# Modulo 7

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Maria Chiara |
|  |  | 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 ?   
   Avviene attraverso il lancio di un eccezione con throw new e il tipo dell’eccezione. È possibile assegnare un messaggio o collegarlo all’eccezione che è stata catturata dal catch, es:   
   catch(exception e){  
    throw new DivideByZeroException(“Divisione per zero non permessa”, e)  
   }*
2. *Quali sono le possibili cause di un’eccezione?  
   Le possibili cause di un eccezione sono infinite, per esempio fornire un indice fuori dal range dell’array, crecare di leggere oggetti nulli, fare operazioni aritmetiche non valide, cercare di inserire dati in memoria quado la memoria è piena,   
   passare un parametro non valido, ecc*
3. *Descrivere il costrutto formale per la gestione di una eccezione fornendo un esempio.  
   Per gestire un eccezione è utilizzato il costruttore try catch:  
   - il try: contiene il codice che potrebbe causare un eccezione,  
   - il gestore catch: è il blocco di codice che gestisce una o più eccezioni, ogni eccezione viene propagata dai metodi finchè non trova il catch, non appena lo trova l’eccezione viene marcata come gestita e viene eseguito il blocco di codice all’interno del gestore catch.  
   - finally: blocco del codice da eseguire sia nel caso che vada o no nel catch.  
   esempio:  
   try{  
    int a=2;  
    int b=0;  
    int result = a/b;  
   } catch(DivideByZeroException e){  
    throw new DivideByZeroException(“Divisione per zero non permessa”, e)  
   }*
4. *Quali sono i possibili casi in cui è necessario lanciare un eccezione da codice? Fornire un esempio.  
   Per lanciare un eccezione vengono creati degli oggetti di tipo Exception tramite la parola throw, è necessario lanciare un eccezione quando: il metodo non può completare la funzionalità definita per esempio quando un parametro di input ha un valore non valido, quando viene eseguita una chiamata non appropriata a un oggetto in base allo stato dell’oggetto, per esempio un tentativo di scrittura in un file di sola lettura, quando un argomento verso un metodo genere un’eccezione in questo caso è necessario creare una istanza di argument exception, l’eccezione originale deve essere passara al costruttore di argument exception come parametro inner exception, per esempio un indice di un array non esistente in input.*
5. *Fornire una panoramica delle eccezioni più comuni e definire a cosa sono dovute.  
   Le eccezioni più comuni sono quelle lanciate dal compilatore, ovvero:  
   - ArithmeticException: eccezione durante un operazione aritmetica  
   - ArrayTypeMismatchException: quando un array non può salvare un dato perchè il suo tipo non è compatibile  
   - DivideByZeroException: tentativo di dividere per zero  
   - IndexOutOfRangeException: quando l’indice della matrice è minore di zero o superiore alla lunghezza dell’array  
   - InvalidCastException: quando un cast esplicito fallisce  
   - NullReferenceException: viene gerato quando si prova a leggere un oggetto il cui valore è null  
   - OutOfMemoryException: quando si tenta di allocare una parte della memerio con new quando la memoria è piena  
   - OverflowException: quando c’è un overflow in un operazione aritmetica  
   - StackOverflowException: quando viene esaurito lo stack di esecuzione da un numero eccessivo di chiamate in sospeso  
   - TypeInitializationException: quando un costruttore genere un eccezione*
6. *Descrivere i ruoli che esistono all’interno di un progetto  
   - product ower: è il proprietario del prodotto finale, quello che descrive cosa vuole,  
   - business analist: è un ruolo fondamentale perchè deve comprendere i bisogni del business e tradurli in linguaggio compressibile al team tecnico  
   - software architet: progetta il sistema, identifica il modo in cui gli sviluppatori dovrebbere costruire il sistema e si pone come guida per tutti gli aspetti tecnici, realizzo lo schema arrchitetturale della soluzione.  
   - sviluppatore: ricevono i disegni funzionali e tecnici dagli analisti e dall’architetto e li tasformano in modelli software eseguibili funzionanti, è il ruolo più importante e della catena il cui lavoro è tangibile a tutti.  
   - team QA: detto anche anche testing, si assicura che: il software prodotto dagli sviluppatori corrisponda ai requisiti tecnici definiti in fase di analisi, il sistema risponda ai termini di logica funzionale, performance e di sicurezza.  
   interagisce sempre con gli sviluppatori iterando il processo di costruzione/riparazione del sistema.  
   - team operations: riceve il software finito e testato e si occupa della configurazione dei requisiti tecnici di esecuzione e manutenzione nel corso del tempo.*
7. *Dare una definizione di UML e spiegare perché si utilizza.  
   UML è un vero e proprio linguaggio di modellazione dotato di sintassi chiara e non ambigua con regole sintattiche e semantiche, viene usato per capire e descrivere le caratteristiche di un nuovo progetto trascendendo le competenze tecniche di tutti quelli che partecipano al progetto.  
   Il progetto viene disegnato prima di scrivere il codice, questo permette di agevolare: la scrittura del codice e renderla più efficiente, la comunicazione tra tutte le figure coinvolti, e le possibili modifiche future.  
   Il linguaggio contiene molti elementi grafici che vengono messi insieme durante la creazione dei diagrammi, l’obbiettivo è quello di creare più viste di un unico progetto per studiarlo da diversi punti di vista, il totale dei diagrammi standar su UML è 13.*

