Set Domande

INGEGNERIA DEL SOFTWARE

INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE (D.M. 270/04)

Docente: *Vetrella Sergio*

Generato il	21/02/2022 18:53:25
N° Domande Aperte	166
N° Domande Chiuse	190

Indice

Lezione 002

01. Una delle seguenti affermazioni, relative al concetto di modello, non è corretta. Quale?

☐ Lo sviluppo di un sistema software richiede di identificare e descrivere il sistema come un insieme di modelli che affrontano i problemi del progettista.

☐ I metodi orientati agli oggetti combinano il modello dell'applicazione e quello delle soluzioni.

☐ Il modello delle soluzioni è il risultato di una trasformazione del modello dell'applicazione.

☐ Il modello dell'applicazione viene inizialmente costruito come un insieme di oggetti e relazioni, che rappresentano i concetti del mondo reale che devono essere manipolate.

02. Una delle affermazioni che seguono non è corretta. Quale?

☐ Un prodotto può essere un sistema, un modello o un documento.

☐ Una risorsa può essere un partecipante, un periodo di tempo o della attrezzatura.

☐ Un compito consuma risorse e produce un prodotto.

☐ Un progetto di sviluppo di un sistema software è articolato in compiti (task), e ogni compito è composto da un certo numero di attività.

03. Che significato assume il termine ruolo nel contesto di un progetto di sviluppo software?

☐ Con il termine ruolo indichiamo il singolo compito assegnato ad un utente in fase di elicitazione dei requisiti.

☐ Con il termine ruolo indichiamo l'elenco ordinato delle procedure che regolano lo sviluppo del software.

☐ Con il termine ruolo indichiamo l'insieme dei casi di test assegnati ad un utente pilota in fase di validazione del sistema.

☐ Con il termine ruolo indichiamo un insieme di responsabilità nel progetto del sistema.

04. Che cos'è un deliverable?

☐ E' un semilavorato ad uso interno da parte di altri sviluppatori.

☐ E' un modello assimilabile ad un design pattern.

☐ E' una componente sotware riusabile.

☐ E' un documento o un prototipo da sottoporre al committente.

05. Una delle seguenti affermazioni, relative al concetto di modello, non è corretta. Quale?

☐ Un modello è utile quando le dimensioni del sistema, o il suo livello di complessità, o ancora problemi di accessibilità e sicurezza lo rendono difficilmente manipolabile in modo diretto.

☐ Un modello è una rappresentazione astratta di un sistema.

☐ Il modello di un sistema software include fin dalle prime fasi di sviluppo la specifica del dominio delle soluzioni, che spiega come dominare la complessità delle possibili soluzioni.

☐ I progettisti di un sistema software costruiscono il modello di un sistema che ancora non esiste.

06. Che cos'è, nel contesto dell'Ingegneria del Software, una metodologia? In che cosa si differenzia da un metodo?

07. Che cos'è, nel contesto dell'Ingegneria del Software, una notazione? Fornisci un esempio.

Lezione 003

01. In merito all'attività di progettazione del sistema, una delle affermazioni seguenti non è corretta. Quale?

- [ ] Durante il system design gli sviluppatori definiscono gli obiettivi del design e decompongono il sistema in sottosistemi più piccoli.
- [ ] Durante il system design vengono selezionati il database management system e il controllo di flusso globale.
- [ ] Durante il system design vengono selezionate la piattaforma hw/sw e le politiche di accesso.
- [ ] Per consentire agli utenti di contribuire alla validazione del modello a oggetti del system design, questo viene rappresentato in una notazione a loro facilmente comprensibile.

02. In merito all'attività di progettazione degli oggetti, una delle affermazioni seguenti non è corretta. Quale?

- [ ] Durante l'object design gli sviluppatori definiscono gli oggetti del dominio delle soluzioni.
- [ ] L'attività di object design include la selezione di componenti riusabili.
- [ ] L'attività di object design include la descrizione precisa degli oggetti e delle interfacce dei sottosistemi.
- [ ] La fase di object design ha lo scopo di colmare la distanza tra il modello dell'analisi e la piattaforma hw/sw identificata nella fase di elicitazione dei requisiti.

03. In merito all'attività di elicitazione dei requisiti, una delle affermazioni seguenti non è corretta. Quale?

- [ ] Le tipologie di attore sono tre: l'utente, il committente e lo sviluppatore.
- [ ] Il risultato dell'attività di elicitazione dei requisiti è una descrizione del sistema in termini di attori e casi d'uso.
- [ ] I casi d'uso sono sequenze generali di eventi che descrivono tutte le possibili interazioni col sistema per una data funzionalità.
- [ ] Durante l'elicitazione dei requisiti il cliente e gli sviluppatori definiscono lo scopo del sistema.

04. In merito all'attività di analisi dei requisiti, una delle affermazioni seguenti non è corretta. Quale?

- [ ] I casi d'uso prodotti nella fase di elicitazione dei requisiti vengono tradotti in un modello relazionale che descrive la struttura delle classi del sistema, ma trascura per il momento la definizione dell'interfaccia utente.
- [ ] Durante l'analisi gli sviluppatori producono un modello del sistema che sia corretto, completo, consistente e non ambiguo.
- [ ] Il risultato è un modello del sistema annotato con attributi, operazioni e associazioni, e descritto in termini sia strutturali che dinamici.
- [ ] Eventuali ambiguità e inconsistenze nei casi d'uso vengono risolte con la collaborazione dell'utente.

05. Il ruolo di manager di un progetto di sviluppo software richiede lo svolgimento di numerose attività. Quale, tra quelle elencate di seguito, è considerata la più critica, e richiede in genere il maggiore investimento di tempo?

- [ ] La gestione del rationale.
- [ ] Il monitoraggio dello stato di avanzamento del progetto.
- [ ] La gestione della configurazione software.
- [ ] Il coordinamento delle comunicazioni tra i partecipanti del gruppo di lavoro.

06. In merito all'attività di validazione (testing), una delle affermazioni seguenti non è corretta. Quale?

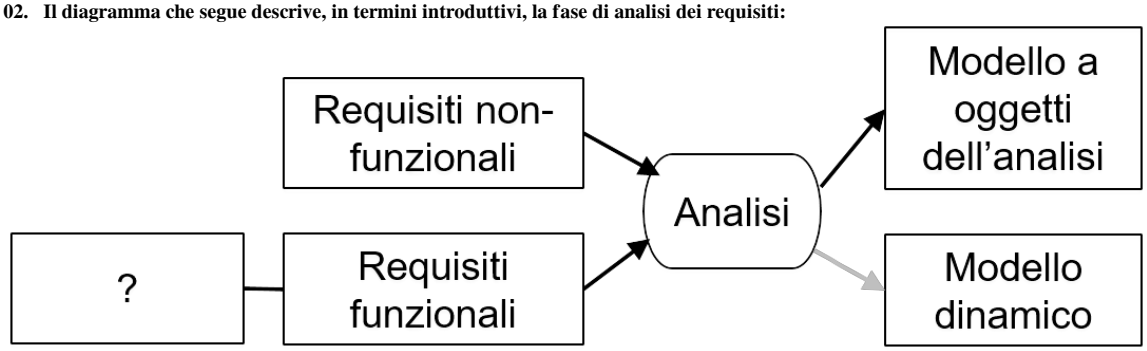
- [ ] Durante i test di integrazione i sottosistemi sono aggregati e confrontati con il modello del system design.
- [ ] Durante lo unit testing gli sviluppatori confrontano il modello dell'object design con ogni oggetto e sottosistema.
- [ ] Durante il system testing il sistema viene eseguito con casi tipici ed eccezioni e confrontato col modello dei requisiti.
- [ ] Durante il testing gli utenti cercano differenze tra il sistema e i suoi modelli, eseguendo il sistema con campioni di dati di output.

Lezione 004

01. Nel seguito sono elencate alcune possibili caratteristiche della notazione adottata in un progetto. Solo tre di queste caratteristiche sono indicate, nella lezione 4, come indispensabili. Qual è l'intrusa?

- [ ] La notazione deve essere ben compresa dai partecipanti al progetto.
- [ ] La notazione deve essere adeguata a rappresentare aspetti specifici del sistema.
- [ ] La notazione deve avere una semantica ben definita.
- [ ] La notazione deve essere direttamente traducibile nel linguaggio di programmazione adottato nel progetto.

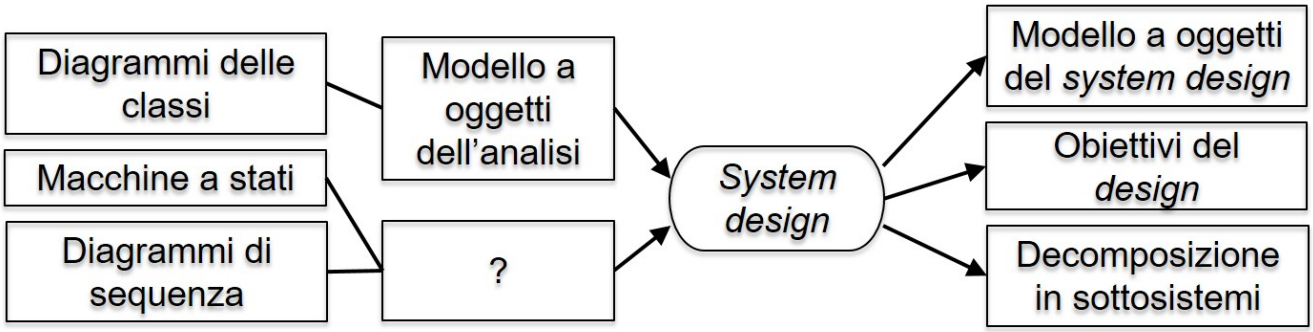
02. Il diagramma che segue descrive, in termini introduttivi, la fase di analisi dei requisiti:



Che etichetta deve essere specificata nel box con il punto interrogativo?

- [ ] Casi di test
- [ ] Modello delle classi
- [ ] Casi d'uso
- [ ] Modello dinamico

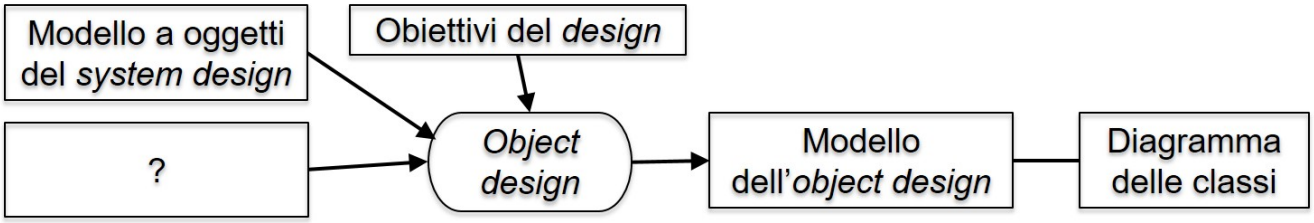
03. Il diagramma seguente descrive in termini introduttivi la fase di system design:



Che etichetta deve essere inserita nel box con il punto interrogativo?

- [ ] Modello dinamico
- [ ] Elicitazione dei requisiti
- [ ] Casi di test
- [ ] Casi d'uso

04. Il diagramma che segue descrive in termini introduttivi la fase di object design:



Che etichetta deve essere inserita nel box con il punto interrogativo?

- [ ] Casi di test
- [ ] Modello dinamico
- [ ] Casi d'uso
- [ ] Decomposizione in sottosistemi

05. In merito al modello dinamico, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I diagrammi di interazione descrivono il comportamento interno del sistema come una sequenza di messaggi scambiati tra un insieme di oggetti.
- [ ] I diagrammi di attività descrivono il comportamento in termini di flussi di controllo e di flussi dei dati.
- [ ] I diagrammi di sequenza descrivono l'ordine in cui i processi paralleli devono essere serializzati.
- [ ] I diagrammi di stato descrivono il comportamento nei termini degli stati che oggetti individuali possono assumere, e delle transizioni tra tali stati.

06. In merito ai diagrammi UML dei casi d'uso, una delle affermazioni che seguono non è corretta. Quale?

- [ ] L'identificazione di attori e casi d'uso ha come effetto la delimitazione dei confini del sistema.
- [ ] I casi d'uso sono utili nelle fasi di elicitazione dei requisiti e di analisi per rappresentare le funzionalità del sistema. Essi mettono a fuoco il comportamento del sistema dal punto di vista dello sviluppatore.
- [ ] Un attore descrive un'entità che interagisce col sistema: un utente, un altro sistema, l'ambiente fisico in cui il sistema si trova ecc.
- [ ] Un caso d'uso descrive una funzione che, offerta dal sistema, produce un risultato visibile per un attore.

07. In merito ai diagrammi UML delle classi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I diagrammi delle classi descrivono il sistema in termini di oggetti, classi, attributi, operazioni e associazioni.
- [ ] Un oggetto è dotato di uno stato, che include i valori dei suoi attributi e i collegamenti (link) con altri oggetti.
- [ ] Gli oggetti sono astrazioni che specificano la struttura e il comportamento in comune ad un insieme di classi.
- [ ] I diagrammi delle classi sono usati per descrivere la struttura di un sistema.

08. In merito ai diagrammi UML delle interazioni, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I diagrammi delle interazioni consentono di identificare ulteriori oggetti partecipanti nel caso d'uso.
- [ ] I diagrammi delle interazioni consentono di formalizzare il comportamento dinamico del sistema.
- [ ] I diagrammi delle interazioni consentono di visualizzare le comunicazioni tra gli oggetti.
- [ ] I diagrammi delle interazioni consentono di modellare le caratteristiche grafiche degli oggetti boundary.

09. In merito ai diagrammi UML degli stati, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I diagrammi degli stati descrivono il comportamento dinamico di un singolo oggetto.
- [ ] Dato uno stato, una transizione identifica uno stato futuro in cui l'oggetto può portarsi.
- [ ] Uno stato rappresenta un particolare insieme di valori che gli attributi dell'oggetto possono assumere.
- [ ] I diagrammi degli stati sono rappresentati mediante grafi diretti aciclici (DAG).

10. In merito ai diagrammi UML delle attività, una delle affermazioni che seguono non è corretta. Quale?

- [ ] L'esecuzione di un'attività può essere innescata da un evento esterno.
- [ ] L'esecuzione di un'attività può essere innescata dalla sopravvenuta disponibilità di un oggetto.
- [ ] Un diagramma di attività descrive il comportamento di un utente in base ai compiti che lo stesso utente affronta.
- [ ] L'esecuzione di un'attività può essere innescata dal completamento di un'altra attività.

11. Quale modello, tra quelli elencati nel seguito, non appartiene all'insieme dei modelli che UML si propone di rappresentare?

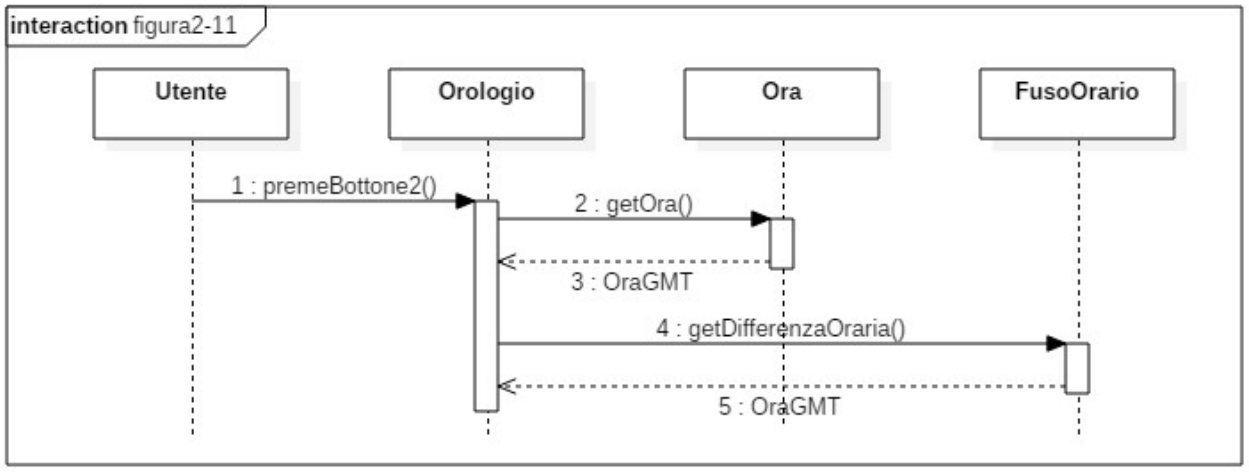
- [ ] Il modello funzionale
- [ ] Il modello dinamico
- [ ] Il modello degli oggetti
- [ ] Il modello costi-benefici

Lezione 005

01. In merito al concetto di operazione, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Le operazioni sono definite nell'ambito di una classe, e possono essere applicate a tutte le istanze di quella classe.
- [ ] Le operazioni definite in una sottoclasse possono essere ereditate e applicate alle istanze dirette della superclasse.
- [ ] E' comunque possibile ridefinire in una sottoclasse un'operazione già definita in una superclasse.
- [ ] Le operazioni definite in una superclasse sono automaticamente ereditate e applicabili alle istanze delle sottoclassi.

