Assignment 2 - Event Management

Back-end / Console-based classes

**Group Participants:**

Student 1: Paolo Tous -- 101325245

Student 2: Robertha Alvarez Diaz -- 101236645

Student 3: Jai Kumar -- 101236674

Student 4: Kanav Bhatia -- 101278920

**Code for Date class:**

using System;

using System.Collections.Generic;

using System.Text;

namespace Event\_Management

{

class Date

{

public int year;

public int month; // 1 Jan, 2, Feb....

public int day; // no error checking required for day

public int hour; //24 hour format

public int minute; //

public Date(int day, int month, int year, int hour, int minute)

{

this.day = day;

this.year = year;

this.month = month;

this.hour = hour;

this.minute = minute;

}

public string viewLongMonth()

{

switch (month)

{

case 1:

return "January";

case 2:

return "February";

case 3:

return "March";

case 4:

return "April";

case 5:

return "May";

case 6:

return "June";

case 7:

return "July";

case 8:

return "August";

case 9:

return "September";

case 10:

return "October";

case 11:

return "November";

case 12:

return "December"; ;

default:

return "-";

}

}

public string viewShortMonth()

{

switch (month)

{

case 1:

return "Jan";

case 2:

return "Feb";

case 3:

return "Mar";

case 4:

return "Apr";

case 5:

return "May";

case 6:

return "Jun";

case 7:

return "July";

case 8:

return "Aug";

case 9:

return "Sep";

case 10:

return "Oct";

case 11:

return "Nov";

case 12:

return "Dec"; ;

default:

return "-";

}

}

public override string ToString()

{

string s = day + " " + viewShortMonth() + " " + year;

s += " at " + hour + ":" + minute;

return s;

}

}

}

**Code for Event Class**

using System;

using System.Collections.Generic;

using System.Text;

namespace Event\_Management

{

class Event

{

private int eventId;

private string eventName;

private string venue;

private Date eventDate;

private int maxAttendees;

private int numAttendees;

private string dateCreated;

private Customer[] attendeeList;

public Event(int eventId, string name, string venue, Date eventDate, int maxAttendees)

{

this.eventId = eventId;

this.eventName = name;

this.venue = venue;

this.eventDate = eventDate;

this.maxAttendees = maxAttendees;

numAttendees = 0;

attendeeList = new Customer[maxAttendees];

dateCreated = DateTime.Now.ToString(@"MM\/dd\/yyyy h\:mm tt");

}

public int getEventId() { return eventId; }

public string getEventName() { return eventName; }

public string getVenue() { return venue; }

public Date getEventDate() { return eventDate; }

public int getEventDay() { return eventDate.day; }

public int getEventMonth() { return eventDate.month; }

public int getEventYear() { return eventDate.year; }

public int getMaxAttendees() { return maxAttendees; }

public int getNumAttendees() { return numAttendees; }

public bool addAttendee(Customer c)

{

if (numAttendees >= maxAttendees) { return false; }

attendeeList[numAttendees] = c;

numAttendees++;

c.setNumBookings(c.getNumBookings() + 1);

return true;

}

private int findAttendee(int custId)

{

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

{

if (attendeeList[x].getId() == custId)

return x;

}

return -1;

}

public bool removeAttendee(int custId)

{

int loc = findAttendee(custId);

if (loc == -1) return false;

attendeeList[loc] = attendeeList[numAttendees - 1];

numAttendees--;

return true;

}

public bool attendeeExist(int custId)

{

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

{

if(attendeeList[i].getId() == custId)

{

return true;

}

}

return false;

}

public string getAttendees()

{

string s = "\nCustomers registered :";

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

{

s = s + "\n" + attendeeList[x].getFirstName() + " " + attendeeList[x].getLastName();

s += "\nCustomer ID: " + attendeeList[x].getId();

}

return s;

}

public override string ToString()

{

string s = "Event: " + eventId + "\nName: " + eventName;

s = s + "\nVenue: " + venue;

s = s + "\nDate:" + eventDate;

s = s + "\nRegistered Attendees:" + numAttendees;

s = s + "\nAvailable spaces:" + (maxAttendees - numAttendees);

s = s + getAttendees();

return s;

