Ingegneria del Software a.a. 2014-15
Prova Scritta del 4 febbraio 2015

Esercizio di sbarramento

COGNOME **NOME** **MATRICOLA**

Rispondere alle seguenti domande. Per ogni domanda, solo una soluzione è corretta. L'esercizio si ritiene superato se si risponde correttamente ad **almeno 6 domande**, la valutazione è di 1 punto per ogni risposta corretta oltre le 6.

Domanda 1

Quale tra le seguenti architetture software trovi essere più adatta per sviluppare un IDE (integrated development environment) che fornisce anche un editor UML e un generatore di codice a partire dal modello UML?

a) Layered model
b) Repository model
c) Pipe and Filter model
d) Model View Controller model

Domanda 2

Quale tra i design pattern visti a lezione è utilizzato spesso nello sviluppo di Framework perchè in grado di realizzare l'inversione di controllo?

a) State Pattern
b) Controller Pattern
c) Template Pattern
d) Façade Pattern

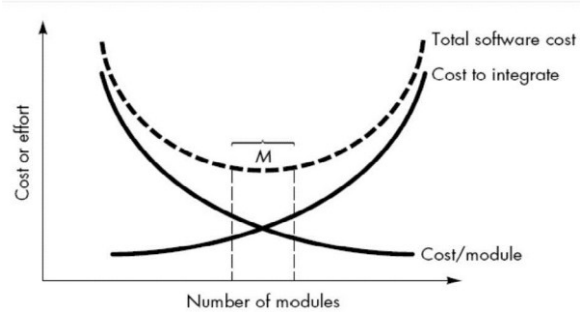
Domanda 3

Considerando i principi di buona progettazione (e programmazione) che problema/i manifesta/no la strategia denominata FORZA BRUTA per implementare un ORM?

a) Accoppia fortemente le classi di dominio al DB e produce classi poco coese
b) Viola il principio di Semplicità
c) Produce un alto Fan-in
d) Aumenta la metrica DIT (una delle metriche proposte da Chidamber & Kemerer nel contesto di software OO) sopra una soglia accettabile

Domanda 4

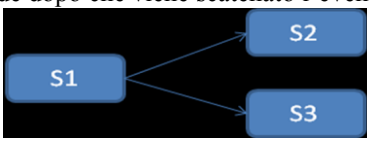
Cosa ci fa capire questa Figura?



a) Questa figura fa capire che: dividere il software in moduli piccoli fa diminuire il costo di integrazione dei moduli che lo compongono
b) Questa figura fa capire che: all' aumentare del costo di un software il numero dei moduli cresce
c) Questa figura fa capire che: dividere il software in moduli sempre più piccoli (infinite volte) fa diventare nullo il costo di sviluppo
d) Questa figura fa capire che: esiste un trade-off (rappresentato dalla regione di costo minimo M) tra il numero di volte che un problema viene decomposto (e quindi il software che lo risolve) e i costi di integrazione

Domanda 5

Data la seguente state machine UML, cosa accade dopo che viene scatenato l'evento "e"?

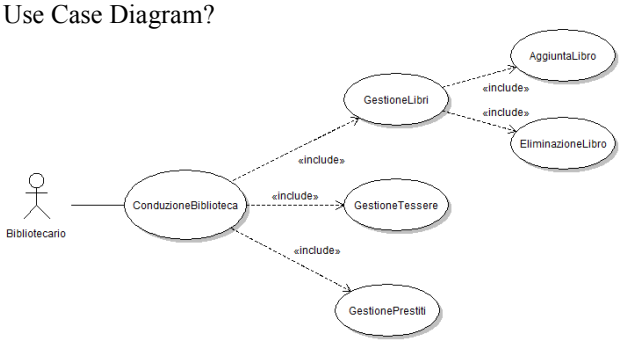


a) La macchina si troverà in S2
b) La macchina si troverà in modo non deterministico in S2 o S3
c) La macchina si troverà contemporaneamente in S2 e S3

d) Non può succedere perchè tale configurazione (cioè: due transizioni con lo stesso evento e due condizioni diverse, entrambe vere) non è ammessa in una state machine UML

Domanda 6

Come valuteresti questa porzione di Use Case Diagram?



a) Bene perchè è stato applicata la best practice di scomposizione funzionale
b) Male perchè la scomposizione funzionale è da evitare nel contesto degli Use Case Diagram
c) Bene perchè il primo use case (ConduzioneBiblioteca) è molto astratto ed ingloba tutte le funzionalità degli use case inclusi
d) Male perchè il numero di <<include>> totali nel diagramma è troppo alto

Domanda 7

Perchè è molto importante a livello di requisiti stabilire in modo chiaro e preciso i limiti/confini del sistema software che si vuole costruire?

a) Per semplificare la fase di analisi dei requisiti e il testing del sistema

b) Perchè in questo modo si evitano incomprensioni tra i vari stakeholder

c) Perchè è provato che molti problemi a progetti software reali sono sorti proprio a causa di limiti/confini non definiti bene

d) Perchè il modello dei casi d'uso prescrive di racchiudere i vari use case in un rettangolo che rappresenta i confini del sistema

Domanda 8

Come si chiama il refactoring applicato nell'esempio seguente?

```
void printOwing() {
        printBanner();

        //print details
        System.out.println ("name:      " + _name);
        System.out.println ("amount     " + getOutstanding());
}
```

⇓

```
void printOwing() {
        printBanner();
        printDetails(getOutstanding());
}

void printDetails (double outstanding) {
        System.out.println ("name:      " + _name);
        System.out.println ("amount     " + outstanding);
}
```

a) Extract method
b) Move method
c) Change method
d) Create method

Domanda 9

Quale tra le seguenti è una delle cinque leggi sull'evoluzione del software di Lehman?

a) **Refactoring continuo**: un sistema deve essere rifattorizzato continuamente per prevenire il design decay/erosion
b) **Costi in aumento**: la manutenzione del software porta inevitabilmente a "Software Monsters" che fanno crescere i costi per tenere il sistema aggiornato e utile
c) **Complessità in aumento**: i cambiamenti complicano il sistema e la struttura si deteriora (a meno di non intervenire per evitarlo)
d) **Working force**: aggiungere forza lavoro ad un progetto software in ritardo, lo farà ritardare ancora di più

Domanda 10

Si supponga di dover testare un metodo in un linguaggio OO che prende in input la rappresentazione numerica dei mesi (dove 1 corrisponde a Gennaio, 2 a Febbraio, etc.)? Quali input dovremmo selezionare per applicare correttamente il criterio **boundary value analysis**?

a) 1 e 12
b) 1, 6 e 12
c) 0, 1, 12, 13
d) 0, 1, 2, 11, 12, 13