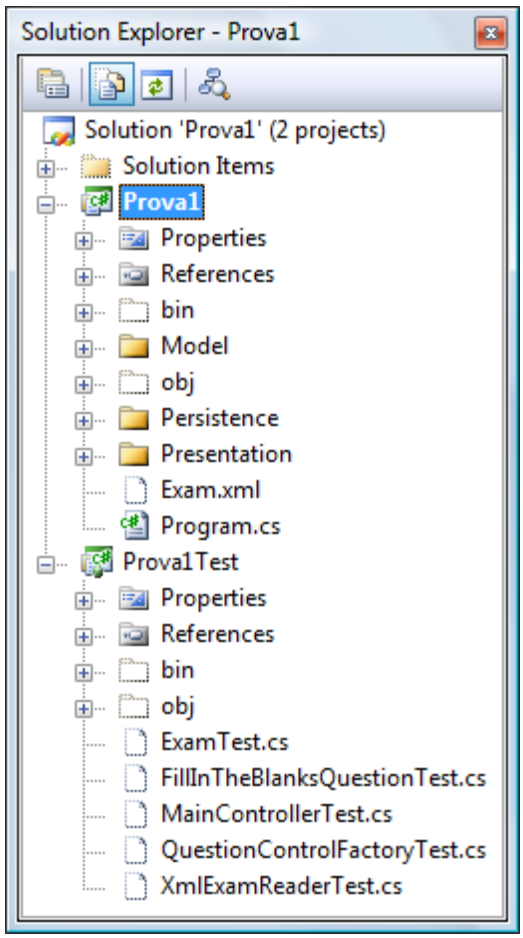
Premessa





Si vuole realizzare un'applicazione per la gestione di un esame a quiz. L'applicazione deve leggere le domande e le relative risposte da un file xml e quindi deve sottoporle all'esaminando in stretta sequenza. Le domande possono essere di due tipi: scelta multipla e *fill in the blanks*. Dopo la conferma di ogni singola risposta, l'applicazione deve calcolare e visualizzare il punteggio totale corrente. Le due figure mostrano come si deve presentare l'interfaccia utente inizialmente e durante l'esecuzione dell'esame a quiz (in particolare, la seconda figura mostra un esempio di domanda del tipo *fill in the blanks*).
Nota: in tutti i diagrammi UML riportati nel seguito, i metodi nelle sezioni marcate con «property» sono proprietà C# e devono contenere sempre la **`get`** e, solo quando necessario, la **`set`**.

Passo 0 – Start kit



Collegarsi al sito "**`http://esamix.labx`**", autenticarsi con la propria matricola e scaricare il file "Prova1Start.zip". Lo *start kit* contiene un'intera *solution* per Visual Studio 2008 (**attenzione**: **non può funzionare su VS 2005!**) con due progetti (ved. figura).

