Programmazione Orientata agli Oggetti in Linguaggio Java

Ereditarietà e Polimorfismo: Introduzione

versione 1.2

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

Sommario

- Panoramica
- Perché Esiste l'Ereditarietà
- Un Esempio di Applicazione
- Un Ulteriore Esempio
- Intuizione sulla Semantica
- Avvertenze per l'Uso

Panoramica

- Finora
  - una presentazione completa delle tecniche della programmazione con gli oggetti
  - come creare, comporre ed utilizzare gli oggetti
- In vari casi
  - abbiamo “sfiorato” l’ereditarietà ed il polimorfismo
  - gestendolo il modo meccanico

Panoramica

- Esempi
  - java.lang.Object e le collezioni (java.util.ArrayList, java.util.LinkedList, java.util.List)
  - java.lang.Exception, java.lang.RuntimeException
  - le regole di JUnit

Panoramica

- In questa parte del corso
  - ⇨approfondiamo tutti i dettagli relativi all'ereditarietà ed al polimorfismo
- I principali concetti
  - ⇨sintassi dell'estensione ("extends")
  - ⇨semantica dell'estensione
  - ⇨utilizzo delle interfacce ("interface", "implements")
  - ⇨semantica del polimorfismo

Panoramica

- Terminologia dell'estensione
  - ⇨gli oggetti possono costituire "gerarchie"
  - ⇨con "classi di base", o "classi padre", o "superclassi"
  - ⇨e "classi derivate", o "classi figlie" o "sottoclassi"
  - ⇨la sottoclasse "estende" la superclasse
  - ⇨una stessa classe può essere padre di una classe X e figlia di una classe Y

Panoramica

- Attenzione al termine “superclasse”
  - ⇨non vuol dire “classe super”, ovvero classe dotata di maggiori funzionalità
  - ⇨ma “classe che sta sopra nella gerarchia”
- Viceversa
  - ⇨tipicamente sono gli oggetti delle sottoclassi ad avere maggiori funzionalità (attributi e metodi) rispetto a quelli delle superclassi

Panoramica

- Ricapitoliamo le regole viste finora
- Regola meccanica n. 1
  - ⇨gli oggetti della sottoclasse ereditano tutte le proprietà ed i metodi degli oggetti della superclasse
  - ⇨possono estenderli aggiungendo ulteriori proprietà e ulteriori metodi
  - ⇨es: Object e Esame

Panoramica

- Regola meccanica n. 2
  - ⇨gli oggetti della sottoclasse possono essere considerati anche oggetti della superclasse
  - ⇨di conseguenza, gli oggetti della sottoclasse sono sostituibili a quelli della superclasse (possono essere utilizzati dove servono gli altri)
  - ⇨es: Object e = new Esame();
  - ⇨NOTA: non vale il viceversa

Perché Esiste l'Ereditarietà

- Perché questi meccanismi ?
  - ⇨perchè rispecchiano aspetti della realtà
- Infatti
  - ⇨gli oggetti software sono fatti per rappresentare oggetti del mondo reale
  - ⇨e gli oggetti del mondo reale sono spesso organizzati in tassonomie

Perché Esiste l'Ereditarietà

- Un esempio di tassonomia
  - ⇨i “Big Jim” della Mattel
  - ⇨esempio vicino alla realtà dei componenti nelle applicazioni OO
- Big Jim
  - ⇨giocattolo commercializzato dalla Mattel tra il 1972 e il 1984 in molte varianti
  - ⇨con diversi accessori e funzionalità

Perché Esiste l'Ereditarietà

- Due modi per leggere la tassonomia
- Discendendo la tassonomia
  - ⇨rapporto di “specializzazione” tra le classi
  - ⇨scendendo verso il basso le classi diventano mano mano più specifiche
  - ⇨gli oggetti delle sottoclassi conservano le caratteristiche degli oggetti delle classi più in alto e ne aggiungono altre
  - ⇨es: Big Jim cacciatore artico

