# Modulo 7

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Nome | Deborah |
|  |  | Cognome | Tucci |
|  |  | 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?*

A runtime, quando viene generata un’eccezione, si associa il tipo di eccezione generata e ripercorrendo lo stack delle chiamate si ricerca un gestore dell’errore dello stesso tipo. Se esiste un gestore nello stack, esso prende in carico l’eccezione e si esegue il codice contenuto nel blocco catch, che è il gestore. Mentre, se non esiste un gestore nello stack, l’eccezione comporta l’interruzione del programma e chiusura dell’applicazione

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

Le eccezioni sono errori generati da una situazione imprevista che si verifica durante l’esecuzione di un programma. Le possibili cause possono essere individuate in errori di codice derivati da sviste dello sviluppatore, ma anche da altre problematiche come da un’autorizzazione fallita e non gestita, da un problema a livello di hardware, o da una chiamata ad una risorsa non disponibile al momento.

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

La gestione di una eccezione può avvenire creando un apposito gestore mediante il costrutto formale try-catch, dove nel try: si inserisce il codice che potrebbe causare un’eccezione, nel catch, che è il blocco gestore dell’eccezione, dunque contiene il codice che gestisce l’errore. Mentre finally è il blocco di codice da eseguire sia nel caso che il programma vada o non vada in errore.

Esempio:

try {

var risultato = 5/0;

Console.WriteLine("Il risultato è {0}", risultato);

} catch (Exception e)

{

Console.WriteLine("Errore!! " + e.Message);

}

finally

{

Console.WriteLine("Arrivederci! ");

}

1. *Quali sono i possibili casi in cui è necessario lanciare un eccezione da codice? Fornire un esempio.*

A volte è necessario lanciare da codice un’eccezione, i casi sono tre:

quando il metodo non può completare la funzionalità definita;

quando viene eseguita una chiamata non idonea a un oggetto, in base allo stato dell’oggetto, come ad esempio quando si tenta di scrivere su un file che è di sola lettura.

Infine, quando un argomento in un metodo genera un’eccezione.

Esempio:

public static void Incremento(int num)

{

if(num != 0){

Var inc = num + 2;

Console.WriteLine(inc);

}

else{

throw new ArgumentException("il numero non può essere zero");

}

1. *Fornire una panoramica delle eccezioni più comuni e definire a cosa sono dovute.*

Tra le eccezioni più comuni nel compilatore abbiamo:

ArithmeticException che è la classe di base per tutte le eccezioni che si possono verificare durante le operazioni aritmetiche quali ad esempio

la DivideByZeroException, che viene generata quando si esegue un tentativo di divisione di un valore intero per zero;

e la OverflowException, generata quando si verifica quando il programma non riesce ad uscire dall’esecuzione di un’operazione.

Inoltre, c’è la IndexOutOfRangeException generata quando viene eseguito un tentativo di indicizzare un array, e l’indice supera i limiti del array.

La OutOfMemoryException viene generata quando c’è un tentativo di allocazione della memoria, ma la memoria è esaurita.

L’InvalidCastException, generata quando si esegue una conversione esplicita dal tipo di base a un tipo derivato e questa conversione non va a buon fine in fase di runtime.

La StackOverflowException, infine, generata quando viene esaurito lo stack di esecuzione da un numero eccessivo di chiamate in sospeso verso il metodo; in genere si verifica in caso di ricorsione mal gestita.

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

I ruoli che esistono all’interno di un progetto sono:

Product Owner che è il proprietario del prodotto finale, il committente, a volte anche utente utilizzatore, che fornisce le caratteristiche principali che deve possedere il prodotto.

Business Analyst, cioè colui che si pone come intermediario tra gli owner e il team tecnico. Dunque deve comprendere i bisogni dell’owner e aver un minimo di competenza tecnica per poterli comunicare al team.

Software Architect, cioè colui che progetta i sistemi, identifica il modo in cui gli sviluppatori dovrebbero realizzare il sistema, e si pone come guida tecnica per il progetto.

Sviluppatori, che ereditano i disegni funzionali e tecnici redatti dagli analisti e dall’architect e li trasformano in modelli software eseguibili e in una soluzione codice funzionante.

