# Modulo 5

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Deborah |
|  |  | Cognome | Tucci |
|  |  | Data | 26/02/2021 |

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. *Spiegare il funzionamento di Enum*

**Enum** è un tipo di dato definito dall’utente, basato sugli interi. Dunque permette di definire un nuovo tipo con nome più comprensibile per l’utente. Ogni valore, ammissibile dal tipo definito, è associato ad un intero. Ma si accede all’intero tramite il nome dell’attributo associato. Ad esempio si può dichiarare un tipo Enum Esito con i soli attributi ammissibili Promosso, Bocciato, Ritirato. Il codice è il seguente:

public enum Esito { Promosso, Bocciato, Ritirato }

un esempio di dichiarazione e inizializzazione della variabile di tipo Esito è:

public Esito valutazione = Esito.Promosso;

1. *Spiegare la differenza tra Array, Collection e Collezioni Generiche*

L’**Array** non rientra nelle collection, ed è un tipo che organizza i dati in una sequenza finita di elementi di uno stesso tipo, dunque ha forte tipizzazione. La sua dimensione deve essere necessariamente stabilita in fase di inizializzazione.

Esempio int[] arrayOfNum = new int[6]; dove 6 indica la dimensione. E non può essere direttamente modificata ma va ridimensionata tramite il metodo Resize creando un nuovo array con la nuova dimensione.

Agli elementi ci si può accedere mediante l’indice che parte da zero.

Ad esempio int num1 = arrayOfNum[3];

Le **Collection** invece sono delle strutture di dati che possono contenere qualsiasi tipo di oggetto, e perciò, a differenza degli array, sono debolmente tipizzati. Un esempio di Collection è l’ArrayList che può contenere al suo interno elementi di diverso tipo, e la dimensione è variabile ad ogni inserimento.

Ad esempio var elenco = new ArrayList(); dove la keyword var indica che il tipo non è stabilito ma sarà dedotto dal compilatore. Come con i seguenti elementi: elenco.Add(4); elenco.Add("Manager"); elenco.Add(1.50);

Infine, le **Collection Generic** sono delle collection che possono contenere un solo tipo di elementi definito dall’utente, e perciò, sono a forte tipizzazione. Un esempio di Collection Generic è la List<T>, dove T è il parametro che stabilisce il tipo specifico che devono avere gli elementi della collezione deciso dall’utente al momento dell’inizializzazione della List. Esempio var lista = new List<decimal>(); quindi nell’inizializzazione non si definisce la dimensione ma il tipo si deve indicare.

1. *Quanti valori di ritorno può avere un metodo? Commentare la risposta.*

Un metodo può avere un solo **valore di ritorno**. Ma è possibile far ritornare più valori con l’utilizzo della keyword out nei parametri del metodo, esempio

public static int GetData (int a, int b, out int result)

{ result = a + b;

return a++ };

oppure ritornando una classe o una struttura che contenga più valori;

o con l’utilizzo delle tuple, cioè strutture che possono contenere oggetti di tipo diverso, fino ad un massimo di 8 elementi, estendibili con l’utilizzo di una nuova tupla.

1. *Spiegare la differenza tra classe, oggetto e tipo.*

Il **tipo** indica una rappresentazione di un concetto, come ad esento il tipo int che fornisce una rappresentazione concreta di un numero reale. I tipo possono essere di tipo Value Type o Reference Type.

Invece la **classe** è un reference type definito dall’utente, nella quale si definiscono i campi, proprietà e metodi.

Mentre l’**oggetto** è l’istanza di una classe ed è caratterizzato da un’identità; da un comportamento definito dall’elaborazione dei metodi; e uno stato definito dai campi e le proprietà.

1. *Descrivere la struttura di una classe*

Una **classe** è composta da dei campi, delle proprietà che accedono ai campi, da metodi, cioè funzioni o procedure che ne definiscono il comportamento, e da costruttori, cioè metodi particolari che servono per creare l’istanza della classe.

1. *Descrivere le differenze tra classe e struct*

Anche le **struct**, come le classi, possono definire i campi, le proprietà e i metodi e hanno i costruttori di dafault. Ma a differenza delle classi, nelle struct non è possibile sovrascrivere un costruttore di default, ma è possibile definire un costruttore necessariamente con parametri. Inoltre, a differenza delle classi, le struct non supportano l’eridetarietà e il metodo Finalize, cioè il distruttore. Un’altra differenza fondamentale è che le classi sono un Reference Type, mentre le struct sono un Value Type.

