MAJOR, ROYAL AND PONTIFICAL UNIVERSITY OF

SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTY OF SCIENCE AND TECHNOLOGY

TRAINING FOR DEVELOPERS

HOSPITAL QUEUING SYSTEM

Student: Luis Fabricio Liceras Rodríguez

Trainner: Mauricio Flores

Sucre – Bolivia

2024

1. INTRODUCTION

This program is designed to facilitate the management of patients requiring medical care in a hospital setting.

The system offers two main types of queues: the "General Queue" and the "Emergency Queue", each designed to handle different types of medical situations and priorities.

The "General Queue" is for patients who do not require immediate attention and can wait in order of arrival to receive treatment. Here, patients can be added, cared for, or referred to medical specialties as needed.

On the other hand, the "Emergency Queue" is dedicated to those patients who require urgent or emergency medical attention. Patients in this queue are categorized into different urgency levels, such as "Emergency," "Urgent," or "None." Depending on the patient's situation, they may be treated immediately, referred to the "General Queue" or, in critical cases, removed from the queue.

In addition to these main queues, the system also offers specialized queues for different medical areas, such as Oncology, Traumatology, Neurology, Cardiology, Nephrology, Pulmonology and Gastroenterology. This allows for more efficient management of patients, ensuring they receive appropriate and timely care based on their specific medical needs.

2. THEORETICAL FRAMEWORK

2.1. Patient Prioritization

Prioritizing patients according to emergency levels is a beneficial way to manage their care, better avoiding the loss of life.

This management program helps when managing the emergency queue by separating patients into emergency, urgency and none, facilitating care and referral to general care if emergency care is not necessary.

2.2. Queue Management

Queue management principles focus on optimizing patient flow through efficient use of resources and minimization of waiting times.

For this program, two types of queues were used, a normal queue and a double ended queue, both based on linked lists, making it easier to enter and remove patients from the queues.

3. PRACTICAL FRAMEWORK

3.1. Resources

The libraries used in this project are two:

java.util.Date: used to manage and insert dates and times for patient entries.

java.util.Scanner: it is a vital library for the user's interaction with the program, it allows the insertion of all types of data in the console, array sizes, number of repetitions, values and data to form the entries.

GeneralQueueRunner: This class is the main class of the program and contains the logic to manage patient queues.

GeneralQueue: An implementation of a generic queue.

EmergencyQueue: An implementation of an emergency queue, which extends the GeneralQueue class.

Entrance: A class that represents a patient entry with various attributes such as name, age, reason for entry, etc.

The queue implementation code was also used, dequeues based on linked list seen in the incrementProject, for its ease of adding and deleting data.

3.2. Procedures And Methods

run(): Main method of the class that starts the execution of the program. Displays a menu of options and executes different actions depending on the option selected by the user.

menu(): Method that displays the options menu for the general queue.

inputOption(): Static method that reads the user-entered option from standard input and returns it as a string.

inputValue(): Static method that reads a numeric value entered by the user and returns it as an integer. It uses a try-catch block to validate user input and ensure that it is a valid integer.

executeGeneralQueue(): Method that executes operations related to the general queue. It uses a switch-case to execute different actions depending on the option selected by the user.
newEntrance(): Private method that creates and returns a new Entrance instance prompting the user to enter the necessary data.
menuEmergency(): Method that displays the options menu for the emergency queue.
executeEmergencyQueue(): Method that executes operations related to the emergency queue. It uses a switch-case to execute different actions depending on the option selected by the user.
newEmergencyEntrance(): Private method that creates and returns a new Emergency Entrance instance, prompting the user to enter the necessary data, including the emergency level.
addExampleEntrances(): Method that adds example entries to queues.
printSpecialistQueues(): Method that prints the specialist queues, showing the size of each queue and the entry data.

4. DATA STRUCTURES TEST SUMMARY

For the GeneralQueueTest class:
testAddFirst(): Tests the add operation to the beginning of the general queue.
testAddLast(): Tests the add operation to the end of the general queue.
testRemoveFirst(): Tests the operation of removing the first element from the general queue.
testRemoveLast(): Tests the operation of removing the last element from the general queue.
testFirst(): Tests the first() method to ensure that it returns the first element of the general queue correctly.

For the EmergencyQueueTest class:
testAddFirst(): Tests the add operation to the beginning of the emergency queue.
testAddLast(): Tests the add operation to the end of the emergency queue.

testRemoveFirst(): Tests the operation of removing the first element of the emergency queue.
testRemoveLast(): Tests the operation of removing the last element from the emergency queue.
testFirst(): Tests the first() method to ensure that it returns the first element of the emergency queue correctly.
testLast(): Tests the last() method to make sure it returns the last item in the emergency queue correctly.
testIsEmpty(): Tests the isEmpty() method to ensure that it returns true if the emergency queue is empty and false if it is not.
testSize(): Test the size() method to make sure it returns the correct size of the emergency queue.