

- 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 ($P \rightarrow N$)
- 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 $[_A-Za-z][_A-Za-z0-9]^* = ('|'\backslash t')^*[=_A-Za-z0-9]+$ 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 $A \rightarrow (B \rightarrow A)$ 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 $A \rightarrow \neg A$ è: SODDISFACTANTE
- In un sistema formale corretto: TUTTE LE FORMULE DEMOSTRABILI SONO VERE
- In un Sistema formale completo: TUTTE LE FORMULE VALIDE SONO DEMOSTRABILI
- 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 ' $p \in ot \cap ou$ '?

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?

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

-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 ($\tau: S \times I \rightarrow U$), 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, seguite 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 INTERFACCIE 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 APPROXIMATA, 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 finite?

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 DIVERA 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:

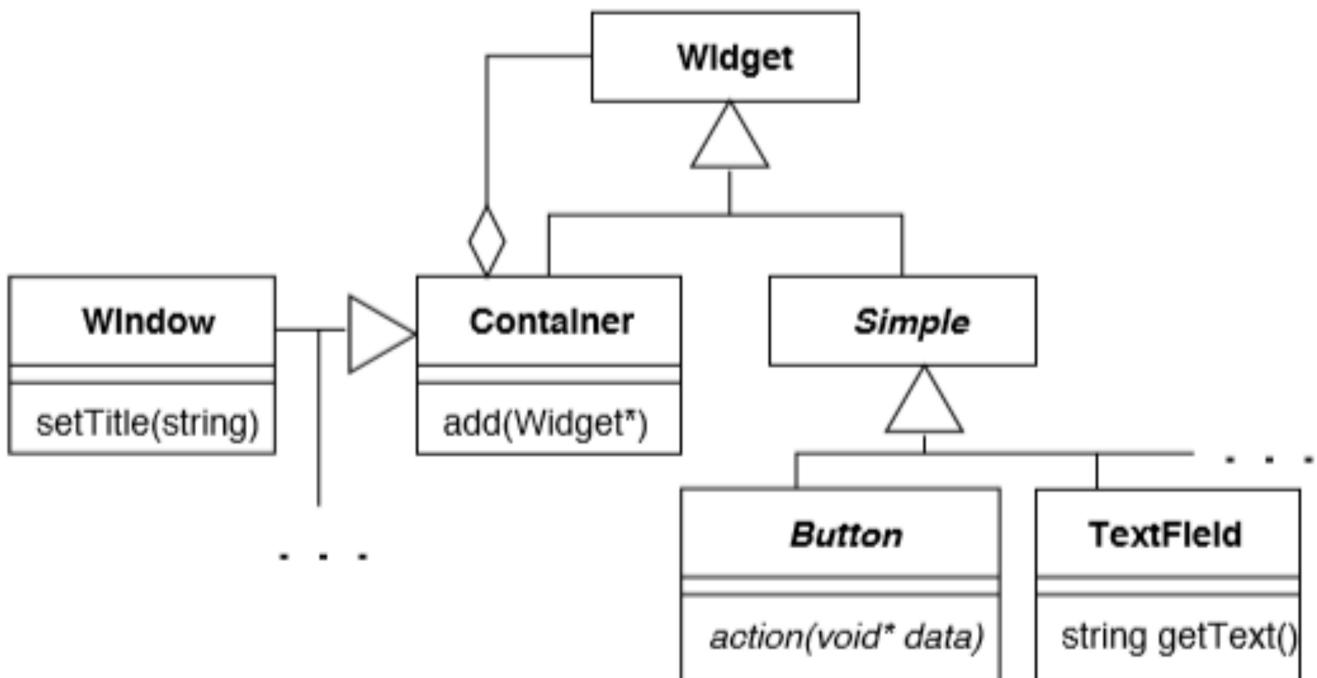
$$F \subseteq (P \times T) \cup (T \times P)$$

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

UN COMPORTAMENTO NON INTERROMPIBILE.

-Tabelle proposizionali:

A	B	$\neg(A \wedge B)$	$\neg A \vee \neg B$	A	B	$A \wedge \neg B$	$\neg(A \Rightarrow B)$	A	B	$A \wedge B$	$(A \wedge B) \Rightarrow B$	x	y	$x \Rightarrow y$	A	B	$A \vee B$	$(A \vee B) \Rightarrow A$		
T	T	F	$F \vee F = F$	T	T	F	$\neg(T \Rightarrow T) = F$	T	T	T	$T \Rightarrow T = T$	T	T	T	T	T	$T \Rightarrow T = T$			
T	F	T	$F \vee T = T$	T	F	T	$\neg(T \Rightarrow F) = T$	T	F	F	$F \Rightarrow F = T$	T	F	F	T	F	$T \Rightarrow T = T$			
F	T	T	$T \vee F = T$	F	T	F	$\neg(F \Rightarrow T) = F$	F	T	F	$F \Rightarrow T = T$	F	T	T	T	F	$T \Rightarrow F = F$			
F	F	T	$T \vee T = T$	F	F	F	$\neg(F \Rightarrow F) = F$	F	F	F	$F \Rightarrow F = T$	F	F	T	F	F	$F \Rightarrow F = T$			
A	B	$A \vee B$	$(A \vee B) \Rightarrow B$	A	B	$A \Rightarrow B$	$A \wedge (A \Rightarrow B)$	(A $\wedge (A \Rightarrow B)$) $\Rightarrow B$	A	B	$A \Leftrightarrow B$	A	B	$A \wedge \neg B$	$\neg(A \Rightarrow B)$	A	B	C	$A \wedge B$	$(A \wedge B) \oplus C$
T	T	T	T	T	T	T	T	T	T	T	F	T	T	F	$\neg(T \Rightarrow T) = F$	T	T	F	T	T
T	F	T	F	T	F	F	F	T	T	F	T	T	F	T	$\neg(T \Rightarrow F) = T$	T	F	F	T	F
F	T	T	T	F	T	F	F	T	F	F	T	F	T	F	$\neg(F \Rightarrow T) = F$	F	T	F	F	F
F	F	F	T	F	F	F	F	T	F	F	F	F	F	F	$\neg(F \Rightarrow F) = F$	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 seiTitle(), per impostare il titolo della finestra. Quest'operazioione 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"

