COMP47500 – Advanced Data Structures in Java

Assignment No.1
Date: 25th February 2024

Title: A Queueing Model for Scheduling House Viewing Appointment Process

Group information	
Group Members:	% Contribution Assignment Workload
1. Ayushi Jain (22203974)	25%
2. Janice Shah (23200136)	25%
3. Chikeluba Okorji (22201350)	25%
4. Pooja Sivakumar (23200203)	25%

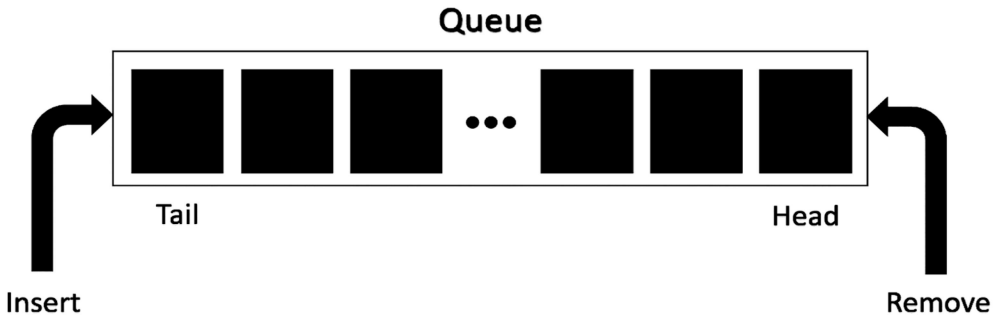
1. Problem Domain Description

Scheduling house viewing appointments can be a cumbersome and inefficient process for both potential buyers and sellers. Traditional methods often involve manual communication and coordination, leading to delays, missed appointments, and frustration. This project proposes a queueing model to streamline the scheduling process and improve efficiency for all parties involved.

Most efficient systems comprise three main entities: sellers, buyers, and pre-defined viewing slots. Sellers define viewing slots for their properties, specifying date, time, and duration. Buyers browse available properties and express interest in specific slots. A queuing system maintains a queue for each viewing slot, ordered by the buyer's requested arrival time. The slot is infinite therefore never reaches its maximum capacity, but subsequent requests are added to the queue and shuffled according to priority. The system then automatically assigns buyers to another fixed-length queue based on their availability to view the property based on the queue order and sends confirmation notifications. Sellers are notified of confirmed appointments and waitlisted buyers.

The Model:

The proposed model focuses on analyzing queue behaviour for individual viewing slots. We will only consider one primary scenario; single server i.e. one property at a time and multiple clients allowed to view the house at different time slots. The model incorporates key parameters like arrival rate in the form of move-in date, queue capacity (maximum number of buyers allowed to view a house), and an infinite queue capacity (number of buyers on the waitlist).



These metrics help assess the efficiency of the scheduling process. The model offers several benefits, including improved efficiency through automation and reduced communication, reduced waiting times through real-time queue visibility, increased transparency by ensuring fairness in scheduling, and data-driven insights for optimizing viewing slot availability and scheduling strategies. The model acknowledges limitations like assuming constant arrival rates and service times, and not accounting for factors like buyer preferences, seller availability, or last-minute cancellations. Future research can explore incorporating these complexities, developing multi-server models for open houses, and integrating with real-time scheduling platforms to further enhance the model's effectiveness. This queueing model provides a valuable framework for analyzing and optimizing the house viewing appointment scheduling process. By understanding queue behaviour and key performance measures, stakeholders can make informed decisions to improve efficiency, reduce waiting times, and enhance the overall experience for both buyers and sellers.

2. Theoretical Foundations of the Data Structure(s) Utilised

The discussed queueing model requires two queue data structures: one acting as a priority queue for appointments based on their proposed move-in date, and the other a fixed-size queue to manage client viewing availability dates.

1. Fixed-Size Queue

   Following the FIFO (First In, First Out) principle, a queue is ideal for representing our viewing date slot queue of size 3 where clients follow a strict order based on availability. Elements enter from the rear and exit from the front.

   We implement the queue using a linked list, also known as a linked list queue. Each element in the queue is a node in the linked list. The head represents the front, and the tail represents the rear. Adding an element (enqueue) involves creating a new node and attaching it to the tail. Removing an element (dequeue) involves removing the node at the head.

