- ***<u>Lezione 15</u>***

UML ha soltanto due primitive in termini di relazioni di aggregazione (relzioni di contenimento)

- Aggregation

Si rappresenta con un rombo vuoto (◇). Ha una semantica per riferimento. Corrisponde alle aggregazioni di Has e Member

- Composition

Si rappresenta con un rombo pieno (◆). Ha una semantica per valore. Corrisponde alle aggregazioni ExclusiveOwns e Owns.

L'*Ereditarietà*, o *generalizzazione* (poiché in UML non esiste il concetto di ereditarietà, ma esiste una primitiva che si chiama generalizzation).
Usata per rappresentare la condivisione di attributi ed operazioni tra classi. Questa classe generica è detta superclasse, mentre le classi specializzate vengono dette sottoclassi o classi derivate.
Una sottoclasse *eredita* attributi ed operazioni della superclasse.
Abbiamo due caratteristiche per quanto riguarda l'ereditarietà:

- Sostituibilità → un oggetto della sottoclasse può essere utilizzato come valore da assegnare ad una variabile della superclasse.

  `Es: una variabile di tipo Frutta può avere un oggetto di tipo Mela come suo valore`

- Polimorfismo → la stessa operazione può avere differenti implementazioni nelle sottoclassi. Il sistema a Run - Time è in grado di disambiguare automaticamente il tipo di oggetto al quale sta chiedendo di fare qualcosa e ovviamente seguire il metodo corretto.

UML supporta queste associazioni di generalizzazione:

Dal punto di vista della sintassi visuale tutto questo si rappresenta con una freccia che va dalla sottoclasse alla superclasse.

Possiamo identificare relazione di tipo generalizzazione quando leggiamo:

- "può essere" (can – be)

  `Es: lo studente può essere un TeachingAssistant (Assistente didattico)`

- "è un tipo di" (is – a – kind – of)

Es: TeachingAssistant è un tipo di studente

UML supporta il concetto di ereditarietà multipla. Permette ad una classe di ereditare il comportamento di più superclassi.

Una rappresentazione di istanze di classi è il Diagramma degli Oggetti. Questo si chiama così proprio perché rappresenta gli oggetti. Questo è utile perché non può esistere un oggetto senza una classe corrispondente.

Si usa per.

- modellare relazioni complesse tra classi (a scopo esemplificativo)
- illustrare le modifiche ai singoli oggetti durante l'evoluzione del sistema
- illustrare la collaborazione tra oggetti durante l'evoluzione del sistema

Possiamo introdurre il Modello Comportamentale Questo modello si concentra sugli aspetti funzionali del sistema da un punto di vista operativo, evidenziando come gli oggetti collaborano ed interagiscono al fine di offrire i servizi che il sistema mette a disposizione.

Questo modello fa uso di vari (diagrammi) formalismi:

- Use case diagram (per descrivere scenari di funzionamento)
- Activity diagram (per descrivere il flusso di elaborazione)
- Sequence diagram (per descrivere l'interazione tra gli oggetti)
- Collaboration diagram (per descrivere l'interazione tra gli oggetti)

Anche questo modello viene costruito in modo iterativo ed incrementale, usando le informazioni del modello dei dati, che a sua volta fa uso del modello comportamentale per identificare operazioni e classi aggiuntive (control classes e boundary classes).

Analizziamo questi diagrammi:

- Diagramma dei casi d'uso (Use Case Diagram)

  Questo diagramma deve descrivere una funzionalità completa. Deve essere una funzionalità visibile dall'esterno. Deve avere un comportamento ortogonale (ogni use case viene eseguito in modo indipendente dagli altri). Non posso avere un caso d'uso se non ho un attore che lo attiva. Rappresenta una funzionalità che produce un risultato significativo per un attore.

Può essere sviluppato a differenti livelli di astrazione (sia in fase di OOA che OOD).

I casi d'uso li identifichiamo sia a partire dai requisiti utente, sia soffermandosi sugli attori e le loro necessità.

UML ci fornisce una sintassi per la specifica dei casi d'uso:

- Caso d'uso che si rappresenta con un'ellisse
- L'attore che si rappresenta con il simbolo di un omino

Questi elementi sono collegati attraverso relazioni, avremo due casi:

- Quando la freccia parte dall'attore e va al caso d'uso → stiamo indicando un attore responsabile dell'attivazione di quel caso d'uso
- Quando la freccia parte dal caso d'uso e va verso l'attore→ stiamo indicando se l'attore è coinvolto nell'esecuzione di quel caso d'uso (lui non lo attiva, è solo coinvolto)

Esistono relazioni tra due casi d'uso (escluso l'attore)

- Caso d'suo A collegato con una relazione di INCLUDE al caso d'uso B → sto indicando che "A include B", cioè quando un attore attiva il caso d'uso A per poter completare A deve necessariamente attivare e completare B
- Caso d'suo A collegato con una relazione di EXTEND al caso d'uso B → sto indicando che "A estende B", cioè quando un attore attiva il caso d'uso B potrei in alcuni casi (non necessariamente) attivare A, che a sua volta può essere attivato o da un attore differente o dallo stesso attore

Infine abbiamo una relazione tra coppie di attori chiamata Generalizzazione. Se io ho un attore A che viene specializzato da un attore B (freccia da B ad A), significa che B può attivare tutti i casi d'uso che attiva A e tutti i casi d'uso suoi specifici.