# Esercitazione di Fine Settimana – Week 1

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Francesca |
|  |  | Cognome | Dicugno |
|  |  | Data | 09/07/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.

1. Quali sono i 3 tipi di runtime disponibili nella piattaforma .NET? Cosa è successo col rilascio di .NET 5?

I 3 tipi di runtime disponibili nella piattaforma di .Net sono: .NET Framework che nasce nel 2001 come nuova piattaforma di sviluppo che contenga diversi paradigmi; .NET CORE, che realizza l’idea di .NET Framework, è una cross-platform (Windows, Linux, Macos) ed è open source; .NET XAMARIN per le Mobile app.

Con il rilascio di .NET 5 Microsoft ha unito I 3 runtime in un’unica piattaforma.

1. Quale è la funzione del Garbage Collector?

.NET è una managed platform e utilizza il Garbage Collector che controlla l’uso della memoria del nostro programma.

1. Descrivere la differenza tra Value Type e Reference Type

I Value Type contengono direttamente il dato nell’ambito dello stack del thread, una copia di un Value Type implica la copia dei dati in esso contenuti, le modifiche hanno effetto solo sull’istanza corrente e non possono essere inizializzati come null. Ne sono esempio gli int, i decimal, i double....

I Reference Type contengono solo un riferimento ad un oggetto nell’ambito dell’heap, la copia di un Reference Type implica la duplicazione del solo reference , le modifiche su due reference modificano l'oggetto a cui puntano e il reference che non referenzia nessuna istanza vale null. Ne sono esempio tutte le classi e le stringhe.

1. Cos’è un Delegate?

E’ la definizione di un nuovo tipo di dato che rappresenta una funzione.

1. Descrivere l’uso delle keyword async / await

Per gestire più attività eseguite in parallelo si utilizza il codice asincrono che fa uso delle keyword async/await . L'async gestisce una funzione asincrona, mentre l’await si aspetta un’operazione asincrona.

1. Data una lista di istanze della classe Votazioni,

public class Votazioni

{

public string Materia { get; set; }

public string Studente { get; set; }

public DateTime Data { get; set; }

public int Voto { get; set; }

}

scrivere una query LINQ (Fluent oppure Query Expression) che restituisca un elenco di materie con il voto medio, quello più alto e quello più basso per ciascuna

//creo una lista voti

var listaVoti =

from voto in votazioni

group voto by voto.Materia into votiGroup

select new

{

Materia = votiGroup.Key,

AverageScore = votiGroup.Average(x => x.Voto),

MaxScore = votiGroup.Max(x => x.Voto),

MinScore = votiGroup.Min(x => x.Voto),

};

1. Descrivere il Factory pattern e perché utilizzarlo:

‎Factory pattern è un modello di progettazione che definisce un'interfaccia per la creazione di un oggetto, ma consente alle classi che implementano l'interfaccia di decidere in quale classe creare un'istanza. Il Factory pattern consente a una classe di posticipare la creazione di un'istanza alle sottoclassi. Il modello factory viene utilizzato per sostituire i costruttori di classi, astraendo il processo di generazione degli oggetti in modo che il tipo dell'oggetto di cui è stata creata un'istanza possa essere determinato in fase di esecuzione. ‎

**Esercitazione Pratica**

Realizzare una Console app (C#) che:

* Effettui il monitoraggio di una cartella in attesa di un file di testo con l'elenco delle spese (*spese.txt*)
* Apra e legga il file. Ogni riga è nel formato:



* Per ogni riga, determini se la spesa è approvata. Esistono diversi livelli di approvazione, a seconda dell'import della spesa
  + **Manager**: spese fino a 400€
  + **Operational Manager**: da 401€ fino a 1000€
  + **CEO**: sopra i 1000€
  + Nessuna spesa sopra i 2500€ è approvata

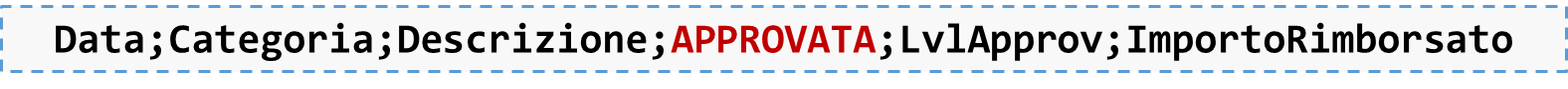
**Usare il Chain of Responsibility pattern**

**(restituire il livello di approvazione)**

* Per ogni spesa approvata, determini l'importo rimborsato sulla base della Categoria
  + **Viaggio**: 100% dell'importo + 50€ fisse
  + **Alloggio**: 100% dell'importo
  + **Vitto**: 70% dell'importo
  + **Altro**: 10% dell'importo

**Usare il Factory pattern**

* Salvi poi le informazioni sulle spese rimborsate e non rimborsate in un file di testo (*spese\_elaborate.txt*)
* Per ogni spesa rimborsata salvare una riga nel formato



* Per ogni spesa non rimborsata salvare una riga nel formato

