Esame di Ingegneria del software

Appello del 31 gennaio 2018

Nome e cognome:
Matricola:

Il punteggio relativo a ciascuna domanda, indicato fra parentesi, è in trentesimi. I candidati devono consegnare entro un'ora dall'inizio della prova.

1 **Disegnare il diagramma di classi** corrispondente al listato di Fig. 1. (5)

2 **Scrivere un programma in C++** che, usando il codice di Fig. 1, legga da ingresso standard un numero positivo K e, per K volte, legga i valori di pressione da due manometri e scriva la media fra i due su uscita standard. (5)

3 **Disegnare uno statechart UML** che specifichi quanto segue: un regolatore deve assicurare che, dopo la chiusura di una certa valvola A (a) o di una valvola B (b), avvenga l'apertura di una valvola di scarico S, apertura che può essere automatica (sa) o comandata dal regolatore (sc), entro Σ secondi; nello stato iniziale, il tempo (rappresentato da una variabile t) scorre a partire da zero; in corrispondenza di ogni chiusura delle valvole A e B il regolatore resta nello stato iniziale ed il tempo viene azzerato; se passano Σ secondi senza che intercorrano chiusure di A o B, il regolatore invia il segnale sc, restando nello stato iniziale; quando si verifica un'apertura automatica di S, il regolatore entra in uno stato di attesa da cui esce quando si richiude A o B, rientrando nello stato iniziale ed azzerando il tempo (suggerimento: serve l'evento temporale after()). (5)

4 **Ridisegnare il diagramma di Fig. 2** come architettura a strati. (5)

5 La classe **Module** contiene una rappresentazione interna del codice sorgente di un programma. Una delle sue operazioni è assemble(), che restituisce una struttura dati di tipo **Object** contenente il risultato della compilazione. Applicare il pattern *Strategy* (Fig. 3) in modo da poter cambiare facilmente il tipo di processore (per esempio ARM, Atmel, PowerPC...) per cui generare il codice. Mostrare l'implementazione dell'operazione assemble() e specificare eventuali argomenti e valori restituiti delle operazioni introdotte. (5)

6 **Rispondere alle seguenti domande.** (5)

In un sistema formale corretto, tutte le formule valide sono dimostrabili.	V ☐ F ☒
Il *CppUnit* è un framework per il testing.	V ☒ F ☐
I membri pubblici di una classe costituiscono la sua interfaccia richiesta.	V ☐ F ☒
Una classe *realizza* un'interfaccia se ne implementa la parte privata.	V ☐ F ☒
Tutte le formule valide sono vere.	V ☒ F ☐

```
class Probe {
public:
    virtual int get_sample(void) =0;
};

class Conditioner {
public:
    virtual int smooth(int raw) =0;
};

class Manometer : public Probe {
    Conditioner* mc;
    int raw;
public:
    Manometer(Conditioner* c): mc(c);
    int raw_data(void);
    int get_sample(void);
};

class PConditioner : public Conditioner {
    // data structures for smoothing
public:
    int smooth(int raw);
};

int
Manometer::
raw_data(void)
{
    // read value from hardware
}

int
Manometer::
get_sample(void)
{
    raw = raw_data();
    return mc->smooth(value);
}

int
PConditioner::
smooth(int value)
{
    // implementation of smoothing algorithm
}
```


Figura 1: Domanda 1.

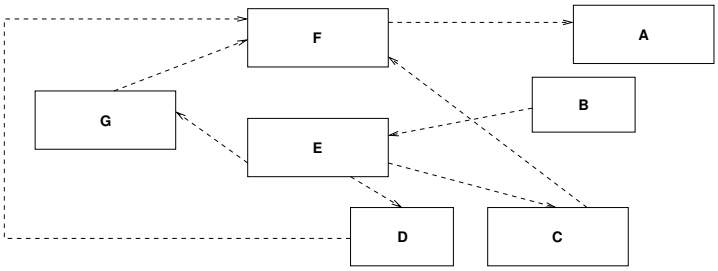

Figura 2: Domanda 4.

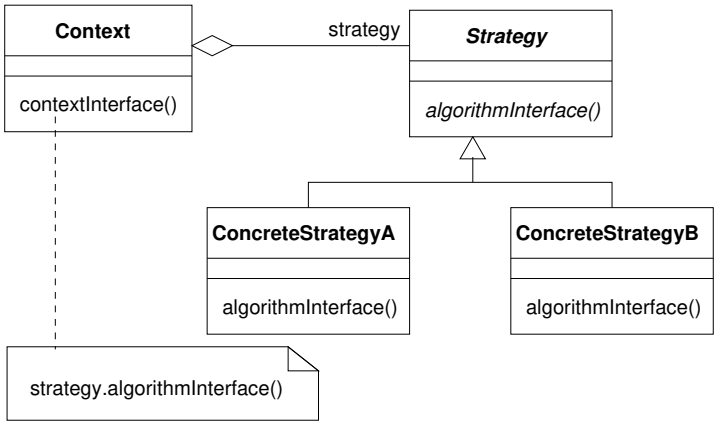

Figura 3: Domanda 5.

