**Domanda 1:**

I Value Type (VT) sono tutti quegli oggetti che contengono direttamente un dato, ne sono un esempio i tipi primitivi (int, double, float..) e DateTime.

I Reference Type (RT) invece sono oggetti che contengono un indirizzo di memoria in cui è effettivamente contenuto il dato, ne sono esempi il tipo string e tutte le classi.

Quando un VT viene dichiarato e inizializzato, viene allocata nello stack una porzione di memoria dove viene salvato il dato.

Nel caso dei RT viene comunque allocata memoria nello stack, però in quella parte di memoria risiede un puntatore che punta ad un’altra porzione di memoria nello heap; questa è quella che effettivamente contiene il dato.

I VT non possono assumere il valore NULL, mentre i RT si.

Quando viene dichiarato un VT ma non istanziato il valore di default è 0, false, o altri a seconda del tipo, mentre nei RT il valore di default assegnato è NULL, ovvero la variabile nello stack non punta a niente.

Abbiamo visto che quando si produce un clone di un VT questo diventa una “entità” a sè infatti se si modifica la variabile originale, il clone non ne risente e vale anche il viceversa.

Al contrario, se si produce un clone di un RT questo rimane “collegato” all’originale perchè di fatto si è copiato l’indirizzo di memoria dove è collocato il dato, non il dato stesso.

**Domanda 2:**

Un delegate è una sorta di puntatore ad un metodo orientato verso la programmazione ad oggetti, infatti si può vedere anche come incapsulamento di un metodo.

La dichiarazione di un delegate è una firma e solo i metodi che avranno la stessa firma del delegate portanno essere invocati dal delegate.

Abbiamo visto ad esempio un delegate per la somma.

delegate int Somma(int primoAddendo, int SecondoAddendo); //questa è la dichiarazione del delegate

//cioè una sorta di firma che vincolerà

// i metodi invocabili

public int MiaSomma(int x, int y){} => return x+y; //metodo invocabile dal delegate //Somma

public int Opposto(int x)() => return -y //metodo non invocabile dal delegate

//Somma

Somma somma = new Somma(MiaSomma); //ho istanziato il delegato

//Somma con somma

Console.WriteLine(“Risultato: “ + somma(2,5)); //Stampo a video il risultato

**Domanda 3:**

Factory è un design pattern di tipo creational che serve per dichiarare e istanziare oggetti con la particolarità di togliere la “responsabilità della creazione” al client richiedente e demandarla ad una classe

Supponiamo che ad un certo punto del program si richiede di avere una certa classe ad esempio Bicicletta.

Utilizzando il design pattern Factory, abbiamo una classe VeicoloFactory che, una volta istanziato un oggetto di tale classe, invoca un metodo come NuovoVeicolo() che crea la bicicletta appoggiandosi ad un’interfaccia.

**Domanda 4:**

Nel Framework .Net sono presenti i seguenti elementi:

Una common infrastructure dove sono presenti Common Infrastructure, Standard Library e gli elementi per la produzione .Net framework, .Net Core e Xamarin.

.Net Framework è stato inserito in .Net soprattutto per garantire retrocompatibilità con applicazioni utilizzate in passato.

.Net Core invece è il framework open source che è stato inserito per sviluppare applicazioni di diverso tipo come desktop, web, ecc..

Infine Xamarin è utilizzato per lo sviluppo Mobile e Game.

Con .Net 5 la Microsoft ha deciso di integrare le tre piattaforme di sviluppo in una unica piattaforma.