Programmazione Orientata agli Oggetti in Linguaggio Java

Ereditarietà e Polimorfismo: Aspetti Metodologici

versione 1.1

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Estensione e Associazione
- Linee Guida
- Alcuni Esempi

Estensione e Associazione

- Nella programmazione a oggetti
  - ⇨due meccanismi principali di collaborazione tra gli oggetti
- L'Associazione
  - ⇨un oggetto mantiene tra le sue proprietà riferimenti ad altri oggetti
- L'Estensione
  - ⇨un oggetto è figlio di un oggetto padre in una gerarchia di ereditarietà

Estensione e Associazione

- I due meccanismi
  - ⇨sembrano molto diversi l'uno dall'altro
  - ⇨il secondo (estensione) sembra molto più potente del primo
- In realtà
  - ⇨vista la semantica, i due meccanismi sono molto simili tra di loro
  - ⇨si basano entrambi su forme di associazione (super nel caso dell'estensione)

Estensione e Associazione

- Pregi dell'estensione
  - ⇨eredità delle funzionalità: l'oggetto figlio può automaticamente rispondere a messaggi che riguardano metodi degli oggetti padre
  - ⇨ereditarietà del tipo: l'oggetto figlio eredita l'interfaccia e quindi il tipo degli oggetti padre
- Attenzione
  - ⇨l'ereditarietà del tipo e il polimorfismo si possono ottenere anche solo con le interface

Estensione e Associazione

- Difetti dell'estensione
  - ⇨molto spesso costringe a violare l'incapsulamento (con protected)
  - ⇨crea una forma di accoppiamento molto forte tra le implementazioni delle classi
  - ⇨infatti la sottoclasse spesso utilizza particolari dell'implementazione della superclasse (es: proprietà)

Estensione e Associazione

- Difetti dell'estensione (continua)
  - ⇨complica alcuni aspetti della scrittura del codice (es: utilizzo dei costruttori con argomenti)
  - ⇨richiede attenzione nell'utilizzo dell'overriding e del binding dinamico (es: bisogna fare attenzione a non sovrascrivere metodi per cui questo è inopportuno; es: verifica password)

Estensione e Associazione

- Difetti dell'associazione
  - ⇨non consente l'ereditarietà automatica dell'implementazione e del tipo
- Pregi dell'associazione
  - ⇨può essere utilizzata per simulare gli stessi effetti dell'ereditarietà
  - ⇨è più semplice da usare
  - ⇨introduce accoppiamento minore e non viola l'incapsulamento

Estensione e Associazione

- Riassumendo
  - ⇨utilizzando l'associazione, l'accoppiamento tra le classi resta basso (le due classi interagiscono solo attraverso l'interfaccia)
  - ⇨utilizzando l'estensione, l'accoppiamento è tipicamente più alto
  - ⇨il legame con la superclasse è decisamente più forte che nell'associazione

Estensione e Associazione

- Esempio
  - ⇨posso facilmente sostituire un'implementazione con un'altra (es: Giorno in associazione con java.util.ArrayList oppure LinkedList >>)
  - ⇨non è facile sostituire la superclasse; l'estensione crea sempre un legame tra due implementazioni

Estensione e Associazione

- Avendo entrambe pregi e difetti
  - ⇨non è sempre opportuno utilizzare l'ereditarietà
  - ⇨in alcuni casi è opportuno privilegiare l'associazione
- Nel seguito
  - ⇨stabiliamo delle linee guida
  - ⇨e vediamo alcuni esempi

Linee Guida

ATTENZIONE
alle linee guida per l'uso dell'ereditarietà

- Linea guida n. 1
  - ⇨privilegiare sempre l'associazione rispetto all'estensione
  - ⇨limitare l'utilizzo dell'estensione mantiene il codice meglio organizzato e più facilmente manutenibile
  - ⇨limita errori dovuti all'utilizzo scorretto dell'ereditarietà e del polimorfismo (meccanismi delicati)

Linee Guida

- Linea guida n. 2
  - ⇨per utilizzare l'ereditarietà, bisogna che le classi coinvolte rappresentino concetti della realtà in relazione "is-a"
  - ⇨ovvero gli oggetti della sottoclasse devono essere casi particolari di oggetti della superclasse
  - ⇨in questo caso la gerarchia è una forma di rappresentazione naturale della realtà ed è ragionevole la "sostituibilità"