|  |
| --- |
| *Scuola* |
| *nome: string* |

1. *Descrivere la differenza tra class diagram e object diagram descrivendo gli elementi di cui possono essere composti. Fornire un esempio.  
   Il class diagram consiste in molte classi connesse tra loro tramite delle relazioni, sono rappresentate da dei rettangoli con un nome con le iniziali di ogni parola maiuscole, per il quale è necessario ascoltare bene come il cliente chiama le varie entità coinvolte.  
   Per ogni classe si possono avere degli attributi che rappresetano le proprietà, descrivono un insieme di valori, compreso quello di default, che la proprietà può assumere quando un oggetto della classe viene istanziato, gli attributi si scrivono con tutte le iniziali di ogni parola maiuscola tranne la prima.  
   Si possono avere inoltre le operazioni, ovvero le azioni che un oggetto della classe può compiere, solitamente sono i verbi che il cliente utilizza per descrivere le azioni delle entità, possono avere zero o più parametri di input e zero o uno di parametri di output, ovvero il risultato dell’operazione.  
   Per le operazioni e gli attributi sono utilizzati dei marcatori di visibilità davanti al nome: + per definirli public, - privati e # protetti.  
   Il class diagram sfrutta l’ereditarietà, ovvero una classe figlia può ereditare gli attributi e le operazioni da una classe padre, viene rappresentata con una freccia che va dalla figlia verso il padre, viene usata quando 2 o più classi hanno un numero di attributi o operazioni comuni.  
   Le classi che non istanziano nessu tipo di oggetto sono dette astratte e si indicano con il nome in corsivo.  
   Le associazioni tra 2 o più classi si indicano con delle linee con un proprio nome e delle molteplicità ai lati delle classi a cui si riferisce, le classi giocano un ruolo all’interno delle classi a cui sono connesse.  
   L’object diagram è un diagramma che descrive gli oggetti istanziati del class daigram in un determinato contesto, serve per fornire un esempio pratico dell’esecuzione del sistema.  
   esempio: class Diagram*

\*..1

1

|  |
| --- |
| *Studente* |
| *nome: string* |

1. *Esempio: use case*

|  |
| --- |
| 1. *Studente* |
| 1. *nome: Pietro rossi* |

|  |
| --- |
| 1. *Scuola* |
| 1. *nome: Macedonio Melloni* |

1. *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)  
Nascono per dare un valore aggiunto alla raccolta di informazioni, facilitano la comunicazione con il cliente finale e tra le persone del team di sviluppo.  
Ogni use case è una lista di scenari ed ogni scenario è una sequenza di passi, in ognuno è presente:  
 - un attore: colui che inizia la sequenza, rappresentato da un omini stilizzato situato a sinistra  
 - ci sono delle possibili pre-condizioni, ovvero i requisiti di esecuzione  
 - i passi dello scenario, rappresentati con degli ellissi in cui all’interno è scritta l’azione che può essere eseguita  
 - le possibili post-condizioni dello scenario completo  
 - in fine l’attore che beneficia dello use case, rappresentato anche lui da un omino stilizzato, che potrebbe essere lo stesso che lo inizia o un altro, in quel caso è posto a destra.  
Sono tutti connessi da delle relazioni, rappresentate con una linea continua.  
Un esempio di use case daigram: un cliente quando usa l’applicazione può chiedere al dottore un’ appuntamento al dottore o ppure la prescrizione di ricette mediche, il dottore una volta ricevute le richieste le manderà al cliente:*

*Diagram

Description automatically generated*

*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. (in un modo l’errore fatale e in un altro quelli generici)*
2. *Creare una custom Exception relativa a un utente non trovato. Dimostrarne l’utilizzo.*