   The linked list approach allows for dynamic resizing as elements come and go, making it adaptable. Both enqueue and dequeue operations have a constant time complexity (O(1)) when head and tail pointers are maintained efficiently. However, neglecting them can lead to O(n) complexity in the worst

case.

2. Priority Queue
   A priority queue efficiently prioritizes data access. Similar to a regular queue, each element has an assigned priority value. Elements are ordered based on this value, with the highest priority at the front. As elements are removed, the next highest-priority element takes its place. Various data structures like arrays, linked lists, binary heaps, and balanced binary search trees can implement priority queues depending on the application's needs. These queues are valuable when processing items based on priority, such as in job scheduling, network routing, and simulations.

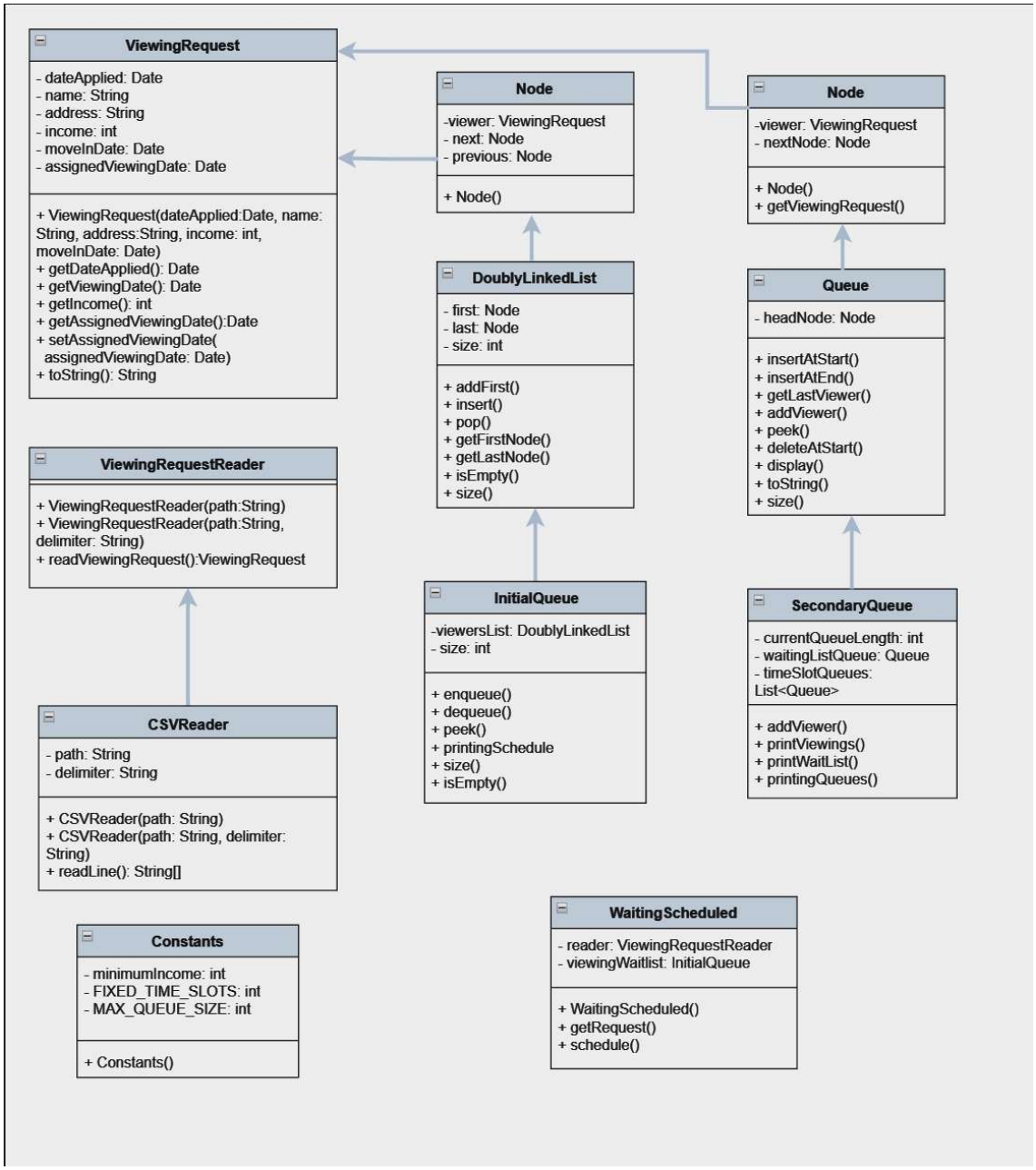
   We chose a double-linked list to create a priority queue for clients due to its dynamic size and ability to efficiently prioritize and reorder data as new clients register. During client insertion, we prioritise them based on their move-in date and update the queue accordingly. This involves reevaluating and recalculating the move-in date of all following clients whenever a new client is inserted. This ensures the queue remains sorted by priority, with the clients with the earliest move-in date at the front.

