Fondamenti di Informatica T-1

Metodi polimorfi: upcast e downcast

Tutor:
Angelo Feraudo
angelo.feraudo2@unibo.it

a.a. 2017/2018

Metodo polimorfo

Un metodo si dice **polimorfo** quando è in grado di adattare il suo comportamento allo specifico oggetto su cui deve operare

In Java, la possibilità di usare variabili di un tipo classe, ad esempio Animale, per referenziare oggetti di classi più specifiche, ad esempio Cane, introduce in astratto la possibilità di avere polimorfismo

Esempio: metodo polimorfo (1)

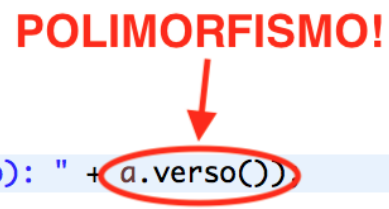
```
public class Animale {

    protected String nome;

    public Animale(String nome) {
        this.nome = nome;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String verso() {
        String s = "";
        s += "Sono l'animale " + nome + " con un verso qualsiasi..";
        return s;
    }


}
```

Esempio: metodo polimorfo (2)

```
public class Cane extends Animale{

    public Cane(String nome) {
        super(nome);
    }

    public String verso() {
        String s = "";
        s += nome + " BAU BAU";
        return s;
    }
    public String ringhia() {
        String s = "";
        s += "grrr";
        return s;
    }
}
```

Esempio: metodo polimorfo (3)

```
public static void main(String args[]) {

    Animale a = new Animale("Ezio");
    Cane c = new Cane("Pluto");
    System.out.println(a.verso());
    System.out.println("Cane: " + c.verso());
    a = c;
    System.out.println("Animale (dopo assegnamento): " + a.verso());

}
```



Il metodo **verso()** è polimorfo:
poichè **a** in questo momento sta referenziando un `Cane`, viene chiamato il metodo `verso()` di `Cane`
Infatti l'output sarà

```
Sono l'animale Ezio con un verso qualsiasi..
Cane: Pluto BAU BAU
Animale (dopo assegnamento): Pluto BAU BAU
```

Polimorfismo: breve introduzione

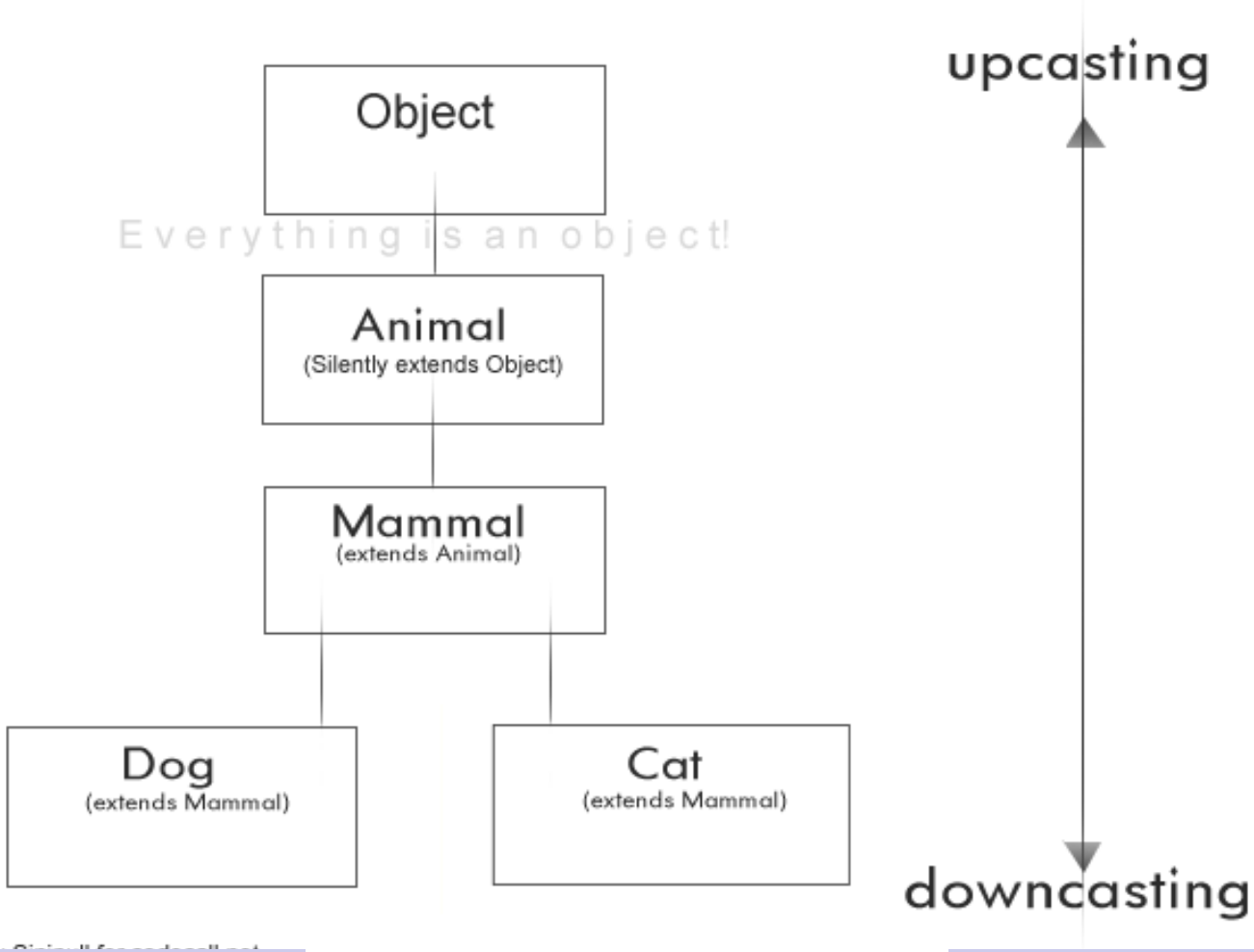
Polimorfismo letteralmente significa "prendere molte forme".
Tradotto in Java:

una variabile di un tipo classe può "prendere la forma", ossia referenziare oggetti, di ognuna delle sue sottoclassi