Linee Guida

- Linea guida n. 3
  - ⇨oltre alla linea guida n. 2, usare le gerarchie solo nei casi in cui è necessario "l'upcast" degli oggetti della sottoclasse
  - ⇨es: utilizzo di collezioni miste (oggetti della sottoclasse, oggetti della superclasse)
  - ⇨es: utilizzo del tipo della superclasse per chiamare metodi polimorfi

Linee Guida

- Linea guida n. 4
  - ⇨in ogni caso, verificare se i problemi di upcast e polimorfismo possono essere risolti utilizzando solo interfacce, senza estensione
  - ⇨utilizzare l'estensione solo come meccanismo per mettere a fattor comune codice delle varie implementazioni
  - ⇨ma solo quando il beneficio è concreto; es: ImpegnoAstratto

Alcuni Esempi

- Primo esempio
  - ⇨Azienda e Indirizzo
  - ⇨supponiamo di aver definito una classe Indirizzo con via, CAP, città
  - ⇨vogliamo costruire una classe i cui oggetti rappresentano aziende
  - ⇨le aziende hanno un indirizzo
- Tra Azienda e Indirizzo
  - ⇨estensione o associazione ?

Alcuni Esempi

❍ Associazione

```
public class Indirizzo {
  private String via, CAP, citta;
  public String getVia {
     return this.via;
  }
  public String getCAP {
     return this.CAP;
  }
  public String getCitta {
     return this.citta;
  }
  ...
}
```

```
public class Azienda {
  private String nome;
  private ArrayList dipendenti;
  private Indirizzo indirizzo;
  public String getVia() {
     return this.indirizzo.getVia();
  }
  public String getCAP() {
     return this.indirizzo.getCAP();
  }
  public String getCitta() {
     return this.indirizzo.getCitta();
  }
  public String getNome() {
     return this.nome;
  }
  public int getNumeroDipendenti() {
     return this.dipendendi.size();
  }
  ...
}
```


Alcuni Esempi

❍ Estensione

```
public class Indirizzo {
  private String via, CAP, citta;
  public String getVia {
     return this.via;
  }
  public String getCAP {
     return this.CAP;
  }
  public String getCitta {
     return this.citta;
  }
  ...
}
```

```
public class Azienda extends Indirizzo {
  private String nome;
  private ArrayList dipendenti;

  public String getNome() {
     return this.nome;
  }
  public int getNumeroDipendenti() {
     return this.dipendendi.size();
  }
  ...
}
```

Alcuni Esempi

- Apparentemente
  - ⇨sembra conveniente il secondo caso
  - ⇨riutilizzo più codice, ne scrivo di meno
- Ma, a pensarci bene...
  - ⇨questo caso viola la linea guida n. 2
  - ⇨un'Azienda NON è un indirizzo nella realtà
  - ⇨es: un'Azienda può avere vari indirizzi
  - ⇨quindi la soluzione migliore in questo caso è l'associazione

Alcuni Esempi

- Secondo esempio
  - ⇨Circonferenza e Punto
  - ⇨supponiamo di voler rappresentare una circonferenza attraverso il suo raggio e le coordinate del suo centro
  - ⇨il centro è un punto e per questo usiamo una classe Punto
- Tra Circonferenza e Punto
  - ⇨estensione o associazione ?

Alcuni Esempi

❍ Associazione

```
public class Punto {
   private double x, y;
   public Punto (double x, double y) {
      this.x = x;
      this.y = y;
   }
   public double getX() {
      return this.x;
   }
   public double getY() {
      return this.y;
   }
}
```

```
public class Circonferenza {
   private double raggio;
   private Punto centro;
   public Circonferenza(double r, double x
                        double y) {
      this.raggio = r;
      this.centro = new Punto(x, y);
   }
   public double getX() {
      return this.centro.getX();
   }
   public double getY() {
      return this.centro.getY();
   }
   public double getRaggio() {
      return this.raggio;
   }
}
```