}

}

}

**Code for EventManager Class**

using System;

using System.Collections.Generic;

using System.Text;

namespace Event\_Management

{

class EventManager

{

private static int currentEventId;

private int maxEvents;

private int numEvents;

private Event[] eventList;

public EventManager(int idSeed, int max)

{

currentEventId = idSeed;

maxEvents = max;

numEvents = 0;

eventList = new Event[maxEvents];

}

public bool addEvent(string name, string venue, Date eventDate, int maxAttendees)

{

if (numEvents >= maxEvents) { return false; }

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

{

if (eventList[i].getVenue() == venue && eventList[i].getEventDay() == eventDate.day

&& eventList[i].getEventMonth() == eventDate.month && eventList[i].getEventYear() == eventDate.year

)

{

Console.WriteLine("Error");

return false;

}

}

Event e = new Event(currentEventId, name, venue, eventDate, maxAttendees);

eventList[numEvents] = e;

numEvents++;

currentEventId++;

return true;

}

private int findEvent(int eid)

{

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

{

if (eventList[x].getEventId() == eid)

return x;

}

return -1;

}

public int getNumEvents()

{

return numEvents;

}

public bool eventExists(int eid)

{

int loc = findEvent(eid);

if (loc == -1) { return false; }

return true;

}

public Event getEvent(int eid)

{

int loc = findEvent(eid);

if (loc == -1) { return null; }

return eventList[loc];

}

public bool deleteEvent(int eid)

{

int loc = findEvent(eid);

if (loc == -1) { return false; }

eventList[loc] = eventList[numEvents - 1];

numEvents--;

return true;

}

public string getEventInfo(int eid)

{

int loc = findEvent(eid);

if (loc == -1) { return "There is no event with id " + eid + "."; }

return eventList[loc].ToString();

}

public string getEventList()

{

string s = "Event List:";

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

{

s = s + "\n" + eventList[x].getEventId() + " \t " + eventList[x].getEventName() + " \t " + eventList[x].getVenue();

}

return s;

}

}

}

**Code for Customer Class**

using System;

using System.Collections.Generic;

using System.Text;

namespace Event\_Management

{

class Customer

{

private int customerId;

private string firstName;

private string lastName;

private string phone;

private int bookings;

public Customer(int cId, string fname, string lname, string ph)

{

bookings = 0;

customerId = cId;

firstName = fname;

lastName = lname;

phone = ph;

}

public int getId() { return customerId; }

public string getFirstName() { return firstName; }

public string getLastName() { return lastName; }

public string getPhone() { return phone; }

public int getNumBookings() { return bookings; }

public void setNumBookings(int bookings)

{

this.bookings = bookings;

}

public override string ToString()

{

string s = "Customer ID: " + customerId;

s = s + "\nName: " + firstName + " " + lastName;

s = s + "\nPhone: " + phone;

s = s + "\nBookings: " + bookings;

return s;

}

}

}

**Code for CustomerManager Class**

using System;

using System.Collections.Generic;

using System.Text;

namespace Event\_Management

{

class CustomerManager

{

private static int currentCustNumber;

private int maxNumCustomers;

private int numCustomers;

private Customer[] customerList;

public CustomerManager(int ccn, int max)

{

currentCustNumber = ccn;

maxNumCustomers = max;

numCustomers = 0;

customerList = new Customer[maxNumCustomers];

}

public bool addCustomer(string fn, string ln, string ph)

{

if (numCustomers >= maxNumCustomers) { return false; }

Customer c = new Customer(currentCustNumber, fn, ln, ph);

currentCustNumber++;

customerList[numCustomers] = c;

numCustomers++;

return true;

}

public int findCustomer(int cid)

{

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

{

if (customerList[x].getId() == cid)

return x;

}

return -1;

}

public bool customerExist(int cid)

{

int loc = findCustomer(cid);

if (loc == -1) { return false; }

return true;

}

public Customer getCustomer(int cid)

