```java
// ESERCIZIO 1: Liste e ArrayList

import java.util.ArrayList;
import java.util.Collections;

class GestioneStudenti {
    private ArrayList<String> studenti;

    public GestioneStudenti() {
        studenti = new ArrayList<>();
    }

    public void aggiungiStudente(String nome) {
        studenti.add(nome);
    }

    public boolean rimuoviStudente(String nome) {
        return studenti.remove(nome);
    }

    public boolean cercaStudente(String nome) {
        return studenti.contains(nome);
    }

    public int contaStudenti() {
        return studenti.size();
    }

    public void visualizzaStudenti() {
        if (studenti.isEmpty()) {
            System.out.println("Nessuno studente presente.");
            return;
        }

        // Crea una copia della lista e la ordina alfabeticamente
        ArrayList<String> studentiOrdinati = new ArrayList<>(studenti);
        Collections.sort(studentiOrdinati);

        System.out.println("Elenco studenti in ordine alfabetico:");
        for (String studente : studentiOrdinati) {
            System.out.println("- " + studente);
        }
    }

    // Metodo main per testare la classe
    public static void main(String[] args) {
        GestioneStudenti gestione = new GestioneStudenti();
```

```
        gestione.aggiungiStudente("Mario Rossi");
        gestione.aggiungiStudente("Anna Bianchi");
        gestione.aggiungiStudente("Luca Verdi");

        gestione.visualizzaStudenti();

        System.out.println("\nNumero studenti: " +
gestione.contaStudenti());

        System.out.println("\nCerco Mario Rossi: " +
                           (gestione.cercaStudente("Mario Rossi") ? "Trovato"
: "Non trovato"));
        System.out.println("Cerco Paolo Neri: " +
                           (gestione.cercaStudente("Paolo Neri") ? "Trovato"
: "Non trovato"));

        System.out.println("\nRimozione di Anna Bianchi: " +
                           (gestione.rimuoviStudente("Anna Bianchi") ?
"Rimosso" : "Non trovato"));

        System.out.println("\nElenco aggiornato:");
        gestione.visualizzaStudenti();
    }
}

// ESERCIZIO 2: Pile (Stack)

import java.util.Stack;

class VerificaEspressioneBilanciata {

    public boolean isBalanced(String espressione) {
        Stack<Character> stack = new Stack<>();

        for (int i = 0; i < espressione.length(); i++) {
            char carattere = espressione.charAt(i);

            if (carattere == '(') {
                // Se troviamo una parentesi aperta, la inseriamo nello
stack
                stack.push(carattere);
            }
            else if (carattere == ')') {
                // Se troviamo una parentesi chiusa, verifichiamo se lo
stack è vuoto
                if (stack.isEmpty()) {
                    return false; // Parentesi chiusa senza corrispondente
parentesi aperta
                }
```

```
                // Rimuoviamo la parentesi aperta corrispondente dallo stack
                stack.pop();
            }
            // Ignoriamo tutti gli altri caratteri
        }

        // L'espressione è bilanciata se lo stack è vuoto alla fine
        return stack.isEmpty();
    }

    // Metodo main per testare la classe
    public static void main(String[] args) {
        VerificaEspressioneBilanciata verifica = new
VerificaEspressioneBilanciata();

        String espressione1 = "(2+3)*(5-2)";
        String espressione2 = "((2+3)*(5-2)";
        String espressione3 = "(2+3))*(5-2)";

        System.out.println("Espressione: " + espressione1 + " -> " +
                           (verifica.isBalanced(espressione1) ? "bilanciata"
: "non bilanciata"));

        System.out.println("Espressione: " + espressione2 + " -> " +
                           (verifica.isBalanced(espressione2) ? "bilanciata"
: "non bilanciata"));

        System.out.println("Espressione: " + espressione3 + " -> " +
                           (verifica.isBalanced(espressione3) ? "bilanciata"
: "non bilanciata"));
    }
}

// ESERCIZIO 3: Code (Queue)

import java.util.LinkedList;
import java.util.Queue;

class CodaStampa {
    private Queue<String> documenti;

    public CodaStampa() {
        documenti = new LinkedList<>();
    }

    public void aggiungiDocumento(String nomeDocumento) {
        documenti.add(nomeDocumento);
        System.out.println("Documento '" + nomeDocumento + "' aggiunto alla
coda di stampa.");
    }
```

```
    public String stampaDocumento() {
        if (documenti.isEmpty()) {
            System.out.println("Nessun documento in coda.");
            return null;
        }

        String documento = documenti.poll(); // Rimuove e restituisce
l'elemento in testa alla coda
