## TestWeek6

1. ***Quali sono le possibili cause di un’eccezione?***

Le possibili cause di eccezione sono:

* *Errore di utilizzo* 🡪 rappresenta un errore nella logica del programma che può generare un’eccezione e può essere risolto modificando il codice che dà l’errore
* *Errore del programma* 🡪 è un errore di run – time che può essere evitato scrivendo codice privo di debug
* *Errore di sistema* 🡪 è un errore di run – time che non può essere gestito a livello di programmazione in modo significativo e di solito non vengono gestiti mediante la gestione di eccezione

1. ***Descrivere il costrutto formale per la gestione di una eccezione fornendo un esempio***

Il costrutto formale per la gestione delle eccezioni è il *try – catch:*

* Nel *try* vien inserito il codice che potrebbe causare un’eccezione
* Nel *catch* viene gestita l’eccezione

Potrebbe essere presente anche un blocco *finally* che viene eseguito sia nel caso che il codice vada o no in errore.

Esempio:

try

{

using (StreamWriter sw = new StreamWriter(path))

{

foreach (int value in myArray)

{

sw.WriteLine(value);

}

Console.WriteLine("Salvataggio dati completato!");

}

}

catch(FileNotFoundException fe)

{

Console.WriteLine("File non trovato!");

}

1. ***Descrivere i ruoli che esistono all’interno di un progetto***

Quando vogliamo realizzare un software sono 6 le figure principali che compongono il nostro team:

* *Product owner* 🡪 è il proprietario del prodotto finale
* *Business Analysts* 🡪 figura centrale tra owner del prodotto e team tecnico
* *Software/solution Architect* 🡪 progetta i sistemi, identifica il modo in cui gli sviluppatori dovrebbero costruire il sistema e si pone come guida per tutti gli aspetti tecnici relativi al progetto
* *Sviluppatori* 🡪 ereditano i disegni funzionali e tecnici redatti dagli analisti e dall’architect
* *Team Quality Assurance* 🡪 chiamato anche *Team Testing,* si assicura che il software scritto dagli sviluppatori corrisponda ai requisiti definiti in fase di analisi. Si assicura che il sistema risponda sia in termini di logica funzionale, sia in termini di performance e sicurezza.
* *Team Operations* 🡪 si occupa di prendere il software terminato e testato e predispone gli ambienti di esecuzione temporanea e finali occupandosi della configurazione di quelli che sono i requisiti tecnici di esecuzione.

1. ***Descrivere la differenza tra class diagram e object diagram descrivendo gli elementi di cui possono essere composti***

Il *Class Diagram* consiste in svariate classi connesse tra di loro tramite delle relazioni. La classe viene rappresentata da un rettangolo; un attributo rappresenta una proprietà di una classe e può avere un tipo e un valore di default. Un’operazione è un’azione che gli oggetti di una certa classe possono compiere. Associati poi ad attributi ed operazioni possono essere specificati dei constraint e delle note. Si possono applicare anche i marcatori di visibilità: public (+), private (-) e protected (#).

L’*object diagram* è invece un grafo che include oggetti e valori stato. Un diagramma di oggetti statici rappresenta le istanze delle classi descritte nel Class Diagram. L’utilizzo è abbastanza limitato ed è solitamente sfruttato per mostrare esempi del funzionamento del software in termini puramente statici.