Esercizio 1

1. Si richiede di implementare un'applicazione Java per gestire forme geometriche come cerchi, quadrati e rettangoli utilizzando il concetto di ereditarietà.
2. **Requisiti:**
    - Creare una classe astratta **FormaGeometrica** che rappresenti una forma geometrica generica con metodi astratti per calcolare l'area e il perimetro.
    - Implementare le seguenti sottoclassi:
        - **Cerchio**: Deve estendere **FormaGeometrica** e avere un costruttore che accetta il raggio come parametro. Deve implementare i metodi astratti per calcolare l'area e il perimetro di un cerchio.
        - **Quadrato**: Deve estendere **FormaGeometrica** e avere un costruttore che accetta la lunghezza del lato come parametro. Deve implementare i metodi astratti per calcolare l'area e il perimetro di un quadrato.
        - **Rettangolo**: Deve estendere **FormaGeometrica** e avere un costruttore che accetta la base e l'altezza come parametri. Deve implementare i metodi astratti per calcolare l'area e il perimetro di un rettangolo.
    - Implementare una classe **Main** che crea istanze di diverse forme geometriche e ne stampa l'area e il perimetro.

Esercizio 2

Si richiede di implementare un'applicazione Java per gestire una squadra di calcio, con giocatori di campo e un portiere, utilizzando il concetto di ereditarietà.

Parte 1: Creazione di Calciatori e Portieri

1. Creare le seguenti classi:
    - **Calciatore**: Rappresenta un calciatore di campo con attributi come nome, partite giocate, gol segnati e valore in euro. Implementare i metodi di accesso e un metodo per calcolare la media gol a partita.
    - **Portiere**: Eredita dalla classe **Calciatore** e rappresenta un portiere con attributi aggiuntivi come gol subiti, rigori subiti e rigori parati. Implementare i metodi di accesso e un metodo per verificare se il portiere è ottimo in base ai rigori parati.
    - Classe principale **Main** per testare le funzionalità delle classi create.

Parte 2: Gestione della Squadra

2. Creare la classe **Squadra** per gestire la rosa dei giocatori:
    - Definire attributi come nome della squadra, anno di fondazione e un ArrayList per memorizzare i giocatori.
    - Implementare i seguenti metodi:
        - **inserisciCalciatore(Calciatore)**: Aggiunge un calciatore alla rosa della squadra.

- **cerca(String)**: Cerca un calciatore per nome e restituisce la posizione nell'ArrayList.
- **modifica(String)**: Cerca un calciatore per nome e restituisce l'oggetto corrispondente.
- **rimuoviCalciatore(int)**: Rimuove un calciatore dalla rosa in base all'indice.
- **rimuoviCalciatore(String)**: Cerca un calciatore per nome e lo rimuove dalla rosa.
- **stampaRosa()**: Stampa le informazioni di tutti i giocatori nella rosa, identificando il ruolo (calciatore di campo o portiere).
- **valoreInEuroSquadra()**: Restituisce il valore complessivo in euro di tutti i giocatori nella rosa.
- **calciatoreConPiuGol()**: Restituisce il calciatore con il maggior numero di gol segnati.

Richieste:

1. **Parte 1:**
    - Creare un oggetto di tipo **Calciatore** e uno di tipo **Portiere**.
    - Visualizzare le informazioni dei due giocatori.
    - Determinare quale dei due giocatori ha giocato più partite.
    - Verificare se il portiere è ottimo in base ai rigori parati.
    - Calcolare la media gol a partita del calciatore.
2. **Parte 2:**
    - Creare un'istanza della classe **Squadra**.
    - Inserire diversi calciatori e un portiere nella rosa della squadra.
    - Testare ciascuno dei metodi implementati nella classe **Squadra**.

Esercizio 3

Realizzare un'applicazione Java per gestire una flotta di veicoli utilizzando concetti di ereditarietà.

Parte 1: Creazione di Veicoli

1. Definire le seguenti classi astratte:
    - **Veicolo**: Rappresenta un veicolo generico con attributi come marca, modello e anno di produzione. Implementare i metodi di accesso.
    - **VeicoloTerrestre**: Eredita dalla classe **Veicolo** e rappresenta un veicolo terrestre con attributi aggiuntivi come numero di ruote e cilindrata del motore.
    - **VeicoloAereo**: Eredita dalla classe **Veicolo** e rappresenta un veicolo aereo con attributi aggiuntivi come numero di motori e autonomia di volo.

Parte 2: Gestione della Flotta

2. Implementare la classe **FlottaVeicoli** per gestire la flotta di veicoli:
    - Definire attributi come nome della flotta e un ArrayList per memorizzare i veicoli.
    - Implementare i seguenti metodi:
        - **aggiungiVeicolo(Veicolo)**: Aggiunge un veicolo alla flotta.
        - **cerca(String)**: Cerca un veicolo per marca e modello e restituisce la sua posizione nell'ArrayList.
        - **rimuoviVeicolo(int)**: Rimuove un veicolo dalla flotta in base all'indice.
        - **stampaFlotta()**: Stampa le informazioni di tutti i veicoli presenti nella flotta.
        - **calcolaNumeroRuote()**: Calcola il numero totale di ruote presenti nella flotta.
        - **calcolaNumeroMotori()**: Calcola il numero totale di motori presenti nella flotta.

Richieste:

1. **Parte 1:**
    - Creare almeno un oggetto di tipo **VeicoloTerrestre** e uno di tipo **VeicoloAereo**.
    - Visualizzare le informazioni dei due veicoli.
    - Cerca un veicolo nella flotta per marca e modello.
2. **Parte 2:**
    - Creare un'istanza della classe **FlottaVeicoli**.
    - Aggiungere diversi veicoli alla flotta.
    - Testare ciascuno dei metodi implementati nella classe **FlottaVeicoli**.

Opzionale

Un Elemento Multimediale è una Immagine, un Filmato o una registrazione Audio identificato da un titolo (una stringa non vuota).
Un elemento è riproducibile se ha una durata (un valore positivo di tipo int) e un metodo play().

Una registrazione Audio è riproducibile e ha associato anche un volume (un valore positivo di tipo int) e i metodi weaker() e louder() per regolarlo. Se riprodotta, ripete per un numero di volte pari alla durata la stampa del titolo concatenato a una sequenza di punti esclamativi di lunghezza pari al volume (una stampa per riga).

Un Filmato è riproducibile e ha associato un volume regolabile analogo a quello delle registrazioni audio e anche una luminosità (un valore positivo di tipo int) e i metodi brighter() e darker() per regolarla. Se riprodotta, ripete per un numero di volte pari alla durata la stampa del titolo concatenato a una sequenza di punti esclamativi di lunghezza pari al volume e poi a una sequenza di asterischi di lunghezza pari alla luminosità (una stampa per riga).

Una Immagine non è riproducibile, ma ha una luminosità regolabile analoga a quella dei filmati e un metodo show() che stampa il titolo concatenato a una sequenza di asterischi di lunghezza pari alla luminosità

Eseguire un oggetto multimediale significa invocarne il metodo show() se è un'immagine o il metodo play() se è riproducibile.

Organizzare opportunamente con classi astratte, interfacce e classi concrete il codice di un lettore multimediale che memorizza 5 elementi (creati con valori letti da tastiera) in un array e poi chiede ripetutamente all'utente quale oggetto eseguire (leggendo un intero da 1 a 5 oppure 0 per finire) e dopo ogni esecuzione fornisce la possibilità di regolarne eventuali parametri (volume / luminosità).