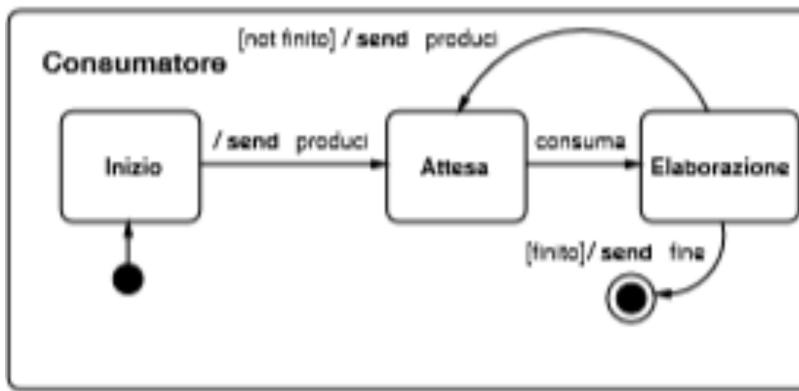
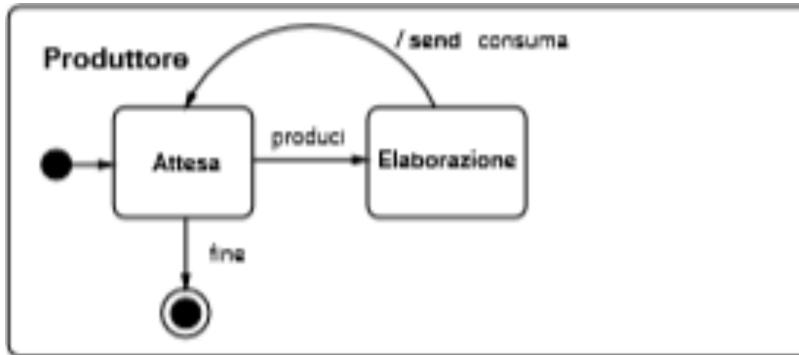
using namespace std;

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

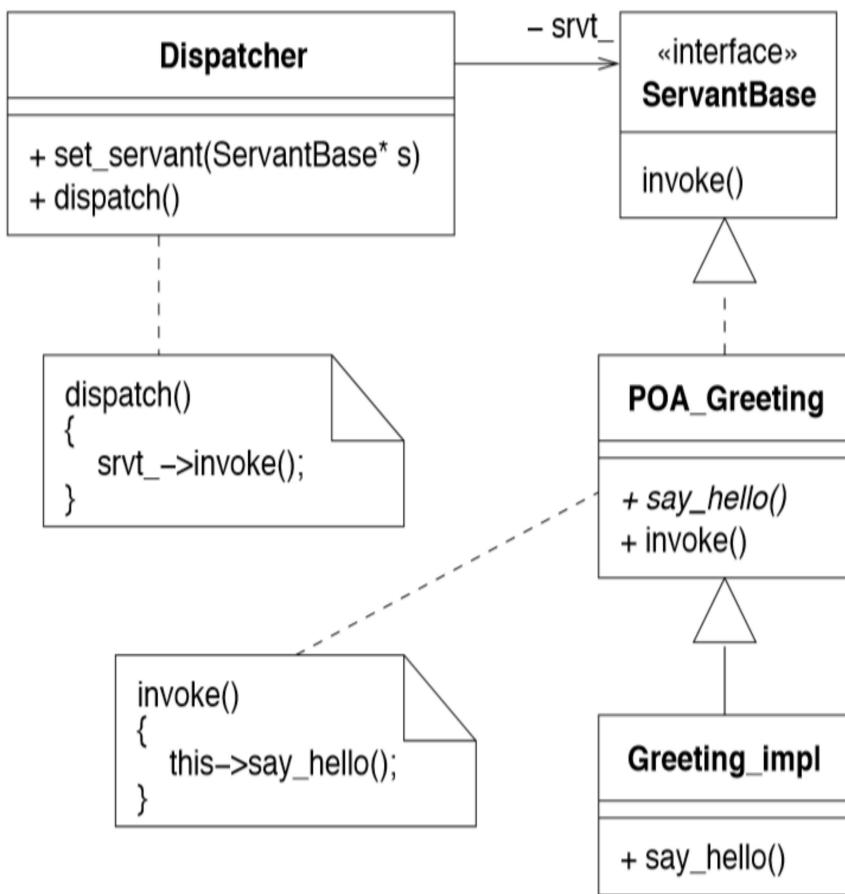
void
MyButton::action(void* data)
{
    cout << tf->getText() << endl;
}