02. Considera il diagramma mostrato qui sotto:



Tra le affermazioni che seguono, una non è corretta. Quale?

- [ ] OraGMT è un'operazione.
- [ ] getOra() è un messaggio.
- [ ] OraGMT è un messaggio.
- [ ] getDifferenzaOraria() è un messaggio.

03. In merito al concetto di attributo, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Gli attributi di un oggetto sono sempre accessibili in lettura ad altre parti del sistema.
- [ ] Gli attributi di un oggetto possono essere accessibili ad altre parti del sistema.
- [ ] Un attributo è dotato di un nome univoco e di un tipo.
- [ ] Una classe definisce gli attributi di cui sono dotate tutte le sue istanze.

04. In merito al concetto di prototipo, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Ogni prototipo viene modificato in base alle indicazioni emerse dagli utenti.
- [ ] Ogni prototipo viene presentato ad un potenziale utente perché lo falsifichi.
- [ ] Gli sviluppatori costruiscono vari prototipi rappresentativi del comportamento finale del sistema.
- [ ] Per poter essere consegnato all'utente, un prototipo deve prima essere sottoposto a determinati controlli di qualità e completezza.

05. In merito al concetto di classe astratta, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Una classe astratta modella attributi e operazioni condivise da più sotto-classi.
- [ ] Una classe astratta può essere istanziata al più una volta (oggetto singleton)
- [ ] In UML, le classi astratte sono rappresentate con il nome in corsivo.
- [ ] Le classi astratte spesso rappresentano concetti generalizzati nel dominio dell'applicazione.

06. In merito al concetto di classe nella modellizzazione orientata agli oggetti e nei linguaggi di programmazione OO, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Una classe è un'astrazione.
- [ ] Una classe incapsula sia la struttura che il comportamento.
- [ ] Una classe può essere istanziata da numerosi oggetti.
- [ ] Una classe non può essere definita in modo incrementale rispetto ad un'altra classe preesistente.

07. Che cosa è un oggetto?

- [ ] Un'entità che incapsula stato e comportamento
- [ ] Una collezione di classi che condividono la stessa struttura
- [ ] Una collezione di classi che condividono lo stesso comportamento
- [ ] Un'astrazione che specifica gli attributi e i comportamenti di un insieme di classi

08. In merito al concetto di sistema, una delle affermazioni che seguono non è corretta. Quale?

- [ ] La decomposizione termina al livello delle singole operazioni, quando le dimensioni del sottosistema sono diventate sufficientemente semplici da renderlo implementabile con una singola istruzione del linguaggio di programmazione.
- [ ] La decomposizione di un sistema in sottosistemi può essere ricorsivamente applicata ai sottosistemi.
- [ ] Alcune parti di un sistema possono a loro volta essere considerate sottosistemi.
- [ ] Un sistema è un insieme organizzato di parti tra loro comunicanti.

09. In merito al concetto di oggetto, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Un oggetto può essere istanza diretta di due o più classi distinte.
- [ ] In UML, un oggetto è rappresentato come un rettangolo col nome sottolineato.
- [ ] Un oggetto è dotato di identità.
- [ ] Un oggetto memorizza i valori dei propri attributi.

10. In merito al concetto di vista (view), una delle affermazioni che seguono non è corretta. Quale?

- [ ] Una vista è la rappresentazione di un sottoinsieme di un modello.
- [ ] Una vista evidenzia gli aspetti rilevanti di un modello secondo una determinata prospettiva.
- [ ] Due viste su uno stesso modello non possono sovrapporsi, neppure parzialmente: devono essere sempre mutuamente esclusive.
- [ ] Una vista ha lo scopo di migliorare la comprensibilità di un modello complesso.

11. In merito al concetto di tipo di dato, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Un tipo di dato è un'astrazione.
- [ ] Un tipo di dato ha un nome univoco.
- [ ] Un tipo di dato può definire o una struttura, oppure un insieme di operazioni, ma non entrambe le cose.
- [ ] Un tipo di dato denota un insieme di valori.

Lezione 006

01. In merito alle differenze tra il concetto di scenario e quello di caso d'uso, solo una delle seguenti affermazioni è corretta. Quale?

- [ ] Un caso d'uso è un'istanza particolare di uno scenario
- [ ] Sia i casi d'uso che gli scenari sono istanze di un diagramma delle classi UML
- [ ] Uno scenario è un'istanza particolare di un caso d'uso
- [ ] Sono la stessa cosa: i concetti di “scenario” e di “caso d'uso” sono sinonimi in Ingegneria del Software

02. Quale, tra le entità sotto elencate, in genere non fa parte della descrizione testuale di un caso d'uso?

- [ ] Gli attori partecipanti
- [ ] Le condizioni d'ingresso e di uscita
- [ ] Il flusso degli eventi
- [ ] Il diagramma degli stati

03. Quale, tra le entità sotto riportate, non è uno dei tipi di relazione che i casi d'uso possono includere?

- [ ] Comunicazione
- [ ] Inclusione
- [ ] Esclusione
- [ ] Estensione

04. In merito alle relazioni che i casi d'uso possono includere, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Le relazioni di comunicazione associano un attore ad un caso d'uso.
- [ ] Una relazione di estensione indica che un'istanza di un caso d'uso «estensore» può includere il comportamento specificato dal caso d'uso «esteso».
- [ ] Due casi d'uso sono in relazione di inclusione se uno dei due include l'altro nel suo flusso degli eventi.
- [ ] Mediante una relazione di ereditarietà un caso d'uso può specializzare un altro caso d'uso più generale, aggiungendo dettagli.

05. In merito al concetto di attore nei diagrammi dei casi d'uso UML, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I diagrammi dei casi d'uso descrivono il comportamento del sistema dal punto di vista degli attori.
- [ ] Gli attori sono rappresentati nei diagrammi dei casi d'uso da questo simbolo:

- [ ] Ogni attore è dotato di nome univoco e di descrizione.
- [ ] Un database può essere un attore.

Lezione 007

01. In merito al concetto di aggregazione nei diagrammi UML delle classi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Un'associazione è in caso particolare di aggregazione.
- [ ] Un'aggregazione denota un aspetto gerarchico della relazione.
- [ ] Un'aggregazione è denotata da una linea con un rombo all'estremità del contenitore.
- [ ] La molteplicità di un'aggregazione piò essere uno-a-molti oppure molti-a-molti.

02. In merito all'uso dei diagrammi UML delle classi in fase di analisi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Lo scopo dei modelli di analisi è descrivere la «zona di copertura» del sistema, ed esplicitarne i confini.
- [ ] Durante l'analisi gli sviluppatori costruiscono diagrammi delle classi per catturare e formalizzare la conoscenza relativa al dominio delle soluzioni.
- [ ] Le classi rappresentano oggetti che partecipano ai diagrammi delle interazioni, descrivendne gli attributi e le operazioni.
- [ ] Le classi rappresentano oggetti che partecipano ai casi d'uso, descrivendone gli attributi e le operazioni.

03. In merito alla specifica del comportamento in un diagramma UML delle classi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il comportamento degli oggetti è specificato dalle operazioni.
- [ ] Un oggetto richiede l'esecuzione di un'operazione da parte di un altro oggetto mandandogli un messaggio.
- [ ] In un linguaggio OO i metodi di una classe sono l'interfaccia delle operazioni che ne definiscono il comportamento.
- [ ] La distinzione tra operazioni e metodi ci permette di distinguere tra la specifica del comportamento (cioè un'operazione) e la sua implementazione (cioè un insieme di metodi che possono anche essere definiti in classi diverse della gerarchia di ereditarietà)

04. In merito al concetto di ereditarietà nei diagrammi UML delle classi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Le super-classi astratte si distinguono perché il loro nome è scritto in grassetto.
- [ ] Le super-classi astratte consentono di diminuire la complessità_x000B_del modello.
- [ ] L'ereditarietà è la relazione tra una classe generale e una o più classi specializzate.
- [ ] L'ereditarietà consente di concentrare in un unico luogo la descrizione di tutti gli attributi e le operazioni che sono in comune a un insieme di classi.

05. In un diagramma UML delle classi, che cos'è un ruolo?

- [ ] E' un'etichetta posta ad un estremo di un'associazione.
- [ ] E' il nome, univoco, con cui si identifica l'utente che può accedere alla classe.
- [ ] E' il nome, univoco, con cui un'associazione viene identificata rispetto alle altre associazioni del diagramma.
- [ ] E' la cardinalità di un'associazione.

06. In merito al concetto di molteplicità di un'associazione, una delle affermazioni che seguono non è corretta. Quale?

- [ ] La molteplicità di un'associazione bidirezionale ha sempre valore maggiore o uguale a 1.
- [ ] La molteplicità dell'estremo di un'associazione è il numero di link che possono originare da un'istaza della classe connessa all'associazione.
- [ ] La molteplicità "molti" si rappresenta con un asterisco "*".
- [ ] Un'associazione uno-a-molti denota tipicamente una composizione.

07. In merito alle associazioni in un diagramma delle classi UML, solo una delle seguenti affermazioni è corretta. Quale?

- [ ] Le associazioni possono essere sia asimmetriche che simmetriche; però solo quelle asimmetriche sono navigabili.
- [ ] Le associazioni possono essere sia asimmetriche che simmetriche; quelle asimmetriche sono navigabili solo in una direzione.
- [ ] Le associazioni sono sempre asimmetriche (unidirezionali, orientate).
- [ ] Le associazioni sono sempre simmetriche (bidirezionali).

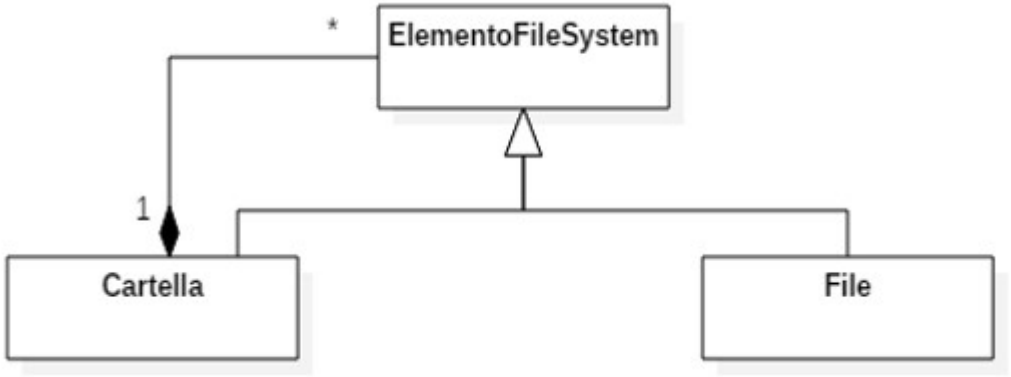
08. In UML le classi e gli oggetti sono rappresentati come scatole composte da tre compartimenti. Qual è il contenuto di tali compartimenti?

- [ ] Il nome in alto, le operazioni al centro e gli attributi in basso.
- [ ] Gli attributi in alto, il nome al centro e le operazioni in basso.
- [ ] Il nome in alto, gli attributi al centro e le operazioni in basso.
- [ ] Le operazioni in alto, il nome al centro e gli attributi in basso.

09. Com'è rappresentata in UML un'aggregazione tra due oggetti o classi?

- [ ] Come un nodo a forma di rombo con tanti archi quanti sono gli oggetti coinvolti
- [ ] Come un arco con un piccolo rombo all'estremo del «contenitore»
- [ ] Come un arco con un piccolo cerchio all'estremo del «contenitore»
- [ ] Come un arco con un piccolo triangolo all'estremo del «contenitore»

10. Con riferimento al diagramma delle classi rappresentato sotto, solo una delle seguenti affermazioni è corretta. Quale?



- [ ] Ogni istanza di ElementoFileSystem è anche un'istanza di Cartella
- [ ] Ogni istanza di ElementoFileSystem è anche un'istanza di File
- [ ] Ogni istanza di ElementoFileSystem può appartenere a più di una istanza di Cartella
- [ ] Ogni istanza di ElementoFileSystem appartiene ad una e una sola istanza di Cartella

11. In un diagramma UML delle classi, che cos'è un'associazione?

- [ ] E' una relazione tra una classe e un oggetto.
- [ ] E' una relazione tra un oggetto e una classe.
- [ ] E' una relazione tra due classi.
- [ ] E' una relazione tra due oggetti.

12. Disegna un diagramma delle classi per rappresentare la relazione tra genitori e figli. Considera che una persona può avere sia genitori che figli. Annota le associazioni con ruoli e molteplicità.

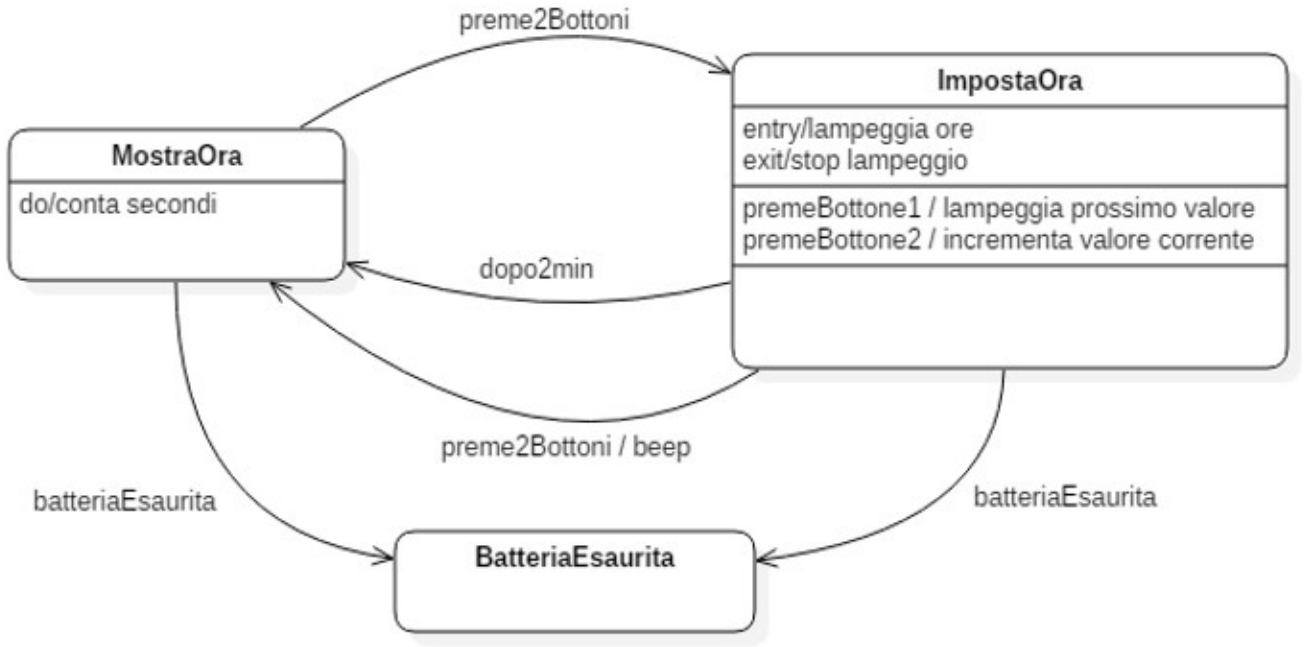
13. Disegna un diagramma delle classi che rappresenta un libro definito così: "un libro è composto da parti, a loro volta composte da capitoli; un capitolo è composto da sezioni". Concentrati sulle classi e le relazioni.

Lezione 008

01. In merito ai diagrammi UML degli stati, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Uno stato è visualizzato come un rettangolo dagli angoli arrotondati.
- [ ] Una transizione è visualizzata come un arco orientato (con una freccia) che connette due stati.
- [ ] Un triangolo nero denota lo stato finale.
- [ ] Un cerchio nero piccolo denota lo stato iniziale.

02. Con riferimento al diagramma della macchina a stati rappresentato qui sotto, solo una delle seguenti affermazioni è corretta. Quale?