Perché Esiste l'Ereditarietà

- Risalendo la tassonomia
  - ⇨rapporto di tipo “is-a” tra le classi
  - ⇨un esemplare di una sottoclasse può essere a tutti gli effetti considerato anche un esemplare delle sue superclassi >> sostituibilità
  - ⇨es: Big Jim cacciatore artico è anche a tutti gli effetti un Big Jim e un Big Jim cacciatore

Perché Esiste l'Ereditarietà

- Nei linguaggi a oggetti
  - ⇨il meccanismo di estensione cerca di riprodurre queste caratteristiche
  - ⇨con due obiettivi
  - ⇨fornire le stesse funzionalità
  - ⇨semplificare la scrittura del codice
- In particolare
  - ⇨ci sono due meccanismi diversi offerti dal linguaggio

Perché Esiste l'Ereditarietà

- Ereditarietà delle funzionalità
  - ⇨o dell'implementazione
  - ⇨la sottoclasse eredita l'implementazione della superclasse (codice di proprietà e metodi)
  - ⇨riproduce il fenomeno di specializzazione semplificando la scrittura delle sottoclassi
  - ⇨che vengono descritte solo "per differenza" rispetto alle superclassi

Perché Esiste l'Ereditarietà

- Infatti
  - ⇨se dovessi creare la classe degli oggetti “Big Jim cacciatore artico”, potrei evitare di ridefinire il metodo “colpo di karate”, ereditato dalla classe Big Jim
  - ⇨erediterei la proprietà “fucile” e il metodo “spara” dalla classe Big Jim cacciatore
  - ⇨e potrei ridefinirle per specializzarle al contesto artico

Perché Esiste l'Ereditarietà

- Ereditarietà del tipo
  - ⇨o ereditarietà dell’interfaccia
  - ⇨la sottoclasse eredita l’interfaccia della superclasse e quindi ne eredita il ruolo, e in ultima analisi, il tipo
  - ⇨riproduce il fenomeno di sostituibilità
  - ⇨es: posso usare un oggetto di tipo Big Jim cacciatore artico dovunque sia necessario un oggetto di tipo Big Jim (>>)

Un Esempio di Applicazione

- Per descrivere questi concetti
  - ⇨un nuovo progetto di riferimento
- Gestione appuntamenti
  - ⇨un'applicazione per gestire archivi di appuntamenti
  - ⇨capace di salvare persistentemente l'archivio su disco
  - ⇨varie tipologie di appuntamenti (per ora: riunione e lezione, ma è estensibile)

