Esercizio 1

Implementare una classe `GestioneLibri` che permetta di gestire una raccolta di libri. Ogni libro è caratterizzato da un titolo (String), un autore (String) e un anno di pubblicazione (int).

La classe `GestioneLibri` deve avere i seguenti metodi:

1. Un costruttore che accetta la dimensione massima della raccolta di libri.
2. Un metodo `aggiungiLibro` che accetta un titolo, un autore e un anno di pubblicazione e aggiunge il libro alla raccolta, se c'è spazio disponibile. Restituisce true se l'operazione ha avuto successo, false altrimenti.
3. Un metodo `rimuoviLibro` che accetta un titolo e rimuove il libro dalla raccolta, se presente. Restituisce true se l'operazione ha avuto successo, false altrimenti.
4. Un metodo `cercaLibro` che accetta un titolo e restituisce l'indice del libro nella raccolta, se presente. Se non viene trovato, restituisce -1.
5. Un metodo `stampaLibri` che stampa a video tutti i libri presenti nella raccolta, con il formato "Titolo: <titolo>, Autore: <autore>, Anno: <anno>".
6. Un metodo `libriPerAutore` che accetta un autore e restituisce un array di stringhe contenente i titoli dei libri scritti da quell'autore, presenti nella raccolta.
7. Un metodo `libriPremaAnno` che accetta un anno e restituisce un array di stringhe contenente i titoli dei libri pubblicati prima di quell'anno, presenti nella raccolta.

Crea una classe `Main` separata per testare la funzionalità della classe `GestioneLibri`.

Suggerimenti:

- Puoi utilizzare un array di oggetti Libro (da creare) per memorizzare i libri nella raccolta.
- Ricorda di gestire correttamente i casi limite, come la raccolta piena o vuota.

Questo esercizio ti permetterà di praticare la programmazione orientata agli oggetti, la gestione delle raccolte di dati con gli array e l'implementazione di metodi per manipolare e accedere ai dati.

Soluzione:

```
// Classe Libro
class Libro {
    private String titolo;
    private String autore;
    private int annoPubblicazione;

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

    public String getTitolo() {
        return titolo;
    }

    public String getAutore() {
        return autore;
    }

    public int getAnnoPubblicazione() {
        return annoPubblicazione;
    }
}

// Classe GestioneLibri
class GestioneLibri {
    private Libro[] raccolta;
    private int numLibri;

    public GestioneLibri(int dimensioneMassima) {
        raccolta = new Libro[dimensioneMassima];
```

```
        numLibri = 0;
    }

    public boolean aggiungiLibro(String titolo, String autore, int
annoPubblicazione) {
        if (numLibri < raccolta.length) {
            raccolta[numLibri] = new Libro(titolo, autore,
annoPubblicazione);
            numLibri++;
            return true;
        }
        return false;
    }

    public boolean rimuoviLibro(String titolo) {
        int indice = cercaLibro(titolo);
        if (indice != -1) {
            for (int i = indice; i < numLibri - 1; i++) {
                raccolta[i] = raccolta[i + 1];
            }
            numLibri--;
            return true;
        }
        return false;
    }

    public int cercaLibro(String titolo) {
        for (int i = 0; i < numLibri; i++) {
            if (raccolta[i].getTitolo().equals(titolo)) {
                return i;
            }
        }
        return -1;
    }
```

```
    public void stampaLibri() {
        for (int i = 0; i < numLibri; i++) {
            Libro libro = raccolta[i];
            System.out.println("Titolo: " + libro.getTitolo() + ",
Autore: " + libro.getAutore() + ", Anno: " +
libro.getAnnoPubblicazione());
        }
    }

    public String[] libriPerAutore(String autore) {
        int count = 0;
        for (int i = 0; i < numLibri; i++) {
            if (raccolta[i].getAutore().equals(autore)) {
                count++;
            }
        }

        String[] libriAutore = new String[count];
        int index = 0;
        for (int i = 0; i < numLibri; i++) {
            if (raccolta[i].getAutore().equals(autore)) {
                libriAutore[index++] = raccolta[i].getTitolo();
            }
        }

        return libriAutore;
    }

    public String[] libriPremaAnno(int anno) {
        int count = 0;
        for (int i = 0; i < numLibri; i++) {
            if (raccolta[i].getAnnoPubblicazione() < anno) {
                count++;
```

```
            }
        }

        String[] libriPremaAnno = new String[count];
        int index = 0;
        for (int i = 0; i < numLibri; i++) {
            if (raccolta[i].getAnnoPubblicazione() < anno) {
                libriPremaAnno[index++] = raccolta[i].getTitolo();
            }
        }

        return libriPremaAnno;
    }
}

// Classe Main per testare la funzionalità
public class Main {
    public static void main(String[] args) {
        GestioneLibri biblioteca = new GestioneLibri(5);

        biblioteca.aggiungiLibro("Il Signore degli Anelli", "J.R.R.
Tolkien", 1954);
        biblioteca.aggiungiLibro("Harry Potter e la Pietra Filosofale",
"J.K. Rowling", 1997);
        biblioteca.aggiungiLibro("Il Codice Da Vinci", "Dan Brown",
2003);
        biblioteca.aggiungiLibro("La Divina Commedia", "Dante Alighieri",
1321);
        biblioteca.aggiungiLibro("Il Vecchio e il Mare", "Ernest
Hemingway", 1951);

        System.out.println("Libri nella raccolta:");
        biblioteca.stampaLibri();

        System.out.println("\nLibri di J.R.R. Tolkien:");
```

```
        String[] libriTolkien = biblioteca.libriPerAutore("J.R.R.
Tolkien");
        for (String titolo : libriTolkien) {
            System.out.println(titolo);
        }

        System.out.println("\nLibri pubblicati prima del 1900:");
        String[] libriPrema1900 = biblioteca.libriPremaAnno(1900);
        for (String titolo : libriPrema1900) {
            System.out.println(titolo);
        }

        biblioteca.rimuoviLibro("Il Vecchio e il Mare");
        System.out.println("\nLibri dopo la rimozione:");
        biblioteca.stampaLibri();
    }
}
```