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 | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| il Produttore fa terminare l'interazione | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| il Produttore invia il segnale consumma | <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| il Consumatore entra in Elaborazione prima del Produttore | <input type="checkbox"/> | <input checked="" type="checkbox"/> |
| il Consumatore entra in Attesa quando riceve produc | <input type="checkbox"/> | <input checked="" type="checkbox"/> |



```

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();
};

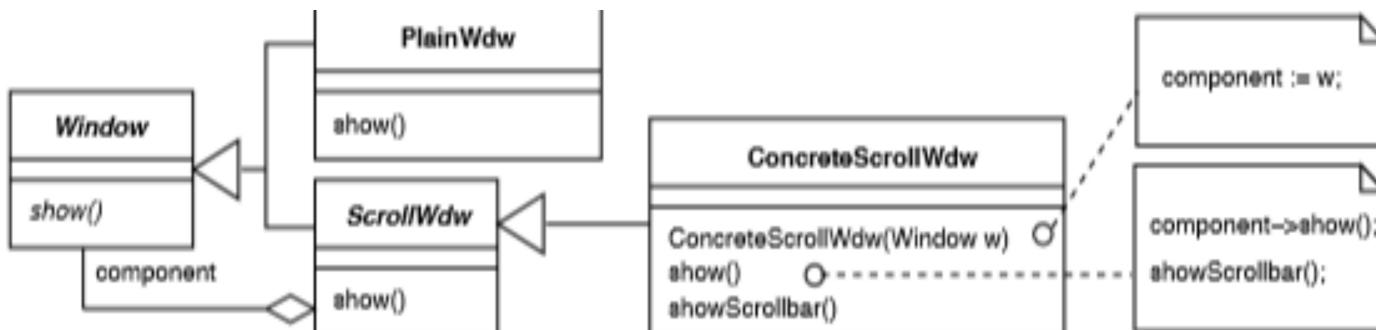
void Dispatcher::set_servant(ServantBase* s)
{
    srvt_ = s;
}

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 F

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`

```

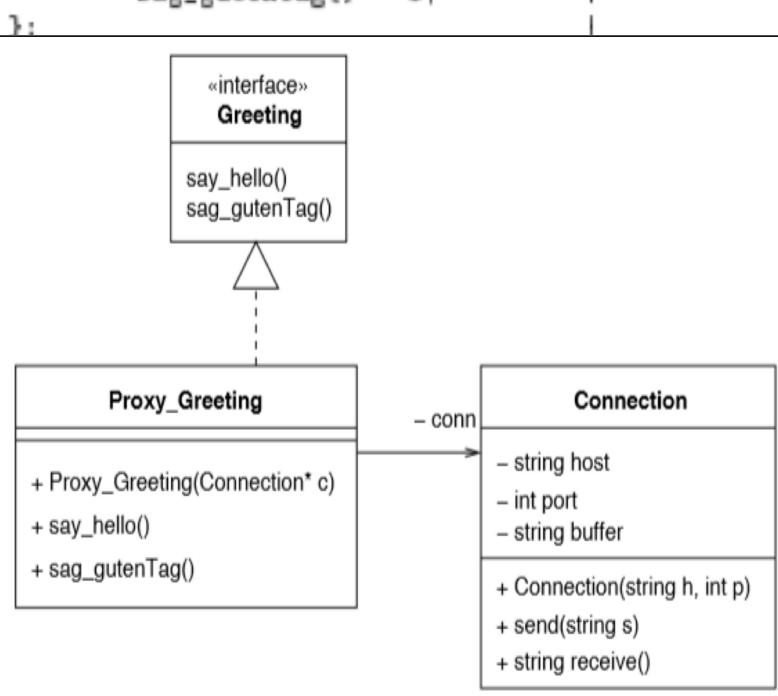
class Connection {
    string host;
    int port;
    string buffer;
public:
    Connection(string h, int p)
        : host(h), port(p) {}
    void send(string s);
    string receive();
};

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

```

class Proxy_Greeting : public Greeting {
    Connections* 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();
}
  
