**Alria Sianoya**

*Activity: Queue*

#include <iostream>

#include <queue>

#include <string>

#include <stdexcept>

#include <iomanip>

using namespace std;

// Class to represent a Person

class Person {

public:

Person(string name, int ticketNumber) : name(name), ticketNumber(ticketNumber) {}

string getName() const {

return name;

}

int getTicketNumber() const {

return ticketNumber;

}

private:

string name;

int ticketNumber;

};

// Class to represent a Queue of Persons

class TicketQueue {

public:

void enqueue(const string& name) {

int ticketNumber = ++ticketCounter; // Increment ticket counter for new ticket number

Person newPerson(name, ticketNumber);

personQueue.push(newPerson);

cout << name << " added to the queue with Ticket #" << setw(3) << setfill('0') << ticketNumber << endl;

}

void dequeue() {

if (personQueue.empty()) {

throw runtime\_error("Queue is empty. Cannot dequeue.");

}

Person person = personQueue.front();

personQueue.pop();

cout << "Dequeue: " << person.getName() << " received a ticket (Ticket #" << setw(3) << setfill('0') << person.getTicketNumber() << ")" << endl;

}

void displayNext() const {

if (personQueue.empty()) {

cout << "Queue is empty." << endl;

return;

}

Person person = personQueue.front();

cout << "Next in line: " << person.getName() << " (Ticket #" << setw(3) << setfill('0') << person.getTicketNumber() << ")" << endl;

}

int getPosition(const string& name) const {

int position = 1;

queue<Person> tempQueue = personQueue; // Copy to preserve the original queue

while (!tempQueue.empty()) {

if (tempQueue.front().getName() == name) {

return position;

}

tempQueue.pop();

position++;

}

return -1; // Not found

}

int size() const {

return personQueue.size();

}

private:

queue<Person> personQueue;

int ticketCounter = 0; // To generate ticket numbers

};

// Main function

int main() {

TicketQueue ticketQueue;

string name;

char option;

cout << "Welcome to Olivia Rodrigo's Concert Ticketing System!" << endl;

while (true) {

cout << "\n1. Enqueue a person" << endl;

cout << "2. Check your position in the queue" << endl;

cout << "3. Dequeue a person" << endl;

cout << "4. Display the next person in the queue" << endl;

cout << "5. Exit" << endl;

cout << "Choose an option: ";

cin >> option;

cin.ignore(); // Clear the newline character from the input buffer

switch (option) {

case '1':

cout << "Enter the name: ";

getline(cin, name);

ticketQueue.enqueue(name);

cout << "Queue size: " << ticketQueue.size() << endl;

break;

case '2':

cout << "Enter your name: ";

getline(cin, name);

int position;

position = ticketQueue.getPosition(name);

if (position != -1) {

cout << name << " is currently at position " << position << " in the queue." << endl;

} else {

cout << name << " is not in the queue." << endl;

}

break;

case '3':

try {

ticketQueue.dequeue();

cout << "Queue size: " << ticketQueue.size() << endl;

} catch (const runtime\_error& e) {

cout << "Error: " << e.what() << endl;

}

break;

case '4':

ticketQueue.displayNext();

break;

case '5':

cout << "Exiting the system. Thank you!" << endl;

return 0;

default:

cout << "Invalid option. Please try again." << endl;

}

}

return 0;

}