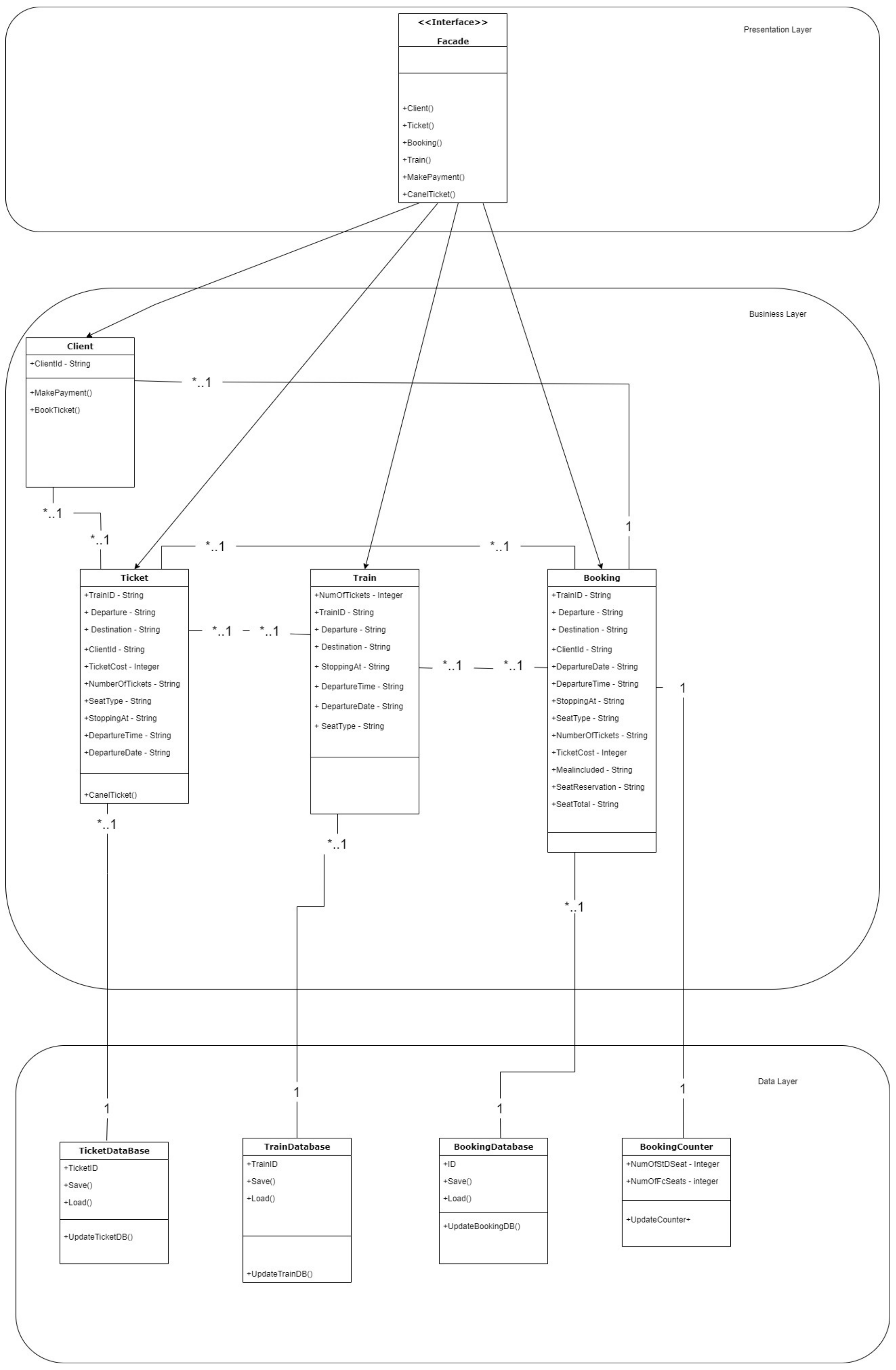
SET08119 2022-3 TR1 002-Object Oriented Software Development

Romeo Mcdonald-40441375

Class Diagram



SET08119 2022-3 TR1 002-Object Oriented Software Development
Romeo Mcdonald-40441375

Advantages of design patterns Essay

Design patterns are a form of reusable solutions for a design problem. They are describes as a set of interacting classes that make up a framework for a solution to a design problem in a specific context or field. In object oriented software development design patters suggest a solution to a problem or issue.
Design patterns can save a considerate amount of time and also increase the speed of the development process. Since programming requires the programmer to consider problems or issues that may occur in future implementation, the use of Design patters can help prevent these problems or issues and they can improve code readability.