- [ ] L'evento dopo2Minuti causa la transizione dallo stato MostraOra allo stato BatteriaEsaurita
- [ ] L'evento preme2Bottoni causa la transizione dallo stato MostraOra allo stato ImpostaOra
- [ ] L'evento batteriaEsaurita causa l'esecuzione dell'azione stop conta secondi
- [ ] L'evento batteriaEsaurita causa l'esecuzione dell'azione beep

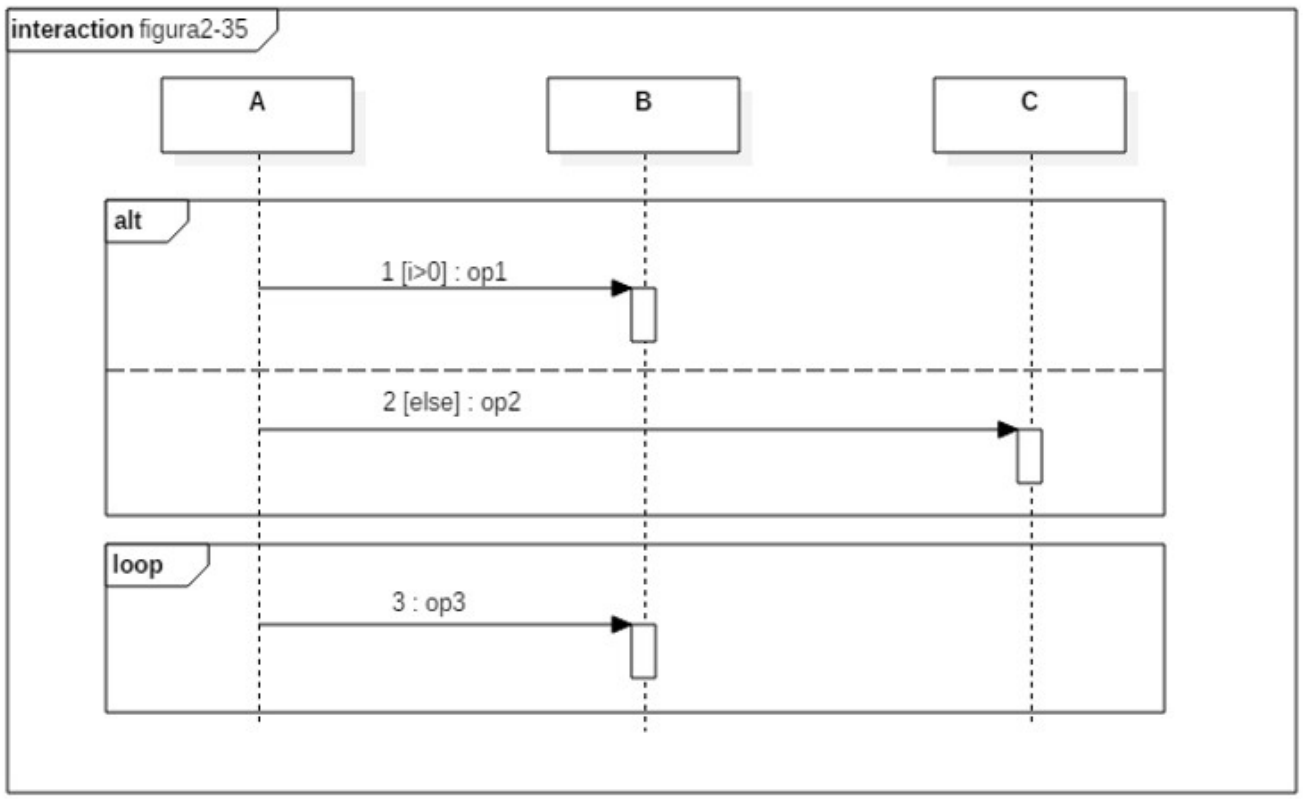
03. In merito ai diagrammi UML degli stati, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Una macchina a stati UML è una notazione per descrivere la sequenza degli stati che un oggetto assume in risposta ad eventi esterni.
- [ ] Uno stato può essere costituito dai valori di più di un attributo.
- [ ] Le macchine a stati UML sono simili agli automi a stati finiti, ma ne differiscono per il fatto che nel caso delle macchine a stati UML il numero degli stati rappresentabili non è necessariamente finito.
- [ ] Le due componenti principali di una macchina a stati sono gli stati e le transizioni.

04. In merito ai diagrammi UML delle interazioni, una delle affermazioni che seguono non è corretta. Quale?

- [ ] La ricezione di un messaggio da parte di un oggetto innesca l'esecuzione di un metodo.
- [ ] A livello di analisi non è possibile specificare messaggi dotati di argomenti.
- [ ] I diagrammi delle interazioni descrivono pattern di comunicazione tra insiemi di oggetti che interagiscono.
- [ ] Le interazioni tra gli oggetti vengono modellate come scambio di messaggi.

05. Considera il diagramma UML di sequenza rappresentato qui sotto:



Una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il frammento etichettato alt denota l'uso di una struttura selettiva.
- [ ] Un'iterazione è denotata da un frammento che contiene una partizione per ogni alternativa.
- [ ] Le alternative sono selezionate da guardie ([i>0], [else]).
- [ ] Il frammento etichettato loop denota l'uso di una struttura iterativa.

06. In merito ai diagrammi UML di comunicazione, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I diagrammi di comunicazione presentano le stesse informazioni dei diagrammi di sequenza.
- [ ] I diagrammi di comunicazione sono più facili da seguire dei diagrammi di sequenza.
- [ ] I diagrammi di comunicazione rappresentano la sequenza dei messaggi numerando le interazioni.
- [ ] I diagrammi di comunicazione sono più compatti dei diagrammi di sequenza.

07. In merito ai diagrammi UML di sequenza, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Nei diagrammi di sequenza le esecuzioni dei metodi sono rappresentate da ellissi orizzontali.
- [ ] I diagrammi di sequenza sono una delle forme che possono assumere i diagrammi delle interazioni.
- [ ] I diagrammi di sequenza rappresentano verticalmente la dimensione temporale.
- [ ] I diagrammi di sequenza rappresentano orizzontalmente gli oggetti che partecipano all'interazione.

Lezione 009

01. In merito ai nodi di controllo che coordinano il flusso del controllo, una delle categorie che seguono non è un tipo di nodo di controllo usabile nei diagrammi UML di attività. Quale?

- [ ] I nodi di iterazione
- [ ] I nodi di biforcazione
- [ ] I nodi di decisione
- [ ] I nodi di riunione.

02. In merito ai diagrammi UML delle attività, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Un diagramma delle attività mostra il flusso degli oggetti necessario al coordinamento delle attività.
- [ ] I diagrammi delle attività rappresentano l'articolazione in sequenza e il coordinamento di attività di basso livello.
- [ ] Un diagramma delle attività denota come una sequenza di attività viene realizzata in termini di una o più classi.
- [ ] I diagrammi di attività sono gerarchici: un'attività è costituita o da un'azione, o da un grafo di sotto-attività con i relativi flussi di oggetti associati.

03. Che cosa rappresenta un diagramma di attività?

- [ ] il comportamento del sistema in termini di interazioni tra insiemi di oggetti
- [ ] I flussi dei dati o dei controlli in un sistema
- [ ] La struttura statica del sistema in termini di oggetti, attributi, operazioni e relazioni
- [ ] Le funzionalità del sistema dal punto di vista dell'utente

04. Nell'ambito delle notazioni adottate nei diagrammi UML, che cos'è una nota? Fornisci un esempio.

05. Nell'ambito delle notazioni adottate nei diagrammi UML, che cos'è un package? Fornisci un esempio.

06. Nell'ambito delle notazioni adottate nei diagrammi UML, che cos'è un vincolo (constraint)? Fornisci un esempio.

07. Nell'ambito delle notazioni adottate nei diagrammi UML, che cos'è uno stereotipo? Fornisci un esempio.

Lezione 010

01. Una delle affermazioni sotto riportate non è corretta. Quale?

- [ ] I requisiti non-funzionali si applicano a numerosi aspetti del sistema, tra cui: usabilità, affidabilità, supportabilità, prestazioni
- [ ] Per ogni requisito funzionale è necessario identificare il corrispondente requisito non-funzionale
- [ ] I requisiti non-funzionali descrivono aspetti del sistema che non sono direttamente correlati con il comportamento funzionale del sistema stesso
- [ ] I requisiti funzionali descrivono l'interazione tra il sistema e il suo ambiente indipendentemente dall'implementazione

02. In merito all'elicitazione dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Gli sviluppatori elicitano i requisiti intervistando gli utenti ed osservando le loro attuali modalità di lavoro.
- [ ] Gli sviluppatori prima rappresentano i processi di lavoro correnti degli utenti come scenari attuali, quindi sviluppano scenari potenziali che descrivono le funzionalità che il futuro sistema offrirà.
- [ ] A mano a mano che la descrizione del sistema evolve e si stabilizza, sviluppatori e committente si accordano su una specifica dei requisiti che comprende requisiti funzionali, non-funzionali, scenari e casi d'uso.
- [ ] Il committente e gli utenti possono richiedere la revisione degli scenari, ma in genere non intervengono sulla formulazione dei casi d'uso.

03. In merito all'elicitazione dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il modello dell'analisi contiene le stesse informazioni della specifica dei requisiti, rappresentate però in modo formale o semi-formale.
- [ ] La specifica dei requisiti viene strutturata e formalizzata durante la fase di analisi, per produrre il modello dell'analisi.
- [ ] L'elicitazione dei requisiti e l'analisi sono fasi strettamente sequenziali.
- [ ] L'elicitazione dei requisiti si concentra sulla descrizione degli scopi del sistema.

04. Una delle attività sotto elencate non è in genere effettuata nella fase di elicitazione dei requisiti. Quale?

- [ ] Identificazione degli scenari: concreti e dettagliati.
- [ ] Identificazione degli attori: tipi di utente diversi.
- [ ] Identificazione dei casi d'uso: indotti dagli scenari.
- [ ] Identificazioni delle principali classi: la struttura del sistema.

05. In merito all'elicitazione dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I requisiti di qualità affrontano problematiche inerenti l'usabilità e l'affidabilità del sistema.
- [ ] I requisiti non-funzionali descrivono aspetti del sistema che non sono direttamente correlati con il comportamento funzionale del sistema stesso.
- [ ] I requisiti funzionali descrivono l'interazione tra il sistema e il suo ambiente indipendentemente dall'implementazione. L'ambiente include l'utente e ogni altro sistema esterno con cui il sistema considerato interagisce.
- [ ] I requisiti di qualità sono requisiti funzionali.

06. La descrizione di uno solo degli aspetti citati sotto non fa parte dei requisiti di un sistema. Quale?

- [ ] Le modalità di interazione tra utente e sistema
- [ ] Le funzionalità del sistema
- [ ] Gli errori che il sistema può rilevare e gestire
- [ ] La struttura interna del sistema

07. In merito all'elicitazione dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] L'elicitazione dei requisiti produce una descrizione del sistema comprensibile al committente.
- [ ] L'analisi dei requisiti produce un modello non ambiguo del sistema cui gli sviluppatori possono fare riferimento.
- [ ] L'ingegneria dei requisiti ha come obiettivo la definizione dei requisiti del sistema in via di progettazione.
- [ ] Un requisito è una caratteristica desiderata che il sistema deve possedere per essere accettato dal project manager.

08. In merito al tema dell'elicitazione dei requisiti, che cosa si intende con affidabilità di un sistema?

09. In merito al tema dell'elicitazione dei requisiti, che cosa si intende con usabilità di un sistema o di una sua componente?

10. In merito al tema dell'elicitazione dei requisiti, i requisiti relativi alle prestazioni (performance) coinvolgono attributi quantificabili del sistema. Cita almeno tre esempi di tali attributi.

11. Durante la progettazione del sistema i requisiti sono frequentemente validati con il committente e con gli utenti. La validazione dei requisiti si articola in controlli di completezza, consistenza, non-ambiguità e correttezza. Descrivi il significato di queste categorie di validazione.

12. In merito al tema dell'elicitazione dei requisiti, che cosa si intende con supportabilità di un sistema?

13. Tre proprietà desiderabili della specifica dei requisiti sono che essa sia realistica, verificabile e tracciabile. Descrivi queste tre proprietà.

14. Che cosa si intende con il termine greenfield engineering?

Lezione 011

01. In merito alle relazioni tra casi d'uso, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Nelle relazioni di inclusione l'evento che attiva il caso incluso (il target della relazione) è descritto nel flusso degli eventi del caso inclusore (source della relazione).
- [ ] La ridondanza tra casi d'uso può essere raccolta a fattore comune usando relazioni di inclusione.
- [ ] Nelle relazioni di esclusione l'evento che attiva il caso esclusore è descritto come post-condizione del caso che viene escluso.
- [ ] Un caso d'uso C1 estende un altro caso d'uso C2 se C2 può includere il comportamento di C1 sotto determinate condizioni.

02. In merito alle relazioni di comunicazione tra attori e casi d'uso, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Documentare le relazioni di attivazione e di comunicazione tra attori e casi d'uso significa specificare il controllo degli accessi al sistema.
- [ ] Le relazioni tra attori e casi d'uso sono identificate in fase di definizione dei diagrammi delle classi.
- [ ] Specificando quali attori possono comunicare con un caso d'uso, indichiamo chi può accedere a quali informazioni, e chi non può.
- [ ] L'attore che attiva il caso d'uso deve essere distinto dagli altri attori che comunicano con esso.

03. Nel corso vengono presentate alcune euristiche che facilitano lo sviluppo di scenari e casi d'uso. Tra le indicazioni che seguono, una non appartiene a tale insieme di euristiche. Quale?

- [ ] Definisci molti scenari non troppo dettagliati, che identifichino il raggio d'azione del sistema. Validali con gli utenti.
- [ ] Non presentare agli utenti più versioni alternative di uno stesso scenario o caso d'uso: farebbero confusione.
- [ ] Usa gli scenari per comunicare con gli utenti e validare le funzionalità.
- [ ] Usa solo mock-up e supporti visuali; il progetto dell'interfaccia utente è un compito separato.

04. Una delle entità che seguono non fa parte dei campi che costituiscono la descrizione testuale di un caso d'uso. Quale?

- [ ] Flusso degli eventi
- [ ] Attori partecipanti
- [ ] Condizioni di elaborazione
- [ ] Nome

05. In merito al concetto di caso d'uso, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Un caso d'uso descrive una serie completa di interazioni derivanti dalla sua attivazione.
- [ ] Un caso d'uso specifica tutti i possibili scenari relativi a una data funzionalità.
- [ ] Un caso d'uso rappresenta un flusso di classi nel sistema.
- [ ] Un caso d'uso è attivato da un attore e può interagire con altri attori.

06. Ci possono essere scenari di vario tipo, con usi diversi nelle diverse attività del ciclo di sviluppo del software. Tra le entità elencate nel seguito, una non è un tipo di scenario presentato nel corso. Quale?

- [ ] Scenari di addestramento
- [ ] Scenari dell'esistente (as-is)
- [ ] Scenari ipotetici (as-if)
- [ ] Scenari di valutazione

07. In merito al tema dell'elicitazione dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Gli scenari sostituiscono i casi d'uso, e possono essere usati al posto dei casi d'uso per descrivere la struttura del sistema.
- [ ] Uno scenario è una descrizione concreta, focalizzata e informale di una singola funzione del sistema dal punto di vista di un singolo attore.
- [ ] Uno scenario è una descrizione narrativa di ciò che gli utenti fanno e sperimentano quando interagiscono col sistema.
- [ ] Gli scenari contribuiscono all'elicitazione dei requisiti proponendosi come strumenti comprensibili agli utenti e al committente.

08. In merito al tema dell'elicitazione dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Identificare gli attori contribuisce a definire i confini del sistema.
- [ ] Ogni sistema software esterno che usa il sistema è un oggetto dell'analisi.
- [ ] Ogni sistema software esterno che usa il sistema è un attore.
- [ ] Il primo passo dell'elicitazione dei requisiti è l'identificazione degli attori.

09. Una delle seguenti attività non fa parte delle attività svolte durante l'elicitazione dei requisiti. Quale?

- [ ] Identificare i requisiti non-funzionali
- [ ] Raffinare i casi d'uso
- [ ] Identificare gli oggetti iniziali dell'analisi
- [ ] Impostare la decomposizione del sistema in sotto-sistemi

10. Uno dei passi in cui si articola l'elicitazione dei requisiti è l'identificazione delle relazioni tra attori e casi d'uso. Relativamente a tale contesto, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Le relazioni di comunicazione tra attori e casi d'uso descrivono il sistema in strati di funzionalità
- [ ] Le relazioni di estensione tra casi d'uso separano i flussi degli eventi eccezionali da quelli normali.
- [ ] Le relazioni di mutua esclusione riducono la complessità del modello.
- [ ] Le relazioni di inclusione riducono la ridondanza tra casi d'uso.

Lezione 012

01. Relativamente all'attività di identificazione degli oggetti iniziali dell'analisi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Gli sviluppatori costruiscono un glossario (data dictionary) che assegna un nome e una descrizione ad ogni oggetto partecipante nei vari casi d'uso.
- [ ] L'identificazione degli oggetti partecipanti produce il modello iniziale degli oggetti dell'analisi.
- [ ] Il committente e gli utenti mantengono aggiornato il glossario a mano a mano che le specifiche dei requisiti evolvono.
- [ ] Uno dei primi ostacoli che sviluppatori e utenti incontrano all'inizio della loro collaborazione è la differente terminologia.

02. Che significato associamo, nel corso, all'acronimo RAD?

- [ ] Requirement Analysis Document
- [ ] Random Assigned Data
- [ ] Root Assigned Display
- [ ] Rapid Association Device

03. In merito al documento di analisi dei requisiti (RAD), una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il RAD descrive completamente il sistema in termini di requisiti funzionali e non-funzionali.
- [ ] Il RAD contiene il glossario.
- [ ] Il RAD viene compilato in fase di elicitazione dei requisiti, e deve essere completato prima di iniziare la fase di analisi.
- [ ] Il RAD si rivolge ai committenti, agli utenti, al project manager, agli analisti e ai progettisti del sistema.