{

int loc = findCustomer(cid);

if (loc == -1) { return null; }

return customerList[loc];

}

public string getCustomerInfo(int cid)

{

int loc = findCustomer(cid);

if (loc == -1) { return "There is no customer with id " + cid + "."; }

return customerList[loc].ToString();

}

public bool deleteCustomer(int cid)

{

int loc = findCustomer(cid);

if (loc == -1) { return false; }

customerList[loc] = customerList[numCustomers - 1];

numCustomers--;

return true;

}

public string getCustomerList()

{

string s = "Customer List:";

s = s + "\nNumber \tName \t \tPhone";

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

{

s = s + "\n" + customerList[x].getId() + "\t" + customerList[x].getFirstName() + "\t" + customerList[x].getLastName() + "\t" + customerList[x].getPhone();

}

return s;

}

}

}

**Code for EventCoordinator Class**

using System;

using System.Collections.Generic;

using System.Text;

namespace Event\_Management

{

class EventCoordinator

{

CustomerManager custMan;

EventManager eventMan;

RSVP[] regs;

private static int ticketID;

public EventCoordinator(int custIdSeed, int maxCust, int eventIdSeed, int maxEvents)

{

custMan = new CustomerManager(custIdSeed, maxCust);

eventMan = new EventManager(eventIdSeed, maxEvents);

regs = new RSVP[maxCust \* maxEvents];

ticketID = 1;

}

//RSVP related methods

public bool register(int cid, int eid) //adds a new RSVP

{

regs[ticketID] = new RSVP(eventMan, custMan, ticketID);

regs[ticketID].setRegID(ticketID); //use the unique generated ID and assign it to RSVP

if (regs[ticketID].registerID(cid, eid))

{

ticketID++;

return true;

}

return false;

}

public string viewRegs() //view all RSVPs

{

string s = "";

for (int i = 1; i < ticketID; i++)

{

s += regs[i].viewRSVP();

s += "\n-----------------\n";

}

return s;

}

//customer related methods

public bool addCustomer(string fname, string lname, string phone)

{

return custMan.addCustomer(fname, lname, phone);

}

public string customerList()

{

return custMan.getCustomerList();

}

public string getCustomerInfoById(int id)

{

return custMan.getCustomerInfo(id);

}

public bool deleteCustomer(int id)

{

return custMan.deleteCustomer(id);

}

// Event related methods

public bool addEvent(string name, string venue, Date eventDate, int maxAttendee)

{

return eventMan.addEvent(name, venue, eventDate, maxAttendee);

}

public string eventList()

{

return eventMan.getEventList();

}

public string getEventInfoById(int id)

{

return eventMan.getEventInfo(id);

}

}

}

**Code for RSVP Class**

using System;

using System.Collections.Generic;

using System.Text;

namespace Event\_Management

{

class RSVP

{

private Event e;

private Customer c;

private EventManager eventMan;

private CustomerManager custMan;

private int regID;

private string date;

public RSVP(EventManager eventMan, CustomerManager custMan, int regID)

{

this.eventMan = eventMan;

this.custMan = custMan;

this.regID = regID;

}

public void setRegID(int regID)

{

this.regID = regID;

}

public int getRegID()

{

return regID;

}

//registers a customer to a specific event

public bool registerID(int cid, int eid)

{

if (custMan.customerExist(cid) && eventMan.eventExists(eid))

{

e = eventMan.getEvent(eid); //e will reference the event object returned here

c = custMan.getCustomer(cid); //c will reference the customer object

if (e.getNumAttendees() < e.getMaxAttendees() && e.attendeeExist(cid) == false)

{

e.addAttendee(custMan.getCustomer(cid));

Console.WriteLine("You may find below information of your ticket: ");

Console.WriteLine(generateTicket(e, c));

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

Console.WriteLine("RSVP has been successfully made");

Console.WriteLine("Press any key to continue ... ");

Console.ReadLine();

return true;

}

}

Console.WriteLine("RSVP Failed");

Console.WriteLine("Press any key to continue ... ");