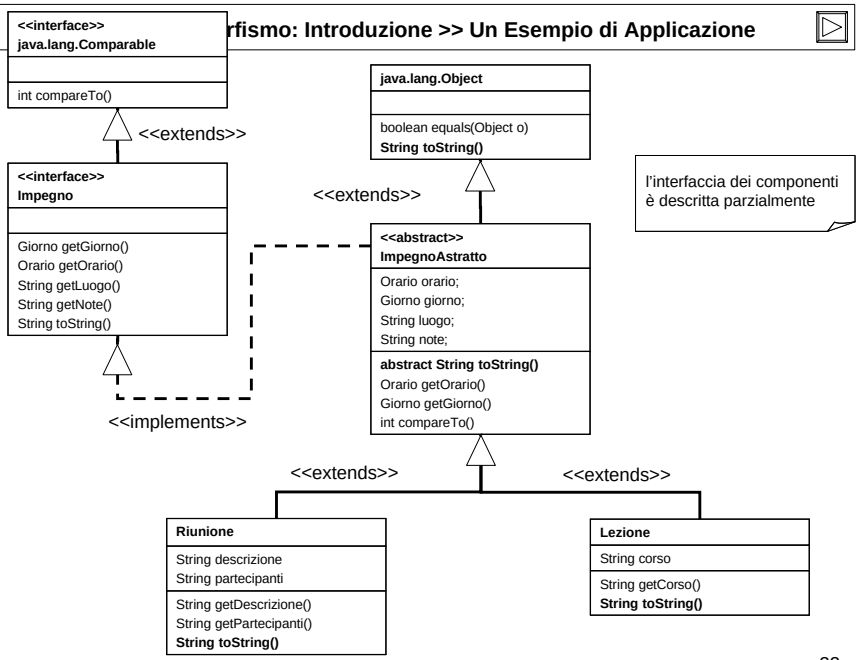
Un Esempio di Applicazione

- I casi d'uso
  - ⇨"Utente crea nuova agenda"
  - ⇨"Utente carica agenda da file"
  - ⇨"Utente visualizza appuntamenti per giorno"
  - ⇨"Utente aggiunge appuntamento"
  - ⇨"Utente elimina appuntamento"
  - ⇨"Utente salva agenda su file"

Un Esempio di Applicazione

- Caratteristiche del progetto
  - ⇨maggiore complessità dei precedenti: 3 package, 16 classi (inclusi i test)
  - ⇨circa 1000 linee di codice
  - ⇨varie funzionalità avanzate: utilizzo delle date, gestione del formato dei file, ordinamento, utilizzo di dizionari associativi ecc.

>> it.unibas.appuntamenti
>> varie\agenda1.txt



Un Esempio di Applicazione

- Dal punto di vista dell'ereditarietà
  - ⇨il progetto esemplifica entrambi gli aspetti
  - ⇨eredità delle funzionalità: sia Lezione che Riunione ereditano le caratteristiche di ImpegnoAstratto
  - ⇨ereditarietà del tipo e polimorfismo, attraverso la definizione dell'interfaccia Impegno

Un Ulteriore Esempio

- Ma
  - ⇨il secondo tipo di ereditarietà ha senso anche senza il primo
- Un ulteriore esempio per dimostrarlo
  - ⇨"Volpi e Conigli"
  - ⇨una variante del "Gioco della Vita"
  - ⇨un esempio di simulazione della vita di volpi e conigli su un campo di gioco

Un Ulteriore Esempio

- Le regole
  - ⇨si gioca su una scacchiera di dimensioni variabili
  - ⇨viene creata una popolazione iniziale di volpi e di conigli
  - ⇨ad ogni passo viene creata una configurazione successiva della scacchiera
  - ⇨sulla base delle azioni di volpi e conigli

Un Ulteriore Esempio

- I conigli
  - ⇨corrono (si muovono) sulla scacchiera
  - ⇨ad ogni passo invecchiano
  - ⇨possono morire di vecchiaia
- Le volpi
  - ⇨vanno a caccia di conigli; se trovano un coniglio a tiro (ovvero nelle celle adiacenti alla cella della volpe) lo mangiano
  - ⇨altrimenti restano ferme e il loro livello di fame cresce
  - ⇨possono morire di fame

Un Ulteriore Esempio

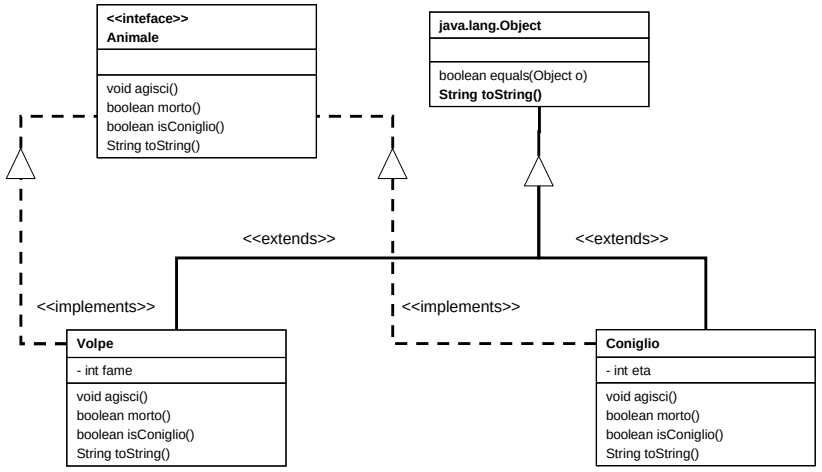
>> it.unibas.volpieconigli
>> config.properties

