

Esame di Ingegneria del software

Appello del 10 gennaio 2018

soluzioni

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 un diagramma di classi** che specifichi quanto segue: un ascensore serve uno stabile di N piani, incluso il piano terra; la cabina è mossa da un motore elettrico; la cabina ha una porta scorrevole azionata da un motore; la porta ha un sensore antischiacciamento e due sensori di fine corsa; nella cabina si trovano una pulsantiera per scegliere il piano di destinazione, ed un indicatore di piano (display); ogni piano ha un pulsante di chiamata. Indicare le molteplicità. Non sono richiesti attributi ed operazioni. (5)
- 2 Disegnare uno statechart UML** che specifichi quanto segue: la porta di un ascensore in attesa a un piano è aperta; quando un utente attraversa la porta, il sensore antischiacciamento manda un segnale di *porta occupata*; quando la porta torna libera, il sensore manda un segnale di *porta libera*; dopo che un utente ha scelto un piano, la porta inizia a chiudersi; se la porta diviene occupata, la porta si blocca; quando la porta si è chiusa completamente, la cabina si mette in moto; quando la cabina raggiunge il piano richiesto, si ferma; quando la cabina si è fermata, la porta si apre e l'ascensore resta in attesa. (5)
- 3 Con riferimento alla Fig. 1, rispondere alle domande.** (5)
get_temp viene eseguita infinite volte.
V F
fire_emergency e power_emergency vengono eseguite lo stesso numero di volte.
V F
Quando p diviene falso, il ciclo contenente pwr_up termina.
V F
La sequenza {fire_alert, fire_emergency} può essere interrotta.
V F
La sequenza {power_alert, power_emergency} può essere interrotta.
V F
- 4 Con riferimento alla Fig. 2, rispondere alle domande.** (5)
Client usa l'interfaccia di **Robot**.
V F
RobotAdapter usa l'interfaccia di **Robot**.
V F
Robot realizza **Actuator**.
V F
Actuator implementa **Client**.
V F
Client dipende da **Actuator**.
V F
- 5 Completare la seguente tabella di verità** dell'operatore booleano ternario *if-then-else* (5)

<i>x</i>	<i>y</i>	<i>z</i>	if <i>x</i> then <i>y</i> else <i>z</i>
F	F	F	F
F	F	T	T
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	F
T	T	F	T
T	T	T	T

6 Rispondere alle seguenti domande.

(5)

Il *calcolo dei sequenti* è un sistema formale.

V \boxtimes F \square

Un assioma è una formula che deve essere dimostrata con una regola d'inferenza.

V \square F \boxtimes

In un sistema formale corretto, tutte le formule dimostrabili sono valide.

V \boxtimes F \square

Tutte le formule vere sono valide.

V \square F \boxtimes

Tutte le formule ben formate sono valide.

V \square F \boxtimes

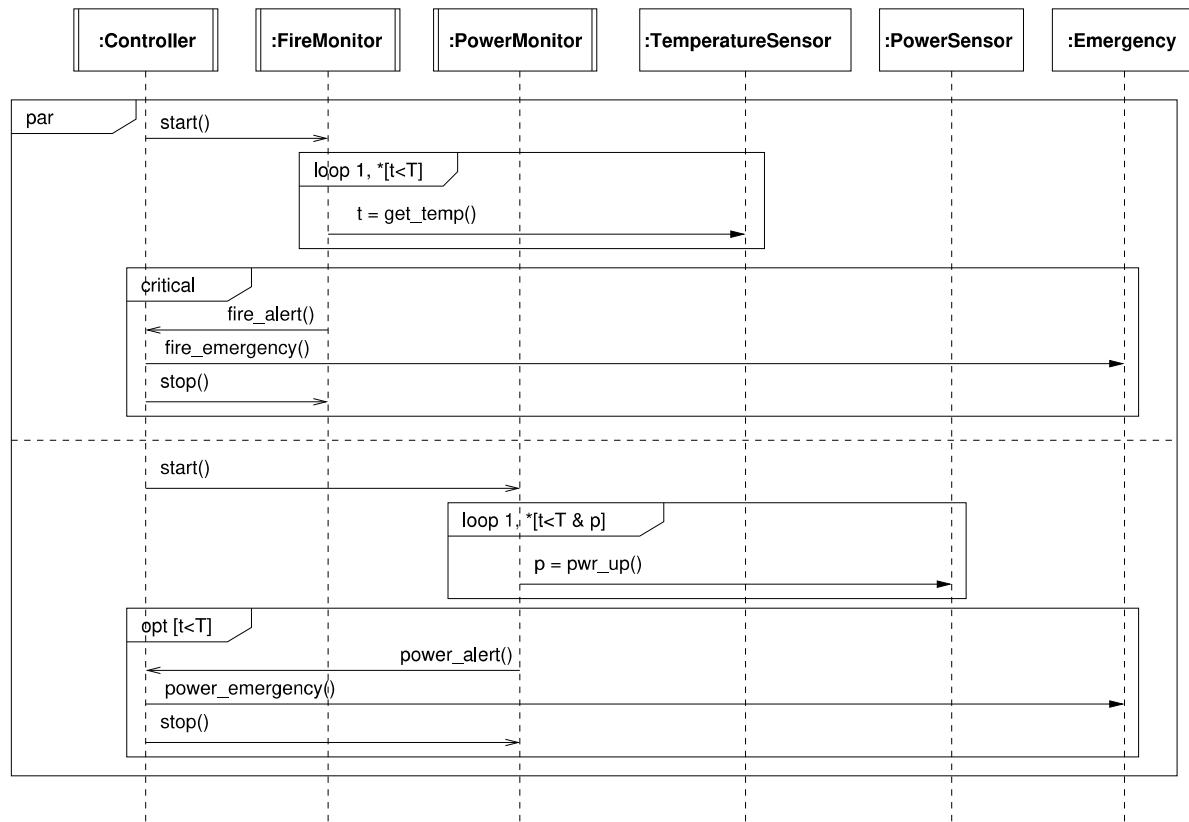


Figura 1: Domanda 3.

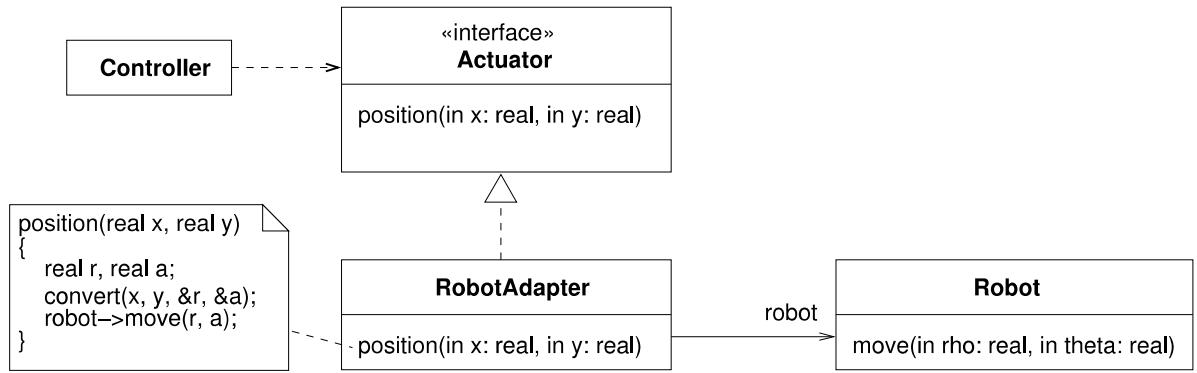


Figura 2: Domanda 4.

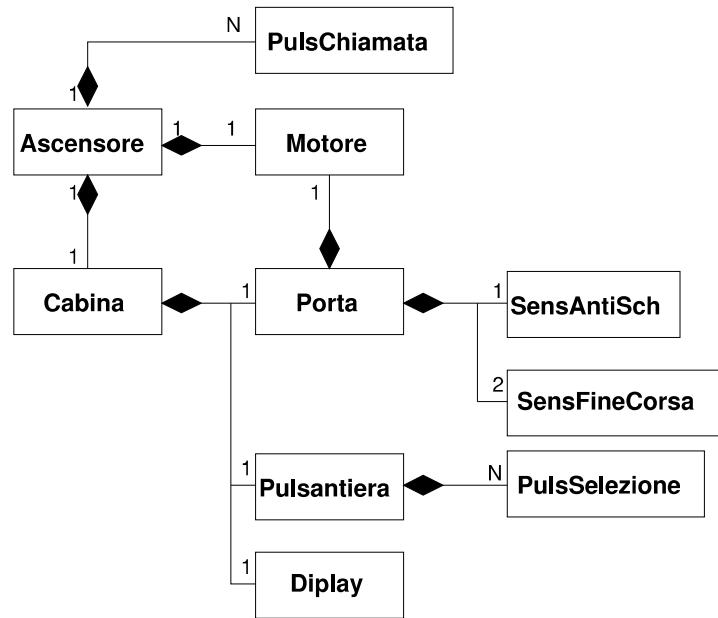


Figura 3: Domanda 1, soluzione.

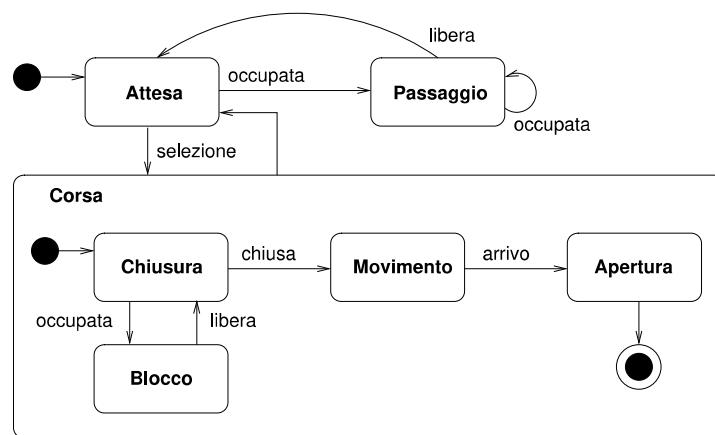


Figura 4: Domanda 2, soluzione.

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)

Il <i>CppUnit</i> è un framework per il testing.	V <input checked="" type="checkbox"/> F <input type="checkbox"/>
I membri pubblici di una classe costituiscono la sua interfaccia richiesta.	V <input type="checkbox"/> F <input checked="" type="checkbox"/>
In un sistema formale corretto, tutte le formule valide sono dimostrabili.	V <input type="checkbox"/> F <input checked="" type="checkbox"/>
Tutte le formule valide sono vere.	V <input checked="" type="checkbox"/> F <input type="checkbox"/>
Una classe <i>realizza</i> un'interfaccia se ne implementa la parte privata.	V <input type="checkbox"/> F <input checked="" type="checkbox"/>
- 6 **Rispondere alle seguenti domande.** (5)

Il <i>CppUnit</i> è un framework per il testing.	V <input checked="" type="checkbox"/> F <input type="checkbox"/>
I membri pubblici di una classe costituiscono la sua interfaccia richiesta.	V <input type="checkbox"/> F <input checked="" type="checkbox"/>
In un sistema formale corretto, tutte le formule valide sono dimostrabili.	V <input type="checkbox"/> F <input checked="" type="checkbox"/>
Tutte le formule valide sono vere.	V <input checked="" type="checkbox"/> F <input type="checkbox"/>
Una classe <i>realizza</i> un'interfaccia se ne implementa la parte privata.	V <input type="checkbox"/> F <input checked="" type="checkbox"/>

```

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
}

```

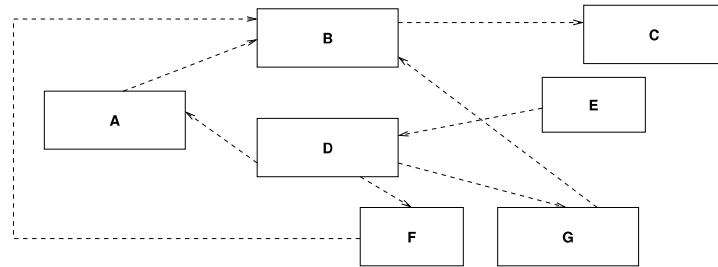


Figura 2: Domanda 4.

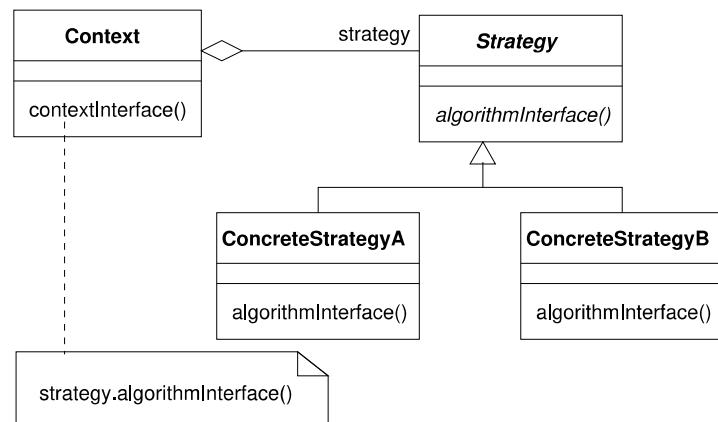


Figura 3: Domanda 5.

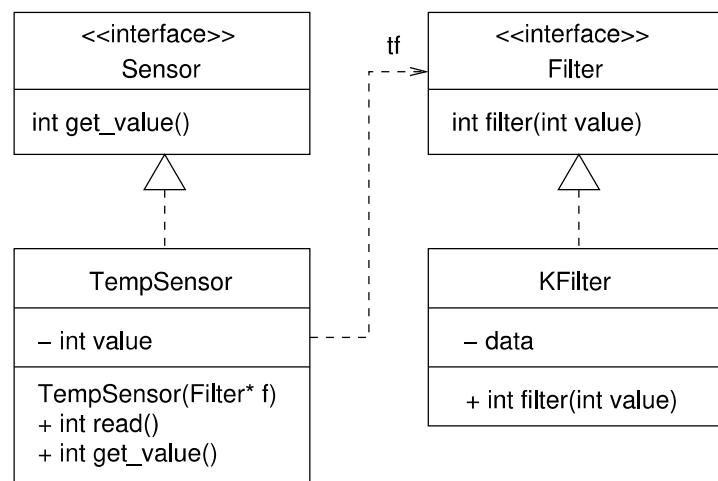


Figura 4: Domanda 1, soluzione.

```

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.

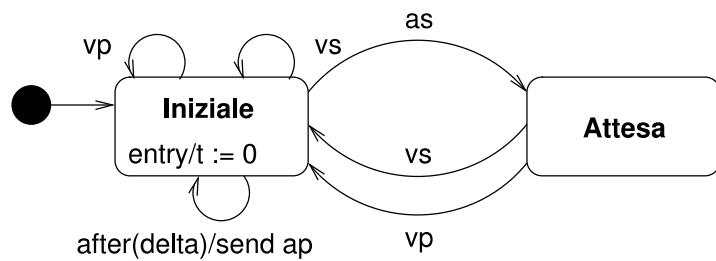


Figura 6: Domanda 3, soluzione.

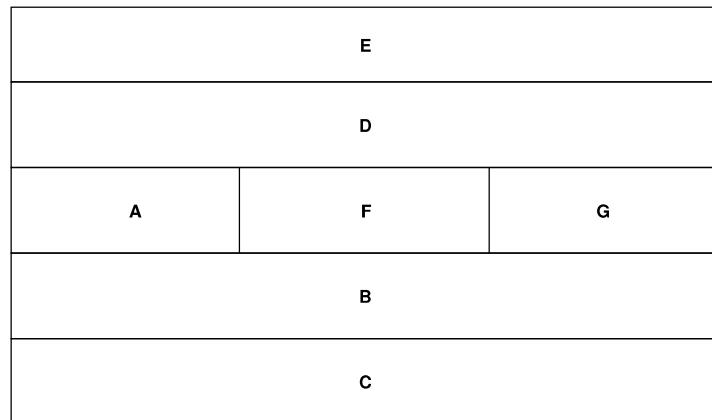


Figura 7: Domanda 4, soluzione.

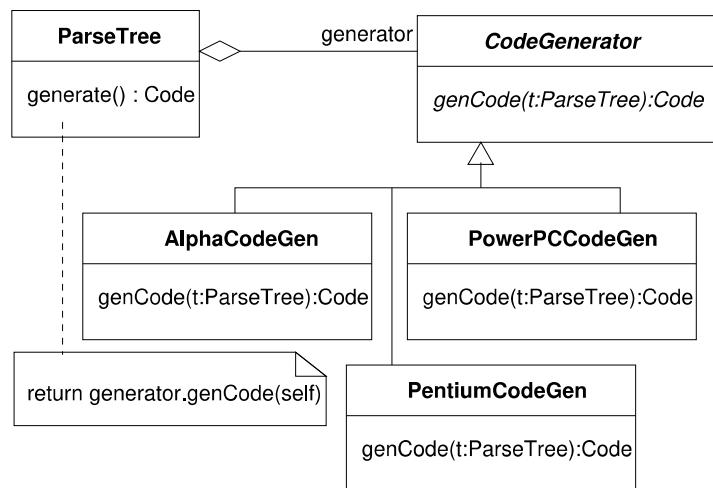


Figura 8: Domanda 5, soluzione.

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
}

```

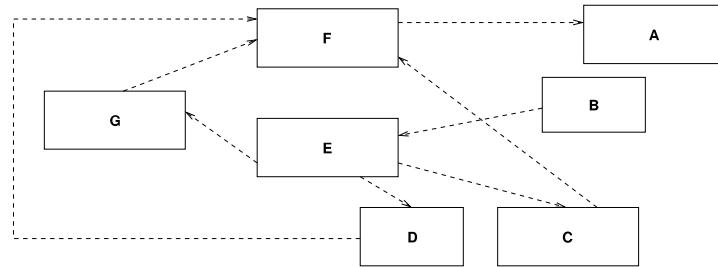


Figura 2: Domanda 4.

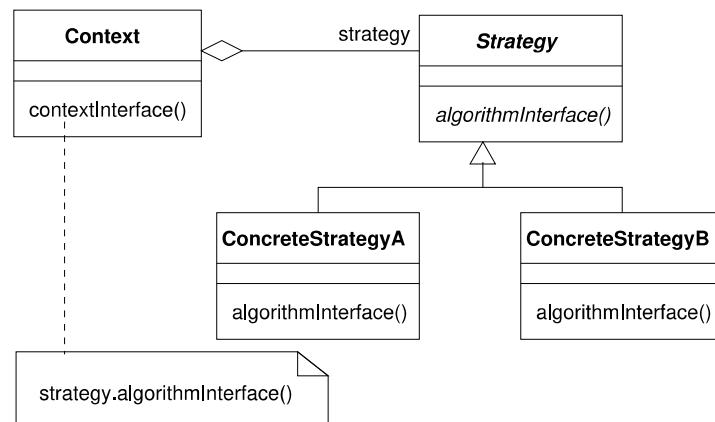


Figura 3: Domanda 5.

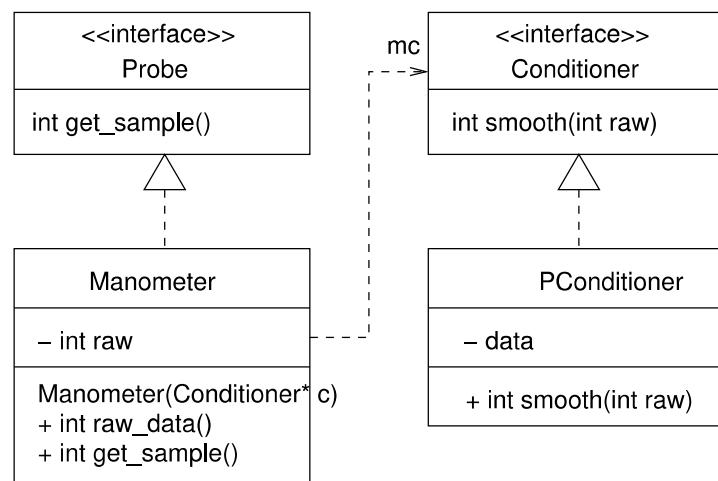


Figura 4: Domanda 1, soluzione.

```

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.

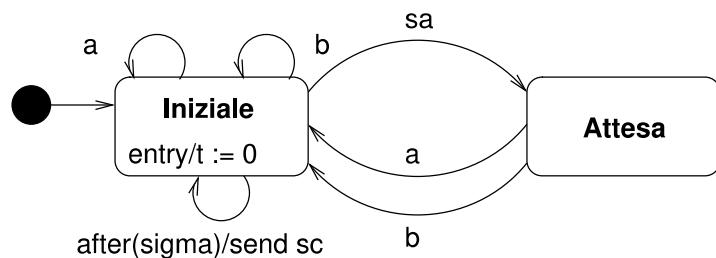


Figura 6: Domanda 3, soluzione.

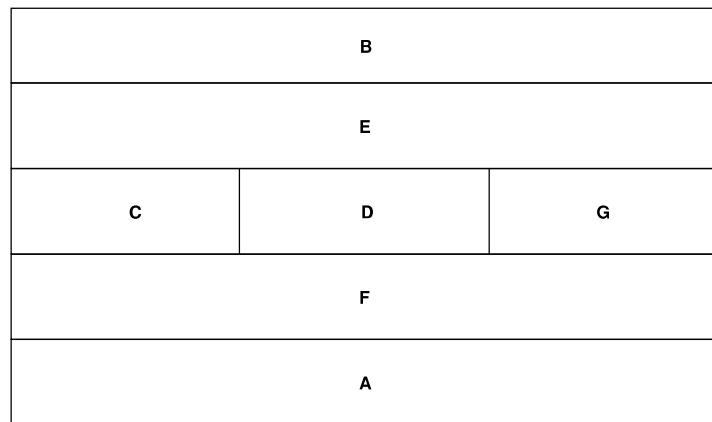


Figura 7: Domanda 4, soluzione.

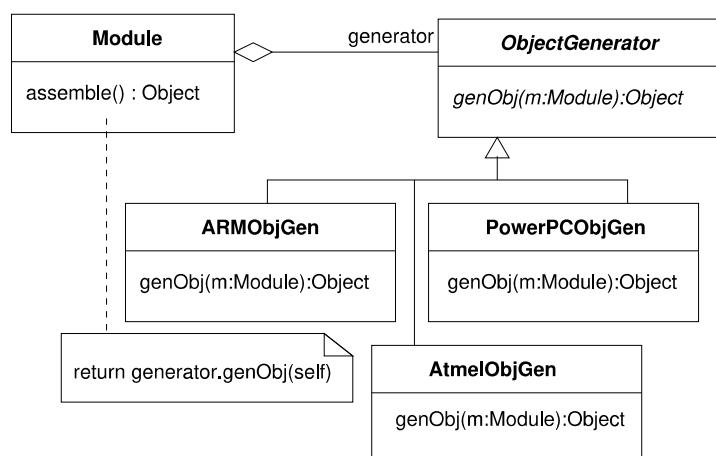


Figura 8: Domanda 5, soluzione.

Esame di Ingegneria del software

Appello del 16 febbraio 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** relativo al seguente problema: la classe astratta **Network** è un aggregato di istanze della classe **Node**, con le operazioni *addNode()* e *removeNode()* per aggiungere e togliere nodi, e un'operazione *createIterator()* che restituisce un'istanza di una classe derivata dalla classe astratta **Iterator**. Quest'ultima permette di accedere in sequenza ai nodi appartenenti ad un oggetto **Network**, per mezzo delle operazioni *first()* e *next()*. L'operazione *isDone()* serve a sapere se l'enumerazione dei nodi è terminata. La classe **Network** viene realizzata dalle classi **Vector** e **Database**, che definiscono contenitori concreti per mezzo di diverse strutture dati. I loro metodi *createIterator()* creano e restituiscono istanze, rispettivamente, delle classi concrete **VectorIter** e **DBIter**. (5)
- 2 **Scrivere un programma in C++** che, usando il framework di Fig. 1, (i) crei una finestra col titolo “Main Window”; (ii) vi inserisca un bottone e un campo di testo; (iii) scriva sull'uscita standard il contenuto del campo testo, quando viene premuto il bottone. (5)
- 3 **Disegnare uno statechart UML** che specifichi quanto segue. Un file può essere di tipo volatile o persistente; un file persistente può essere locale o su nastro, mentre un file volatile non viene mai messo su nastro; un file è locale mentre viene creato o copiato su o da memoria RAM, e mentre è memorizzato su disco; al termine della creazione o copia, il file va su nastro se permanente o su disco se volatile; il comando **read** copia un file da disco a memoria, il comando **load** sposta un file da nastro a disco, il comando **store** sposta da disco a nastro, il comando **remove** sposta un file permanente da disco a nastro, o cancella un file volatile; i file su nastro devono essere trasferiti su disco per essere letti. (5)
- 4 **Con riferimento alla Fig. 2, rispondere alle domande.** (5)

	V	F
<i>Statement</i> implementa Codesegment .	<input type="checkbox"/>	<input checked="" type="checkbox"/>
una Expression può contenere dei CodeSegment .	<input type="checkbox"/>	<input checked="" type="checkbox"/>
una Instruction fa parte di un CodeSegment .	<input checked="" type="checkbox"/>	<input type="checkbox"/>
tutti gli <i>Statement</i> sono Expression .	<input type="checkbox"/>	<input checked="" type="checkbox"/>
<i>generate()</i> restituisce un oggetto di tipo CodeSegment .	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- 5 Disegnare un diagramma di classi relativo al seguente problema: Si hanno delle (5) classi **Square**, **Circle** etc., che rappresentano figure geometriche. Ognuna di esse ha un'operazione **area():int** che restituisce l'area della figura. (a) Si definisca l'interfaccia di una classe **Container** che contenga un insieme di figure qualsiasi e che permetta di i) aggiungere una figura, ii) ottenere un riferimento (o un puntatore) a una figura individuata da un indice, e iii) ottenere il numero di figure contenute; (b) si definisca una classe **Client** con un'operazione **total_area** che restituisce l'area totale, mostrandone l'implementazione.
- 6 Disegnare un diagramma di stato che modelli il (5)
comportamento della seguente classe C++:

