

- UML è linguaggio unificato
 - ↳ ci sono molti compromessi
- Modello di oggetti in UML

L'ANALISI

- Input: requisiti
- Output: schema concettuale
- Obiettivo: costruire un modello pratico completo e semplice
 - concentrarsi sul "cosa" e non sul "come"
- Serve a rispondere sui requisiti e prendere decisioni ad alto livello sulla strutturazione dell'applicazione

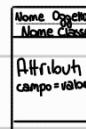
È un linguaggio fatto di diagrammi grafici:

- strutturali (class & object)
- comportamentali (statechart cenni)
- architettonici

classi (solo concettuali)



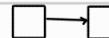
Oggetti



Gli oggetti hanno valori con identità, due oggetti con stessi valori sono oggetti distinti

Le associazioni legano le classi tra di loro
(o relazioni)

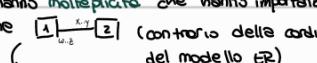
Tra più classi ci possono essere più associazioni



· Alcune associazioni hanno un verso (di percorrenza)

↔ → navigarlo in entrambi i versi

· Le associazioni hanno molteplicità che hanno importanza per la traduzione



1..x ↔ x..1 (contrario della cardinalità del modello ER)

1 ha min x e max y

2 ha min w e max z

0..* = * qualsiasi

1..1 = 1

i numeri precisi sono rari

o 0..* (qualsunque)

o 1..* (almeno d)

o 0..1 (possibile niente)

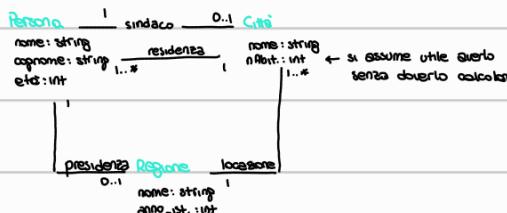
o 1..1

ES 1

persone (nome, cognome, etc)

città residenza (nome, abitanti, sindaco)

regione (name, anno istit., presidente)

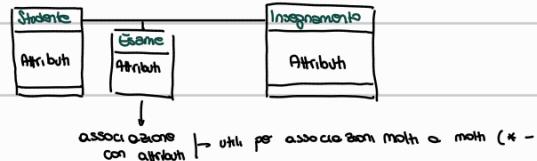


Nomi di ruoli



, fondamentale sulle associazioni con se stessi

Anche le associazioni possono avere attributi

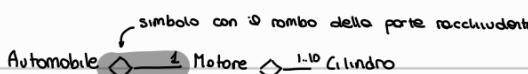


associazione con attributi → utili per associazioni molti a molti (* - *)

Una coppia di oggetti o esiste o meno, non posso avere più copie degli stessi oggetti o non sarebbe più un'associazione → devo promuoverlo a oggetto

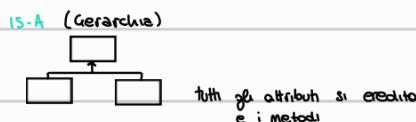
Associazioni tra più oggetti vengono definite **il-are** (con il numero di oggetti) ← usate solo se non divisibili in 3 classi.

part-of :



Composizione

part-of più forte; l'oggetto senza quello contenente non ha senso. Es capitolo non ha senso senza libro



Operazioni e Metodi

NomeMetodo (parametri) si dividono in interrogazioni (selettori) e modifiche
↳ si mettono solo i metodi importanti
declarazione NomeOperazione (NomeParametro:Tipo, ...): TipoRisultato (se omesso è void)

Le postcondizioni (o parole) descrivono cosa fa l'operazione; le precondizioni devono descrivere le condizioni che i dati devono rispettare (le descrive a parole)

I tipi possono essere retti, insiem, tipi primitivi o classi definite nell'UML

ES biblioteca

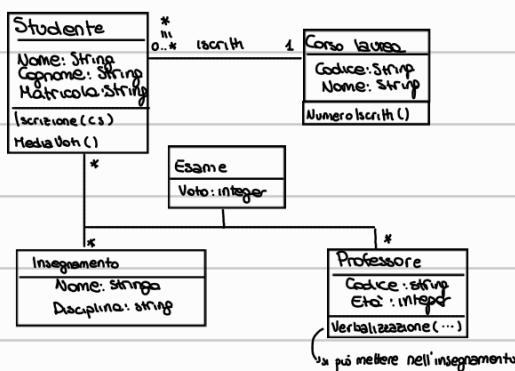
Acquisisce (l:libro): \varnothing libro l viene acquistato dalla biblioteca
Precondizioni: La biblioteca non possiede \varnothing libro l

