INTRODUZIONE ALLE CLASSI

CORSO DI GAME PROGRAMMING
1° ANNO

Docente **Davide Caio**

ACCADEMIA ITALIANA VIDEOGIOCHI

- Classi
- Costruttori
- Modificatori di visibilità
- Esercizi

CLASSE - DEFINIZIONE

Il concetto di **classe** è alla base di ogni linguaggio di programmazione orientato agli oggetti ed ha la potenza di definire le caratteristiche di un insieme di oggetti che hanno proprietà e compiono azioni uguali.

Esistono tanti tipi di automobili al mondo, ma tutte condividono caratteristiche in comune (4 ruote, hanno un colore, una potenza del motore). Tutte le automobile appartengono alla classe automobile che ne definisce le caratteristiche comuni.

In pratica, possiamo vedere una classe come una **collezione di variabili, metodi e proprietà.**

In C# ogni tipo creato attraverso una classe sarà un **tipo di RIFERIMENTO.**

KEYWORD CLASS

Per definire una classe si utilizza keyword class

```
class MyClass {

}
```

Ogni volta che viene definita una nuova classe, viene anche automaticamente creato un **costruttore** di default dal compilatore. *Il costruttore è il metodo che viene invocato nel momento della creazione di un oggetto di una determinata classe.*

COSTRUTTORE DI UNA CLASSE

Il costruttore è quindi il codice che viene eseguito nel momento in cui creiamo una nuova istanza di quella classe (keyword **new**). Il costruttore di default non fa praticamente nulla. Risulta quindi essenziale poterlo esplicitare. Il costruttore è l'unico metodo della classe **senza un tipo di ritorno** e con lo **stesso nome della classe**.

```
class MyClass {
  //costruttore
      public MyClass () {
            Console.WriteLine ("creato un nuovo oggetto di tipo MyClass");
      }
      //override costruttore
      public MyClass (int numero) {
            Console.WriteLine ("creato un oggetto di tipo MyClass e il numero
passato nel costruttore è: " + numero);
      }
}
```

CREARE UNA ISTANZA DI UNA CLASSE

Un **oggetto** non è nient'altro che una *istanza* di una classe. Potete pensare a un oggetto come una parte di memoria in cui ci sono i dati dell'oggetto, a cui **una variabile** di quel tipo **punta**.

MyClass newObjMyClass = **new** MyClass ();

In questo modo ho dichiarato una variabile newObjMyClass di tipo MyClass attraverso la keyword new seguita dal costruttore della classe. ATTENZIONE → fino a quando non inizializzo la variabile newObjMyClass il suo valore sarà **null**.

CAMPI DI UNA CLASSE

Una classe può avere una serie di **campi**, che rappresentano le informazioni immagazzinate all'interno della classe.

```
class MyClass {

    int myInt;

    //costruttore
    public MyClass () {
        Console.WriteLine ("creato un nuovo oggetto di tipo MyClass");
    }
    //overloading costruttore
    public MyClass (int numero) {
        myInt = numero;
        Console.WriteLine ("creato un nuovo oggetto di tipo MyClass");
    }
}
```

METODI DI UNA CLASSE

Una classe può avere una serie di **metodi**, funzioni che solitamente e*ffettuano delle operazioni sui campi* stessi della classe. Questi devono sempre avere un tipo di ritorno, a differenza del costruttore.

```
class MyClass {
        int myInt;
        //costruttore
        public MyClass () {
                Console.WriteLine (“è stato creato un nuovo oggetto di tipo MyClass”);
        }
        //override costruttore
        public MyClass (int numero) {
                myInt = numero;
                Console.WriteLine (“è stato creato un nuovo oggetto di tipo MyClass”);
        }

        public void IncrementMyInt () {
                myInt ++;
        }
}
```

KEEP CALM

Abbiamo visto molti concetti fino ad ora. Cerchiamo di capire bene di cosa stiamo parlando. Estendiamo la nostra classe con un metodo per stampare il valore del campo myInt

```
class MyClass {
....
....
public void PrintMyInt () {
        Console.WriteLine (myInt);
}

}
```

```
MyClass obj1 = new MyClass (10);
MyClass obj2 = new MyClass (10);

obj1.PrintMyInt ();
//Cosa viene stampato a console?

obj2.PrintMyInt ();
//Cosa viene stampato a console?

obj1.IncrementMyInt ();

obj1.PrintMyInt ();
//Cosa viene stampato a console?
obj2.PrintMyInt ();
//Cosa viene stampato a console?
```

STRUCT VS CLASS

Entrambi i costrutti servono per definire un nuovo tipo di dato all'interno di C#.

La differenza, come già detto, è che:

- La struct serve per creare un nuovo tipo di variabile di **tipo di VALORE**
- La class serve per creare un nuovo tipo di variabile di **tipo di RIFERIMENTO**

ESERCIZIO

Proviamo a creare una class PlayerClass e una struc PlayerStruct e cerchiamo di evidenziare bene questa differenza.

MODIFICATORI DI VISIBILITÀ / ACCESSO

Ogni **membro** (campo, metodo o proprietà) di una classe ha un determinato **livello di visibilità**. Questo livello di visibilità imposta dove è visibile quel membro all'interno del nostro progetto. I principali modificatori di visibilità sono:

public -> il membro è visibile ovunque nel mio progetto (e nei progetti in cui importo la mia classe).

protected -> il membro è visibile solo nella classe stessa e nelle classi che ereditano da essa.

private -> il membro è visibile solo all'interno della definizione della classe stessa.

internal -> il membro è visibile ovunque all'interno dello stesso assembly.

ESERCIZIO 1

Creare una classe per rappresentare il concetto di automobile.
Un'automobile ha un colore, una potenza (in cavalli), una tipologia (3 porte oppure 5 porte) una marca e un modello.
Dopo aver costruito la classe, creare un metodo che ritorni la conversione in kilowatt della potenza della automobile.

Istanziare quindi due oggetti della classe creata e manipolare i parametri.

ESERCIZIO 2

Ampliamo l’esercizio 1 con il concetto di concessionario.
Per rappresentare il concetto di concessionario dobbiamo ampliare il concetto di automobili ed assegnargli un prezzo.

Un concessionario è un negozio di automobili. Ha un nome, un indirizzo e il suo parco auto.

Scrivere un metodo della classe creata che, accettando un prezzo come parametro, ritorni tutte le macchine in quel concessionario che abbiano un prezzo inferiore al parametro inserito.