```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 */

package com.mycompany.teatromoros8;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 *
 * @author Alex Fernández
 */
public class TeatroMoroS8 {
    static Scanner scanner = new Scanner(System.in);

    // Constantes
    static final int MAX_SALES = 100;
    static final int ROWS = 5;
    static final int COLUMNS = 10;
    static final double BASE_PRICE = 10000.0;

    // Listas
    static List<String> discounts = new ArrayList<>();
    static List<Reservation> reservations = new ArrayList<>();

    // Arreglos
    static Customer[] customers = new Customer[MAX_SALES]; // Arreglo de clientes
    static int[] saleIds = new int[MAX_SALES]; // Arreglo de IDs de ventas
    static boolean[][] seatMap = new boolean[ROWS][COLUMNS]; // Arreglo para gestionar los asientos

    // Variables auxiliares
    static int saleIndex = 0;

    // Clase para representar una reserva
    static class Reservation {
        int reservationId;
        int customerId; // ID del cliente relacionado
        String customerName; // Nombre del cliente
        String discountType; // Tipo de descuento aplicado
        int row;
        int column;
        double finalPrice;

        public Reservation(int reservationId, int customerId, String customerName, String discountType, int row, int column, double finalPrice) {
            this.reservationId = reservationId;
            this.customerId = customerId;
            this.customerName = customerName;
            this.discountType = discountType;
            this.row = row;
            this.column = column;
            this.finalPrice = finalPrice;
```

```
53          }
54
55          @Override
56          public String toString() {
57              return "Reserva #" + reservationId + ": Cliente " + customerName + " (ID: "
    + customerId + "), Descuento: " + discountType + ", Asiento [" + row + "," + column +
    "], Precio final: $" + finalPrice;
58          }
59      }
60
61      // Clase para representar un cliente
62      static class Customer {
63          int id;
64          String name;
65          String discountType;
66
67          public Customer(int id, String name, String discountType) {
68              this.id = id;
69              this.name = name;
70              this.discountType = discountType;
71          }
72
73          public String getName() {
74              return name;
75          }
76
77          public int getId() {
78              return id;
79          }
80
81          public String getDiscountType() {
82              return discountType;
83          }
84      }
85
86      public static void main(String[] args) {
87          initializeDiscounts(); // Inicializamos la lista de descuentos
88          showMenu();
89      }
90
91      static void initializeDiscounts() {
92          discounts.add("estudiante:0.10");
93          discounts.add("tercera edad:0.15");
94          discounts.add("ninguno:0.0");
95      }
96
97      static void showMenu() {
98          int option;
99          do {
100             System.out.println("\n--- Teatro Moro - Sistema Optimizado ---");
101             System.out.println("1. Vender entrada");
102             System.out.println("2. Mostrar reservas");
103             System.out.println("3. Eliminar reserva");
104             System.out.println("4. Actualizar asiento de reserva");
105             System.out.println("5. Salir");
106             option = getValidOption(1, 5);
107             switch (option) {
```

```
                case 1:
                    sellTicket();
                    break;
                case 2:
                    showReservations();
                    break;
                case 3:
                    deleteReservation();
                    break;
                case 4:
                    updateReservation();
                    break;
                case 5:
                    displaySeatMap();
                    break;
                default:
                    System.out.println("Opción inválida.");
            }
        } while (option != 0);
    }

    static void sellTicket() {
        if (saleIndex >= MAX_SALES) {
            System.out.println("Se alcanzó el límite de ventas.");
            return;
        }

        System.out.print("Ingrese el nombre del cliente: ");
        String name = scanner.nextLine();

        System.out.print("¿Es estudiante/tercera edad/ninguno? (Ingresar solo una categoría): ");
        String discountType = getValidDiscountType();

        int row = getValidRow();

        int column = getValidSeat();

        if (seatMap[row][column]) {
            System.out.println("El asiento ya está ocupado.");
            return;
        }

        double discount = calculateDiscount(discountType);
        double finalPrice = BASE_PRICE - (BASE_PRICE * discount);

        // Crear nuevo cliente
        Customer customer = new Customer(saleIndex + 1, name, discountType);
        customers[saleIndex] = customer;

        // Crear nueva reserva
        Reservation reservation = new Reservation(saleIndex + 1, customer.getId(), customer.getName(), discountType, row, column, finalPrice);
        reservations.add(reservation);

        // Marcar el asiento como ocupado
        seatMap[row][column] = true;
```

```
163
164         // Guardar ID de la venta
165         saleIds[saleIndex] = customer.getId();
166
167         System.out.println("¡Entrada vendida con éxito!");
168         System.out.println(reservation);
169
170         saleIndex++;
171     }
172
173     static double calculateDiscount(String discountType) {
174         // Buscar el tipo de descuento en la lista
175         for (String discount : discounts) {
176             String[] parts = discount.split(":");
177             if (parts[0].equals(discountType)) {
178                 return Double.parseDouble(parts[1]);
179             }
180         }
181         return 0.0; // Si no encuentra, no hay descuento