04. Descrivi la struttura del documento di analisi dei requisiti (RAD).

Lezione 013

01. Il modello dell'analisi, prodotto in fase di analisi dei requisiti, è composto da tre modelli individuali. Quale, tra i quattro elementi sotto elencati, non è una componente del modello dell'analisi?

- [ ] Il modello degli oggetti dell'analisi
- [ ] Il modello dinamico
- [ ] Il modello costi-benefici
- [ ] Il modello funzionale

02. In merito all'analisi dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Gli utenti e il committente non partecipano alla fase di analisi: il loro coinvolgimento si esaurisce nella fase di elicitazione dei requisiti.
- [ ] In fase di analisi, la specifica dei requisiti prodotta nella fase di elicitazione viene formalizzata; inoltre le condizioni al contorno e i casi eccezionali vengono esaminati in dettaglio.
- [ ] In fase di analisi vengono rimssi errori ed ambiguità ancora presenti nel RAD.
- [ ] L'analisi produce un modello del sistema corretto, completo, consistente, non ambiguo.

03. In merito al modello degli oggetti dell'analisi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il modello degli oggetti dell'analisi definisce l'interfaccia e l'implementazione dei principali metodi del sistema.
- [ ] Il modello degli oggetti dell'analisi è rappresentato tipicamente da diagrammi delle classi UML.
- [ ] Il modello degli oggetti dell'analisi cataloga i principali concetti visibili all'utente.
- [ ] Il modello degli oggetti dell'analisi si concentra sui concetti, le proprietà e le relazioni che il sistema manipola.

04. In merito al modello dinamico definito in fase di analisi dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il modello dinamico contiene diagrammi UML di sequenza.
- [ ] Il modello dinamico contiene diagrammi UML delle macchine a stati.
- [ ] Il modello dinamico evidenzia le relazioni di ereditarietà tra le classi.
- [ ] Il modello dinamico si concentra sul comportamento del sistema.

05. Nel contesto dell'analisi dei requisiti, che cosa intendiamo per generalizzazione? Fornisci un esempio.

06. Nel contesto dell'analisi dei requisiti, che cosa sono gli oggetti control? Fornisci una definizione ed un esempio.

07. Nel contesto dell'analisi dei requisiti, che cosa sono gli oggetti boundary? Fornisci una definizione ed un esempio.

08. Nel contesto dell'analisi dei requisiti, che cosa sono gli oggetti entity? Fornisci una definizione ed un esempio.

09. Nel contesto dell'analisi dei requisiti, che cosa intendiamo per specializzazione? Fornisci un esempio.

Lezione 014

01. Nel corso vengono suggerite alcune euristiche per l'identificazione, in fase di analisi, degli oggetti control. Quale, tra le indicazioni che seguono, non è una delle euristiche suggerite nel corso?

- [ ] Identifica un oggetto control per ogni attore del caso d'uso.
- [ ] Identifica un oggetto control per ogni caso d'uso.
- [ ] Assicurati che le condizioni di ingresso e uscita del caso d'uso siano ben definite.
- [ ] Identifica un oggetto control per ogni classe.

02. In merito all'uso di diagrammi di sequenza in fase di analisi dei requisiti, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I diagrammi di sequenza evidenziano la distribuzione dei comportamenti tra gli oggetti che partecipano ad un caso d'uso o ad uno scenario.
- [ ] I diagrammi di sequenza sono utili agli sviluppatori per chiarire aspetti oscuri dei requisiti.
- [ ] I diagrammi di sequenza sono meno efficaci dei casi d'uso nella comunicazione con gli utenti.
- [ ] I diagrammi di sequenza connettono gli scenari ai diagrammi delle macchine a stati.

03. Nel corso vengono suggerite alcune euristiche per l'identificazione, in fase di analisi, degli oggetti boundary. Quale, tra le indicazioni che seguono, non è una delle euristiche suggerite nel corso?

- [ ] Identifica le componenti dell'interfaccia utente necessarie per attivare i casi d'uso.
- [ ] Identifica i moduli, i formulari e le schede necessari per l'input dei dati nel sistema.
- [ ] Identifica gli aspetti visivi dell'interfaccia utente: font, colori, immagini ecc.
- [ ] Identifica i messaggi, le notifiche, gli avvisi che il sistema usa per rispondere all'utente.

04. Nel corso vengono suggerite alcune euristiche per l'identificazione, in fase di analisi, degli oggetti entity. Quale, tra le indicazioni che seguono, non è una delle euristiche suggerite nel corso?

- [ ] Le sorgenti e le destinazioni dei dati sono buoni candidati a diventare oggetti entity.
- [ ] I verbi che gli utenti usano frequentemente per descrivere le funzionalità desiderate del sistema sono buoni candidati a diventare oggetti entity.
- [ ] I nomi ricorrenti nei casi d'uso sono buoni candidati a diventare oggetti entity.
- [ ] I termini che sviluppatori o utenti hanno bisogno di chiarire per comprendere un caso d'uso sono buoni candidati a diventare oggetti entity.

05. Nel corso vengono suggerite alcune euristiche per lo sviluppo, in fase di analisi, dei diagrammi di sequenza. Quale, tra le indicazioni che seguono, non è una delle euristiche suggerite nel corso?

- [ ] Altri oggetti control possono essere creati dall'oggetto boundary che ha iniziato il caso d'uso.
- [ ] La terza colonna è l'oggetto entity che gestisce il resto del caso d'uso.
- [ ] La seconda colonna è un oggetto boundary che l'attore ha usato per iniziare il caso d'uso.
- [ ] La prima colonna rappresenta l'attore che inizia il caso d'uso.

Lezione 017

01. Durante la fase di progettazione del sistema (system design), una delle attività che seguono non è considerata strategica per la costruzione del sistema. Quale?

- [ ] La definizione delle politiche di accesso.
- [ ] La gestione dei dati persistenti.
- [ ] L'elicitazione dei requisiti non-funzionali.
- [ ] Il controllo globale del flusso.

02. Nell'ambito dell'attività di decomposizione del sistema in sotto-sistemi, una delle affermazioni che seguono non è corretta. Quale?

- [ ] La decomposizione iniziale del sistema si avvale di stili architetturali standard.
- [ ] L'identificazione degli obiettivi del design permette di definire le qualità che il sistema esibirà.
- [ ] Una volta che tutti i sotto-sistemi sono stati identificati e descritti formalmente, solo allora si può procedere con l'identificazione degli obiettivi del design.
- [ ] Ogni sotto-sistema viene tipicamente sviluppato da un diverso gruppo di lavoro.

03. Tra le entità elencate nel seguito, una non è definita (o prodotta) in fase di system design. Quale?

- [ ] L'architettura software.
- [ ] Gli obiettivi del design.
- [ ] I casi d'uso limite.
- [ ] I casi d'uso standard.

04. Relativamente ai design goals identificati in fase di system design, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I design goals descrivono le qualità del sistema che gli sviluppatori devono ottimizzare.
- [ ] I design goals aiutano a decidere nelle situazioni di compromesso.
- [ ] I design goals forniscono indicazioni anche nella fase di object design.
- [ ] I design goals sono derivati dai requisiti funzionali.

05. Quale, tra le definizioni che seguono, ti sembra più corretta per identificare le caratteristiche di un sotto-sistema?

- [ ] Un sotto-sistema è una componente hardware periferica rispetto al sistema centrale.
- [ ] Un sotto-sistema è uno strato di software collocato in basso nello stack dell'applicazione.
- [ ] Un sotto-sistema è un modulo software sviluppato per fornire servizi aggiuntivi rispetto a quelli centrali dell'applicazione.
- [ ] Un sotto-sistema è una parte sostituibile del sistema che, dotata di interfacce ben definite, incapsula lo stato e il comportamento delle classi che contiene.

06. In merito alla fase di decomposizione del sistema, una delle affermazioni che seguono non è corretta. Quale?

- [ ] La decomposizione del sistema va talvolta evitata, perché ne deriva software difficilmente riusabile.
- [ ] Durante la fase di decomposizione è necessario risolvere tutte le problematiche trasversali al sistema.
- [ ] La decomposizione consente di dominare la complessità del progetto.
- [ ] La decomposizione del sistema fornisce indicazioni rilevanti alla successiva fase di object design.

07. Che cos'è l'interfaccia di un sotto-sistema?

- [ ] E' l'insieme dei controlli con cui un operatore può configurare il sotto-sistema.
- [ ] E' l'insieme delle operazioni che il sotto-sistema rende disponibili.
- [ ] E' lo strato software che incapsula le interazioni tra il sotto-sistema e il sistema operativo ospite.
- [ ] E' l'insieme dei messaggi che le classi che costituiscono il sotto-sistema si scambiano tra loro.

Lezione 018

01. In merito ai concetti di partizionamento e stratificazione di un sistema, una delle affermazioni che seguono non è corretta. Quale?

☐ La decomposizione gerarchica di un sistema si arresta quando i sotto-sistemi sono abbastanza semplici da essere affidati ad un singolo sviluppatore, o a un piccolo gruppo di sviluppatori.

☐ Una decomposizione in sotto-sistemi è il risultato di un'attività o di partizionamento o di stratificazione, ma non di entrambe. I due metodi non possono essere applicati insieme.

☐ Ogni sotto-sistema aggiunge un certo costo computazionale (overhead) a causa delle sue interfacce verso gli altri sotto-sistemi.

☐ Il partizionamento o la stratificazione eccessivi possono aumentare la complessità del sistema.

02. Relativamente ai concetti di coupling e coesione, una delle affermazioni che seguono non è corretta. Quale?

☐ Minimizzare il coupling tra i sotto-sistemi facilita il testing dell'intero sistema.

☐ Massimizzare la coesione di un sotto-sistema ne favorisce il riuso.

☐ Minimizzare il coupling in un sistema riduce le dipendenze interne ad ogni sotto-sistema.

☐ Massimizzare la coesione di un sotto-sistema lo rende più semplice.

03. Relativamente al concetto di coesione, una delle affermazioni che seguono non è corretta. Quale?

☐ Se un sotto-sistema contiene numerosi oggetti non reciprocamente correlati, il suo livello di coesione è basso.

☐ Una proprietà desiderabile della decomposizione in sotto-sistemi è che essa produca sotto-sistemi con alto livello di coesione.

☐ Se un sotto-sistema contiene molti oggetti che sono collegati l'uno all'altro ed eseguono compiti simili, il suo livello di coesione è alto.

☐ La coesione è la misura quantitativa delle dipendenze tra due sotto-sistemi.

04. Relativamente al concetto di coupling, una delle affermazioni che seguono non è corretta. Quale?

☐ Una proprietà desiderabile della decomposizione in sotto-sistemi è che questi risultino ragionevolmente disaccoppiati: in tal modo l'impatto di errori e future modifiche sarà localizzato in un solo sotto-sistema.

☐ Se due sotto-sistemi sono accoppiati in modo lasco (basso livello di coupling) essi sono relativamente indipendenti, e le modifiche apportate ad uno dei due sotto-sistemi avranno poco impatto sull'altro.

☐ Se due sotto-sistemi sono strettamente accoppiati (elevato livello di coupling), le modifiche apportate a uno di essi avranno probabilmente un impatto significativo anche sull'altro.

☐ Il coupling è la misura quantitativa delle dipendenze internamente a un sotto-sistema.

05. Relativamente al concetto di layering (stratificazione) di un sistema, una delle affermazioni che seguono non è corretta. Quale?

☐ In un'architettura chiusa ogni layer può accedere solo al layer immediatamente sottostante.

☐ Ogni layer ospita al più un sotto-sistema.

☐ L'insieme dei layers di un sistema è gerarchico.

☐ In un'architettura aperta un layer può accedere a tutti i layers sottostanti.

06. Descrivi le architetture a strati aperte. Fornisci un esempio.

07. Fornisci una definizione del concetto di coesione relativamente a un sotto-sistema software.

08. La riduzione del coupling in sede di progettazione dei sotto-sistemi è sempre un valore da perseguire in assoluto? In quali casi le controindicazioni prevalgono?

09. Descrivi le architetture a strati chiuse. Fornisci un esempio.

10. Che cosa si intende, in Ingegneria del software, con il termine coupling?

11. Quali sono i vantaggi di un'architettura a strati?

12. Fornisci una definizione di partizionamento di un sistema.

13. Quali sono gli svantaggi di un'architettura a strati?

Lezione 019

01. Nell'architettura Model-View-Controller come interagiscono tra loro i sotto-sistemi Model e View?

- [ ] I due sotto-sistemi non comunicano direttamente.
- [ ] Model e View interagiscono prevalentemente attraverso la mediazione di Controller, che espone un'API per l'accesso ai servizi offerti.
- [ ] Model e View interagiscono attaverso il repository di sistema.
- [ ] I cambiamenti nello stato del sotto-sistema Model sono propagati al sotto-sistema View mediante un protocollo subscriber/notifier.

02. Tra gli aspetti elencati nel seguito, uno non entra a far parte del concetto di architettura software presentato nel corso. Quale?

- [ ] La gestione delle condizioni limite.
- [ ] La decomposizione del sistema in sotto-sistemi.
- [ ] L'articolazione degli scenari in casi d'uso.
- [ ] Il controllo globale del flusso.

03. Quale, tra le seguenti, non identifica un'architettura software presentata nel corso?

- [ ] Client / Server
- [ ] Model-View-Controller
- [ ] SQL Injection
- [ ] Repository

04. Quale, tra le seguenti, non identifica un'architettura software presentata nel corso?

- [ ] Singleton
- [ ] Peer-to-peer
- [ ] Three-tier
- [ ] Pipe & Filter

05. Relativamente allo stile architetturale Repository, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il controllo del flusso è effettuato esclusivamente dai sotto-sistemi, mediante l'attivazione di triggers e stored procedures.
- [ ] I sotto-sistemi accedono e modificano una singola struttura dati.
- [ ] Il repository non ha conoscenza degli altri sotto-sistemi.
- [ ] I sotto-sistemi sono relativamente indipendenti e interagiscono solo attraverso il repository.

06. Relativamente allo stile architetturale Repository, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Un difetto dell'architettura Repository è che in essa risulta difficile controllare la concorrenza tra i sotto-sistemi e l'integrità dei dati.
- [ ] I compilatori e gli ambienti per lo sviluppo del software adottano spesso lo stile architetturale Repository.
- [ ] I sistemi articolati su un'architettura Repository che effettua il controllo globale del flusso in base allo stato delle proprie strutture dati centrali prendono talvolta il nome di sistemi blackboard.
- [ ] Lo stile architetturale repository è tipicamente usato con sistemi gestiti da database.

07. Relativamente allo stile architetturale Repository, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Una volta che il repository centrale è ben definito, è facile aggiungere ulteriori servizi nella forma di altri sotto-sistemi.
- [ ] In un'architettura Repository la coesione di ogni sotto-sistema è bassa, quindi è difficile cambiare il repository senza dover cambiare anche tutti i sotto-sistemi.
- [ ] Il repository centrale può rapidamente diventare un «collo di bottiglia» (bottleneck), sia dal punto di vista delle prestazioni che da quello della modificabilità.
- [ ] Le architetture in stile Repository sono adeguate per applicazioni con compiti di elaborazione di dati complessi e soggetti a frequenti modifiche.

08. Relativamente allo stile architetturale Model-View-Controller (MVC), una delle affermazioni che seguono non è corretta. Quale?

- [ ] Nell'architettura MVC il sotto-sistema View rappresenta la presentazione all'utente.
- [ ] Nell'architettura MVC il sotto-sistema Model rappresenta la conoscenza del dominio.
- [ ] L'accoppiamento tra il sotto-sistema Model e il sotto-sistema Controller è elevato.
- [ ] Nell'architettura MVC il sotto-sistema Controller rappresenta la logica che governa l'interazione tra sistema e utente.

09. Relativamente allo stile architetturale Peer-to-peer, una delle affermazioni che seguono non è corretta. Quale?

- [ ] In un'architettura Peer-to-peer tutti i sotto-sistemi sono sia client che server.
- [ ] Il flusso di controllo all'interno di ogni sotto-sistema è indipendente dagli altri, eccetto che per la sincronizzazione delle richieste e delle risposte.
- [ ] La topologia tipica dell'architettura Peer-to-peer è ad albero.
- [ ] I sistemi Peer-to-peer sono più difficili da progettare dei sistemi Client/Server, perché introducono la possibilità di deadlock.

10. Relativamente allo stile architetturale Client / Server, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il flusso di controllo nel client è indipendente da quello nel server, tranne che nei momenti di sincronizzazione in cui vengono inoltrate le richieste o ricevuti i risultati.
- [ ] Nello stile architetturale client/server un sotto-sistema, il server, offre servizi a istanze di altri sotto-sistemi detti client.
- [ ] Le interazioni con l'utente vengono in genere gestite direttamente dal server.
- [ ] Il server non ha nozione dei client.

11. Relativamente allo stile architetturale Pipe & Filter, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Le associazioni tra ogni sotto-sistema e il successivo sono chiamate «pipes».
- [ ] Nello stile architetturale Pipe & Filter ogni sotto-sistema elabora i dati da uno o più input e manda i risultati in output ad altri sotto-sistemi.
- [ ] Ogni sotto-sistema viene eseguito in modo sequenziale: la sua esecuzione inizia solo quando il sotto-sistema che lo precede nella catena ha terminato.
- [ ] Ogni sotto-sistema conosce solo il contenuto e il formato dei dati ricevuti dalle pipes di input, non i sotto-sistemi che li hanno prodotti.