```

class Buffer2 {
    enum state { EMPTY, HALF, FULL };
    state s;
public:
    Buffer2() : s(EMPTY) {};
    void data_in();
    void data_out();
};

void
Buffer2::
data_in()
{
    switch (s) {
        case EMPTY: s = HALF; break;
        case HALF:  s = FULL; break;
        case FULL:  break;
    }
};

void
Buffer2::
data_out()
{
    switch (s) {
        case EMPTY: break;
        case HALF:  s = EMPTY; break;
        case FULL:  s = HALF; break;
    }
};

```

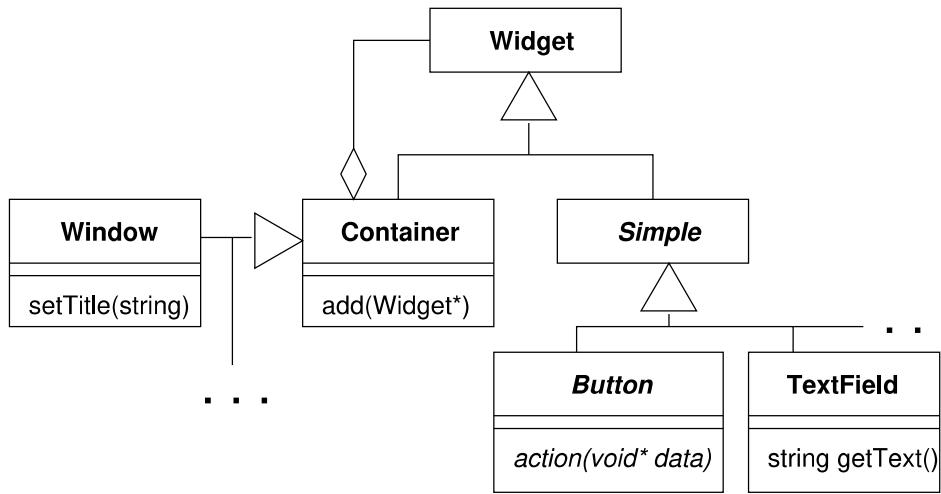


Figura 1: Domanda 2.

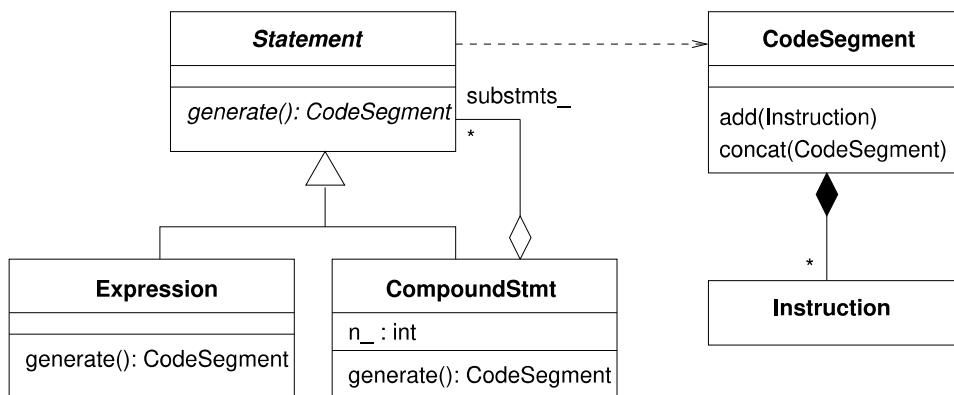


Figura 2: Domanda 4.

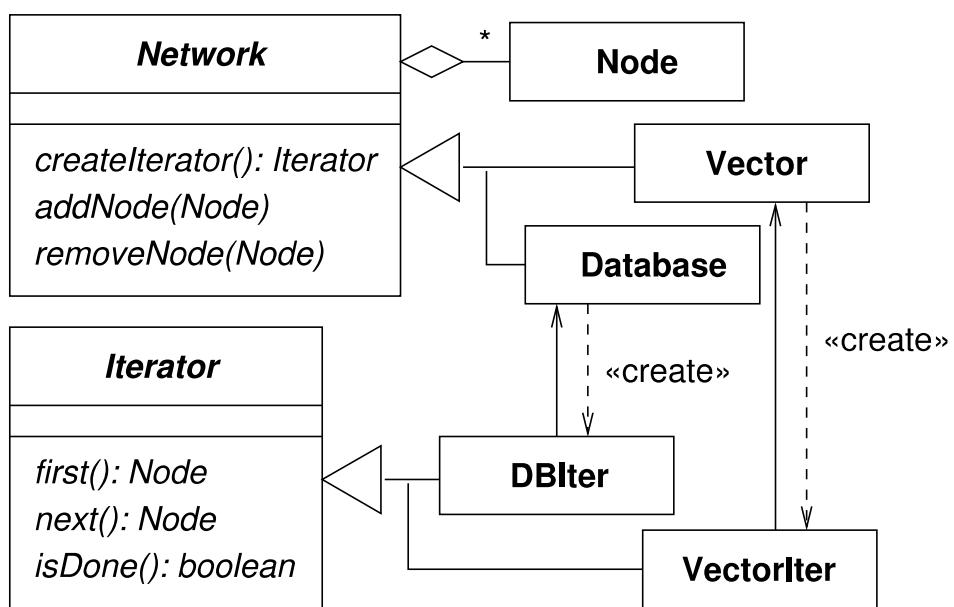


Figura 3: Domanda 1, soluzione.

```

#include <iostream>
#include "widgets.h"

using namespace std;

class MyButton : public Button {
    TextField* tf;
public:
    MyButton(TextField* t) : tf(t) {};
    void action(void* data);
};

void
MyButton::
action(void* data)
{
    cout << tf->getText() << endl;
}

int
main()
{
    Window* w = new Window;
    TextField* t = new TextField;
    MyButton* b = new MyButton;