Prestito (l:libro, p:Persona): l viene dato a p
Pre: libro possiede l e non è in prestito

Restituzione(l:Libro): \varnothing libro l viene restituito
Pre: l è in prestito

InPrestito (l:Libro): boolean : Restituisce true se il libro è in prestito

Presti (p:Persona): vettore(Libro) : restituisce il vettore di libri prestati a p



Le relazioni ternarie sono praticamente sempre M a M a M



Diagrammi comportamentali → come si evolvono

• Diagrammi di casi d'uso → spiega le interazioni tra utenti e sistemi

Diagrammi di sequenza

Esempio di interazione tra oggetti

Utente Centrale



Activity Diagram

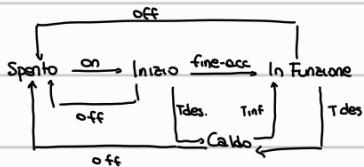
Enfatti quando un'operazione viene fatta con quali operazioni aspettare prima di andare in esecuzione

Diagrammi di Stato

- o Stato → valore di un insieme di attributi specifici

- o Eventi → Azione che provoca una transizione di stato

Gli stati e gli eventi si riferiscono ad un **singolo oggetto**



Esiste il concetto di **stato iniziale** e **stato finale**; possono essere molteplici

—→ —→ ⊙

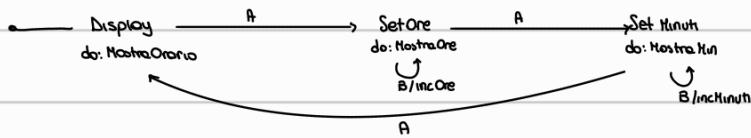
Si possono aggiungere delle condizioni es. Spento on[acqua non gelata] Inizio

hanno una certa durata & eseguita negli stati

Esistono **azioni** e **attività**
↓
eseguito nell'evento
evento [condizione]/azione

do: attività

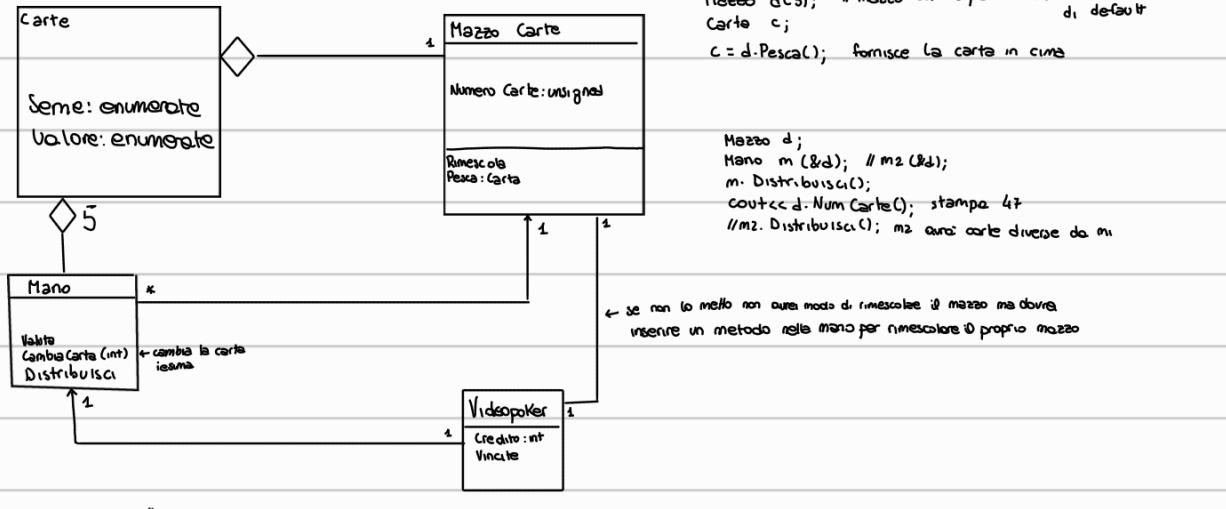
Esempio : Display a cristalli. Tasto A cambia selezione B incrementa di 1



Gli stati possono essere anonimi e contenendo un'operazione che al termine passa allo stato successivo

ciao, come stai?

