using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

class Program

{

static void Main(string[] args)

{

AirlineCoordinator airCoor = new AirlineCoordinator();

int numFlights = 0;

bool success = true;

int key = getValidChoice();

mainMenu();

while (key != 7)

{

switch (key)

{

case 1: //add a flight

int flightNum;

numFlights++;

int maxSeats;

string destination;

string origin = "";

Console.Clear();

Console.WriteLine("----------Add Flight-----------");

Console.Write("Please enter the flight number: ");

while (!int.TryParse(Console.ReadLine(), out flightNum))

{

Console.Write("Please enter a valid flight number: ");

}

Console.Write("Please enter the maximum number of seats: ");

while (!int.TryParse(Console.ReadLine(), out maxSeats))

{

Console.Write("Please enter a valid number of seats: ");

}

Console.Write("Please enter the port of origin: ");

origin = Console.ReadLine();

Console.Write("Please enter the destination port: ");

destination = Console.ReadLine();

success = airCoor.addFlight(flightNum, origin, destination, maxSeats);

Console.WriteLine("Flight successfully added..");

numFlights++;

Console.WriteLine();

Console.WriteLine("Press any key to return to the main menu.");

Console.ReadKey();

break;

case 2: //add a customer

string fName, lName, phone;

Console.Clear();

Console.WriteLine("---------Add Customer-----------");

Console.Write("Please enter the customer's first name: ");

fName = Console.ReadLine();

Console.Write("Please enter the customer's last name: ");

lName = Console.ReadLine();

Console.Write("Please enter the customer's phone: ");

phone = Console.ReadLine();

success = airCoor.addCustomer(fName, lName, phone);

if (success)

{

Console.WriteLine("Customer successfully added..");

}

else

{

Console.WriteLine("Customer was not added..");

}

Console.WriteLine();

Console.WriteLine("Press any key to return to the main menu.");

Console.ReadKey();

break;

case 3: //display flight list

Console.Clear();

Console.WriteLine("Flight List: ");

string s = airCoor.flightList();

Console.WriteLine(s + "\n");

Console.WriteLine("Press any key to return to the main menu.");

Console.ReadKey();

break;

case 4: //displaying customer list

Console.Clear();

Console.WriteLine("Customer List:");

Console.WriteLine("Number\tName\t\tPhone");

string cus = airCoor.customerList();

Console.WriteLine(cus + "\n");

Console.WriteLine("Press any key to return to the main menu.");

Console.ReadKey();

break;

case 5: //delete a customer

Console.Clear();

int custId;

Console.WriteLine("Customer List:");

Console.WriteLine("Number\tName\t\tPhone");

cus = airCoor.customerList();

Console.WriteLine(cus + "\n");

Console.Write("Enter customer id to delete: ");

while(!int.TryParse(Console.ReadLine(), out custId))

{

Console.Write("Please enter a valid id: ");

}

success = airCoor.deleteCustomer(custId);

if (success)

{

Console.WriteLine("Customer with id " + custId + " was deleted..");

}

else

{

Console.WriteLine("Customer with id " + custId + " was not found..");

}

Console.WriteLine();

Console.WriteLine("Press any key to return to main menu.");

Console.ReadKey();

break;

case 6: //delete a flight

Console.Clear();

int flightId;

Console.WriteLine("Flight List: ");

s = airCoor.flightList();

Console.WriteLine(s + "\n");

Console.Write("Enter flight number to delete: ");

while(!int.TryParse(Console.ReadLine(), out flightId))

{

Console.Write("Please enter a valid flight number: ");

}

success = airCoor.deleteFlight(flightId);

if(success)

{

Console.WriteLine("Flight with id " + flightId + " was deleted..");

}

else

{

Console.WriteLine("Flight with id " + flightId + " was not found..");

}

Console.WriteLine("\nPress any key to return to main menu.");

Console.ReadKey();

break;