Currently there are three different type of Design patterns available:

Creational Design Patters
Behavioural Design Patters
Structural Design Patterns

Creational Design Patterns

Creational Design patterns is a design pattern in software engineering that deals with class instantiation and the object creation mechanism, to create objects in an appropriate manner suitable to the situation.
The class creation pattern uses inheritance during the instantiation process and the object creational patter uses inheritance during the delegation process for instantiation.

There are six types of Creational Design Patterns:

Abstract Factory Pattern, Singleton Pattern, Factory Method Pattern, Prototype Pattern, Builder Pattern and Object Pool Pattern.

Behavioural Design Patters

Behavioural Design Patters in software engineering are design patterns that can identify the communication patterns that objects have in common, it then realizes these patterns with the intention of making the interactions between objects easier.

There are ten types of Behavioural Design Patters:
Chain of responsibility, Command Pattern, Interpreter Pattern, Iterator Pattern, Mediator Pattern, Memento Pattern, Observer Pattern, State Pattern, Strategy Pattern and Template Pattern.

Structural Design Pattern

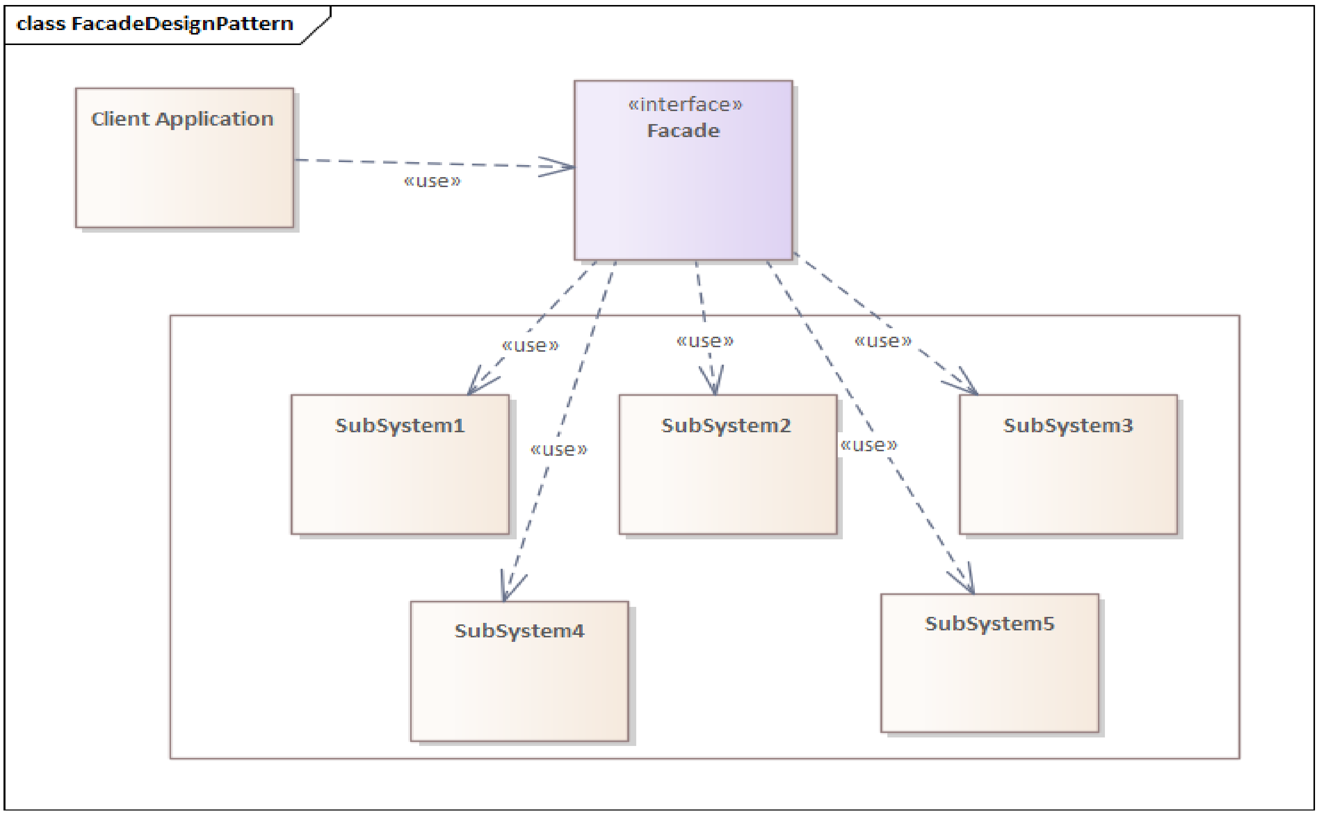
Structural Design Patters is a way to show how different classes or objects can be put together to make a larger structure that can achieve multiple goals altogether. Structural Design Patters demonstrate how unique parts of a system and be put together in a flexible and extensive manner.

