ESONERO SAS DEL 15/06/21

DOMANDA 1

Dire se le affermazioni sono vere o false.

1) Un attributo di una classe software rappresenta una responsabilità "di conoscenza" per le istanze di tale classe:
   VERA

2) Un progetto orientato agli oggetti vede un sistema come una comunità di oggetti con responsabilità che collaborano:
   VERA

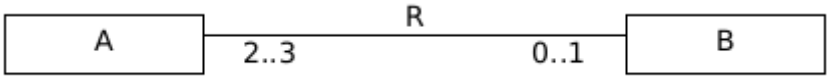
3) L'utilizzo di pattern GRASP è finalizzato alla scoperta dei requisiti del sistema:
   FALSA

4) I pattern GRASP sono alla base dello sviluppo guidato dalle responsabilità:
   VERA

5) La decisione sull'assegnazione delle responsabilità è svolta attraverso i soli diagrammi di classe:
   FALSA

DOMANDA 2

Si consideri il diagramma seguente:



Si supponga A = {a1 , a2 , a3 , a4} e B = {b1 , b2}.
Dire quali delle seguenti affermazioni sono vere e quali false.

1) R può essere {(a1, b1), (a2, b1), (a3, b1), (a4, b2)}.
   FALSA

2) R può essere {(a1, b1), (a2, b2)}.
   FALSA

3) R può essere {(a1, b1), (a2, b2), (a3, b1), (a4, b2)}.
   VERA

4) R può essere {(a1, b1), (a2, b2), (a2, b2), (a3, b2)}.
   FALSA

DOMANDA 3

Dire se le affermazioni sono vere o false.

1) I requisiti non funzionali sono descritti completamente dai casi d'uso.
   FALSA

2) Un caso d'uso rappresenta l'insieme di funzionalità di un sistema.
   FALSA

3) In UP, la disciplina dei requisiti ha l'obiettivo di produrre una lista dei requisiti, capire il contesto del sistema, catturare i requisiti funzionali e i requisiti non-funzionali.
   VERA

4) I requisiti funzionali non catturati dai casi d'uso vengono descritti nelle Specifiche Supplementari.
   FALSA

5) Un caso d'uso rappresenta una maniera di utilizzare il sistema da parte di un utente per raggiungere un suo obiettivo.
   VERA

DOMANDA 4

Si considerino i seguenti problemi di progettazione:

1. Come progettare per gestire un insieme di algoritmi o politiche variabili ma correlati? Come progettare per consentire di modificare questi algoritmi o politiche?
2. Come permettere di assegnare una o più responsabilità addizionali ad un oggetto in maniera dinamica ed evitare il problema della relazione statica? Come provvedere un'alternativa più flessibile al meccanismo di sottoclasse ed evitare il problema di avere una gerarchia di classi complessa?
3. Come gestire interfacce incompatibili, o fornire un'interfaccia stabile a comportamenti simili ma con interfacce diverse?

Associare la corrispondente soluzione progettuale.

1) Definisci un'interfaccia subscriber o listener (ascoltatore). Gli oggetti subscriber implementano questa interfaccia. Il publisher registra dinamicamente i subscriber che sono interessati ai suoi eventi, e li avvisa quando si verifica un evento.
   NON È PRESENTE IL PROBLEMA CORRISPONDENTE

2) Converti l'interfaccia originale di un componente in un'altra interfaccia, attraverso un oggetto adattatore intermedio.
   PROBLEMA (3)

3) Inglobare l'oggetto all'interno di un altro che aggiunge le nuove funzionalità.
   PROBLEMA (2)

4) Definisci ciascuna strategia in una classe separata, con un'interfaccia comune.
   PROBLEMA (1)

DOMANDA 5

Dire quali delle seguenti affermazioni sono vere e quali false.

1) Il modello di dominio è un dizionario visuale delle classi concettuali.
   VERA

2) Il modello di dominio include la definizione di oggetti, associazioni e attributi di classi software.
   FALSA

3) L'analisi linguistica dei casi d'uso nel formato dettagliato è una fonte di ispirazione per la costruzione del modello di dominio.
   VERA

4) Il modello di dominio riporta i concetti significativi relativi ai casi d'uso.
   VERA

5) Il modello di dominio è una rappresentazione testuale delle classi concettuali, oggetti reali del dominio.
   FALSA

DOMANDA 6

Dire quali delle seguenti affermazioni sono vere e quali false.

1) Il meccanismo di delega è preferibile al meccanismo di specializzazione per il riuso del codice.
   VERA

2) I pattern GoF prediligono l'utilizzo del meccanismo di ereditarietà per ottenere il polimorfismo.
   VERA

3) I pattern GoF incentivano l'uso dell'ereditarietà come meccanismo di riuso del codice, in particolare attraverso il pattern composite.
   FALSA

4) La composizione di oggetti è un meccanismo di riuso del codice detto black-box.
   VERA

5) L'ereditarietà rispetta l'incapsulamento.
   FALSA