}

key = getValidChoice();

}

Console.WriteLine("Thank you for choosing XYZ Airline!\nPress any button to exit");

Console.ReadKey();

}

public static void mainMenu()

{

Console.Clear();

Console.WriteLine("Welcome to XYZ Airline Limited.\nPlease select a choice from the menu below:\n");

Console.WriteLine("1: Add Flight");

Console.WriteLine("2: Add Customer");

Console.WriteLine("3: View Flights");

Console.WriteLine("4: View Customers");

Console.WriteLine("5: Delete Customer");

Console.WriteLine("6: Delete Flight");

Console.WriteLine("7: Exit");

}

public static int getValidChoice()

{

int key;

mainMenu();

while(!int.TryParse(Console.ReadLine(), out key) || key < 1 || key > 7)

{

mainMenu();

Console.WriteLine("Please select one of the options");

}

return key;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

class FlightManager

{

private int maxFlights;

private static int numFlights;

private Flight[] fList;

public FlightManager(int maxFlt)

{

maxFlights = maxFlt;

numFlights = 0;

fList = new Flight[maxFlights];

}

//getters

public int getMaxFlights() { return maxFlights; }

public int getNumFlights() { return numFlights; }

class getFlight : Flight

{

private int flightNo;

public getFlight(int flNo, string or, string det, int mSeats, int flNum) : base(flNo, or, det, mSeats)

{ flightNo = flNum; }

}

public bool addFlight(int fltNo, string origin, string dest, int maxSeats)

{

if (numFlights >= maxFlights) return false;

Flight f = new Flight(fltNo, origin, dest, maxSeats);

fList[numFlights] = f;

numFlights++;

return true;

}

protected int findFlight(int flightNo)

{

for(int i = 0; i < numFlights; i++) { if (fList[i].getFlightNumber() == flightNo) return i; }

return -1;

}

public bool flightExist(int flightNo)

{

if (findFlight(flightNo) > 0) return true;

return false;

}

public bool deleteFlight(int flightNo)

{

int position = -1;

for (int i = 0; i < numFlights; i++)

{

if (fList[i].getFlightNumber() == flightNo)

position = i;

}

if (position == -1) return false;

for (int x = position; x < numFlights - 1; x++) {

fList[x] = fList[x + 1];

fList[x + 1] = null;

}

numFlights--;

return true;

}

public string getFlightList()

{

string str = "";

for(int i = 0; i< numFlights; i++)

{

str += fList[i].getFlightNumber() + " from "

+ fList[i].getOrigin() + " to " + fList[i].getDestination()+ Environment.NewLine;

}

return str;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

class Customer

{

private int customerId;

private string firstName;

private string lastName;

private string phone;

public Customer(int id, string fn, string ln, string ph)

{

customerId = id;

firstName = fn;

lastName = ln;

phone = ph;

}

//getters

public int getCustomerId() { return customerId; }

public string getFirstName() { return firstName; }

public string getLastName() { return lastName; }

public string getPhone() { return phone; }

public string toString()

{

string s = "Customer Id: " + customerId + "\nFirst Name: " + firstName + "\nLast Name: " + lastName + "\nPhone Number: " + phone;

return s;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

class AirlineCoordinator

{

private FlightManager flightMan;

private CustomerManager customerMan;

public int numFlights;

public int numCust;

public AirlineCoordinator()

{

customerMan = new CustomerManager(100, 100);

flightMan = new FlightManager(10);

}

public bool addFlight(int flNo, string origin, string dest, int mSeats)

{

numFlights++;

bool success = flightMan.addFlight(flNo, origin, dest, mSeats);

if (success) { return true; }

return false;

}

public bool addCustomer(string fName, string lName, string phone)

{

numCust++;

bool success = customerMan.addCustomer(fName, lName, phone);

if (success) { return true; }

return false;

}

public string flightList()

{

string s = "";

for (int x = 0; x < numFlights; x++)