    w->setTitle("Main Window");
    w->add(t);
    w->add(b);
}

```

Figura 4: Domanda 2, soluzione.

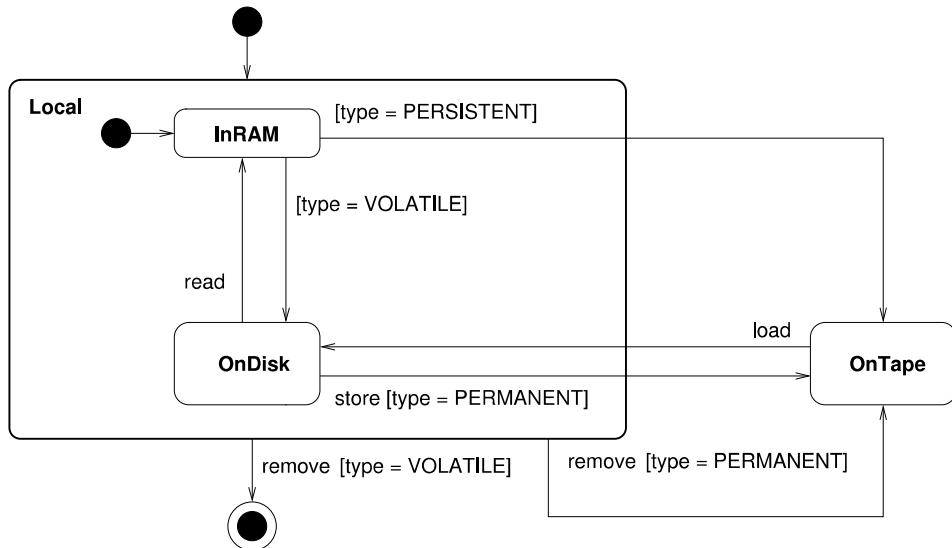


Figura 5: Domanda 3, soluzione.

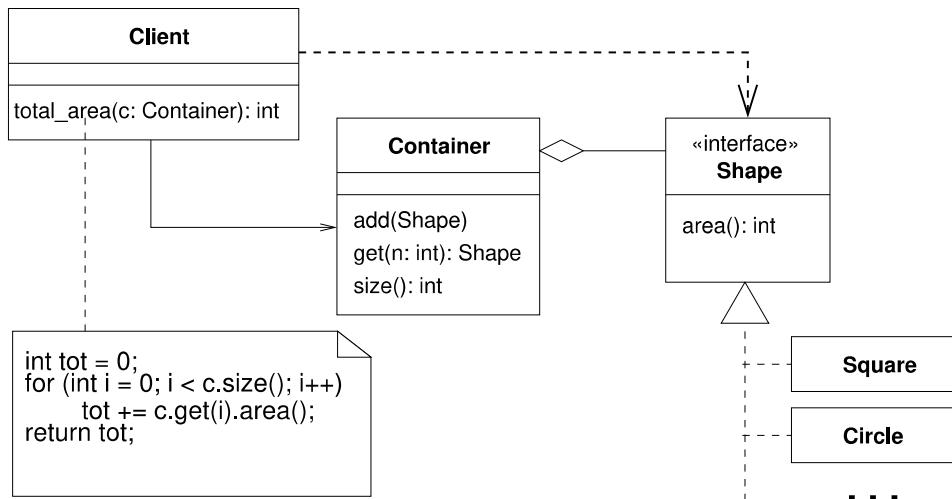


Figura 6: Domanda 5, soluzione.

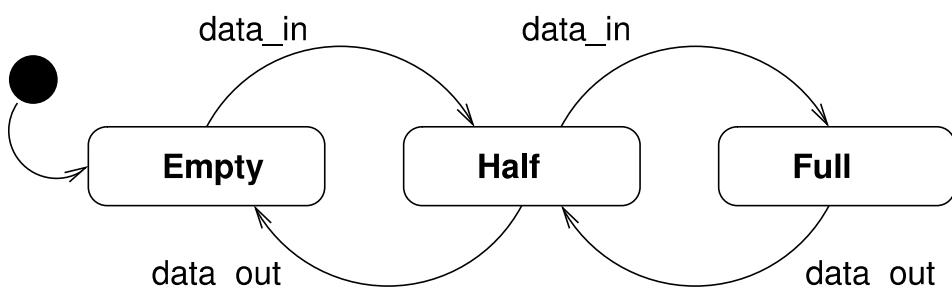


Figura 7: Domanda 6, soluzione.

Esame di Ingegneria del software

Appello del 24 settembre 2019

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 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
- 2 Nel modello OO, un metodo è (1)
un'operazione privata.
un'operazione pubblica.
l'implementazione di un'operazione.
- 3 Un guasto è (1)
un comportamento scorretto rispetto alle specifiche.
un difetto del codice sorgente.
un errore di progetto o di programmazione.
- 4 I linguaggi formali (1)
hanno una sintassi grafica.
sono standardizzati.
hanno una semantica di tipo matematico.
- 5 Negli Automi a Stati Finiti le uscite (1)
dipendono dalla marcatura
dipendono dallo stato e dall'ingresso
dipendono dalle condizioni di guardia
- 6 Con riferimento alla Fig. 1, (5)
il **Produttore** inizia l'interazione
il **Produttore** fa terminare l'interazione
il **Produttore** invia il segnale **consuma**
il **Consumatore** entra in **Elaborazione** prima del **Produttore**
il **Consumatore** entra in **Attesa** quando riceve produci
- 7 Con riferimento alla Fig. 2, (5)
un **ConcreteScrollWdw** contiene un **Window**
un **ConcreteScrollWdw** è un **PlainWdw**
ConcreteScrollWdw implementa **PlainWdw**
ConcreteScrollWdw estende il comportamento di **PlainWdw**
un **Window** contiene un **PlainWdw**

V	F
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>

V	F
<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>

8 Con riferimento alla Fig. 3,

	(5)
V	F
<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>

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

9 Disegnare un diagramma di classi che rappresenti la seguente applicazione:

un'agenda elettronica permette di (i) inserire coppie di stringhe (*nome, numero*) in un elenco, (ii) cancellare coppie fornendo il nome, e (iii) cercare i numeri corrispondenti ai nomi. L'elenco viene memorizzato in un file. Strutturare l'applicazione separando le funzioni della gestione dell'elenco e della gestione del file contenente l'elenco. L'elenco, oltre alle operazioni di inserimento, cancellazione e ricerca, ha un'operazione `enumera()` per accedere in sequenza alle coppie, e un'operazione `azzera()` per riposizionare tale sequenza sulla prima coppia. Queste due operazioni vengono usate dal gestore del file. Il programma principale (non rappresentato) accede all'elenco ed al file attraverso un'unica classe.

10 Riprogettare la soluzione dell'esercizio precedente

(5)

prevedendo che l'elenco possa venire implementato con una lista oppure con una tabella, e che si applichi il pattern *Iterator* (Fig. 4) al posto delle operazioni `enumera()` e `azzera()`.

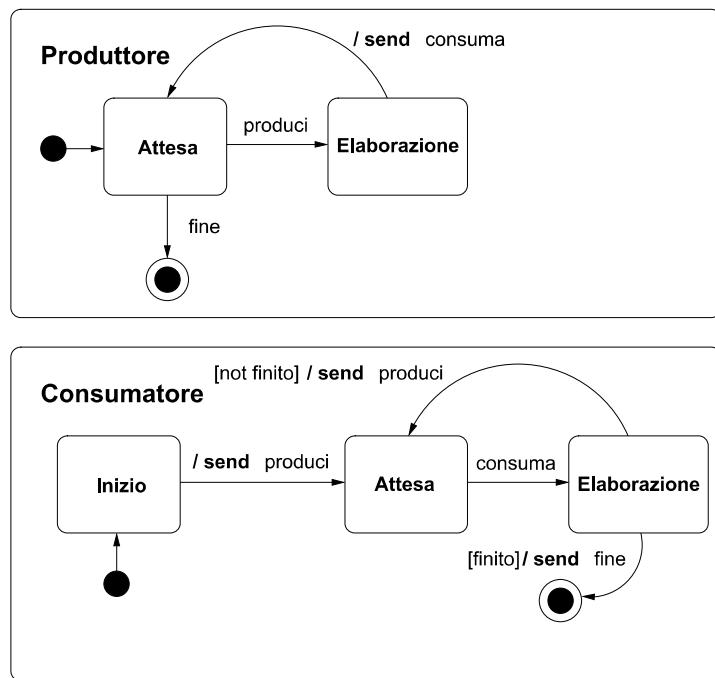


Figura 1: Domanda 6.

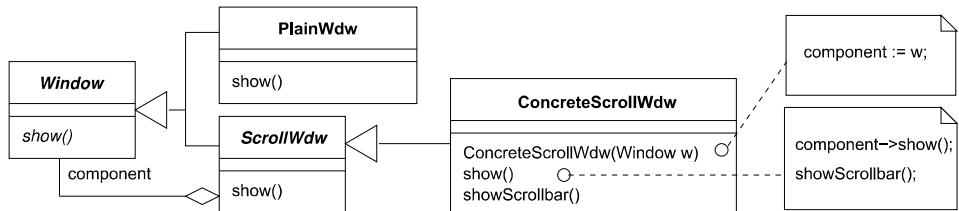


Figura 2: Domanda 7.

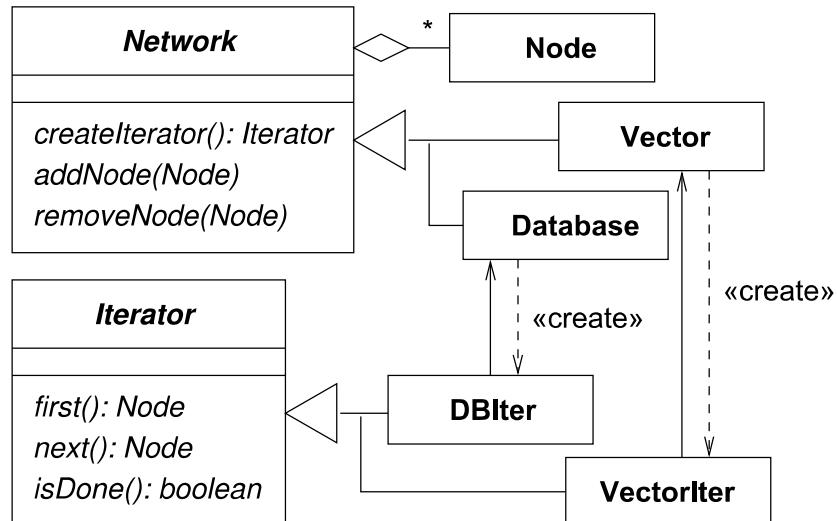


Figura 3: Domanda 8.

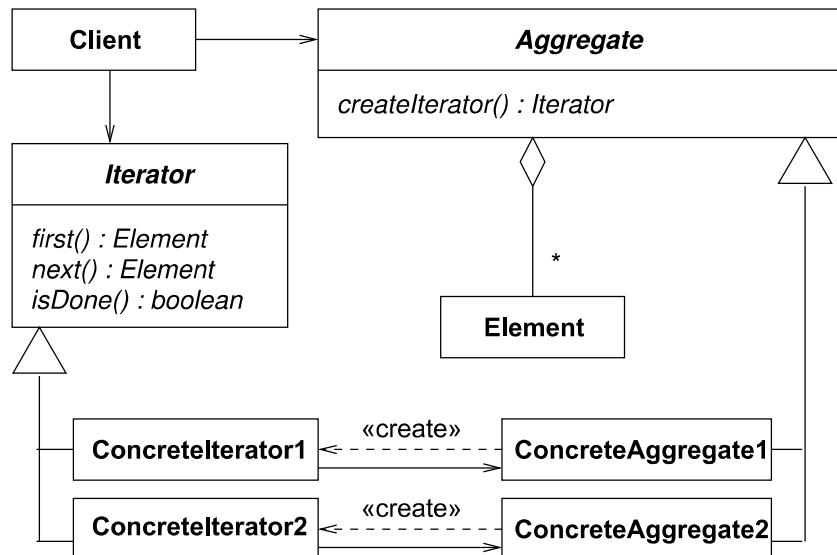


Figura 4: Domanda 10.

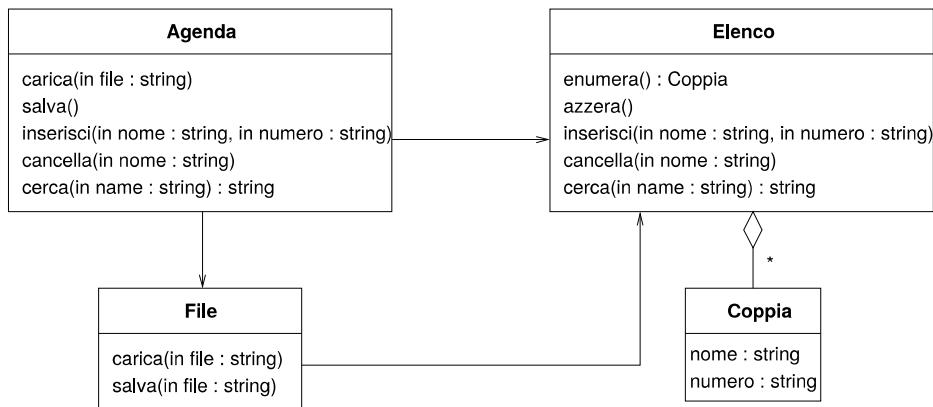


Figura 5: Domanda 9, soluzione.

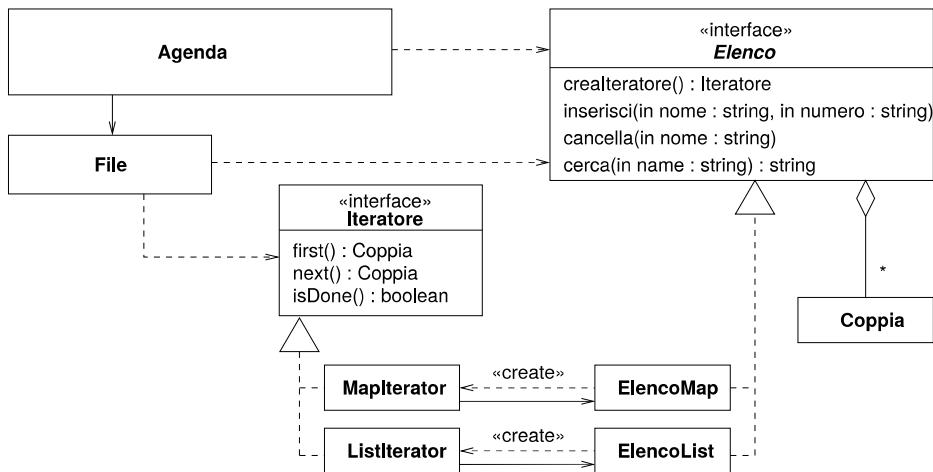


Figura 6: Domanda 10, soluzione.

Esame di Ingegneria del software

Appello del 6 giugno 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 una macchina a stati gerarchica** che specifichi quanto segue: (5)
l'interazione fra un utente ed un centralino si attiva quando un utente (chiamante) solleva la cornetta. Il chiamante può selezionare il chiamato componendo numeri di tre cifre, oppure premere un tasto che sceglie un numero memorizzato. Terminata la selezione, il centralino entra nella fase di chiamata. Quando il chiamato accetta la chiamata, inizia la fase di conversazione, che termina quando il chiamato si sconnette oppure il chiamante riaggancia, e l'interazione torna inattiva. In ogni fase attiva dell'interazione il chiamante può riagganciare, disattivandola.
- 2** Il diagramma di istanze di Fig. 1 rappresenta una particolare espressione aritmetica binaria. **Disegnare un diagramma di classi** compatibile col diagramma di Fig. 1. (5)
- 3 Disegnare il diagramma di sequenza** corrispondente al diagramma di comunicazione in Fig. 2. (5)
- 4 Con riferimento alla Fig. 3, rispondere alle domande.** (5)
L'attività **Start pump P** è concorrente a **Open valve B**.
Il sistema riceve il segnale **A and B open**.
Il sistema riceve il segnale **Start**.
La valvola **C** si può aprire solo dopo che la pompa **P** si è fermata.
Start pump P è concorrente a **Open valve C**.
- 5 Dimostrare con una tabella di verità** l'equivalenza di $a \Rightarrow b$ e $a \Rightarrow (a \Rightarrow b)$. (5)

a	b	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{1=2}$
F	F	T	T	T
F	T	F	F	T
T	F	T	T	T
T	T	T	T	T
- 6 Rispondere alle seguenti domande.** (5)
Le *precondizioni* sono responsabilità dei moduli clienti.
Le operazioni *protette* di una classe sono visibili solo ai metodi della stessa classe.
In un sistema formale completo, tutte le formule valide sono dimostrabili.
Tutte le formule valide sono vere.
I *componenti* UML rappresentano moduli fisici.

logici

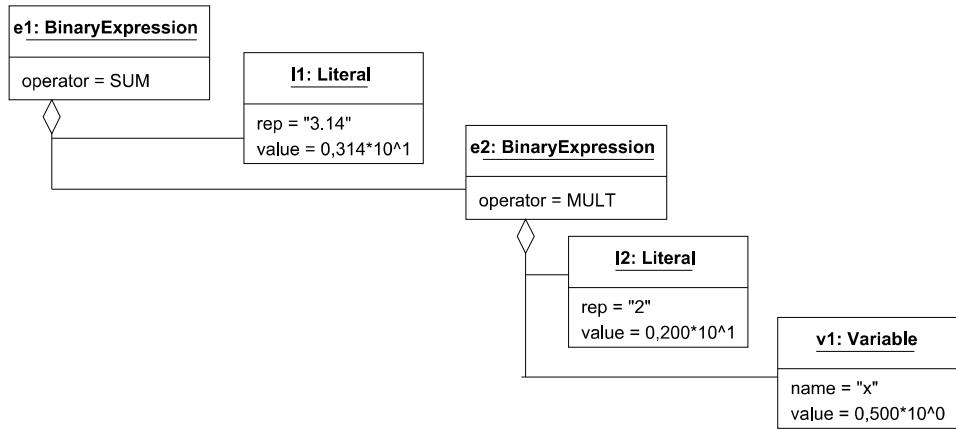


Figura 1: Domanda 2.

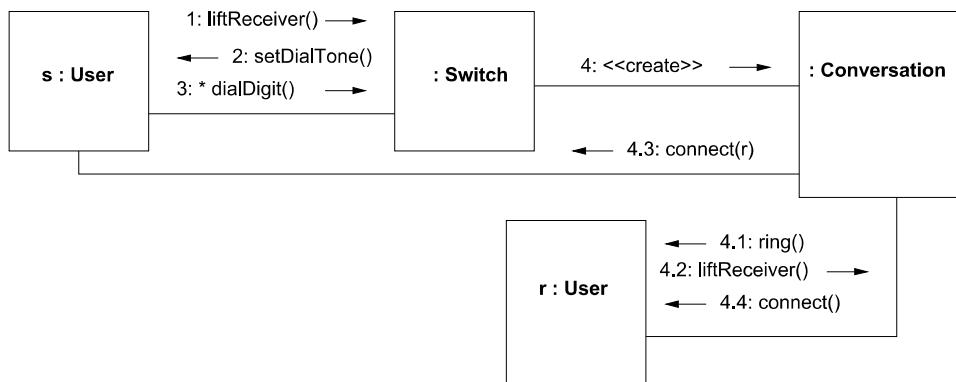


Figura 2: Domanda 3.

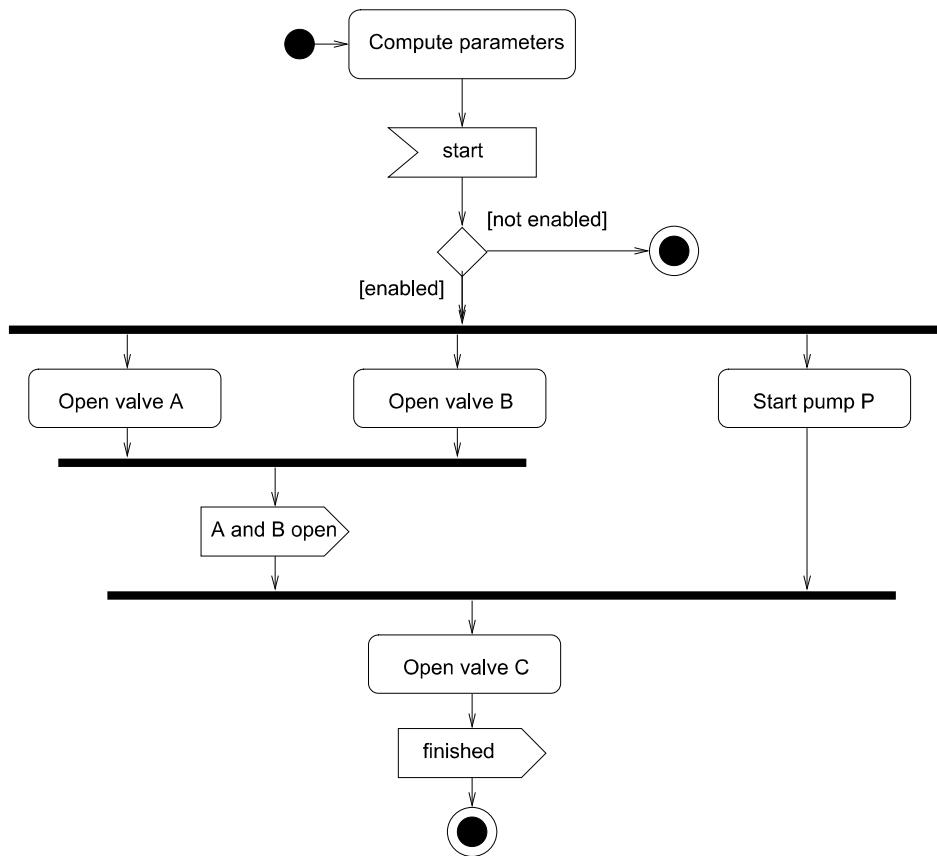
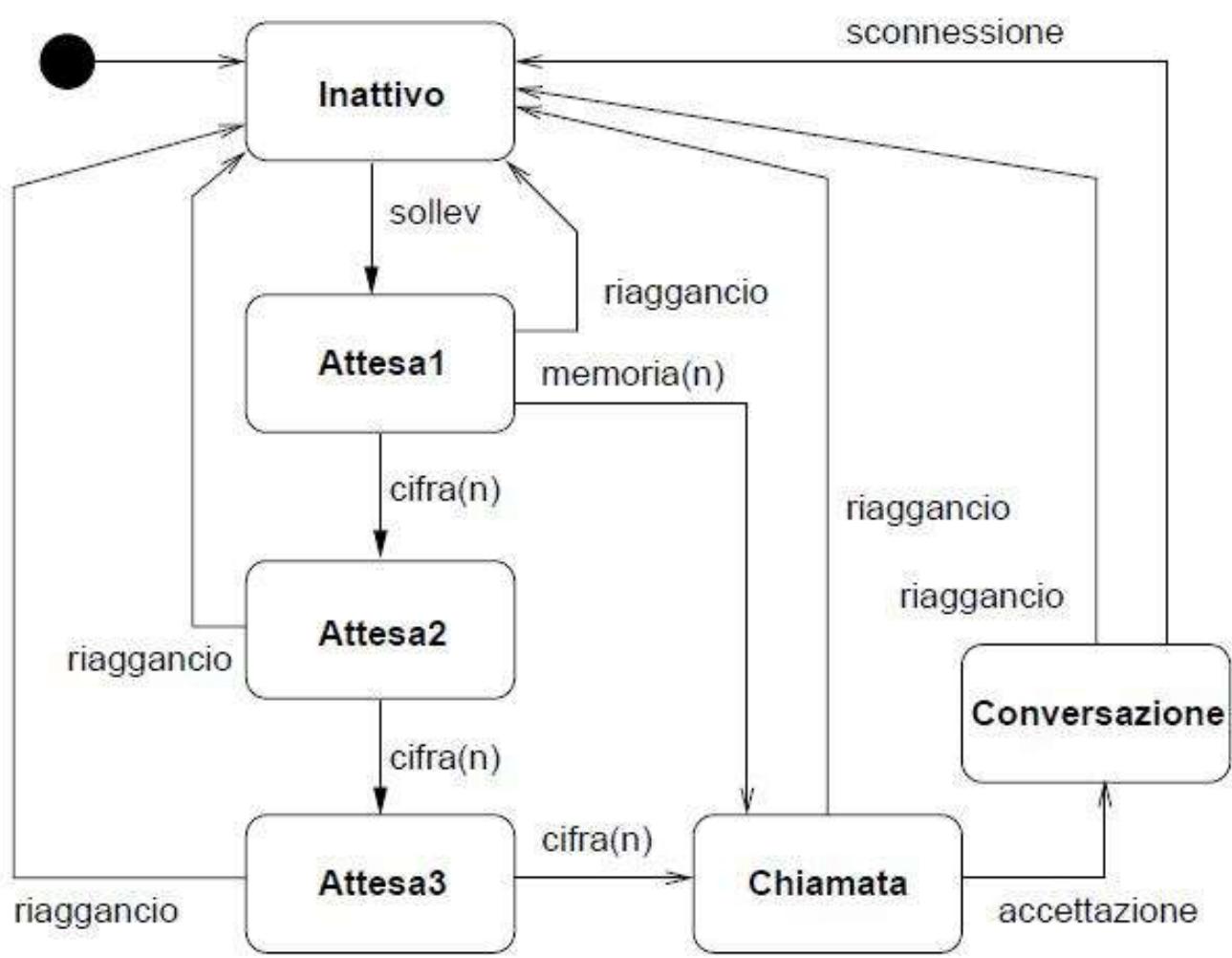
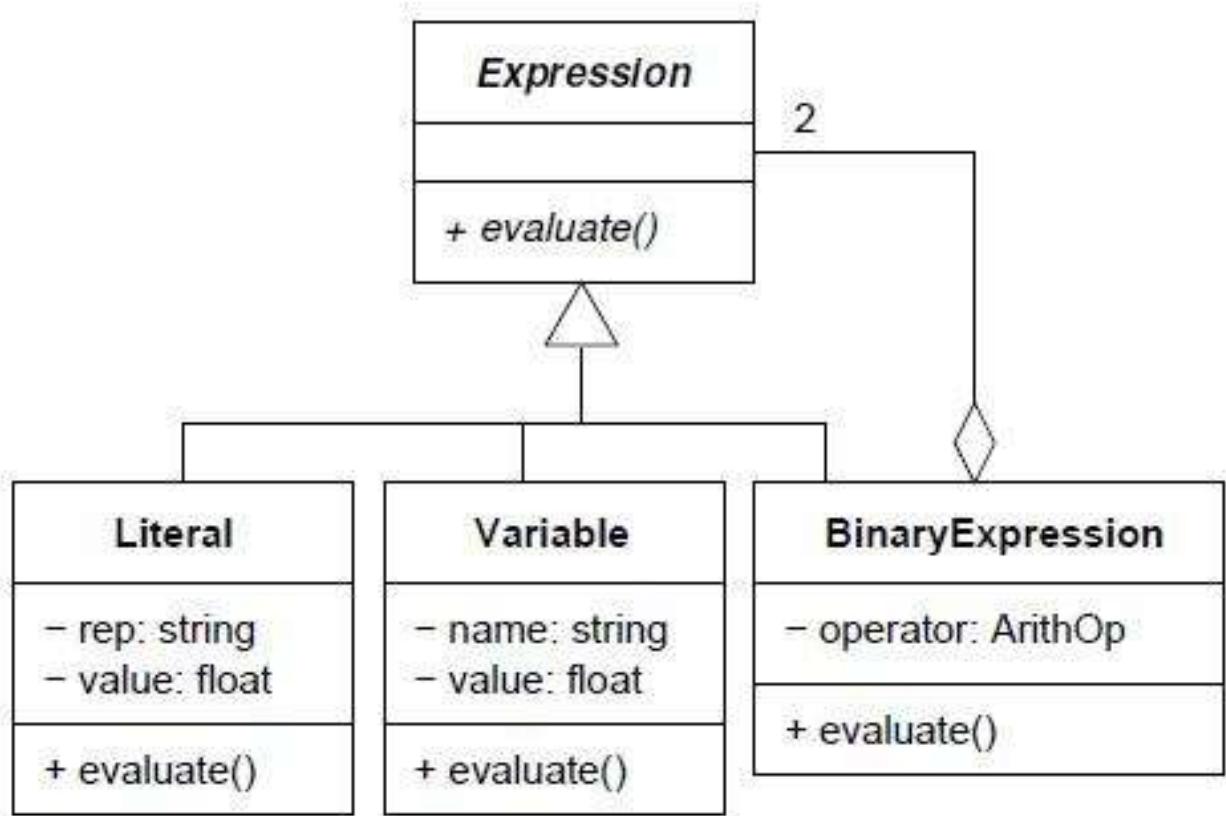


Figura 3: Domanda 4.





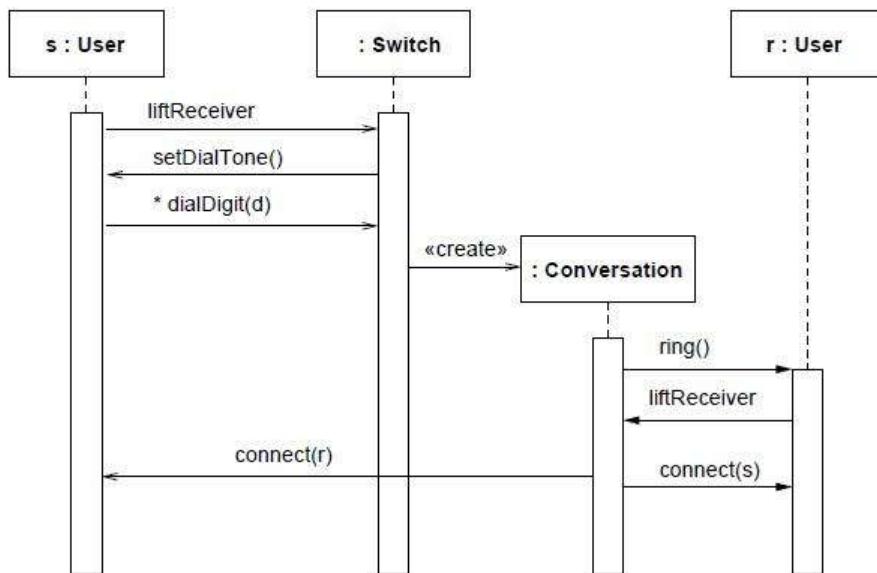


Figura 3.30: Un diagramma di sequenza.

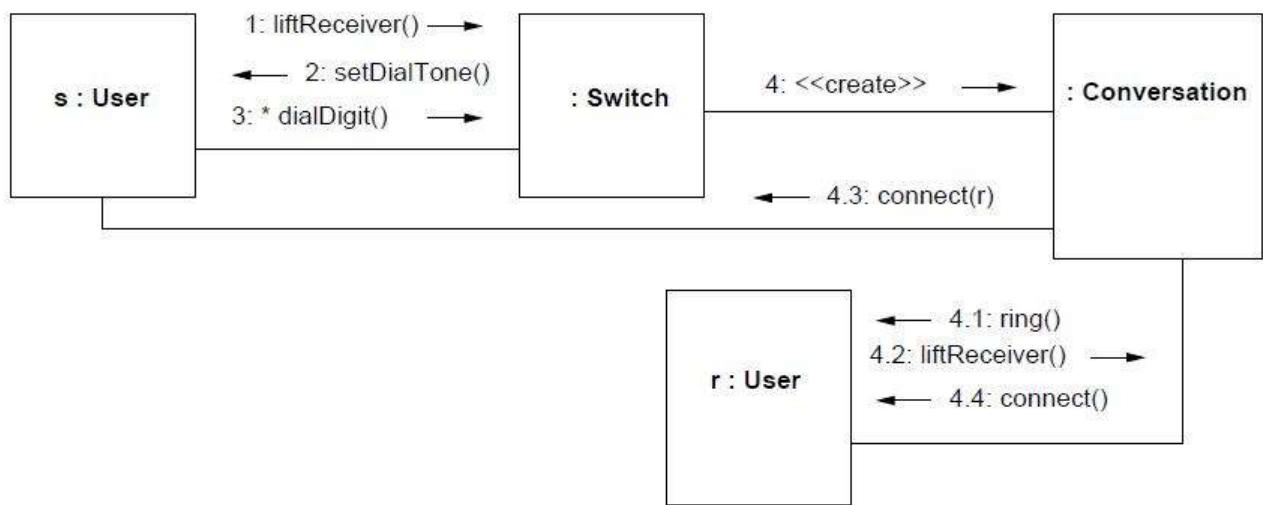


Figura 3.31: Un diagramma di comunicazione.

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.

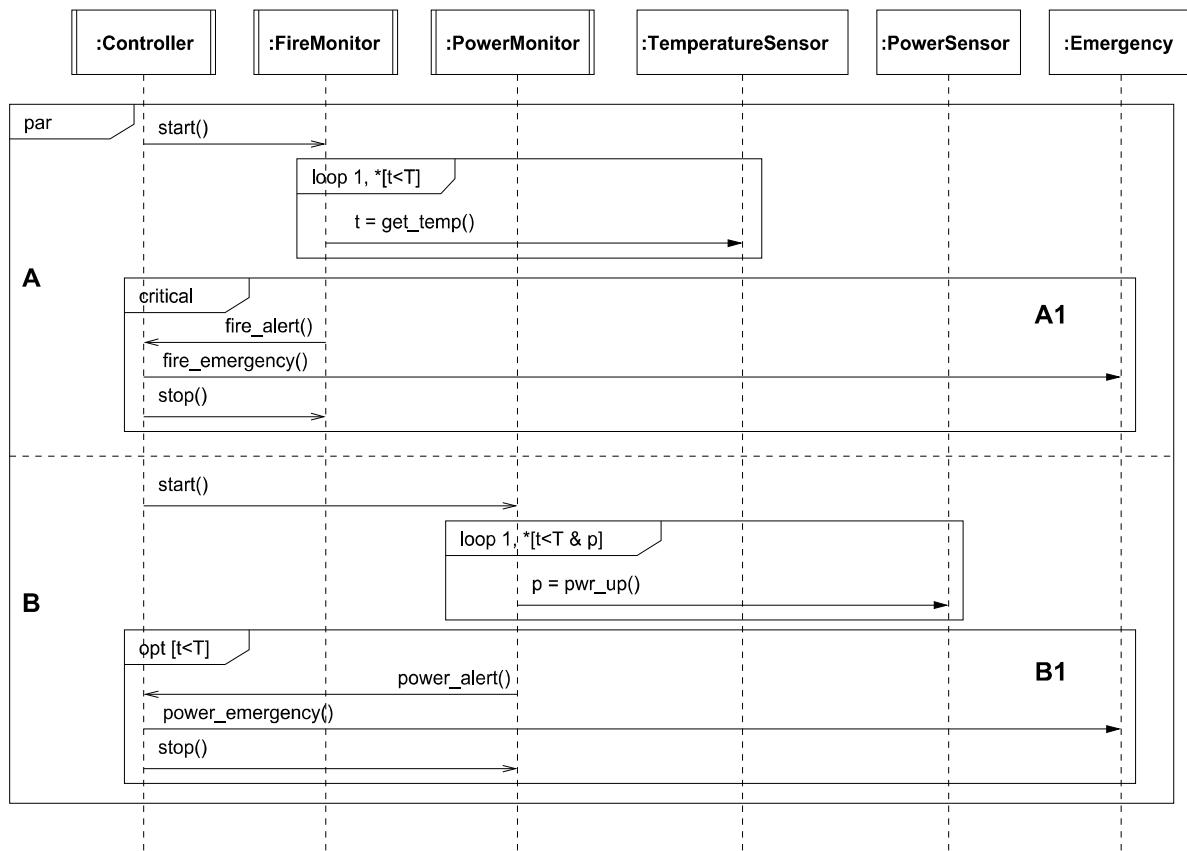


Figura 1: Domanda 4.

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

FireMonitor è un'istanza di una classe attiva.

V F

L'operazione `get_temp()` viene eseguita una volta sola.

V F

Le operazioni del frammento **B** vengono eseguite dopo il frammento **A**.

V F

Il frammento **A1** può essere interrotto.

V F

Il frammento **B1** può essere interrotto.

V F

2 Dimostrare con una tabella di verità l'equivalenza di $\neg(a \Rightarrow b) \Leftrightarrow a \wedge \neg b$. (5)

1 2

a	b	1	2	$1 \equiv 2$
F	F	F	F	T
F	T	T	T	T
T	F	F	F	T
T	T	F	F	T

- 3 Disegnare uno Statechart che descriva il seguente sistema: (5)
- Un orologio ha due modi di funzionamento: **Display**, in cui mostra l'ora, e **Setting**, in cui si rimette l'ora. Questo modo di funzionamento comprende tre sottostati: **SettingHour**, **SettingMinute**, **SettingSecond**. L'orologio ha due tasti: mode e set. Il tasto mode serve a passare ciclicamente dallo stato iniziale **Display** ai tre sottostati **Setting** (nell'ordine detto). Il tasto set serve a incrementare di 1, ogni volta che viene premuto, il valore indicato nello stato corrente. Nello stato **Display** non ha effetto. Le ore vanno da 0 a 23, i minuti e i secondi da 0 a 59.
- 4 Con riferimento alla Fig. 2, rispondere alle domande. (5)
- Display** è una classe attiva.
- Display** può invocare Counter::increment().
- Display** può invocare Counter::get_min().
- Clock** può invocare Display::reset().
- Counter** eredita da **Display**.
- 5 Una tautologia è (1)
- vera in qualsiasi interpretazione.
- falsa in qualsiasi interpretazione.
- indecidibile in qualsiasi interpretazione.
- 6 In UML, la relazione *A realizza B* significa: (1)
- A eredita da B.
- A e B hanno la stessa interfaccia.
- A implementa l'interfaccia di B.
- 7 Nel calcolo proposizionale la *funzione di valutazione* (1)
- assegna un valore ai simboli proposizionali.
- assegna un valore ai connettivi.
- assegna un valore alle formule.
- 8 Nel modello orientato agli oggetti, un *legame* è (1)
- un'istanza di un'associazione.
- un'istanza di una generalizzazione.
- uno stereotipo di associazione.
- 9 In un sistema formale completo (1)
- tutte le formule dimostrabili sono valide.
- tutte le formule valide sono dimostrabili.
- tuuti gli assiomi sono validi.
- 10 Disegnare un diagramma di classi che descriva quanto segue: (5)

Un servizio ha un'interfaccia **Interf** costituita dall'operazione process(in d: Data): ResType, che è implementata nella classe **Implen**. La classe **Adapter** ha l'operazione compute(in d: Data, out r: ResType) che chiama l'operazione process(in d: Data): ResType di un oggetto di tipo **Interf**, un costruttore, e un membro privato di tipo **IRef** che può contenere un riferimento a oggetti di tipo **Interf**. Scrivere l'implementazione (e la dichiarazione) del costruttore di **Adapter** e dell'operazione compute(in d: Data, out r: ResType) oltre a disegnare il diagramma.

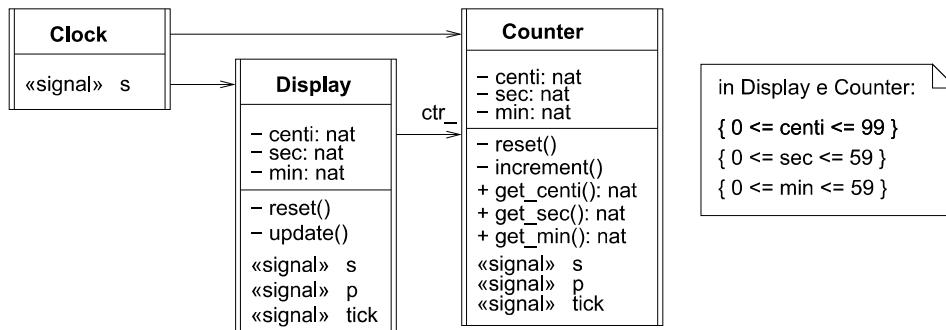
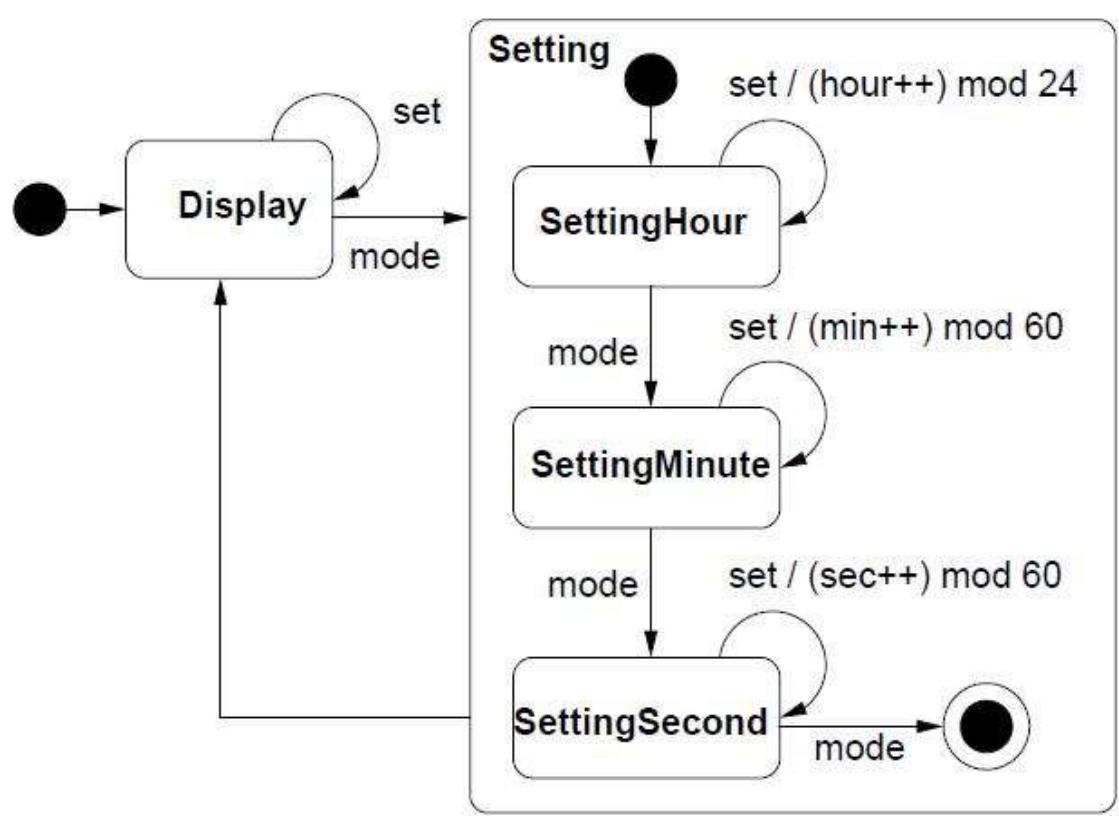


Figura 2: Domanda 4.



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.

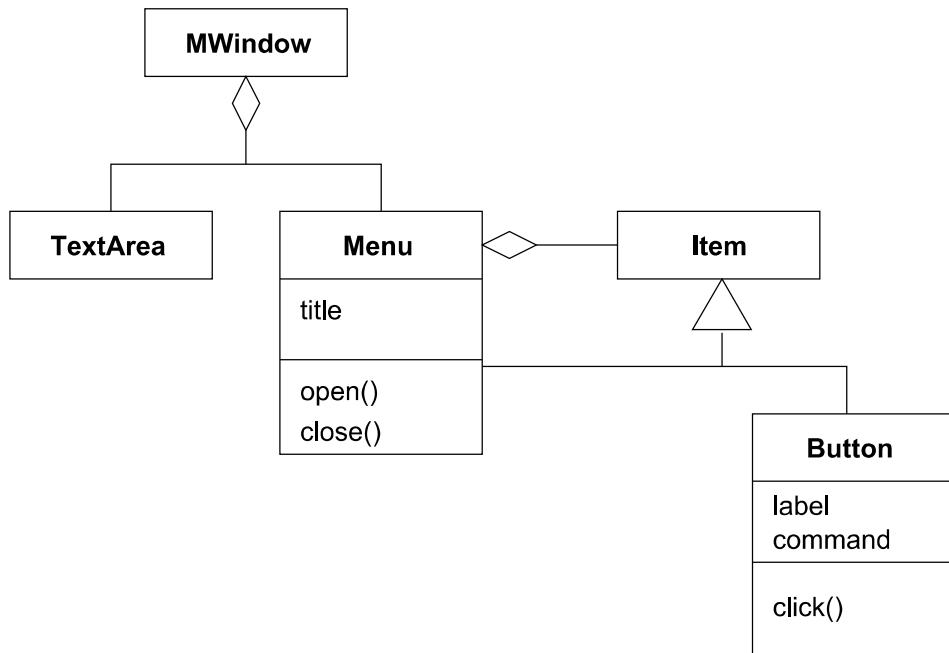


Figura 1: Domande 1–5.

- 1 In Fig. 1, (1)
Un oggetto **Menu** può contenere oggetti **Button**

La classe **Menu** deriva dalla classe **Button**

La classe **Menu** contiene la classe **Button**
- 2 In Fig. 1, (1)
La classe **Menu** deriva dalla classe **Mwindow**

Un oggetto **Mwindow** può contenere oggetti **Menu**

Un oggetto **Menu** può contenere oggetti **Mwindow**
- 3 In Fig. 1, (1)
Un oggetto **Button** può contenere oggetti **Menu**

La classe **Button** deriva dalla classe **Item**

La classe **Button** è base della classe **Item**
- 4 In Fig. 1, (1)

	La classe Item è base della classe Button	<input checked="" type="checkbox"/>
	La classe Item contiene la classe Button	<input type="checkbox"/>
	Un oggetto Button può contenere oggetti Item	<input type="checkbox"/>
5	In Fig. 1,	(1)
	Menu eredita l'operazione click	<input type="checkbox"/>
	Menu eredita l'operazione open	<input type="checkbox"/>
	Menu implementa l'operazione open	<input checked="" type="checkbox"/>
6	Disegnare una macchina a stati che specifichi quanto segue: un motore può girare in due versi, ma non può passare direttamente da un verso all'altro, dovendo essere fermato prima di invertire il movimento. Il suo controllore accetta i segnali stop , forward (senso orario) e reverse (senso antiorario).	(5)
7	Scrivere le dichiarazioni corrispondenti allo schema di Fig. 2.	(5)
8	In Fig. 3, HashTable	(1)
	implementa HTKey .	<input type="checkbox"/>
	richiede HTKey .	<input checked="" type="checkbox"/>
	offre HTKey .	<input type="checkbox"/>
9	In Fig. 3, KeyString	(1)
	realizza HTKey .	<input checked="" type="checkbox"/>
	dipende da HTKey .	<input type="checkbox"/>
	appartiene a HTKey .	<input type="checkbox"/>
10	In Fig. 3, lasciando HashTable immutata si può sostituire KeyString con un'altra classe?	(1)
	no, HashTable può usare solo chiavi KeyString .	<input type="checkbox"/>
	sí, HashTable può usare chiavi di altro tipo.	<input checked="" type="checkbox"/>
	sí, HashTable può usare chiavi di qualsiasi tipo.	<input type="checkbox"/>
11	In Fig. 3, Object	(1)
	implementa HashTable .	<input type="checkbox"/>
	deriva da HashTable .	<input type="checkbox"/>
	appartiene a HashTable .	<input checked="" type="checkbox"/>
12	In Fig. 3, put()	(1)
	è polimorfica.	<input checked="" type="checkbox"/>
	è astratta.	<input type="checkbox"/>
	è protetta.	<input type="checkbox"/>
13	Il modello a cascata è	(1)
	un metodo di progetto orientato agli oggetti	<input type="checkbox"/>
	un processo di sviluppo del SW con fasi sequenziali separate	<input checked="" type="checkbox"/>
	un linguaggio formale di specifica	<input type="checkbox"/>
14	I modelli evolutivi	(1)
	sviluppano il sistema in passi incrementali	<input checked="" type="checkbox"/>
	si basano sempre su metodi formali	<input type="checkbox"/>
	sono adatti soprattutto ad applicazioni ben conosciute	<input type="checkbox"/>
15	Le applicazioni che mantengono grandi quantità di informazioni si dicono	(1)
	orientate ai dati	<input checked="" type="checkbox"/>

	in tempo reale	<input type="checkbox"/>
	orientate agli oggetti	<input type="checkbox"/>
