Verifica Ilaria Bonelli 07/05/2021

1. VALUE TYPE: concetto di base di variabile e che viene usato estensivamente nella scrittura di software. Contengono direttamente il dato nell’ambito dello stack del thread. Una copia implica la copia dei dati in esso contenuti. Null non è ammessa per rappresentare i value type che comprendono i tipi primitivi: int, bool, double, ecc. anche gli Enum sono dei value type e le struct(definite dall’utente). Se istanzio un intero, il suo valore viene salvato direttamente dentro allo stack, dove è direttamente recuperabile. REFERENCE TYPE: contengono un riferimento ad un oggetto nell’ambito dell’heap; la copia implica la duplicazione del solo riferimento; le modifiche su due reference modificano l’oggetto a cui puntano. Il reference che non referenzia nessuna istanza vale null. Tutte le classi sono reference type, anche string, che è un tipo primitivo, però è una classe immutabile. Se istanzio l’oggetto di una classe sto realizzando un reference type che nello stack conterrà un puntatore all’oggetto, mentre l’oggetto vero e proprio è salvato nell’heap.
2. I Delegate Sono l’equivalente dei puntatori a funzione di c/c++. Una sorta di dichiarazione ad un metodo che può essere istanziato cioè una specie di via di mezzo tra una classe e una dichiarazione di metodo. Utilizzati per avere tante tipologie diverse di implementazioni di un metodo. Serve la parola chiave “delegate”, il tipo di ritorno, il nome del metodo e il parametro passato. Il compilatore crea di conseguenza una classe che deriva da System.Delegate oppure System.MulticastDelegate (solo il compilatore le può chiamare). Da programma il delegate viene istanziato passandogli nel costruttore il nome del metodo di cui si vuole creare il delegate. Il metodo istanza deve avere le stesse caratteristiche del delegato, stessi parametri e stesso tipo ritornato. Poi posso richiamarlo con “del(parametro)”.
3. Secondo la “Gang of Four”, il factory design pattern afferma che una factory è un oggetto che viene usato per creare altri oggetti. La factory dunque è una classe con un metodo che creerà e ritornerà tipi differenti di oggetti a seconda del parametro ricevuto in input. Nel factory pattern dunque si crea un oggetto senza esporre la logica che sta dietro alla sua realizzazione all’utente. Quindi se abbiamo una classe base e un certo numero di classi figlie, se vogliamo istanziare e ritornare l’oggetto di una delle classi figlie, dobbiamo usare il factory design pattern e quindi, a runtime, avremo un oggetto di tipo “simile sulla base dei parametri che gli passiamo.
4. Assert è una classe della libreria Xunit che fornisce una serie di metodi statici utili per scrivere dei test e per verificare il fallimento o il funzionamento di un test; con assert si verifica quindi che durante il testing ci siano le condizioni richieste per dare esito positivo. Serve in sostanze per definire una sorta di “prova del 9” su un metodo scritto da noi.