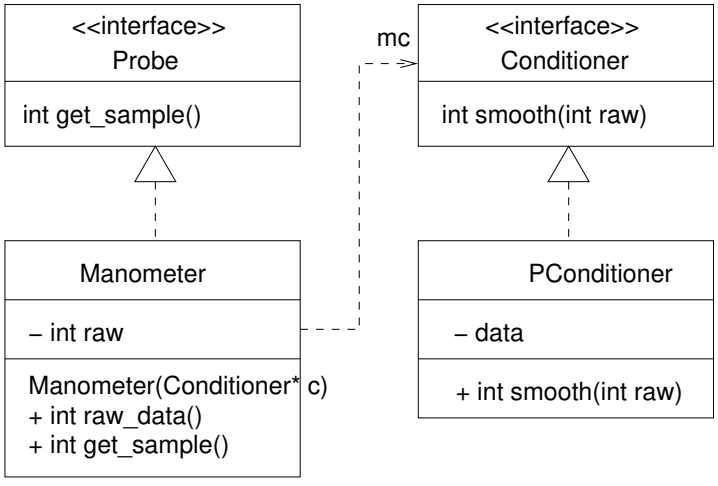

Figura 4: Domanda 1, soluzione.

```
int
main()
{
    int N;
    cin >> N;

    KFilter kf1;
    KFilter kf2;
    TempSensor ts1(&kf1);
    TempSensor ts2(&kf2);

    for (int i = 0; i < N; i++) {
        ts1.read();
        int v1 = ts1.get_temp();
        ts2.read();
        int v2 = ts2.get_temp();
        cout << (v1 + v2)/2 < endl;
    }
}
```

Figura 5: Domanda 2, soluzione.

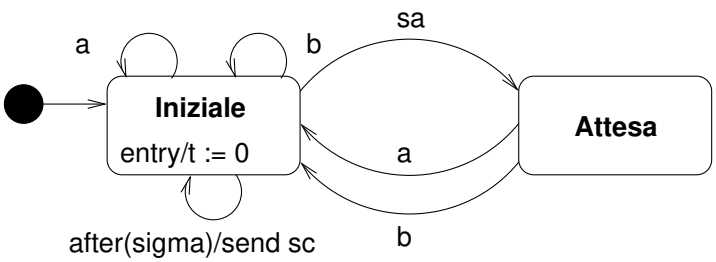

Figura 6: Domanda 3, soluzione.

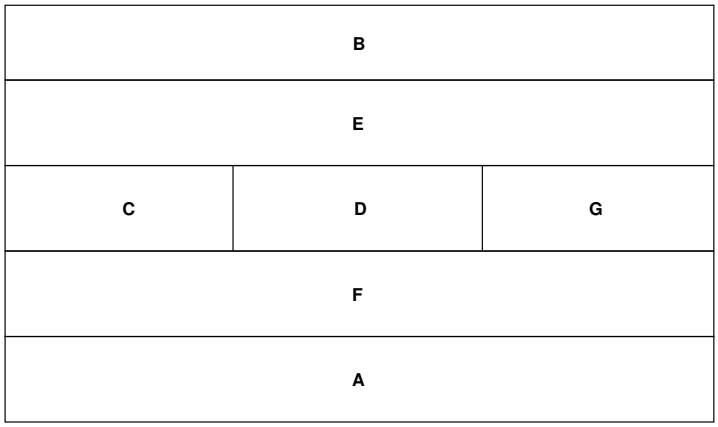

Figura 7: Domanda 4, soluzione.

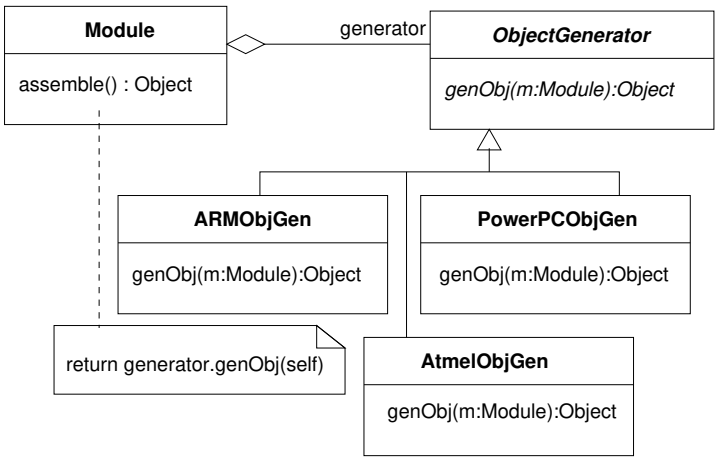

Figura 8: Domanda 5, soluzione.