16	Le applicazioni che reagiscono a stimoli esterni si dicono	(1)
	orientate alle funzioni	<input type="checkbox"/>
	concorrenti	<input type="checkbox"/>
	orientate al controllo	<input checked="" type="checkbox"/>
17	L'analisi dei requisiti è	(1)
	la definizione dei sottosistemi	<input type="checkbox"/>
	la definizione delle proprietà e dei comportamenti richiesti	<input checked="" type="checkbox"/>
	la documentazione del processo si sviluppo	<input type="checkbox"/>
18	Cosa significa che il SW è “non lineare”?	(1)
	I sistemi complessi hanno un’architettura a strati.	<input type="checkbox"/>
	Piccole modifiche nel codice causano grandi cambiamenti di comportamento.	<input checked="" type="checkbox"/>
	Il grafo di controllo può contenere dei cicli.	<input type="checkbox"/>
19	Cosa s'intende per <i>information hiding</i>?	(1)
	Impedire l'accesso a dati personali.	<input type="checkbox"/>
	Impedire l'accesso a dettagli implementativi.	<input checked="" type="checkbox"/>
	Impedire l'accesso al codice sorgente.	<input type="checkbox"/>
20	Il test di unità	(1)
	Avviene di solito nella fase di codifica.	<input checked="" type="checkbox"/>
	Viene pianificato in fase di analisi e specifica dei requisiti.	<input type="checkbox"/>
	Fa parte della manutenzione del SW.	<input type="checkbox"/>
21	Un modello di processo è	(1)
	una procedura standardizzata	<input type="checkbox"/>
	una generalizzazione di una famiglia di processi di sviluppo	<input checked="" type="checkbox"/>
	una metodologia di specifica dei requisiti	<input type="checkbox"/>
22	I sistemi in tempo reale sono caratterizzati da	(1)
	condivisione di risorse.	<input type="checkbox"/>
	vincoli sui tempi di risposta.	<input checked="" type="checkbox"/>
	prestazioni elevate.	<input type="checkbox"/>

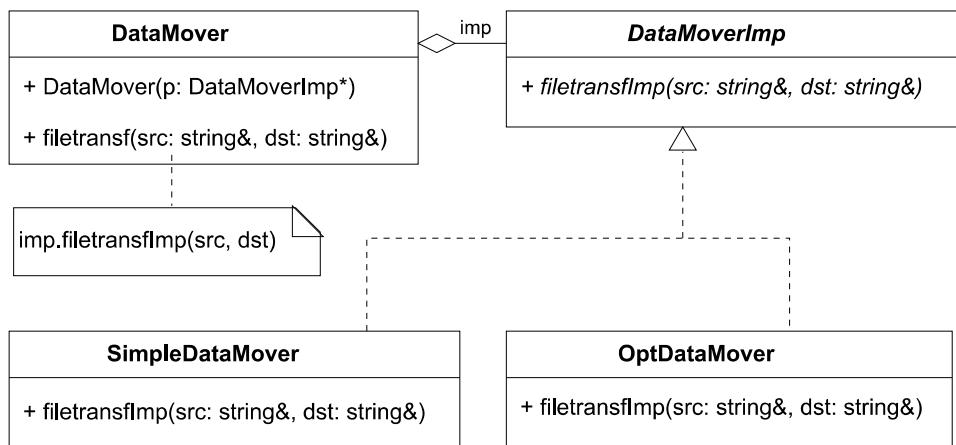


Figura 2: Domanda 7.

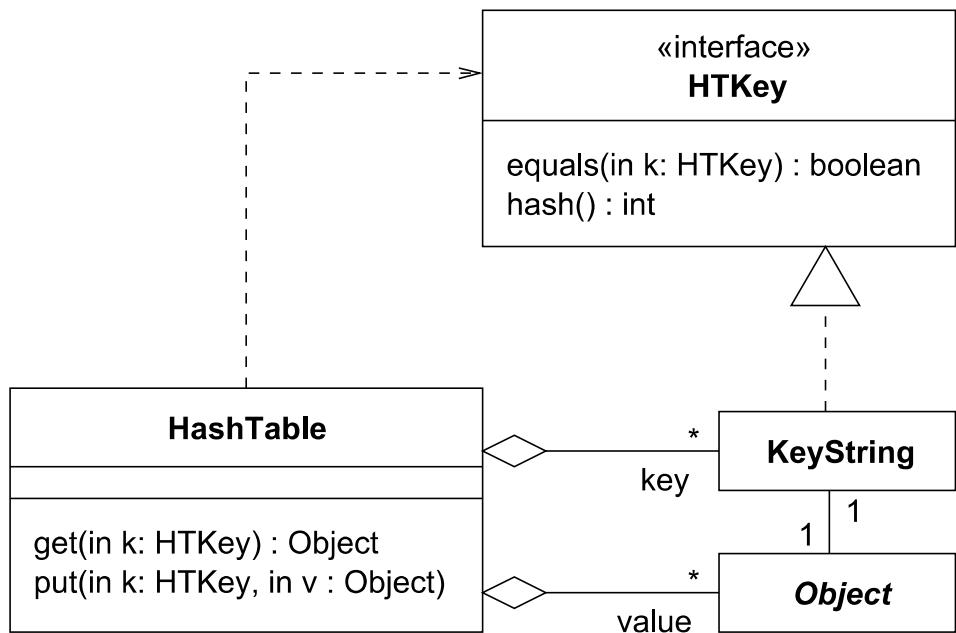


Figura 3: Domande 8–12.

Esami di Ingegneria dei sistemi software e Ingegneria del software
(ZZ304, II222, 304II)

Appello del 15 gennaio 2013

Nome e cognome:

Matricola:

Codice esame: II222 ZZ304 304II

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 un diagramma di classi che risolva il seguente problema:** (5)
Un sistema può essere un **Bus** o una **Card**. Ad un **Bus** si possono collegare zero o più **Bus** e zero o più **Card**. Ogni componente (bus o card) ha un prezzo. Vogliamo rappresentare la struttura di un sistema e calcolarne il prezzo complessivo. Applicare il design pattern Composite e indicare l'implementazione dell'operazione che calcola il prezzo.
- 2 Disegnare uno Statechart che descriva il seguente sistema:** (5)
Un orologio ha due modi di funzionamento: **Display**, in cui mostra l'ora, e **Setting**, in cui si rimette l'ora. Questo modo di funzionamento comprende tre sottostati: **SettingHour**, **SettingMinute**, **SettingSecond**. L'orologio ha due tasti: mode e set. Il tasto mode serve a passare ciclicamente dallo stato iniziale **Display** ai tre sottostati **Setting** (nell'ordine detto). Il tasto set serve a incrementare di 1, ogni volta che viene premuto, il valore indicato nello stato corrente; nello stato **Display** non ha effetto. Le ore sono rappresentate da una variabile che va da 0 a 23, i minuti e i secondi da due variabili che vanno da 0 a 59.
- 3 Un modello di processo è** (1)
una procedura standardizzata
una generalizzazione di una famiglia di processi di sviluppo
una metodologia di specifica dei requisiti
- 4 I requisiti funzionali** (1)
si specificano con i Diagrammi di Flusso dei Dati
specificano le caratteristiche di qualità
descrivono cosa deve fare il sistema
- 5 Un sistema formale è corretto se** (1)
tutte le formule dimostrabili sono vere
non contiene errori
tutte le formule vere sono dimostrabili

6	Le Espressioni Regolari	(1)
	sono delle formule logiche	<input type="checkbox"/>
	sono un formalismo di specifica dei dati di tipo semantico	<input type="checkbox"/>
	sono un formalismo di specifica dei dati di tipo sintattico	<input checked="" type="checkbox"/>
7	Negli Automi a Stati Finiti le uscite	(1)
	dipendono dalla marcatura	<input type="checkbox"/>
	dipendono dallo stato e dall'ingresso	<input checked="" type="checkbox"/>
	dipendono dalle condizioni di guardia	<input type="checkbox"/>
8	Cosa significa che il SW è “non lineare”?	(1)
	I sistemi complessi hanno un’architettura a strati.	<input type="checkbox"/>
	Piccoli cambiamenti nel codice causano grandi cambiamenti di comportamento.	<input checked="" type="checkbox"/>
	Il grafo di controllo può contenere dei cicli.	<input type="checkbox"/>
9	Cosa s'intende per <i>information hiding</i>?	(1)
	Impedire l'accesso a dati personali.	<input type="checkbox"/>
	Impedire l'accesso a dettagli implementativi.	<input checked="" type="checkbox"/>
	Impedire l'accesso al codice sorgente.	<input type="checkbox"/>
10	Il test di unità	(1)
	Avviene di solito nella fase di codifica.	<input checked="" type="checkbox"/>
	Viene pianificato in fase di analisi e specifica dei requisiti.	<input type="checkbox"/>
	Fa parte della manutenzione del SW.	<input type="checkbox"/>
11	Nelle reti di Petri lo stato del sistema è rappresentato	(1)
	da un place particolare.	<input type="checkbox"/>
	dall'insieme delle transizioni abilitate.	<input type="checkbox"/>
	dalla marcatura della rete.	<input checked="" type="checkbox"/>
12	I sistemi in tempo reale sono caratterizzati da	(1)
	condivisione di risorse.	<input type="checkbox"/>
	vincoli sui tempi di risposta.	<input checked="" type="checkbox"/>
	prestazioni elevate.	<input type="checkbox"/>
13	In Fig. 1, Time	(2)
	viene definita dallo sviluppatore.	<input checked="" type="checkbox"/>
	è predefinita.	<input type="checkbox"/>
	viene generata dal compilatore IDL.	<input type="checkbox"/>
14	In Fig. 1, Time_impl	(2)
	viene definita dallo sviluppatore.	<input checked="" type="checkbox"/>
	è predefinita.	<input type="checkbox"/>
	viene generata dal compilatore IDL.	<input type="checkbox"/>
15	In Fig. 1, POA_Time	(2)
	viene definita dallo sviluppatore.	<input type="checkbox"/>
	è predefinita.	<input type="checkbox"/>
	viene generata dal compilatore IDL.	<input checked="" type="checkbox"/>
16	In Fig. 1, Time_impl	(2)
	è scritta in IDL.	<input type="checkbox"/>
	è scritta nello stesso linguaggio del server.	<input checked="" type="checkbox"/>
	è scritta nello stesso linguaggio del cliente.	<input type="checkbox"/>

17 In Fig. 1, **POA_Time**

(2)



è un proxy.

è uno scheletro.

è un *object adapter*.

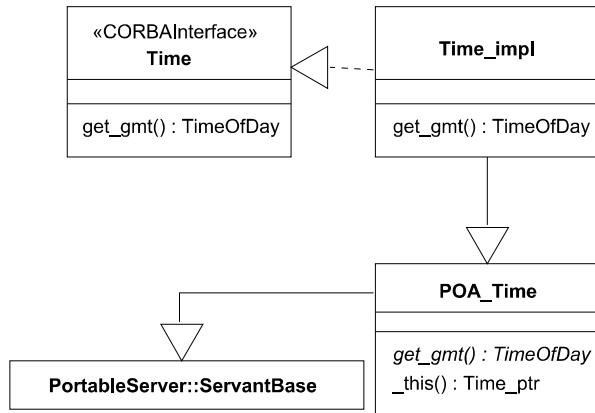


Figura 1: Domande 13–17.

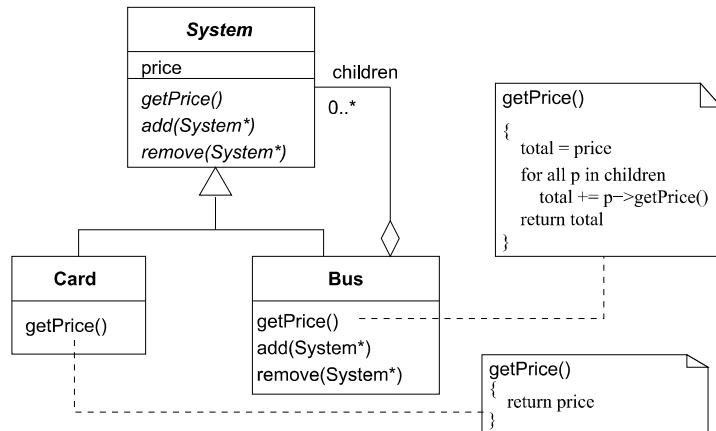


Figura 2: Domanda 1, soluzione.

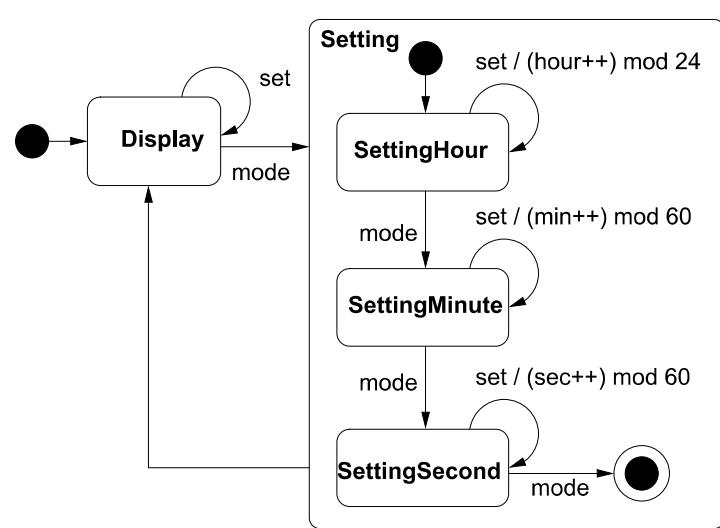


Figura 3: Domanda 2, soluzione.

Esame di Ingegneria del software
(360II, 374II)

Appello del 15 gennaio 2013

Nome e cognome:

Matricola:

Codice esame: 360II 374II

Il punteggio relativo a ciascuna domanda, indicato fra parentesi, è in trentesimi. Alcune domande hanno due punteggi, uno dei quali negativo, valido per le risposte sbagliate. I candidati devono consegnare entro un'ora dall'inizio della prova.

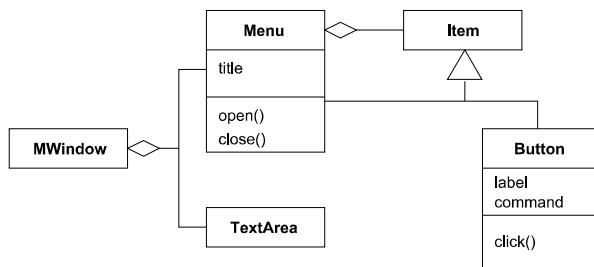


Figura 1: Domande 1–5.

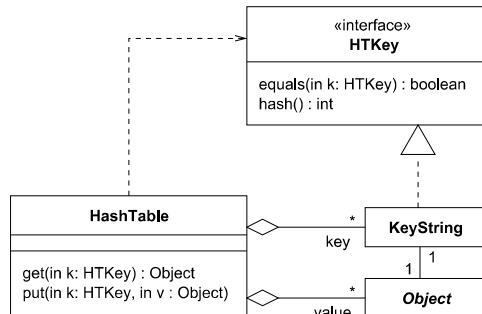


Figura 2: Domande 6–10.

- | | |
|--|---|
| <p>1 In Fig. 1,</p> <p>Un oggetto Menu può contenere oggetti Button</p> <p>La classe Menu deriva dalla classe Mwindow</p> <p>La classe Menu contiene la classe Button</p> | <p>(1, -1)</p> <p><input checked="" type="checkbox"/></p> <p><input type="checkbox"/></p> <p><input type="checkbox"/></p> |
| <p>2 In Fig. 1,</p> <p>La classe Menu deriva dalla classe Mwindow</p> <p>Un oggetto Mwindow può contenere oggetti Menu</p> <p>Un oggetto Menu può contenere oggetti Mwindow</p> | <p>(1, -1)</p> <p><input type="checkbox"/></p> <p><input checked="" type="checkbox"/></p> <p><input type="checkbox"/></p> |

3	In Fig. 1,	(1, -1)
	Un oggetto Button può contenere oggetti Menu	<input type="checkbox"/>
	La classe Button deriva dalla classe Item	<input checked="" type="checkbox"/>
	La classe Button è base della classe Item	<input type="checkbox"/>
4	In Fig. 1,	(1, -1)
	La classe Item è base della classe Button	<input checked="" type="checkbox"/>
	La classe Item contiene la classe Button	<input type="checkbox"/>
	Un oggetto Button può contenere oggetti Item	<input type="checkbox"/>
5	In Fig. 1,	(1, -1)
	Menu eredita l'operazione click	<input type="checkbox"/>
	Menu eredita l'operazione open	<input type="checkbox"/>
	Menu implementa l'operazione open	<input checked="" type="checkbox"/>
6	In Fig. 2, HashTable	(1, -1)
	implementa HTKey .	<input type="checkbox"/>
	richiede HTKey .	<input checked="" type="checkbox"/>
	offre HTKey .	<input type="checkbox"/>
7	In Fig. 2, KeyString	(1, -1)
	realizza HTKey .	<input checked="" type="checkbox"/>
	dipende da HTKey .	<input type="checkbox"/>
	appartiene a HTKey .	<input type="checkbox"/>
8	In Fig. 2, lasciando HashTable immutata si può sostituire KeyString con un'altra classe?	(1, -1)
	no, HashTable può usare solo chiavi KeyString .	<input type="checkbox"/>
	sí, HashTable può usare chiavi di altro tipo.	<input checked="" type="checkbox"/>
	sí, HashTable può usare chiavi di qualsiasi tipo.	<input type="checkbox"/>
9	In Fig. 2, Object	(1, -1)
	implementa HashTable .	<input type="checkbox"/>
	deriva da HashTable .	<input type="checkbox"/>
	appartiene a HashTable .	<input checked="" type="checkbox"/>
10	In Fig. 2, put()	(1, -1)
	è polimorfica.	<input checked="" type="checkbox"/>
	è astratta.	<input type="checkbox"/>
	è protetta.	<input type="checkbox"/>
11	Cosa significa che il SW è “non lineare”?	(1)
	I sistemi complessi hanno un’architettura a strati.	<input type="checkbox"/>
	Piccoli cambiamenti nel codice causano grandi cambiamenti di comportamento.	<input checked="" type="checkbox"/>
	Il grafo di controllo può contenere dei cicli.	<input type="checkbox"/>
12	Cosa s'intende per <i>information hiding</i>?	(1)
	Impedire l'accesso a dati personali.	<input type="checkbox"/>
	Impedire l'accesso a dettagli implementativi.	<input checked="" type="checkbox"/>
	Impedire l'accesso al codice sorgente.	<input type="checkbox"/>
13	Il test di unità	(1)
	Avviene di solito nella fase di codifica.	<input checked="" type="checkbox"/>
	Viene pianificato in fase di analisi e specifica dei requisiti.	<input type="checkbox"/>

- Fa parte della manutenzione del SW. (1)
- 14 I sistemi in tempo reale sono caratterizzati da** (1)
 condivisione di risorse.

 vincoli sui tempi di risposta.

 prestazioni elevate.
- 15 Una fase è:** (1)
 un periodo in cui si svolge un'attività.

 un obiettivo da realizzare.

 un'attività prevista dalle specifiche.
- 16 Disegnare una macchina a stati** che specifichi quanto segue: un motore (5)
 può girare in due versi, ma non può passare direttamente da un verso all'altro,
 dovendo essere fermato prima di invertire il movimento. Il suo controllore
 accetta i segnali stop, forward e reverse.
- 17 Un Editor usa degli elementi Immagine** (5)
 che offrono le operazioni draw(), che disegna l'immagine, e getExtent(), che
 restituisce una struttura BBox con le dimensioni dell'immagine. Un'immagine
 può essere reale, cioè rappresentata completamente, oppure essere un segna-
 posto contenente le dimensioni dell'immagine e il nome del file da cui caricare
 l'immagine. Inizialmente l'editor inserisce nel documento solo dei segnaposto,
 e crea le immagini reali solo quando devono essere mostrate. I costruttori delle
 classi usate per le immagini prendono come argomento il nome del file; per le
 immagini reali, il costruttore copia il file in un campo di tipo data.
 Applicando il pattern *Proxy* (Fig. 3), **disegnare un diagramma delle classi**
corrispondente a quanto descritto.
- 18 Con riferimento all'esercizio precedente:** (5)
 scrivere in C++ le dichiarazioni delle classi;
 implementare l'operazione draw() della classe usata per i segnaposto.

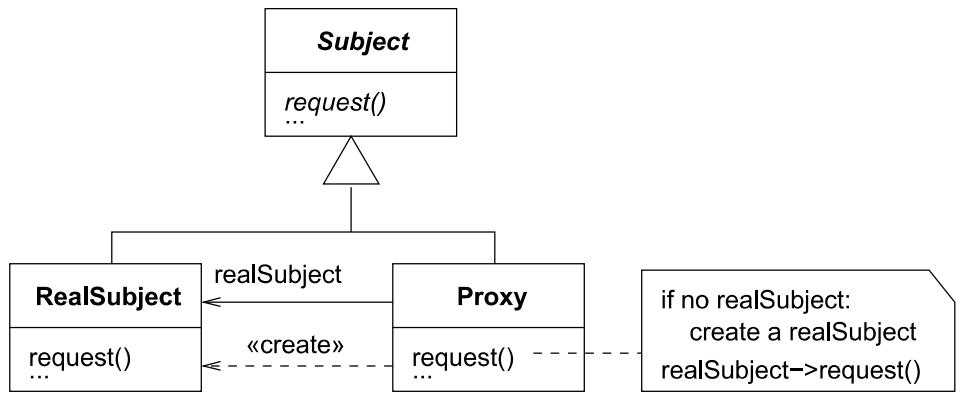


Figura 3: Domanda 17.

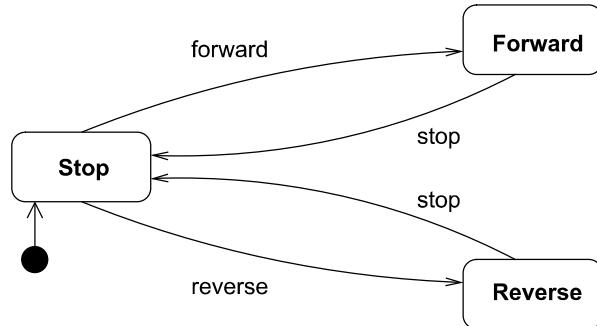


Figura 4: Domanda 16, soluzione.

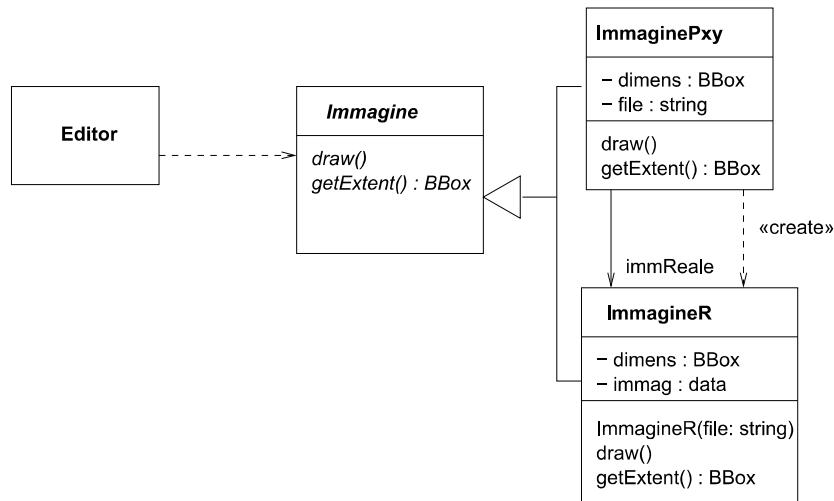


Figura 5: Domanda 17, soluzione.

