Esercizio 1: Liste e ArrayList

Implementare una classe `GestioneStudenti` che utilizzi un `ArrayList<String>` per memorizzare i nomi degli studenti di una classe. La classe deve fornire i seguenti metodi:

1. `aggiungiStudente(String nome)` : aggiunge uno studente alla lista
2. `rimuoviStudente(String nome)` : rimuove uno studente dalla lista dato il suo nome
3. `cercaStudente(String nome)` : verifica se uno studente è presente nella lista
4. `contaStudenti()` : restituisce il numero di studenti nella lista
5. `visualizzaStudenti()` : visualizza tutti gli studenti presenti nella lista in ordine alfabetico

Esercizio 2: Pile (Stack)

Implementare una classe `VerificaEspressioneBilanciata` che, utilizzando uno Stack, verifichi se una espressione matematica ha le parentesi bilanciate. Ad esempio:

- `(2+3)*(5-2)` → bilanciata
- `((2+3)*(5-2)` → non bilanciata
- `(2+3))*(5-2)` → non bilanciata

La classe deve implementare il metodo `boolean isBalanced(String espressione)` che restituisce `true` se l'espressione è bilanciata, `false` altrimenti.

Esercizio 3: Code (Queue)

Implementare una simulazione di una coda di stampa utilizzando la struttura dati `Queue` . Creare una classe `CodaStampa` con i seguenti metodi:

1. `aggiungiDocumento(String nomeDocumento)` : aggiunge un documento alla coda di stampa
2. `stampaDocumento()` : "stampa" (rimuove e visualizza) il documento in testa alla coda
3. `visualizzaCoda()` : visualizza tutti i documenti in coda senza modificare la coda stessa
4. `contaDocumenti()` : restituisce il numero di documenti ancora in coda
5. `svuotaCoda()` : svuota completamente la coda

Esercizio 4: Code a priorità (PriorityQueue)

Implementare una classe `GestioneTasks` che utilizzi una `PriorityQueue` per gestire una lista di attività con priorità. Ogni task è rappresentato dalla classe `Task` con attributi:

- `String descrizione` : descrizione del task
- `int priorita` : livello di priorità (valori più bassi = priorità più alta)

Implementare i seguenti metodi:

1. `aggiungiTask(String descrizione, int priorita)` : aggiunge un task alla coda
2. `eseguiTaskPrioritario()` : rimuove e visualizza il task con priorità più alta
3. `visualizzaTaskPrioritario()` : visualizza (senza rimuovere) il task con priorità più alta
4. `contaTasks()` : restituisce il numero di task nella coda

Esercizio 5: HashMap

Implementare una classe `ConteggioParole` che utilizzi una `HashMap` per contare la frequenza delle parole in un testo. La classe deve fornire i seguenti metodi:

1. `analizzaTesto(String testo)` : analizza il testo e aggiorna le frequenze delle parole
2. `getFrequenza(String parola)` : restituisce il numero di occorrenze di una parola
3. `getParolaPiuFrequente()` : restituisce la parola più frequente nel testo
4. `visualizzaFrequenze()` : visualizza tutte le parole con le rispettive frequenze

Esercizio 6: LinkedList

Implementare una classe `RegistroPresenze` che utilizzi una `LinkedList` per tenere traccia delle presenze giornaliere in un corso. Ogni presenza è rappresentata dalla classe `Presenza` con attributi:

- `String nomeStudente` : nome dello studente
- `String data` : data della presenza (formato "DD/MM/YYYY")

Implementare i seguenti metodi:

1. `registraPresenza(String nomeStudente, String data)` : registra una nuova presenza
2. `rimuoviPresenza(String nomeStudente, String data)` : rimuove una presenza specifica
3. `getPresenzeStudente(String nomeStudente)` : restituisce una lista delle date in cui lo studente era presente
4. `getStudentiPresenti(String data)` : restituisce una lista degli studenti presenti in una data specifica
5. `visualizzaRegistro()` : visualizza tutte le presenze registrate

Esercizio 7: HashSet

Implementare una classe `BibliotecaUnivoca` che utilizzi un `HashSet` per gestire una collezione di libri unici. La classe deve fornire i seguenti metodi:

1. `aggiungiLibro(String titolo, String autore)` : aggiunge un libro alla collezione
2. `rimuoviLibro(String titolo, String autore)` : rimuove un libro dalla collezione

3. `cercaLibro(String titolo, String autore)` : verifica se un libro è presente nella collezione
4. `contaLibri()` : restituisce il numero di libri nella collezione
5. `visualizzaLibri()` : visualizza tutti i libri nella collezione

Esercizio 8: TreeMap

Implementare una classe `RubricaTelefonica` che utilizzi un `TreeMap` per gestire contatti telefonici ordinati alfabeticamente. La classe deve fornire i seguenti metodi:

1. `aggiungiContatto(String nome, String numeroTelefono)` : aggiunge un contatto alla rubrica
2. `rimuoviContatto(String nome)` : rimuove un contatto dalla rubrica
3. `getNumero(String nome)` : restituisce il numero di telefono di un contatto
4. `cercaContatto(String nome)` : verifica se un contatto è presente nella rubrica
5. `visualizzaContatti()` : visualizza tutti i contatti in ordine alfabetico