There are seven types of Structural Design Patters:

Adapter Pattern, Bridge Pattern, Composite pattern, Decorator Pattern, Facade Pattern, Flyweight Pattern and Proxy Pattern.

Facade Design Pattern

In the Structural Design Pattern list Facade is an interesting pattern. Facade does literally as its name implies, it puts a mask over the complexity of the structure of the system. The structural design pattern Facade offers a simplified and limited interface to a complex system of classes, library or framework by hiding the complexity behind the interface.
Facade can diminish the complexity of the application, it can also helps to put undesired dependencies in one place. Facade has tree layers:
The Presentation layer where the interface is usually placed. The Business layer, where the hard code is usually placed. The Data layer, where the data for the system is stored.



Reference: https://dzone.com/articles/facade-design-pattern-in-java

SET08119 2022-3 TR1 002-Object Oriented Software Development
Romeo Mcdonald-40441375

C# code from my classes.

MainWindow.xaml.cs

```
using Business;
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace BookMyTrain
{

    //AUTHOR: Romeo Mcdonald
    //CLASS DISCRIPTION: Class conatains the values of booking
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        //decalring instances
        Train train = new Train();
        Ticket ticket = new Ticket();
        Booking booking = new Booking();
        Client client = new Client();

        //data manager for storing trains
        private Data _dataManager;

        public MainWindow()
        {
            InitializeComponent();
        }

        private void ComboBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {

        }

        //on button click create client, train, tikcet and booking
        private void Create_Bookng_Click(object sender, RoutedEventArgs e)
        {


            //creating client
            //if user does not input a name show message box
            if (client.ClientId == "" )
            {
                MessageBox.Show("Please Enter a Name");
            }
            // Else customer's first name is what was entered in the Name box
            else
            {
                client.ClientId = Client_Name_Box.Text;
            }



            //creating train
            train = new Train();
            train.Departure = cmbx_From.Text;
            train.Destination = cmbx_Destination.Text;
            train.TrainId = cmbx_Train.Text;
            train.DepartureTime = cmbx_Departure_Time.Text;
            train.DepartureDate = cmbx_Departure_Date.Text;
            train.SeatType = cmbx_Seat_Selection.Text;
            train.StoppingAt = cmbx_Stopping_At.Text;


            //creating ticket
            ticket = new Ticket();
            ticket.Departure = cmbx_From.Text;
            ticket.Destination = cmbx_Destination.Text;
            ticket.TrainId = cmbx_Train.Text;
            ticket.DepartureTime = cmbx_Departure_Time.Text;
            ticket.DepartureDate = cmbx_Departure_Date.Text;
            ticket.SeatType = cmbx_Seat_Selection.Text;
            ticket.StoppingAt = cmbx_Stopping_At.Text;
            ticket.ClientID = Client_Name_Box.Text;
            ticket.TicketCost = booking.TicketCalculator(train, booking);



            //Creating booking
            booking = new Booking();
            booking.Departure = cmbx_From.Text;
            booking.Destination = cmbx_Destination.Text;
            booking.TrainId = cmbx_Train.Text;
            booking.DepartureTime = cmbx_Departure_Time.Text;
            booking.DepartureDate = cmbx_Departure_Date.Text;
            booking.SeatType = cmbx_Seat_Selection.Text;
            booking.StoppingAt = cmbx_Stopping_At.Text;
            booking.ClientID = Client_Name_Box.Text;
            booking.TicketCost = booking.TicketCalculator(train, booking);
            booking.MealIncluded = cmbx_Meal_Selection.Text;
            booking.SeatReservation = cmbx_Reservation.Text;

            //Message box incase user doesnt input the number of tickets required
            //if number of tickets is blank show message
            if (num_of_tickets_Box.Text == "")
            {
                MessageBox.Show("Please enter the number of tickets as an Integer(eg
1, eg 2, eg 3)");
            }
            // Else users number of tickets is that which was entered in the num of
ticket box
            else
            {
                booking.NumOfTickets = Convert.ToInt32(num_of_tickets_Box.Text);