        System.out.println("Stampa in corso: '" + documento + "'");
        return documento;
    }

    public void visualizzaCoda() {
        if (documenti.isEmpty()) {
            System.out.println("Nessun documento in coda.");
            return;
        }

        System.out.println("Documenti in coda di stampa:");
        int posizione = 1;

        // Utilizziamo un array temporaneo per visualizzare la coda senza
modificarla
        String[] array = documenti.toArray(new String[0]);
        for (String documento : array) {
            System.out.println(posizione + ". " + documento);
            posizione++;
        }
    }

    public int contaDocumenti() {
        return documenti.size();
    }

    public void svuotaCoda() {
        int numeroDocumenti = documenti.size();
        documenti.clear();
        System.out.println("Coda di stampa svuotata. " + numeroDocumenti + "
documenti rimossi.");
    }

    // Metodo main per testare la classe
    public static void main(String[] args) {
        CodaStampa coda = new CodaStampa();

        coda.aggiungiDocumento("Relazione_Fisica.pdf");
        coda.aggiungiDocumento("Tesina_Storia.docx");
        coda.aggiungiDocumento("Presentazione_Informatica.pptx");
```

```
        System.out.println("\nNumero documenti in coda: " +
coda.contaDocumenti());

        System.out.println("\nVisualizzazione della coda:");
        coda.visualizzaCoda();

        System.out.println("\nStampa documento:");
        coda.stampaDocumento();

        System.out.println("\nCoda aggiornata:");
        coda.visualizzaCoda();

        System.out.println("\nSvuotamento coda:");
        coda.svuotaCoda();

        coda.visualizzaCoda();
    }
}

// ESERCIZIO 4: Code a priorità (PriorityQueue)

import java.util.PriorityQueue;
import java.util.Comparator;

class Task {
    private String descrizione;
    private int priorita;

    public Task(String descrizione, int priorita) {
        this.descrizione = descrizione;
        this.priorita = priorita;
    }

    public String getDescrizione() {
        return descrizione;
    }

    public int getPriorita() {
        return priorita;
    }

    @Override
    public String toString() {
        return "Task: " + descrizione + " (priorità: " + priorita + ")";
    }
}

class GestioneTasks {
    private PriorityQueue<Task> tasks;
```

```
    public GestioneTasks() {
        // Utilizziamo un comparatore che mette prima i task con priorità
più bassa
        tasks = new PriorityQueue<>
(Comparator.comparingInt(Task::getPriorita));
    }

    public void aggiungiTask(String descrizione, int priorita) {
        Task nuovoTask = new Task(descrizione, priorita);
        tasks.add(nuovoTask);
        System.out.println("Task aggiunto: " + nuovoTask);
    }

    public Task eseguiTaskPrioritario() {
        if (tasks.isEmpty()) {
            System.out.println("Nessun task presente.");
            return null;
        }

        Task taskPrioritario = tasks.poll(); // Rimuove e restituisce
l'elemento con priorità più alta
        System.out.println("Esecuzione task prioritario: " +
taskPrioritario);
        return taskPrioritario;
    }

    public Task visualizzaTaskPrioritario() {
        if (tasks.isEmpty()) {
            System.out.println("Nessun task presente.");
            return null;
        }

        Task taskPrioritario = tasks.peek(); // Visualizza senza rimuovere
        System.out.println("Task prioritario: " + taskPrioritario);
        return taskPrioritario;
    }

    public int contaTasks() {
        return tasks.size();
    }

    // Metodo main per testare la classe
    public static void main(String[] args) {
        GestioneTasks gestione = new GestioneTasks();

        gestione.aggiungiTask("Completare relazione", 3);
        gestione.aggiungiTask("Riparare bug critico", 1);
        gestione.aggiungiTask("Aggiornare database", 2);

        System.out.println("\nNumero di task: " + gestione.contaTasks());
```

```
        System.out.println("\nVisualizzazione task prioritario:");
        gestione.visualizzaTaskPrioritario();

        System.out.println("\nEsecuzione task prioritario:");
        gestione.eseguiTaskPrioritario();

        System.out.println("\nTask prioritario aggiornato:");
        gestione.visualizzaTaskPrioritario();
    }
}

// ESERCIZIO 5: HashMap

import java.util.HashMap;
import java.util.Map;

class ConteggioParole {
    private HashMap<String, Integer> frequenze;

    public ConteggioParole() {
        frequenze = new HashMap<>();
    }

    public void analizzaTesto(String testo) {
        // Pulisce il testo e lo divide in parole
        String[] parole = testo.toLowerCase()
                               .replaceAll("[.,;:!?()\\[\\]{}\"'\\-_]", " ")
