Data Structures Final Project Report

Group 5

Elisee Chrys Gnagne, Edward Ofosu Mensah, Stanley Tulani Ndlovu, Salome Kaluki Mutemwa

C S222_B: DATA STRUCTURES AND ALGORITHMS

Lecturer: Dr Govindha Yeluripai

August 3, 2024

Event Class

The `Event` class models an event with various attributes including a unique identifier (id), priority, title, date, time, location, and description. It automatically generates a unique ID for each new event using a static counter. The class includes a list of common date formats for parsing date strings, which are tried sequentially until a valid date is found. If none of the formats match the input, an `IllegalArgumentException` is thrown to signal an invalid date format. The constructors of the `Event` class allow for flexible event creation by accepting either a date string or a `LocalDate` object, ensuring that events can be instantiated in a variety of formats.

The primary constructor of the class takes multiple parameters, including an integer for priority, a title, a date string, time, location, and description. It attempts to parse the date string using predefined formats and assigns it to the event. This is useful when the date is not already in 'LocalDate' format or data inputted by the user. Alternatively, a secondary constructor allows creating an event directly from a `LocalDate` object, offering an additional option for date representation. Both constructors ensure that every event is initialised with a unique ID, maintaining a consistent and trackable identification system for each event instance.

The class also has the getter and setter methods for the event class attributes and a static method which generates a unique ID for each event and returns the next unique ID. It also provides a string representation of the event and returns a string describing the event with its details.

SortingCalendar Class

'SortingCalendar' class employs a linked list to store the event instances. It entails methods to handle various tasks such as createEvent, modifyEvent, deleteEvent, viewEvent, searchEvent, sortEvents, and generateSummary. The createEvent method adds new events to

the linked list by parsing event details from user input. The method takes details of the event from the user and using splitting and parsing, we are able to create a new event instance. The modifyEvent method allows for updating specific attributes of an event, such as title or date, by searching for the event in the list and applying changes. The deleteEvent method removes an event from the list based on its ID, handling both the case where the event is at the head of the list or elsewhere. It also returns a message when the event entered by the user is not found.

Additionally, the viewEvents method prints all events by implementing the filtering logic, while the searchEvent method locates events based on specific attributes like title, location or date. The sortEvents method sorts the events based on a specified attribute and employs the filtering logic (printList()) and sort method, and the generateSummary method is intended for generating summaries for a given date range. The generateSummary method implements a summary generation logic that will provide a friendlier interaction with the user. The class also has a main method that allows the user to enter their input with options such as creating an event, modifying an event, deleting an event, viewing events, searching for events, sorting events and generating summaries for the events. This class provides a comprehensive approach to managing events with flexible input handling, event modification, and sorting capabilities, leveraging a linked list for efficient storage and manipulation of events.

Calendar Class

‘Calendar’ Class provides functionalities to manage a calendar structure for a given year. It includes methods for determining if a year is a leap year, for creating a calendar with months and days and to sort the events by title, date, or priority using the EventsSorter class. The createCalendar method initialise a `HashMap` to represent the calendar, where each month maps to another `HashMap` that tracks days and their associated events. The number

of days in February is adjusted based on whether the year is a leap year. Each day is represented by an `ArrayList` that will hold `Event` objects, allowing for event management across the entire year.

Additionally, the `sortEvents` method is used to sort events within the calendar by specific attributes such as title, date, or priority. It leverages the `EventSorter` class to perform the sorting using merge sort. This method iterates through all months and days in the calendar, applying the sorting to each day's list of events. The `main` method demonstrates the creation of a calendar for the year 2024 and prints the structure of the calendar, showing the months and their respective days.

EventSorter Class

The `EventSorter` class is designed to sort a list of `Event` objects using the merge sort algorithm. This class contains methods to recursively split the list of events into smaller sublists, sort each sublist, and then merge them back together in a sorted order. The `mergeSort` method is the main function that handles the splitting and recursive sorting, while the `merge` method is responsible for merging the sorted sublists. The class allows sorting by different attributes of the `Event` objects in the compareBtAttribute method, such as date, title, and priority, and supports both ascending and descending order.

The use of merge sort in this class is beneficial for several reasons. First, merge sort is a stable sorting algorithm, meaning it preserves the relative order of equal elements, which can be important when sorting events with the same attribute values. Second, merge sort has a predictable time complexity of O(n log n), making it efficient for large lists of events. The recursive nature of merge sort allows it to handle the sorting process in a systematic way, ensuring that the events are sorted correctly based on the specified attribute. This makes the `EventSorter` class a robust and reliable tool for organising events in a calendar application.

The compareByAttribute method enhances the flexibility by allowing the sorting criterion to be easily changed.