Team Quality Assurance, detto anche Team di Testing, assicura che il software scritto dal team di sviluppo corrisponda ai requisiti definiti in fase di analisi, sia in termini di logica funzionale, sia in termini di performance e di sicurezza.

Team Operations che prende il software finito e testato e predispone gli ambienti di Staging, cioè esecuzione temporanea di test e ambienti di Produzione finali.

1. *Dare una definizione di UML e spiegare perché si utilizza.*

L’UML è l’acronimo di Unified Modeling Language, uno linguaggio di modellazione standard accettato universalmente per l’analisi e design di sistemi complessi. È un linguaggio semi formale e grafico cioè basato su diagrammi per specificare, visualizzare, realizzare, modificare, documentare le caratteristiche un sistema complesso, durante il suo sviluppo da diversi punti di vista.

Viene utilizzato l’UML perché appunto offre un approccio standardizzato per lo scambio di informazioni tra i team di sviluppo software con differente livello tecnico, agevolando la comunicazione. Inoltre, i diagrammi UML sono più facili da consultare rispetto all’analisi dell’intero codice sorgente.

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

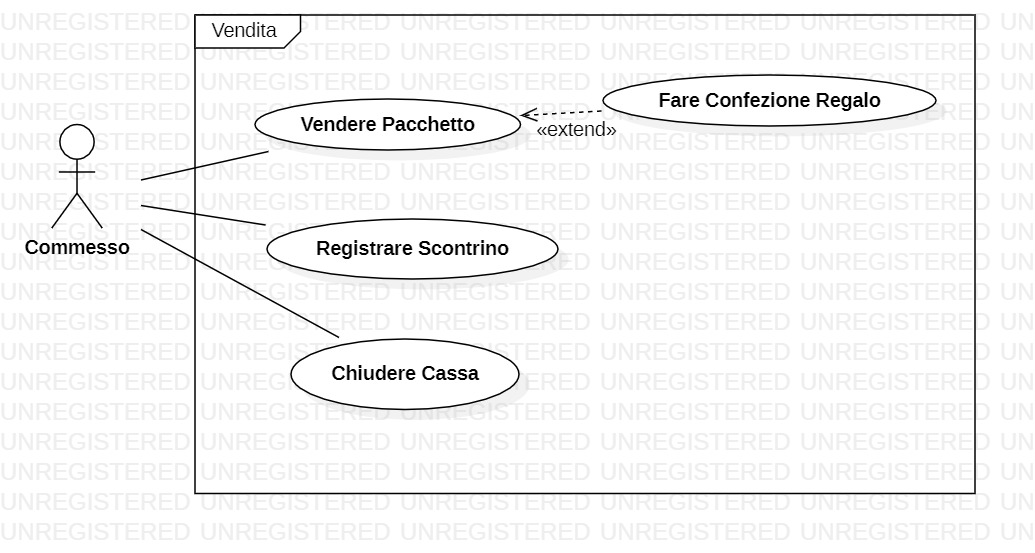
Il Class Diagram è un diagramma di classificatori di elementi connessi tra loro sulla base di relazioni statiche. Questo diagramma contiene oltre alle classi anche le interfacce, astrazioni, associazioni tra elementi. Mentre, l’Object Diagram è un grafo che include oggetti e valori di stato, dunque oggetti concreti. Un diagramma di oggetti statici rappresenta le istanze effettive di classi descritte nel class diagram nel contesto di un preciso momento della sua esecuzione.

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)*

Lo Use Case Diagram ha l’obiettivo di fornire una rappresentazione grafica che mette in evidenza le funzionalità che il sistema dovrà avere ad alto livello, dunque i requisiti funzionali e gli attori che interagiscono con il sistema. Quindi, rappresenta uno scenario, cioè una sequenza di eventi che viene iniziata da una persona o da un altro sistema, definiti attori. Quindi gli elementi in gioco sono gli attori che iniziano o subiscono l’azione, gli use case che rappresentano i requisiti funzionali e le associazioni tra di essi.

Nell’esempio gli use case, dunque le funzionalità del prodotto sono vendere pacchetto che può essere esteso dallo scenario fare confezione regalo; poi vi è registrare scontrino e chiudere cassa. Mentre l’attore è il commesso che avvia queste azioni.



*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.*