1 Teoria

- Che cos'è l'ereditarietà in Java? A cosa serve?
- Come si dichiara una classe che estende un'altra classe?
- Qual è la differenza tra classe base (superclasse) e classe derivata (sottoclasse)?
- Cos'è il metodo `super` e quando si usa?
- Che cos'è il polimorfismo in Java? Come si manifesta?
- Cos'è l'overriding di un metodo? Come si differenzia dall'overloading?
- È possibile chiamare un metodo della superclasse da una sottoclasse? Come?
- Cos'è il binding dinamico (late binding) nei metodi?
- Che ruolo hanno i metodi astratti e le classi astratte nell'ereditarietà?
- Cos'è un'interfaccia in Java e come differisce dall'ereditarietà delle classi?

2 Ereditarietà e Polimorfismo

Esercizio 1: Libreria Digitale con ereditarietà multipla indiretta

Descrizione:

- Progettare un sistema per una libreria digitale.
- Esiste una classe astratta `ElementoBiblioteca` con:
  - Attributi: `titolo`, `anno_pubblicazione`
  - Metodo astratto: `descrizione()`
- Da essa derivano:
  - `Libro` (aggiunge autore, numero pagine)
  - `Rivista` (aggiunge numero, mese)
- Entrambe devono ereditare anche da un'interfaccia `Stampabile` (o classe astratta) con il metodo `stampa()`.
- Creare una classe `Ebook` che eredita da `Libro` e implementa anche `Stampabile`, ma con logica diversa dalla stampa cartacea.
- Implementare una funzione `stampa_info(elemento: ElementoBiblioteca)` che mostri polimorficamente le informazioni, indipendentemente dal tipo concreto.

Sfida extra: Se un `Ebook` viene trattato come `ElementoBiblioteca`, assicurarsi che la chiamata a `stampa()` esegua la versione corretta.

Esercizio 2: Sistema di pagamento polimorfico con override parziale

Descrizione:

- Creare una gerarchia di classi per un sistema di pagamenti:
  - `Pagamento` (classe astratta) con:
    - Attributo `importo`
    - Metodo astratto `esegui()`
    - Metodo concreto `riepilogo()` che mostra l'importo e chiama polimorficamente un metodo `dettagli_transazione()` (che può essere sovrascritto)
  - `PagamentoCarta` e `PagamentoBonifico` ereditano e implementano `esegui()` e `dettagli_transazione()`.
  - `PagamentoCriptovaluta` eredita da `Pagamento` ma non sovrascrive `dettagli_transazione()` (usa quello di default se presente).

Sfida extra: Nel `riepilogo()`, se una classe non ridefinisce `dettagli_transazione()`, far sì che venga mostrato un messaggio generico.

Esercizio 3: Zoo con metodi sovrascritti e chiamate alla superclasse

Descrizione:

- Creare una gerarchia:
  - `Animale` (con attributi `nome`, `specie`, metodo `verso()` e metodo `info()`).
  - `Mammifero` e `Uccello` ereditano da `Animale`.
  - `Pinguino` eredita da `Uccello` ma sovrascrive `verso()` e `info()`:
    - In `info()` deve chiamare prima la versione della superclasse, poi aggiungere dettagli extra.
- Creare `Zoo` con una lista di `Animale` e un metodo `mostra_tutti()` che chiami polimorficamente `info()` su ogni animale.

Sfida extra: Aggiungere una funzione che accetta un `Animale` e, se è un `Pinguino`, chiama un metodo aggiuntivo `nuota()` senza rompere l'uso polimorfico per gli altri animali.