



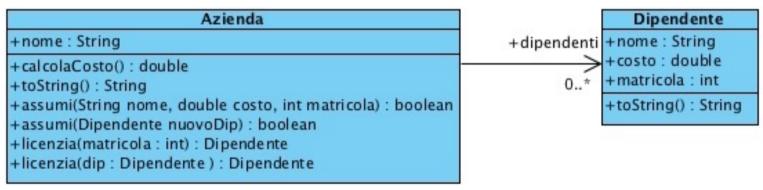


Programmazione con Java

Incapsulamento: Esercizi

Aggiornare le classi della precedente esercitazione aggiungendo i seguenti metodi:

- I metodi toString() che restituiscono una rappresentazione testuale degli oggetti Dipendente (i suoi attributi) e Azienda (elenco dipendenti e costo totale)
- Il metodo calcolaCosto() che restituisce il costo totale dei dipendenti
- I metodi assumi (nome, matricola, costo) e licenzia (matricola) aggiungono e rimuovono un dipendente da quelli associati all'azienda (se possibile)
- Creare le versioni (overload) dei metodo assumi (...) e licenzia (...) che prendano in ingresso un oggetto di tipo Dipendente
 - Nb: evitate i "code clone"! (ovvero fate in modo che uno dei due metodi richiami l'altro)





- Realizzate una programma (main) che testi le nuove modifiche:
 - Dichiara e istanzia un'azienda con massimo 3 dipendenti
 - Crea un dipendente (matricola 1, scegliete voi gli altri valori)
 - Stampa a video le informazioni del dipendente
 - L'azienda assume il dipendente appena creato
 - Stampa le informazioni dell'azienda (toString())
 - Assume un dipendente con matricola=2 (e altri valori a vostra scelta)
 - Stampate le informazioni a video e verificate che tutto funzioni
 - Licenziate il dipendente con matricola 1, e stampate a video
 - Assumete un nuovo dipendente con matricola 3
 - Stampate (dove si trova il dipendente 3 nell'array...?)



- In Dipendente, aggiungete
 - Un costruttore prende i valori per i 3 attributi e li imposta alla creazione di un oggetto di tipo Dipendente
 - Un costruttore senza il parametro costo, che lo imposta un valore di default costo=1000
 - Un costruttore senza parametri, che inizializza nome="innominato", costo=1000 e matricola=0
 - Il metodo equals che restituisce true se il dipendente in input è identico o ha la stessa matricola di quello dato



In Azienda:

- Il nome e la dimensione dell'array sono impostati attraverso il costruttore
- Aggiungete un costruttore "di default" che:
 - Imposta la dimensione dell'array a 1
 - Imposta nome="prova"

```
Azienda
-nome : String
+Azienda()
+Azienda(nome : String, numDip : int)
+getNome(): String
+calcolaCosto(): double
+numDip(): int
+toString(): String
+assumi(String nome, double costo, int matricola): boolean
+assumi(String nome, double costo): boolean
+assumi(Dipendente nuovoDip): boolean
+licenzia(matricola: int): Dipendente
+licenzia(dip : Dipendente ) : Dipendente
-getNewMatricola(): int
+contains(dip : Dipendente) : boolean
+contains(int: matricola): boolean
```



- Aggiungete 2 metodi nella classe Azienda:
 - contains (Dipendente d), che restituisce true sse azienda contiene già un dipendente uguale (considerando solo la matricola) o identico
 - contains (int matricola), che restituisce true sse azienda contiene già un dipendente uguale (considerando solo la matricola)
- Verificate nel main che la vostra azienda contenga già il dipendente d1 e un dipendente con matricola 5



Esercizio: Rettangolo

- I metodi calcolaArea() e calcolaPerimetro() restituiscono l'area e il perimetro
- I metodi confrontaArea() e confrontaPerimetro() restituiscono true se l'area/perimetro del rettangolo è maggiore del valore passato in input
 - Sono in overloading con una versione che accetta in input un altro Rettangolo con cui confrontare area/perimetro
- Il metodo toString() stampa base, altezza, area e perimetro

```
Rettangolo

+base : int

+altezza : int

+calcolaArea() : int

+calcolaPerimetro() : int

+confrontaArea(altro : Rettangolo) : boolean

+confrontaArea(altraA : int) : boolean

+confrontaPerimetro(altro : Rettangolo) : boolean

+confrontaPerimetro(altroP : int) : boolean

+toString() : String
```



Esercizio: Rettangolo

- Realizzare programma di test che:
 - Dichiara e istanzia due rettangoli;
 - Stampa a video i dati dei 2 rettangoli (toString());
 - Stampa a video il risultato di r1.confrontaArea (r2) e r1.confrontaPerimetro (r2)
 - Stampa a video il risultato di r1.confrontaArea (10) e r1.confrontaPerimetro (10)
 - Aggiungete un metodo static confrontaAree (Rettangolo r1, Rettangolo r2) che restituisce true se l'area di r1 è maggiore di quella di r2
 - Nel main, usate questo nuovo metodo per confrontare r1 e r2