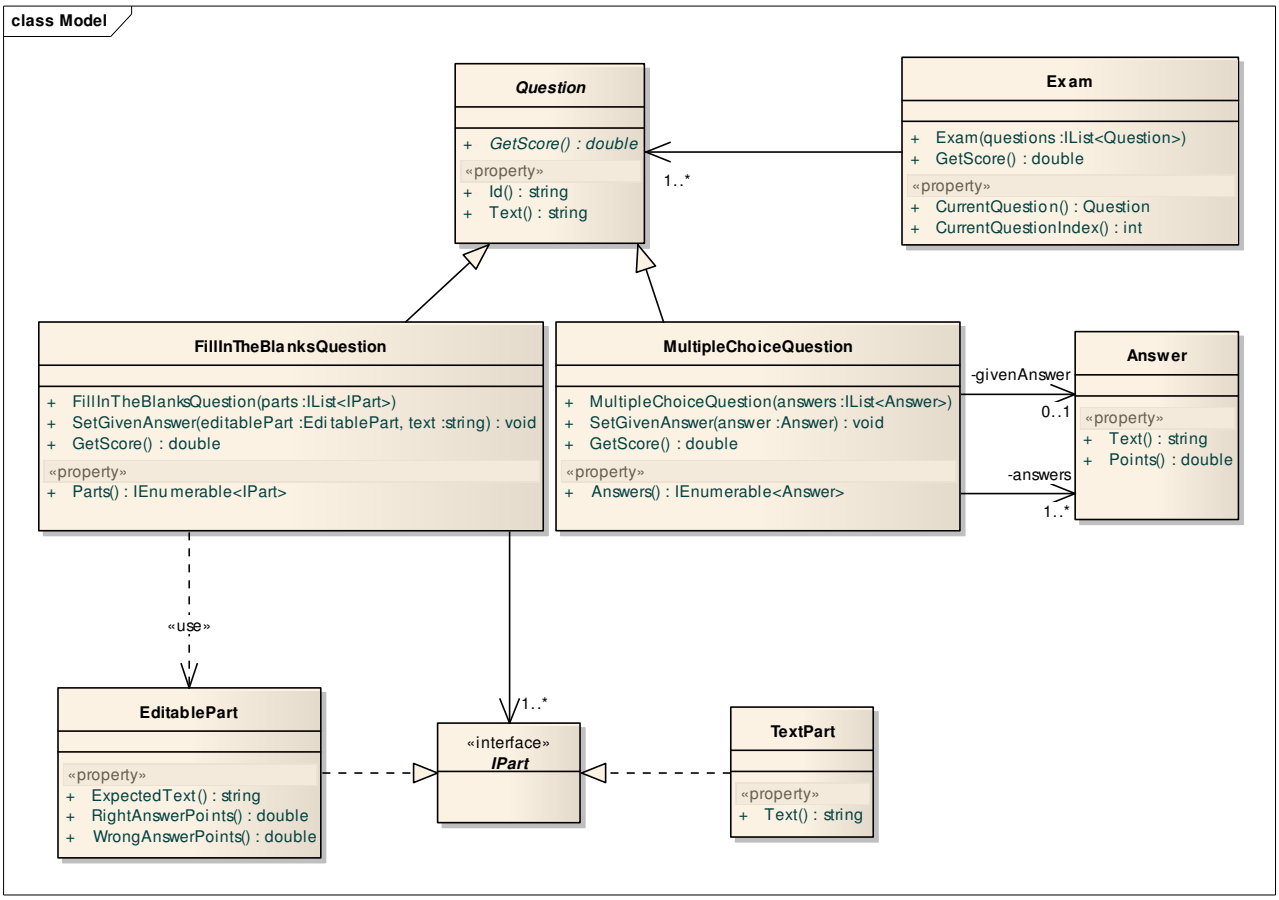
Il **progetto Prova1** dovrà essere completato dal candidato e contiene tre cartelle (che corrispondono ad altrettanti *namespace*): **`Model`**, che deve contenere tutte le classi riguardanti il modello, **`Persistence`**, che deve contenere tutte le classi riguardanti la persistenza, e **`Presentation`**, che deve contenere tutte le classi riguardanti la *user interface*. "Exam.xml" è il file xml che descrive l'esame a quiz e da cui è desumibile il formato di tutti gli elementi che compongono l'esame.

Il **progetto Prova1Test** comprende una serie di file di test che inizialmente non sono inclusi nel progetto e che in seguito (come indicato nel testo del compito) dovranno essere inclusi nel progetto e fatti girare. Per eseguire i test: dal menù principale di VS 2008, "Test" → "Run" → "All Tests in Solution". L'esecuzione dei test non è obbligatoria, ma se effettuata permette di verificare la correttezza di buona parte del codice scritto. I test saranno comunque utilizzati in fase di correzione dell'esame.
Per una corretta esecuzione dei test, se nel testo del compito vengono specificate delle **precondizioni** riguardanti gli argomenti passati a un metodo, è necessario utilizzare in modo opportuno **`ArgumentNullException`** e **`ArgumentException`**.

Al termine dell'esame, **consegnare esclusivamente i singoli file sorgenti contenenti le classi realizzate o completate**; in particolare:
Exam.cs, FillInTheBlanksQuestion.cs, XmlExamReader.cs, QuestionControlFactory.cs, MainController.cs, MainForm.cs + MainForm.Designer.cs,
MultipleChoiceControl.cs + MultipleChoiceControl.Designer.cs.
ATTENZIONE: **tutti i file sorgenti consegnati** (ad esclusione dei .Designer) **devono contenere** in un commento iniziale i dati del candidato: **cognome, nome e numero di matricola**.

Passo 1 – Modello

Il seguente diagramma UML mostra tutte le classi riguardanti il modello.



In particolare, la classe astratta **`Question`** è specializzata dalle classi **`MultipleChoiceQuestion`**, che descrive una domanda a scelta multipla, e **`FillInTheBlanksQuestion`**, che descrive una domanda di tipo *fill in the blanks*. Una domanda di tipo *fill in the blanks* è costituita da più parti (**`IPart`**), di tipo diverso: alcune contengono un testo fisso (**`TextPart`**), altre un campo bianco editabile (**`EditablePart`**). Ogni oggetto di tipo **`EditablePart`** contiene le informazioni che permettono di verificare la correttezza della risposta data dall'esaminando e di calcolare il relativo punteggio parziale.

