Test di fine settimana – Week2

Nome Valentina

Cognome Comerci

Data 28/5/21

1. Dare una definizione di classe statica

- Una classe statica e' una classe che contiene unicamente membri statici
- Le classi statiche sono dette sealed e pertanto non possono essere ereditate per creare oggetti. Possono ereditare solo dalla classe Object.
- per accedere le "proprietà'" di queste classi devo richiamare la classe stessa

2. Quali tipologie di ereditarietà sono consentite in C# e come si definisce? Fornire un esempio

In c# abbiamo principalmente tre dei 4 tipi di ereditarietà' esistenti:

1) Simple Inheritance

- Dato un veicolo generico con proprietà quali (ruote, targa)
- Posso creare una classe derivata da quella veicolo, una classe Automobile che erediterà tutte le proprietà della classe veicolo e alle quali potro' aggiungerne altre più specifiche, relative solo a Automobile(esempio: porte).
- L'ereditarietà funziona solo top- bottom quindi Automobile eredita proprietà di Veicolo ma non viceversa

2)Hierarchical Inheritance

- seguendo il precedente esempio, dalla classe Veicolo posso creare un' altra classe, Moto, allo stesso livello di Automobile
- Questa classe a sua volta erediterà le proprietà ruote e targa, NON la proprietà porte
- a queste proprietà ereditate ne potro' aggiungere altre specifiche e uniche a Moto (sospensioni)

3) Multilevel Inheritance

- Automobile può a sua volta avere una classe "figlia"
- Questa erediterà rispettivamente sia le proprietà di Veicolo, sia quelle specifiche della classe da cui derivano, alle quali si potranno aggiungere altre proprietà ad essa specifiche.

3. Elencare le principali caratteristiche della classe System.Object.

- In .NET tutto deriva dalla classe OBJECT in un ereditarietà multilevel
- se non specifichiamo una classe da cui ereditare il compilatore in automatico la deriverà da OBJECT
- Tutto ciò che deriva da object ne deriva anche i metodi

  alcuni sono:

- ToString
- GetHashCode
- Equals
- Finalize
- GetType

4. Descrivere le due fasi di gestione delle eccezioni

Le eccezioni possono essere gestite con il blocco try/catch evitando di interrompere l'esecuzione dell'applicazione:

try- Serve a raccogliere le istruzioni che rappresentano la fonte di possibile errore/eccezione

```
int a = 0;
bool esito = true;
try
{
    a = Convert.ToInt16(Console.ReadLine());
    esito = true;
}
catch (FormatException ex)
{
    Console.WriteLine("Inserisci un numero e non una stringa");
    esito = false;
```

```
        }
```

catch- Serve a catturare l’errore specifico e presentare un messaggio coerente all’utente

```
catch (FormatException ex)
        {
            Console.WriteLine("Inserisci un numero e non una stringa");
            esito = false;
        }
```

Ed eventualmente anche :

finally- Per eseguire un'operazione obbligatoriamente alla fine a prescindere dall’esito

```
finally
        {
            a = 0;
        }
```