Consigli pratici per usare al meglio Java

Informatica II

28/3/2019

Angelo Gargantini

Campi o metodi?

- Alcune volete non sappiamo se una caratteristica degli oggetti che sto implementando va messa come CAMPO o METODO.
  - Ad esempio:
    - Un rettangolo ha una base, una altezza e una area
- Cosa mettere come campi?
  - I campi caratterizzano ogni oggetti in modo unico e quando costruisco un oggetto devo inizializzarli
  - I metodi interrogano l'oggetto per sapere quale è lo stato (oppure cambiano lo stato dell'oggetto)
  - I campi possono cambiare da oggetto a oggetto
  - In genere NON posso dedurre i campi semplicemente da altri campi (dallo stato dell'oggetto), altrimenti a quel punto meglio un metodo
  - Esempio:
    - Dalla base NON riesco a dedurre l'altezza di un rettangolo -> base e altezza devono essere due campi distinti
    - L'area invece dipende dagli altri due campi: uso un metodo

Quando usare l'ereditarietà o la composizione?

- Quando uso l'ereditarietà?
- Quando uso la composizione (campo del mio oggetto)?
  - Ad esempio: Studente Persona Matricola
  - Potrei mettere?
    - class Studente extends Matricola{ - uso l'eredità per Studente da Matricola
    - class Studente{ Persona datiPersonali; - uso la composizione per Studente da Persona
  - Va bene? **Sintatticamente sì – anche**
- Per l'ereditarietà la relazione IS_A = E' UN
- Per la composizione la relazione HAS_A = HA UN
- Quindi…
  - Meglio invece:
    - class Studente extends Persona{ - uso l'eredità per Studente da Persona
    - class Studente{ Matricola matricola; - uso la composizione per Studente da Matricola

Quando usare l'ereditarietà o un campo per indentificare sotto tipi?

- Se ho degli oggetti che si comportano in modo diverso, metto un campo che li distingue o faccio sottoclassi diverse?
- Esempio: Prenotazione che può essere individuale o di gruppo
- Faccio?
  - Come campo
    - enum TipoPrenotazione {individuale, digruppo}
    - class Prenotazione { TipoPrenotazione tipo}
  - Come sottoclassi:
    - class Prenotazione {} class PrenotazoneIndividuale extends Prenotazione …
- Se il comportamento è molto simile e il tipo può cambiare uso il campo
- Se il comportamento è diverso e non può cambiare uso l'ereditarietà.

Quando usare classi astratte

- Quando uso sottoclassi astratte?

- Ad esempio Figura, perché astratta? Potrei fare
  `getArea(){return 0;}` e poi fare overriding …

- Uso la classe astratta quando:
  - Alcune funzionalità verranno implementate dalle sottoclassi e questi sono metodi astratti
  - Non creo istanze di quella classe, anche se c'è il costruttore

Quali campi mettere nelle sottoclasse e quali nella superclasse

- Metti i campi nella superclasse per evitare duplicazioni
  - Pull up
  - Se ho un campo identico nelle sottoclassi -> lo porto nella superclasse
- Metti i campi nella sottoclassi per evitare campi inutili che servono solo nelle sottoclassi
  - Push down
  - Se ho un campo che serve solo ad una sottoclasse -> lo porto nella sottoclasse

- Attenzione:
  - Inizializzo i campi della superclasse nel suo costruttore non nel costruttore della sottoclasse
  - NON DUPLICO MAI gli stessi campi nella superclasse e nella sottoclasse

Quando fare overriding

- Quando la funzionalità è specifica delle sottoclassi ridefinisco i metodo e faccio overriding
- La sottoclasse avrà il metodo implementato in modo diverso
- Buoni candidati sono
  - toString
  - equals

- Non faccio mai un metodo nella super classe e poi uso if this instanceof .. per decidere cosa fare: uso invece overriding
- In generale non uso mai instanceof per decidere il comportamento degli oggetti

Quando fare overloading?

- Quando una certa funzionalità può essere richiamata con diversi ingressi, allora posso fare diversi metodi con lo stesso NOME ma diversa segnatura
- Candidato tipico i costruttori
  - Posso costruire certi oggetti passando anche dei valori di default, allora faccio costruttori con segnature semplificate
  - In questi casi usa this
  - Chiamo spesso un metodo da un altro con lo stesso nome ma diversa segnatura (delego ad altro metodo simile)