# Esercitazione di Fine Settimana – Week 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Deborah |
|  |  | Cognome | Tucci |
|  |  | Data | 02/04/2021 |

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 da un metodo in C# è possibile utilizzare nel return le tuple, oppure la keyword out tra i parametri oltre al valore di return, o ritornare una classe con tutti i valori necessari.

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

In C# esistono due tipologie di casting tra tipi: la conversione implicita e la conversione esplicita.

Nella conversione implicita non è richiesto un costrutto particolare ed è utilizzata per convertire tipi di più piccole dimensioni in quelli più grandi e non vi è perdita di dati. Esempio la conversione da una classe derivata alla classe base, o di un intero in un double.

int num = 234;

double number = num;

Mentre nella conversione esplicita è richiesto un costrutto di cast ed è utilizzata per convertire tipi di più grandi dimensioni in quelli di più piccole dimensioni, e in questo caso potrebbe esserci una perdita di informazioni, o il cast potrebbe non andar a buon fine. Esempio la conversione da una classe base a quella derivata, o di un tipo double in un intero. Esempio:

double number = 435.60;

int num = (int)number;

1. Quali sono gli utilizzi della keywork static?

La keywork static è utilizzata per dichiarare un membro statico, che non appartiene ad un oggetto specifico. Static può essere usato per dichiarare una classe statica, cioè una classe che non può essere istanziata. E per accedere ai membri di una classe statica si utilizza il nome della classe stessa, dato che non esiste una variabile di istanza.

Inoltre nelle classi è possibile utilizzare la keyword static a campi, metodi, proprietà, operatori per renderli indipendenti dagli oggetti della classe.

1. Descrivere le differenze tra le interfacce utilizzabili per le collection

Per le Collections è possibile utilizzare diverse interfacce quali ad esempio:

IEnumerable<T> che implementa l’interfaccia IEnumerator e definisce il metodo GetEnumerator, richiesto per il costrutto foreach.

ICollection<T>, che implementa una classe generica di collection e possiede diversi metodi e proprietà come il Count, Add, Remove e Clear.

IList<T>, che deriva dall’interfaccia ICollection<T> ed è utilizzata per implementare liste di elementi accessibili tramite indice e possiede dei metodi quali ad esempio Insert e RemoveAt, nei quali si specifica la posizione dell’elemento, e possono esserci dei duplicati.

IDictionary<TKey, TValue>, che implementa classi di collections generiche di elementi composti da chiave e valore, accessibili tramite la chiave

ISet<T>, che implementa gli insiemi, deriva dall’interfaccia ICollection<T>, non ammette duplicati e possiede ad esempio i metodi di unione ed intersezione tra due insiemi

1. Cos’è un Generic?

Un Generic è un costrutto di linguaggio che consente di scrivere classi, interfacce e metodi indipendenti dal tipo. È possibile scrivere un solo metodo, classe o interfaccia per tipi differenti. È compito del compilatore riconoscere il tipo da trattare.

**Esercitazione Pratica**

* Realizzare una classe Account per gestire un conto bancario, con le seguenti proprietà:
  + *Numero di Conto*
  + *Nome della Banca*
  + *Saldo*
  + *Data Ultima Operazione*
  + *Lista di Movimenti*
* Realizzare l’overload degli operatori + e – in modo che sia possibile aggiungere movimenti attivi e passivi alla lista dei movimenti (l’overload dovrà anche occuparsi di aggiornare il Saldo e la Data di Ultima Operazione)
* Realizzare un metodo Statement() che stampi i dati del conto, inclusa la lista dei movimenti
* Realizzare una gerarchia di classi per rappresentare movimenti bancari (Movement). Tutte le classi avranno le proprietà
  + *Importo*
  + *Data del Movimento*
  + Realizzare le classi che rappresentano:
    - CashMovement, con la proprietà aggiuntiva *Esecutore*
    - TransfertMovement, con le proprietà aggiuntive *Banca d’Origine* e *Banca Destinazione*
    - CreditCardMovement, con le proprietà aggiuntive *Tipo* (enum con i valori AMEX, VISA, MASTERCARD, OTHER) e *Numero di Carta*
  + 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
* Realizzare una Console app che
  + Crei un nuovo Account
  + Permetta di inserire diversi tipi di Movimenti (input dall'utente)
  + Stampi i dati del conto e i movimenti