   Initially, our algorithm traversed the linked list from the head to find the new client position, recalculating the move-in date for all remaining clients until the tail. This resulted in a time complexity of O(n) in both best and worst cases. To optimize, we modified the algorithm to traverse from the tail. While traversing, we simultaneously recalculate the move-in date until the appropriate insertion position is found. This approach significantly improves the best-case time complexity to O(1) while maintaining the worst-case O(n) complexity.

3. **Data**
   Due to ethical considerations and data privacy regulations, acquiring real-world data for this project was not feasible. To overcome this challenge, we opted for a data generation approach that ensured both data integrity and privacy. We employed a combination of techniques, such as scenario-based data creation and anonymized data utilization. We developed hypothetical scenarios reflecting various appointment scheduling situations, encompassing diverse factors like client application date, name, address, average income and proposed move-in date. This approach allowed for testing the system's functionality under various conditions without compromising privacy. This approach enabled us to generate comprehensive data for testing and evaluation purposes while adhering to strict data privacy and ethical guidelines.

   1. *Application Date:* When the application was submitted
   2. *Client Name:* Full name of the applicant(s).
   3. *Address*: Current or previous address of the applicant(s).
   4. *Income:* Financial information for income verification.
   5. *Move-In Date:* Desired date for occupancy.

4. Implementation Design (UML Diagram)



5. Code Implementation

GitHub Link: https://github.com/JaniceShah/ViewingQueues

6. Experiment Results

1. Analysis of Doubly-Linked List

While a simple first-come-first-served (FCFS) queue is efficient for managing buyers' application requests, it becomes inefficient when dealing with varying application times to view a property. Clients with shorter move-in dates can

get stuck behind clients with longer ones, leading to schedule delays and customer dissatisfaction.

To address this issue, we implemented a priority queue that prioritizes buyers based on their move-in dates. This ensures clients with shorter move-in dates are processed first, minimizing congestion and delays. While effective, the initial implementation involved traversing the entire queue to insert a new buyer, resulting in an O(n) time complexity. To improve efficiency, we adopted a double-linked list data structure. This allows for constant time insertion and removal while maintaining the priority order based on a move-in date.

Further optimization involved updating the enqueue algorithm to traverse the linked list from the tail instead of the head. This approach efficiently locates the insertion point and minimises the number of affected buyers for move-in date recalculation. In the best-case scenario, this reduces the time complexity to O(1). However, the worst-case scenario (inserting at the head) still requires traversing the entire list, resulting in O(n) complexity.

By combining a priority queue with a double-linked list and a tail-to-head traversal approach, we achieve an efficient queuing system for scheduling buyers. This system prioritises buyers based on their move-in date, minimising delays and congestion while ensuring optimal resource utilization. This approach balances the trade-off between time complexity and performance, leading to a queuing system that effectively meets the needs of house viewing operations.

7. **Video Link of Implementation**
   Link:
   https://drive.google.com/file/d/1ZB3WaCbgWeDvdKx9fadPVRN0JbFXHJDY/view?usp=drive_link

8. **Conclusion and Comments**

In conclusion, the proposed queueing model offers a systematic approach to streamline the scheduling process for house viewing appointments, benefiting both buyers and sellers. By leveraging queue data structures and prioritization algorithms, the model effectively manages appointment requests, reduces waiting times, and enhances overall efficiency.

The utilization of fixed-size and priority queues ensures that viewing slots are allocated fairly and efficiently, taking into account the urgency of buyers' move-in dates. Through the implementation of a double-linked list and optimized traversal algorithms, the model achieves a balance between performance and time complexity, resulting in a robust scheduling system capable of handling varying application times with minimal delays.

Overall, the queueing model offers a valuable framework for analyzing and optimizing house viewing appointment scheduling processes. By embracing automation and

data-driven insights, stakeholders can make informed decisions to improve efficiency, reduce waiting times, and enhance the overall experience for both buyers and sellers in the real estate market.