                ticket.NumOfTickets = Convert.ToInt32(num_of_tickets_Box.Text);
            }


            //code for diplay box
            tbx_Display.Text = "Booking information: "+ "\n" +
                                "\n" + "Train ID: " +train.TrainId + "           " +
"Client Name: " + booking.ClientID + "\n" +
                                "Departing From: " + train.Departure + "           "+
"Destination: " + train.Destination + "\n" +
                               "Train departs at: " + train.DepartureTime + "
" + "Date of departure: " + train.DepartureDate + "\n" +
                               "Type of seat: " + train.SeatType + "           " +
"Train Stopping at: " + train.StoppingAt + "\n" +
                               "Total Cost: " + booking.TicketCost *
booking.NumOfTickets +"           " + "Number of Tikcets booked: "+
booking.NumOfTickets + "\n" +
                               "Meal Included: " + booking.mealIncluded + "           "
+ "Seat Reserved: " + booking.SeatReservation;


            //save the booking once it is created
            //Data.Save(_dataManager);
        }



    }
}
```

Booking.cs

```
using BookMyTrain;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    //AUTHOR: Romeo Mcdonald
    //CLASS DISCRIPTION: Class conatains the values of booking
    internal class Booking
    {

        //Declaring the strings and ints for booking class



        private string trainid;
        private string departure;
        private string destination;
        public string departureTime;
        public string departureDate;
        public string stoppingAt;
        public string seatType;
        private string clientid;
        private int numoftikets;
        private int ticketcost;
        public string mealIncluded;
        private string seatReservation;
        private int seatTotal;


        //public getters and setters


        public string TrainId
        {
            get { return trainid; }
            set { trainid = value; }
        }

        public string Departure
        {
            get { return departure; }
            set { departure = value; }
        }
        public string Destination
        {
            get { return destination; }
            set { destination = value; }
        }
        public string DepartureTime
        {
            get { return departureTime; }
            set { departureTime = value; }
        }
        public string DepartureDate
        {
            get { return departureDate; }
            set { departureDate = value; }
        }
        public string ClientID
        {
            get { return clientid; }
            set { clientid = value; }
        }
        public int NumOfTickets
        {
            get { return numoftikets; }
            set { numoftikets = value; }

        }
        public int TicketCost
        {
            get { return ticketcost; }
            set { ticketcost = value; }
        }
        public string StoppingAt
        {
            get { return stoppingAt; }
```

```
            set { stoppingAt = value; }
        }
        public string SeatType
        {
            get { return seatType; }
            set { seatType = value; }
        }
        public string MealIncluded
        {
            get { return mealIncluded; }
            set { mealIncluded = value; }
        }
        public string SeatReservation
        {
            get { return seatReservation; }
            set { seatReservation = value; }
        }
        public int SeatTotal
        {
            get { return seatTotal; }
            set { seatTotal = value; }
        }


        //calculator for ticket cost
        public int TicketCalculator(Train train, Booking booking)
        {
            int price = 0;
            if ((train.Departure == "Edinburgh Waverley" || train.Departure == "London 
King's Cross") &&
                (train.Destination == "Edinburgh Waverley" || train.Destination == 
"London King's Cross"))
            {
                price += 50;
            }
            else
            {
                price += 25;
            }
            if (booking.seatType == "FirstClassSeat")
            {
                price += 10;
            }
            if (booking.seatReservation == "Yes")
            {
                price += 5;
            }
            if (booking.NumOfTickets != 1)
            {
                price += price * booking.NumOfTickets;
            }
            return price;
        }




    }
}
```

