# Test di fine settimana – Week2

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Alessia |
|  |  | Cognome | Gualtieri |
|  |  | Data | 28/5/2021 |

Leggete attentamente ogni domanda e argomentare quanto più possibile fornendo anche degli esempi.  
ATTENZIONE: Le domande a risposta multipla possono contenere più risposte corrette.

1. ***Dare una definizione di classe statica***

*Una classe è statica quando ogni metodo che la compone, ogni proprietà e ogni campo è statico. In questo modo il valore assunto da ogni campo è uguale per tutti gli oggetti istanziati di tale classe, e se viene modificato il suo valore è cambiato in ogni oggetto di tale classe.*

*Un esempio di classe statica è Console.*

1. ***Quali tipologie di ereditarietà sono consentite in C# e come si definisce? Fornire un esempio***

*In C# sono consentite:*

*- l’ereditarietà semplice, cioè una superclasse che ha una sola sottoclasse.*

*Esempio: la classe ‘Figura’ con sottoclasse solo la classe ‘Poligoni’;*

*- l’ereditarietà multipla, una superclasse ha una sottoclasse e tale sottoclasse è superclasse di un’altra sottoclasse.*

*Esempio: la classe ‘Figura’ con sottoclasse la classe ‘Poligoni’, e la classe ‘Poligoni’ con sottoclasse la classe ‘Rettangolo’;*

*- l’ereditarietà polimorfa, cioè una superclasse ha più sottoclassi figlie tra loro allo stesso livello.*

*Esempio: la classe ‘Poligoni’ con sottoclassi le classi ‘Quadrato’, ‘Rettangolo’, ‘Triangolo’ tra loro non collegate.*

1. ***Elencare le principali caratteristiche della classe System.Object.***

*La classe Object è la superclasse di ogni altra classe, e viene sottintesa. Vengono ererditati anche tutti i suoi metodi, tra cui sono presenti dei metodi (come ToString, GetHashCode, Equals, Finalize... ) in modalità* ***virtual****, cioè che possono essere sovrascritte (****override****) dalle sottoclassi, in modo da essere personalizzate in base alla classe che si vuole creare.*

1. ***Descrivere le due fasi di gestione delle eccezioni.***

*Le eccezioni come prima cosa devono poter essere generate, e per farlo bisogna usare la keyword* ***throw*** *all’interno di un qualche metodo. Ciò permette, in caso di utilizzo del tale metodo, di poter generare l’eccezione specificata in caso di una qualche condizione soddisfatta. Tutte le eccezioni sono sottoclassi della più generica Exception.*

*Quindi quando si richiama un metodo che può generare una eccezione bisogna controllare che tale eccezione non blocchi l’esecuzione del programma, gestendola. Per farlo è necessario usare il comando* ***try catch (finally)****:*

*- all’interno del blocco try viene inserita la chiamata al metodo/ai metodi che possono richiamare una eccezione;*

*- nel blocco catch viene gestita l’eccezione in caso questa si verifichi (può essere la stampa di un messaggio di errore o qualsiasi altro comando). Possono essere presenti più eccezioni, preferibilmente da gestire in blocchi catch differenti per poter differenziare il comportamento in caso di ciascuna eccezione, tenendo conto di inserirli in ordine di specificità (dal più specifico al meno specifico)*

*- il blocco finally è opzionale e indica un pezzo di codice che viene eseguiso sia se l’eccezione non viene riscontrata sia se viene riscontrata.*

***Esercizio Pratico***

Creare una Console Application che gestisca i Task dell’utente.

Per Task viene inteso un oggetto che ha una descrizione, una data di scadenza e un livello di importanza (Basso, Medio, Alto).

L’utente può:

* Vedere i Task inseriti
* Aggiungere un nuovo Task
* Eliminare un Task
* Filtrare i Task per importanza

Requisiti Tecnici:

- Recuperare i task da file

- Salvare i Task in un file

- Utilizzare adeguatamente il concetto di classe

- Dividere le funzionalità in relative funzioni e procedure

- Commentare le scelte algoritmiche

- Mettere una nomenclatura conforme

- Verificare le date di scadenza dei task: devono essere posteriori o uguali rispetto alla data di inserimento

- Controllare l’input utente

Opzionale: Utilizzare Enum

Mettere il codice dell’esercizio in un Repository di GitHub.