# Modulo 5

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Mara |
|  |  | Cognome | Barani |
|  |  | 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*

*Un Enum è un tipo definito dall’utente per esprimere più valori possibili di una certa variabile. Per esempio:*

*public Enum SemeCarta {*

*Cuori,*

*Quadri,*

*Picche,*

*Fiori*

*}*

*È una numerazione basata su interi, perché a ciascun valore possibile di SemeCarta viene associato un intero (a partire dal primo nome al quale sarà associato 0). C’è quindi una corrispondenza biunovoca tra i nomi in enum e i numeri interi. Si può accedere ad un valore di enum tramite il suo nome, es: SemeCarta.Picche, oppure tramite un intero, es. (SemeCarta)2.*

*In definitiva, l’utilizzo di enum semplifica molto il codice, perché possiamo definire un tipo che contiene i valori che ci interessano e viene creata una corrispondenza tra questi valori e i numeri interi.*

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

*Un Array è un insieme finito di elementi dello stesso tipo, la cui lunghezza deve essere nota a priori. Gli array sono fortemente tipizzati, cioè in un array è possibile inserire solo elementi dello stesso tipo.*

*Una Collection è un insieme di elementi definita nel namespace System.Collections. La lunghezza non deve essere nota a priori, perché si possono aggiungere un numero teoricamente infinito di elementi durante la scrittura del codice. Inoltre gli elementi possono essere di tipi diversi. Per questo le collections sono debolmente tipizzate. System.Collections contiene: ArrayList, Hashtable, Stack, Queue.*

*Una Collezione Generica è un insieme di elementi dello stesso tipo ed è definita del namespace System.Collections.Generics. Sono una via di mezzo tra gli array e le collections, perché come gli array sono tipizzati, cioè possono accettare elementi di un unico tipo, e come le collections non hanno una lunghezza fissa, ma può variare in fase di compilazione. Le collezioni generiche sono fortemente tipizzate e si indicano con <T>, dove T sta per tipo. System.Collection.Generics contiene per esempio: List<T>, Dictionary<Tkey, Tvalue>, Stack<T>, Queue<T> .*

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

*Un metodo ha un solo valore di ritorno (tramite return). Però è possibile fargli ritornare più valori in diversi modi:*

* *Utilizzando la parola chiave “out” quando passiamo un parametro dall’esterno. In questo modo possiamo salvare un risultato che ci interessa nella variabile passata con out;*
* *Utilizzando come valore di ritorno una classe o una struttura;*
* *Utilizzando le tuple. Con le tuple possiamo ritornare fino ad 8 oggetti di tipo diverso.*

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

*Un tipo è una rappresentazione concreta di un concetto reale. Per esempio, int e la rappresentazione di un numero intero.*

*Una classe è un tipo definito dall’utente. Una classe è un oggetto di tipo reference ed è formata da membri, che sono: campi, proprietà, metodi ed eventi.*

*Un oggetto è l’istanza di una classe, e può contenere dati, funzioni e procedure. Queste ultime due possono sfruttare le informazioni di stato dell’oggetto, cioè i dati, per fare delle elaborazioni. Un oggetto è caratterizzato da:*

* *un’identità, perché è distinto dagli altri*
* *un comportamento, perché compie elaborazioni tramite metodi*
* *uno stato, perché memorizza dati tramite campi e proprietà*

1. *Descrivere la struttura di una classe*

*Una classe è formata da membri, che sono: campi, proprietà, metodi ed eventi.*

* *Campi: sono variabili interne (private) di una classe che ne descrivono lo stato*
* *Proprietà: permettono di accedere ai campi in lettura e/o scrittura grazie a metodi ldi lettura (get) e di scrittura (set).*
* *Get e set accedono ai campi e l’associazione tra get e set e i campi viene definita “proprietà”.*
* *Metodi: definiscono un comportamento o un’elaborazione relative ad un oggetto. Un metodo è una routine associata ad una classe, quindi è caratterizzato da possibili modificatori di accesso (per esempio public, private...), tipo di ritorno o void, nome del metodo e parametri in ingresso. è possibile fare l’overloading di un metodo, cioè definire un altro metodo con lo stesso nome ma tipi e numero di parametri in ingresso e valore di ritorno diversi. I costruttori di una classe sono particolari tipi di metodi.*
* *Eventi: elementi che permettono di inviare delle notifiche in corrispondenza di certi avvenimenti.*

1. *Descrivere le differenze tra classe e struct*

*Una classe e una struttura sono molto simili, perché entrambe vengono utilizzate per definire dei tipi creati dall’utente tramite campi, proprietà, metodi ed eventi. Le differenze tra le due sono:*

* *Le classi supportano l’ereditarietà mentre le strutture no;*
* *Le classi supportano costruttori di default, mentre le struct no (deve avere per forza dei parametri in ingresso);*
* *Le classi sono di tipo reference mentre le strutture di tipo value;*
* *Le classi supportano il Finalize mentre le struct no. Infatti, essendo le strutture di tipo value, una volta che è finito lo scope, queste vengono automaticamente eliminate dallo stack. Questo non accade per le classi, che essendo di tipo value sono salvate nel managed heap, e quindi c’è bisogno del garbage collector per liberare la memoria.*

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

*Un Generic consente di scrivere classi, metodi ed interfacce indipendenti dal tipo. Quindi è possibile scrivere un solo metodo per tipi diversi invece che scrivere metodi diversi per ciascun tipo. In questo modo abbiamo meno codice da scrivere a da cambiare se ci fosse bisogno, e possiamo riutilizzarlo per più tipi. Possiamo farlo perché con i Generics il compilatore si accorge di che tipo stiamo inserendo.*

*Un metodo generico lo definiamo in questo modo:*

*public static void Swap<T> (ref T primo, ref T secondo) {*

*T temp=primo;*

*primo=secondo;*

*secondo=temp;*

*}*

*e possiamo richiamarlo, anche omettendo le <>, a seconda del tipo:*

*Swap(int primo, int secondo)*

*Swap(float primo, float secondo)*

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

*Un metodo static può essere utilizzato senza la necessità di istanziare oggetti, mentre per utilizzare un metodo d’istanza è necessario creare l’istanza di un oggetto. Es:*

*public class myClass{*

*public static int One;*

*public int Two;*

*}*

*MyClass.One 🡪 chiamata a metodo statico*

*MyClass c1=new MyClass();*

*c1.Two 🡪 chiamata a metodo d’istanza*

*Static viene utilizzato per tutte quelle informazioni che non dipendono dall’istanza in sè ma che sono comuni a tutti gli oggetti dello stesso tipo.*

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

*Gli accessor sono i metodi speciali get e set per leggere e srivere i campi di una classe/struttura.*

*I modificatori sono parole chiave usate per modificare la dichiarazione di tipi definiti dall’utente (classi, strutture, enum...) e membri di tipi (campi, proprietà, metodi...). Noi in particolare abbiamo visto gli access modifiers, utilizzati per dichiarare l’accessibilità di tipi e membri di tipi tramite le principali parole chiave: public (accessibilità da ovunque), private (non accessibile all’esterno), protected (accessibile solo dalle classi derivate).*

*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.*