Client.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Metadata.Ecma335;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Navigation;

namespace BookMyTrain
{
    //AUTHOR: Romeo Mcdonald
    //CLASS DISCRIPTION: Class conatains the values of client


    internal class Client
    {

        //Declaring the strings and ints for client class

        private string clientID;

        //public getters and setters

        public string ClientId
        {
            get { return clientID; }
            set { clientID = value; }
        }

    }
}
```

Ticket.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{

    //AUTHOR: Romeo Mcdonald
    //CLASS DISCRIPTION: Class conatains the values of Ticket

    internal class Ticket
    {

        //Declaring the strings and ints for ticket class


        private string trainid;
        private string departure;
        private string destination;
        public string departureTime;
        public string departureDate;
        public string stoppingAt;
        public string seatType;
        private string clientid;
        private int numoftikets;
        private int ticketcost;



        //public getters and setters

        public string TrainId
        {
            get { return trainid; }
            set { trainid = value; }
        }

        public string Departure
        {
            get { return departure; }
            set { departure = value; }
        }
        public string Destination
        {
            get { return destination; }
            set { destination = value; }
        }
        public string DepartureTime
        {
            get { return departureTime; }
            set { departureTime = value; }
        }
        public string DepartureDate
        {
            get { return departureDate; }
            set { departureDate = value; }
        }
        public string ClientID
        {
            get { return clientid; }
            set { clientid = value; }
        }
        public int NumOfTickets
        {
            get { return numoftikets; }
            set { numoftikets = value; }

        }
        public int TicketCost
        {
            get { return ticketcost; }
            set { ticketcost = value; }
        }
        public string StoppingAt
        {
            get { return stoppingAt; }
            set { stoppingAt = value; }
        }
        public string SeatType
        {
            get { return seatType; }
            set { seatType = value; }
        }


    }
}
```

Train.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Business
{
    internal class Train
    {
        //AUTHOR: Romeo Mcdonald
        //CLASS DISCRIPTION: Class conatains the values of booking


        //Declaring the strings and ints for train class

        private string trainId;
        private string departure;
        private string destination;
        private string stoppingAt;
        private string seatType;
        private string departureTime;
        private string departureDate;


        //public getters and setters
        public Train()
        {
        }
        public string DepartureTime
        {
            get { return departureTime; }
            set { departureTime = value; }
        }
        public string DepartureDate
        {
            get { return departureDate; }
            set { departureDate = value; }
        }


        public string StoppingAt
        {
            get { return stoppingAt; }
            set { stoppingAt = value; }
        }

        public string TrainId
        {
            get { return trainId; }
            set { trainId = value; }
        }
        public string SeatType
        {
            get { return seatType; }
            set { seatType = value; }
        }

        public string Departure
        {
            get { return departure; }
            set { departure = value; }
        }
        public string Destination
        {
            get { return destination; }
            set { destination = value; }
        }