- Un oggetto di una sottoclasse può sostituire un oggetto della superclasse: "Un Cane è un Animale" (supponendo di avere una classe Cane che eredita da una classe Animale)
- Il contrario non è vero. Non si può sostituire un elemento di una sottoclasse con uno della superclasse "Un Animale non è un Cane"

Downcast e Upcast

- **Upcasting** è un casting verso una superclasse
- **Downcasting** è un casting verso una sottoclasse



Esempio: Downcast e Upcast

Upcast

```
Animale animal = new Cane("Pluto");
```

Downcast

```
Cane castDog = (Cane) animal;
```

N.B. Nel caso scrivessimo:

```
Animale animal = new Animale("Pluto");
Cane notADog = (Cane) animal;
```

La JVM scatenerebbe un'eccezione (ClassCastException). Questo perchè il tipo di `animal` a runtime è **Animale**, quindi quando la JVM cerca di fare il cast nota che `animal` non è realmente un `Cane`

Esempio: Downcast e Upcast(2)

Per risolvere questo problema basta fare un controllo sul tipo tramite l'operatore **instanceof**

```
Animale[] animali = new Animale[10];
// inizializzazione del vettore animali
// ...
// recupero dell'animale i-esimo:
Animale animal = animali[i]; /* Forse un cane?  Forse
un gatto?  Forse un altro animale?  */
if (animal instanceof Cane){
     Cane castDog = (Cane) animal;
}
```

Esercizio 1 (1)

Scrivere un programma che crei le seguenti classi:

- La classe **Piastrella** che fornisce un solo metodo **getArea()** e una variabile **lato** (reale).
- Si forniscano quindi tre diverse implementazioni di Piastrella,ovvero **PiastrellaQuadrata** (che ha come parametro solo il lato), **PiastrellaTriangolare** e **PiastrellaRettangolare** (che hanno come parametri la base ed il lato).
- Queste classi, oltre ad i metodi getter e setter, devono fornire un'opportuna implementazione di **getArea()** dichiarato nella classe Piastrella.

Esercizio 1 (2)

- La classe **Pavimento** è formata da un tipo di piastrella ed un intero rappresentante il numero di piastrelle di cui questo è composto.
- Oltre ad i metodi getter e setter si scriva il metodo **getSuperficie()**, il quale dà in output la superficie totale del pavimento.
- Si scriva inoltre un main di prova per testare il corretto funzionamento delle classi

Esercizio 2 (1)

Scrivere un programma che definisca opportune classi, costruttori e metodi per la gestione di eventi turistici. I dati gestiti dal programma sono relativi a:

- **Eventi**, caratterizzati da codice univoco generato automaticamente dal sistema, località, descrizione e vettore di feedback (booleani che rappresentano se un utente consiglia la meta).
- Gli eventi si dividono in: **mostre** (caratterizzate da mese di inizio e mese di fine, codice: M+progressivo), **tour guidati** (con elenco dei giorni del mese in cui si svolgono, numero di posti disponibili, codice: T+progressivo) e **spettacoli** (con indicazione se lo spettacolo è all'aperto, codice: S+progressivo).
- Si supponga che tali informazioni siano contenute in una classe **Turismo** in un vettore eventi, di massimo 100 elementi.

Esercizio 2 (2)

Si definiscano i seguenti metodi:

- Metodo **consigliato**, che indica se un evento è consigliato o meno (feedback positivi >feedback negativi).
- Metodo **controllaDisponibilità**, che dati una data nel formato gg/mm/aaaa, il numero di posti richiesti e una previsione meteo controlla se un evento è disponibile:
  - per le **mostre** si controlli che il mese richiesto sia compreso nell'intervallo di apertura della mostra;
  - per i **tour guidati**, che il numero di posti richiesti sia inferiore al numero di posti disponibili e che il giorno sia incluso nei giorni del mese in cui il tour viene svolto;
  - per gli **spettacoli**, si controlli se lo spettacolo è nei mesi estivi (Giugno, Luglio, Agosto), che il meteo sia sereno e il posto in cui si svolge sia all'aperto, altrimenti si controlli solo se lo spettacolo viene svolto al chiuso.

Esercizio 2 (3)

- Metodo **trovaEventi**, che, data una località, stampi a video le informazioni sugli eventi nella località scelta (si ridefinisca il metodo toString() della classe Evento e delle sottoclassi).
- Metodo **piuConsigliato**, che restituisca l'evento con il maggior numero di feedback positivi.
- Metodo **tourNonEsauriti**, che restituisca un vettore contenente i tour per i quali sono ancora disponibili posti.

Esercizio 2 (4)

Il programma principale deve, infine, svolgere i seguenti punti:

- Creare un'istanza di Turismo e popolare il vettore con varie istanze.
- Invocare il metodo **trovaEventi**.
- Invocare il metodo **piuConsigliato**, mostrando a video l'evento restituito e l'informazione sulla disponibilità di tale evento per una data inserita dall'utente, per un gruppo di 7 persone e con previsione di tempo nuvoloso.
- Invocare il metodo **tourNonEsauriti**, mostrando a video i dati dei tour restituiti dal metodo.