#pragma once

#include "stdafx.h"

#include "Accounts.h"

#include "Booking.h"

#include <iostream>

#include <string>

#include <array>

#include <vector>

#include <conio.h>

#include <ctime>

using namespace std;

// START OF BOOKING BEHAVIOURS

// Booking constructor

Booking::Booking()

{

iID = 0; // Assigns starting booking ID to 0

fCost = 25; // Assigns all bookings to a cost of £25

}

// Booking destructor

Booking::~Booking()

{

}

// Gets confirmation on the £25 booking cost for an appointment

bool& Booking::BookingCostConfirmation(bool &bCheckPass, \_\_int16 &iChoicePass)

{

cout << "There is a " << fCost << " fee for booking an appointment on our system\n";

cout << "1: Accept, 2: Decline\n";

// Get customer confirmation choice

cin >> iChoicePass;

if (iChoicePass == 1)

{

bCheckPass = true; // Accept

}

else

{

bCheckPass = false; // Decline

}

return bCheckPass;

}

// Get and store booking details

void Booking::GetBookingDetails(const \_\_int16 &iDoctorNumberPass, const \_\_int16 &iSurgeryIDPass, const string &sDoctorNamePass, const string &sCustomerNamePass, const string &sCustomerAilmentPass, const string &sBookingDatePass)

{

iDoctorID = iDoctorNumberPass; // Doctor ID for booking

sDoctorName = sDoctorNamePass; // Doctor name for booking

sCustomerName = sCustomerNamePass; // Customer name for booking

sBookingDate = sBookingDatePass; // Booking date for booking

sCustomerAilment = sCustomerAilmentPass; // Customer ailment for booking

}

// Review all bookings within the system using customer and doctor details

void Booking::ReviewAllBookings()

{

cout << "Booking ID: " << iID << ", Doctor ID: " << iDoctorID << ", Doctor Name: " << sDoctorName << ", Customer Name: "

<< sCustomerName << ", Ailment: " << sCustomerAilment << ", Cost: " << fCost << ", Booked: " << sBookingDate << '\n';

}

// Returns doctor ID back to add 1 appointment slot back to the correct doctor tied to the last booking in the system

\_\_int16 Booking::ReturnDoctorID()

{

return iDoctorID;

}

// END OF BOOKING BEHAVIOURS