```



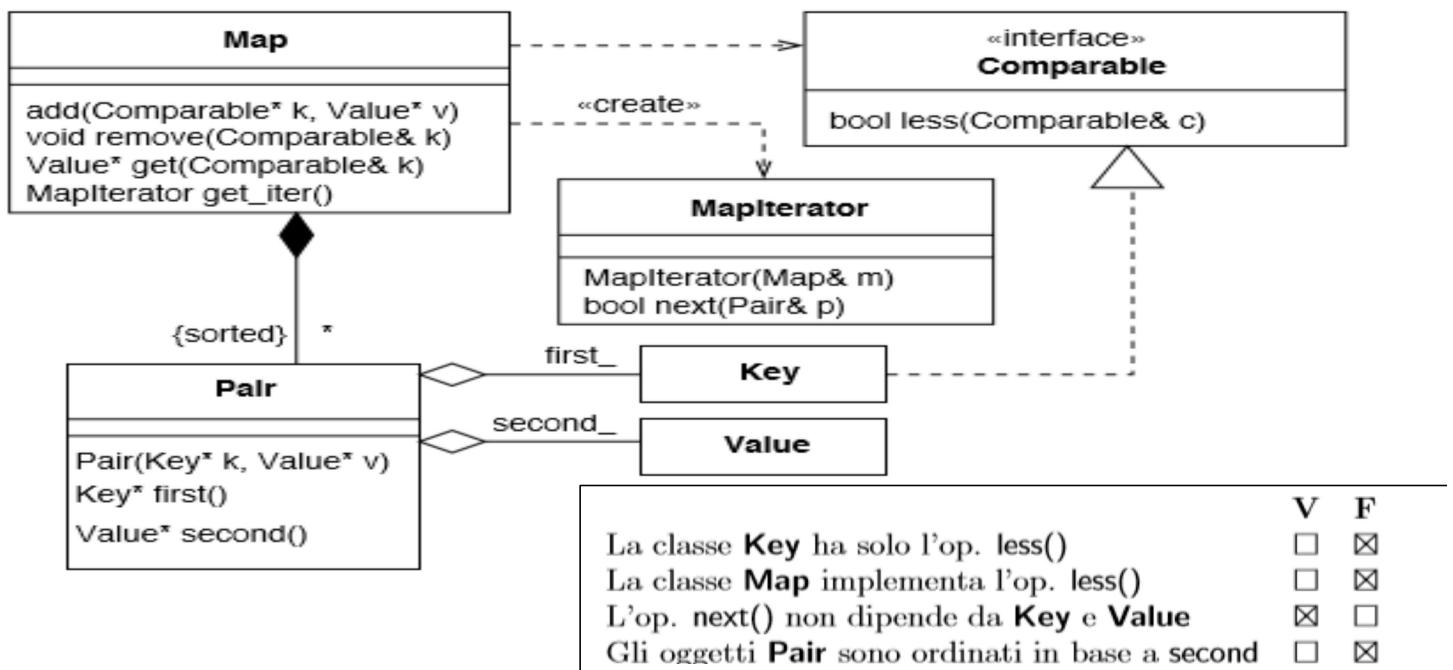
```

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

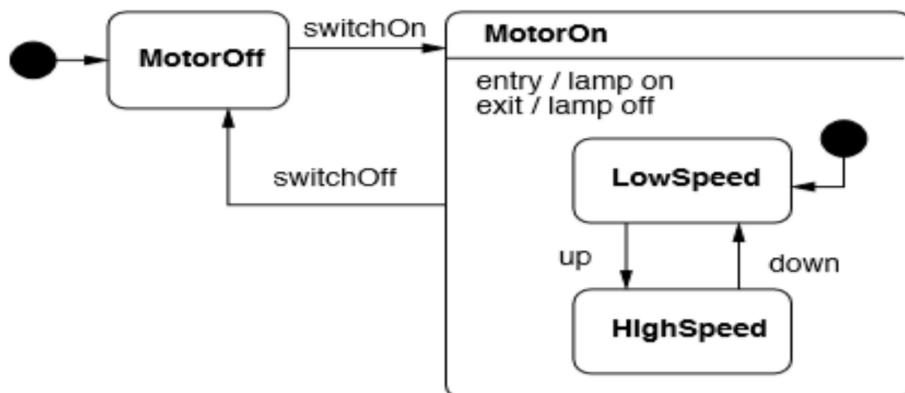
```

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



V F

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

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

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

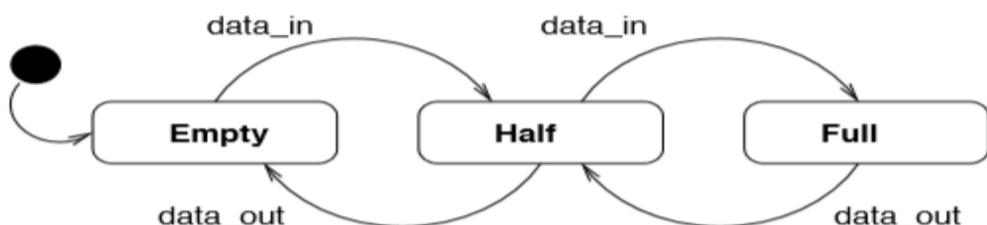
Durante il funzionamento c'è sempre una luce accesa.

Il sistema non ha uno stato finale.

