UML

Unified Modeling Language

Cos'è UML

- E' un linguaggio di **progettazione**, da non confondere con i linguaggi di **programmazione** (C, C++, Java,...)
- Fornisce una serie di diagrammi per rappresentare ogni tipo di modellazione
- Alcuni ambienti di programmazione sono in grado di convertire diagrammi UML in codice e viceversa

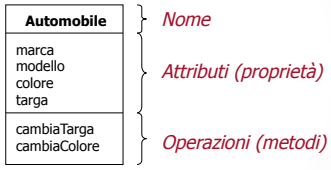
Diagrammi UML

- diagramma dei casi d'uso (use case)
- diagramma delle classi (class)
- diagramma di sequenza (sequence)
- diagramma di collaborazione (collaboration)
- diagramma di stato (statechart)
- diagramma delle attività (activity)
- diagramma dei componenti (component)
- diagramma di distribuzione (deployment)

Diagramma delle classi

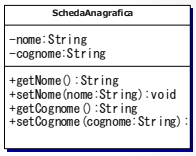
- Rappresenta le classi che compongono il sistema, cioè le collezioni di oggetti, ciascuno con il proprio stato e comportamento (attributi ed operazioni)
- Specifica, mediante associazioni, le relazioni fra le classi.

Un esempio

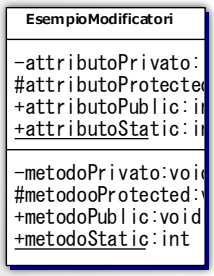


Classe Metodi e Attributi

```
public class SchedaAnagrafica {

    private String nome;

    private String cognome;

    public String getNome() {
      return nome;
    }
    public void setNome(String nome) {
      this.nome = nome;
    }
    public String getCognome() {
      return cognome;
    }
    public void setCognome(String cognome) {
      this.cognome = cognome;
    }
}
```

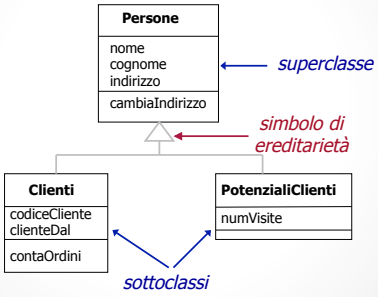


Modificatori

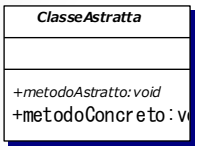


- +Public: Libero Accesso
- #Protected: Accessibile dalle Sottoclassi
- -Private: Accessibile solo all'interno della classe
- Static: Accessibili anche senza creare istanze

Ereditarietà

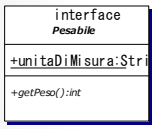


Classi Astratte e Metodi Astratti



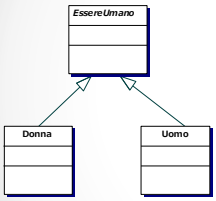
- Una Classe Astratta contiene metodi privi di implementazione
- Per questa ragione non può essere istanziata
- Il corsivo permette di distinguere le parti astratte da quelle concrete

Interfacce




```
public interface Pesabile {

  public static String unitaDiMisura;

  public int getPeso();
}
```

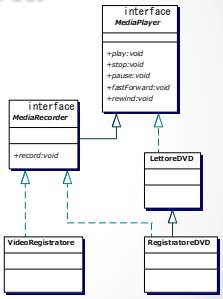
Ereditarietà




```
public abstract class EssereUmano {
    ....
}

public class Donna extends EssereUmano {
    ....
}

public class Uomo extends EssereUmano {
    ....
}
```

Interfacce ed ereditarietà multipla

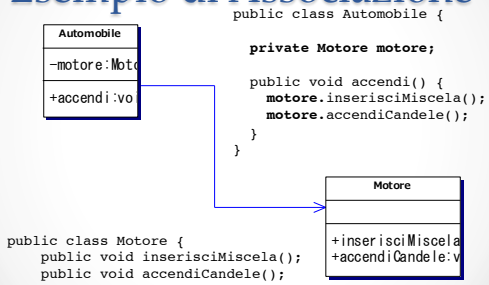
Nota: L'ereditarietà multipla tra interfacce permette di porre in essere esempi di ereditarietà a diamante



Associazione

- Un'**Associazione** rappresenta la possibilità che un'istanza ha di inviare un messaggio ad un'altra istanza
- In UML viene rappresentata con una freccia, in Java viene implementata tipicamente con un reference

Esempio di Associazione




```
public class Automobile {

  private Motore motore;

  public void accendi() {
    motore.inserisciMiscela();
    motore.accendiCandele();
  }
}
```

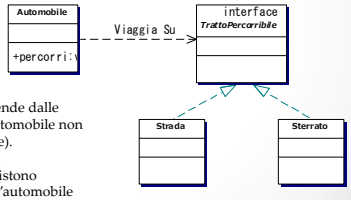
```
public class Motore {
    public void inserisciMiscela();
    public void accendiCandele();
}
```

Dipendenza

- La **Dipendenza** indica che un determinato oggetto può, in certe circostanze, chiamare i metodi di un altro pur senza possederne un'istanza
- La classe dipendente presuppone l'esistenza della classe da cui dipende.
- Non vale il viceversa
- In UML la dipendenza viene rappresentata con una freccia tratteggiata. In java tipicamente l'oggetto dipendente riceve un'istanza dell'oggetto da cui dipende come argomento di una chiamata a metodo

Dipendenza

```
public class Automobile {

  public void percorri(TrattoPercorribile p) {
  ....
  }
}
```



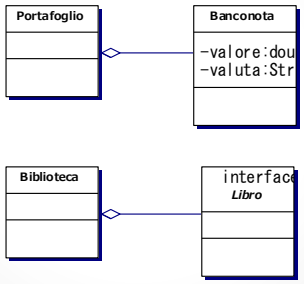
Nota: L'automobile dipende dalle strade (senza strade l'automobile non avrebbe senso di esistere).

Al contrario le strade esistono indipendentemente dall'automobile

Aggregazione

- L'**Aggregazione** rappresenta un'associazione uno a molti
- Esprime concetto "è parte di" (part of), che si ha quando un insieme è relazionato con le sue parti
- In UML l'aggregazione viene rappresentato con una freccia con la punta a diamante; in Java viene implementata con un array dinamico tipo "Vector"

Esempi di Aggregazione



Composizione

- Una Composizione è una relazione uno a molti che implica una forma di esclusività
- E' un caso particolare di aggregazione in cui:
  - la parte (componente) non può esistere da sola, cioè senza la classe composto
  - una componente appartiene ad un solo composto
- La distruzione dell'oggetto che rappresenta il "tutto" provoca la distruzione a catena delle "parti"
- Il diamante si disegna pieno

Esempi di Composizione