# Esercitazione di Fine Settimana – Week 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Michela |
|  |  | Cognome | Putzu |
|  |  | Data | 17/09/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.

1. Descrivere le modalità per ritornare più valori da un metodo in C#

Per ritornare più valori in C# possiamo utilizzare:

* Una classe/struttura con tutti i valori necessari
* Una tupla
* La parola chiave out

1. Descrivere le due tipologie di casting tra tipi in C#

Possiamo avere due tipologie di casting:

* ***Conversioni implicite***: non è richiesta alcuna sintassi speciale perché la conversione ha sempre successo e non verranno persi dati 🡪 conversioni da tipi integrali più piccoli a più grandi e conversione da classi derivate a classi base.
* ***Conversioni esplicite*** (cast): le conversioni esplicite richiedono un’espressione di cast. Il cast è richiesto quando si potrebbe perdere informazione nella conversione o quando la conversione potrebbe non riuscire per altri motivi.

1. Quali sono gli utilizzi della keyword static?

E’ una classe che non può essere istanziata, non è quindi possibile utilizzare l’operatore new. Poiché non esiste una variabile dell'istanza, si accede ai membri di una classe statica tramite il nome stesso della classe. È più frequente dichiarare una classe non statica con alcuni membri statici, che dichiarare un'intera classe come statica. Due utilizzi comuni di campi statici sono: tenere un conteggio del numero di oggetti di cui è stata creata un'istanza o archiviare un valore che deve essere condiviso fra tutte le istanze.

1. Descrivere le modalità di implementazione di un evento in C#

Un evento è un membro che permette alla classe di inviare notiche verso l’esterno. L’evento mantiene una lista di subscriber che vengono iterati per eseguire la notifica. Quindi si parla di:

* Publisher 🡪 inoltra gli eventi a tutti i subscriber
* Subscriber 🡪 riceve gli eventi dal publisher

Ciascun subscriber dovrà essere aggiunto alla lista del publisher oppure rimosso.

Su C# l’evento si dichiara con “event” e lo si invoca come un normale metodo: per eseguire il subscribe è sufficiente aggiungere all’evento un delegate. E’ buona norma che il delegare per un evento passi due parametri:

* Sender 🡪 oggetto che scatena l’evento
* Args 🡪istanza di una classe che deriva da EventArgs.

1. A cosa serve l’interfaccia IEnumerable<T>? Come si implementa in una nostra classe?

IEnumerable<T> è l'interfaccia di base per le raccolte nello spazio dei nomi System.Collections.Generic. IEnumerable<T> contiene un singolo metodo che è necessario implementare quando si implementa questa interfaccia; GetEnumerator, che restituisce un oggetto IEnumerator<T>. IEnumerator<T> restituito consente di scorrere l'insieme esponendo una proprietà Current.

**Esercitazione Pratica**

* Realizzare una classe Warehouse per gestire un Magazzino Merci, con le seguenti proprietà:
  + *Id Magazzino (GUID)*
  + *Indirizzo*
  + *Importo Totale Merci in giacenza*
  + *Data Ultima Operazione*
  + *Lista delle Merci in giacenza*
* Realizzare **l’overload degli operatori + e –** in modo che sia possibile aggiungere e rimuovere Merci dalla lista (l’overload dovrà anche occuparsi di aggiornare l’Importo Totale e la Data di Ultima Operazione)
* Realizzare un metodo StockList() che stampi i dati del Magazzino, inclusa la lista delle Merci in giacenza
* Realizzare una gerarchia di classi per rappresentare le Merci (Good). Tutte le classi avranno le proprietà
  + *Codice Merce*
  + *Descrizione*
  + *Prezzo*
  + *Data di Ricevimento*
  + *Quantità in Giacenza*
  + Realizzare le classi che rappresentano:
    - ElectronicGood, con la proprietà aggiuntiva *Produttore*
    - PerishableGood, con le proprietà aggiuntive *Data di Scadenza* e *Modalità di Conservazione* (enum con i valori FREEZER, FRIDGE e SHELF)
    - SpiritDrinkGood, con le proprietà aggiuntive *Tipo* (enum con i valori WHISKY, WODKA, GRAPPA, GIN e OTHER) e *Gradazione Alcoolica*
  + Tutte le classi saranno dotate di costruttore che accetti tutti i parametri necessari per popolare le proprietà
  + Tutte le classi dovranno implementare la propria versione del metodo ToString() e visualizzare tutti i dati
  + ***OPZIONALE***: realizzare una procedura di caricamento dati da un file della lista delle Merci in giacenza (l’implementazione di eventi per notificare le fasi di caricamento dati costituisce un bonus)
  + ***OPZIONALE 2***: utilizzare una (o più) eccezione custom per gestire tutte le tipologie di errori che dovessero verificarsi durante l’utilizzo della classe Warehouse e delle classi Good
* Realizzare una Console app che
  + Crei un nuovo Magazzino
  + Permetta di ricevere diverse tipologie di Merci (gestire l’input dall'utente)
  + Stampi i dati del Magazzino e le Merci in giacenza