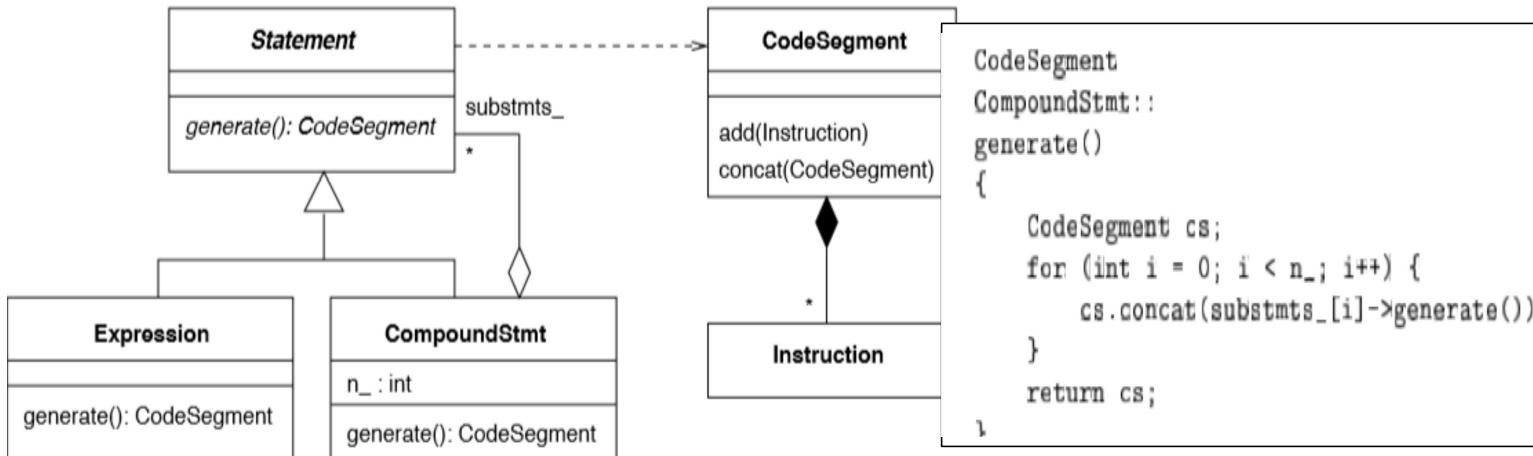
	Empty	Half	Full
Empty			
Half			
Full			

Tabella 1: Rappresentazione tabulare

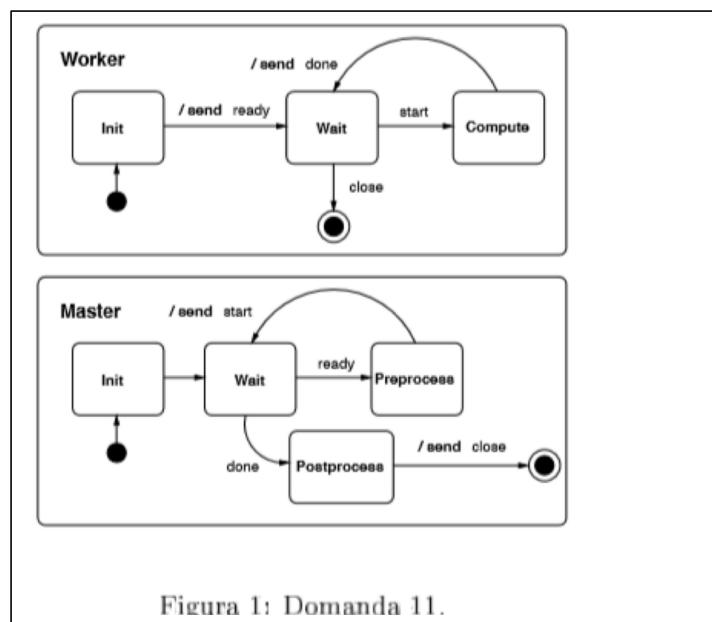
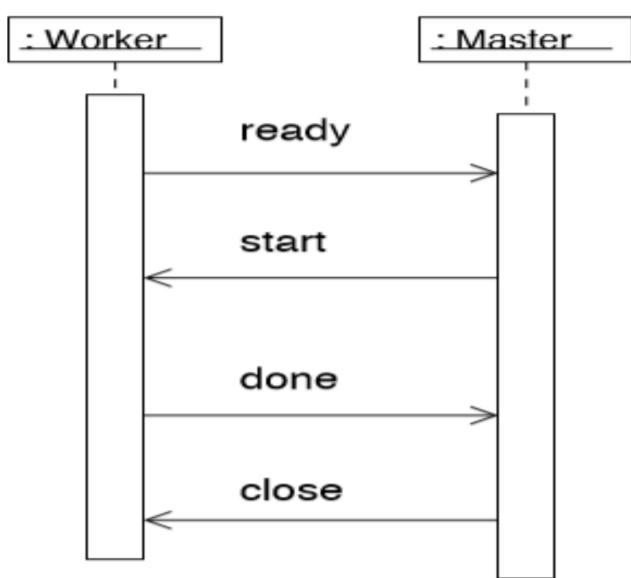
11 Disegnare l'ASF definito dalla Tab. 1.



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

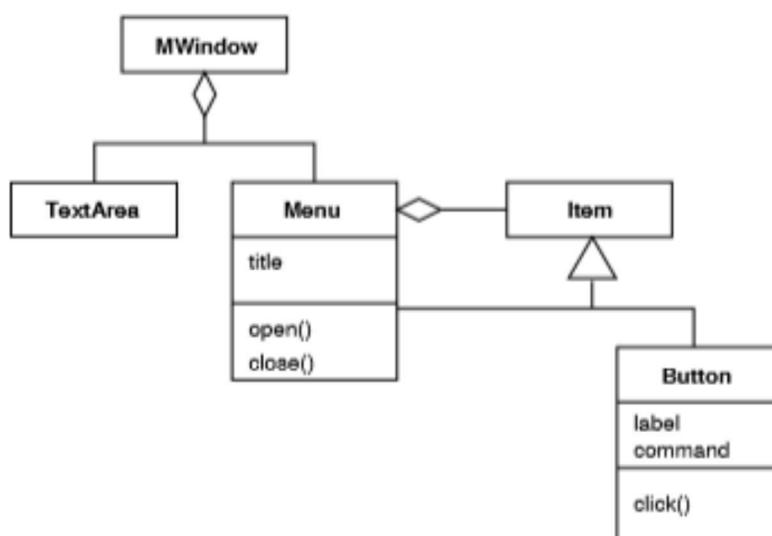


Disegna il diagramma di sequenza che mostra l'interazione fra due istanze delle classi Worker e Master il cui comportamento è specificato dallo Statechart