Poker online



class carte

enum class Seme { Cuori, quadri, fiori, picche };
enum class Valori { asso, re, donna, fante, dieci, ..., tre, due }; // senso decrescente

Si dovrà fare vector<Carte> m; perché la carta non dipende dal mazzo

Posso giocare con numero zemi o valori diversi (<max>) i valori sono messi come costanti pubbliche assegnati dal costruttore

Il mazzo ha solo un valore vector<Carte> m;

Per mescolare il mazzo si usa il Fisher-Yates shuffle nel 1938

Tiro un numero a caso tra 0 e N-1 e lo swap con la prima posizione

Per tra 1 e N-1 e lo scambio con il secondo

Per rimescolare faccio resize con tutte le carte (anche se ho più mazzi)
Si ricordino i mazzi con tutte le carte a seconda del numero di mazzi

$mazza[K] = Carta(i / Num_Valori, i \% Num_Valori)$

Mescolare :

```

for (int i=0; i<mazza.size(); i++)
{
    j = random(i, mazza.size());
    swap(mazza[i], mazza[j]);
}
    
```

driver mazza → crea un mazzo con N pacchetti

→ pesca Y carte

→ vedo quante ne rimangono

→ rimedolo

} funzioni per provare le funzioni del mazzo

Il make fa il primo ma se aggiungo nella prima posizione all: tutti gli exe per farlo fare tutti i make e non solo il primo

Mano

è un vettore di carte interne alla mano; non ha accessi esterni
 \uparrow
5 elementi

Per valutare i punti ho un enum ordinato in base ai punti

Mano(Mazzo* m); ← dal costruttore e non ho modo poi di modificarlo

Punto Valuta() const; ← complicato

```

Mano::Mano (Mazzo* m)
: carte (5) ← il costruttore crea una mano di 5 carte di valori → costruttore default con simboli di cuori
{
    if (m->NUM_MAZZI != 1)
        throw logic_error;
    mazzo_utilizzato = m;
    // Eventuale distribuzione per consistenza ← ma mano inconsistente
}

```

Mano::Distribuisca

```

    { for (int i=0; i<5; i++)
        carte[i] = mazzo->PesaCarta();
    }

```

La volta crea 2 vettori, uno lungo quanti i valori e uno di simboli → conto cosa ho

valori di coppia
↑
2 tipi di punti → valori uguali o simboli uguali
↓

coppia 1, 2 massimi	4 - 1 => ho poker	se 1° max è 4 non prendo il secondo, negli altri casi
	3 - 2 => full	
	3 - 1 => tris	
	2 - 2 => doppio coppia	
	2 - 1 => coppia < vestito >	
	1 - 1 => tante possibilità	if annodati ogni sf per solo

array Si può usare per vettori statici ← ha gli stessi operatori di selezione
↑ no push_back etc...

voltata → usa solo i vettori delle carte ← non ha lunghezza fissata; dipende dal tipo di
↳ restituisce un punto Mazzo

Inizio calcolando il primo e secondo max (anche valori uguali)
↳ coppia restituita è data da J a A come valori

per i massimi uso un pair

conta: riempie i due vettori

cerca massimi: parto dal terzo valore e i primi due sono i primi massimi
↳ qui dico entrambi a ∵ perché il vettore ha valori positivi
↳ allora non c'è secondo max che io confronto anche lui

so che è +1 ma ci sarebbe occupamento

```

default: if (scala) tutte carte hanno simboli
    { if (colore)
        { // ho scala reale
            if (ho asso e re)
                { // ho scala reale max
                    {
                        else
                            // scala reale normale
                    }
                }
            else
                { vede se ho un 5 nei simboli
                    { if (colore)
                        // colore
                    }
                }
            else
                // niente
        }
    }

```

Scelta

- cerco il primo non zero ↑ con le f. chiamate
- vedo se il valore è +1 o se sono arrivato in fondo
- cerco 4 carte a 1 di fila con while
- re ne trovo una +1 return false
- la scala minima verifica per asso, 5, 4, 3, 2 (condizione esplicita)

Gli operatori di output sono fondamentali per debug anche se non lo si usa nel programma

Generatori randomici sono pseudo random e vengono generati a partire da un simbolo