{

s = flightMan.getFlightList();

}

return s;

}

public string customerList()

{

string s = "";

for (int x = 0; x < numCust; x++)

{

s = customerMan.getCustomerList();

}

return s;

}

public bool deleteCustomer(int cId)

{

if (customerMan.deleteCustomer(cId)) return true;

return false;

}

public bool deleteFlight(int flightNo)

{

if (flightMan.deleteFlight(flightNo)) return true;

return false;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

class CustomerManager

{

private int maxCustomers;

private static int numCustomers = 0;

private static int currentCustNo;

public Customer[] customerList;

public CustomerManager(int seed, int maxCust)

{

maxCustomers = maxCust;

currentCustNo = seed;

numCustomers = 0;

customerList = new Customer[maxCust];

}

public int getMaxCustomer() { return maxCustomers; }

public int getNumCustomer() { return numCustomers; }

public bool addCustomer(string fN, string lN, string ph)

{

if (numCustomers >= maxCustomers)

{ return false; }

Customer c = new Customer(currentCustNo, fN, lN, ph);

currentCustNo++;

customerList[numCustomers] = c;

numCustomers++;

return true;

}

protected int findCustomer(int cId)

{

for (int x = 0; x < numCustomers; x++)

{

if (customerList[x].getCustomerId() == cId)

{ return x; }

}

return -1;

}

public bool customerExist(int cId)

{

int place = findCustomer(cId);

if (place > 0)

return true;

return false;

}

public bool deleteCustomer(int cId)

{

int position = -1;

for (int i = 0; i < numCustomers; i++)

{

if (customerList[i].getCustomerId() == cId)

position = i;

}

if (position == -1) return false;

for (int x = position; x < numCustomers-1; x++) {

customerList[x] = customerList[x + 1];

customerList[x + 1] = null;

}

numCustomers--;

return true;

}

public string getCustomerList()

{

string fullList = "";

for (int i = 0; i < numCustomers; i++)

{

fullList += customerList[i].getCustomerId() + "\t"

+ customerList[i].getFirstName() + " " + customerList[i].getLastName() +

"\t" + customerList[i].getPhone() + Environment.NewLine;

}

return fullList;

}

class getCustomer : Customer

{

private int curCusId;

public getCustomer(int id, string fn, string ln, string ph, int cci) : base(id, fn, ln, ph)

{curCusId = cci;}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assignment1

{

class Flight

{

private int flightNumber;

private string origin;

private string destination;

private int maxSeats;

private static int numPassengers;

private Customer[] customerList;

public Flight(int flNo, string or, string det, int mSeats)

{

flightNumber = flNo;

origin = or;

destination = det;

maxSeats = mSeats;

customerList = new Customer[mSeats];

}

//getters

public int getFlightNumber() { return flightNumber;}

public string getOrigin() { return origin; }

public string getDestination() { return destination; }

public int getMaxSeats() { return maxSeats; }

public int getNumPassengers() { return numPassengers; }

public string getPassengerList()

{

string str = "";

for (int i = 0; i < numPassengers; i++) { str += customerList[i]; }

return str;

}

public bool addPassenger(Customer cust)

{

if (numPassengers >= maxSeats) return false;

Customer c = customerList[numPassengers];

numPassengers++;

return true;

}

public bool findPassenger(int custId)

{

for (int i = 0; i < numPassengers; i++)

{ if (customerList[i].getCustomerId() == custId) return true; }

return false;

}

public bool removeFlight(int cId)

{

int position = -1;

for (int i = 0; i < numPassengers; i++)

{

if (customerList[i].getCustomerId() == cId)

position = i;

}

if (position == -1) return false;

customerList[position] = customerList[numPassengers - 1];

numPassengers--;

return true;

}

public string toString()

{

string s = flightNumber + "\tfrom" + origin + "\tto" + destination;

return s;

}

}

}