# Test di valutazione – Modulo 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Ramiela |
|  |  | Cognome | Gurra |
|  |  | Data | 18.10.2019 |

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. *Cosa si intende quando si utilizza il tipo “var” durante la dichiarazione delle variabili?*

la variabile non ha un tipo specifico quindi può assumere qualunque tipo

il motore di .NET determina il tipo a runtime time

il compilatore C# determina il tipo a compile time

la variabile ha un tipo implicito che potrebbe essere anche esplicitamente dichiarato a codice

1. *A cosa servono il Costruttore e il Distruttore di una classe?*

A creare e distruggere la classe a cui sono applicate

A gestire le operazioni di inizializzazione e pulizia delle risorse di una classe

A permettere ad una classe di gestire in maniera efficiente il Garbage Collector

A inizializzare una classe senza che la stesse venga terminata dal Garbage Collector

1. *Quali sono i pilastri sulla quale si basa la OOP? Dare una definizione di ciascuno di essi descrivendo per quale motivo sono fondamentali per questo tipo di approccio allo sviluppo*

*La programmazione ad oggetti è basata su oggetti e classi .I pilastri sono l’ereditarietà, l’incapsulamento ed il polimorfismo.*

*L'ereditarietà è una caratteristica dei linguaggi di programmazione orientati a oggetti che consente di definire una classe di base con funzionalità specifiche (relative a dati e comportamento) e classi derivate che ereditano o eseguono l'override di tali funzionalità.*

*L'incapsulamento indica che un gruppo di proprietà, metodi e altri membri correlati vengono considerati come una singola unità o un singolo oggetto.*

*Il polimorfismo indica la capacità di usare più classi in modo intercambiabile, anche se in ognuna di esse le stesse proprietà o gli stessi metodi sono implementati in modi diversi.*

1. *Qual è la differenza tra overloading e overriding? Fornire un esempio di una classe per cui si applica un override di un metodo e un overload dello stesso metodo*

*L’overloading dei metodi è il meccanismo con cui C# permette la definizione di più metodi con lo stesso nome. Per poter avere l’overloading l’unica regola da rispettare è quella di avere una lista di argomenti differente, è l’unico elemento utilizzato per distinguere i metodi tra di loro. Il tipo di ritorno e la visibilità, possono essere diversi tra loro ma non sono utilizzati per distinguere i due metodi tra di loro.*

***public class esempio***

***{***

***public void faiCose(int id)***

***{}***

***public void faiCose(string name)***

***{}***

***}***

*L’overriding è il meccanismo che permette la sovrapposizione dei metodi di una superclasse da parte di una sottoclasse. Per poter sovrapporre un metodo proveniente da una superclasse bisogna che il nuovo metodo abbia lo stesso nome, lo stesso tipo di ritorno e la stessa lista di parametri.*

***public class esempio***

***{***

***public virtual void faiCose(int id)***

***{}***

***}***

***public class esempio1 : esempio***

***{***

***public override void faiCose(int id)***

***{}***

***}***

1. *Cosa è il Garbage Collector e a cosa serve nel mondo .NET?*

*Il Garbage Collector di .NET gestisce l'allocazione e il rilascio di memoria per il programma. Ogni volta che si crea un nuovo oggetto, si alloca memoria per l'oggetto. La memoria non è infinita. Alla fine il Garbage Collector deve eseguire una raccolta per liberare memoria. Il modulo di ottimizzazione del Garbage Collector consente di determinare il momento migliore per l'esecuzione di una raccolta sulla base delle allocazioni in corso. Durante l'esecuzione di una raccolta, il Garbage Collector verifica la presenza di oggetti non più usati dall'applicazione ed esegue le operazioni necessarie per reclamare la relativa memoria.*

1. *Qual è la differenza tra Values Type e Reference Type?*

*I tipi Reference sono sempre allocati in heap e i nuovi operatori restituiscono l'indirizzo di memoria dell'oggetto.Essendo allocata memoria nello heap avviene una garbage collection.*