↑
operazioni casuali a partire da X (simbolo)

c' sono generatori che forniscono il valore iniziale dopo $2^{19973} - 1$ volte ↴

in C si usava come simbolo 'time'

in C++ c'è una libreria che fornisce delle funzioni random ← si usa per generare il simbolo di questo algoritmo

static mantiene il valore nelle chiamate successive delle funzioni

(rand() % (b-a+1))+a ← in C

```

in cpp
    {
        random_device rd; ← no semc
        mt_19937 gen(rd()); ← algoritmo di generazione
        uniform_int_distribution<int> dist(a,b); ← distribuzione uniforme
        return dist(gen);
    }

```

Videopoker

Mano e Mazza come oggetti nel videopoker senza uso di pointer ← unico pointer è mano → mazza per il multiplayer che usano lo stesso mazzo

Funzioni: Gioca(); AggiungiCredito(); funzioni ausiliarie di output

private: f ausiliari della gioco (es. combina carte etc...)

```

C. default (credito) ← vettore con i punti per giri vinti
    : vincite{0,0,...,800}, mano (& mazza) ← singolo pacchetto
    { credito = c; ←

```

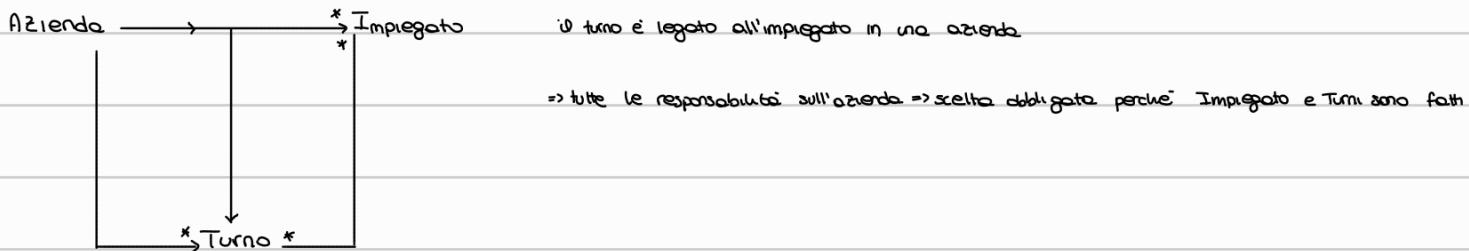
DistribuisceMano()

```

{
    mazza.Rimescola();
    mano.Distribuisce();
}

```

ES 6.10 UML esercizio azienda



In azienda carri un vettore di puntatori a impiegati ; vettore di turni ; vettore di copie <impiegato,turno>

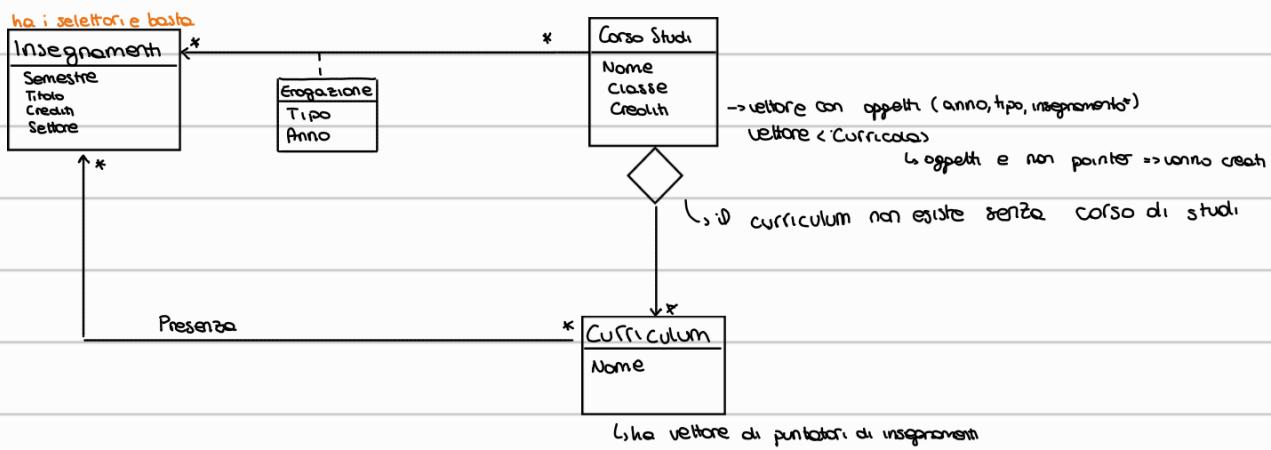
- ↳ correttamente puntatori.
- ↳ lui propone, per le specifiche (impiegati devono essere presenti nel vett.), di inserire gli indici nella coppia.
- Funziona per le altre due associazioni di azienda
- => forza l'utilizzo di turni e impiegati esistenti dovendo fare necessariamente i controlli
- => class/struct esterna public perché interno

