# Test di fine settimana – Week 3

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome |  |
|  |  | Cognome |  |
|  |  | Data |  |

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.*

*Sia le classic he le strutture possono definire proprietà e metodi, ma a differenza delle classi, le struct non supportano l’ereditarietà, l’inizializzazione di membri e il metodo Finalize per il Garbage Collector. Inoltre, hanno il costruttore no args implicito in presenza di altri costruttori anche senza che venga dichiarato, mentre nelle classi dovrebbe essere esplicitato. Infine le struct sono dei value type mentre le classi sono reference type.*

1. *Nell’istruzione* ***Persona p = new Dipendente();*** *la variabile p si comporta in maniera polimorfa. Perché? Cosa si intende per comportamento polimorfico?*

*La variabile p si comporta in maniera polimorfa perchè viene dichiarata come variabile di tipo Persona ma istanziata come variabile di tipo Dipendente, che è una sottoclasse di Persona. Ciò comporta che sulla variabile p si possono usare solo i metodi definiti nella classe Persona, ma nel caso in cui questi metodi siano di tipo virtual e vi sia stato fatto un override nella sottoclasse Dipendente, chiamado quel metodo su p in realtà si sta chiamando il metodo della sottoclasse. In caso in cui non venga fatto alcun override ma nella sottoclasse esista un metodo con la stessa firma ciò non avviene, e su p viene applicato il metodo della superclasse Persona. Ciò consente di creare array contenenti oggetti della superclasse ma usare ciascun oggetto in modo più specifico con i metodi della sottoclasse.*

1. *In che modo è possibile utilizzare la combinazione di keyword del linguaggio virtual e override? Fornire un esempio.*

*Le keyword* ***virtual*** *e* ***override*** *servono, nel caso di gerarchia tra classi, per modificare il metodo contrassegnato con virtual in una superclasse nella sottoclasse, usando appunto il termine override. In questo modo si può usare il polimorfismo delle variabili come spiegato sopra.*

*Es: nella classe* ***Object*** *è presente la definizione del metodo* ***ToString()*** *in modalità virtual. Ogni classe è sottoclasse di Object, quindi si può fare l’override. Per esempio nella classe Persona:*

*public override string ToString()*

*{*

*return $“Nome: {Nome} – Cognome: {Cognome}”;*

*}*

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

*Array, Collection e Collezioni Generiche sono tutti collezioni/raggruppamenti di oggetti. La principale differenza consiste nei tipo che possono permettere:*

***Array****: è necessario dichiarare da subito di che tipo si vuole fare la collezione. È inoltre necessario indicare la dimensione che deve avere l’array, e una volta stabilita non può essere cambiata.*

*Es: int[] a = new int[3];*

***Collection****: si intendono per esempio arraylist, code, pile... Sono dette a tipizzazione debole perchè possono contenere oggetti di qualisasi tipo, anche in una stessa collection. Se si vuole leggere e usare tali oggetti per il loro specifico tipo è necessario effettuare un cast. Non hanno una dimensione fissata a priori, e può essere cambiata nel corso della compilazione.*

***Collezioni Generiche****: Sono delle collections che permettono di essere definite per qualsiasi tipo, ma una volta fissato il tipo deve restare quello per tutto il resto della compilazione. Sono per esempio le liste List<T>, i dizionari Dictionaries<TKey,TValue>, le code Queue<T>, le pile Stack<T>. Essendo collections hanno la dimensione variabile durante il corso della compilazione.*

***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.*