UNIVERSITÀ DEGLI STUDI DI PISA

Corso di Laurea in Ingegneria Informatica

c/o Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni

Esame di Ingegneria dei Sistemi Software

Appello del 23 gennaio 2006

Nome e cognome:
Matricola:

Il punteggio in trentesimi associato a ciascuna domanda è indicato fra parentesi.

Scegliere una risposta per ciascuna domanda:

1 Si dice che il SW è *malleabile* perché: (1)
i programmi sono facilmente adattabili a diverse applicazioni. ☐
i programmi non occupano spazio fisico. ☐
i programmi sono facilmente modificabili. ☒

2 *Definizione* e *specifica* dei requisiti (1)
sono sinonimi. ☐
hanno due diversi livelli di dettaglio. ☒
appartengono risp. allo studio di fattibilità e alla fase di analisi. ☐

3 Il CVS è (1)
uno strumento per la gestione delle versioni del codice sorgente. ☒
una metodologia di sviluppo per sistemi concorrenti. ☐
un linguaggio di specifica. ☐

4 Nel modello Cleanroom (1)
gli sviluppatori verificano il codice staticamente. ☒
gli sviluppatori verificano il codice dinamicamente. ☐
il codice viene generato automaticamente. ☐

5 Se p è un place, $^{\bullet}p$ è (1)
l'insieme dei place di ingresso di p. ☐
l'insieme delle transizioni di ingresso di p. ☒
la marcatura di p. ☐

Rispondere alle domande, usando solo lo spazio disponibile:

6 Che cos'è una *specifica*? (3)

Una descrizione precisa dei requisiti di un sistema o di una sua parte.

7 In un processo di sviluppo, che cos'è una *fase*? (3)

Un intervallo di tempo in cui svolge un'attività.

8 A cosa serve il framework *DejaGNU*? (3)

Al test di unità.

9 Che cos'è l'*alpha test*? (3)

Un test di sistema che si svolge usando l'applicazione all'interno dell'azienda produtrice.

10 Che cos'è un sistema concorrente? (3)

Un sistema formato da piú processi intercomunicanti e soggetti a vincoli di sincronizzazione reciproca.

11 **Scrivere le dichiarazioni in C++ corrispondenti allo schema UML di Fig. 1.** (4)
(Implementare l'associazione fra **Editor** e **Button** con `std::list<>` o con un array di dimensione a piacere).

12 **Con riferimento all'esercizio precedente:** (6)
implementare `Button::on_click()`, che deve eseguire il comando associato al bottone;
implementare `OpenCmd::file_prompt()`, che deve chiedere all'utente il nome del documento da aprire e restituirlo al chiamante;
implementare `OpenCmd::execute()`, che deve aprire il documento specificato dall'utente, dopo averlo registrato nell'editor;
supponendo già implementate tutte le altre operazioni, scrivere un programma principale che apra un documento.

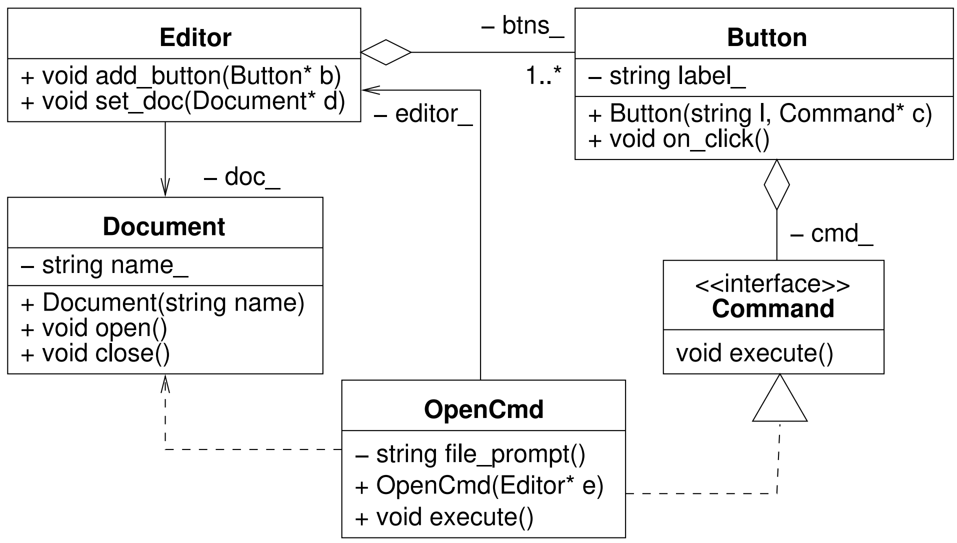

Figura 1: Domanda 11.

```cpp
class Command {
public:
    virtual void execute() = 0;
};


class OpenCmd : public Command {
    Editor* editor_;
    string file_prompt();
public:
    OpenCmd(Editor* e) : editor_(e) {};
    void execute();
};

class Editor {
    Document* doc_;
    list<Button*> btns_;
public:
    void add_button(Button* b);
    void set_doc(Document* d);
};
```

```cpp
class Button {
    string label_;
    Command* cmd_;
public:
    Button(string l, Command* c)
            : label_(l), cmd_(c) {};
    void on_click();
};

class Document {
    string name_;
public:
    Document(string name) {};
    void open();
    void close();
 };
```

Figura 2: Domanda 11, soluzione.

```cpp
void
Button::
on_click()
{
    cmd_->execute();
}

string
OpenCmd::
file_prompt()
{
    string s;
    cout << "enter file name: ";
    cin >> s;
    return s;
}
```

```cpp
void
OpenCmd::
execute()
{
    string name = file_prompt();
    Document* doc = new Document(name);
    editor_->set_doc(doc);
    doc->open();
}

int
main()
{
    Editor e;
    OpenCmd oc(&e);
    Button ob("Open", &oc);
    e.add_button(&ob);
    ob.on_click();
}
```

Figura 3: Domanda 12, soluzione.