12. Qual è il principale vantaggio dello stile architetturale Pipe & Filter? Per quali tipologie di sistemi è indicato?

13. Descrivi lo stile architetturale Pipe & Filter. Indica un esempio in cui tale stile è ampiamente adottato.

14. Qual è il vantaggio insito nell'adozione di uno stile architetturale Three-tier?

15. Descrivi lo stile architetturale Four-tier. Fornisci un esempio.

16. Descrivi lo stile architetturale Three-tier.

17. Fornisci una definizione del concetto di deadlock di due processi. Quale stile architetturale, tra quelli presentati nel corso, è potenzialmente soggetto a deadlock?

18. Fornisci una definizione del concetto di operazione callback. Quale stile architetturale, tra quelli presentati nel corso, fa tipicamente uso di operazioni callback?

Lezione 020

01. Relativamente alla definizione degli obiettivi del design, una delle affermazioni che seguono non è corretta. Quale?

- ☐ Alcuni obiettivi del design possono essere inferiti dal dominio dell'applicazione.
- ☐ Alcuni obiettivi del design possono essere inferiti dai requisiti non-funzionali.
- ☐ La definizione degli obiettivi del design è il passo finale del system design.
- ☐ Alcuni obiettivi del design devono essere elicitati dal committente.

02. In fase di identificazione degli obiettivi del design possiamo fare riferimento a cinque gruppi di criteri progettuali. Citane almeno quattro.

03. Un gruppo di criteri progettuali su cui si basa l'identificazione degli obiettivi del design prende in considerazione la manutenzione del sistema. Fornisci degli esempi delle caratteristiche del sistema tipicamente coinvolte.

04. Un gruppo di criteri progettuali su cui si basa l'identificazione degli obiettivi del design prende in considerazione i costi associati al sistema. Fornisci degli esempi delle tipologie di costo tipicamente coinvolte.

05. Un gruppo di criteri progettuali su cui si basa l'identificazione degli obiettivi del design prende in considerazione la dependability (affidabilità) del sistema. Fornisci degli esempi delle caratteristiche del sistema tipicamente coinvolte.

06. Un gruppo di criteri progettuali su cui si basa l'identificazione degli obiettivi del design prende in considerazione le prestazioni del sistema. Fornisci degli esempi delle caratteristiche del sistema tipicamente coinvolte.

07. Un gruppo di criteri progettuali su cui si basa l'identificazione degli obiettivi del design prende in considerazione i criteri dell'utente finale. Fornisci degli esempi delle caratteristiche del sistema tipicamente coinvolte.

Lezione 021

01. Un design pattern specifico si dimostra spesso utile nella decomposizione in sotto-sistemi. Quale?

☐ Factory

☐ Singleton

☐ Observer

☐ Façade

02. Relativamente all'attività di identificazione dei sotto-sistemi durante il system design, una delle affermazioni che seguono non è corretta. Quale?

☐ L'identificazione dei sotto-sistemi è soggetta a revisione ogni volta che nuove problematiche progettuali emergono.

☐ Determinare i sotto-sistemi durante il system design è un'attività simile a quella di determinare gli oggetti durante la fase di analisi: alcune euristiche per l'identificazione degli oggetti sono riproponibili per i sotto-sistemi.

☐ L'identificazione dei sotto-sistemi è soggetta ad iterazioni che, all'inizio, possono richiedere cambiamenti drastici al modello del design.

☐ Durante l'identificazione dei sotto-sistemi è consentito decomporre un sotto-sistema complesso nelle sue parti più semplici, ma non è ammesso fondere due o più sotto-sistemi in uno unico.

03. Da che cosa viene derivata, di norma, la versione iniziale della decomposizione in sotto-sistemi?

☐ Dai requisiti non-funzionali.

☐ Dai diagrammi degli stati UML.

☐ Dai diagrammi di sequenza UML.

☐ Dai requisiti funzionali.

04. Una delle seguenti indicazioni non rappresenta un'euristica valida per raggruppare gli oggetti in sotto-sistemi. Quale?

☐ E' utile creare un sotto-sistema dedicato per gli oggetti usati per trasferire dati tra i sotto-sistemi .

☐ E' utile raggruppare gli oggetti in modo che sia massimo il numero di associazioni che attraversano i confini dei sotto-sistemi.

☐ Tutti gli oggetti identificati in uno specifico caso d'uso sono assegnati allo stesso sotto-sistema .

☐ Tutti gli oggetti di un sotto-sistema dovrebbero essere correlati sul piano funzionale.

05. Descrivi le conseguenze (positive ed, eventualmente, negative) che possono derivare dall'adozione del design pattern Façade.

06. Abbiamo classificato gli obiettivi del design in 5 categorie: prestazioni, dependability, costo, manutenzione e utente finale. Assegna una o più categorie al seguente obiettivo:La cassa automatica deve resistere ad attacchi di «forza bruta» (ad es. un utente che cerchi di scoprire un codice segreto provando tutte le possibilità).

07. Abbiamo classificato gli obiettivi del design in 5 categorie: prestazioni, dependability, costo, manutenzione e utente finale. Assegna una o più categorie al seguente obiettivo:L'alloggiamento del distributore di biglietti deve riservare spazio per installare nuovi bottoni qualora il numero di tariffe aumenti.

08. Abbiamo classificato gli obiettivi del design in 5 categorie: prestazioni, dependability, costo, manutenzione e utente finale. Assegna una o più categorie al seguente obiettivo: Il distributore di biglietti deve essere in grado di emettere un biglietto anche nel caso di caduta della connessione di rete.

09. Abbiamo classificato gli obiettivi del design in 5 categorie: prestazioni, dependability, costo, manutenzione e utente finale. Assegna una o più categorie al seguente obiettive: Dopo aver digitato un comando, l'utente deve ricevere una risposta entro un secondo.

10. Descrivi le conseguenze (positive ed, eventualmente, negative) che possono derivare dall'adozione del design pattern Observer.

11. Descrivi la soluzione proposta dal design pattern Observer.

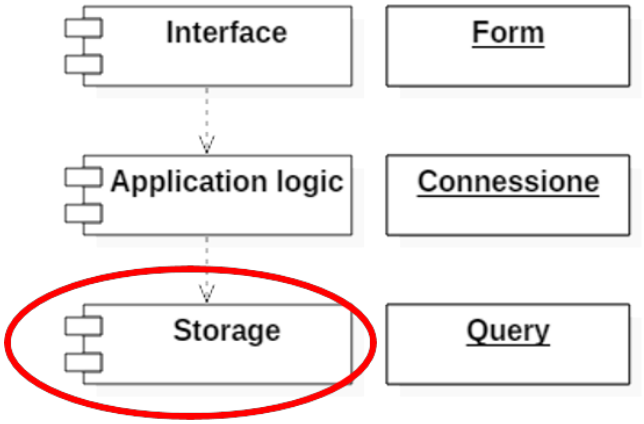
12. Quale problema di progettazione viene affrontato con l'uso del design pattern Observer?

13. Fornisci una definizione di design pattern.

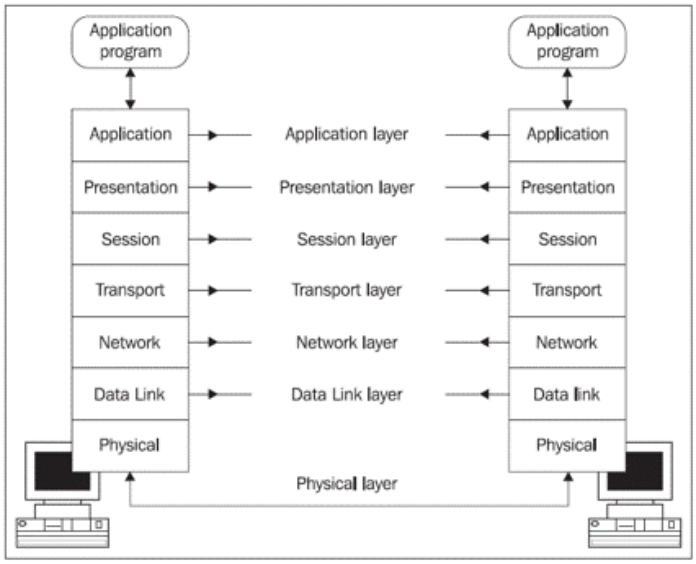
14. Quale problema di progettazione viene affrontato con l'uso del design pattern Façade?

15. Quali elementi fanno parte della struttura di un design pattern?

16. In molte architetture, come ad es. quelle three-tier e four-tier, la memorizzazione di oggetti persistenti è affidata a un tier dedicato. Quali obiettivi del design hanno condotto a questa decisione?



17. Elenca gli obiettivi del design che sarebbe difficile soddisfare usando un'architettura chiusa con molti strati, come ad es. l'esempio OSI-ISO.



Lezione 022

01. In merito all'attività progettuale di mappatura hardware/software , una delle affermazioni seguenti non è corretta. Quale?

- [ ] I componenti software commerciali devono essere scritti nello stesso linguaggio di programmazione usato per sviluppare le altre parti del sistema.
- [ ] In fase di definizione della mappatura hardware/software vengono selezionati componenti software riusabili.
- [ ] I componenti software commerciali di serie devono essere incapsulati per minimizzare le dipendenze e facilitare l'intercambiabilità.
- [ ] In fase di definizione della mappatura hardware/software possono essere definiti nuovi sotto-sistemi.

02. In merito all'attività progettuale che definisce la gestione dei dati persistenti, una delle affermazioni seguenti non è corretta. Quale?

- [ ] La gestione delle problematiche di persistenza identifica dove i dati persistenti devono essere memorizzati.
- [ ] La gestione delle problematiche di persistenza definisce le modalità di accesso ai dati persistenti.
- [ ] La gestione delle problematiche di persistenza seleziona quali dati devono essere persistenti.
- [ ] La gestione delle problematiche di persistenza definisce la struttura delle classi e le operazioni effettuabili sui dati del sistema.

03. L'attività progettuale di definizione del controllo degli accessi richiede ai progettisti di fornire risposte a numerose domande. Una delle domande seguenti non attiene a tale attività. Quale?

- [ ] Quali attori possono accedere a quali dati?
- [ ] Sotto-sistemi diversi possono adottare politiche di controllo degli accessi diverse?
- [ ] È possibile modificare dinamicamente il controllo dell'accesso?
- [ ] Come viene realizzato il controllo dell'accesso?

04. L'attività progettuale di definizione del flusso di controllo richiede ai progettisti di fornire risposte a numerose domande. Una delle domande seguenti non attiene a tale attività. Quale?

- [ ] In che modo il sistema mette in sequenza le operazioni?
- [ ] Il sistema è guidato dagli eventi?
- [ ] Il sistema può gestire più di un'interazione utente in ogni dato istante?
- [ ] Il sistema può essere controllato in remoto, ad es. via Internet?

05. In merito all'attività progettuale di definizione del flusso di controllo, una delle affermazioni seguenti non è corretta. Quale?

- [ ] Se si sceglie un flusso del controllo guidato dagli eventi, i sotto-sistemi devono rendere disponibili degli event handlers.
- [ ] Le scelte operate in fase di definizione del flusso di controllo determinano il numero minimo e massimo dei file contemporaneamente aperti nel sistema.
- [ ] La scelta del flusso del controllo ha effetti sulle interfacce dei sotto-sistemi.
- [ ] Se il flusso del controllo prevede un meccanismo di concorrenza basato sui threads, i sotto-sistemi devono garantire la mutua esclusione nelle sezioni critiche del codice.

06. In merito all'attività progettuale di identificazione delle condizioni limite, una delle affermazioni seguenti non è corretta. Quale?

- [ ] L'identificazione delle condizioni limite include la determinazione delle quantità minime di risorse (memoria centrale, spazio disco, versione del sistema operativo ecc.) necessarie al sistema.
- [ ] L'identificazione delle condizioni limite include la definizione delle modalità con cui vengono gestite le eccezioni.
- [ ] L'identificazione delle condizioni limite include la definizione delle modalità con cui viene inizializzato il sistema.
- [ ] L'identificazione delle condizioni limite include la definizione delle modalità con cui viene spento il sistema.

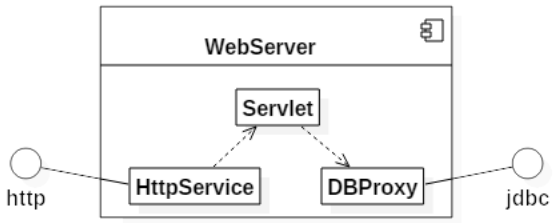
07. L'attività progettuale di mappatura hardware/software richiede ai progettisti di fornire risposte a numerose domande. Una delle domande seguenti non attiene a tale attività. Quale?

- [ ] Come vengono gestiti gli eventi eccezionali?
- [ ] Quali funzionalità sono attribuite ai vari nodi?
- [ ] Qual è la configurazione hardware del sistema?
- [ ] Come comunicano tra loro i nodi?

08. Descrivi la soluzione proposta dal design pattern Flyweight.

09. Quale problema di progettazione viene affrontato con l'uso del design pattern Flyweight?

10. Considera il diagramma UML di deployment riportato qui sotto:



Fornisci una descrizione verbale delle informazioni desumibili dal diagramma.

11. Descrivi la funzione e la struttura dei diagrammi UML di deployment. Fornisci un esempio.

12. Considera il pattern MVC, generalizzazione del design pattern Observer:

Indica in che misura la struttura di MVC facilita oppure ostacola il seguente obiettivo del design:Controllo degli accessi (la garanzia che solo gli utenti autorizzati possano accedere a parti specifiche del model)

13. Considera il pattern MVC, generalizzazione del design pattern Observer:

Indica in che misura la struttura di MVC facilita oppure ostacola il seguente obiettivo del design:Modificabilità (l'aggiunta di nuovi attributi nel model)

14. Considera il pattern MVC, generalizzazione del design pattern Observer:

Indica in che misura la struttura di MVC facilita oppure ostacola il seguente obiettivo del design:Ottimizzazione dei tempi di risposta (il tempo tra l'input dell'utente e l'aggiornamento di tutte le views)

15. **Considera il pattern MVC, generalizzazione del design pattern Observer:**

Indica in che misura la struttura di MVC facilita oppure ostacola il seguente obiettivo del design:Estendibilità (l'aggiunta di nuovi tipi di view)

16. **Quale problema di progettazione viene affrontato con l'uso del design pattern Simple Factory?**

17. **Descrivi la soluzione proposta dal design pattern Iterator.**

18. **Quale problema di progettazione viene affrontato con l'uso del design pattern Iterator?**

19. **Descrivi la soluzione proposta dal design pattern Proxy.**

20. **Quale problema di progettazione viene affrontato con l'uso del design pattern Proxy?**

21. **Descrivi la soluzione proposta dal design pattern Adapter.**

22. **Quale problema di progettazione viene affrontato con l'uso del design pattern Adapter?**

23. **Descrivi la soluzione proposta dal design pattern Singleton.**

24. **Quale problema di progettazione viene affrontato con l'uso del design pattern Singleton?**

25. **Descrivi la soluzione proposta dal design pattern Simple Factory.**

Lezione 023

01. Alcune attività del system design devono essere svolte a livello di sistema, prima che la decomposizione in sotto-sistemi assegni a gruppi di lavoro separati lo sviluppo in parallelo delle varie componenti. Una tra le attività seguenti non è riconducibile al system design. Quale?

- [ ] La mappatura hardware/software
- [ ] Il controllo degli accessi
- [ ] La gestione dei dati persistenti
- [ ] La specifica delle operazioni

Lezione 024

01. Relativamente alla mappatura tra sotto-sistemi, processori e componenti, una delle affermazioni che seguono non è corretta. Quale?

☐ Selezionare la configurazione hardware può include la decisione di fare uso di macchine virtuali.

☐ E' importante allocare i componenti software nei vari nodi senza introdurre nuovi oggetti ne' sotto-sistemi.

☐ L'allocazione di oggetti e sotto-sistemi ai nodi può avvenire solo dopo che la configurazione hardware è stata definita.

☐ La mappatura dei sotto-sistemi sullo hardware viene effettuata prima di altre attività del system design, perché ha un impatto significativo sulle prestazioni e la complessità del sistema.

02. Relativamente alla mappatura tra sotto-sistemi, processori e componenti, una delle affermazioni che seguono non è corretta. Quale?

☐ Gli sviluppatori devono scegliere con cura i componenti che useranno per sviluppare il sistema.

☐ Allocare i sotto-sistemi ai nodi hardware ci permette di distribuire le funzionalità e le capacità di elaborazione dove sono maggiormente richieste.

☐ Gli sviluppatori devono mantenere la consapevolezza che l'integrazione dei componenti potrebbe richiedere molto lavoro.

☐ La distribuzione delle funzionalità semplifica la memorizzazione, il trasferimento, la replica e la sincronizzazione dei dati tra i sotto-sistemi.