Console.ReadLine();

return false;

}

public string generateTicket(Event e, Customer c)

{

date = DateTime.Now.ToString(@"MM\/dd\/yyyy h\:mm tt");

string s = "Ticket Info: \n";

s += "Date Purchased: " + date;

s += "\nRSVP ID: " + regID + "\n";

s += "------------------------";

s += "\nEvent Info: \n" + e.ToString() + "\n";

s += "------------------------";

s += "\nCustomer Info: \n" + c.ToString();

return s;

}

public string viewRSVP()

{

string s = "Date: " + date;

s += "\nRSVP Number: " + regID;

s += "\nCustomer Name: " + c.getFirstName() + " " + c.getLastName();

s += "\nEvent ID: " + e.getEventId();

return s;

}

}

}

**Code for Main**

using System;

namespace Event\_Management

{

class Program

{

static EventCoordinator eCoord;

public static void deleteCustomer()

{

int id;

Console.Clear();

Console.WriteLine(eCoord.customerList());

Console.Write("Please enter a customer id to delete:");

id = getIntChoice();

if (eCoord.deleteCustomer(id))

{

Console.WriteLine("Customer with id {0} deleted..", id);

}

else

{

Console.WriteLine("Customer with id {0} was not found..", id);

}

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

Console.ReadKey();

}

public static void viewCustomers()

{

Console.Clear();

Console.WriteLine(eCoord.customerList());

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

Console.ReadKey();

}

public static void viewSpecificCustomer()

{

int id;

string cust;

Console.Clear();

Console.WriteLine(eCoord.customerList());

Console.Write("Please enter a customer id to View:");

id = getIntChoice();

Console.Clear();

cust = eCoord.getCustomerInfoById(id);

Console.WriteLine(cust);

Console.WriteLine("\nPress any key to continue return to the previous menu.");

Console.ReadKey();

}

public static void addCustomer()

{

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();

if (eCoord.addCustomer(fname, lname, phone))

{

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

}

else

{

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

}

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

Console.ReadKey();

}

public static void registerCustomer()

{

int cid = 0;

int eid = 0;

bool validEvent = false;

bool validCustomer = false;

//show the list of customers and events

Console.Clear();

Console.WriteLine("-----------List of Customers----------");

Console.WriteLine(eCoord.customerList());

Console.WriteLine("-----------List of Events-------------");

Console.WriteLine(eCoord.eventList());

Console.WriteLine("-----------RSVP Event-----------------");

//prompt the user and use both ids to call the register method via the EventCoordinator object

while (validEvent == false)

{

Console.WriteLine("Please enter the Event ID: ");

if (Int32.TryParse(Console.ReadLine(), out eid))

{

validEvent = true;

}

}

while (validCustomer == false)

{

Console.WriteLine("Please enter the Customer ID: ");

if (Int32.TryParse(Console.ReadLine(), out cid))

{

validCustomer = true;

}

}

if (validCustomer == true && validEvent == true)

{

eCoord.register(cid, eid);

}

}

public static void registrations()

{

//this will view all processed RSVPs

Console.Clear();

Console.WriteLine("-----------List of RVSP----------");

Console.WriteLine(eCoord.viewRegs());

Console.WriteLine("Press any key to continue ...");

Console.ReadLine();

}

public static void addEvent()

{

string eventName, venue;

Date eventDate;

int maxAttendees;

int day, month, year, hour, minute;

Console.Clear();

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

Console.Write("Please enter the name of the Event:");

eventName = Console.ReadLine();

Console.Write("Please enter venue for the event:");

venue = Console.ReadLine();

Console.Write("Please enter the Day of the event:");

day = getIntChoice();

Console.Write("Please enter the Month of the event (as an integer):");

month = getIntChoice();

Console.Write("Please enter the Year of the event:");

year = getIntChoice();

Console.Write("Please enter the Hour the event starts in 24 hour format:");

hour = getIntChoice();

Console.Write("Please enter the Minute the event starts:");

minute = getIntChoice();

eventDate = new Date(day, month, year, hour, minute);