```

class Immagine {
public:
    virtual void draw() = 0;
    virtual BBox getExtent() = 0;
};

class ImmagineR : public Immagine {
    BBox dimens_;
    data immag_;
public:
    ImmagineR(string f);
    virtual void draw();
    virtual BBox getExtent();
};

class ImmaginePxy : public Immagine {
    BBox dimens_;
    string file_;
    ImmagineR* immReale_;
public:
    ImmaginePxy(string f);
    virtual void draw();
    virtual BBox getExtent();
};

void
ImmaginePxy::
draw()
{
    if (immReale_ == 0)
        immReale_ = new ImmagineR(file_);
    immReale_->draw();
}

```

Figura 6: Domanda 18, soluzione.

Esame di Ingegneria del software

Appello del 7 ottobre 2017

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.

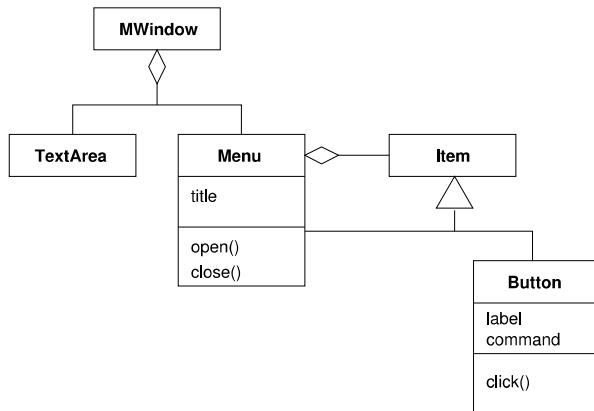


Figura 1: Domande 1–5.

- 1 In Fig. 1, (1)
 - Un oggetto **Menu** può contenere oggetti **Button**
 - La classe **Menu** deriva dalla classe **Button**
 - La classe **Menu** contiene la classe **Button**
- 2 In Fig. 1, (1)
 - La classe **Menu** deriva dalla classe **Mwindow**
 - Un oggetto **Mwindow** può contenere oggetti **Menu**
 - Un oggetto **Menu** può contenere oggetti **Mwindow**
- 3 In Fig. 1, (1)
 - Un oggetto **Button** può contenere oggetti **Menu**
 - La classe **Button** deriva dalla classe **Item**
 - La classe **Button** è base della classe **Item**
- 4 In Fig. 1, (1)
 - La classe **Item** è base della classe **Button**
 - La classe **Item** contiene la classe **Button**
 - Un oggetto **Button** può contenere oggetti **Item**
- 5 In Fig. 1, (1)
 - Menu** eredita l'operazione **click**
 - Menu** eredita l'operazione **open**

	Menu implementa l'operazione open	<input checked="" type="checkbox"/>
6	Disegnare una macchina a stati che specifichi quanto segue: un motore può girare in due versi, ma non può passare direttamente da un verso all'altro, dovendo essere fermato prima di invertire il movimento. Il suo controllore accetta i segnali stop , forward e reverse .	(5)
7	Scrivere le dichiarazioni corrispondenti allo schema di Fig. 3.	(5)
8	In Fig. 4, HashTable implementa HTKey . richiede HTKey . offre HTKey .	(1) <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
9	In Fig. 4, KeyString realizza HTKey . dipende da HTKey . appartiene a HTKey .	(1) <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
10	In Fig. 4, lasciando HashTable immutata si può sostituire KeyString con un'altra classe? no, HashTable può usare solo chiavi KeyString . sí, HashTable può usare chiavi di altro tipo. sí, HashTable può usare chiavi di qualsiasi tipo.	(1) <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
11	In Fig. 4, Object implementa HashTable . deriva da HashTable . appartiene a HashTable .	(1) <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
12	In Fig. 4, put() è polimorfica. è astratta. è protetta.	(1) <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
13	Il modello a cascata è un metodo di progetto orientato agli oggetti un processo di sviluppo del SW con fasi sequenziali separate un linguaggio formale di specifica	(1) <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>
14	I modelli evolutivi sviluppano il sistema in passi incrementali si basano sempre su metodi formali sono adatti soprattutto ad applicazioni ben conosciute	(1) <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
15	Le applicazioni che mantengono grandi quantità di informazioni si dicono orientate ai dati in tempo reale orientate agli oggetti	(1) <input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
16	Le applicazioni che reagiscono a stimoli esterni si dicono orientate alle funzioni concorrenti orientate al controllo	(1) <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>

17	L'analisi dei requisiti è	(1)
	la definizione dei sottosistemi	<input type="checkbox"/>
	la definizione delle proprietà e dei comportamenti richiesti	<input checked="" type="checkbox"/>
	la documentazione del processo si sviluppo	<input type="checkbox"/>
18	Cosa significa che il SW è “non lineare”?	(1)
	I sistemi complessi hanno un’architettura a strati.	<input type="checkbox"/>
	Piccole modifiche nel codice causano grandi cambiamenti di comportamento.	<input checked="" type="checkbox"/>
	Il grafo di controllo può contenere dei cicli.	<input type="checkbox"/>
19	Cosa s'intende per <i>information hiding</i>?	(1)
	Impedire l'accesso a dati personali.	<input type="checkbox"/>
	Impedire l'accesso a dettagli implementativi.	<input checked="" type="checkbox"/>
	Impedire l'accesso al codice sorgente.	<input type="checkbox"/>
20	Il test di unità	(1)
	Avviene di solito nella fase di codifica.	<input checked="" type="checkbox"/>
	Viene pianificato in fase di analisi e specifica dei requisiti.	<input type="checkbox"/>
	Fa parte della manutenzione del SW.	<input type="checkbox"/>
21	Un modello di processo è	(1)
	una procedura standardizzata	<input type="checkbox"/>
	una generalizzazione di una famiglia di processi di sviluppo	<input checked="" type="checkbox"/>
	una metodologia di specifica dei requisiti	<input type="checkbox"/>
22	I sistemi in tempo reale sono caratterizzati da	(1)
	condivisione di risorse.	<input type="checkbox"/>
	vincoli sui tempi di risposta.	<input checked="" type="checkbox"/>
	prestazioni elevate.	<input type="checkbox"/>

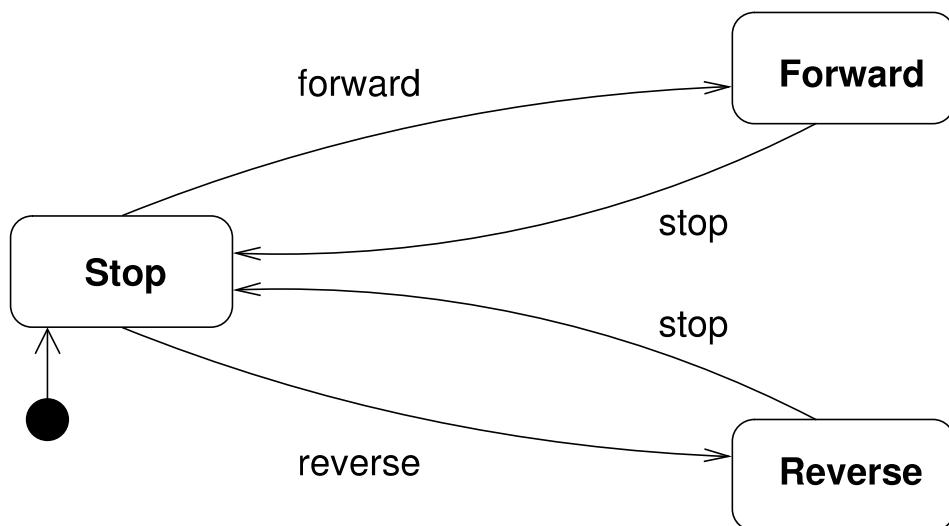


Figura 2: Domanda 6, soluzione.

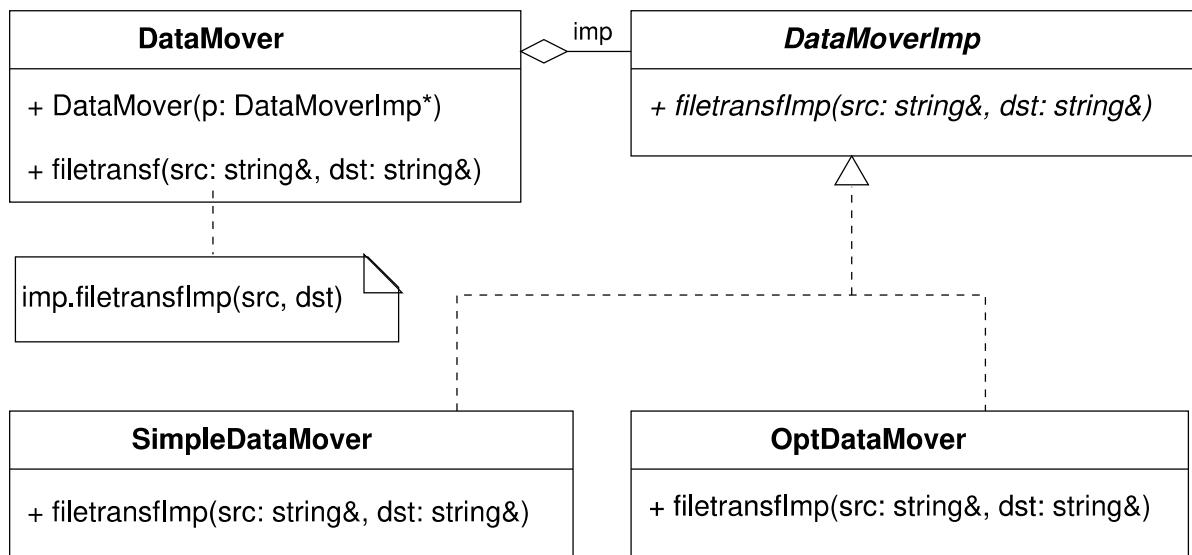


Figura 3: Domanda 7.

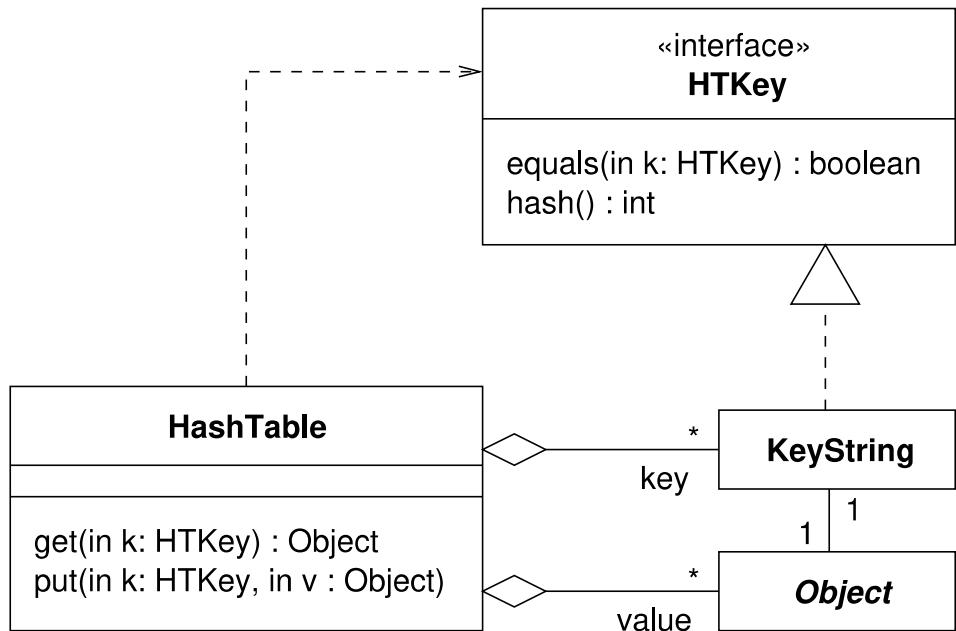


Figura 4: Domande 8–12.

```

#include <iostream>
#include <string>

using namespace std;

class DataMoverImp {
public:
    virtual void filetransfImp(const string& src, const string& dst) = 0;
};

class SimpleDataMover : public DataMoverImp {
public:
    void filetransfImp(const string& src, const string& dst);
};

class OptDataMover : public DataMoverImp {
public:
    void filetransfImp(const string& src, const string& dst);
};

