# Modulo 1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Deborah |
|  |  | Cognome | Tucci |
|  |  | Data | 26/03/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile **fornendo anche degli esempi**.  
ATTENZIONE: Le domande a risposta multipla possono contenere più risposte corrette.

1. *Dare una panoramica degli elementi principali nel Framework .Net. Cosa è successo con .Net 5?*

.Net si basa su Application Models suddivisi in tre gruppi di tecnologie, .Net Framework, che include WPF, Windows Forms e ASP.NET, per app solo su Window; poi .Net Core, che include UWP e ASP.NET Core, open-source e cross-platform; infine Xamarin, per iOS, Android e OS X, per lo sviluppo di App community e giochi.

Essi si basano sulla libreria comune .Net Standard Library, cioè un insieme di API che le piattaforme .Net devono implementare.

Questa a sua volta si basa su infrastruttura comune standard composta da compilatori, linguaggi e componenti runtime. Infatti il codice sorgente a questo livello viene convertito in Intermediate Language e in fase di esecuzione viene compilato in istruzioni per la CPU che successivamente le esegue.

Con l’introduzione del .Net 5 ci si orienta sempre più verso le Application Web e quindi si prevede un unico insieme di tecnologie cross-platform, tranne quella mobile al momento.

1. *Definire i principi su cui si basa il linguaggio C#*

Il linguaggio C# è un linguaggio orientato agli oggetti fortemente tipizzato, che si basa sui principi di ereditarietà, incapsulamento e polimorfismo.

Incapsulamento è quel principio in base al quale parti di codice, come proprietà o classi, hanno una accessibilità differente, come ad esempio public o protected per le classi, o get e set per le proprietà, in base ai livelli di astrazione, o protezione delle informazioni.

Mentre, il polimorfismo è il principio in base al base al quale si può trattare un’istanza di un tipo come fosse un’istanza di un altro tipo. Questo è possibile in presenza di una derivazione, quando ad esempio si vuole specializzare l’implementazione di un metodo della classe base, che deve essere marcato come virtual o abstrarct, mentre il metodo polimorfico ha bisogno della keyword override.

1. *Dare una spiegazione di Garbage Collector.*

Il Garbage Collector è il meccanismo che si occupa di liberare la memoria, rimuovendo le risorse non più utilizzate, e compattando quelle ancora valide, in modo da poter rendere disponibili le celle di memoria per le risorse future.

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

I Reference Type, a differenza dei Value Type, contengono solo un riferimento all’oggetto (indirizzo cella), nell’ambito dell’heap, e può essere null. Inoltre le modifiche sui reference modificano l’oggetto a cui puntano, e la copia di un reference type consiste nella copia solo della referenza e non dell’oggetto. Un esempio di Refernce Type sono le classi e string.

Invece, i Value Type, a differenza dei Refence Type, sono mantenuti nello stack e contengono direttamente il dato, e non ammettono direttamente il null. Inoltre le modifiche sui Value Type valgono solo sull’istanza corrente, e una copia consiste nella copia dei cadi contenuti nel Value Type. Essi sono i tipi primitivi come int, bool ecc, e gli enum.

1. *Cosa è un delegate?*

Un delegate è l’equivalente .Net dei puntatori a funzione del C/C++, ma a differenza di quest’ultimi sono tipizzati. Quando si crea un'istanza di un delegate, è possibile associare l'istanza a qualsiasi metodo con una firma compatibile e un tipo restituito. I delegate vengono utilizzati per passare metodi come parametri ad altri metodi. Una dichiarazione di un delegate definisce, mediante compilatore, una classe che deriva dalla classe System.Delegate o System.MulticastDelegate.

1. *Scrivere una query LINQ : Date le gare disponibili, ricavare il tempo medio, massimo e minimo per disciplina.*

var tempiByDisciplina = gare.GroupBy(gr => gr.ID)

.Join(partecipazioni,

g => g.Key,

p => p.IDGara,

(g, p) => new

{

Disciplina = g.Key,

Tempo = p.TempoImpiegato

})

.Select(gareByDisciplina => new

{

Disciplina = gareByDisciplina.Disciplina,

TempoMedio = partecipazioni.Average(t => t.TempoImpiegato),

TempoMax = partecipazioni.Max(t => t.TempoImpiegato),

TempoMin = partecipazioni.Min(t => t.TempoImpiegato)

});

1. *Definire cosa è il design Pattern Factory*

Il Design Pattern Factory è un pattern di tipo creazionale e descrive una soluzione generale per la creazione di oggetti in base a diverse caratteristiche, nascondendo la gerarchia del quale fanno parte. Infatti vi è una classe intermedia Factory, alla quale si interfaccia il client, e che si occupa della creazione concreta dell’oggetto accedendo all’interfaccia pubblica della gerarchia.

*Mettere la prova pratica e teorica su Github.*