Event Scheduler And Calendar

Nyameye Akumia

Nyameye Awurama Ampomaa Obeng-Akuamoah

Nana Fouvle Yaaba Nkrumah

Ashesi University

CS222: Data Structures and Algorithms

Dr Govindha Yeluripati

2 August 2024

Project Overview

The Event Scheduler and Calendar for Students is a command-line application designed to help students manage their academic and personal schedules efficiently. This in-memory application allows users to create, modify, delete, and view events within a virtual calendar. Key features include event management, searching and sorting capabilities, and generating event summaries. The application targets students who need a straightforward, fast, and flexible tool to organise their time without requiring persistent storage.

Objectives

User-Friendly: Provide an intuitive CLI for managing events.

Efficient In-Memory Management: Utilize efficient data structures for storing and retrieving event data.

Comprehensive Filtering: Enable users to filter, search, and sort events based on various attributes.

Event Summarization: Generate summaries of events for specified date ranges.

Design And Implementation

We use four main classes: **Event**, **EventCollection**, **EventSummary**, and the **Main** application.

The **Event** class represents individual events in the calendar system, encapsulating attributes such as title, venue, description, eventID, organization, dateTime, and priorityStatus. It provides a constructor for initialisation, accessor and mutator methods for data manipulation, and a display method for formatted output. The class implements the Comparable interface, allowing events to be naturally ordered by date and time. It also overrides toString(), equals(), and hashCode() methods for proper object representation and usage in collections. Key features

include immutable eventID and organization fields, using LocalDateTime for precise timing, toggleable priority, and a comprehensive display method. This well-structured class forms the core data model for event management, offering functionality for creating, modifying, comparing, and displaying events in a calendar application.

The EventCollection class is a robust implementation for managing a set of Event objects, utilizing a HashSet as its core data structure for efficient storage and retrieval. It provides a comprehensive suite of operations including adding, removing, modifying, and retrieving events. The class implements flexible sorting functionality using Java's Comparator interface, allowing events to be sorted based on various attributes. It also features a powerful search capability using Java streams, enabling efficient filtering of events based on specified criteria. The view method offers a way to display events matching certain filters. Throughout its implementation, the class makes effective use of switch statements for attribute-based operations, enhancing extensibility and readability. By leveraging Java 8+ features such as lambda expressions and streams, the class achieves concise and efficient code, particularly in its search and modification methods. This design ensures scalable and performant event management, suitable for a wide range of calendar and scheduling applications.

The EventSummary class is designed to generate and display a summary of events from an EventCollection. It offers two constructors: one that takes an entire EventCollection, and another that accepts an EventCollection along with start and end dates to filter events within a specific time range. The class utilizes a Set<Event> to store the relevant events, leveraging Java streams in the date-range constructor to efficiently filter events. The core functionality is encapsulated in the private generateSummary() method, which calculates and prints key statistics about the events, including the total count, number of unique organizers, count of high-priority

events, and a list of organizers. This method demonstrates effective use of Java collections and loops to aggregate data. Overall, the EventSummary class provides a concise and efficient way to generate overviews of event data, making it a valuable tool for quick analysis in event management systems.

Data Structures And Algorithms

1. HashSet<Event> in EventCollection: Justification: Provides O(1) average time complexity for add, remove, and contains operations. Ideal for storing unique events and efficient retrieval.
2. ArrayList<Event> in EventCollection's sort method: Justification: Allows for efficient sorting using Java's built-in sorting algorithm (TimSort), which has O(n log n) time complexity.
3. Stream API in search and filter operations: Justification: Enables concise and efficient filtering of events, especially useful for complex queries.
4. Comparator in sort method: Justification: Provides flexible sorting based on different event attributes without modifying the Event class.

Performance (time complexity) Analysis And Optimization Techniques

1. Add/Remove operations: O(1) average case due to HashSet
2. Get event by ID: O(n) worst case, as it iterates through the set
3. Modify event: O(n) worst case, as it searches through the set
4. Sort: O(n log n) due to TimSort algorithm
5. Search: O(n) as it potentially examines all events
6. View: O(n) as it iterates through all events

Challenges Faced And Solutions Implemented

Date and Time Handling

- **Problem:** Managing date and time across different time zones to ensure accurate event scheduling presented challenges
- **Solution:** The `java.time` package was employed for robust date and time manipulation. To handle time zones accurately, `DateTime` was used where necessary, ensuring consistent and correct event scheduling.

User Input Validation

- **Problem:** Invalid user inputs could cause the application to crash or behave unexpectedly.
- **Solution:** Comprehensive input validation and error handling were implemented to ensure the CLI gracefully managed incorrect or unexpected inputs. This included validating dates, times, and ensuring required fields were populated.

Instructions For Using The Application

To use the application, refer to the `README.md` file for further instructions.