                               .split("\\s+");

        for (String parola : parole) {
            if (!parola.isEmpty()) {
                // Incrementa il conteggio per ogni parola
                frequenze.put(parola, frequenze.getOrDefault(parola, 0) +
1);
            }
        }
    }

    public int getFrequenza(String parola) {
        return frequenze.getOrDefault(parola.toLowerCase(), 0);
    }

    public String getParolaPiuFrequente() {
        if (frequenze.isEmpty()) {
            return null;
        }

        String parolaPiuFrequente = null;
        int frequenzaMassima = 0;
```

```
        for (Map.Entry<String, Integer> entry : frequenze.entrySet()) {
            if (entry.getValue() > frequenzaMassima) {
                parolaPiuFrequente = entry.getKey();
                frequenzaMassima = entry.getValue();
            }
        }

        return parolaPiuFrequente;
    }

    public void visualizzaFrequenze() {
        if (frequenze.isEmpty()) {
            System.out.println("Nessuna parola analizzata.");
            return;
        }

        System.out.println("Frequenza delle parole:");
        for (Map.Entry<String, Integer> entry : frequenze.entrySet()) {
            System.out.println("- \"" + entry.getKey() + "\": " +
entry.getValue());
        }
    }

    // Metodo main per testare la classe
    public static void main(String[] args) {
        ConteggioParole conteggio = new ConteggioParole();

        String testo = "Questo è un esempio di testo. Questo testo contiene
parole ripetute. " +
                       "Le parole ripetute servono per testare il conteggio
delle parole.";

        conteggio.analizzaTesto(testo);

        System.out.println("Analisi del testo:\n" + testo + "\n");

        conteggio.visualizzaFrequenze();

        String parolaCercata = "parole";
        System.out.println("\nFrequenza della parola \"" + parolaCercata +
"\": " +
                           conteggio.getFrequenza(parolaCercata));

        String parolaPiuFrequente = conteggio.getParolaPiuFrequente();
        System.out.println("\nLa parola più frequente è: \"" +
parolaPiuFrequente + "\" (" +
                           conteggio.getFrequenza(parolaPiuFrequente) + "
occorrenze)");
    }
```

```
}

// ESERCIZIO 6: LinkedList

import java.util.LinkedList;
import java.util.List;
import java.util.Iterator;

class Presenza {
    private String nomeStudente;
    private String data;

    public Presenza(String nomeStudente, String data) {
        this.nomeStudente = nomeStudente;
        this.data = data;
    }

    public String getNomeStudente() {
        return nomeStudente;
    }

    public String getData() {
        return data;
    }

    @Override
    public String toString() {
        return nomeStudente + " (presente il " + data + ")";
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;

        Presenza altra = (Presenza) obj;
        return nomeStudente.equals(altra.nomeStudente) &&
data.equals(altra.data);
    }
}

class RegistroPresenze {
    private LinkedList<Presenza> presenze;

    public RegistroPresenze() {
        presenze = new LinkedList<>();
    }

    public void registraPresenza(String nomeStudente, String data) {
        // Verifica se la presenza è già registrata
```

```
        Presenza nuovaPresenza = new Presenza(nomeStudente, data);
        for (Presenza p : presenze) {
            if (p.equals(nuovaPresenza)) {
                System.out.println("Presenza già registrata per " +
nomeStudente + " in data " + data);
                return;
            }
        }

        presenze.add(nuovaPresenza);
        System.out.println("Presenza registrata per " + nomeStudente + " in
data " + data);
    }

    public boolean rimuoviPresenza(String nomeStudente, String data) {
        Iterator<Presenza> iterator = presenze.iterator();
        while (iterator.hasNext()) {
            Presenza p = iterator.next();
            if (p.getNomeStudente().equals(nomeStudente) &&
p.getData().equals(data)) {
                iterator.remove();
                System.out.println("Presenza rimossa per " + nomeStudente +
" in data " + data);
                return true;
            }
        }

        System.out.println("Presenza non trovata per " + nomeStudente + " in
data " + data);
        return false;
    }

    public List<String> getPresenzeStudente(String nomeStudente) {
        List<String> date = new LinkedList<>();

        for (Presenza p : presenze) {
            if (p.getNomeStudente().equals(nomeStudente)) {
                date.add(p.getData());
            }
        }

        return date;
    }

    public List<String> getStudentiPresenti(String data) {