03. Relativamente all'identificazione degli oggetti persistenti, una delle affermazioni che seguono non è corretta. Quale?

☐ Possiamo identificare gli oggetti persistenti esaminando tutte le classi che devono sopravvivere lo spegnimento e ripartenza del sistema.

☐ Gli oggetti complessi non possono essere resi persistenti mediante un database relazionale, che può memorizzare solo tipi di dato semplici.

☐ Nell'identificare gli oggetti persistenti dobbiamo tenere conto degli eventi eccezionali (errori, mancanza di risorse) che causano la terminazione forzata del programma.

☐ Nell'identificare gli oggetti persistenti dobbiamo tenere conto delle chiusure controllate.

04. Per garantire che la decomposizione in sotto-sistemi soddisfi tutti i requisiti non-funzionali e che, in fase di implementazione, tutti i vincoli vengano rispettati, in fase di decomposizione del sistema devono essere svolte alcune attività. Quale, tra le seguenti, non è una di tali attività?

☐ L'identificazione dei servizi .

☐ La progettazione del flusso di controllo globale .

☐ L'identificazione delle condizioni limite .

☐ L'ispezione dei test di usabilità.

05. Per garantire che la decomposizione in sotto-sistemi soddisfi tutti i requisiti non-funzionali e che, in fase di implementazione, tutti i vincoli vengano rispettati, in fase di decomposizione del sistema devono essere svolte alcune attività. Quale, tra le seguenti, non è una di tali attività?

☐ La mappatura tra sotto-sistemi, processori e componenti .

☐ L'identificazione e la memorizzazione di dati persistenti .

☐ Il controllo degli accessi .

☐ L'identificazione delle interfacce e delle operazioni di tutti gli oggetti.

06. Relativamente all'identificazione dei dati persistenti, una delle affermazioni che seguono non è corretta. Quale?

☐ L'identificazione dei dati persistenti può avere implicazioni sulla gestione della concorrenza.

☐ L'identificazione dei dati persistenti può avere implicazioni sulla strategia generale di controllo del flusso.

☐ L'identificazione dei dati persistenti non ha effetti significativi sulla decomposizione in sotto-sistemi.

☐ Sebbene memorizzare i dati su semplici file sia talvolta una soluzione semplice ed economica, usare un database permette di effettuare queries complesse sui dati. e può migliorare la flessibilità del sistema.

07. In merito alle strategie di scelta dei meccanismi di persistenza, discuti vantaggi e svantaggi dell'uso di un database ad oggetti.

08. In merito alle strategie di scelta dei meccanismi di persistenza, discuti vantaggi e svantaggi dell'uso di semplici file.

09. Nel contesto della progettazione di sistemi software, che cosa intendiamo col termine persistenza?

10. Che cos'è una macchina virtuale? Perché questo concetto è interessante nel contesto del system design?

11. In merito alle strategie di scelta dei meccanismi di persistenza, discuti vantaggi e svantaggi dell'uso di un database relazionale.

Lezione 025

01. Relativamente all'identificazione dei servizi, una delle seguenti affermazioni non è corretta. Quale?

☐ Mettendo a fuoco le dipendenze tra i sotto-sistemi validiamo lo stato corrente dell'architettura del sistema.

☐ Mettendo a fuoco le dipendenze tra i sotto-sistemi specifichiamo in dettaglio le responsabilità di ogni sotto-sistema.

☐ Mettendo a fuoco le dipendenze tra i sotto-sistemi aumentiamo la coesione di ogni sotto-sistema.

☐ Mettendo a fuoco le dipendenze tra i sotto-sistemi evidenziamo eventuali omissioni nella decomposizione iniziale.

02. Con riferimento all'identificazione dei servizi offerti da ogni sotto-sistema, una delle seguenti attività non fa parte delle attività svolte in fase di system design . Quale?

☐ Dotare ogni servizio di un nome.

☐ Riconsiderare le dipendenze tra i sotto-sistemi.

☐ Specificare le operazioni, i parametri e i vincoli di ogni servizio.

☐ Definire un'interfaccia per ogni servizio identificato.

03. Quale tipologia di oggetti viene tipicamente usata per realizzare il meccanismo di controllo del flusso?

☐ Le decisioni inerenti il controllo del flusso sono incapsulate in un design pattern Simple Factory che genera la corretta sequenza di operazioni.

☐ Un insieme di oggetti boundary integra nelle proprie variabili di stato le dipendenze che rappresentano la corretta sequenza delle operazioni.

☐ Un insieme di oggetti entity genera gli eventi che controllano la corretta sequenza di esecuzione delle operazioni.

☐ Un insieme di oggetti control registrano gli eventi esterni, memorizzano i loro valori di stato temporanei, e chiamano in sequenza corretta i metodi di altri oggetti boundary ed entity.

04. Considera la problematica relativa al controllo degli accessi, e in particolare lo strumento che nel corso abbiamo definito matrice degli accessi. In questo contesto, una sola delle seguenti affermazioni è vera. Quale?

☐ Le righe rappresentano i sotto-sistemi in cui il sistema globale è decomposto, le colonne rappresentano le possibili modalità di accesso (lettura, scrittura ecc.) e nelle celle ci sono gli elenchi degli utenti che hanno accesso ad ogni sotto-sistema nelle varie modalità

☐ Le righe rappresentano le classi a cui vogliamo controllare l'accesso, le colonne rappresentano gli oggetti istanze di tali classi, e nelle celle ci sono gli elenchi degli utenti che hanno accesso a tali oggetti

☐ Le righe rappresentano gli attori, le colonne rappresentano le classi a cui vogliamo controllare l'accesso

☐ La matrice è triangolare: sia le righe che le colonne rappresentano gli attori, e nelle celle c'è l'informazione relativa ai permessi reciproci di accesso

05. Un possibile approccio per rappresentare la matrice degli accessi è la lista delle capabilities. Descrivine la struttura.

06. Che cos'è, nel contesto del system design, il flusso del controllo globale? Fornisci un esempio.

07. Quali problematiche vengono affrontate nell'attività progettuale che nel corso è stata indicata come "controllo degli accessi", nel contesto del system design?

08. Che cos'è una matrice degli accessi? Descrivine la struttura.

09. Un possibile approccio per rappresentare la matrice degli accessi è la tabella globale degli accessi. Descrivine la struttura.

10. Un possibile approccio per rappresentare la matrice degli accessi è la lista di controllo degli accessi. Descrivine la struttura.

11. I possibili approcci per rappresentare la matrice degli accessi sono:

- **la tabella globale degli accessi**
- **la lista di controllo degli accessi**
- **la lista delle capabilities.**

Confrontali dal punto di vista delle prestazioni.

12. Descrivi il meccanismo basato su threads per il controllo del flusso delle azioni.

13. Descrivi il meccanismo procedurale di controllo del flusso delle azioni.

14. Descrivi il meccanismo guidato da eventi per il controllo del flusso delle azioni.

15. In quale situazione il meccanismo procedurale di controllo del flusso delle azioni si dimostra particolarmente utile? In quale situazione, invece, non andrebbe adottato?

16. Confronta il meccanismo di controllo del flusso basati su eventi e con quello basato su threads. Sotto quali aspetti il primo è preferibile al secondo?

Lezione 026

01. In merito alla verifica di completezza del system design, una delle domande che seguono non contribuisce a determinare se ogni aspetto problematico del system design è stato affrontato. Quale?

- [ ] È definita una politica degli accessi per ogni attore?
- [ ] È stata effettuata un'accurata rivisitazione di tutti i casi d'uso, per determinare se manca la descrizione di qualche funzionalità?
- [ ] Ad ogni caso d'uso è stato assegnato un oggetto control?
- [ ] Tutte le condizioni boundary sono state gestite?

02. In merito alla verifica di leggibilità del system design, una delle domande che seguono non contribuisce a determinare se sviluppatori non coinvolti nel system design possono comprenderlo. Quale?

- [ ] Le entità (sotto-sistemi, classi) dotate di nome simile denotano concetti simili?
- [ ] I codici sorgente degli oggetti entity sono adeguatamente commentati?
- [ ] I nomi dei sotto-sistemi sono comprensibili?
- [ ] Le entità (sotto-sistemi, classi) sono tutte descritte allo stesso livello di dettaglio?

03. In merito alla verifica di realismo del system design, una delle domande che seguono non contribuisce a determinare se il sistema corrispondente può essere realizzato. Quale?

- [ ] Le collezioni di oggetti vengono scambiate tra i sotto-sistemi in modo coerente?
- [ ] I requisiti di prestazione e di affidabilità sono stati rivisti nel contesto della decomposizione in sotto-sistemi?
- [ ] Nel sistema sono incluse nuove tecnologie o componenti? Se sì, di queste componenti o tecnologie sono state valutate l'appropriatezza e la robustezza? Come?
- [ ] Le problematiche di concorrenza (contese, deadlock) sono state affrontate?

04. In merito alla gestione delle eccezioni, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Nel caso di interruzione della connessione internet, il sistema salva il suo stato temporaneo per poterlo recuperare al ritorno online.
- [ ] Nell'identificare le condizioni boundary a livello di sotto-sistema, gli sviluppatori possono scrivere casi d'uso boundary che specificano come ogni singolo oggetto interagirà con l'evento.
- [ ] Nel caso di un errore dell'utente, il sistema deve visualizzare un messaggio significativo e comprensibile che faciliti un input corretto.
- [ ] Nell'identificare le condizioni boundary, gli sviluppatori possono progettare componenti in grado di tollerare eventuali avarie.

05. Relativamente all'attività di revisione del system design, una delle affermazioni che seguono non è corretta. Quale?

- [ ] Il processo di revisione può essere affidato a sviluppatori che lavorano in un altro progetto.
- [ ] E' importante incentivare i revisori, e fornire loro strumenti atti a scoprire e riportare i problemi progettuali.
- [ ] Il processo di revisione può essere affidato a sviluppatori che non erano stati coinvolti nel system design.
- [ ] Il processo di revisione può essere affidato agli utenti finali.

06. In merito alla verifica di correttezza del system design, una delle domande che seguono non contribuisce a determinare se il modello dell'analisi può essere mappato sul modello del system design. Quale?

- [ ] Ogni obiettivo del design corrisponde a un requisito non-funzionale?
- [ ] Ogni caso d'uso può essere mappato su un insieme di sotto-sistemi?
- [ ] Ogni sotto-sistema può essere ricondotto a un caso d'uso o a un requisito non-funzionale?
- [ ] Tutti i sotto-sistemi hanno adeguate definizioni?

07. In merito alla verifica di coerenza del system design, una delle domande che seguono non contribuisce a determinare se il modello non contiene contraddizioni. Quale?

- [ ] Ci sono sotto-sistemi o, all'interno di un sotto-sistema, classi con lo stesso nome?
- [ ] Ci sono obiettivi progettuali che violano uno o più requisiti non-funzionali?
- [ ] Sono state assegnate priorità agli obiettivi progettuali in reciproco conflitto?
- [ ] Le politiche degli accessi sono tutte coerenti coi requisiti di sicurezza?

08. Quali situazioni vengono descritte dai boundary use cases? Cita almeno un esempio.

09. Che cos'è un'eccezione? Elenca le cause possibili.

10. Seleziona il meccanismo di flusso di controllo che ritieni più appropriato per un Web server che deve sostenere carichi di lavoro elevati (molti accessi contemporanei) . Motiva la tua scelta.

11. Seleziona il meccanismo di flusso di controllo che ritieni più appropriato per l'interfaccia a manipolazione diretta (GUI) di un word processor . Motiva la tua scelta.

12. Seleziona il meccanismo di flusso di controllo che ritieni più appropriato per un sistema di controllo in tempo reale dotato di vari sensori ed attuatori . Motiva la tua scelta.

13. Considera un sistema che include un Web server e un server di database. Gli utenti usano un Web browser per accedere ai dati attraverso il Web server, oppure possono usare un client proprietario che accede direttamente al database. Disegna un diagramma di deployment UML che rappresenta la mappatura hardware/software del sistema.

Lezione 027

01. In merito alla fase di object design, una delle affermazioni che seguono non è corretta. Quale?

☐ In fase di object design è possibile identificare nuovi oggetti del dominio delle soluzioni .

☐ In fase di object design è necessario identificare gli attori coinvolti nei casi d'uso eccezionali.

☐ In fase di object design vengono adattati e configurati componenti standard .

☐ In fase di object design è necessario specificare con precisione le interfacce di ogni sotto-sistema e classe.

02. Quale, tra le attività elencate nel seguito, non viene svolta nella fase di object design?

☐ La specifica dettagliata dei servizi, che descrive con precisione l'interfaccia esposta da ogni classe .

☐ Il riuso di componenti standard e di design patterns per sfruttare soluzioni esistenti

☐ La ristrutturazione e ottimizzazione del modello degli oggetti, che interviene sul modello dell'object design migliorandone la comprensibilità e l'estendibilità , e lo modifica ulteriormente per applicare criteri di miglioramento delle prestazioni.

☐ La mappatura dei sottosistemi nelle componenti hardware della piattaforma .

03. Nel contesto dell'object design, descrivi l'attività di ottimizzazione del modello del sistema.

04. Nel contesto dell'object design, descrivi l'attività di riuso.

05. La fase di object design prevede quattro tipi di attività, presentate qui in ordine alfabetico:

- **ottimizzazione del modello**
- **ristrutturazione del modello**
- **riuso di componenti e patterns**
- **specifica dettagliata delle interfacce**

In che ordine vengono eseguite? Perché?

06. Nel contesto dell'object design, descrivi l'attività di ristrutturazione del modello del sistema.

07. Nel contesto dell'object design, descrivi l'attività di specifica dettagliata delle interfacce.

Lezione 028

01. Con quali scopi delega ed ereditarietà possono essere usate insieme? Una delle risposte che seguono non è corretta. Quale?

- ☐ Per incapsulare codice legacy preesistente e solo parzialmente riusabile .
- ☐ Per ottimizzare il riuso di operazioni private a livello di codice.
- ☐ Per disaccoppiare classi che specificano una politica da classi che realizzano il meccanismo.
- ☐ Per disaccoppiare interfacce astratte dalle relative implementazioni .

02. Che cosa si intende con il termine legacy code?

- ☐ Un database con molti dati obsoleti, e in quanto tale di scarsa utilità.
- ☐ Un sistema, un programma o un componente software correlato ad un ambiente operativo obsoleto ma ancora in uso in un'organizzazione.
- ☐ Un sistema applicativo nel dominio delle attività legali e forensi.
- ☐ Un sistema, un programma o un componente software che fa pesante uso dell'ereditarietà.

03. In merito alla problematica del riuso, una tra le affermazioni che seguono non è corretta. Quale?

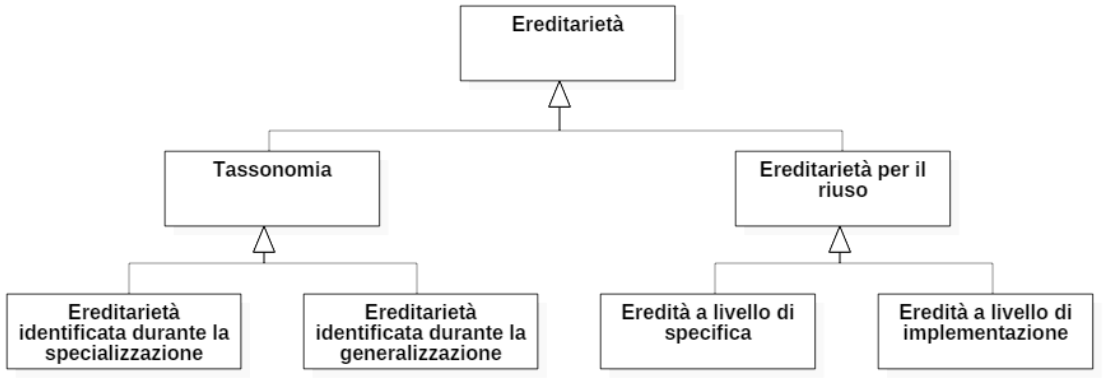
- ☐ Gli oggetti boundary sono tipicamente visibili all'utente.
- ☐ Durante il system design vengono raffinati e dettagliati oggetti dell'applicazione legati alla piattaforma hardware / software.
- ☐ La maggior parte degli oggetti entity sono oggetti dell'applicazione indipendenti da uno specifico sistema.
- ☐ Gli oggetti control sono tipicamente oggetti della soluzione.

04. Qual è la ragione per cui usiamo l'ereditarietà in fase di object design? Fornisci un esempio.

05. Qual è la differenza tra gli oggetti dell'applicazione e gli oggetti della soluzione? Fornisci un esempio.

06. Quali sono le ragioni per cui usiamo l'ereditarietà e la generalizzazione a livello di specifica, nel modello dell'analisi?

