# Esercitazione di Fine Settimana – Week 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Giuseppe |
|  |  | Cognome | Nobile |
|  |  | 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#
2. Descrivere le due tipologie di casting tra tipi in C#

CASTING IMPLICITO: in genere viene fatto in automatico dal programma, per esempio quando si deve convertire un tipo più piccolo in uno più grande (es.: int -> double)

CASTING ESPLICITO: richiede la scrittura di espressioni di conversione da parte del programmatore, eseguito quando si rischia di perdere informazioni con la conversione di tipi (es.: double -> int)

1. Quali sono gli utilizzi della keyword static?

Viene utilizzata quando vogliamo creare delle classi, dei metodi o delle proprietà statiche. Una classe statica è utile quando ci serve solo un’ istanza di un oggetto, una propeirtà statica è utile quando sappiamo che quelle proprietà rimarranno invariate, mentre un metodo possiamo definirlo crearlo statico quando è un’istanza della classe e non dell’oggetto (es.: Console.WriteLine())

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

Si può implementare in due modi:

* Come Publisher, dove invia notifiche a tutti i suoi Subcriber
* Come Subscriber, dove riceve notifiche dal Publisher

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

Serve a fare operazioni su un insieme di oggetti in sequenza, utilizzando un enumeratore

**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