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

maxAttendees = getIntChoice();

if (eCoord.addEvent(eventName, venue, eventDate, maxAttendees))

{

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

}

else

{

Console.WriteLine("The event was not added..");

}

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

Console.ReadKey();

}

public static void viewEvents()

{

Console.Clear();

Console.WriteLine(eCoord.eventList());

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

Console.ReadKey();

}

public static void viewSpecificEvent()

{

int id;

string ev;

Console.Clear();

Console.WriteLine(eCoord.eventList());

Console.Write("Please enter an event id to View:");

id = getIntChoice();

Console.Clear();

ev = eCoord.getEventInfoById(id);

Console.WriteLine(ev);

Console.WriteLine("\nPress any key to continue return to the previous menu.");

Console.ReadKey();

}

public static string customerMenu()

{

string s = "Andrew's Event Management Limited.\n";

s += "Customer Menu.\n";

s += "Please select a choice from the menu below:\n";

s += "1: Add Customer \n";

s += "2: View Customers \n";

s += "3: View Customer Details \n";

s += "4: Delete Customer\n";

s += "5: Return to the main menu.";

return s;

}

public static string eventMenu()

{

string s = "Andrew's Event Management Limited.\n";

s += "Event Menu.\n";

s += "Please select a choice from the menu below:\n";

s += "1: Add Event \n";

s += "2: View all Events \n";

s += "3: View Event Details \n";

s += "4: Return to the main menu.";

return s;

}

public static string registrationMenu()

{

string s = "Star Wars Event Management Limited.\n";

s += "Event Registration Menu.\n";

s += "Please select a choice from the menu below:\n";

s += "1: RSVP for event \n";

s += "2: View RSVPs \n";

s += "3: Return to the main menu.";

return s;

}

public static string mainMenu()

{

string s = "Star Wars Event Management Limited.\n";

s += "Please select a choice from the menu below:\n";

s += "1: Customer Options \n";

s += "2: Event Options \n";

s += "3: RSVP for Event \n";

s += "4: Exit";

return s;

}

public static void runCustomerMenu()

{

string menu = customerMenu();

int choice = getValidChoice(5, menu);

while (choice != 5)

{

if (choice == 1) { addCustomer(); }

if (choice == 2) { viewCustomers(); }

if (choice == 3) { viewSpecificCustomer(); }

if (choice == 4) { deleteCustomer(); }

choice = getValidChoice(5, menu);

}

}

public static void runEventMenu()

{

string menu = eventMenu();

int choice = getValidChoice(4, menu);

while (choice != 4)

{

if (choice == 1) { addEvent(); }

if (choice == 2) { viewEvents(); }

if (choice == 3) { viewSpecificEvent(); }

choice = getValidChoice(4, menu);

}

}

public static void runRegistrationMenu()

{

string menu = registrationMenu();

int choice = getValidChoice(3, menu);

while (choice != 3)

{

if (choice == 1) { registerCustomer(); }

if (choice == 2) { registrations(); }

choice = getValidChoice(3, menu);

}

}

public static int getValidChoice(int max, string menu)

{

int choice;

Console.Clear();

Console.WriteLine(menu);

while (!int.TryParse(Console.ReadLine(), out choice) || (choice < 1 || choice > max))

{

Console.Clear();

Console.WriteLine(menu);

Console.WriteLine("Please enter a valid choice:");

}

return choice;

}

public static int getIntChoice()

{

int choice;

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

{

Console.WriteLine("Please enter an integer:");

}

return choice;

}

public static void runProgram()

{

string menu = mainMenu();

int choice = getValidChoice(4, menu);

while (choice != 4)

{

if (choice == 1) { runCustomerMenu(); }

if (choice == 2) { runEventMenu(); }

if (choice == 3) { runRegistrationMenu(); }

choice = getValidChoice(4, menu);

}

}

static void Main(string[] args)

{

eCoord = new EventCoordinator(200, 1000, 101, 5000);

runProgram();

Console.WriteLine("Thank you for using Star Wars Event Management Limited System. ");

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

}