        List<String> studenti = new LinkedList<>();

        for (Presenza p : presenze) {
            if (p.getData().equals(data)) {
                studenti.add(p.getNomeStudente());
```

```
            }
        }

        return studenti;
    }

    public void visualizzaRegistro() {
        if (presenze.isEmpty()) {
            System.out.println("Registro presenze vuoto.");
            return;
        }

        System.out.println("Registro presenze:");
        for (Presenza p : presenze) {
            System.out.println("- " + p);
        }
    }

    // Metodo main per testare la classe
    public static void main(String[] args) {
        RegistroPresenze registro = new RegistroPresenze();

        registro.registraPresenza("Mario Rossi", "15/05/2024");
        registro.registraPresenza("Anna Bianchi", "15/05/2024");
        registro.registraPresenza("Luca Verdi", "15/05/2024");
        registro.registraPresenza("Mario Rossi", "16/05/2024");
        registro.registraPresenza("Anna Bianchi", "16/05/2024");

        System.out.println("\nVisualizzazione registro completo:");
        registro.visualizzaRegistro();

        String nomeStudente = "Mario Rossi";
        List<String> datePresenza =
registro.getPresenzeStudente(nomeStudente);
        System.out.println("\nDate di presenza per " + nomeStudente + ":");
        for (String data : datePresenza) {
            System.out.println("- " + data);
        }

        String data = "15/05/2024";
        List<String> studentiPresenti = registro.getStudentiPresenti(data);
        System.out.println("\nStudenti presenti il " + data + ":");
        for (String studente : studentiPresenti) {
            System.out.println("- " + studente);
        }

        System.out.println("\nRimozione presenza:");
        registro.rimuoviPresenza("Mario Rossi", "15/05/2024");

        System.out.println("\nRegistro aggiornato:");
```

```
        registro.visualizzaRegistro();
    }
}

// ESERCIZIO 7: HashSet

import java.util.HashSet;
import java.util.Objects;

class Libro {
    private String titolo;
    private String autore;

    public Libro(String titolo, String autore) {
        this.titolo = titolo;
        this.autore = autore;
    }

    public String getTitolo() {
        return titolo;
    }

    public String getAutore() {
        return autore;
    }

    @Override
    public String toString() {
        return "\"" + titolo + "\" di " + autore;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;
        if (obj == null || getClass() != obj.getClass()) return false;

        Libro libro = (Libro) obj;
        return titolo.equals(libro.titolo) && autore.equals(libro.autore);
    }

    @Override
    public int hashCode() {
        return Objects.hash(titolo, autore);
    }
}

class BibliotecaUnivoca {
    private HashSet<Libro> libri;

    public BibliotecaUnivoca() {
```

```
        libri = new HashSet<>();
    }

    public boolean aggiungiLibro(String titolo, String autore) {
        Libro nuovoLibro = new Libro(titolo, autore);
        boolean aggiunto = libri.add(nuovoLibro);

        if (aggiunto) {
            System.out.println("Libro aggiunto: " + nuovoLibro);
        } else {
            System.out.println("Libro già presente: " + nuovoLibro);
        }

        return aggiunto;
    }

    public boolean rimuoviLibro(String titolo, String autore) {
        Libro libro = new Libro(titolo, autore);
        boolean rimosso = libri.remove(libro);

        if (rimosso) {
            System.out.println("Libro rimosso: " + libro);
        } else {
            System.out.println("Libro non trovato: " + libro);
        }

        return rimosso;
    }

    public boolean cercaLibro(String titolo, String autore) {
        Libro libro = new Libro(titolo, autore);
        return libri.contains(libro);
    }

    public int contaLibri() {
        return libri.size();
    }

    public void visualizzaLibri() {
        if (libri.isEmpty()) {
            System.out.println("Biblioteca vuota.");
            return;
        }

        System.out.println("Libri in biblioteca:");
        for (Libro libro : libri) {
            System.out.println("- " + libro);
        }
    }
```

```
    // Metodo main per testare la classe
    public static void main(String[] args) {
        BibliotecaUnivoca biblioteca = new BibliotecaUnivoca();

        biblioteca.aggiungiLibro("Il nome della rosa", "Umberto Eco");
        biblioteca.aggiungiLibro("1984", "George Orwell");
        biblioteca.aggiungiLibro("Il piccolo principe", "Antoine de Saint-
Exupéry");

        // Prova ad aggiungere un libro duplicato
        biblioteca.aggiungiLibro("Il nome della rosa", "Umberto Eco");