Esercizio 2

Implementare un sistema di gestione di un'agenzia di viaggi. Il sistema deve includere le seguenti classi:

1. `Destinazione`: Rappresenta una destinazione di viaggio, con attributi come nome, paese, descrizione e costo base (in euro).
2. `Viaggio`: Rappresenta un viaggio organizzato dall'agenzia, con attributi come destinazione, data di partenza, data di ritorno, costo totale (calcolato in base al costo base della destinazione e alla durata del viaggio), e un identificatore univoco.
3. `Cliente`: Rappresenta un cliente dell'agenzia, con attributi come nome, cognome, email, e un identificatore univoco.
4. `Prenotazione`: Rappresenta una prenotazione di un viaggio da parte di un cliente, con attributi come viaggio, cliente, data di prenotazione e stato (prenotato, confermato, cancellato).
5. *`AgenziaViaggi`*: Gestisce l'intera agenzia, con metodi per aggiungere/rimuovere destinazioni, creare viaggi, effettuare prenotazioni e cancellazioni, e altre funzionalità come:
    - Cercare viaggi per destinazione, data di partenza o costo
    - Elencare le prenotazioni di un determinato cliente
    - Calcolare il fatturato totale dell'agenzia in un determinato periodo
    - Generare statistiche sui viaggi più prenotati o sulle destinazioni più popolari

Inoltre, implementare le seguenti regole di business:

- Il costo totale di un viaggio è calcolato in base al costo base della destinazione e alla durata del viaggio (ad esempio, costo base + 10% del costo base per ogni giorno di durata).
- Una prenotazione può essere effettuata solo se il viaggio non è sold-out (ad esempio, con un limite massimo di 20 prenotazioni per viaggio).
- Una prenotazione può essere cancellata senza penali fino a 14 giorni prima della data di partenza. Oltre tale periodo, viene applicata una penale pari al 20% del costo totale del viaggio.
- Uno sconto del 10% sul costo totale viene applicato ai viaggi prenotati almeno 6 mesi prima della data di partenza.