    }
}
```

Data.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
using static System.Formats.Asn1.AsnWriter;

namespace Business
{
    //AUTHOR: Romeo Mcdonald
    //CLASS DISCRIPTION: Class conatains the values of booking

    internal class Data
    {
        //saves in the bin folder becasue we didnt give it a path
        private static string _filename = "Trains.xml";

        //lisT
        public List<Train> TrainsData
        {
            get;
            private set;
        }
        public List<Train> Trains
        {
            get;
            private set;
        }
       public List<Booking> Bookings
        {
            get;
            private set;
        }
        public List<Ticket> Tickets
        {
            get;
            private set;
        }

        public Data()
           : this(new List<Train>())


        {

        }

        //update methods for train, ticket and booking database
        public Data(List<Train> trains)
        {
            Trains = trains;
            UpdateData();
        }
        public void Add(Train train)
        {

            Trains.Add(train);


        }
        public Data(List<Booking> bookings)
        {
            Bookings = bookings;
            UpdateData();
        }
        public void Add(Booking booking)
        {

            Bookings.Add(booking);


        }
        public Data(List<Ticket> tickets)
        {
            Tickets = tickets;
            UpdateData();
        }
        public void Add(Ticket tickets)
        {

            Tickets.Add(tickets);


        }

        //load data stored
        public static Data Load()
        {
            //if the file doesnt exist- create a new "data manager"
            if (!File.Exists(_filename))
                return new Data();

            //otherwise load the trains
            using (var reader = new StreamReader(new FileStream(_filename,
FileMode.Open)))
            {
                var serilizer = new XmlSerializer(typeof(List<Train>));
                var trains = (List<Train>)serilizer.Deserialize(reader);
                return new Data(trains);
            }

        }


        //update data stored
        public void UpdateData()
        {
            TrainsData = Trains.Take(4).ToList();//takes the first 4 trains
        }


        //serializer to save data
        public static void Save(Data dataManager)
        {
            //orders the files if it already exists
            using (var writer = new StreamWriter(new FileStream(_filename,
FileMode.Create)))
            {
                var serilizer = new XmlSerializer(typeof(List<Train>));
                serilizer.Serialize(writer, dataManager.Trains);
            }
        }
    }
}
```

UnitTest1.cs

```
namespace TestProject1
{
    //AUTHOR: Romeo Mcdonald
    //CLASS DISCRIPTION: Class conatains the values of booking
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1()
        {

           //arrange
            Train train = new Train();
            train.trainId = "train1";
            train.departure = "Edinburgh Waverley";
            train.destination="London Kings Cross";
            train.stoppingAt="NewCastle, York";
            train.seatType="Standdard";
            train.departureTime="9.00";
            train.departureDate="1/11/22";

            Booking booking = new Booking();
            booking.trainId = "train1";
            booking.clientName = "Mark Smith";
            booking.departure = "Edinburgh Waverley";
            booking.destination = "London Kings Cross";
            booking.stoppingAt = "NewCastle, York";
            booking.seatType = "Standdard";
            booking.departureTime = "9.00";
            booking.departureDate = "1/11/22";
            booking.numberOfTickets = 1;
            booking.mealIncluded = "Yes";
            booking.seatReserved = "Yes";

            string expected = "train1"+"Mark Smith"+
                               "Edinburgh Waverley " + "london Kings Scorss"+
                               "9.00 " + "1/11/22" +
                               "Standard" + "NewCastle, York" +
                               "50: " + "5: "   +
                               "Yes" + "Yes";
            //Act
            string test = train.ToString();

            //Assert
            Assert.AreEqual(expected, test);

        }
        public void TestMethod2()
        {
            //arrange
            Train train = new Train();
            train.trainId = "train2";
            train.departure = "London Kings Cross";
            train.destination = "Edinburgh Waverley";
            train.stoppingAt = "Not Stopping";
            train.seatType = "FirstClass";
            train.departureTime = "10.00";
            train.departureDate = "1/11/22";

            Booking booking = new Booking();
            booking.trainId = "train2";
            booking.clientName = "Mark Jhonson";
            booking.departure = "London Kings Cross";
            booking.destination = "Edinburgh Waverley";
            booking.stoppingAt = "Not Stopping";
            booking.seatType = "FirstClass";
            booking.departureTime = "10.00";
            booking.departureDate = "1/11/22";
            booking.numberOfTickets = 5;
            booking.mealIncluded = "No";
            booking.seatReserved = "No";


            string expected = "train2" + "Mark Jhonson" +
                              "london Kings Scorss" + "Edinburgh Waverley " +
                              "1.00 " + "1/11/22" +
                              "FirstClass" + "Not Stopping" +
                              "250: " + "5: " +
                              "No" + "No";
            //Act
            string test = train.ToString();

            //Assert
            Assert.AreEqual(expected, test);

        }
    }
}
```