Esami di Ingegneria dei sistemi software e Ingegneria del software

Appello del 23 aprile 2012

Nome e cognome:
Matricola:

Codice esame: ☐ II222 ☐ ZZ304 ☐ 304II ☐ 374II

I candidati che sostengono l'esame di Ingegneria del software con **codice 374II** devono rispondere alle prime **dodici** domande. Il punteggio relativo a ciascuna domanda (indicato fra parentesi) è in ventesimi e verrà convertito in trentesimi all'atto della registrazione.

I candidati che sostengono gli esami con altri codici devono rispondere a tutte le domande. Il punteggio relativo a ciascuna domanda (indicato fra parentesi) è in trentesimi.

Tutti i candidati devono consegnare entro quarantacinque minuti dall'inizio della prova.

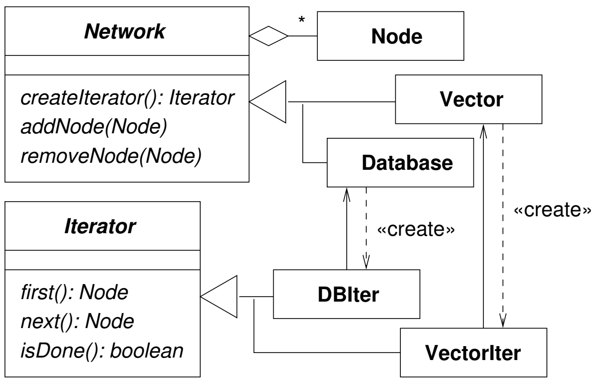

Figura 1: Domanda 1.

1 Con riferimento alla Fig. 1, rispondere alle domande. (5)

	V	F
Un **Network** è composto da istanze di **Node**	☐	☐
Un **Network** è composto da istanze di **Vector**	☐	☐
createIterator() è implementato da **Node**	☐	☐
Database deriva da **Vector**	☐	☐
Si può accedere ai nodi di un **Network** senza conoscerne l'implementazione	☐	☐

2 Descrivere in UML la seguente struttura: (5)

La finestra principale di un'applicazione contiene due menú ed un'area testo. Un menú ha un titolo e zero o piú voci. Una voce può essere un bottone o un menú. Un menú si può aprire e chiudere. Un bottone ha un'etichetta e un comando. Un bottone si può "cliccare".

3	**Il test di unità**	(1)
	Avviene di solito nella fase di codifica.	☐
	Viene pianificato in fase di analisi e specifica dei requisiti.	☐
	Fa parte della manutenzione del SW.	☐
4	**I modelli evolutivi**	(1)
	sviluppano il sistema in passi incrementali	☐
	si basano sempre su metodi formali	☐
	sono adatti soprattutto ad applicazioni ben conosciute	☐
5	**La modularità del sistema influisce sulla gestione del processo di sviluppo?**	(1)
	sí, perché richiede l'uso di strumenti CASE	☐
	sí, perché facilita la ripartizione del lavoro	☐
	no, perché è solo un aspetto tecnico del progetto	☐
6	**Un sistema formale è corretto se**	(1)
	tutte le formule dimostrabili sono vere	☐
	non contiene errori	☐
	tutte le formule vere sono dimostrabili	☐
7	**Un sistema che soddisfa i requisiti funzionali è**	(1)
	corretto.	☐
	completo.	☐
	robusto.	☐
8	**Nelle reti di Petri, una transizione è viva se**	(1)
	in ogni marcatura raggiungibile esiste una sequenza di scatti che la abilita.	☐
	è sempre abilitata.	☐
	esiste almeno una marcatura raggiungibile in cui è abilitata.	☐
9	**Un difetto del codice sorgente è**	(1)
	una anomalia.	☐
	un guasto.	☐
	un errore.	☐
10	**Negli Statechart, le computazioni non interrompibili sono modellate da**	(1)
	stati annidati.	☐
	attività.	☐
	azioni.	☐
11	**I diagrammi delle classi si possono usare**	(1)
	nella fase di progetto.	☐
	nelle fasi di progetto e codifica.	☐
	nelle fasi di analisi, progetto e codifica.	☐
12	**L'*information hiding* serve a nascondere i dettagli implementativi**	(1)
	all'utente	☐
	agli altri moduli	☐
	alla concorrenza	☐

13 **Con riferimento al codice di Fig. 2, disegnare un diagramma di classi che lo descriva.** (5)

14 Con riferimento all'esercizio precedente, (5)

scrivere un programma che stampi sull'uscita standard il risultato delle due operazioni della classe **Proxy_Greeting**, supponendo che ci si connetta al port 1492 della macchina `issw.unipi.it`.

```
class Connection {                   |   class Proxy_Greeting : public Greeting {
    string host;                     |       Connection* conn;
    int port;                        |   public:
    string buffer;                   |       Proxy_Greeting(Connection* c)
public:                              |           : conn(c) {};
    Connection(string h, int p)      |       string say_hello();
        : host(h), port(p) {};       |       string sag_gutenTag();
    void send(string s);             |   };
    string receive();                |
};                                   |   string
                                     |   Proxy_Greeting::
class Greeting {                     |   say_hello()
public:                              |   {
    virtual string                   |       conn->send("say_hello");
        say_hello() = 0;             |       return conn->receive();
    virtual string                   |   }
        sag_gutenTag() = 0;          |
};                                   |

                         string
                         Proxy_Greeting::
                         sag_gutenTag()
                         {
                             conn->send("sag_gutenTag");
                             return conn->receive();
                         }
```

Figura 2: Domanda 17.