- La logica del gioco
  - verificare se dopo n passi della simulazione sopravvivono solo volpi o solo conigli
  - anche se, alla fine, comunque moriranno tutti
- Alcune caratteristiche interessanti
  - anche in questo caso varie classi (14) e circa 1000 linee di codice
  - passaggio dei parametri al main
  - utilizzo di file di configurazione esterni

Un Ulteriore Esempio

- L'ereditarietà
  - è principalmente ereditarietà di tipo
  - basata sull'utilizzo dell'interface Animale, che descrive i metodi che un oggetto deve fornire per poter appartenere al gioco
- Attenzione
  - c'è sempre e comunque anche una forma implicita di ereditarietà delle funzionalità per via della presenza della classe Object

Un Ulteriore Esempio



Intuizione sulla Semantica

- Obiettivo di questo ciclo di lezioni
  - ⇨descrivere nei dettagli la sintassi e la semantica dell'estensione e dell'ereditarietà
  - ⇨e le sue applicazioni avanzate (polimorfismo)
- L'intuizione fondamentale
  - ⇨il funzionamento dell'ereditarietà è basato su un meccanismo fondamentale
  - ⇨la creazione di "associazioni gerarchiche di oggetti"

Intuizione sulla Semantica

- Associazione gerarchica di oggetti
  - ⇨la creazione di un oggetto di una sottoclasse scatena sempre automaticamente la creazione di un oggetto della superclasse
  - ⇨i due oggetti sono in associazione
  - ⇨questo processo si ripete lungo la gerarchia
  - ⇨al termine, tipicamente è stato creato un' associazione di oggetti e non un oggetto solo

Intuizione sulla Semantica

- Esempio
  - ⇨creare un oggetto di tipo Riunione scatena la creazione di un oggetto di tipo ImpegnoAstratto
  - ⇨questo scatena la creazione di un oggetto di tipo Object
  - ⇨un gruppo di tre oggetti in associazione tra di loro, che collaborano per eseguire i loro compiti

Intuizione sulla Semantica

- In effetti
  - la semantica dell'ereditarietà e del polimorfismo è basata su questo meccanismo
  - collaborando, gli oggetti sono in grado di fornire le funzionalità di ereditarietà
  - e di comportarsi in modo "polimorfo"

Avvertenze per l'Uso

- Nota
  - i programmatori inesperti tendono ad abusare nell'uso dell'ereditarietà
  - per via dell'effetto "novità"
- Una frase ricorrente su questo argomento
  - "*un programmatore inesperto che impara l'ereditarietà ha in mano un martello di cui si innamora al punto che comincia a vedere chiodi dappertutto*"

Avvertenze per l'Uso

- Viceversa
  - ⇨come tutti i meccanismi del linguaggio ha vantaggi e svantaggi
  - ⇨deve essere utilizzata in modo controllato
  - ⇨e rispettando specifiche linee guida
  - ⇨nel seguito verranno discussi anche questi aspetti

Riassumendo

- Panoramica
- Perché Esiste l'Ereditarietà
- Un Esempio di Applicazione
- Un Ulteriore Esempio
- Intuizione sulla Semantica
- Avvertenze per l'Uso

Ringraziamenti

- L'idea dell'applicazione "Volpi e conigli" è ispirata all'applicazione "Foxes and Rabbits" sviluppata da Barnes e Kolling come corredo al libro "Objects First with Java: A Practical Introduction using BlueJ"

Termini della Licenza