Scenario 3: Train Booking System

import java.util.ArrayList;

import java.util.Scanner;

class BookingSystem {

    private final int TOTAL\_SEATS = 10;

    private String[] seatArray;

    private ArrayList<String> seatList;

    private boolean useArray = true;

    private Scanner scanner;

    public BookingSystem() {

        seatArray = new String[TOTAL\_SEATS];

        seatList = new ArrayList<>();

        scanner = new Scanner(System.in);

        // Initially, all seats are empty in both structures

        for (int i = 0; i < TOTAL\_SEATS; i++) {

            seatList.add(null);

        }

    }

    public void run() {

        int choice;

        do {

            displayMenu();

            choice = getIntInput("Enter your choice: ");

            switch (choice) {

                case 1:

                    toggleStructure();

                    break;

                case 2:

                    bookSeat();

                    break;

                case 3:

                    cancelSeat();

                    break;

                case 4:

                    searchAvailableSeats();

                    break;

                case 5:

                    displayAllSeats();

                    break;

                case 6:

                    System.out.println("Exiting... Goodbye!");

                    break;

                default:

                    System.out.println("Invalid choice.");

            }

            System.out.println();

        } while (choice != 6);

    }

    private void displayMenu() {

        System.out.println("\t\t\t\t\t\t\t=====================================");

        System.out.println("\t\t\t\t\t\t\t\tTRAIN SEAT BOOKING SYSTEM");

        System.out.println("\t\t\t\t\t\t\t=====================================");

        System.out.println("\t\t\t\t\t\t\tCurrent Mode: " + (useArray ? "Array (Fixed)" : "ArrayList (Dynamic)"));

        System.out.println("1. Toggle Data Structure");

        System.out.println("2. Book Seat");

        System.out.println("3. Cancel Booking");

        System.out.println("4. Search Available Seats");

        System.out.println("5. Display All Bookings");

        System.out.println("6. Exit");

        System.out.println("=====================================");

    }

    private void toggleStructure() {

        useArray = !useArray;

        System.out.println("\t\t\t\t\t\t\tSwitched to: " + (useArray ? "Array" : "ArrayList"));

    }

    private void bookSeat() {

        String name = getStringInput("Enter passenger name: ");

        int seatNumber = getIntInput("Enter seat number (0 to " + (TOTAL\_SEATS - 1) + "): ");

        if (seatNumber < 0 || seatNumber >= TOTAL\_SEATS) {

            System.out.println("Invalid seat number.");

            return;

        }

        if (useArray) {

            if (seatArray[seatNumber] == null) {

                seatArray[seatNumber] = name;

                System.out.println("Seat booked!");

            } else {

                System.out.println("Seat already booked.");

            }

        } else {

            if (seatList.get(seatNumber) == null) {

                seatList.set(seatNumber, name);

                System.out.println("Seat booked!");

            } else {

                System.out.println("Seat already booked.");

            }

        }

    }

    private void cancelSeat() {

        int seatNumber = getIntInput("Enter seat number to cancel: ");

        if (seatNumber < 0 || seatNumber >= TOTAL\_SEATS) {

            System.out.println("Invalid seat number.");

            return;

        }

        if (useArray) {

            if (seatArray[seatNumber] != null) {

                System.out.println("Booking cancelled for: " + seatArray[seatNumber]);

                seatArray[seatNumber] = null;

                shiftSeatsArray(seatNumber);

            } else {

                System.out.println("Seat is already empty.");

            }

        } else {

            if (seatList.get(seatNumber) != null) {

                System.out.println("Booking cancelled for: " + seatList.get(seatNumber));

                seatList.remove(seatNumber);

                seatList.add(null);

            } else {

                System.out.println("Seat is already empty.");

            }

        }

    }

    private void shiftSeatsArray(int fromIndex) {

        for (int i = fromIndex; i < TOTAL\_SEATS - 1; i++) {

            seatArray[i] = seatArray[i + 1];

        }

        seatArray[TOTAL\_SEATS - 1] = null;

    }

    private void searchAvailableSeats() {

        System.out.println("Available Seats:");

        if (useArray) {

            for (int i = 0; i < TOTAL\_SEATS; i++) {

                if (seatArray[i] == null) {

                    System.out.println("Seat " + i + " is available.");

                }

            }

        } else {

            for (int i = 0; i < TOTAL\_SEATS; i++) {

                if (seatList.get(i) == null) {

                    System.out.println("Seat " + i + " is available.");

                }

            }

        }

    }

    private void displayAllSeats() {

        System.out.println("Current Bookings:");

        if (useArray) {

            for (int i = 0; i < TOTAL\_SEATS; i++) {

                System.out.println("Seat " + i + ": " + (seatArray[i] != null ? seatArray[i] : "Empty"));

            }

        } else {

            for (int i = 0; i < TOTAL\_SEATS; i++) {

                System.out.println("Seat " + i + ": " + (seatList.get(i) != null ? seatList.get(i) : "Empty"));

            }

        }

    }

    private int getIntInput(String msg) {

        System.out.print(msg);

        while (!scanner.hasNextInt()) {

            System.out.println("Please enter a valid number.");

            scanner.next();

            System.out.print(msg);

        }

        return scanner.nextInt();

    }

    private String getStringInput(String msg) {

        System.out.print(msg);

        scanner.nextLine();

        return scanner.nextLine();

    }

}

public class scenario3 {

    public static void main(String[] args) {

        BookingSystem system = new BookingSystem();

        system.run();

    }

}

Time Complexity Analysis

## Time Complexity Comparison

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Array (Fixed Size) | ArrayList (Dynamic) | Time Complexity |
| Booking Seat | O(1) if index known | O(1) amortized | Constant time for both (if end) |
| Canceling Seat | O(n) | O(n) | Need to shift elements |
| Search Availability | O(n) | O(n) | Linear scan for available seat |

**Analysis:**

**Array (Fixed Size):**

**• Pros:**  
 - Fixed memory usage.  
 - Direct access to seat index: O(1) for booking.

**• Cons:**  
 - Not flexible — resizing not possible.  
 - Canceling a booking requires shifting all seats after the index (O(n)).  
 - Poor for dynamic or unpredictable seat allocation.

**ArrayList:**

**• Pros:**  
 - Dynamically grows — more flexible for future expansion.  
 - Same indexing access as an array.  
 - Easy to manage complex seat objects with metadata (e.g., passenger name, timestamp).

• Cons:  
 - Shifting still required on deletion (O(n)).  
 - Slight memory overhead due to dynamic resizing and internal structure.

Conclusion: Which is better for large-scale systems?

ArrayList is better for large-scale systems because:

• Scalable: Can dynamically grow without needing to predefine size.  
• Maintainable: Easier to manage real-world seat objects.  
• Java Collections support: Rich APIs make coding simpler and cleaner.

In high-scale systems, where flexibility, maintenance, and future changes are key, ArrayList is the preferred data structure.