Realizzare la classe `Exam`. La classe descrive un esame mediante una collezione di **`Question`** (passata al costruttore e non vuota) e un indice (**`CurrentQuestionIndex`**) che permette di selezionare la domanda corrente nella collezione e che inizialmente deve valere -1. La proprietà **`CurrentQuestion`** deve restituire la **`Question`** correntemente selezionata da **`CurrentQuestionIndex`** oppure **`null`**. Il metodo **`GetScore`** deve restituire il punteggio totale corrente; a tal fine, si utilizzi in modo opportuno il metodo omonimo di **`Question`**. Una volta realizzata **`Exam`**, è possibile includere nel progetto **`Prova1Test`** la classe **`ExamTest`** e lanciare i test.

Realizzare la classe `FillInTheBlanksQuestion`. La classe descrive una domanda di tipo *fill in the blanks* mediante una collezione di parti (passata al costruttore e non vuota). La proprietà **`Parts`** restituisce la collezione di parti. Il metodo **`SetGivenAnswer`** deve associare (mediante un dizionario) a un'**`EditablePart`** (che deve appartenere alla collezione di parti) il corrispondente testo inserito dall'esaminando. Il metodo **`GetScore`** deve restituire la sommatoria dei punteggi di tutte le **`EditablePart`** che fanno parte della domanda. Il punteggio di un'**`EditablePart`** vale (**`givenText`** è il testo inserito dall'esaminando):

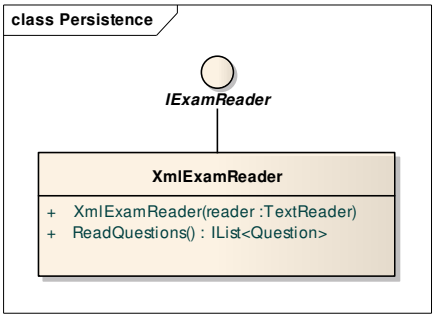
- **`0`**, se **`givenText`** è **`null`** o è una stringa vuota;
- **`RightAnswerPoints`**, se **`givenText`** è uguale all'**`ExpectedText`**;
- **`WrongAnswerPoints`**, se **`givenText`** è diverso dall'**`ExpectedText`**.

Tutti i confronti tra le stringhe devono essere *case-insensitive*.

Una volta realizzata la classe **`FillInTheBlanksQuestion`**, è possibile includere nel progetto **`Prova1Test`** la classe **`FillInTheBlanksQuestionTest`** e lanciare i test.

Passo 2 – Persistenza

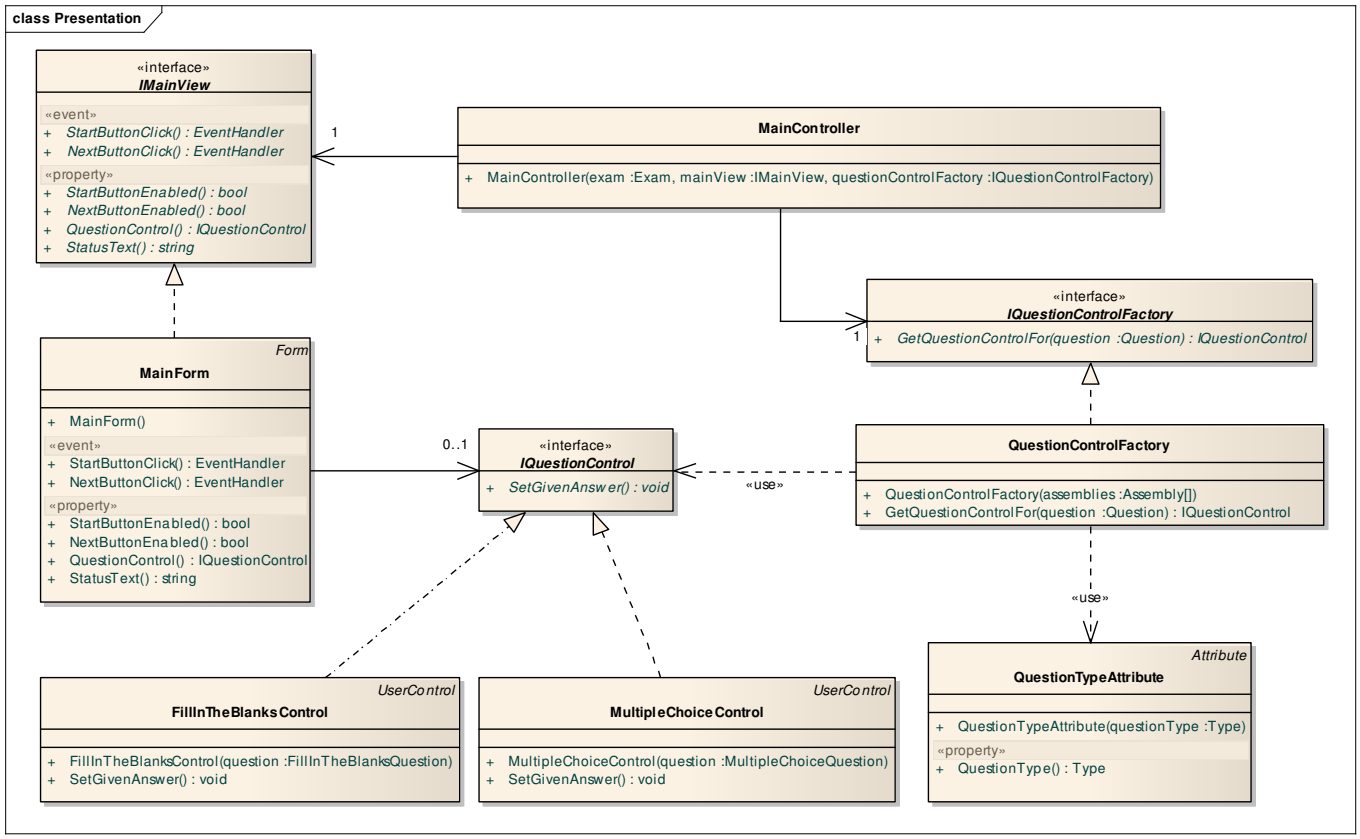
Il seguente diagramma UML mostra le classi che realizzano lo strato di persistenza.