Alcuni Esempi

❍ Estensione

```
public class Punto {
   protected double x, y;
   public Punto (double x, double y) {
      this.x = x;
      this.y = y;
   }
   public double getX() {
      return this.x;
   }
   public double getY() {
      return this.y;
   }
}
```

```
public class Circonferenza extends Punto {
   private double raggio;
   public Circonferenza(double r, double x
                        double y) {
      super(x, y);
      this.raggio = r;
   }
   public double getRaggio() {
      return this.raggio;
   }
}
```

Alcuni Esempi

- Apparentemente, anche in questo caso
  - ⇨sembra conveniente l'estensione
- In effetti
  - ⇨la linea guida n. 2 è in questo caso discutibile
  - ⇨una Circonferenza in alcuni casi è un Punto (se il raggio è uguale a 0)
  - ⇨quindi il criterio non è del tutto chiaramente applicabile

Alcuni Esempi

- Ma...
  - ⇨la soluzione con estensione viola la linea guida n. 3
  - ⇨è del tutto inopportuno utilizzare oggetti di tipo Circonferenza al posto di punti
  - ⇨si tratta di cose diverse: figure geometriche bidimensionali e figure monodimensionali
  - ⇨quindi non farò mai l'upcast
  - ⇨quindi è preferibile l'associazione

Alcuni Esempi

- Terzo esempio
  - ⇨Impiegato e Persona
  - ⇨supponiamo di voler rappresentare i dati degli impiegati di un'azienda
  - ⇨gli impiegati hanno nome, cognome, CF e data di nascita
  - ⇨cosa che hanno tutte le persone
  - ⇨infatti, nella realtà, un Impiegato è una (is-a) Persona

Alcuni Esempi

- Vale la pena di adottare l'estensione ?
  - ⇨definire la classe Persona
  - ⇨definire Impiegato per estensione di Persona
- In questo caso
  - ⇨conviene farsi guidare dalle linee guida
- In particolare
  - ⇨la linea guida n. 2 è a posto
  - ⇨restano la 3 e la 4

Alcuni Esempi

- La domanda da farsi
  - ⇨esistono altri oggetti che sono Persone nell'applicazione (es: Dirigenti) ?
  - ⇨ci sono circostanze in cui può essere utile fare l'upcast da Impiegato a Persona ?
  - ⇨è possibile risolvere il problema con un'interfaccia Persona; ovvero: quali sono i reali vantaggi collegati all'estensione ?

Alcuni Esempi

- Un ultimo esempio
  - ⇨le estrazioni del Lotto
  - ⇨è necessario rappresentare le estrazioni (sequenze di sei numeri distinti tra 1 e 90)
  - ⇨e le giocate (sequenze di lunghezza variabile di numeri distinti tra 1 e 90)
  - ⇨nei due casi ci sono operazioni ricorrenti (es: verifica della correttezza dei numeri della sequenza)

Alcuni Esempi

- Una prima soluzione
  - ⇨definisco la classe Estrazione con i metodi relativi alle verifiche dei numeri
  - ⇨definisco la classe Giocata duplicando il codice necessario per le verifiche
- Soluzione scadente
  - ⇨il codice non andrebbe mai duplicato

Alcuni Esempi

- Una seconda soluzione
  - ⇨definisco la classe Estrazione con i metodi relativi alle verifiche dei numeri
  - ⇨definisco la classe Giocata per estensione di Estrazione, in modo da ereditarne i metodi
- Le linee guida
  - ⇨la 2 è discutibile (Giocata is-a Estrazione ?)
  - ⇨ma la 3 taglia la testa al toro: non ci sarà mai l'upcast

Alcuni Esempi

- La soluzione migliore
  - ⇨definisco la classe ListaDelLotto per rappresentare sequenze di numeri distinti tra 1 e 90 con dimensioni variabili, con tutti i metodi per le verifiche (posso costruirla per associazione con ArrayList)
  - ⇨definisco Estrazione per associazione con ListaDelLotto
  - ⇨definisco Giocata per associazione con ListaDelLotto

Alcuni Esempi

- In questo modo
  - ⇨evito di duplicare il codice (il codice è scritto una volta sola in ListaDelLotto e viene riutilizzato da Estrazione e da Giocata)
  - ⇨evito di introdurre una gerarchia di dubbio utilizzo

Riassumendo

- Estensione e Associazione
- Linee Guida
- Alcuni Esempi

Termini della Licenza