Esercizio 1: Simulazione di un Gioco di Ruolo

Introduzione: In questo progetto, lo scopo è sviluppare una simulazione di un semplice gioco di ruolo (RPG) in Java. Il gioco includerà personaggi giocanti (PG) e avversari, e i giocatori avranno la possibilità di combattere i mostri in un'arena virtuale.

Obiettivi:

1. Creare classi per rappresentare i giocatori e i mostri.
2. Implementare un'arena di combattimento.
3. Consentire ai giocatori di attaccare e difendersi.
4. Gestire il turno di gioco.

Dettagli della consegna:

1. **Creazione delle classi:**
    - Crea una classe **Giocatore** con i seguenti attributi:
        - **nome**: il nome del giocatore.
        - **classe**: la classe del giocatore (es. guerriero, mago, ladro, ecc.).
        - **puntiVita**: i punti vita del giocatore.
        - **puntiMana**: i punti mana del giocatore.
    - Crea una classe **Mostro** con attributi simili a quelli di **Giocatore**.
2. **Implementazione dei metodi:**
    - In entrambe le classi (**Giocatore** e **Mostro**), implementa un metodo **attacca()** che simula un attacco e riduce i punti vita dell'avversario.
    - Implementa un metodo **difendi()** che permette al giocatore di ridurre il danno subito.
    - Aggiungi un metodo **stampaStato()** che stampa lo stato corrente del personaggio (nome, punti vita, punti mana, ecc.).
3. **Gestione dell'arena di combattimento:**
    - Crea una classe **Arena** che gestisce gli scontri tra giocatori e mostri.
    - Implementa un metodo **combatti()** che simula uno scontro tra un giocatore e un mostro.
    - Il combattimento dovrebbe consistere in una serie di turni in cui i personaggi si alternano nell'attaccare e nel difendere fino a quando uno dei due è sconfitto.
    - Aggiungi un metodo **mostraVincitore()** che determina il vincitore dello scontro e lo stampa a schermo.

4. **Gestione del turno di gioco:**
    - Implementa un sistema di turni in modo che i personaggi si alternino nell'attaccare e nel difendere.
    - Assicurati che i giocatori possano attaccare solo quando è il loro turno e che possano difendere quando è il turno dell'avversario.
    - Continua i turni fino a quando uno dei personaggi viene sconfitto.

Esercizio 2: Sviluppo di un'applicazione per la Gestione di un'agenzia di viaggi

Introduzione: L'obiettivo di questo progetto è sviluppare un'applicazione Java per la gestione di un'agenzia di viaggi. L'applicazione dovrà consentire agli utenti di visualizzare, aggiungere, modificare e eliminare pacchetti viaggio, gestire le prenotazioni e generare rapporti sulle vendite.

Obiettivi:

1. Creare classi per rappresentare i pacchetti viaggio, i clienti e le prenotazioni.
2. Implementare metodi per la gestione dei pacchetti viaggio e delle prenotazioni.
3. Consentire agli utenti di effettuare prenotazioni per i pacchetti viaggio.
4. Generare rapporti sulle vendite e sulle prenotazioni.

Dettagli della consegna:

1. **Creazione delle classi:**
    - Crea una classe **PacchettoViaggio** con attributi come destinazione, durata, prezzo, posti disponibili, ecc.

      Per gestire i viaggi, sono previsti i seguenti metodi:

        - **aggiungiPacchettoViaggio(PacchettoViaggio pacchetto)**: Aggiunge un nuovo pacchetto viaggio all'inventario.
        - **modificaPacchettoViaggio(int id, PacchettoViaggio nuovoPacchetto)**: Modifica un pacchetto viaggio esistente identificato dall'ID.
        - **rimuoviPacchettoViaggio(int id)**: Rimuove un pacchetto viaggio dall'inventario identificato dall'ID.
        - **visualizzaPacchettiViaggio()**: Visualizza tutti i pacchetti viaggio disponibili.
        - **cercaPacchettiViaggio(String destinazione, int durata, double prezzoMassimo)**: Cerca pacchetti viaggio in base alla destinazione, alla durata e al prezzo massimo.
        - **applicaSconto(double percentualeSconto)**: Applica uno sconto percentuale a tutti i pacchetti viaggio.
    - Crea una classe **Cliente** con attributi come nome, cognome, email, ecc.

      Per gestire i clienti, sono previsti i seguenti metodi:

        - **registraCliente(Cliente cliente)**: Registra un nuovo cliente nel sistema.
        - **modificaCliente(String email, Cliente nuovoCliente)**: Modifica i dettagli di un cliente esistente identificato dall'email.
        - **eliminaCliente(String email)**: Elimina un cliente dal sistema identificato dall'email.
        - **visualizzaClienti()**: Visualizza tutti i clienti registrati nel sistema.
        - **cercaCliente(String keyword)**: Cerca clienti in base a una parola chiave nel nome, nel cognome o nell'email.

- Crea una classe **Prenotazione** per gestire le prenotazioni dei clienti per i pacchetti viaggio.

  Per gestire le prenotazioni, sono previsti i seguenti metodi:

  - **effettuaPrenotazione(String emailCliente, int idPacchetto)**: Effettua una prenotazione per un cliente per un pacchetto viaggio specifico.
  - **annullaPrenotazione(String emailCliente, int idPacchetto)**: Annulla una prenotazione per un cliente per un pacchetto viaggio specifico.
  - **visualizzaPrenotazioniCliente(String emailCliente)**: Visualizza tutte le prenotazioni di un cliente.
  - **visualizzaPrenotazioniPacchetto(int idPacchetto)**: Visualizza tutte le prenotazioni per un pacchetto viaggio specifico.

2. **Funzionalità aggiuntive:**
   - Aggiungi la possibilità per gli utenti di cercare pacchetti viaggio per destinazione, durata, prezzo, ecc.
   - Implementa un sistema di sconti o promozioni per i pacchetti viaggio.
   - Aggiungi funzionalità per la gestione dei pagamenti e delle ricevute.
3. **Interfaccia utente:**
   - Crea un'interfaccia utente intuitiva che consenta agli utenti di navigare facilmente attraverso i pacchetti viaggio, effettuare prenotazioni e generare rapporti.