*I Value type invece sono solitamente allocati nello stack e la variabile che rappresenta l'oggetto non contiene un puntatore ad esso, ma direttamente il valore dell'oggetto stesso. I Value type non sono sotto il controllo del Garbage Collector.*

1. *Cos’è e a cosa serve LINQ? Dare una definzione dello stesso, e descrivere per quale motivo è diventato fondamentale per aumentare la produttività di uno sviluppatore .NET*

*LINQ sta per Language INtegrated Query.* *LINQ è un framework per eseguire interrogazioni su sorgenti dati all’interno del linguaggio. E diventato fondamentale perchè ti da la comodita di poter fare delle query dentro il codice, in questo modo hai cmpleto controllo sul codice.*

1. *Quali sono le differenze tra una classe, una classe astratta e un’interfaccia? Elencare quante più caratteristiche possibili di ciascuno di questi costrutti*

*Una classe è un tipo definito dall'utente.Una classe è fatta di proprietà e metodi.* *Non è ammessa l’ereditarietà multipla.*

*Una classe può derivare unicamente da una sola altra classe.*

*Una classe è astratta (abstract) se contiene almeno un metodo astratto.Un metodo è astratto se dichiara la sua firma, ma non fornisce alcuna implementazione. Se una classe derivata deve poter provvedere una nuova implementazione di un metodo, nella classe base questo deve essere contrassegnato come virtuale (virtual).La classe derivata che voglia fornire una implementazione sostitutiva di una classe base deve marcare con override il metodo.*

*Un’interfaccia è simile ad una classe astratta pura, ossia con soli metodi e/o proprietà astratte. è priva di qualsiasi implementazione e di modificatore di accessibilità (public, private).Un’interfaccia definisce un contratto che la classe che la implementa deve rispettare.Una classe può implementare più interfacce contemporaneamente.*

1. Come si definisce una “classe parametrica”? Fornire un esempio della stessa, fornendo sia il codice implementativo che il suo utilizzo pratico

public void Swap<T>(ref T x, ref T y) {

T Temp = x;

x = y;

y = Temp;

}

public class GenericList<T> {

public void Add<T>(){ }

}

var items = new GenericList<Customer>();

items.Add(new Customer(){…});

1. *Esercitazione pratica. Si chiede di creare un app console in .NET Framework che, in fase di esecuzione, permetta di inserire e visualizzare un catalogo di automobili e biciclette (entrambe le entità avranno le medesime funzioni).* 
   * *Quando l’applicazione viene avviata permette di selezione se si vuole lavorare sul catalogo delle biciclette (selezione da menu “A”) o quello delle automobili (selezione da menu “B”).*
   * *Dovrà essere possibile inserire una bicicletta (o un’automobile) e salvarla su un “database” realizzato con un file di json*
   * *Dovrà essere possibile visualizzare il contenuto intero del database dell’oggetto di riferimento*
   * *Dovrà essere possibile visualizzare solo gli elementi che corrispondono al criterio di ricerca sul campo “Modello”.*
   * *La bicicletta è definita dal campo “Modello”, “Marca”, “NumeroTelaio” e “IsElettrica”;*
   * *L’automobile è definita da “Modello”, “Marca”, “NumeroCavalli”, “IsDiesel” e “DataImmatricolazione”.*
   * *Ogni entità è caratterizzata da un campo “Id” numerico che deve essere univoco nella stessa tipologia di oggetti*
   * *Dovrà essere presente una funzione che permette di eseguire la creazione dei mezzi di locomozione di una famiglia (N biciclette e 1 automobile), passando le quantità e i valori come parametri alla funzione CreaMezziDellaFamiglia(int numeroDiBiciclette, string string marcaBiciclette, string[] modelliBiciclette, string marcaAutomobile, string modelloAutomobile)*

[Tot: \_\_/10]