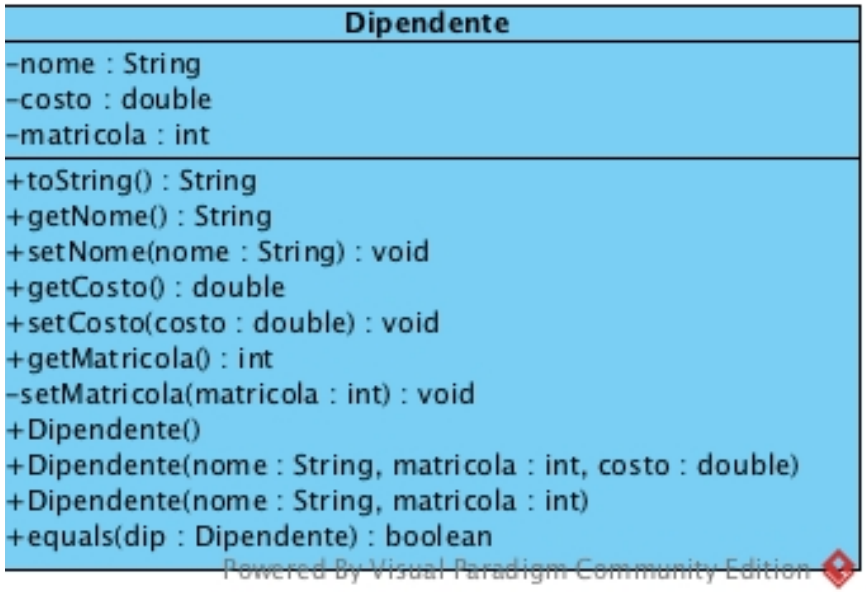
Programmazione con Java

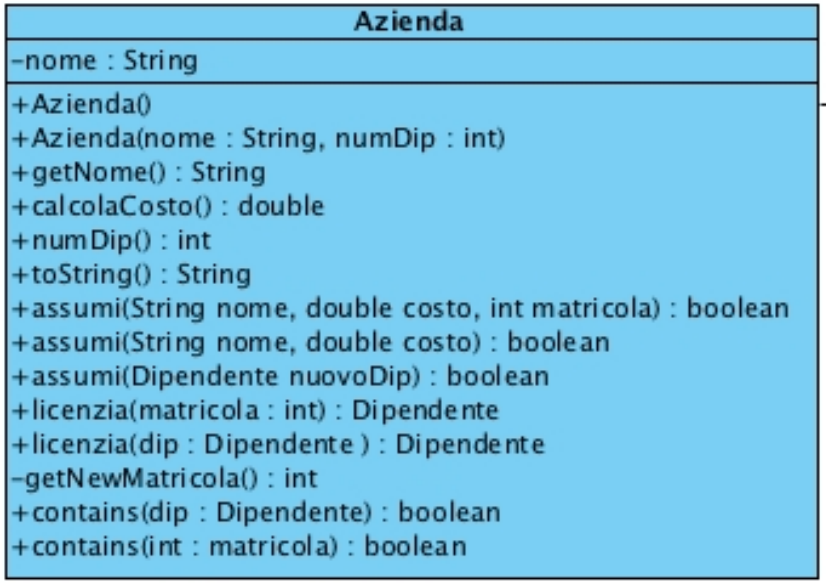
Information Hiding and Constructors: Esercizi

Esercizio: Azienda e Dipendente

- In `Dipendente`, aggiungete
  - i `set` e i `get` per i campi `nome` e `costo`: `matricola` è immutabile
  - Un costruttore prende i valori per i 3 attributi e li setta
  - Un costruttore senza il costo, che lo setta automaticamente a 1000,
  - Un costruttore senza parametri, che inizializza anche il nome a “prova” e matricola a 0
  - Il metodo `equals` restituisce `true` se il dipendente in input è `identico` a `this` o se ha stessa `matricola`