Crea una classe `Main` separata per testare il funzionamento del sistema, creando istanze di destinazioni, viaggi, clienti ed effettuando operazioni di prenotazione, cancellazione e generazione di statistiche.

Soluzione:

```
import java.util.Date;


public class Main {
    public static void main(String[] args) {
        AgenziaViaggi agenzia = new AgenziaViaggi();


        // Aggiungi destinazioni
        agenzia.aggiungiDestinazione(new Destinazione("Roma", "Italia",
"La città eterna", 500));
        agenzia.aggiungiDestinazione(new Destinazione("Parigi",
"Francia", "La città della luce", 700));
        agenzia.aggiungiDestinazione(new Destinazione("New York", "Stati
Uniti", "La Grande Mela", 1000));


        // Crea viaggi
        agenzia.creaViaggio(0, new Date(), new
Date(System.currentTimeMillis() + 86400000 * 7)); // Viaggio a Roma, 7
giorni
        agenzia.creaViaggio(1, new Date(), new
Date(System.currentTimeMillis() + 86400000 * 14)); // Viaggio a Parigi,
14 giorni
        agenzia.creaViaggio(2, new Date(), new
Date(System.currentTimeMillis() + 86400000 * 10)); // Viaggio a New York,
10 giorni


        // Aggiungi clienti
        agenzia.aggiungiCliente(new Cliente("Mario", "Rossi",
"mario@example.com"));
        agenzia.aggiungiCliente(new Cliente("Luigi", "Verdi",
"luigi@example.com"));


        // Effettua prenotazioni
        agenzia.effettuaPrenotazione(0, 0, new Date()); // Prenotazione
di Mario per il viaggio a Roma
        agenzia.effettuaPrenotazione(1, 1, new Date()); // Prenotazione
di Luigi per il viaggio a Parigi
```

```
        // Annulla una prenotazione
        agenzia.annullaPrenotazione(0); // Annullamento della
prenotazione di Mario

        // Stampa statistiche
        agenzia.generaStatistiche();
    }
}

// Classe destinazione
class Destinazione {
    private String nome;
    private String paese;
    private String descrizione;
    private double costoBase;

    public Destinazione(String nome, String paese, String descrizione,
double costoBase) {
        this.nome = nome;
        this.paese = paese;
        this.descrizione = descrizione;
        this.costoBase = costoBase;
    }

    // Getters
    public String getNome() {
        return nome;
    }

    public String getPaese() {
        return paese;
    }

    public String getDescrizione() {
```

```
        return descrizione;
    }

    public double getCostoBase() {
        return costoBase;
    }
}

// Classe viaggio
class Viaggio {
    private static int nextId = 1;

    private int id;
    private int idDestinazione;
    private Date dataPartenza;
    private Date dataRitorno;
    private double costoTotale;
    private int[] prenotazioni = new int[20];
    private int numPrenotazioni = 0;

    public Viaggio(int idDestinazione, Date dataPartenza, Date
dataRitorno) {
        this.id = nextId++;
        this.idDestinazione = idDestinazione;
        this.dataPartenza = dataPartenza;
        this.dataRitorno = dataRitorno;
        this.costoTotale = calcolaCostoTotale();
    }

    // Calcola il costo totale del viaggio
    private double calcolaCostoTotale() {
        long durata = (dataRitorno.getTime() - dataPartenza.getTime()) /
(1000 * 60 * 60 * 24); // in giorni
```

```
        Destinazione destinazione =
AgenziaViaggi.destinazioni[idDestinazione];
        return destinazione.getCostoBase() + (0.1 *
destinazione.getCostoBase() * durata);
    }

    // Getters
    public int getId() {
        return id;
    }

    public int getIdDestinazione() {
        return idDestinazione;
    }

    public Date getDataPartenza() {
        return dataPartenza;
    }

    public Date getDataRitorno() {
        return dataRitorno;
    }

    public double getCostoTotale() {
        return costoTotale;
    }

    public int[] getPrenotazioni() {
        return prenotazioni;
    }

    public int getNumPrenotazioni() {
        return numPrenotazioni;
    }
```

```
    // Aggiungi prenotazione
    public void aggiungiPrenotazione(int idPrenotazione) {
        prenotazioni[numPrenotazioni++] = idPrenotazione;
    }
}