182     }
183
184     static void showReservations() {
185         if (reservations.isEmpty()) {
186             System.out.println("No hay reservas registradas.");
187         } else {
188             for (Reservation reservation : reservations) {
189                 System.out.println(reservation);
190             }
191         }
192     }
193
194     static void deleteReservation() {
195         System.out.print("Ingrese el ID de la reserva a eliminar: ");
196         int id = scanner.nextInt();
197         scanner.nextLine();
198
199         Reservation r = findReservationById(id);
200         if (r != null) {
201             // Liberar el asiento
202             seatMap[r.row][r.column] = false;
203             // Eliminar el cliente de la lista de clientes
204             customers[r.customerId - 1] = null;  // Eliminar el cliente del arreglo (el
    cliente tiene un ID basado en el índice de venta)
205
206             // Eliminar el ID de venta del arreglo saleIds
207             saleIds[r.customerId - 1] = 0;  // Eliminar el ID de venta
208
209             // Eliminar la reserva de la lista de reservas
210             reservations.remove(r);
211             System.out.println("Reserva eliminada correctamente.");
212         } else {
213             System.out.println("No se encontró una reserva con ese ID.");
214         }
215     }
216
217     static void updateReservation() {
218         System.out.print("Ingrese el ID de la reserva a actualizar: ");
```

```
219         int id = scanner.nextInt();
220         scanner.nextLine();
221 
222         Reservation r = findReservationById(id);
223         if (r != null) {
224             // Liberar el asiento anterior
225             seatMap[r.row][r.column] = false;
226 
227             int newRow = getValidRow();
228             int newColumn = getValidSeat();
229 
230             if (seatMap[newRow][newColumn]) {
231                 System.out.println("El nuevo asiento ya está ocupado.");
232                 return;
233             }
234 
235             // Asignar el nuevo asiento
236             r.row = newRow;
237             r.column = newColumn;
238             seatMap[newRow][newColumn] = true;
239 
240             System.out.println("Reserva actualizada correctamente.");
241         } else {
242             System.out.println("No se encontró una reserva con ese ID.");
243         }
244     }
245 
246     // UTILIDADES
247     static void displaySeatMap() {
248         System.out.println("\nMapa de asientos (X = ocupado, O = libre):");
249         for (int i = 0; i < ROWS; i++) {
250             System.out.print("Fila " + i + ": ");
251             for (int j = 0; j < COLUMNS; j++) {
252                 System.out.print(seatMap[i][j] ? "X " : "O ");
253             }
254             System.out.println();
255         }
256     }
257 
258     static Reservation findReservationById(int id) {
259         for (Reservation r : reservations) {
260             if (r.reservationId == id) {
261                 return r;
262             }
263         }
264         return null;
265     }
266 
267     static int getValidRow() {
268         int row = -1;
269         boolean validRow = false;
270 
271         while(!validRow) {
272             System.out.print("Seleccione fila (0-4): ");
273             if (scanner.hasNextInt()) {
274                 row = scanner.nextInt();
275                 scanner.nextLine();
```

```
276                 if (row >= 0 && row < 5) {
277                     validRow = true;
278                 } else {
279                     System.out.println("Fila fuera de rango.");
280                 }
281             } else {
282                 System.out.println("Entrada inválida. Ingrese un número.");
283                 scanner.nextLine();
284             }
285         }
286 
287         return row;
288     }
289 
290     static int getValidSeat() {
291         int seat = -1;
292         boolean validSeat = false;
293 
294         while (!validSeat) {
295             System.out.print("Seleccione asiento (0-9): ");
296             if (scanner.hasNextInt()) {
297                 seat = scanner.nextInt();
298                 scanner.nextLine();
299 
300                 if (seat >= 0 && seat < 10) {
301                     validSeat = true;
302                 } else {
303                     System.out.println("Asiento fuera de rango.");
304                 }
305             } else {
306                 System.out.println("Entrada inválida. Ingrese un número: ");
307                 scanner.nextLine(); // Limpiar el buffer
308             }
309         }
310         return seat;
311     }
312 
313     static String getValidDiscountType() {
314         String discountType = scanner.nextLine().toLowerCase();
315         while (!discountType.equals("estudiante") && !discountType.equals("tercera
    edad") && !discountType.equals("ninguno")) {
316             System.out.print("Tipo de descuento inválido. Ingrese 'estudiante',
    'tercera edad' o 'ninguno': ");
317             discountType = scanner.nextLine().toLowerCase();
318         }
319         return discountType;
320     }
321 
322     static int getValidOption(int min, int max) {
323         int option = -1;
324         boolean validOption = false;
325 
326         while (!validOption) {
327             System.out.print("Seleccione una opción (" + min + "-" + max + "): ");
328             if (scanner.hasNextInt()) {
329                 option = scanner.nextInt();
330                 scanner.nextLine();
```

```
                if (option >= min && option <= max) {
                    validOption = true;
                } else {
                    System.out.println("Opción fuera de rango.");
                }
            } else {
                System.out.println("Entrada inválida. Ingrese un número.");
                scanner.nextLine();
            }
        }

        return option;
    }
}
```