1. *Descrivere le differenze tra Reference Type e Value Type*

Le variabili *Reference Type* archiviano i riferimenti ai relativi dati (oggetti), mentre le variabili dei tipi *Value Type* contengono direttamente i dati. Con i *Reference Type*, due variabili possono fare riferimento allo stesso oggetto. Di conseguenza le operazioni su una variabile possono influire sull'oggetto a cui fa riferimento l'altra variabile. Con i *Value Type*, ogni variabile ha una propria copia dei dati e non è possibile che le operazioni su una variabile influiscano sull'altra.

1. *Cosa è un delegate?*

I delegate sono l’equivalente .NET dei puntatori a funzioni del C/C++ unmanaged, ma hanno il grosso vantaggio di essere tipizzati.

A differenza dei puntatori a funzione, i delegati sono orientati a oggetti, indipendenti dai tipi e sicuri. Il tipo delegato è definito dal nome del delegato.

La sua dichiarazione in C# si fa con la parola chiave DELEGATE.

Es. delegate void MyDelegate(int i);

Il compilatore crea di conseguenza una classe che deriva da System.Delegate oppure System.MulticastDelegate con il nome MyDelegate.

Le due classi sono della classi speciali e possono essere derivate solo dal compilatore.

Per specificare un nuovo metodo con Delegate si usa la parola chiave new.

Per istanziare il delegate gli viene passato nel costruttore il nome del metodo di cui vuole creare il delegate.

1. *Definire come costruire il design pattern Factory*  e qual è la sua utilità

Si tratta di un pattern creazionale basato su classi e viene utilizzato per creare degli oggetti senza conoscerne i dettagli ma delegando un Creator che, in base alle informazioni ricevute, saprà quale oggetto restituire. Questo pattern consente di separare il Client dal Framework permettendo di modificare i dettagli implementativi senza dovere modificare il Client.