# Modulo 7

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Carlotta |
|  |  | Cognome | Colla |
|  |  | Data | 12/03/2021 |

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. *Come avviene il lancio di un’eccezione a runtime ?*Il lancio di un’eccezione a runtine avviene attraverso “throw new” e il tipo dell’eccezione, questo genera una nuova eccezione. È possibile assegnare anche un messaggio all’eccezione, ad esempio: throw new FileNotFoundException(“Il file non è stato trovato”) oppure collegarla all’accezione che l’ha generata (ovvero quella catturata dal catch), es.  
   catch (exception exc) {  
   throw new FileNotFoundException(“il file non è stato trovato”, e) }  
   Questo rilancio dovrà essere catturato da un altro catch che avrà quel tipo di eccezione o un catch generico (con Exception exc).
2. *Quali sono le possibili cause di un’eccezione?*Le possibili cause di un’eccezione sono infinite, esisto quelle matematiche che generano eccezioni con le divisioni per zero, le eccezioni su lettura e scrittura di dati come FileNotFoundException, eccezioni sugli array come IndexOutOfRangeException... Le eccezioni possono essere generate sia dal compilatore sia dal programmatore attraverso la parola “throw new” e il tipo di eccezioni.
3. *Descrivere il costrutto formale per la gestione di una eccezione fornendo un esempio.*Il costrutto per gestire le eccezione è il try-catch (o try-catch-finally). Es  
   try { //operazioni che possono creare un’eccezione  
   int elemento = array[index]; }  
   catch (IndexOutOfRangeException exc) //tipo di eccezione che gestisce  
   {//gestisce l’eccezione  
   Console.WriteLine(“L’indice è fuore dal range”);}  
   Nel try si inserisce il codice che può generare un’eccezione e nel catch viene catturata e gestita.Ogni catch cattura solo un tipo di eccezione, se l’eccezione è diversa non viene catturata. La gerarchia dei catch deve essere dalla più specifica alla meno specifica (exception e) perchè l’eccezione verrà gestita dal primo catch compatibile che trova.
4. *Quali sono i possibili casi in cui è necessario lanciare un eccezione da codice? Fornire un esempio.*I casi in cui è necessario lanciare un’eccezione sono 3:  
   - Se un argomento verso un metodo genera un’eccezione: ad esempio viene passato un indice di un array non valido.  
   - Chiamata non appropriata ad un oggetto: ad esempio quando si prova a scrivere su un file di sola lettura  
   - Quando un metodo non può completare la sua funzionalità perchè viene passato un parametro non valido.ad esempio: una funzione prende come parametro una stringa su cui deve fare dei calcoli  
   public static void CalcolaStringa(string nome){  
   try {  
   if(nome == nul) { throw new ArgumentException(“Il nome è voto”); *}*catch (ArgumentException exc) {//la gestisco}
5. *Fornire una panoramica delle eccezioni più comuni e definire a cosa sono dovute.*Alcune delle eccezioni più comuni generate dal compilatore sono:  
   - IndexOutOfRange : quando si esce dal range di un array   
   - NullReferencesException : quando si cerca di fare dei calcoli su un oggetto nullo  
   - IOException: quando ci sono degli errori nel leggere/scrivere i dati   
   - DivideByZeroException: è un’eccezione matematica  
   - InvalidCastException: quando fallisce un cast tra due tipi diversi  
   - StackOverFlowException : quando lo stack delle chiamate è pieno
6. *Descrivere i ruoli che esistono all’interno di un progetto*All’interno di un progetto esistono 6 ruoli:  
   - Product owner : è il cliente che richiede il progetto, non ha quasi mai competenze informatiche ma conosce perfettamente le logiche di business. Spiega agli analisti come dovrebbe essere il progetto.  
   - Analisti: gestistono la comunicazione con il product owner, hanno sia competenze di business sia alcune competenze tecniche così da poter descrivere il progetto all’architetto.  
   - Architetto: ha grandi capacità tecniche, per questo è la figura incaricata a disegnare il progetto, attraverso diagrammi UML per poterli dare agli sviluppatori.  
   - Sviluppatori : sono coloro che creeranno il codice per realizzare il progetto; creando il codice possono aiutare anche ad arricchire il progetto dell’architetto, è uno dei ruoli più importanti.  
   - Team QA : chiamato anche gruppo di testing, il loro compito è qello di effettuare test sul codice creato dagli sviluppatori e vedere se risponde alle esigenze funzionali e logiche raccolte dagli analisti, se qualche requisito non viene soddisfatto il codice viene rimando agli sviluppatori per correggerlo, altrimenti viene mando alla fase finale.  
   - Team operation : questa è la fase finale, il nuovo software viene inserito e testato nell’ambiente finale e reso disponibile (se supera i test) agli utenti, questa fase si occupa anche della manutenzione nel corso del tempo.
7. *Dare una definizione di UML e spiegare perché si utilizza.*UML è un liguaggio per creare dei diagrammi, ogni diagramma ha il compito di evidenziare determinati aspetti che saranno utili ai diversi ruoli all’interno del progetto. Questo linguaggio è indipendente dal linguaggio di programmazione che verrà utilizzato per realizzare il progetto. Il suo scopo è quello di astrarre ad alto livello per migliorare la comunicazione tra i diversi ruoli, che possono avere diverse competente tecniche, e per rendere comprensibile a tutti concetti tecnici complicati. Il grande vantaggio di questi diagramma è il poter visualizzare come sarà la soluzione finale prima ancora di aver scritto il codice, questo permette di capire a priori se possono esserci dei problemi e corregerli, senza dover riscrivere intere parti di codice.  
   La stesura di questi diagrammi richiede del tempo che spesso il business owner reputa “sprecato”, in realtà, saltare questa fase porta a molta incoprensione e a un quasi sicuro ritardo sulla consegna o al fallimento di un progetto (consegna di un software diverso dal quello richiesto).
8. *Descrivere la differenza tra class diagram e object diagram descrivendo gli elementi di cui possono essere composti. Fornire un esempio.*Il class diagram è uno dei modelli più utilizzato, fornisce informazioni sulla struttura del progetto:quante e quali classi avrà, i loro campi e metodi e le relazioni tra esse. I suoi elementi sono le classi (classi concrete e astratte), interfacce e le relazoni che possono essere relazioni normali, derivazioni o implementazioni.  
   L’object diagram serve per visualizzare un esempio degli oggetti attivi in un determinato momento e le relazioni tra essi. Serve per mostrare il funzionamento del software, il suo utilizzo è abbastanza limitato.  
   I suoi elementi solo le classi attive con degli esempi dei loro campi e le relazioni tra esse.
9. *Fornire un esempio di use case diagram. Quali elementi sono in gioco e qual è l’obiettivo del diagramma?*