07. Commenta il seguente diagramma, indicando le peculiarità dei diversi tipi di ereditarietà.



08. Quali sono gli svantaggi connessi con l'uso dell'ereditarietà? Fornisci un esempio.

09. Che cosa si intende per ereditarietà stretta? Fornisci un esempio.

10. Enuncia il principio di sostituzione di Liskov.

11. La delega sostituisce bene l'ereditarietà di un determinato tipo. Quale? Con quali vantaggi?

12. Fornisci una definizione del meccanismo di delega.

Lezione 029

01. Tra le possibili cause di cambiamento abbiamo considerato la necessità di affidarsi a nuovi fornitori e nuove tecnologie: i componenti commerciali sono spesso sostituiti da equivalenti di altre marche. Quale, tra i design pattern sotto elencati, ci può aiutare a fronteggiare questa situazione?

- [ ] Composite
- [ ] Bridge
- [ ] Command
- [ ] Singleton

02. Tra le possibili cause di cambiamento abbiamo considerato la necessità di sviluppare nuove implementazioni: talvolta le prestazioni di un sistema si rivelano, in fase di test, insoddisfacenti, ed emerge la necessità di adottare strutture dati ed algoritmi più efficienti. Quale, tra i design pattern sotto elencati, ci può aiutare a fronteggiare questa situazione?

- [ ] Command
- [ ] Strategy
- [ ] Composite
- [ ] Singleton

03. Tra le possibili cause di cambiamento abbiamo considerato la necessità di sviluppare nuove funzionalità: il contesto applicativo evolve nel tempo, i committenti sviluppano nuova consapevolezza delle potenzialità della tecnologia, cambiano le normative, il sistema deve far fronte a nuovi requisiti. Quale, tra i design pattern sotto elencati, ci può aiutare a fronteggiare questa situazione?

- [ ] Command
- [ ] Adapter
- [ ] trategy
- [ ] Abstract Factory

04. Considera il design pattern Bridge. Quale, tra le finalità sotto riportate, meglio descrive il problema che tale pattern affronta?

- [ ] Disaccoppiare una classe che determina la politica di un servizio da un insieme di meccanismi che la implementano, in modo che i meccanismi possano essere cambiati senza che il client debba essere modificato.
- [ ] Isolare il client da diverse piattaforme che offrono differenti implementazioni degli stessi concetti .
- [ ] Convertire l'interfaccia di una classe esistente (legacy) in una diversa interfaccia richiesta dal client, così che il client e la classe legacy possano lavorare insieme senza subire modifiche.
- [ ] Disaccoppiare un'interfaccia dalla sua implementazione così che l'implementazione possa essere sostituite, possibilmente a run-time.

05. Considera il design pattern Adapter. Quale, tra le finalità sotto riportate, meglio descrive il problema che tale pattern affronta?

- [ ] Disaccoppiare una classe che determina la politica di un servizio da un insieme di meccanismi che la implementano, in modo che i meccanismi possano essere cambiati senza che il client debba essere modificato.
- [ ] Convertire l'interfaccia di una classe esistente (legacy) in una diversa interfaccia richiesta dal client, così che il client e la classe legacy possano lavorare insieme senza subire modifiche.
- [ ] Isolare il client da diverse piattaforme che offrono differenti implementazioni degli stessi concetti .
- [ ] Disaccoppiare un'interfaccia dalla sua implementazione così che l'implementazione possa essere sostituite, possibilmente a run-time.

06. Considera il design pattern Strategy. Quale, tra le finalità sotto riportate, meglio descrive il problema che tale pattern affronta?

- [ ] Isolare il client da diverse piattaforme che offrono differenti implementazioni degli stessi concetti .
- [ ] Disaccoppiare un'interfaccia dalla sua implementazione così che l'implementazione possa essere sostituite, possibilmente a run-time.
- [ ] Convertire l'interfaccia di una classe esistente (legacy) in una diversa interfaccia richiesta dal client, così che il client e la classe legacy possano lavorare insieme senza subire modifiche.
- [ ] Disaccoppiare una classe che determina la politica di un servizio da un insieme di meccanismi che la implementano, in modo che i meccanismi possano essere cambiati senza che il client debba essere modificato.

07. Considera il design pattern Abstract Factory. Quale, tra le finalità sotto riportate, meglio descrive il problema che tale pattern affronta?

☐ Disaccoppiare una classe che determina la politica di un servizio da un insieme di meccanismi che la implementano, in modo che i meccanismi possano essere cambiati senza che il client debba essere modificato.

☐ Isolare il client da diverse piattaforme che offrono differenti implementazioni degli stessi concetti .

☐ Disaccoppiare un'interfaccia dalla sua implementazione così che l'implementazione possa essere sostituite, possibilmente a run-time.

☐ Convertire l'interfaccia di una classe esistente (legacy) in una diversa interfaccia richiesta dal client, così che il client e la classe legacy possano lavorare insieme senza subire modifiche.

08. Perché l'uso di opportuni design patterns ci aiuta a gestire le situazioni di cambiamento?

09. Nel progettare un sistema software complesso dobbiamo prevedere le esigenze di cambiamento che emergeranno in futuro. Elenca almeno tre possibili cause di cambiamento.

Lezione 030

01. Considera il design pattern Command. Quale, tra le finalità sotto riportate, meglio descrive il problema che tale pattern affronta?

☐ Incapsulare le richieste così che possano essere eseguite, annullate o accodate indipendentemente dalla loro struttura interna.

☐ Disaccoppiare una classe che determina la politica di un servizio da un insieme di meccanismi che la implementano, in modo che i meccanismi possano essere cambiati senza che il client debba essere modificato.

☐ Isolare il client da diverse piattaforme che offrono differenti implementazioni degli stessi concetti .

☐ Convertire l'interfaccia di una classe esistente (legacy) in una diversa interfaccia richiesta dal client, così che il client e la classe legacy possano lavorare insieme senza subire modifiche.

Lezione 031

01. Considera il design pattern Composite. Quale, tra le finalità sotto riportate, meglio descrive il problema che tale pattern affronta?

☐ Rappresentare una gerarchia di ampiezza e profondità variabili, in cui i nodi foglia (terminali) e i nodi intermedi possono essere trattati uniformemente attraverso un'interfaccia comune.

☐ Isolare il client da diverse piattaforme che offrono differenti implementazioni degli stessi concetti .

☐ Disaccoppiare una classe che determina la politica di un servizio da un insieme di meccanismi che la implementano, in modo che i meccanismi possano essere cambiati senza che il client debba essere modificato.

☐ Convertire l'interfaccia di una classe esistente (legacy) in una diversa interfaccia richiesta dal client, così che il client e la classe legacy possano lavorare insieme senza subire modifiche.

02. Descrivi le caratteristiche di un blackbox framework.

03. Quali sono le principali differenze tra gli application frameworks e le librerie di classi, dal punto di vista dello sviluppatore di un'applicazione software?

04. Qual è la principale differenza tra gli application frameworks e i design patterns dal punto di vista del progettista di software?

05. Descrivi le caratteristiche di un whitebox framework.

06. Che cos'è un enterprise application framework? Quali sono le sue specificità rispetto a un generico application framework?

07. Che cos'è un middleware framework? Quali sono le sue specificità rispetto a un generico application framework?

08. Che cos'è un framework infrastrutturale? Quali sono le sue specificità rispetto a un generico application framework?

09. Che cos'è un application framework? Per che cosa serve?

Lezione 032

01. Quali attività definiscono il ruolo del class user?

02. Quali responsabilità definiscono il ruolo del class implementor?

03. Nel contesto dell'object design, che cosa si intende per specifica delle postcondizioni?

04. Nel contesto dell'object design, che cosa si intende per specifica delle precondizioni?

05. Nel contesto dell'object design, che cosa si intende per specifica delle invarianti?

06. Che cosa intendiamo per visibilità di un'operazione?

07. Che cos'è la signature di un'operazione? Fornisci un esempio.

08. Quali attività definiscono il ruolo del class extender?

09. Che cosa specifica il tipo di un attributo?

10. Fornisci una descrizione di ognuno dei livelli di visibilità di un attributo disponibili in UML.

Lezione 033

01. In merito alla specifica della visibilità di attributi e operazioni effettuata durante il raffinamento del modello dell'object design, una delle affermazioni che seguono non è corretta. Quale?

☐ La visibilità delle operazioni permette di distinguere tra operazioni che sono parte dell'interfaccia esposta dalla classe, e quelle che sono metodi di utilità cui solo la classe può accedere.

☐ La determinazione della visibilità di ogni attributo richiede di scegliere quali attributi saranno pubblici e direttamente modificabili da ogni altra classe.

☐ Nel caso di classi astratte o di classi che dovranno essere estese mediante ereditarietà, è possibile definire private quegli attributi e quelle operazioni che saranno accessibili solo dalle sottoclassi.

☐ La determinazione della visibilità di ogni attributo richiede di scegliere quali attributi dovranno essere accessibili solo indirettamente mediante le operazioni offerte dalla classe.

02. In merito ai contratti, una delle affermazioni che seguono non è corretta. Quale?

☐ Ogni class implementor o extender garantisce che siano rispettati i vincoli specificati nel contratto della classe che rende disponibile.

☐ Il contratto di una classe specifica la deadline entro cui gli sviluppatori renderanno disponibile tale classe.

☐ Ogni class user deve rispettare i vincoli specificati nel contratto per usare correttamente la classe .

☐ I contratti sono vincoli che consentono agli sviluppatori di condividere le stesse assunzioni sui servizi offerti dalla classe.

03. Uno dei termini elencati nel seguito non identifica un tipo di vincolo espresso nei contratti. Quale?

☐ Variante

☐ Invariante

☐ Precondizione

☐ Postcondizione

04. In merito agli invarianti specificati nei contratti, una delle affermazioni che seguono non è corretta. Quale?

☐ Un invariante è un predicato che è sempre vero per tutte le istanze della classe.

☐ Gli invarianti specificano vincoli sui valori costanti memorizzati nella classe.

☐ Gli invarianti sono vincoli associati con classi o interfacce.

☐ Gli invarianti sono usati per specificare vincoli di consistenza tra attributi della classe.

05. In merito alle precondizioni specificate in un contratto, una delle affermazioni che seguono non è corretta. Quale?

☐ Le precondizioni sono usate per specificare vincoli che un class implementor o un class extender deve rispettare prima di codificare l'operazione.

☐ Le precondizioni sono associate a operazioni specifiche.

☐ Le precondizioni sono usate per specificare vincoli che un class user deve rispettare prima di chiamare l'operazione.

☐ Una precondizione è un predicato che deve essere vero prima che un'operazione sia chiamata.

06. Tra le attività riportate nel seguito, una non è connessa con la specifica delle interfacce in fase di object design. Quale?

☐ Specifica della visibilità .

☐ Specifica di invarianti, precondizioni e postcondizioni .

☐ Identificazione degli oggetti entity.

☐ Specifica di tipi e signatures .

07. Una delle attività che seguono non appartiene alla specifica di tipi e signatures effettuata durante il raffinamento del modello dell'object design. Quale?

☐ Identificazione della corrispondenza delle classi e degli attributi del modello degli oggetti con i tipi built-in del linguaggio di programmazione adottato.

☐ Rappresentazione delle relazioni tra classi mediante componenti esistenti.

☐ Identificazione del rango (range) di ogni attributo dell'oggetto.

☐ Identificazione di un design pattern riusabile per l'implementazione della classe.

08. In merito alle postcondizioni specificate in un contratto, una delle affermazioni che seguono non è corretta. Quale?

☐ Una postcondizione è un predicato che deve essere vero dopo una chiamata ad un'operazione.

☐ Le postcondizioni sono associate a operazioni specifiche.

☐ Le postcondizioni sono usate per specificare vincoli che un class implementor o un class extender deve garantire dopo che l'operazione è stata chiamata.

☐ Le postcondizioni sono usate per specificare vincoli che un class user deve rispettare dopo aver chiamato l'operazione.

09. Considera una classe Esame che rappresenta un esame universitario di un dato insegnamento; tra le varie operazioni possibili, la classe offre l'operazione iscrivi(Studente), che inserisce un'iscrizione. Fornisci un esempio di postcondizione relativamente a tale operazione.

10. Considera una classe Esame che rappresenta un esame universitario di un dato insegnamento; tra le varie operazioni possibili, la classe offre l'operazione iscrivi(Studente), che inserisce un'iscrizione. Fornisci un esempio di invariante relativamente a tale operazione.

11. Che cos'è OCL (Object Constraint Language)?

12. Come si rappresenta graficamente in UML un vincolo su un elemento del modello?

13. Considera una classe Esame che rappresenta un esame universitario di un dato insegnamento; tra le varie operazioni possibili, la classe offre l'operazione iscrivi(Studente), che inserisce un'iscrizione. Fornisci un esempio di precondizione relativamente a tale operazione.

Lezione 034

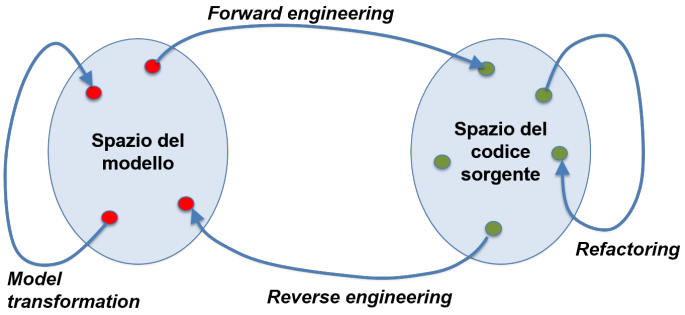
01. In merito alle trasformazioni che consentono di mappare nel codice il modello del sistema, uno dei termini sotto riportati non rappresenta un tipo di trasformazione presentato nel corso. Quale?

- ☐ La realizzazione delle associazioni.
- ☐ La mappatura dei contratti con lo schema del database .
- ☐ La mappatura dei contratti con le eccezioni .
- ☐ L'ottimizzazione .

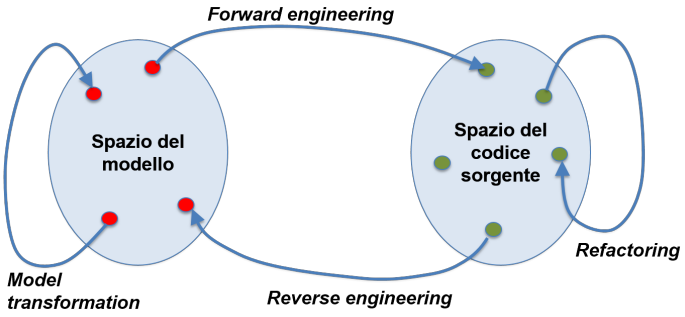
02. Una delle attività riportate qui sotto non fa parte delle attività di ottimizzazione. Quale?

- ☐ Eliminare dal codice i commenti per ridurre le dimensioni dei file sorgenti.
- ☐ Ridurre la molteplicità delle associazioni per velocizzare le queries .
- ☐ Aggiungere associazioni ridondanti per aumentare l'efficienza del sistema.
- ☐ Aggiungere attributi derivati da altri attributi per migliorare i tempi di accesso agli oggetti.

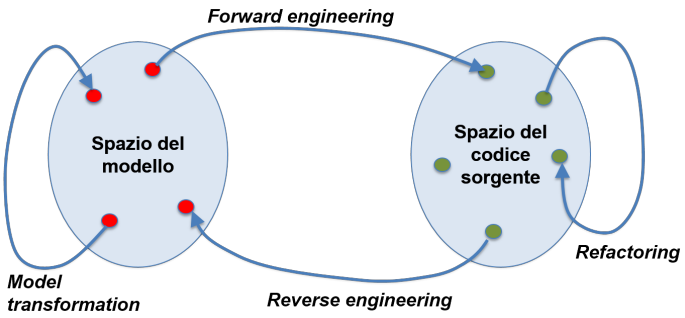
03. Il diagramma qui sotto mostra dominio e codominio di quattro tipi di trasformazione relativi al mapping:
Descrivi lo scopo e fornisci un esempio di model transformation.



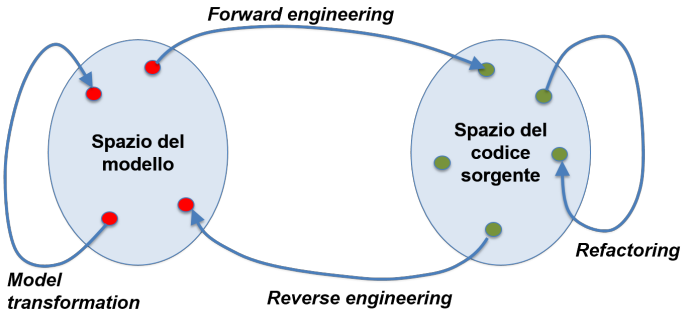
04. Il diagramma qui sotto mostra dominio e codominio di quattro tipi di trasformazione relativi al mapping:
Descrivi lo scopo e le caratteristiche del refactoring.



05. Il diagramma qui sotto mostra dominio e codominio di quattro tipi di trasformazione relativi al mapping:
Descrivi lo scopo le caratteristiche del forward engineering.



06. Il diagramma qui sotto mostra dominio e codominio di quattro tipi di trasformazione relativi al mapping:
Descrivi lo scopo e le caratteristiche del reverse engineering.



