Ingegneria del Software B

Allievi della Laurea Specialistica in Ingegneria Informatica
Tema d'esame - 7 Settembre 2007 – ore 10.00-11.30

NOME: …… COGNOME: …….
MATRICOLA: …….. FIRMA: ……..

Le risposte chiuse valgono 1/30 ciascuna. Il valore degli esercizi è riportato nel prospetto a lato.

Esercizio	**1a**	**1b**	**1c**	**2**	**3**	**4**	**5a**	**5b**
Valore	1	2	2	2	3	4	1,5	2,5
Valutazione								

Risposte chiuse

Affermazione	**Vera o falsa?**
Nell'esecuzione di un programma orientato agli oggetti che rispetta il principio di sostituzione di Liskov non si possono verificare up-call	
La relazione di dipendenza fra package è transitiva	
Il linguaggio Java supporta l'ereditarietà multipla di un *interface* da più *interface*	
Il design pattern Model-View-Controller separa la logica di business dalla logica di presentazione	
L'attività più costosa nell'ambito della manutenzione è la comprensione del sistema da manutenere	
La squadra che esegue la manutenzione di un sistema è composta interamente da persone che hanno partecipato al testing in grande del sistema stesso	
Il testing di accettazione è un'attività che fa parte del testing in grande	
L'impalcatura per il testing (stub, driver e oracle) non è necessaria per il testing in grande	
Le ispezioni del codice devono essere effettuate conclusa l'attività di testing	
La checklist per l'ispezione del codice può comprendere anche controlli di aderenza agli standard aziendali	
Le carte di Gantt servono per risolvere i conflitti di allocazione temporale delle risorse a disponibilità illimitata di un progetto	
A ciascun livello di maturità di processo stabilito da CMM (Capability Maturity Model) corrisponde un livello distinto di qualità di prodotto	

Esercizi

1) Sia dato il design pattern Abstract Factory.

   a) Classificare tale pattern e descriverne l'intento.
   b) Descrivere la sua struttura.
   c) Derivare un diagramma degli oggetti, contenente almeno due istanze di ogni classe, compatibile con la struttura di cui al punto precedente.

2) Chiarire il concetto di sistema ereditato (*legacy system*).

3) Illustrare il concetto di refactoring e alcune tecniche di refactoring.

4) Creare un insieme di test funzionali di unità per un modulo la cui specifica è la seguente:

   acquisito in ingresso l'orario settimanale di *n* insegnamenti, determinare se esistono sovrapposizioni di orario relative ai docenti degli stessi.

   Una "sovrapposizione di orario relativa a un docente" si verifica se l'orario prevede che il medesimo docente tenga due insegnamenti (di cui è titolare) in una stessa ora.

   Il modulo restituisce il valore *true* se esiste almeno una sovrapposizione di orario relativa a un docente, *false* altrimenti.

   Si assuma che valgano le seguenti precondizioni:
   - ogni insegnamento ha un solo docente titolare,
   - un docente può essere titolare al più di due insegnamenti,
   - l'orario settimanale di ogni insegnamento è espresso da 3 blocchi, ciascuno dei quali
     - è della lunghezza di 2 ore,
     - è contenuto entro l'arco temporale giornaliero che va dalle ore 8.00 alle ore 13.00,
     - è collocato in un giorno lavorativo distinto rispetto agli altri 2 blocchi relativi allo stesso insegnamento,
     - è corredato dell'indicazione del docente relativo.

5) Sia dato il seguente programma.

```
main(){
1     scanf("%d", &a);
2     scanf("%d", &b);
3     if (a == b)
4          do a--
5          while a > 0;
6     else a = 2;
7     if (b == 1) {
8          a = a + b;
9          b = a*5;
10         if (a == 4) {
11              a--;
12              b = b + 3; }
      }
13    a--;
14    printf("%d\n", a);
15    printf("%d\n", b);
      }
```

   a) Tracciarne il CFG (Control Flow Graph).
   b) Stabilire, mediante esecuzione simbolica, se il cammino <1,2,3,4,5,7,8,9,10,13,14,15> è percorribile (*feasible*).