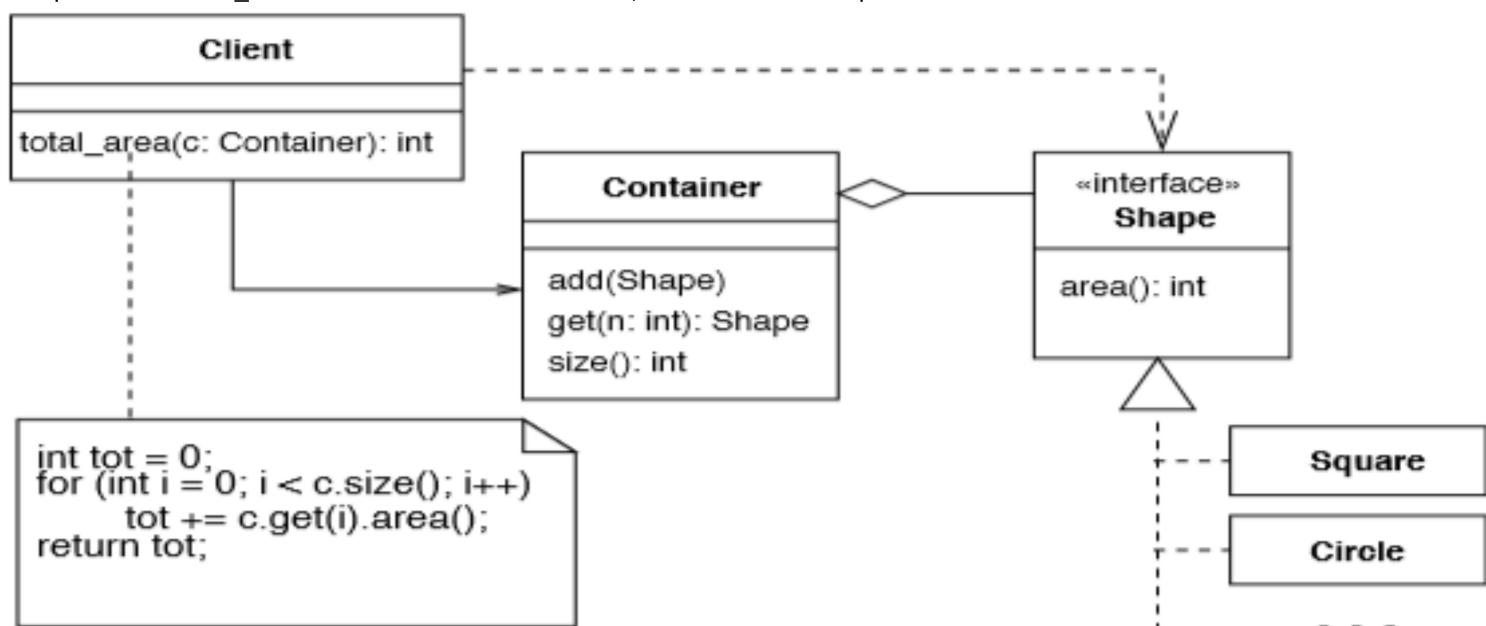
Descrivere in UML la seguente struttura:

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".

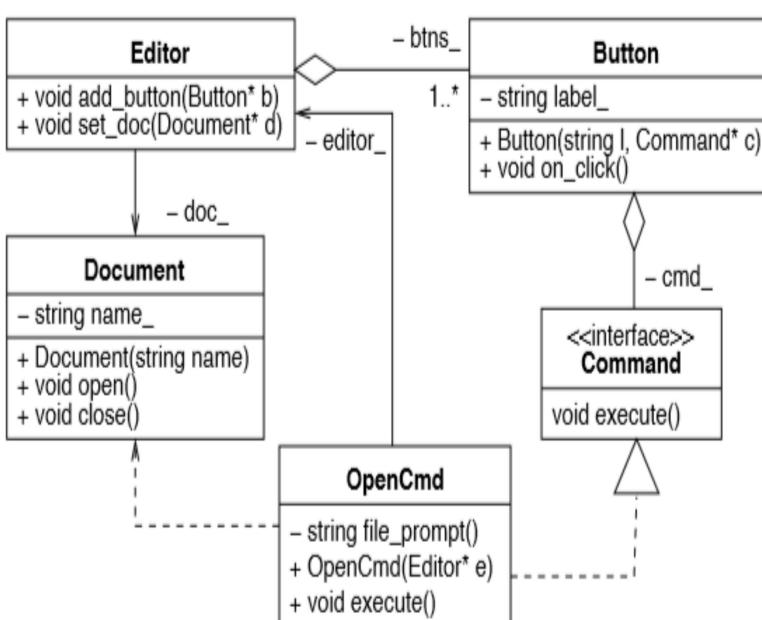




Disegnare un diagramma di classi relativo al seguente problema: Si hanno delle classi **Square**, **Circle** etc. che rappresentano figure geometriche. Ognuna di esse ha un'operazione `area():int` che restituisce l'area della figura. Si definisca l'interfaccia di una classe **Container** che contenga 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::evecute(), che deve aprire il documento specificato dall'utente, dopo averlo registrato nell'editor; supponendo già implementate tutte le altre operazioni, scrivere un programma principale che apra un documento.

```

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