// Classe cliente
class Cliente {
    private static int nextId = 1;

    private int id;
    private String nome;
    private String cognome;
    private String email;

    public Cliente(String nome, String cognome, String email) {
        this.id = nextId++;
        this.nome = nome;
        this.cognome = cognome;
        this.email = email;
    }

    // Getters
    public int getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }

    public String getCognome() {
```

```
        return cognome;
    }

    public String getEmail() {
        return email;
    }
}

// Classe prenotazione
class Prenotazione {
    private int idViaggio;
    private int idCliente;
    private Date dataPrenotazione;
    private String stato;

    public Prenotazione(int idViaggio, int idCliente, Date
dataPrenotazione) {
        this.idViaggio = idViaggio;
        this.idCliente = idCliente;
        this.dataPrenotazione = dataPrenotazione;
        this.stato = "prenotato";
    }

    // Getters
    public int getIdViaggio() {
        return idViaggio;
    }

    public int getIdCliente() {
        return idCliente;
    }

    public Date getDataPrenotazione() {
        return dataPrenotazione;
```

```
    }

    public String getStato() {
        return stato;
    }

    // Metodo per confermare la prenotazione
    public void confermaPrenotazione() {
        stato = "confermato";
    }

    // Metodo per annullare la prenotazione
    public void annullaPrenotazione() {
        stato = "cancellato";
    }
}

// Classe AgenziaViaggi
class AgenziaViaggi {
    public static Destinazione[] destinazioni = new Destinazione[10];
    public static Viaggio[] viaggi = new Viaggio[100];
    public static Cliente[] clienti = new Cliente[100];
    public static Prenotazione[] prenotazioni = new Prenotazione[1000];
    private static int numDestinazioni = 0;
    private static int numViaggi = 0;
    private static int numClienti = 0;
    private static int numPrenotazioni = 0;

    // Aggiungi destinazione
    public void aggiungiDestinazione(Destinazione destinazione) {
        destinazioni[numDestinazioni++] = destinazione;
    }
```

```
    // Crea viaggio
    public void creaViaggio(int idDestinazione, Date dataPartenza, Date
dataRitorno) {
        Viaggio viaggio = new Viaggio(idDestinazione, dataPartenza,
dataRitorno);
        viaggi[numViaggi++] = viaggio;
    }

    // Aggiungi cliente
    public void aggiungiCliente(Cliente cliente) {
        clienti[numClienti++] = cliente;
    }

    // Effettua prenotazione
    public void effettuaPrenotazione(int idViaggio, int idCliente, Date
dataPrenotazione) {
        Viaggio viaggio = viaggi[idViaggio];
        if (viaggio.getNumPrenotazioni() < 20) { // Verifica se il
viaggio non è sold-out
            Prenotazione prenotazione = new Prenotazione(idViaggio,
idCliente, dataPrenotazione);
            prenotazioni[numPrenotazioni++] = prenotazione;
            viaggio.aggiungiPrenotazione(numPrenotazioni - 1);
        } else {
            System.out.println("Il viaggio è sold-out, impossibile
effettuare la prenotazione.");
        }
    }

    // Annulla prenotazione
    public void annullaPrenotazione(int idPrenotazione) {
        Prenotazione prenotazione = prenotazioni[idPrenotazione];
        Date today = new Date();
        long diffInMillies =
Math.abs(viaggi[prenotazione.getIdViaggio()].getDataPartenza().getTime()
- today.getTime());
```

```
        long diffInDays = diffInMillies / (1000 * 60 * 60 * 24);


        if (diffInDays <= 14) { // Annullamento senza penali entro 14 giorni dalla partenza
            prenotazione.annullaPrenotazione();
        } else { // Applicazione di una penale del 20% del costo totale oltre i 14 giorni
            double penale = 0.2 * viaggi[prenotazione.getIdViaggio()].getCostoTotale();
            System.out.println("È trascorso il periodo di annullamento senza penali. Verrà applicata una penale di " + penale + " euro.");
        }
    }


    // Genera statistiche
    public void generaStatistiche() {
        // Implementazione delle statistiche
    }
}
```