class DataMover {
    DataMoverImp* imp;
public:
    DataMover(DataMoverImp* p) : imp(p) {};
    void filetransf(const string& src, const string& dst);
};

```

Figura 5: Domanda 7, soluzione.

- Se i dati di test si scelgono in base ai requisiti: SI FA UN TEST FUNZIONALE
- “Polimorfismo” significa: AVERE PIÙ FUNZIONI DI UNA STESSA INTERFAZIA
- Una formula deducibile usando solo gli assiomi : UN TEOREMA
- Negli statechart i segnali sono: UN TIPO DI EVENTI
- Nelle reti di Petri la marcatura è: UNA FUNZIONE CHE ASSOCIA POSTI AI NUMERI NATURALI (P->N)
- Quale di queste stringhe viene riconosciuta dall’ER: ‘[1-9].[0-9]+[EE](+-)?[0-9]+?’
- Se, In una rete di Petri, una transazione u è abilitata solo dopo lo scatto di transizione t, LE DUE TRANSIZIONI SONO IN SEQUENZA
- Negli Statechart un’azione è: UN COMPORTAMENTO NON INTERROMPIBILE
- Nelle architetture CORBA, un proxy viene usato: SUL LATO CLIENTE
- Una classe si dice concreta se: È INTERAMENTE IMPLEMENTATA
- Con l’eredità multipla: UNA CLASSE PUÒ DERIVARE DIRETTAMENTE DA PIÙ BASI
- Il controllo di qualità avviene: NEL CORSO DI TUTTO IL PROCESSO DI SVILUPPO
- L’insieme delle formule di linguaggio è definite: DALLA SINTESI
- In UML l’architettura fisica si rappresenta: CON I DIAGRAMMI DI DEPLOYMENT
- In C++ le operazioni polimorfiche sono: VIRTUALI
- Il test funzionale è basato: SULLE SPECIFICHE
- Un compilatore IDL: PRODUCE CODICE SORGENTE
- Una formula ben formata: È SINTATTICAMENTE CORRETTA
- Si dice che il SW è malleabile perchè: I PROGRAMMI SONO FACILMENTE MODIFICABILI
- Definizione e specifica dei requisiti: HANNO DUE DIVERSI LIVELLI DI DETTAGLI
- Il CVS è: UNO STRUMENTO PER LA GESTIONE DELLE VERSIONI DEL CODICE SORGENTE
- Nel modello Cleanroom: GLI SVILUPPATORI VERIFICANO IL CODICE STATICAMENTE
- Se p è un place, *p è: L’INSIEME DELLE TRANSIZIONI DI INGRESSO DI P
- L’implementazione di un oggetto CORBA si chiama: SERVANT
- Il piano di test di Sistema: DEFINISCE I TEST DI CONVALIDA
- una specifica si dice formula se: È RIGOROSA
- Un’espressione regolare: DESCRIVE UN INSIEME DI SEQUENZE DI SIMBOLI
- un modulo si dice coeso se: OFFRE SERVIZI OMOGENEI
- Il test di unità: AVVIENE DI SOLITO NELLA FASE DI CODIFICA
- Cosa significa che il SW è “non lineare”? : PICCOLI CAMBIAMENTI NEL CODICE PORTANO A GRANDI CAMBIAMENTI DI COMPORTAMENTO
- Nelle reti di Petri lo stato del Sistema è rappresentato: DALLA MARCATURA DELLA RETE
- I sistemi in tempo reale sono caratterizzati da: VINCOLI SUI TEMPI DI RISPOSTA
- Cosa s’intende per information hiding?: IMPEDIRE L’ACCESSO A DETTAGLI IMPLEMENTATIVI
- Il valore di verità di una tautologia non dipende: DALLA FUNZIONE DI VALUTAZIONE
- Il diagramma delle classi descrive un Sistema dal punto di vista: STATICO
- Negli Statechart un cambiamento è: UN EVENTO
- Il test di stress è: UN TEST DI SISTEMA
- Un processo di sviluppo è: UN INSIEME DI ATTIVITÀ PIANIFICATE
- L’ER [_A-Za-z][_A-Za-z0-9]*=(‘ ‘|’\t’)*[_A-Za-z0-9]+ riconosce: BACKGND = DARKBLUE3
- un teorema è: UNA FORMULA DEDUCIBILE SOLO USANDO GLI ASSIOMI
- Quali di queste applicazioni sono orientate di controllo?: INTERFACCE GRAFICHE
- Una classe astratta: È PARZIALMENTE IMPLEMENTATA
- Il debugging è la ricerca e rimozione: DI DIFETTI
- Il significato delle formule è dato: DALLA SEMANTICA DEL LINGUAGGIO
- Il test di integrazione serve a collaudare: L’INTERFACCIAVIMENTO FRA I MODULI
- Quale di queste ER riconosce la stringa duudu?: D((U|D)U)*
- La formula A->(B->A) è: VALIDA
- Il piano di test di sistema viene prodotto: NELLA FASE DI ANALISI E SPECIFICA
- La marcatura di una rete di Petri: ASSOCIA UN INTERO NON NEGATIVO AD OGNI PLACE
- La formula A->notA è: SODDISFACENTE
- In un sistema formale corretto: TUTTE LE FORMULE DIMOSTRABILI SONO VERE
- In un Sistema formale completo: TUTTE LE FORMULE VALIDE SONO DIMOSTRABILI
- Nelle reti di Petri l’abilitazione di una transizione dipende: DALLA MARCATURA DEI PLACE DI INGRESSO

- Gli strumenti CASE servono: A DEFINIRE DEI MODELLI
- I design pattern sono: DEGLI SCHEMI DI SOLUZIONI DEI PROBLEMI RICORRENTI
- il beta test è: UN COLLAUDO FATTO DA UTENTI SELEZIONATI
- Nelle reti di Petri si può avere un conflitto se: DUE TRANSIZIONI DEI PLACE DI INGRESSO COMUNE
- Le grammatiche non contestuali sono formalismi: ORIENTATI AI DATI
- Un modulo fisico è costituito da: UNO O PIÙ FILE
- Un'operazione protetta: VIENE USATA SOLO NELLA SUA CLASSE E CLASSI DERIVATE
- Il criterio di copertura delle condizioni si usa: NEL TEST STRUTTURALE
- Quale di queste stringhe viene riconosciuto dall'ER '-?[0-9]+?': 123
- Un oggetto è: UN'ASTRAZIONE DI UN'ENTITÀ INDIVIDUALE, CARATTERIZZATA DA IDENTITÀ, ATTRIBUTI E OPERAZIONI
- Quale di queste ER riconosce gli identificatori C++: [-A-Za-z][_A-Za-z0-9]*
- In fase di analisi dei requisiti si possono usare: I DIAGRAMMI DELLE CLASSI E DEI CASI D'USO
- Nascondere l'implementazione di un modulo: SERVE A RIDURRE LE DIPENDENZE FRA MODULI
- un Sistema si dice robusto se: SI COMPORTA ACCETTABILMENTE IN CONDIZIONI NON PREVISTE
- Un modello di processo è: UNA GENERALIZZAZIONE DI UNA FAMIGLIA DI PROCESSI DI SVILUPPO
- I requisiti di funzione: DESCRIVONO COSA DEVE FARE IL SISTEMA
- Le espressioni Regolari: SONO UN FORMALISMO DI SPECIFICA DEI DATI DI TIPO SINTATTICO
- Negli Automi a Stati finite le uscite: DIPENDONO DALLO STATO E DALL'INGRESSO
- Il test di unità: AVVIENE DI SOLITO NELLA FASE DI CODIFICA
- che cosa è una fase?: UN PERIODO IN CUI SI SVOLGE UN'ATTIVITÀ

-Qual'è differenza fra un libreria e un framework?

UNA LIBRERIA OFFRE MODULI SEMPLICI DA COMPORRE PER OTTENERE FUNZIONI PIÙ COMPLESSE, MENTRE UN FRAMEWORK OFFRE DEI COMPONENTI COMPLESSI INTERAGENTI SECONDO MECCANISMI PREDEFINITI CHE VENGONO SPECIALIZZATI PER ADATTARLI AGLI APPLICAZIONI SPECIFICHE.

-Qual'è la differenza fra anomalia e un guasto?

UNA ANOMALIA È UN DIFETTO DEL CODICE SORGENTE, UN GUASTO È UN COMPORTAMENTO NON CORRETTO RISPETTO ALLE SPECIFICHE.

-Che cos'è una partizione, nel progetto di Sistema?

UN SOTTOSISTEMA CHE REALIZZA UNA FUNZIONE DEL SISTEMA COMPLESSIVO.

-Quali diagrammi UML descrivono l'architettura fisica?

I DIAGRAMMI DI DEPLOYMENT E I DIAGRAMMI DEI COMPONENTI.

-Che cos'è il servizio naming?

UN SERVIZIO CORBA CHE PERMETTE DI OTTENERE UN RIFERIMENTO A UN OGGETTO REMOTO IDENTIFICATO DA UN NOME SIMBOLICO.

-Se t e u sono transizioni, che cosa significa ' $p \in \bullet t \cap \bullet u$ '?

IL POSTO P APPARTIENE SIA AL PREINSIEME DI T CHE A QUELLO DI U, OVVERO È IN INGRESSO A TUTTE E DUE LE TRANSIZIONI.

-Cos'è un'architettura a strati?

UN'ARCHITETTURA IN CUI IL SISTEMA È COMPOSTO IN SOTTOINSIEMI ORDINATI IN LIVELLI IN MODO TALE CHE OGNI STRATO OFFRA SERVIZI AI SOTTOINSIEMI DI LIVELLO SUPERIORE IMPLEMENTANDOLI PER MEZZO DEI SERVIZI OFFERTI DAI SOTTOSISTEMI A LIVELLO INFERIORE.

-Che cosa sono gli oggetti transitory nelle architetture CORBA?

OGGETTI CHE ESISTONO SOLO FINCHÈ ESISTE IL POA CHE LI HA CREATI.

-Che cosa sono i modelli di processo evolutivi?

MODELLI DI PROCESSI IN CUI IL SOFTWARE VIENE SVILUPPATO INCREMENTALMENTE IN PASSI SUCCESSIVI, OGNUNO DEI QUALI PRODUCE UNA PARTE O UNA VERSIONE DEL SISTEMA.

-Che cosa significa visibilità protetta?

UN'ENTITÀ CON VISIBILITÀ PROTETTA È ACCESSIBILE SOLO DALLE CLASSI DERIVATE.

-A cosa servono i modelli generic?

A RAPPRESENTARE LA STRUTTURA COMUNE DI ALGORITMI E CLASSI, METTENDONE IN EVIDENZA LE PARTI VARIABILI PER MEZZO DI PARAMETRI.

-A cosa serve il servizio eventi nelle architetture CORBA?

AD OTTENERE UN ACCOPPIAMENTO MENO STRETTO TRA I COMPONENTI DISTRIBUITI, PERMETTENDO LO SCAMBIO DI EVENTI ASINCRONI.

-Negli statechart, che cos'è una guardia?

UNA CONDIZIONE CHE DEVE ESSERE VERA AFFINCHÈ UNA TRANSIZIONE POSSA AVVENIRE.

-In una rete Petri, che cos'è e cosa rappresenta una marcatura?

UNA FUNZIONE CHE ASSOCIA UN INTERO NON NEGATIVO A CIASCUN PLACCE, RAPPRESENTA LO STATO DEL SISTEMA MODELLATO DALLA RETE.

-Quali sono i meccanismi di estensione del linguaggio UML?

VINCOLI, STEREOTIPI, VALORI ETICHETTATI (O PROPRIETÀ)

-Che cos'è un oggetto attivo?

UN OGGETTO CAPACE DI ATTIVARE UN FLUSSO DI CONTROLLO.

-Che cos'è una specifica?

UNA DESCRIZIONE PRECISA DEI REQUISITI DI UN SISTEMA O DI UNA SUA PARTE.

-In un processo di sviluppo, che cos'è una fase?

UN INTERVALLO DI TEMPO IN CUI SI SVOLGE UN'ATTIVITÀ

-A cosa serve il framework DejaGNU?

AI TEST DI UNITÀ

-Che cos'è l'alpha test?

UN TEST DI SISTEMA CHE SI SVOLGE USANDO L'APPLICAZIONE ALL'INTERNO DELL'AZIENDA PRODUTTRICE.

-Cos'è un Sistema concorrente?

UN SISTEMA FORMATO DA PIÙ PROCESSI INTERCOMUNICANTI E SOGGETTI A VINCOLI DI SINCRONIZZAZIONE RECIPROCA

-Che cos'è una classe concreta?

UNA CLASSE ISTANZIABILE, PERCHÉ INTERAMENTE IMPLEMENTATA.

-Qual'è il modo più utile di usare l'eredità multipla?

DEFINIRE UN'INTERFACCIA COME COMPOSIZIONE DI ALTRE INTERFACCIE

-A cosa serve il servizio naming nelle architetture CORBA?

SERVE AD OTTENERE UN RIFERIMENTO A UN OGGETTO REMOTO IDENTIFICATO DA UN NOME SIMBOLICO.

-Negli statechart, quali informazioni si possono associare ad una transizione?

NOME ED ATTRIBUTI DELL'EVENTO, CONDIZIONE DI GUARDIA ED AZIONE.

-Nelle reti Petri, quando si dice che due transizioni sono in sequenza?

QUANDO, IN UNA MARCATURA M , UAN DELLE DUE (SIA U), È ABILITATA SOLO DOPO CHE È SCATTATA L'ALTRA (SIA T):

$$M[T] \wedge \neg(M[U]) \wedge M[TU]$$

-Quali sono gli elementi costitutivi di un Automata a Stati Finiti?

L'INSIEME DEGLI STATI (S), L'INSIEME DEGLI INGRESSI (I), L'INSIEME DELLE USCITE (U), LA FUNZIONE DI TRANSIZIONE ($d: S \times I \rightarrow S$), LA FUNZIONE DI USCITA ($t: S \times I \rightarrow U$), LO STATO INIZIALE $s_0 \in S$

-Quale è lo scopo della semantica di un linguaggio?

ASSOCIARE UN SIGNIFICATO ALLE FORMULE DEL LINGUAGGIO.

-Che cosa s'intende per "test strutturale"?

UN TEST IN CUI LA SELEZIONE DEI DATI DI TEST SI BASA SULLA CONOSCENZA DEL CODICE SORGENTE.

-Qual'è la principale differenza fra una classe e un entità?

LE ENTITÀ NON HANNO OPERAZIONI.

-Le transizioni acetate da un database sono costituite da un'operazione di inizio, seguita da zero o più operazioni di lettura o scrittura, seguite da un commit o un abort. Descrivere con un ER

$$S(R|W)^*(C|A)$$

-Che cosa sono i vincoli?

CONDIZIONI IMPOSTE AL SISTEMA SPECIFICATO.

-Che relazione c'è fra modularità di un progetto e l'organizzazione dell'attività di sviluppo?

LA MODALITÀ PERMETTE LA RIPARTIZIONE DEL LAVORO FRA DIVERSI GRUPPI DI SVILUPPO.

-Quali sono i principali requisiti dei sistemi distribuiti di grandi dimensioni?

SCALABILITÀ, RIUSABILITÀ, INDIPENDENZA DELLA PIATTAFORMA DI CALCOLO E INDIPENDENZA DAI LINGUAGGI DI PROGRAMMAZIONE.

-Come si rappresenta lo stato di un Sistema nelle reti di Petri?

PER MEZZO DELLA MARCATURA.

-Che cos'è un compilatore IDL?

UN PROGRAMMA CHE AVENDO IN INGRESSO DELLE DICHIARAZIONI DI INTERFACCE IDL PRODUCE LE DICHIARAZIONI NECESSARIE PER SCRIVERE IN DATO LINGUAGGIO LE PARTI SERVER E CLIENTE DELL'APPLICAZIONE SPECIFICATA DA TALI INTERFACE.

-Quali sono i tipi di manutenzione del SW?

CORRETTIVA, ADATTIVA E PERFETTIVA.

-A cosa serve il blocco catch in c++?

A GESTIRE LE ECCEZIONI SOLLEVATE NELL'ESECUZIONE DEL BLOCCO TRY CORRISPONDENTE.

-Nelle reti di Petri quando si dice che una transizione è viva?

PER OGNI MARCATURA M RAGGIUNGIBILE DA M_0 ESISTE UNA SEQUENZA DI SCATTI CHE A PARTIRE DA M ABILITA LA TRANSAZIONE.

-Definire il criterio di copertura delle decisioni

OGNI ARCO DEL GRAFO DI CONTROLLO DEVE ESSERE PERCORSO ALMENO UNA VOLTA.

-Che cos'è una classe?

LA DESCRIZIONE DI UN INSIEME DI OGGETTI CON STRUTTURA E COMPORTAMENTO SIMILI.

-che cos'è un prototipo:

UNA VERSIONE APROSSIMATA, PARZIALE, MA FUNZIONANTE DELL'APPLICAZIONE CHE VIENE SVILUPPATA

-Che cosa significa la formula $M_1[s] \rightarrow M_2[t]$?

SE LA TRANSIZIONE s È ABILITATA NELLA MARCATURA M_1 , ALLORA LA TRANSIZIONE t È ABILITATA NELLA MARCATURA M_2 .

-Qual'è la principale differenza tra statechart e Automi a Stati finiti?

GLI STATECHART PERMETTONO UNA DESCRIZIONE STRUTTURATA E GERARCHICA DEGLI STATI DI UN AUTOMA, CHE POSSONO ESSERE DECOMPOSTI IN SOTTOSTATI.

-Che cos'è lo studio di fattibilità?

UN'ATTIVITÀ VOLTA A STABILIRE SE UN PRODOTTO È TECNICAMENTE REALIZZABILE ED ECONOMICAMENTE CONVENIENTE A PROPORRE ALCUNE STRATEGIE PER LA SUA REALIZZAZIONE E A STIMARE I COSTI.

-A cosa servono i diagrammi di deployment?

SERVONO A DESCRIVERE L'ARCHITETTURA HARDWARE, MOSTRANDO I NODI COMPUTAZIONALI, LE LORO INTERCONNESSIONI, ED EVENTUALMENTE I COMPONENTI SOFTWARE ASSOCIAZI AI NODI.

-Che cosa sono i modelli di processo evolutivo?

MODELLO DI PROCESSI IN CUI IL SOFTWARE VIENE SVILUPPATO INCREMENTALMENTE IN PASSI SUCCESSIVI, OGNUNO DEI QUALI PRODUCE UNA PARTE O UNA VERSIONE DEL SISTEMA.

-Quali tipi di eventi sono previsti in UML?

CHIAMATE DI OPERAZIONE, CAMBIAMENTI, SEGNALI, EVENTI TEMPORALI

-Con quali diagrammi UML si descrivono le interazioni fra oggetti?

DIAGRAMMI DI SEQUENZA E DI COLLABORAZIONE

-A cosa servono i classificatory <<interface>>?

UN <<INTERFACE>> PERMETTE DI RAPPRESENTARE UN'INTERFACCIA INDIPENDENTEMENTE DALLE ENTITÀ CHE LA IMPLEMENTANO.

-Che cosa sono le sequenze (sequence<>) nel linguaggio UML?

ARRAY UNIDIMENSIONALI CON UN NUMERO VARIABILE DI ELEMENTI.

-Che cosa sono i requisiti non funzionali?

CARATTERISTICHE DI QUALITÀ O VINCOLI.

-Che cos'è la sicurezza?

LA CAPACITÀ DI FUNZIONARE SENZA ARRECARE DANNI A PERSONE O COSE.

-Nelle architetture CORBA, a cosa servono le policy?

A CONFIGURARE IL POA.

-Quali sono i cinque elementi costitutivi di una rete di Petri?

INSIEME DEI POSTI, INSIEME DELLE TRANSIZIONI, RELAZIONE DI FLUSSO PESI E MARCATURA INIZIALE.

-Quando si dice che un Sistema formale è completo

QUANDO TUTTE LE FORMULE VALIDE SONO DEMONSTRABILI

-Che cos'è un prototipo usa e getta?

UN PROTOTIPO CON UNA STRUTTURA MOLTO DIVERSA DA QUELLA DEL PRODOTTO FINALE, QUINDI DA SCARTARE DOPO AVERLO USATO PER INFORMAZIONI UTILI PER LO SVILUPPO.

-Nelle reti di petri cos'è la regola dello scatto?

LA LEGGE CHE STABILISCE COME SI EVOLVE LA MARCATURA DI UNA RETE IN SEGUITO ALLO SCATTO DI UNA TRANSIZIONE.

-Che cos'è una transizione di completamento?

UNA TRANSIZIONE NON ETICHETTATA DA UN EVENTO, CHE AVVIENE AL TERMINE DELL'ATTIVITÀ ASSOCIATA ALLO STATO DI PARTENZA.

-Quali componenti comprende la specifica CORBA?

MODELLO COMPUTAZIONALE, INFRASTRUTTURA DI COMUNICAZIONE, LINGUAGGIO PER LA DESCRIZIONE INTERFACE, LIBRERIE DI INTERFACCE STANDARDIZZATE.

-Quali sono le funzioni del Portable Object Adapter?

CREARE, ATTIVARE, DISATTIVARE E DISTRUGGERE GLI OGGETTI CORBA, DI REGISTRARE SERVENTI PER GLI OGGETTI E DI SMISTARE FRA I SERVENTI LE RICHIESTE FATTE AGLI OGGETTI.

-In quale fase del processo di sviluppo si fa il test di unità?

NEL CORSO DELLA FASE DI CODIFICA.

-Che cos'è un'eccezione?

UNA CONDIZIONE ANOMALA CHE SI PUÒ VERIFICARE NELL'USO DI UN MODULO.

-Che cosa è la relazione di flusso nelle reti di Petri?

LA RELAZIONE CHE ASSOCIA POSTI A TRANSIZIONI E TRANSIZIONI A POSTI:

$$F \subseteq (P \times T) \cup (T \times P)$$

-Che cosa è un'azione, nel formalismo degli statechart?

UN COMPORTAMENTO NON INTERROMPIBILE.

-Tabellle proposizionali:

A	B	$\neg(A \wedge B)$	$\neg A \vee \neg B$
T	T	F	$F \vee F = F$
T	F	T	$F \vee T = T$
F	T	T	$T \vee F = T$
F	F	T	$T \vee T = T$

A	B	$A \wedge \neg B$	$\neg(A \Rightarrow B)$
T	T	F	$\neg(T \Rightarrow T) = F$
T	F	T	$\neg(T \Rightarrow F) = T$
F	T	F	$\neg(F \Rightarrow T) = F$
F	F	F	$\neg(F \Rightarrow F) = F$

A	B	$A \wedge B$	$(A \wedge B) \Rightarrow B$
T	T	T	$T \Rightarrow T = T$
T	F	F	$F \Rightarrow T = T$
F	T	F	$F \Rightarrow T = T$
F	F	F	$F \Rightarrow F = T$

x	y	$x \Rightarrow y$
T	T	T
T	F	F
F	T	T
F	F	T

A	B	$A \vee B$	$(A \vee B) \Rightarrow A$
T	T	T	$T \Rightarrow T = T$
T	F	T	$T \Rightarrow T = T$
F	T	T	$T \Rightarrow F = F$
F	F	F	$F \Rightarrow F = T$

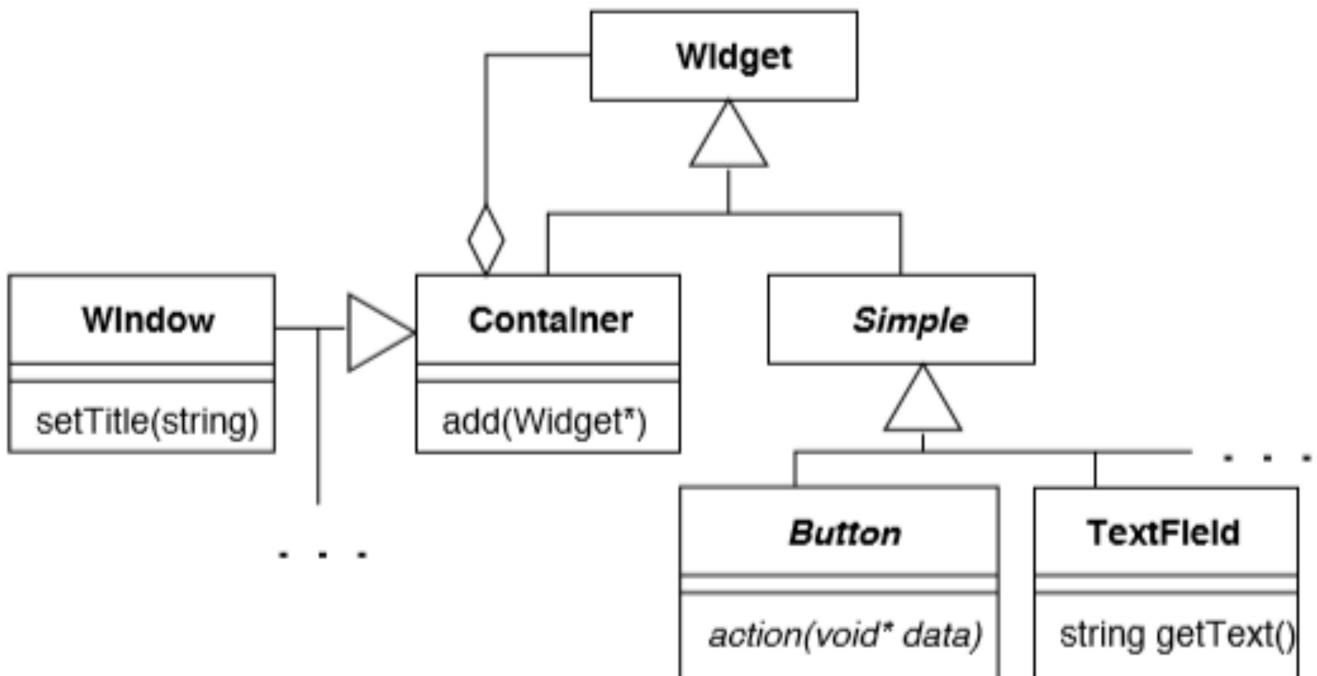
A	B	$A \vee B$	$(A \vee B) \Rightarrow B$
T	T	T	$T \Rightarrow T = T$
T	F	F	$F \Rightarrow T = T$
F	T	T	$T \Rightarrow T = T$
F	F	T	$F \Rightarrow T = T$

A	B	$A \Rightarrow B$	$A \wedge (A \Rightarrow B)$	$(A \wedge (A \Rightarrow B)) \Rightarrow B$
T	T	T	T	$T \Rightarrow T = T$
T	F	F	F	$F \Rightarrow T = T$
F	T	T	F	$T \Rightarrow T = T$
F	F	T	F	$F \Rightarrow T = T$

A	B	$A \Leftrightarrow B$
T	T	T
T	F	F
F	T	F
F	F	T

A	B	$A \wedge \neg B$	$\neg(A \Rightarrow B)$
T	T	F	$\neg(T \Rightarrow T) = F$
T	F	T	$\neg(T \Rightarrow F) = T$
F	T	F	$\neg(F \Rightarrow T) = F$
F	F	F	$\neg(F \Rightarrow F) = F$

A	B	C	$A \wedge B$	$(A \wedge B) \oplus C$
T	T	F	T	T
T	F	F	F	F
F	T	F	F	F
F	F	F	F	F



Un widget può essere di tipo Container o Simple. I widget di tipo Container possono contenere altri widget, ed hanno l'operazione add() per aggiungere un widget al contenitore. Quest'operazione prende come argomento un puntatore al widget che viene aggiunto. Un widget di tipo conteiner è Window, che ha l'operazione setTitle(), per impostare il titolo della finestra. Quest'operazione prende come argomento una stringa di caratteri. Due widget di tipo Simple sono button e TextField, con le operazioni descritte nel testo. L'operazione action() non è implementata: l'implementazione viene fornita dalla classe derivata e prende come argomento un puntatore a una struttura dati di tipo generico.

Scrivi un programma c++ che crea una finestra col titolo “Main Window”, ci inserisce un bottone e un campo di testo, scrive sull’uscita standard il contenuto del campo testo, quando viene premuto il bottone, includere il file widgets.h

```

#include <iostream>
#include "widgets.h"