```

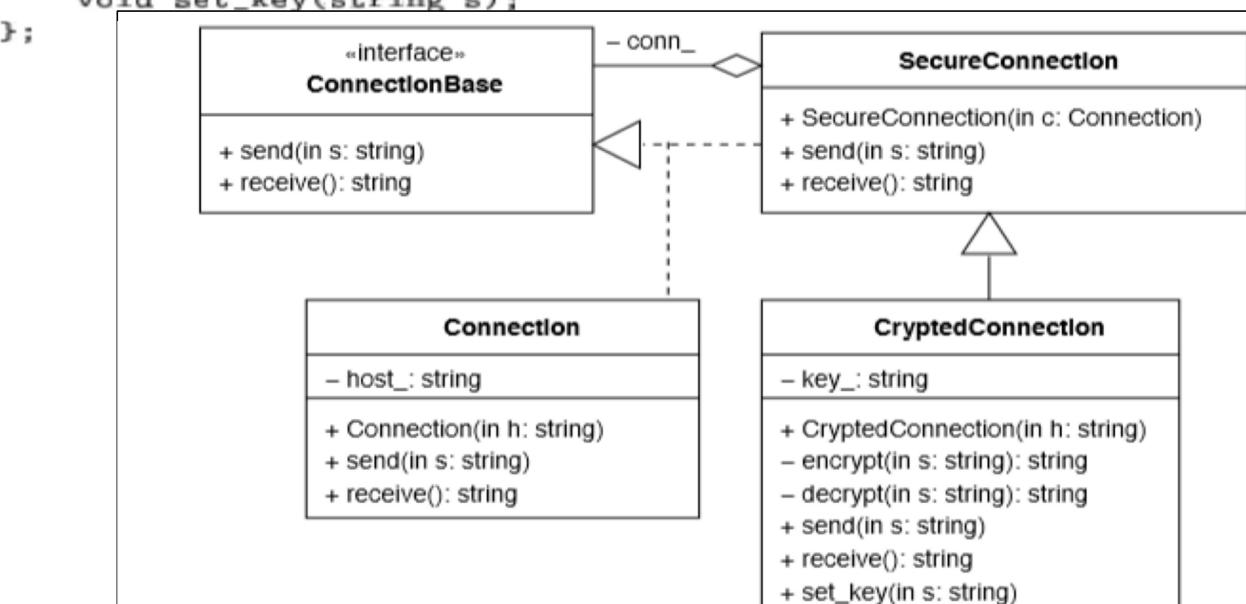
class ConnectionBase {
public:
    virtual void send(string s) = 0;
    virtual string receive() = 0;
};

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

```



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();
};