Realizzare la classe `XmlExamReader`. La classe implementa l'interfaccia **`IExamReader`** e deve permettere di leggere le domande da un documento xml. Il costruttore accetta come argomento un **`TextReader`**. Il metodo **`ReadQuestions`** legge il documento xml e genera una collezione di **`Question`**. Poiché la classe deve gestire la lettura di tutti i tipi di domande, è **indispensabile** suddividere la lettura in più metodi: **`ReadMultipleChoiceQuestion`**, **`ReadFillInTheBlanksQuestion`**, **`ReadFillInTheBlanksPart`**, ecc. Una volta realizzata la classe **`XmlExamReader`**, è possibile includere nel progetto **`Prova1Test`** la classe **`XmlExamReaderTest`** e lanciare i test.

Passo 3 – Interfaccia utente

Il seguente diagramma UML mostra le classi che realizzano l'interfaccia utente.



La visualizzazione e l'interazione con l'esaminando dei diversi tipi di domanda sono gestite da *user control* specifici. Tali *user control* sono classi concrete che implementano l'interfaccia **`IQuestionControl`** e che hanno un costruttore che prende come argomento una **`Question`** del tipo che sono in grado di gestire. L'associazione tra tipo di *user control* e tipo di domanda è 1:1 ed è fornita dall'attributo **`QuestionTypeAttribute`**. Includere nel progetto **`Prova1`** le classi **`FillInTheBlanksControl`** e **`MultipleChoiceControl`**, che sono gli *user control* specifici per i due tipi di domande utilizzati nell'applicazione. La classe **`FillInTheBlanksControl`** è un esempio completo di *user control* specifico, mentre la classe **`MultipleChoiceControl`** dovrà essere completata **in seguito**.

Realizzare la classe `QuestionControlFactory`. La classe implementa l'interfaccia **`IQuestionControlFactory`** e deve gestire la creazione di controlli specifici in grado di visualizzare i vari tipi di domande. A tal fine, il costruttore cerca nella collezione di **`Assembly`** passata come argomento (**attenzione**, l'argomento è di tipo

params) tutte le classi non astratte che implementano l'interfaccia **IQuestionControl** e che sono marcate con l'attributo **QuestionTypeAttribute**. Tali classi devono essere memorizzate in un dizionario insieme al tipo specifico di **Question** che sono in grado di visualizzare (fornito dalla proprietà **QuestionType** dell'attributo). Il metodo **GetQuestionControlFor** deve trovare il tipo di controllo che gestisce il tipo di domanda passata come argomento (se il tipo di domanda non è gestibile → **ApplicationException**) e restituire una nuova istanza del controllo, avendo cura di passare come argomento del costruttore la domanda stessa. Una volta realizzata la classe **QuestionControlFactory**, è possibile includere nel progetto **Prova1Test** la classe **QuestionControlFactoryTest** e lanciare i test.

