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 N e, per N volte, legga i valori di temperatura da due sensori e scriva la media fra i due su uscita standard. (5)

3 **Disegnare uno statechart UML** che specifichi quanto segue: un pacemaker deve assicurare che, dopo una contrazione ventricolare spontanea (vs) o indotta dal pacemaker (vp), avvenga una contrazione atriale spontanea (as) o indotta (ap) entro Δ secondi; nello stato iniziale, il tempo (rappresentato da una variabile t) scorre a partire da zero; in corrispondenza di ogni evento ventricolare il pacemaker resta nello stato iniziale ed il tempo viene azzerato; se passano Δ secondi senza che intercorrano eventi ventricolari o atriali, il pacemaker invia il segnale ap, restando nello stato iniziale; quando si verifica un evento atriale spontaneo, il pacemaker entra in uno stato di attesa da cui esce quando riceve un evento atriale, 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 **ParseTree** contiene una rappresentazione interna del codice sorgente di un programma. Una delle sue operazioni è generate(), che restituisce una struttura dati di tipo **Code** contenente il risultato della compilazione. Applicare il pattern *Strategy* (Fig. 3) in modo da poter cambiare facilmente il tipo di processore (per esempio Alpha, Pentium, PowerPC...) per cui generare il codice. Mostrare l'implementazione dell'operazione generate() e specificare eventuali argomenti e valori restituiti delle operazioni introdotte. (5)

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

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

```
class Sensor {
public:
    virtual int get_value(void) =0;
};

class Filter {
public:
    virtual int filter(int value) =0;
};

class TempSensor : public Sensor {
    Filter* tf;
    int value;
public:
    TempSensor(Filter* f): tf(f);
    int read(void);
    int get_value(void);
};

class KFilter : public Filter {
    // data structures for filtering
public:
    int filter(int value);
};

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

int
TempSensor::
get_temp(void)
{
    value = read();
    return tf->filter(value);
}

int
KFilter::
filter(int value)
{
    // implementation of filter algorithm
}
```


Figura 1: Domanda 1.

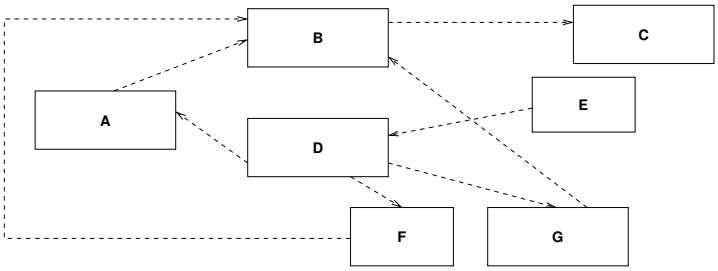

Figura 2: Domanda 4.

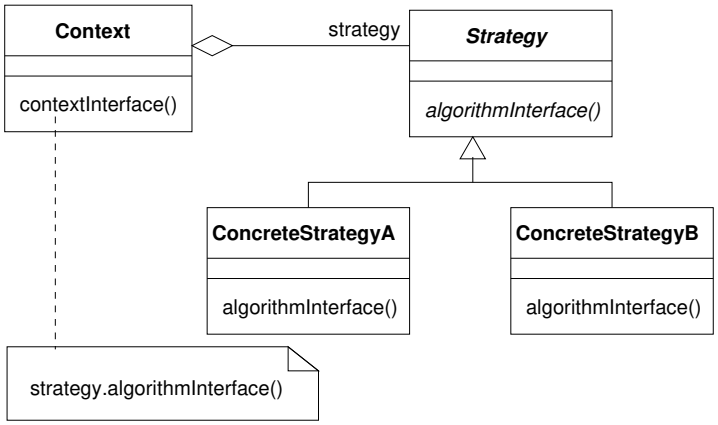

Figura 3: Domanda 5.