        System.out.println("\nNumero libri in biblioteca: " +
biblioteca.contaLibri());

        System.out.println("\nVisualizzazione biblioteca:");
        biblioteca.visualizzaLibri();

        String titolo = "1984";
        String autore = "George Orwell";
        System.out.println("\nRicerca libro \"" + titolo + "\" di " + autore
+ ": " +
                           (biblioteca.cercaLibro(titolo, autore) ? "Trovato"
: "Non trovato"));

        titolo = "La Divina Commedia";
        autore = "Dante Alighieri";
        System.out.println("Ricerca libro \"" + titolo + "\" di " + autore +
": " +
                           (biblioteca.cercaLibro(titolo, autore) ? "Trovato"
: "Non trovato"));

        System.out.println("\nRimozione libro:");
        biblioteca.rimuoviLibro("Il piccolo principe", "Antoine de Saint-
Exupéry");

        System.out.println("\nBiblioteca aggiornata:");
        biblioteca.visualizzaLibri();
    }
}

// ESERCIZIO 8: TreeMap

import java.util.TreeMap;

class RubricaTelefonica {
    private TreeMap<String, String> contatti;

    public RubricaTelefonica() {
        contatti = new TreeMap<>();
```

```
    }

    public void aggiungiContatto(String nome, String numeroTelefono) {
        contatti.put(nome, numeroTelefono);
        System.out.println("Contatto aggiunto: " + nome + " - " +
numeroTelefono);
    }

    public boolean rimuoviContatto(String nome) {
        String numero = contatti.remove(nome);

        if (numero != null) {
            System.out.println("Contatto rimosso: " + nome + " - " +
numero);
            return true;
        } else {
            System.out.println("Contatto non trovato: " + nome);
            return false;
        }
    }

    public String getNumero(String nome) {
        return contatti.get(nome);
    }

    public boolean cercaContatto(String nome) {
        return contatti.containsKey(nome);
    }

    public void visualizzaContatti() {
        if (contatti.isEmpty()) {
            System.out.println("Rubrica vuota.");
            return;
        }

        System.out.println("Contatti in rubrica (ordine alfabetico):");
        for (java.util.Map.Entry<String, String> entry :
contatti.entrySet()) {
            System.out.println("- " + entry.getKey() + ": " +
entry.getValue());
        }
    }

    // Metodo main per testare la classe
    public static void main(String[] args) {
        RubricaTelefonica rubrica = new RubricaTelefonica();

        rubrica.aggiungiContatto("Mario Rossi", "333-1234567");
        rubrica.aggiungiContatto("Anna Bianchi", "333-7654321");
        rubrica.aggiungiContatto("Luca Verdi", "333-9876543");
```

```
        rubrica.aggiungiContatto("Carla Neri", "333-1122334");

        System.out.println("\nVisualizzazione rubrica:");
        rubrica.visualizzaContatti();

        String nome = "Anna Bianchi";
        System.out.println("\nRicerca numero di " + nome + ": " +
                           (rubrica.cercaContatto(nome) ?
rubrica.getNumero(nome) : "Non trovato"));

        nome = "Giulia Gialli";
        System.out.println("Ricerca numero di " + nome + ": " +
                           (rubrica.cercaContatto(nome) ?
rubrica.getNumero(nome) : "Non trovato"));

        System.out.println("\nAggiornamento contatto:");
        rubrica.aggiungiContatto("Mario Rossi", "333-8765432"); //
Sovrascrive il numero precedente

        System.out.println("\nRimozione contatto:");
        rubrica.rimuoviContatto("Luca Verdi");

        System.out.println("\nRubrica aggiornata:");
        rubrica.visualizzaContatti();
    }
}

// CLASSE PRINCIPALE PER ESEGUIRE TUTTI GLI ESERCIZI

public class EserciziStruttureDati {
    public static void main(String[] args) {
        System.out.println("===== ESERCIZIO 1: GESTIONE STUDENTI =====");
        GestioneStudenti.main(args);

        System.out.println("\n\n===== ESERCIZIO 2: VERIFICA ESPRESSIONE
BILANCIATA =====");
        VerificaEspressioneBilanciata.main(args);

        System.out.println("\n\n===== ESERCIZIO 3: CODA DI STAMPA =====");
        CodaStampa.main(args);

        System.out.println("\n\n===== ESERCIZIO 4: GESTIONE TASKS =====");
        GestioneTasks.main(args);

        System.out.println("\n\n===== ESERCIZIO 5: CONTEGGIO PAROLE =====");
        ConteggioParole.main(args);

        System.out.println("\n\n===== ESERCIZIO 6: REGISTRO PRESENZE
=====");
        RegistroPresenze.main(args);
```

```
        System.out.println("\n\n===== ESERCIZIO 7: BIBLIOTECA UNIVOCA
=====");
        BibliotecaUnivoca.main(args);

        System.out.println("\n\n===== ESERCIZIO 8: RUBRICA TELEFONICA
=====");
        RubricaTelefonica.main(args);
    }
}
```