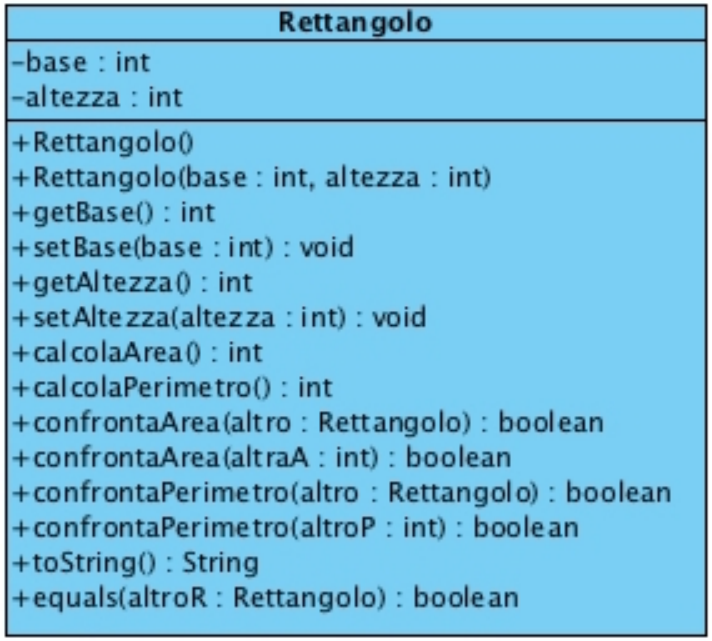


- In `Azienda`:
  - L'array diventa privato
    - il metodo `numDip()` restituisce il numero massimo possibile dei dipendenti (lunghezza array)
  - La dimensione dell'array è passata attraverso il costruttore con i parametri, assieme al nome
  - Nel costruttore di Default:
    - la dimensione dell'array deve essere settata a 1 di default
    - Il nome è settato a "prova"
  - Il metodo privato `getNewMatricola`() restituisce un numero di `matricola` non presente nell'array (max +1)
  - Il nuovo `assumi` crea un dipendente con nome e costo passati in input e matricola non già presente nell'array, e lo assume
  - Aggiornate i metodi `contains` così che usino il concetto di "Dipendente uguale"



Esercizio: Rettangolo

- Base e Altezza diventano privati, e accessibili tramite i set/get
- Il costruttore di default inizializza base ed altezza a 1, l'altro prende come parametri i due valori
- Il metodo `equals` ritorna `true` se `base` ed `altezza` sono uguali a quelle del rettangolo in input (o se è lo stesso oggetto)
- Provate poi a modificare il metodo `equals`, così che restituisca `true` se le `aree` dei due rettangoli sono uguali (a prescindere da base ed altezza)



- I metodi `calcolaArea`(), `calcolaPerimetro`(), `confrontaArea`, `confrontaPerimetro` e `toString()` sono quelli dell'esercizio precedente

Esercizio: Rettangolo

- Realizzare programma (main) che:
  - Dichiara e istanzia un rettangolo senza parametri (e verifica che abbia base ed altezza pari a 1);
  - Ne istanzia un altro con base ed altezza a vostra scelta
  - Stampa a video i dati dei 2 rettangoli (`toString()`);
  - Stampa a video il risultato di r1.`equals(r2)`
  - Dopo aver modificato l'equals, testatelo creando un nuovo rettangolo, con base ed altezza invertiti rispetto al secondo
    - Provate anche ad implementare una `equals` che restituisca `true` se `base` e `altezza` sono invertiti
- Nella precedente esercitazione avevamo creato un metodo confrontaAree, che confrontava le aree di due rettangoli in input: aggiornatelo in base alla nuova visibilità degli attributi... Cambia qualcosa?
- Aggiornate i due setter in modo che mettano `base`, o `altezza`, uguali a 1 se il parametro in input è minore o uguale a zero, e testateli nel main