```

```

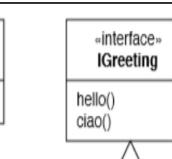
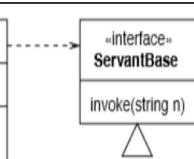
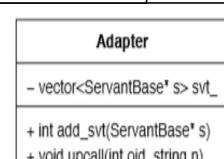
void Adapter::
upcall(int s, string n)
{
    svt_[s]->invoke(n);
}

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.

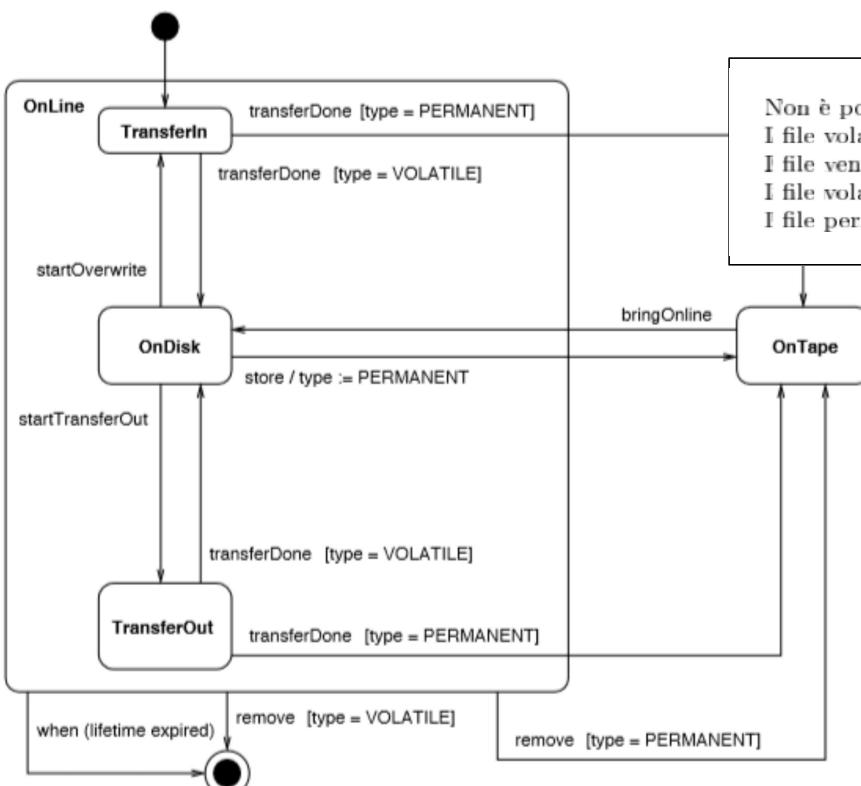
Implementare l'operazione Greeting_skl::invoke().

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



Non è possibile rimuovere un file su nastro
I file volatili non si possono rimuovere prima di transferDone
I file vengono creati nello stato **TransferIn**
I file volatili possono diventare permanenti
I file permanenti non possono stare su disco

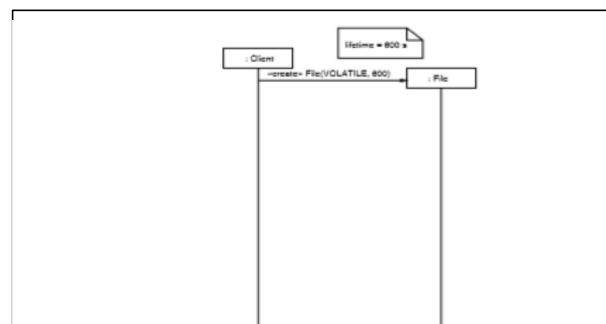


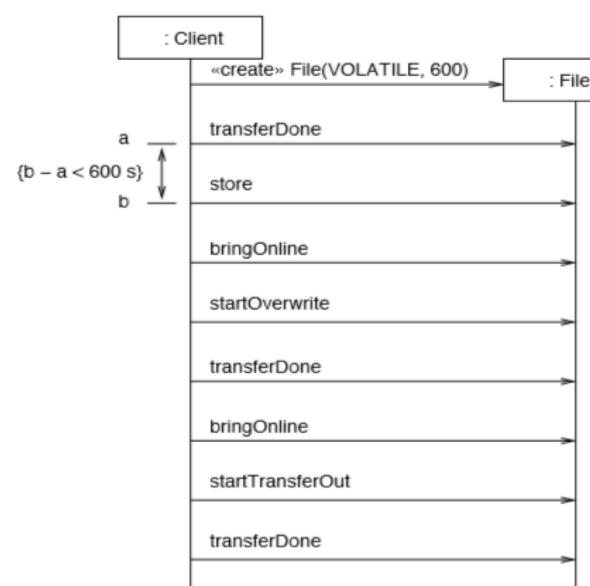
Figura 2: Domanda 12.

File

Completere il diagramma di sequenza di Fig. 2

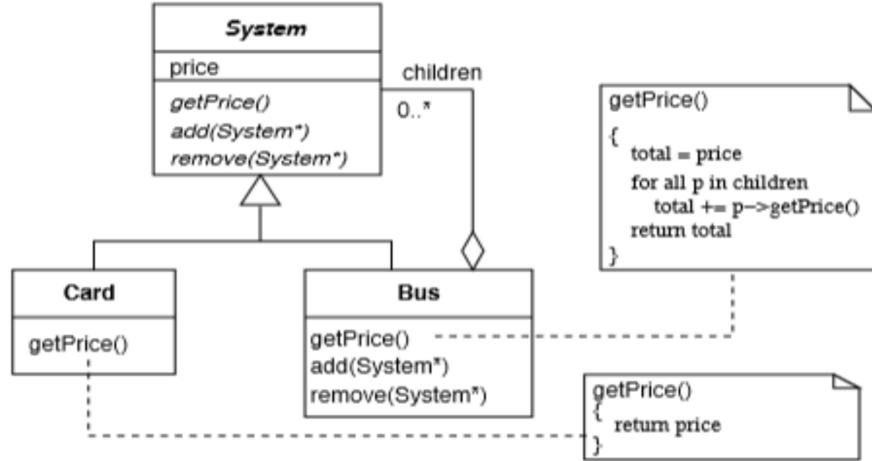
supponendo che il comportamento degli oggetti di tipo **File** sia specificato dal diagramma di Fig. 1 e mostrando le operazioni richieste per

- (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

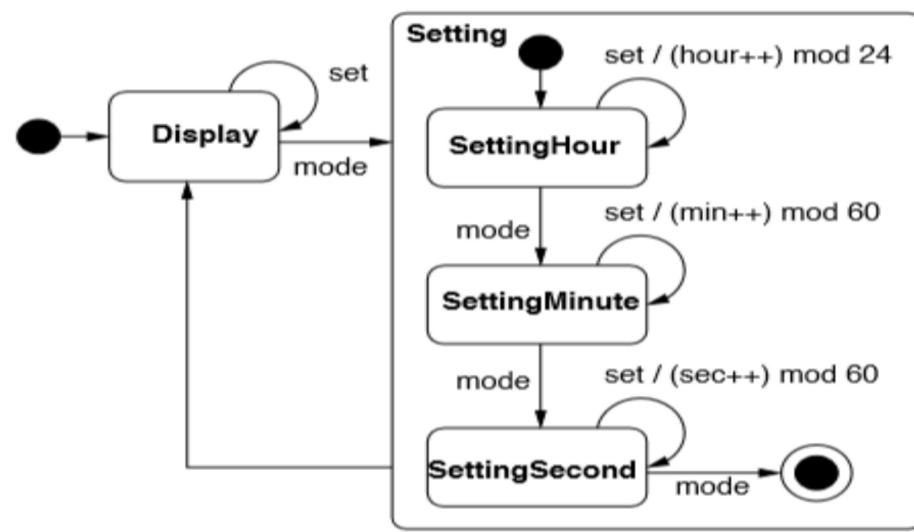
Disegnare un diagramma di classi che risolva il seguente problema:
Un sistema può essere un **Bus** o una **Card**. Ad un **Bus** si possono collegare zero o piú **Bus** e zero o piú **Card**. Ogni componente (bus o card) ha un prezzo. Vogliamo rappresentare la struttura di un sistema e calcolarne il 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.

