# Test di fine settimana – Week 3

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Isabella |
|  |  | Cognome | Centofanti |
|  |  | Data | 04/06/2021 |

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. *Elencare le principali differenze tra strutture e classi.*

*Le strutture assomigliano molto alle classi come costruzione,ma presentano delle differenze importanti.*

*Innanzitutto la keyword è struct; per creare una struttura in C# bisogna prima creare una classe, poi modificarne il nome(non esiste una Struttura da creare!)*

*Entrambe possono definire data member, proprietà e campi.*

*Le strutture però,a differenza delle classi, non supportano il costruttore di default.*

*Inoltre, non supportano l’ereditarietà. La differenza principale sta nel fatto che le classi sono reference type, mentre le strutture sono value type.*

*Public struct Persona*

*{*

*Public int Eta; //è value type,non mi servono i get e set!!!*

*public string Nome;*

*public Persona(int eta,string nome){*

*Eta = eta,*

*Nome = nome,*

*}*

*Persona persona = new Persona(23, “mario”);*

*Persona persona2 = persona; //se andrò a modificare persona 2,non cambierà anche persona!*

1. *Nell’istruzione* ***Persona p = new Dipendente();*** *la variabile p si comporta in maniera polimorfa. Perché? Cosa si intende per comportamento polimorfico?  
   Il polimorfismo è uno dei principi base della programmazione a oggetti. Significa “avere più forme”,quindi metodi e oggetti possono avere diverse forme.*

*Un oggetto, può avere una classe che estende col principio dell’ereditarietà, da cui erediterà classi e metodi, che verranno copiati tali e quali o potranno essere modificati nella classe figlia. Se ho più classi figlie, è il modo migliore per trasferire le caratteristiche comuni e permettere di implementare in maniera diversa i metodi.*

*In questo caso, p è di tipo formale Persona, di cui potrà richiamare campi/proprietà e metodi;*

*p.Nome = “carlo”; // poniamo che la classe Persona abbia come proprietà nome*

*E di tipo concreto Dipendente: Per poter richiamare le caratteristiche di Dipendente, si dovrà eseguire un cast.*

*Dipendente d = (Dipendente)p*

*d.Stipendio = 1200.00; // poniamo che la classe Dipendente abbia come proprietà stipendio*

1. *In che modo è possibile utilizzare la combinazione di keyword del linguaggio virtual e override? Fornire un esempio.  
     
   L’override è un tipo di polimorfismo legato all’ereditarietà. Quando una classe figlia eredità proprietà e metodi da una classe padre, ha la possibilità di implementare quel metodo andandolo a modificare o semplicemente può richiamarlo*

*Un metodo viene definito virtual (nella classe padre)per dare la possibilità alla classe che lo implementa(classe figlia) di modificarlo.*

*Public Class persona*

*{*

*Public string Nome{get;set;}*

*Public int Eta{get;set;}*

*Public virtual string ToString() //do la possibilità alla classe che lo implementa di modificarlo*

*{*

*Return Nome +Eta;*

*}*

*}*

*Public class Studente*

*{*

*Public string Matricola{get;set;}*

*Public override string ToString() //lo sovrascrivo*

*{*

*Return base.ToString() + MAtricola;*

*}*

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

*Un Array è un insieme finito e ordinato di elementi di uno stesso tipo*

*Int [] numero = new int [6]*

*Avrò un insieme di soli numeri interi, con una dimensione finita. Ogni posizione ha un indice,quindi posso risalire all’elemento in quella posizione*

*In un progetto c’è la necessità di raggruppare oggetti omogenei e le collections servono a questo. Abbiamo 2 classi che hanno diverse specializzazioni in base agli utilizzi e derivano da*

*System.Collections*

*System.Collections.Generics*

*Abbiamo tipi diversi di collection,in base a cosa mi serve aggiungere; ognuno ha metodi diversi, perché estendono diverse interfacce(IEnumerable,ICollection,IList,IDictionary….) Hanno una tipizzazione debole*

*ArrayList : è una vera e propria classe concreta che implementa i metodi presenti in IList,ICollection,IEnumerable. Nell’AL tutto quello che inserisco sono object vari, può contenere elementi di tipo diverso! La dimensione della lista è dinamica. Metodi: Add(it),Remove(it),Clear()…*

*Stack: è un’insieme in cui il primo elemento una volta inserito diventa poi pian piano che vengono aggiunti altri elementi diventa l’ultimo. Inserimento e cancellazione in cima alla pila. Metodi Push e Pop*

*Queue: elementi aggiunti in coda*

*HashTable : coppie chiave valore. Sono utili per trovare determinati elementi*

*Dictionary: sempre coppia chiave valore, utili per rcuperare un insie,e di valori*

*I generics invece vengono usati per scrivere/mantenere meno codice! CI aiutano a gestire casi che fanno parte di più classi.*

*Sono efficaci se ho dei metodi che devo generalizzare, per più elementi di tipo diverso. List<T> T è il mio tipo generico che andrò a gestire in base alla situazione. Non è possibile assegnare null a una classe generics: un tipo generics può essere istanziato come un value type(no nullable!).*

*Posso anche definire delle regole(costraints): usando il where definisco i vincoli sulle classi o gli elementi a cui fanno riferimento i generics.*

*Esercitazione pratica*

*Creare una Console App per la gestione di un carrello di un e-commerce.*

*Il sito ha degli utenti iscritti. (Inserire almeno un utente già iscritto).*

***L’utente*** *è definito con*

* ***Username,***
* ***Password,***
* ***Nome e Cognome****.*

*L’e-commerce prevede che vengano messi in vendita prodotti d’abbigliamento, alimentari e di elettronica tutti caratterizzati da:*

* *un* ***codice****,*
* *una* ***descrizione****,*
* *il* ***prezzo****,*
* *una* ***percentuale di sconto****.*

*I prodotti di tipo abbigliamento specificano anche una* ***taglia*** *(string) e il* ***brand*** *(string), i prodotti alimentari hanno anche una* ***data di scadenza*** *mentre i prodotti di elettronica un* ***produttore*** *(string.*

*Per ogni utente è previsto un solo carrello contenente le righe di dettaglio dell’ordine e il prezzo totale da pagare (nota: è importante risalire dall’utente al carrello e non il viceversa).*

*Ciascuna riga di dettaglio contiene le seguenti caratteristiche:*

* ***prodotto****,*
* *la* ***quantità ordinata****,*
* *il* ***prezzo totale*** *al netto dello sconto (rispetto alla quantità e al prezzo “pieno” del prodotto),*
* *il* ***prezzo totale scontato*** *(calcolato rispetto alla percentuale di sconto del singolo prodotto).*

*All’accesso, viene chiesto username e password. Se sono corrette si accede al menu.*

1. *Aggiungi prodotto al carrello\**
2. *Elimina prodotto*
3. *Modifica la quantità di un prodotto già inserito*
4. *Stampa a video riepilogo del carrello dell’utente (formato a piacere)*
5. *Esci*

*\*Nel caso sia inserito un prodotto che già esiste nel carrello questo va a modificare la quantità del prodotto precedentemente inserito.*

*Requisiti tecnici:*

*- Rappresentare opportunamente la gerarchia dei prodotti.*

*- Utilizzare la collection adeguata considerando che ogni prodotto ha un codice univoco.*

*Consigli:*

*Creare metodi ad hoc per creare le entità di partenza “precaricate” nell’e-commerce.*

*Mettere la prova pratica e teorica su Github.*