# Esercitazione di Fine Settimana – Week 2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Graziella |
|  |  | Cognome | Caputo |
|  |  | Data | 14/05/2021 |

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#

In C# è possibile e a volte necessario restituire più valori da una funzione al metodo chiamante, lo si può fare sfruttando vari approcci e dichiarando come tipo di ritorno uno dei seguenti parametri:

* Un array
* Oggetto di tipo classe / struttura
* Un tipo Collections
* Una tupla

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

In C # sono disponibili due tipologie di casting:

* Casting implicito: consiste nell’effettuare una conversione che non comporta una perdita di informazione. Non è necessaria alcuna sintassi perché la conversione è indipendente dai tipi e i dati non vengono persi. Questo accade quando il tipo di variabile da cui copio può essere rappresentato nel nuovo tipo ad esempio un int può essere rappresentato come double o long ma non è possibile il contrario.
* Casting esplicito: consiste nell’effettuare una conversione manuale del dato se se si prevede una possibile perdita di informazioni durante la conversione oppure se la conversione non riesce per altri motivi.

1. Quali sono gli utilizzi della keywork static?

Normalmente i metodi della classe sono associati alle istanze dell’oggetto, nello specifico a quello su cui questi metodi vengono richiamati. Vi sono però delle circostanze in cui occorrono dei metodi legati direttamente alla classe, in questi casi è necessario specificare il modificatore di accesso “static”. Si usano i metodi statici ogni qualvolta occorra definire un metodo che agisce nel contesto della classe ma non necessariamente sugli oggetti della classe stessa. E’ possibile anche definire classi statiche nel caso in cui non sia necessario istanziare oggetti di quel tipo, o si vogliano incapsulare ad esempio metodi con soli parametri di input e output che non devono operare sugli oggetti.

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

Bisogna utilizzare la parola chiave "evento" con la variabile di tipo delegato per dichiarare un evento. Usare il delegato incorporato EventHandler o EventHandler <TEventArgs> per eventi costum. La classe publisher genera un evento e la classe subscriber si registra per un evento e fornisce il metodo gestore dell’ evento. Va assegnato un nome al metodo che genera un evento con il nome dell'evento. Una volta registrati a un evento con l'operatore += (o eventualmente annullata l'iscrizione utilizzando l'operatore - =) vanno passati i dati degli eventi utilizzando EventHandler <TEventArgs>.

1. Cos’è un Generic?

C # consente di definire classi, interfacce, classi astratte, campi, metodi, metodi statici, proprietà, eventi, delegati e operatori che siano idipendenti dal tipo di dato. Un tipo generico viene dichiarato specificando un parametro di tipo tra parentesi angolari dopo un nome di tipo, ad es. TypeName <T> dove T è un parametro di tipo, in questo modo si adatta l’istanza di ua variabile un oggetto o un metodo a qualsiasi tipo i dato rendendo il codice flessibile favorendone il riuso.

**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