Esercizio #1
Scrivere la classe **Veicolo** che ha i seguenti attributi:

- **targa** - identificativo univoco del veicolo
- **prezzo** - numero decimale con il prezzo base del veicolo
- **categoria** - Valori possibili [AUTO, MOTO, CAMION]

Le proprietà devono essere accessibili solo dalla classe **Veicolo** e deve essere possibile leggerle tramite opportuni metodi getter ma non modificarle, tranne la proprietà **categoria**, che può essere modificata da altre classi tramite un setter.

Istanziate 3 veicoli nel `main`, create un array con essi e iterate su ognuno stampando la targa di ogni veicolo.

Esercizio #2
Estendere il punto 1 aggiungendo 3 ulteriori classi:

- **Auto**
- **Moto**
- **Camion**

La classe **Veicolo** deve diventare astratta e le 3 classi sopra devono ereditare da essa.
Ogni classe concreta deve implementare il metodo `calcolaCosto()` in modo appropriato per il tipo di veicolo specifico. Ad esempio, il costo di un'**Auto** potrebbe essere calcolato in base al prezzo base con un'IVA del 22%, mentre il costo di un **Camion** potrebbe essere calcolato includendo un costo aggiuntivo per il peso massimo trasportabile.

Nel `main`, crea diverse istanze di veicoli (ad esempio auto, moto e camion), mettile in un unico array e calcola i costi totali utilizzando il polimorfismo per trattare tutti i veicoli in modo uniforme.

Esercizio #3
Estendere il punto 2 aggiungendo la classe **Bicicletta** con le seguenti caratteristiche:

- **marca**
- **modello**
- **numeroSerie**

Definire un'interfaccia con un metodo `checkDisponibilità()` comune a **Veicolo** e **Bicicletta**, che indica se il veicolo è disponibile o meno per il noleggio (basta un print in console).

Nel `main`, creare un array con veicoli e biciclette e chiamare il metodo `checkDisponibilità()` per ognuno di essi.

Questo esercizio è strutturato in modo da coprire gli stessi concetti (incapsulamento, ereditarietà, polimorfismo e interfacce), ma applicati a un contesto differente.