Esercizio 1: Gestione Veicoli (Ereditarietà e Interfacce)

Crea un programma che gestisce una flotta di veicoli.

1. Definisci un'interfaccia `Veicolo` con metodi come `avvia()`, `ferma()`, e `getTipo()`.
2. Crea una classe astratta `MezzoDiTrasporto` che implementa `Veicolo` e ha un attributo `targa`.
3. Implementa due classi concrete: `Auto` e `Moto`, ereditando da `MezzoDiTrasporto`.
4. Scrivi un programma che memorizza diversi veicoli in un array o una lista e permette di avviarli o fermarli.

Esercizio 2: Gestione Biblioteca (Liste ed Ereditarietà)

Crea un semplice sistema per gestire una biblioteca.

1. Definisci una classe astratta `MaterialeBibliotecario` con attributi `titolo` e `annoPubblicazione`.
2. Crea due classi che ereditano: `Libro` e `Rivista`. Aggiungi attributi specifici (es. `autore` per i libri, `numeroEdizione` per le riviste).
3. Implementa una classe `Biblioteca` che usa una `List<MaterialeBibliotecario>` per memorizzare i materiali.
4. Aggiungi metodi per aggiungere, rimuovere e visualizzare i materiali.

Esercizio 3: Gestione Animali (Ereditarietà e Interfacce)

Crea un programma che modella diversi animali e le loro caratteristiche.

Requisiti:

1. Crea un'interfaccia `Animale` con i metodi `emettiVerso()` e `muoviti()`.
2. Crea una classe astratta `AnimaleBase` che implementa `Animale` e ha un attributo `nome`.
3. Implementa due classi concrete `Cane` e `Gatto`, che ereditano da `AnimaleBase`.
4. Utilizza un array o una lista per memorizzare diversi animali e farli interagire.

Esercizio 4: Gestione Dipendenti (Ereditarietà e Liste)

Crea un sistema per gestire i dipendenti di un'azienda.

Requisiti:

1. Crea una classe astratta `Dipendente` con attributi `nome`, `stipendio` e un metodo astratto `calcolaStipendio()`.
2. Crea due classi `Impiegato` e `Manager`, che ereditano da `Dipendente`.
    - Un impiegato ha uno stipendio fisso.
    - Un manager ha uno stipendio fisso più un bonus.
3. Implementa una classe `Azienda` che memorizza i dipendenti in una lista e stampa gli stipendi totali.

Esercizio 5: Sistema di Prenotazione di Viaggi (Ereditarietà, Interfacce, Liste, Eccezioni)

Creiamo un sistema per gestire prenotazioni di diversi tipi di viaggi.

Requisiti:

1. Crea un'interfaccia `Prenotabile` con un metodo `prenota()`.
2. Crea una classe astratta `Viaggio` con attributi come `destinazione`, `prezzo` e un metodo `descrizione()`.
3. Implementa due classi `ViaggioAereo` e `ViaggioTreno`, con caratteristiche specifiche (es. `compagniaAerea`, `numeroPosto`).
4. Gestisci le prenotazioni con una classe `SistemaPrenotazioni` che utilizza una `List<Viaggio>`.
5. Aggiungi gestione delle eccezioni per evitare la prenotazione di un viaggio già prenotato.

Esercizio 6: Sistema di Pagamenti (Ereditarietà, Interfacce, Polimorfismo, Liste, Eccezioni)

Creiamo un sistema che gestisce pagamenti con diversi metodi.

Requisiti:

1. Crea un'interfaccia `Pagabile` con un metodo `effettuaPagamento(double importo)`.
2. Crea una classe astratta `MetodoPagamento` con attributi `saldoDisponibile`.
3. Implementa le classi `CartaDiCredito` e `PayPal`, con comportamenti specifici per i pagamenti.
4. Aggiungi una classe `SistemaPagamenti` che gestisce una lista di metodi di pagamento.
5. Implementa eccezioni personalizzate per gestire il saldo insufficiente.