*(Non usare gli esempi presentati in classe)*Gli elementi di uno use case sono:  
- Attori: in uno use case è presente uno o più attori. L’attore principale, ovvero quello che compie l’azione è a sinistra del grafico, quello che subisce l’azione (se c’è) è a destra e possono essere presenti anche degli attori intermediari posti all’interno del grafico.  
- Casi d’uso: sono le azioni che possono compiere i vari attori, possono esserci azioni facoltative indicate con <<extend>> che l’attore può decidere se compierle o no.  
Ogni use case rappresenta uno scenario, ogni scenario può avere più azioni.  
L’obiettivo di questo diagramma è quello di visualizzare le interazioni tra i diversi attori in gioco. Questo porta a una raccolta più approfondita delle informazioni dal product owner perchè anche lui è in grado di leggerlo e migliorarlo.

Esempio di use case diagram: una palestra utilizza un software per registrare gli abbonamenti. È possibile creare un nuovo abbonamento, disdirlo, modificarlo e rinnovarlo.  
In questo scenario sono presenti due attori: il cliente che esegue l’azione e l’impiegato della palestra che la subisce.

Nel caso in cui si voglia creare un nuovo abbonamento non ci sono vincoli, una persona può avere più abbonamenti.  
Negli altri casi il cliente deve essere registrato e avere già un abbonamento.

*Esercitazione pratica*

*UML*

*Data un’applicazione che recuperi da un database i dati relativi alla gestione di una scuola, fare un object diagram, un class diagram e un use case diagram tenendo conto che:*

*Ogni classe presente nella scuola è composta da studenti. Un insegnante può impartire lezioni di più materie ed essere assegnato a più classi.*

*Uno studente inoltre può essere valutato in più occasioni per materia.*

*Uno studente è definito da Nome, cognome e data di nascita.*

*Una classe è definita da un codice e un livello.*

*Un insegnante ha un nome, cognome e una data di nascita. Può inserire una votazione a sistema indicando lo studente, la materia e il giorno della prova.*

*EXCEPTION HANDLING*

1. *Dimostrare come gestire un’eccezione data dalla mancata connessione a un database filtrando il tipo per errore fatale.*
2. *Creare una custom Exception relativa a un utente non trovato. Dimostrarne l’utilizzo.*