L7 - Interfacce - Avanzati

Java · polimorfismo via interfaccia, array eterogenei, piu' interfacce

Materiale didattico di Moreno Sardella · morenosardella.com

Prima di iniziare. Questi esercizi sono piu' impegnativi: combinano piu' concetti e si avvicinano al livello della verifica. Progetta prima su carta (classi, metodi, dati), poi scrivi il codice. Le soluzioni in fondo mostrano *una* via corretta: la tua puo' differire, conta che la logica funzioni e che il codice compili.

Esercizi

Esercizio 1 — Array di Pagabile

POLIMORFISMO

Crea due classi diverse che implementano **Pagabile** (es. Bolletta e Stipendio, con getImporto() calcolato diversamente). Mettile in un array **Pagabile[]** e stampa tipo, id e importo di ciascuna con un solo ciclo.

```
Pagabile[] v = { new Bolletta(...), new Stipendio(...) };
for (Pagabile p : v) // usa getTipo/getId/getImporto
```

Esercizio 2 — Totale polimorfico

SOMMA SULL'INTERFACCIA

Scrivi **static double totale(Pagabile[] v)** che somma i getImporto() di tutti gli elementi, indipendentemente dalla classe concreta.

```
public static double totale(Pagabile[] v) {
    double t = 0;
    // for-each + getImporto()
}
```

Esercizio 3 — Due interfacce insieme

IMPLEMENTS MULTIPLO

Crea **Confrontabile** con **int confronta(Object o)** e fai implementare a **Prodotto** sia **Pagabile** sia **Confrontabile** (confronto sul prezzo).

```
class Prodotto implements Pagabile, Confrontabile { /* ... */ }
```

Esercizio 4 — Ordinare per importo

INTERFACCIA + ALGORITMO

Dato un **Pagabile[]**, ordinalo per importo crescente (bubble sort) usando solo i metodi dell'interfaccia. Stampa la lista ordinata.

```
if (v[j].getImporto() &gt; v[j+1].getImporto()) // scambia
```

Soluzioni

Confronta solo dopo aver provato da solo. Esistono soluzioni equivalenti: conta la logica, non la copia esatta.

Soluzione 1

```
class Bolletta implements Pagabile {
    private String id; private double imp;
    Bolletta(String id,double i){this.id=id;imp=i;}
    public double getImporto(){return imp;}
    public String getTipo(){return "Bolletta";}
    public String getId(){return id;}
}
class Stipendio implements Pagabile {
    private String id; private double base; private int mesi;
    Stipendio(String id,double b,int m){this.id=id;base=b;mesi=m;}
    public double getImporto(){return base * mesi;}
    public String getTipo(){return "Stipendio";}
    public String getId(){return id;}
}
// for (Pagabile p : v) System.out.println(p.getTipo()+";"+p.getId()+";"+p.getImporto());
```

Soluzione 2

```
public static double totale(Pagabile[] v) {
    double t = 0;
    for (Pagabile p : v) t += p.getImporto();
    return t;
}
```

Soluzione 3

```
interface Confrontabile { int confronta(Object o); }
class Prodotto implements Pagabile, Confrontabile {
    private String id; private double prezzo;
    public double getImporto(){return prezzo;}
    public String getTipo(){return "Prodotto";}
    public String getId(){return id;}
    public int confronta(Object o){
        Prodotto p = (Prodotto) o;
        return Double.compare(this.prezzo, p.prezzo);
    }
}
```

Soluzione 4

```
for (int i=0;i&lt;v.length-1;i++)
  for (int j=0;j&lt;v.length-1-i;j++)
    if (v[j].getImporto() &gt; v[j+1].getImporto()) {
        Pagabile t=v[j]; v[j]=v[j+1]; v[j+1]=t;
    }
for (Pagabile p : v) System.out.println(p.getId()+": "+p.getImporto());
```