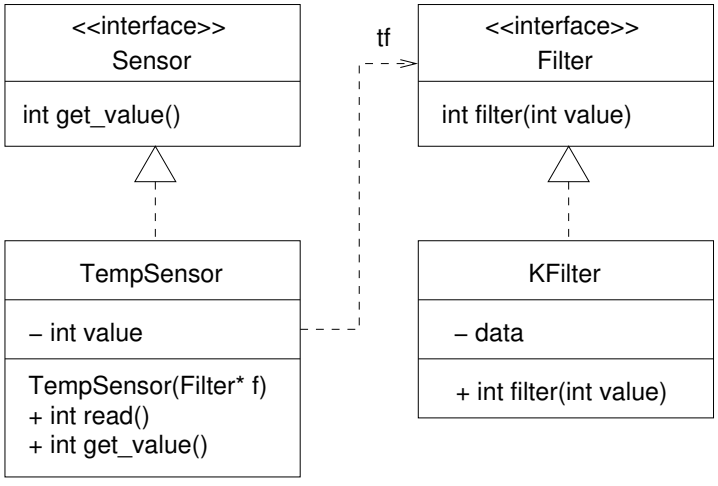

Figura 4: Domanda 1, soluzione.

```
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.

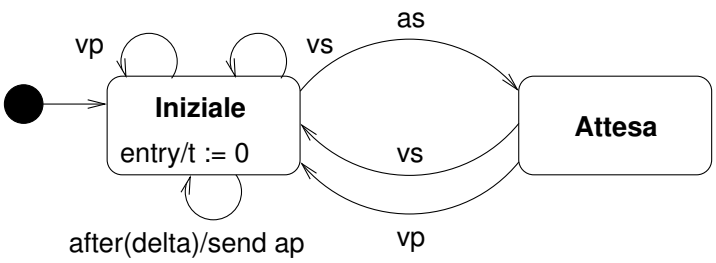

Figura 6: Domanda 3, soluzione.

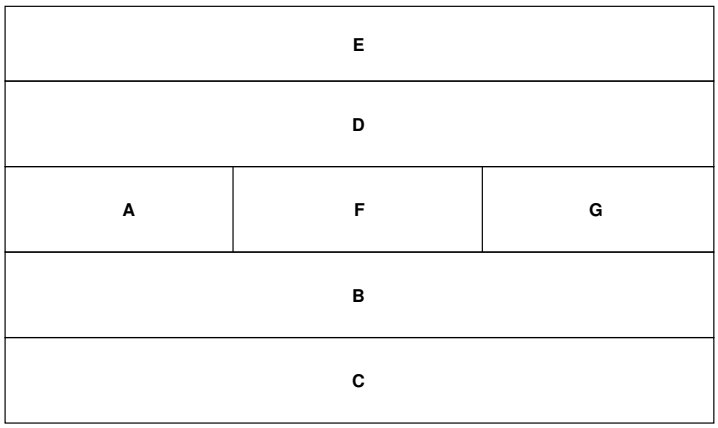

Figura 7: Domanda 4, soluzione.

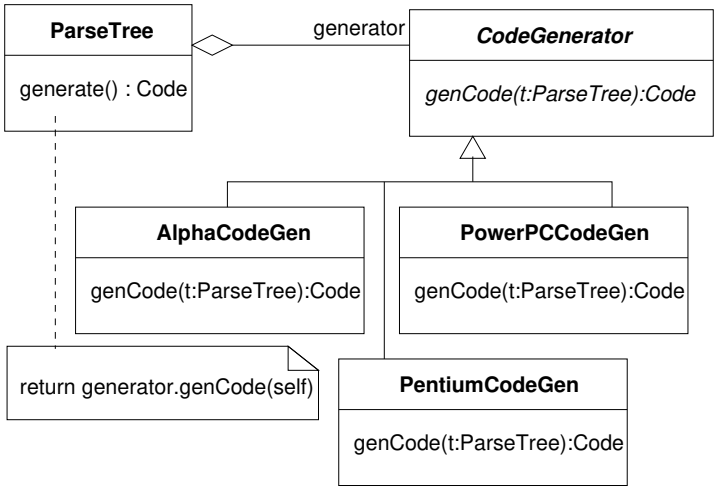

Figura 8: Domanda 5, soluzione.