Lezione 035

01. In merito alla mappatura delle associazioni UML con le collezioni disponibili nei linguaggi OO, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I linguaggi OO rendono disponibili le references, in cui un oggetto memorizza un riferimento (handle) unidirezionale a un altro oggetto .
- [ ] Le associazioni sono concetti UML che denotano insiemi di collegamenti (link) bidirezionali tra due o più oggetti.
- [ ] I linguaggi OO rendono disponibili le collezioni, oggetti che contengono riferimenti ad altri oggetti.
- [ ] Le collezioni sono sempre ordinate.

02. Nel corso vengono presentate alcune euristiche per mappare contratti ed eccezioni. Una delle indicazioni che seguono non fa parte di tali euristiche. Quale?

- [ ] Riusa il codice di controllo dei vincoli, incapsulandolo in metodi riusabili.
- [ ] Ometti il codice di controllo per le precondizioni.
- [ ] Concentrati sulle interfacce dei sotto-sistemi e ometti il codice di controllo dei metodi protetti e privati.
- [ ] Concentrati sui contratti dei componenti più duraturi, ad es. gli oggetti entity.

03. In merito alla mappatura dei contratti con le eccezioni , quale costrutto Java permette di segnalare la violazione di un contratto?

- [ ] Il metodo switch/case.
- [ ] L'istruzione throw.
- [ ] L'interfaccia Runnable.
- [ ] La classe Catch.

04. Una delle attività elencate nel seguito non è inquadrabile nell'ottimizzazione dell'object design. Quale?

- [ ] Anticipare le elaborazioni pesanti.
- [ ] Collassare alcuni oggetti in attributi.
- [ ] Aggiungere associazioni per ottimizzare i percorsi di accesso.
- [ ] Salvare nella memoria cache i risultati delle elaborazioni pesanti.

05. Nel contesto delle trasformazioni di mapping, descrivi il principio di unicità del criterio.

06. Nel contesto delle trasformazioni di mapping, descrivi il principio di validazione.

07. Nel contesto delle trasformazioni di mapping, descrivi il principio di località.

08. Nel contesto delle trasformazioni di mapping, descrivi il principio di isolamento.

Lezione 036

01. Nel contesto della mappatura del modello degli oggetti in uno schema relazionale per la persistenza dei dati, come vengono mappate le relazioni di ereditarietà?

02. Nel contesto della mappatura del modello degli oggetti in uno schema relazionale per la persistenza dei dati, come vengono mappate le associazioni?

03. Nel contesto della mappatura del modello degli oggetti in uno schema relazionale per la persistenza dei dati, che cos'è una chiave esterna?

04. Nel contesto della mappatura del modello degli oggetti in uno schema relazionale per la persistenza dei dati, che cos'è una chiave primaria?

05. Descrivi le caratteristiche di uno schema relazionale per la persistenza dei dati.

06. Nel contesto della mappatura del modello degli oggetti in uno schema relazionale per la persistenza dei dati, come vengono mappate le classi?

Lezione 037

01. Un concetto importante nel contesto del testing è quello di componente di test. Quale tra le seguenti definizioni è corretta?

- [ ] Un componente di test un'implementazione parziale di un test stub che dipende dal componente da testare.
- [ ] Un componente di test è una parte del sistema che può essere testata in isolamento.
- [ ] Un componente di test è un'implementazione parziale di un test driver da cui il componente da testare dipende.
- [ ] Un componente di test è un insieme di dati in ingresso e di risultati attesi che esercitano un test case per far emergere i difetti.

02. Con riferimento allo unit testing, solo una delle definizioni che seguono è corretta. Quale?

- [ ] Lo unit testing trova le differenze tra il modello dei casi d'uso e il sistema.
- [ ] Lo unit testing trova le differenze tra il modello del system design e alcuni sotto-sistemi integrati.
- [ ] Lo unit testing trova le differenze tra i requisiti non-funzionali e le prestazioni reali del sistema.
- [ ] Lo unit testing trova le differenze tra la specificazione di un oggetto e la sua realizzazione come componente.

03. Un concetto importante nel contesto del testing è quello di test case. Quale tra le seguenti definizioni è corretta?

- [ ] Un test case è un'implementazione parziale di un componente che dipende dal componente da testare.
- [ ] Un test case è una parte del sistema che può essere testata in isolamento.
- [ ] Un test case è un insieme di dati in ingresso e di risultati attesi che esercitano un componente di test per far emergere i suoi difetti.
- [ ] Un test case è un'implementazione parziale di un componente da cui il componente da testare dipende.

04. In merito al concetto di testing, una delle affermazioni che seguono non è adeguata. Quale?

- [ ] Il testing è il tentativo sistematico e pianificato di trovare difetti in un sistema software.
- [ ] Un test ha successo quando evidenzia un difetto.
- [ ] Il sistema è pronto per il rilascio quando nessuno dei test riesce a falsificarne il comportamento.
- [ ] Il testing è il processo teso a dimostrare che un sistema software è privo di difetti.

05. Con riferimento al testing strutturale, solo una delle definizioni che seguono è corretta. Quale?

- [ ] Il testing strutturale trova le differenze tra il modello del system design e alcuni sotto-sistemi integrati.
- [ ] Il testing strutturale trova le differenze tra la specificazione di un oggetto e la sua realizzazione come componente.
- [ ] Il testing strutturale trova le differenze tra i requisiti non-funzionali e le prestazioni reali del sistema.
- [ ] Il testing strutturale trova le differenze tra il modello dei casi d'uso e il sistema.

06. Con riferimento al testing funzionale, solo una delle definizioni che seguono è corretta. Quale?

- [ ] Il testing funzionale trova le differenze tra il modello del system design e alcuni sotto-sistemi integrati.
- [ ] Il testing funzionale trova le differenze tra la specificazione di un oggetto e la sua realizzazione come componente.
- [ ] Il testing funzionale trova le differenze tra il modello dei casi d'uso e il sistema.
- [ ] Il testing funzionale trova le differenze tra i requisiti non-funzionali e le prestazioni reali del sistema.

07. Con riferimento ai concetti alla base del testing, solo una delle affermazioni che seguono è corretta. Quale?

- [ ] L'affidabilità del software è una misura del tempo minimo di funzionamento del sistema senza che si presenti un comportamento errato.
- [ ] L'affidabilità del software è la frequenza con cui si presenta un comportamento errato del sistema.
- [ ] L'affidabilità del software è una misura del tempo medio di funzionamento del sistema senza che si presenti un comportamento errato.
- [ ] L'affidabilità del software è la probabilità che un sistema software non causi un comportamento errato del sistema per un tempo specificato sotto specifiche condizioni.

08. Con riferimento ai concetti alla base del testing, solo una delle definizioni che seguono è corretta. Quale?

- [ ] Una failure è una qualunque deviazione del comportamento osservato da quello specificato.
- [ ] Una failure è uno stato del sistema che precede un errore.
- [ ] Una failure è un difetto meccanico o algoritmico che causa uno stato errato.
- [ ] Una failure è un comportamento imprevisto del sistema che non è possibile correggere.

09. Con riferimento ai concetti alla base del testing, solo una delle definizioni che seguono è corretta. Quale?

- [ ] Un errore (o stato errato) è una qualunque deviazione del comportamento osservato da quello specificato.
- [ ] Un errore (o stato errato) è un difetto meccanico o algoritmico che causa un fault.
- [ ] Un errore (o stato errato) un comportamento imprevisto del sistema che non è possibile correggere.
- [ ] Un errore (o stato errato) è uno stato del sistema che precede una failure.

10. Con riferimento ai concetti alla base del testing, solo una delle definizioni che seguono è corretta. Quale?

- [ ] Un fault (o bug) è un comportamento imprevisto del sistema che non è possibile correggere.
- [ ] Un fault (o bug) è è una qualunque deviazione del comportamento osservato da quello specificato.
- [ ] Un fault (o bug) è è un difetto meccanico o algoritmico che causa uno stato errato.
- [ ] Un fault (o bug) è è uno stato del sistema che precede una failure.

11. Con riferimento al performance testing, solo una delle definizioni che seguono è corretta. Quale?

- [ ] Il performance testing trova le differenze tra il modello dei casi d'uso e il sistema.
- [ ] Il performance testing trova le differenze tra i requisiti non-funzionali e le prestazioni reali del sistema.
- [ ] Il performance testing trova le differenze tra il modello del system design e alcuni sotto-sistemi integrati.
- [ ] Il performance testing trova le differenze tra la specificazione di un oggetto e la sua realizzazione come componente.

12. Un concetto importante nel contesto del testing è quello di test stub. Quale tra le seguenti definizioni è corretta?

- [ ] Un test stub è un insieme di dati in ingresso e di risultati attesi che esercitano un componente di test per far emergere i suoi difetti.
- [ ] Un test stub è una parte del sistema che può essere testata in isolamento.
- [ ] Un test stub è un'implementazione parziale di un componente da cui il componente da testare dipende.
- [ ] Un test stub è un'implementazione parziale di un componente che dipende dal componente da testare.

13. Un concetto importante nel contesto del testing è quello di test driver. Quale tra le seguenti definizioni è corretta?

- [ ] Un test driver è un'implementazione parziale di un componente da cui il componente da testare dipende.
- [ ] Un test driver è un'implementazione parziale di un componente che dipende dal componente da testare.
- [ ] Un test driver è un insieme di dati in ingresso e di risultati attesi che esercitano un componente di test per far emergere i suoi difetti.
- [ ] Un test driver è una parte del sistema che può essere testata in isolamento.

14. Che cos'è un white-box test?

15. Che cos'è un black-box test?

16. Quali relazioni possono sussistere tra due test cases?

17. Elenca gli attributi che descrivono un test case.

18. Nel contesto del fault detection, quali sono le caratteristiche della tecnica denominata debugging? Quali tipi di debugging sono presentati nel corso?

19. Nel contesto del fault detection, quali sono le caratteristiche della tecnica denominata review? Quali tipi di review sono presentati nel corso?

20. Nel contesto delle tecniche tese ad aumentare l'affidabilità di un sistema, che cosa intendiamo con fault tolerance?

21. **Nel contesto delle tecniche tese ad aumentare l'affidabilità di un sistema, che cosa intendiamo con fault detection?**

22. **Nel contesto delle tecniche tese ad aumentare l'affidabilità di un sistema, che cosa intendiamo con fault avoidance?**

23. **Che cosa intendiamo con il termine test di regressione?**

Lezione 038

01. Uno dei termini sotto riportati non identifica un tipo di test di usabilità. Quale?

- [ ] Test di caso d'uso
- [ ] Test di prodotto
- [ ] Test di prototipo
- [ ] Test di scenario

02. Uno degli obiettivi elencati nel seguito non rappresenta una motivazione dell'attività di unit testing. Quale?

- [ ] Facilitare l'identificazione e la correzione dei difetti, perché pochi componenti sono coinvolti nel test .
- [ ] Ricercare i difetti nei meccanismi con cui componenti diversi interagiscono e si scambiano dati .
- [ ] Ridurre la complessità delle attività di test, grazie al fatto che lo unit test si concentra su componenti di dimensioni limitate
- [ ] Consentire il parallelismo nelle attività di test: ogni componente può essere testato indipendentemente dagli altri.

03. In merito ai test di usabilità, una delle affermazioni che seguono non è corretta. Quale?

- [ ] I test di usabilità adottano un approccio sistematico confrontando il comportamento del sistema con i requisiti non-funzionali del progetto.
- [ ] I test di usabilità indagano il livello di comprensione che l'utente ha del sistema.
- [ ] I test di usabilità indagano il look-and-feel dell'interfaccia utente, il layout geometrico delle videate, le sequenze di interazioni e, in determinati casi, l'ergonomia del sistema.
- [ ] I test di usabilità cercano di determinare le differenze tra il sistema e le aspettative degli utenti.

04. Fornisci una descrizione della tecnica di state-based testing.

05. Fornisci una descrizione della tecnica di equivalence testing.

06. Fornisci una descrizione della tecnica di path testing.

07. Fornisci una descrizione della tecnica di boundary testing.

08. Elenca i passi in cui si articola l'attività di ispezione dei componenti nel contesto del testing.

Lezione 039

01. Il sandwich testing è una delle possibili strategie su cui si basano i test di integrazione orizzontali. Quale, tra le seguenti descrizioni, meglio si adatta al sandwich testing?

☐ Il sandwich testing mappa la decomposizione in sotto-sistemi su tre livelli: un obiettivo, un livello superiore e uno inferiore. In questo modo non è necessario definire drivers e stubs per i livelli superiore ed inferiore.

☐ Il sandwich testing assume che tutti i componenti siano già stati testati individualmente, e quindi possano essere testati insieme come un unico sistema, in un unico passo di integrazione.

☐ Il sandwich testing parte dal livello più alto, testando uno alla volta i suoi sotto-sistemi con l'ausilio di stubs che simulano i componenti dei livelli sottostanti. Quando tutti i componenti di un livello sono stati testati, si integrano (sempre uno alla volta) i componenti del livello successivo, e così via. Non sono necessari test drivers.

☐ Il sandwich testing parte testando individualmente ogni componente appartenente al layer più basso dell'architettura, poi procede salendo di un livello e integrando i sotto-sistemi costituiti dal secondo livello, poi sale al terzo ecc.

02. Il top-down testing è una delle possibili strategie su cui si basano i test di integrazione orizzontali. Quale, tra le seguenti descrizioni, meglio si adatta al top-down testing?

☐ Il top-down testing parte testando individualmente ogni componente appartenente al layer più basso dell'architettura, poi procede salendo di un livello e integrando i sotto-sistemi costituiti dal secondo livello, poi sale al terzo ecc.

☐ Il top-down testing mappa la decomposizione in sotto-sistemi su tre livelli: un obiettivo, un livello superiore e uno inferiore. In questo modo non è necessario definire drivers e stubs per i livelli superiore ed inferiore.

☐ Il top-down testing parte dal livello più alto, testando uno alla volta i suoi sotto-sistemi con l'ausilio di stubs che simulano i componenti dei livelli sottostanti. Quando tutti i componenti di un livello sono stati testati, si integrano (sempre uno alla volta) i componenti del livello successivo, e così via. Non sono necessari test drivers.

☐ Il top-down testing assume che tutti i componenti siano già stati testati individualmente, e quindi possano essere testati insieme come un unico sistema, in un unico passo di integrazione.

03. Nel contesto dei test di integrazione, una delle affermazioni che seguono non è corretta. Quale?

☐ L'ordine con cui i componenti sono testati non influenza l'impegno totale di lavoro necessario per il test d'integrazione.

☐ Questa procedura permette di testare parti del sistema sempre più complesse circoscrivendo la localizzazione di possibili difetti.

☐ L'integration testing rivela i difetti che non sono stati evidenziati in fase di unit testing, prendendo in considerazione insiemi di componenti.

☐ Si inizia testando due componenti insieme; quando non appaiono difetti un terzo componente viene aggiunto al gruppo, e così via.

04. Il big bang testing è una delle possibili strategie su cui si basano i test di integrazione orizzontali. Quale, tra le seguenti descrizioni, meglio si adatta al big bang testing?

☐ Il big bang testing assume che tutti i componenti siano già stati testati individualmente, e quindi possano essere testati insieme come un unico sistema, in un unico passo di integrazione.

☐ Il big bang testing parte dal livello più alto, testando uno alla volta i suoi sotto-sistemi con l'ausilio di stubs che simulano i componenti dei livelli sottostanti. Quando tutti i componenti di un livello sono stati testati, si integrano (sempre uno alla volta) i componenti del livello successivo, e così via. Non sono necessari test drivers.

☐ Il big bang testing parte testando individualmente ogni componente appartenente al layer più basso dell'architettura, poi procede salendo di un livello e integrando i sotto-sistemi costituiti dal secondo livello, poi sale al terzo ecc.

☐ Il big bang testing mappa la decomposizione in sotto-sistemi su tre livelli: un obiettivo, un livello superiore e uno inferiore. In questo modo non è necessario definire drivers e stubs per i livelli superiore ed inferiore.

05. Il bottom-up testing è una delle possibili strategie su cui si basano i test di integrazione orizzontali. Quale, tra le seguenti descrizioni, meglio si adatta al bottom-up testing?

☐ Il bottom-up testing parte dal livello più alto, testando uno alla volta i suoi sotto-sistemi con l'ausilio di stubs che simulano i componenti dei livelli sottostanti. Quando tutti i componenti di un livello sono stati testati, si integrano (sempre uno alla volta) i componenti del livello successivo, e così via. Non sono necessari test drivers.

☐ Il bottom-up testing assume che tutti i componenti siano già stati testati individualmente, e quindi possano essere testati insieme come un unico sistema, in un unico passo di integrazione.

☐ Il bottom-up testing mappa la decomposizione in sotto-sistemi su tre livelli: un obiettivo, un livello superiore e uno inferiore. In questo modo non è necessario definire drivers e stubs per i livelli superiore ed inferiore.

☐ Il bottom-up testing parte testando individualmente ogni componente appartenente al layer più basso dell'architettura, poi procede salendo di un livello e integrando i sotto-sistemi costituiti dal secondo livello, poi sale al terzo ecc.

06. Qual è il principale vantaggio dei test di integrazione verticali? Qual è lo svantaggio?

07. Elenca le principali attività connesse con il system testing.