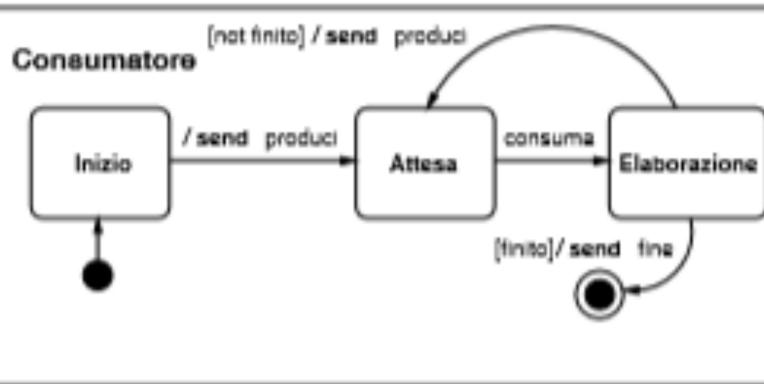
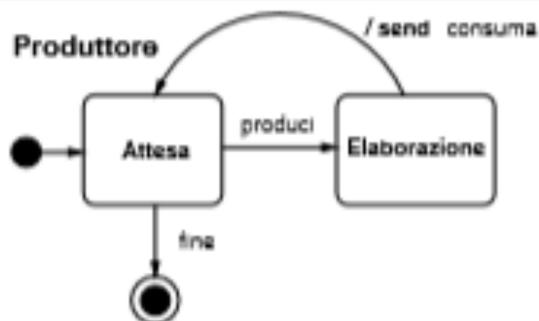
using namespace std;

class MyButton : public Button {
    TextField* tf;
public:
    MyButton(TextField* t) : tf(t) {}
    void action(void* data);
};

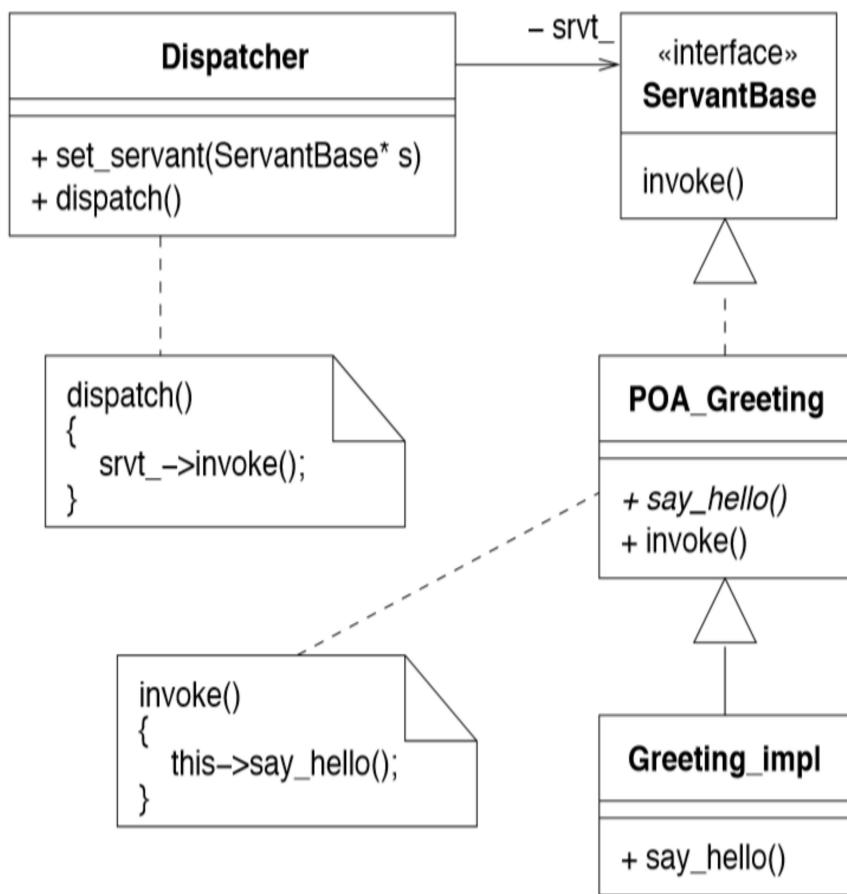
void
MyButton::action(void* data)
{
    cout << tf->getText() << endl;
}

int
main()
{
    Window* w = new Window;
    TextField* t = new TextField;
    MyButton* b = new MyButton;

    w->setTitle("Main Window");
    w->add(t);
    w->add(b);
}
  
```



- il **Produttore** inizia l'interazione
- il **Produttore** fa terminare l'interazione
- il **Produttore** invia il segnale **consumma**
- il **Consumatore** entra in **Elaborazione** prima del **Produttore**
- il **Consumatore** entra in **Attesa** quando riceve **producì**



```

class ServantBase {
public:
    virtual void invoke() = 0;
};

class Dispatcher {
    ServantBase* srvt_;
public:
    void set_servant(ServantBase* s);
    void dispatch();
};

class POA_Greeting : public ServantBase {
public:
    virtual void say_hello() = 0;
    virtual void invoke();
};

class Greeting_Impl : public POA_Greeting {
    virtual void say_hello();
};

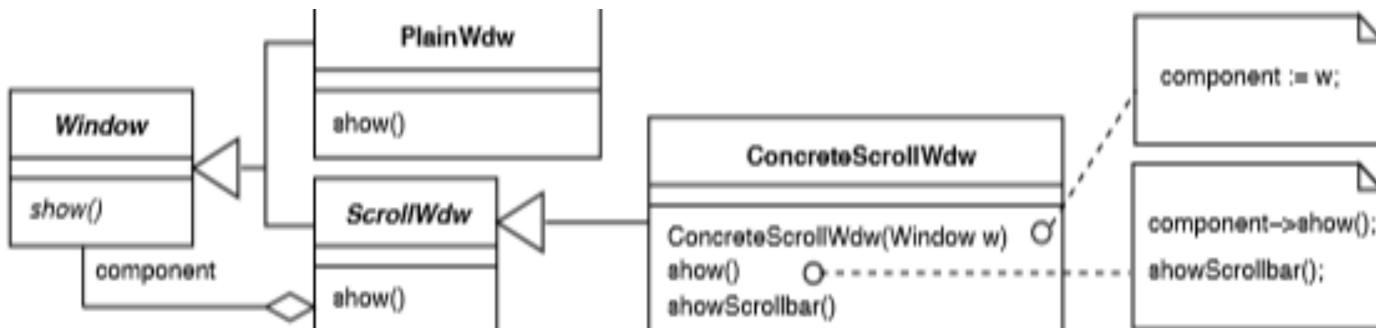
void Dispatcher::set_servant(ServantBase* s)
{
    srvt_ = s;
}

void Dispatcher::dispatch()
{
    srvt_->invoke();
}

void POA_Greeting::invoke()
{
    this->say_hello();
}

void Greeting_Impl::say_hello()
{
    cout << "Hello!" << endl;
}

```



un **ConcreteScrollWdw** contiene un **Window**

V

un **ConcreteScrollWdw** è un **PlainWdw**

ConcreteScrollWdw implementa **PlainWdw**

ConcreteScrollWdw estende il comportamento di **PlainWdw**

un **Window** contiene un **PlainWdw**

Dato il codice disegnare un diagramma delle classi e scrivere un programma che stampi sull'uscita standard il risultato della classe *Proxy_Greeting*, supponendo che ci si connetta al port 1492 della macchina `issw.unipi.it`

```

class Connection {
    string host;
    int port;
    string buffer;
public:
    Connection(string h, int p)
        : host(h), port(p) {}
    void send(string s);
    string receive();
};

class Greeting {
public:
    virtual string
        say_hello() = 0;
    virtual string
        sag_gutenTag() = 0;
};

```

```

class Proxy_Greeting : public Greeting {
    Connection* conn;
public:
    Proxy_Greeting(Connection* c)
        : conn(c) {}
    string say_hello();
    string sag_gutenTag();
};

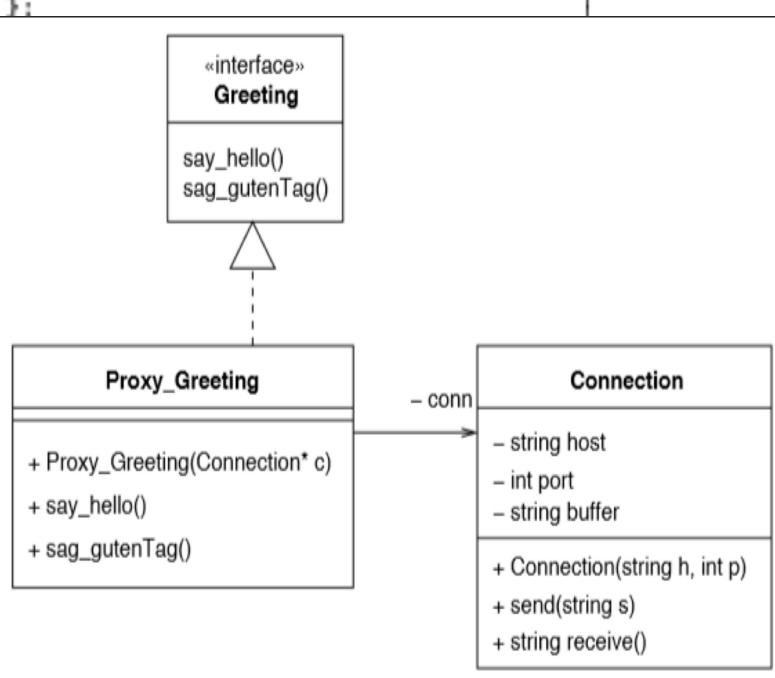
string
Proxy_Greeting::
say_hello()
{
    conn->send("say_hello");
    return conn->receive();
}

```

```

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

```

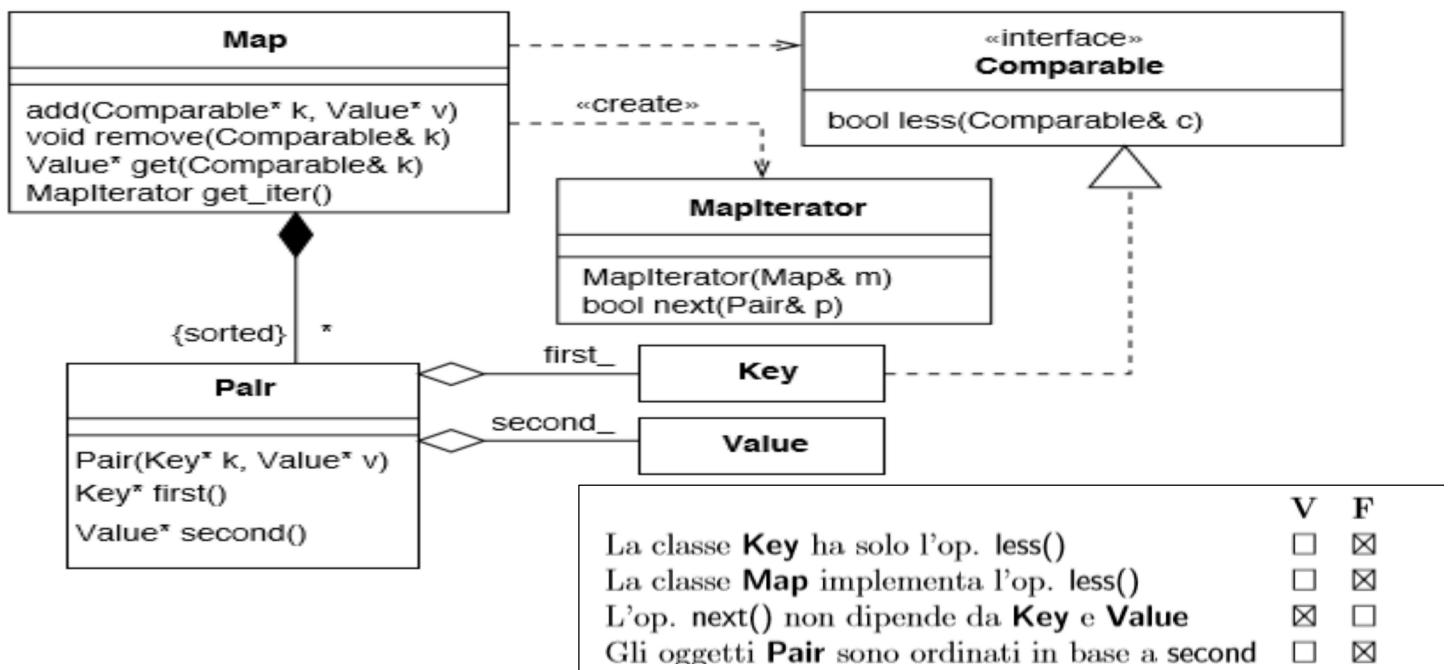


```

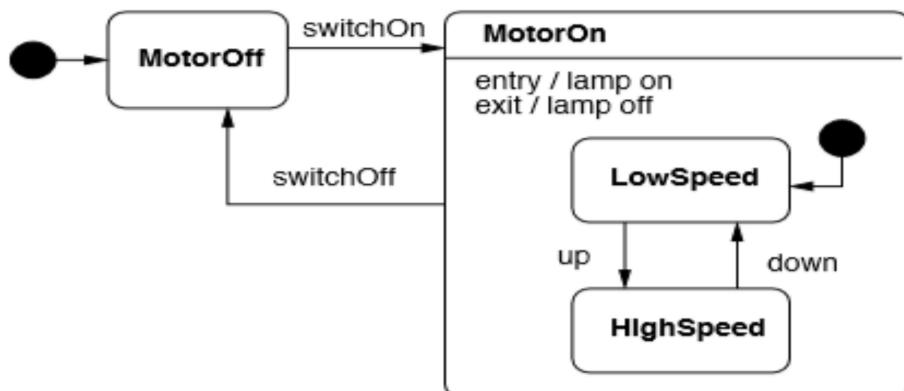
int
main()
{
    Connection c("issw.unipi.it", 1492);
    Proxy_Greeting g(&c);
    cout << g.say_hello() << endl;
    cout << g.sag_gutenTag() << endl;
    return (0);
}

```

Rispondi alle domande con vero o falso relative alla figura



Rispondi alle domande con vero o falso relative alla figura



Lo stato iniziale del sistema è **LowSpeed**.

V F

L'evento `switchOn` causa sempre l'accensione del motore.

Il motore si può spengere (`switchOff`) solo nello stato **LowSpeed**.

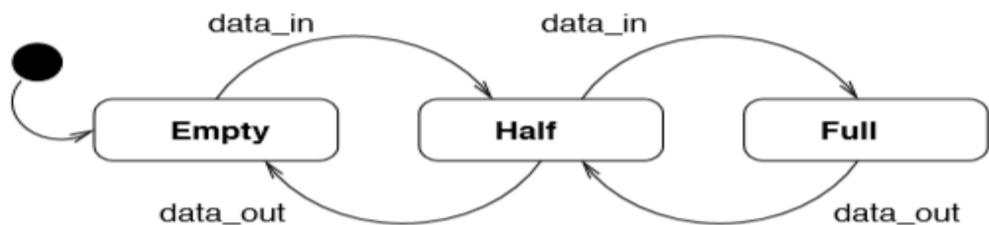
Durante il funzionamento c'è sempre una luce accesa.

Il sistema non ha uno stato finale.

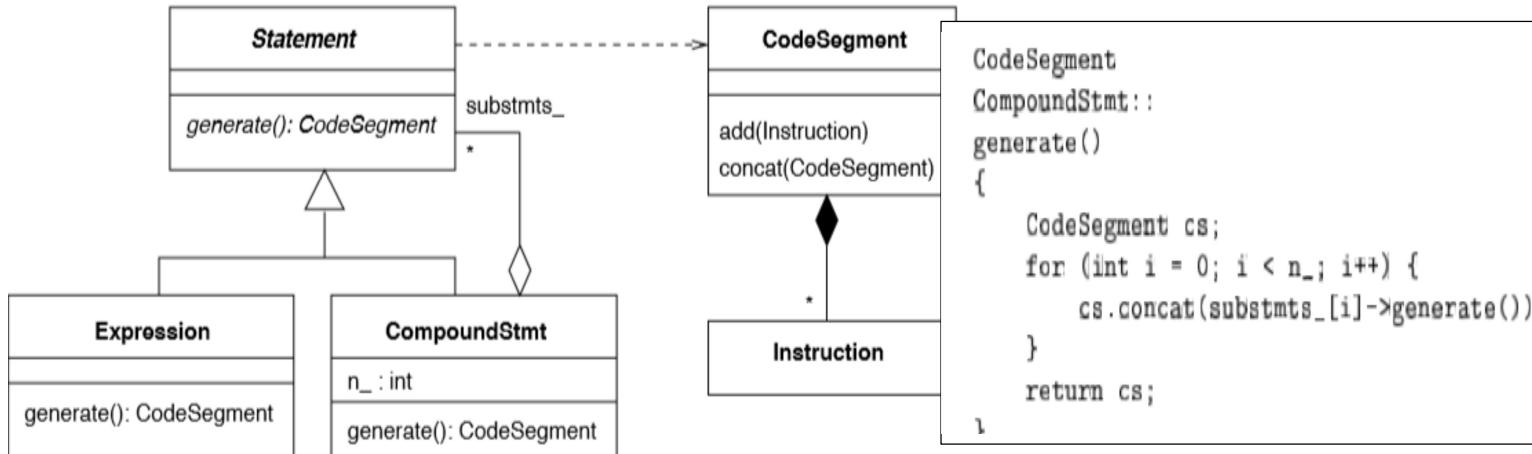
	Empty	Half	Full
Empty	<code>data_out</code>	<code>data_in</code>	<code>data_in</code>
Half		<code>data_out</code>	
Full			

Tabella 1: Rappresentazione tabulare

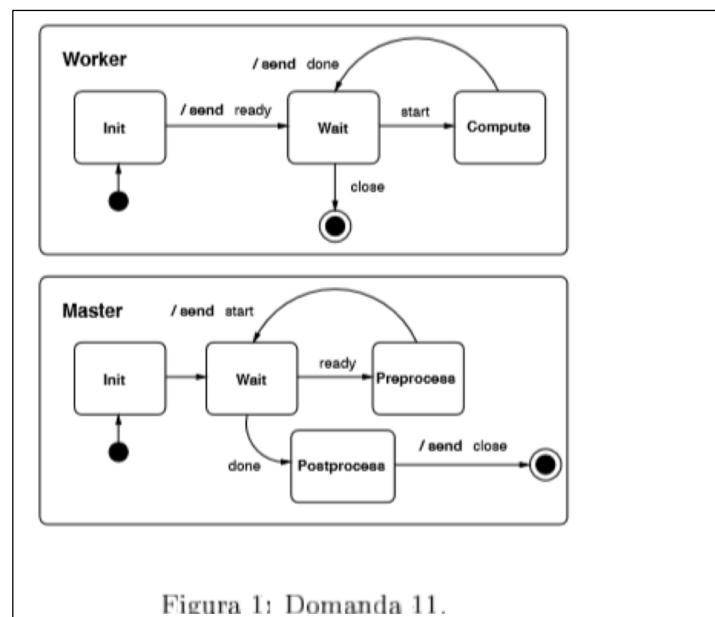
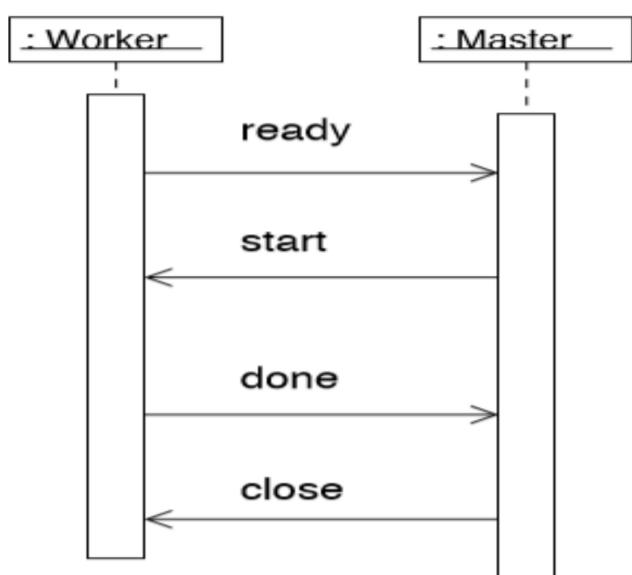
11 Disegnare l'ASF definito dalla Tab. 1.



Con riferimento alla figura 1 implementare l'operazione CompoundStmt::generate() supponendo che substmts_ sia un vettore di puntatori a Statement di dimensione n_. l'operazione concat() aggiunge il proprio argomento a un CodeSegment

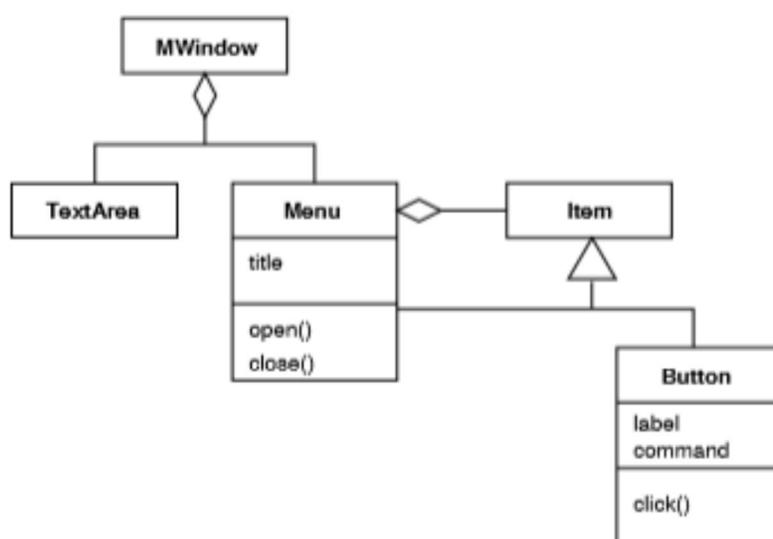


Disegna il diagramma di sequenza che mostra l'interazione fra due istanze delle classi Worker e Master il cui comportamento è specificato dallo Statechart

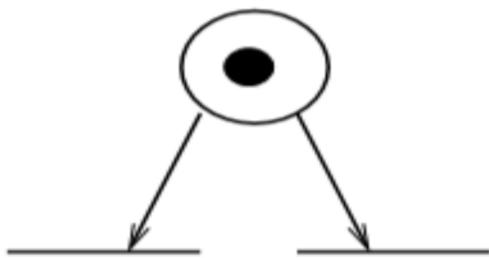


Descrivere in UML la seguente struttura:

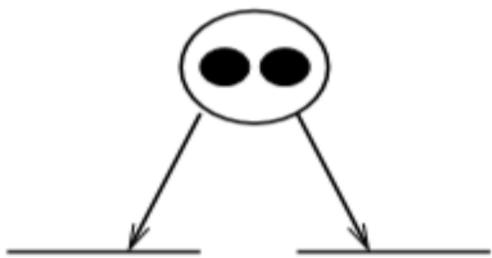
La finestra principale di un'applicazione contiene due menu ed un'area testo. Un menu ha un titolo e zero o più voci. Una voce può essere un bottone o un menu. Un menu si può aprire e chiudere. Un bottone ha un'etichetta e un comando. Un bottone si può "cliccare".



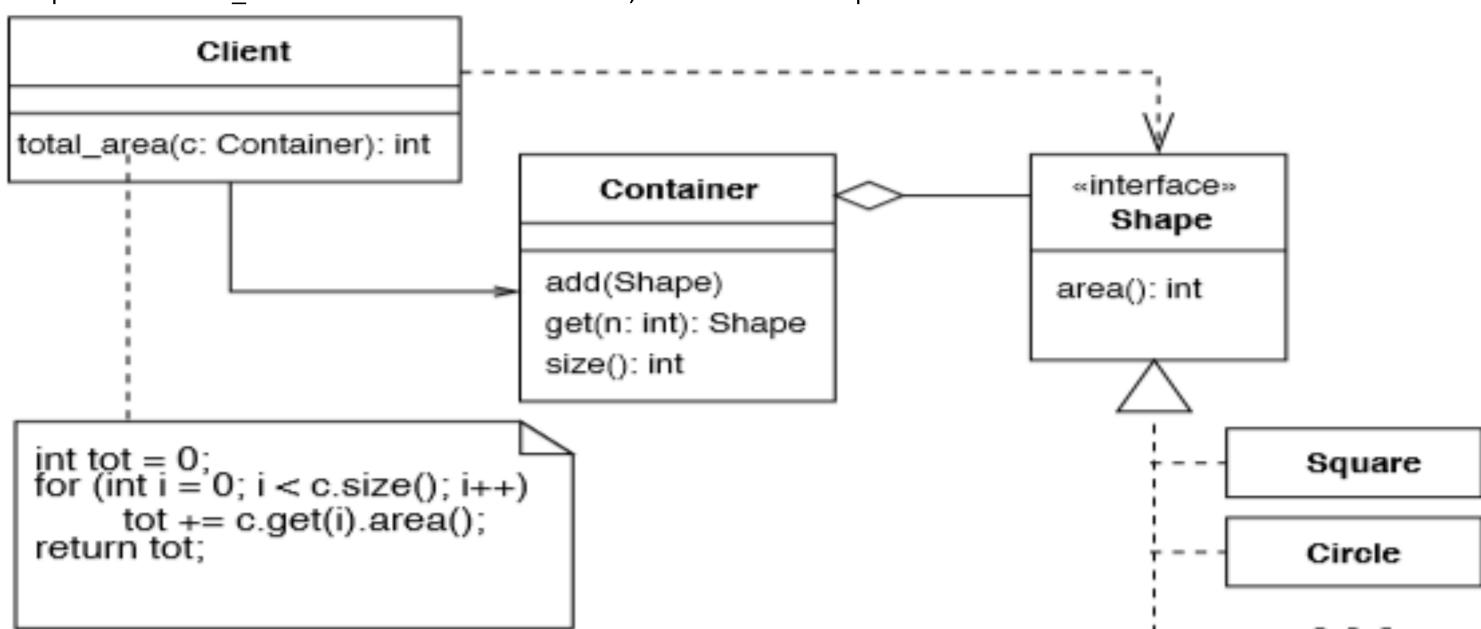
Conflitto effettivo



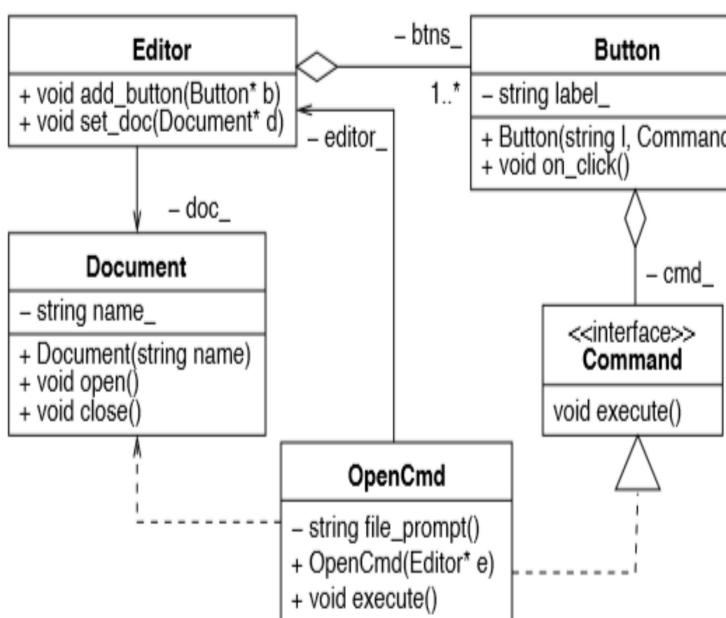
concorrenza effettiva



Disegnare un diagramma di classi relativo al seguente problema: Si hanno delle classi **Square**, **Circle** etc. che rappresentano figure geometriche. Ognuna di esse ha un'operazione **area():int** che restituisce l'area della figura. Si definisca l'interfaccia di una classe **Container** che contenga un insieme di figure qualsiasi e che permetta di aggiungere una figura, ottenere un riferimento (o un puntatore) a una figura individuata da un indice e ottenere il numero di figure contenute. Si definisca una classe **Client** con un'operazione **total_area** che restituisce l'area totale, mostrandone l'implementazione.



Scrivere le dichiarazioni in c++ corrispondenti al seguente UML