Java.util.HashMap Class

The `java.util.HashMap` class is a part of the Java Collections Framework and provides the implementation of the Map interface. It stores key-value pairs and allows for efficient retrieval of values based on their corresponding keys. This class uses a hash table to store the map entries, ensuring that operations such as adding, removing, and accessing elements have an average time complexity of O(1). The `HashMap` class permits null values and the null key, and it does not guarantee any specific order of the entries.

A key feature of `HashMap` is its internal hashing mechanism, which calculates the hash code of each key and determines the bucket in which the entry is stored. This reduces the number of entries that need to be checked when searching for a key, enhancing performance. The class includes methods for standard operations, such as `put()`, `get()`, `remove()`, and `containsKey()`, as well as advanced operations like `putIfAbsent()`, `computeIfAbsent()`, and `merge()`. The `HashMap` class is not synchronised, meaning that if multiple threads access a hashmap concurrently and at least one thread modifies the map, it must be externally synchronised.

SearchingEvents class

The `SearchingEvent` class is designed to manage and search for events based on attributes such as title, date, and location. This class utilises binary search algorithms to efficiently find events within a sorted list, which significantly improves search speed and accuracy. The class includes methods for adding events, ensuring that the list remains sorted to facilitate quick searches. This approach is particularly useful for applications that require frequent and rapid querying of large event datasets, providing a streamlined and user-friendly way to retrieve specific events.

The class features three primary search methods: searchEventsByMeeting, searchEventByDate, and searchEventByLocation. Each method targets a specific event attribute, allowing for precise and focused searches. The searchEventsByMeeting method looks for events based on their title, while the searchEventByDate method utilises binary search to quickly locate events by date. The searchEventByLocation method searches through the list to find events held at a particular location. By maintaining a sorted list and implementing these targeted search functions, the SearchingEvent class ensures efficient and accurate event retrieval.

EventScheduler class

The `EventScheduler` class is designed to efficiently manage and search for events using attributes such as title, date, and location. By leveraging binary search algorithms, this class ensures quick and accurate retrieval of events from a sorted list. The binary search approach enhances performance, making it ideal for applications requiring frequent queries on large datasets. This structured and methodical approach to event management ensures that users can easily find and interact with their events, offering a seamless and intuitive experience.

The class includes three key methods for searching events: searchEventsByMeeting, searchEventByDate, and searchEventByLocation. The searchEventsByMeeting method allows users to search for events by their title, facilitating quick identification of specific events. The searchEventByDate method enables efficient searching based on the event's date, ensuring that users can easily locate events scheduled for a particular day. Lastly, the searchEventByLocation method helps users find events at a specified location. Each of these methods returns the event that matches the given criteria, providing a precise and user-friendly way to manage and access event information.

AutomateMain class

The `AutomateMain` class provides a comprehensive solution for managing and searching events stored in an input file. Upon running the program, the user is prompted to enter the filename containing the event details, which are then loaded into a calendar structure. The calendar is a nested `HashMap` where the outer key is the month and the inner key is the day, each pointing to a list of `Event` objects. The program includes an interactive menu with options to search for events by title, date, or location, and to sort events by various attributes such as date, title, or priority. The search functionality leverages binary search algorithms for efficiency, and the sort functionality ensures that events can be ordered based on user preferences.

The loadEventsFromFile method is responsible for reading event data from a specified file, parsing the event attributes, and organising them into the calendar structure. Each line in the input file represents an event, with attributes separated by commas. The method handles potential formatting issues and ensures that events are correctly parsed and added to the calendar. The interactive menu allows users to perform searches and sorts, displaying the time taken for each operation to highlight efficiency. The program provides clear feedback for invalid inputs and ensures a user-friendly experience.

Year_Calendar class

The `Year_Calendar` class is a Java implementation that manages and prints calendar data for multiple years using a nested `HashMap` structure. This class maintains a collection of calendars, each represented by a year and organised by month and day. The core functionality includes adding new years to the calendar, retrieving the calendar for a specific year, and printing the calendar structure for any given year. When a new year is added, the class initialises it with a calendar structure provided by the Calendar.createCalendar(year) method, ensuring that each year’s calendar is accurately represented.

The main method of the Year_Calendar class demonstrates its usage by adding and printing calendars for a range of years from 2024 to 2028. The program initialises an `Year_Calendar` instance, iterates through the specified years, adds each year to the calendar, and prints the resulting calendar structure. This approach provides a clear overview of how the `Year_Calendar` class manages multiple years and maintains the calendar data in an organised manner, allowing for efficient retrieval and display of calendar information.

References

DeepLearning4J Documentation. (n.d.). Retrieved from https://deeplearning4j.org/docs/latest/

HashMap (Java SE 14 & JDK 14). (n.d.). *Class hashmap*. https://download.java.net/java/GA/jdk14/docs/api/java.base/java/util/HashMap.html