1. *Definire cosa è un Generic e descriverne possibili utilizzi.*

Un **Generic** permette di scrivere classi e metodi generici indipendenti dal tipo con l’utilizzo della keyword T. Anzichè scrivere metodi differenti per tipi differenti è possible scrivere un unico metodo per tipi definiti in fase di utilizzo. Questo consente di scrivere un codice più flessibile perchè si scrive un solo metodo che accetta più tipi. Quindi si possono usare le Collections Generic. Utilizzando con i Generics è il compilatore si accorge del tipo inserito, questo permette di avere meno codice da scrivere e mantenere, più riuso del codice.

Esempio public static void Scambio<T> ( T a, T b) { ... }

1. *Descrivere le differenze tra metodo statico e d’istanza*

Un metodo **static** non dipende dalla classe e non può essere legato ad una istanza. L’esecuzione di un metodo statico da altre classi si attiva indicando il nome della classe seguito dal nome del metodo. Mentre i metodi **d’istanza** dipendono dalla classe relativa e sono utilizzabili dalle istanze della classe. In pratica dalla classe si crea un oggetto che invocherà l’esecuzione dei metodi di istanza. Esempio il metodo Next di Random.

1. *Fornire una definizione di modificatore e accessor. Spiegarne l’uso e le differenze.*

I modificatori stabilistono la visibilità del membro. Vi sono diversi livelli di visibilità, quali:

Public con la quale si rende il metodo o la classe accessibile da tutte le classi anche dall’esterno.

Protected accessibile solo dalle classi derivate

Private invece visibile solo all’interno della classe, non accessibile dall’esterno. Dunque un campo private è legato ad una classe gestito all’interno e non modificabile dall’esterno. Esempio ad un campo private vi può accedere una proprietà public ma interna alla classe nella quale sono dichiarati.

Internal invece è accessibile solo all’interno dell’assembly, quindi si usano all’interno di C# ma non ci si può accedere.

Internal protected, infine, è la combinazione dei due modificatori di accesso. Quindi un membro internal protected è accessibile dall'assembly corrente o dai tipi che derivano dalla classe che li contiene.

Mentre gli accessor stabiliscono l’accessibilità ad esempio delle proprietà ai campi in lettura con get e/o in scrittura con set.

*Esercitazione pratica*

*Creare una Console App che gestisca l’iscrizione ad un esame di uno Studente.*

*Lo studente è definito con:*

* *Nome*
* *Cognome*
* *AnnoDiNascita*
* *Immatricolazione*
* *Esami*
* *RichiestaLaurea*

*L’immatricolazione ha le seguenti caratteristiche:*

* *Matricola*
* *DataInizio*
* *CorsoDiLaurea*
* *FuoriCorso*
* *CFUAccumulati*

*Un Corso di laurea è dato da un Nome, AnniDiCorso, i cfu per ottenere la laurea e una lista di corsi associati.*

*Un Corso ha un nome e dei CFU.*

*Un Esame si riferisce ad un corso e tiene conto se esso è stato passato.*

*I possibili nomi dei Corsi di Laurea possono essere solo i seguenti: Matematica, Fisica, Informatica, Ingegneria, Lettere.*

*La matricola dello studente deve essere univoca, autogenerata e read-only.*

*Uno studente può richiedere un esame solo se esso è presente nel Corso di Laurea associato allo studente, se i CFU del corso associato all’esame non superino i CFU massimi del Corso di laurea e se non ha il flag RichiestaLaurea assegnato a vero.*

*Nel caso le condizioni siano verificate, lo studente aggiunge l’esame alla lista Esami.*

*Scrivere inoltre un metodo EsamePassato che, dato un esame, vada ad aggiornare i CFU accumulati dallo studente, metta il flag Passato sull’esame e verifichi se con tale esame sono stati raggiunti i CFU necessari per richiedere la laurea (e quindi metta il flag Richiestalaurea a true);*

*Requisiti tecnici:*

*-Specificare almeno 3 costruttori*

*-Usare almeno una volta enum*

*Consigli:*

*-Potrebbe essere utile creare un paio di metodi ad hoc per creare al volo delle liste di corsi, corsi di laurea…*

*-Visto che le classi sono collegate strettamente l’una con le altre, verificate l’inizializzazione di ciascuna sia adeguata e che i riferimenti siano corretti.*

*Mettere la prova pratica e teorica su Github.*