Realizzare la classe MainController (il controller dell'applicazione). Il costruttore di tale classe, prende in ingresso un **Exam**, un **IMainView** e un **IQuestionControlFactory** (tutti gli argomenti non devono essere **null**); il costruttore deve: abilitare il bottone "Start", disabilitare il bottone "Next" e registrarsi agli eventi **StartButtonClick** e **NextButtonClick** (ved. interfaccia **IMainView**). Il gestore di **StartButtonClick** deve: disabilitare il bottone "Start", abilitare il bottone "Next", assegnare zero all'indice di domanda corrente dell'esame e aggiornare l'interfaccia utente, come descritto nel seguito. Il gestore di **NextButtonClick** deve: invocare il metodo **SetGivenAnswer** sul controllo corrente (proprietà **QuestionControl** - in questo modo il controllo aggiorna la domanda corrente con le risposte fornite dall'utente), incrementare di uno l'indice della domanda corrente e aggiornare l'interfaccia utente. Per aggiornare l'interfaccia utente, si deve: assegnare alla proprietà **StatusText** il testo contenente il punteggio corrente dell'esame e, se la **CurrentQuestion** dell'esame non è **null**, assegnare alla proprietà **QuestionControl** il controllo in grado di gestire la domanda, dopo averlo recuperato tramite la *factory*; se **CurrentQuestion** è **null**, occorre disabilitare il bottone "Next" e assegnare **null** a **QuestionControl**. Una volta realizzata la classe **MainController**, è possibile includere nel progetto **Prova1Test** la classe **MainControllerTest** e lanciare i test.

Realizzare la classe MainForm (la *form* principale dell'applicazione). La classe deve implementare l'interfaccia **IMainView** in modo che possa essere controllata dal **MainController**. Mediante *designer*, inserire: un **ToolStrip** (contenente i bottoni "Start" e "Next"), uno **StatusStrip** (contenente una *label*) e un **Panel** (che deve occupare tutto lo spazio rimanente e che, in seguito, conterrà il controllo relativo alla domanda corrente). Implementare l'interfaccia **IMainView** nel seguente modo: gli eventi **StartButtonClick** e **NextButtonClick** possono essere realizzati utilizzando la sintassi **add/remove** in modo da registrare gli *handler* direttamente sull'evento **Click** del bottone corrispondente (non è obbligatorio utilizzare questa tecnica); le proprietà **StartButtonEnabled** e **NextButtonEnabled** devono essere realizzati utilizzando la proprietà **Enabled** del bottone corrispondente; la proprietà **StatusText** deve essere realizzata utilizzando la *label* dello **StatusStrip**; infine, la proprietà **QuestionControl** deve essere realizzata mediante un campo privato e nel metodo **set**, se e solo se il nuovo valore è diverso da quello corrente, occorre inserire il nuovo controllo in modo opportuno nel pannello principale della *form* (impostare il **Dock** del nuovo controllo a **Fill**).
A questo punto, è possibile togliere i commenti nel codice di **Program.Main** ed eseguire l'applicazione. Ovviamente non saranno visibili le domande a scelta multipla, poiché si deve ancora completare il controllo che gestisce le **MultipleChoiceQuestion**.

Completare la classe MultipleChoiceControl. Mediante *designer*, inserire un primo pannello con **Dock** a **Top** (pannello superiore) e un secondo pannello con **Dock** a **Fill** (pannello principale). Nel pannello superiore, inserire due **Label** con **Dock** a **Top** e **AutoSize** a **true**; la label più in alto dovrà contenere il codice (**Id**) della domanda, quella più in basso, il testo (**Text**) della domanda. Il costruttore prende in ingresso una **MultipleChoiceQuestion** e, tramite un metodo **Build**, costruisce la relativa interfaccia utente. Nel metodo **Build**, assegnare alle corrispondenti **Label** l'**Id** e il **Text** della domanda e, per ogni risposta possibile, creare un **RadioButton** utilizzando come testo il testo della risposta, memorizzare nel **Tag** del **RadioButton** la risposta stessa e aggiungere il **RadioButton** al pannello centrale; infine aggiungere un ultimo **RadioButton** utilizzando come testo "Nessuna risposta data" e con **Checked** a **true**. Tutti i **RadioButton** devono avere la proprietà **AutoSize** a **true**. Per implementare l'interfaccia **IQuestionControl** occorre definire il metodo **SetGivenAnswer**. In questo caso, il metodo del controllo deve invocare il metodo **SetGivenAnswer** della domanda, passando come argomento la risposta selezionata dall'esaminando (memorizzata nel **Tag** del **RadioButton** correntemente selezionato).