```

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);
};

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();
};
  
```

Con riferimento all' esercizio precedente, implementare **Button::onclik()**, 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 apre un documento.

```

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

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

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();
}

```

Disegnare l'UML del seguente codice

```

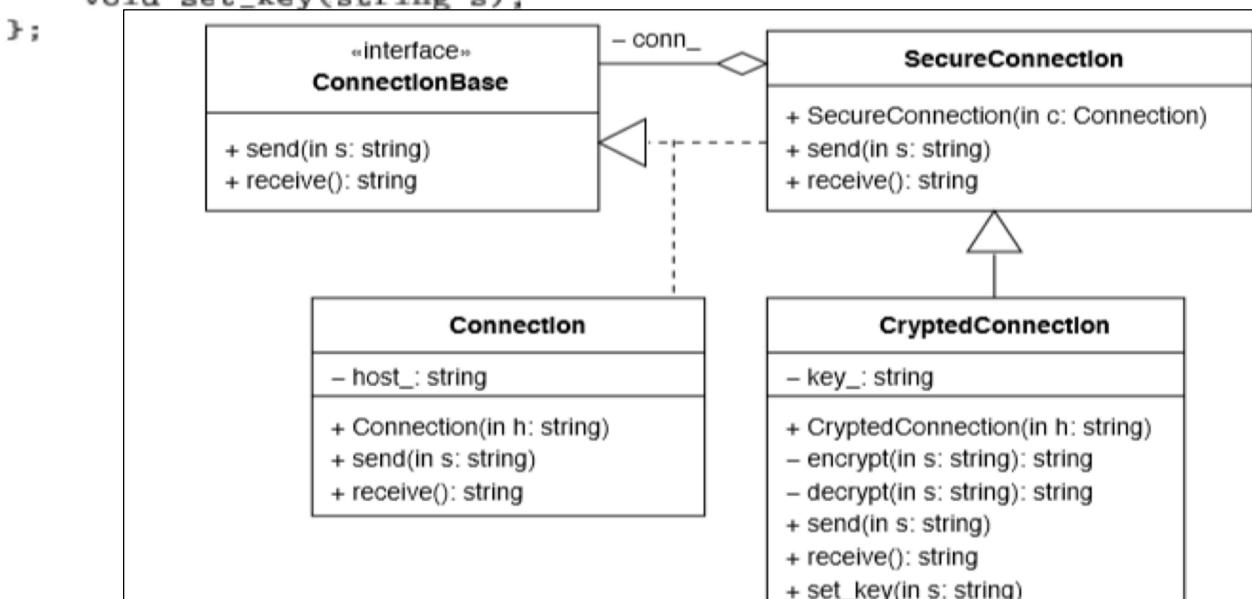
class ConnectionBase {
public:
    virtual void send(string s) = 0;
    virtual string receive() = 0;
};

class Connection : public ConnectionBase {
    string host_;
public:
    Connection(string h) : host_(h) {};
    void send(string s);
    string receive();
};

class SecureConnection : public ConnectionBase {
    ConnectionBase* conn_;
public:
    SecureConnection(Connection* c) : conn_(c) {};
    void send(string s);
    string receive();
};

class CryptedConnection : public SecureConnection {
    string key_;
    string encrypt(string s);
    string decrypt(string s);
public:
    CryptedConnection(Connection* c) : SecureConnection(c) {};
    void send(string s);
    string receive();
    void set_key(string s);
};

```



CryptedException::send(), receive, ssend, ConnectionBase

12 Con riferimento all'esercizio precedente:

implementare CryptedException::send(), che deve cifrare una stringa e inviarla;
 implementare CryptedException::receive(), che deve ricevere una stringa e decifrarla;
 scrivere una funzione (globale) ssend(ConnectionBase* c, string s) che deve inviare un messaggio su una connessione;
 supponendo già implementate tutte le altre operazioni, scrivere un programma principale che mandi il messaggio "ciao" allo host `ing.unipi.it:13`, in chiaro, e il messaggio "zdravstvuj" allo host `kremvax.kgb.su:17`, in cifra, usando la `ssend()` in tutti e due i casi (usare una chiave a scelta).

```

void
CryptedException::
send(string s)
{
    SecureConnection::send(encrypt(s));
}

string
CryptedException::
receive()
{
    return decrypt(SecureConnection::receive());
}

void
ssend(ConnectionBase* c, string s)
{
    c->send(s);
}

int
main()
{
    Connection ing("ing.unipi.it:13");
    ssend(&ing, "ciao");
    Connection kremvax("kremvax.kgb.su:17");
    CryptedException sc(&kremvax);
    sc.set_key("abracadabra");
    ssend(&sc, "zdravstvuj");
}

```

Disegna l'uml del codice

```

class SvtBase {
public:
    virtual void invoke(string n) = 0;
};

class Adapter {
    vector<SvtBase*> svt_;
public:
    int add_svt(SvtBase* s);
    void upcall(int oid, string n);
};

class IGreeting {
public:
    virtual void hello() = 0;
    virtual void ciao() = 0;
};

class Greeting_skl : public IGreeting,
                     public SvtBase {
public:
    virtual void invoke(string n);
};

class Greeting_svt : public Greeting_skl {
    virtual void hello();
    virtual void ciao();
};

```

```

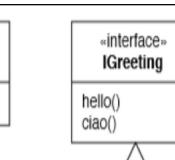
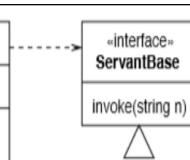
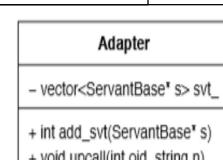
void Adapter::
upcall(int s, string n)
{
    svt_[s]->invoke(n);
}

int Adapter::
add_svt(SvtBase* s)
{
    svt_.push_back(s);
    return (svt_.size() - 1);
}

void Greeting_svt::
hello()
{
    cout << "Hello!" << endl;
}

void Greeting_svt::
ciao()
{
    cout << "Ciao!" << endl;
}

```



Adapter, upcall, add_svt(), invoke, Greeting_skl, upcall

Con riferimento all'esercizio precedente:

I parametri di Adapter::upcall() sono l'identificatore di un oggetto di tipo derivato da SvtBase e il nome di un'operazione che deve essere eseguita da tale oggetto.

L'operazione Adapter::add_svt() aggiunge un oggetto di tipo derivato da SvtBase e ne restituisce l'identificatore.

L'operazione Greeting_skl::invoke() riceve in ingresso il nome di un'operazione ed esegue l'operazione corrispondente, implementata da una classe derivata da IGreeting.

Implementare l'operazione Greeting_skl::invoke().

Scrivere un programma principale che stampi i messaggi "Ciao!" e "Hello!" usando Adapter::upcall().

```
void
Greeting_skl::
invoke(string n)
{
    if (n == "hello") {
        this->hello();
    } else if (n == "ciao") {
        this->ciao();
    }
}

int
main()
{
    Adapter a;
    Greeting_svt g;
    int svt = a.add_svt(&g);
    a.upcall(svt, "hello");
    a.upcall(svt, "ciao");
}
```

TransferIn vero falso

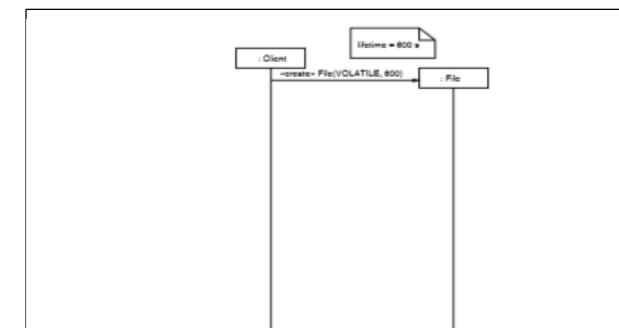
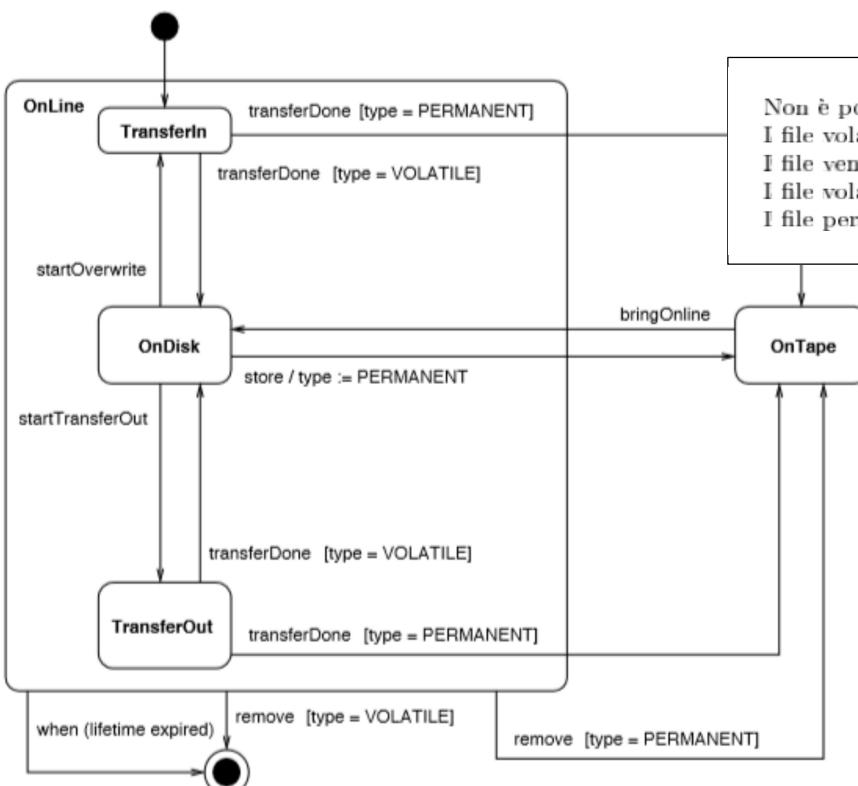


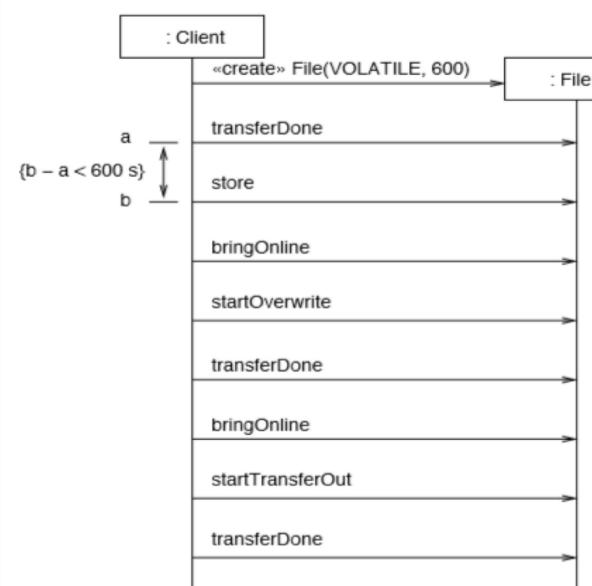
Figura 2: Domanda 12.

File

Completere il diagramma di sequenza di Fig. 2

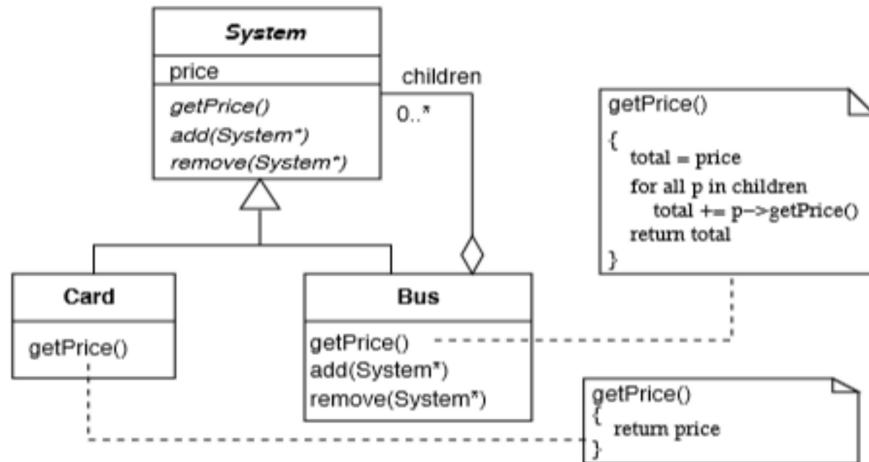
supponendo che il comportamento degli oggetti di tipo File sia specificato dal diagramma di Fig. 1 e mostrando le operazioni richieste per

- (1) mettere il file su nastro;
- (2) aggiornare il file con nuovi dati;
- (3) rileggere il file;
- (4) rimettere su nastro il file aggiornato.



Bus Card

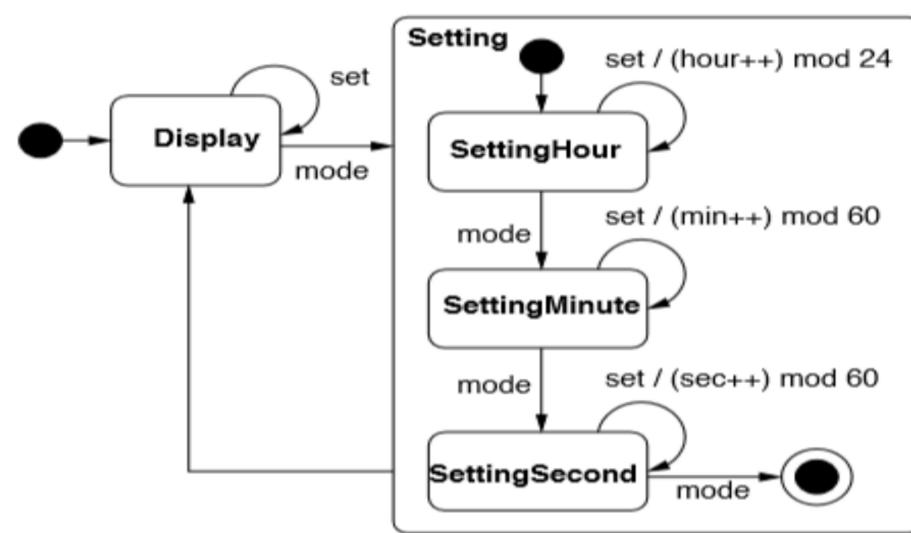
Disegnare un diagramma di classi che risolva il seguente problema:
Un sistema può essere un **Bus** o una **Card**. Ad un **Bus** si possono collegare zero o più **Bus** e zero o più **Card**. Ogni componente (bus o card) ha un prezzo. Vogliamo rappresentare la struttura di un sistema e calcolarne il prezzo complessivo. Applicare il design pattern Composite e indicare l'implementazione dell'operazione che calcola il prezzo.



SettingHour SettingMinute SettingSecond

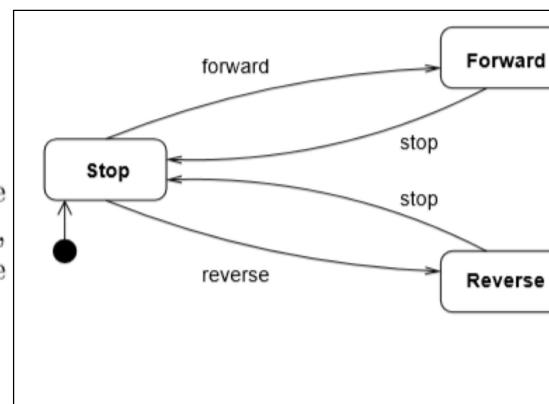
Disegnare uno Statechart che descriva il seguente sistema:

Un orologio ha due modi di funzionamento: **Display**, in cui mostra l'ora, e **Setting**, in cui si rimette l'ora. Questo modo di funzionamento comprende tre sottostati: **SettingHour**, **SettingMinute**, **SettingSecond**. L'orologio ha due tasti: **mode** e **set**. Il tasto **mode** serve a passare ciclicamente dallo stato iniziale **Display** ai tre sottostati **Setting** (nell'ordine detto). Il tasto **set** serve a incrementare di 1, ogni volta che viene premuto, il valore indicato nello stato corrente; nello stato **Display** non ha effetto. Le ore sono rappresentate da una variabile che va da 0 a 23, i minuti e i secondi da due variabili che vanno da 0 a 59.



Macchina a stati, stop, forward, reverse

Disegnare una macchina a stati che specifichi quanto segue: un motore può girare in due versi, ma non può passare direttamente da un verso all'altro, dovendo essere fermato prima di invertire il movimento. Il suo controllore accetta i segnali **stop**, **forward** e **reverse**.

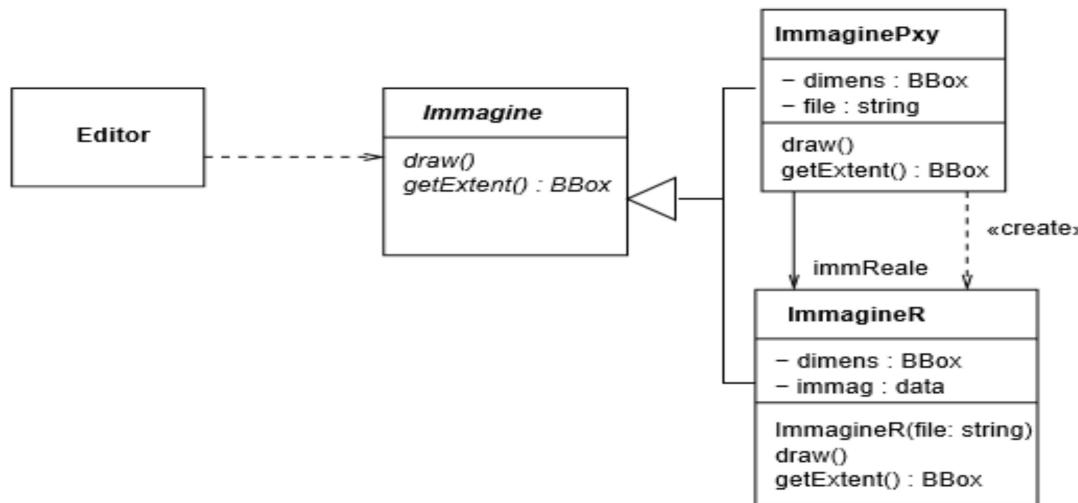


getExtent, draw BBox

Un Editor usa degli elementi Immagine

che offrono le operazioni `draw()`, che disegna l'immagine, e `getExtent()`, che restituisce una struttura BBox con le dimensioni dell'immagine. Un'immagine può essere reale, cioè rappresentata completamente, oppure essere un segnaposto contenente le dimensioni dell'immagine e il nome del file da cui caricare l'immagine. Inizialmente l'editor inserisce nel documento solo dei segnaposto, e crea le immagini reali solo quando devono essere mostrate. I costruttori delle classi usate per le immagini prendono come argomento il nome del file; per le immagini reali, il costruttore copia il file in un campo di tipo `data`.

Applicando il pattern *Proxy* (Fig. 3), disegnare un diagramma delle classi corrispondente a quanto descritto.



Con riferimento all'esercizio precedente:

scrivere in C++ le dichiarazioni delle classi;

implementare l'operazione `draw()` della classe usata per i segnaposto.

```
class Immagine {
public:
    virtual void draw() = 0;
    virtual BBox getExtent() = 0;
};

class ImmagineR : public Immagine {
    BBox dimens_;
    data immag_;
public:
    ImmagineR(string f);
    virtual void draw();
    virtual BBox getExtent();
};

class ImmaginePxy : public Immagine {
    BBox dimens_;
    string file_;
    ImmagineR* immReale_;
public:
    ImmaginePxy(string f);
    virtual void draw();
    virtual BBox getExtent();
};

void
ImmaginePxy::draw()
{
    if (immReale_ == 0)
        immReale_ = new ImmagineR(file_);
    immReale_->draw();
}
```