# Esercitazione di Fine Settimana – Week 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Katia |
|  |  | Cognome | Caracciolo |
|  |  | 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#  
   - ritornando una classe: la classe conterrà tutti i valori che si vuole ritornare  
   - ritornando una tupla: dalla versione 8 di C#, è un insieme di valori ordinato che deve essere restituito nell’ordine in cui si decide di avere il ritorno  
   - con parametro out: deve essere assegnato prima del return. Un esempio di utilizzo si trova nel metodo TryParse: ritorna un booleano ma prende come parametri una stringa e un parametro out che viene poi salvato al di fuori del metodo stesso.
2. Descrivere le due tipologie di casting tra tipi in C#  
   Il casting può essere implicito o esplicito: in quello implicito il tipo viene convertito direttamente senza problemi, mentre in quello esplicito è necessario specificare il tipo nel quale deve avvenire la conversione.  
   Es. int i = 3;  
    int? a = i; //conversione implicita  
     
    double d = 43.2;  
    int b = d; //non è possibile  
    int b = (int)d; //conversione esplicita
3. Quali sono gli utilizzi della keyword static?  
   I membri static sono membri di classe, sono richiamabili al di fuori dalla classe stessa e non possono essere istanziati., static quindi lavora con informazioni che devono essere comuni a tutte le istanze della classe. Una classe si dice static se contiene solo membri static.
4. Descrivere le modalità di implementazione di un evento in C#  
   L’evento mantiene una lista di Publisher e di Subscriber. Per poter ricevere le notifiche dal Publisher, il Subscriber dovrà essere aggiunto alla sua lista (subscribe).  
   Pertanto un evento necessita di un delegate che in genere ha come parametri un object (l’oggetto che scatena l’evento) e un’istanza di EventArgs (una struttura dati che contiene informazioni supplementari).
5. A cosa serve l’interfaccia IEnumerable<T>? Come si implementa in una nostra classe?  
   IEnumerable dispone di un metodo GetEnumerator() che ritorna un enumeratore, rende gli elementi ciclabili in una collezione.  
   Si implementa con 2 metodi GetEnumerator() che restituiranno un IEnumerable e un IEnumerable<T>

**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
  + ***OPZIONALE 3***: implementare IEnumerable
* 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