Azienda con costruttore (uns. max-impiegati) → Azienda::Azienda (unsigned max-impiegati)
: MAX-IMPIEGATO (max-impiegato)

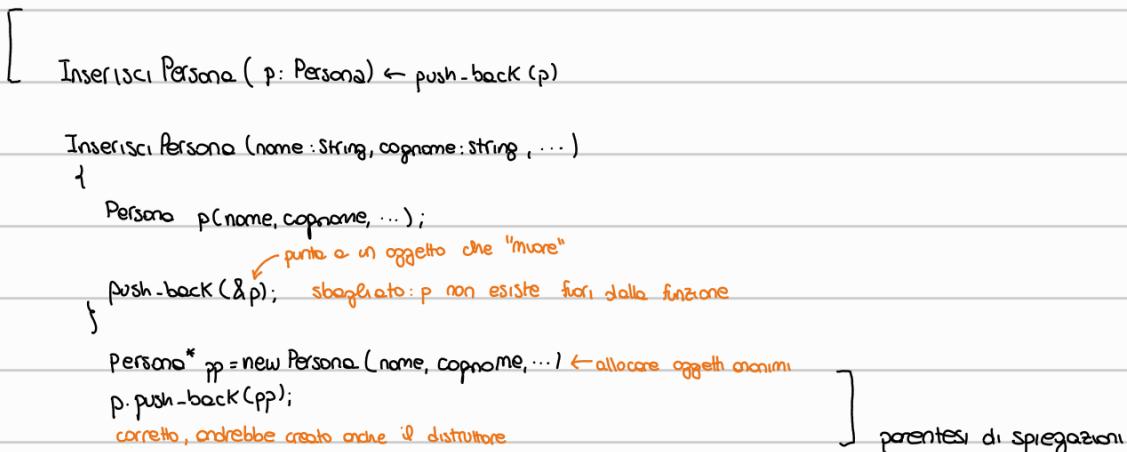
In Vedifassegnazione torrso Pair<Impiegato*, Turno*>

Precondizioni: if (condizione_da_evitare) throw invalid_argument(" ... ");

ES 6.13 Corso di studi



Per l'output implemento l'operatore "<<" in tutte le classi e le sfrutto per quelle più alte nella gerarchia



Currículum

```

string nome;
vector<Insegnamento*> insegnamenti;
unsigned crediti_tot; ← crediti di tutti gli insegnamenti, dato ridondante ma comodo per quando si vuole la somma dei crediti
int CercaInsegnamento ( Insegnamento* in ) const;
enum
    curso di studi, faccio una classe erogazione (Tipo, Anno, Insegnamento*)
public: ...
    CreaCurr ( string Nome );
    InsInInsegn ( Ins* in, Tipologia t, uns. a )
    InsInCurr ( Ins* in, string curr )
  
```

InserisciInIns
if (CercaIns (in) == -1)
 erogazioni.push-back (Erop. (in, t, a)); ← lo attacca

creaCurr
precond.
currcola.push-back (Curr. (nome)); ← lo crea

InserisciInCurr
c = CercaIns
i = CercaCurr
|→ precondizioni
currcola [c]. InsInIns (in);

consiglio **Somma Crediti** (tipologia +) ← restituisce i crediti della tipologia

f esterna ← vett<Corso Studi> n crediti dei corsi ^{obbligatori} i in comune a tutti

Insegnamenti in comune → vettore da CorsoStudi[0] di tipo obbligatori
↳ solo l'insegnamento, non la tripla

faccio il giro di ins. e se lo trovo come non obbligatorio/non lo trovo lo tolgo dal vettore
↓

dovrò usare un while che incremento l'indice solo se non elimino

mi rimane il vett di tutti gli ins. obbligatori in comune di tipo obbligatorio ← faccio la somma alla fine dei crediti

6.14



soluzione che non richiede l'uso di una matrice ← perché ho la casella