DEVELOPMENT OF SOFTWARE APPLICATIONS- FALL 2020- HOMEWORK SUBMISSION REPORT

Author(s)

Haddou Sana, CP5TZK
Feras Harah, EXDFL8

TASK

FLIGHT BOOKING MANAGER

We used 4 scripts:

The code of Booking script:

This is the main part where we call the functions in the other scripts.

```
using System;

using System.Collections.Generic;



public class Booking

{


        public static void Main(string[] args)

        {



                IList<Flights> flights = new List<Flights>();



                do

                {
```

The menu that will be shown to the operator in order to choose from it the action that he will do to serve the client:

```
                Console.WriteLine("*********************");

                Console.WriteLine("Flight Booking Manager");

                Console.WriteLine("*********************");

                Console.WriteLine("Please type '1' to add a flight");

                Console.WriteLine("Please type '2' to check the available flights");

                Console.WriteLine("Please type '3' to check the available free places in a specific flight");

                Console.WriteLine("Please type '4' to book flight for a passenger");

                Console.WriteLine("Please type '5' to calculate the price of the booking flight for a passenger");
```

```
Console.WriteLine("Please type '6' to check the services available for the VIP passengers");

int choix = Convert.ToInt32(Console.ReadLine());
```

<mark>when choosing the number the switch will allow us in the program to execute the instruction wanted.</mark>

```
switch (choix)

    {

    case 1:

        addFlight((List<Flights>)flights);

        break;
```

<mark>Case one adds flights to the system.</mark>

```
    case 2:

        Console.WriteLine("To print all the available flights type: 1");

        Console.WriteLine("To print all the available flights for a specific destination type: 2");

        Console.WriteLine("To print all the available flights for a specific destination from a specific point of depart type: 3");

        Console.WriteLine("To print all the available flights for a specific date: 4");

        Console.WriteLine("To print all the available flights for a specific destination from a specific point of depart type for a specific date: 5");

        Console.WriteLine("To print all the available flights for a specific destination from a specific point of depart type for a specific date: 6");

        int c = Convert.ToInt32(Console.ReadLine());

switch (c)

        {

            case 1:

            availableFlights((List<Flights>)flights);

            break;


            case 2:

                Console.WriteLine("enter the destination: ");

                string des = Convert.ToString(Console.ReadLine());
```

```
availableFlights((List<Flights>)flights, des);

        break;

                case 3:

                        Console.WriteLine("enter the destination: ");

                        string des1 = Convert.ToString(Console.ReadLine());

Console.WriteLine("enter the point of the depart: ");

                        string dep = Convert.ToString(Console.ReadLine());

availableFlights((List<Flights>)flights, dep, des1);

        break;


                case 4:

                        Console.WriteLine("enter the year of the flight: ");

                        int year = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the month of the flight: ");

                        int month = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the day of the flight: ");

                        int day = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the hour of the flight: ");

                        int hour = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the minutes of the flight: ");

                        int minutes = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the senconds of the flight: ");

int seconds = Convert.ToInt32(Console.ReadLine());

DateTime calendar = new DateTime(year, month, day, hour, minutes, seconds);

                        availableFlights((List<Flights>)flights, calendar);

                        break;


                case 5:
```

```csharp
Console.WriteLine("enter the year of the flight: ");

int yeari = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the month of the flight: ");

int monthi = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the day of the flight: ");

int dayi = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the hour of the flight: ");

int houri = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the minutes of the flight: ");

int minutesi = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the senconds of the flight: ");

int secondsi = Convert.ToInt32(Console.ReadLine());

DateTime calendarr = new DateTime(yeari, monthi, dayi, houri, minutesi, secondsi);

Console.WriteLine("enter the destination: ");

                        string dest = Convert.ToString(Console.ReadLine());

availableFlights((List<Flights>)flights, dest, calendarr);

        break;


            case 6:

Console.WriteLine("enter the year of the flight: ");

int yearia = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the month of the flight: ");

int monthia = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the day of the flight: ");

int dayia = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the hour of the flight: ");

int houria = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the minutes of the flight: ");
```

```
                    int minutesia = Convert.ToInt32(Console.ReadLine());

                    Console.WriteLine("enter the senconds of the flight: ");

                    int secondsia = Convert.ToInt32(Console.ReadLine());

                    DateTime calendarra = new DateTime(yearia, monthia, dayia, houria, minutesia, secondsia);

                    Console.WriteLine("enter the destination: ");

                    String desti = Convert.ToString(Console.ReadLine());

                    Console.WriteLine("enter the depart: ");

                    String depart = Convert.ToString(Console.ReadLine());

                    Booking.availableFlights((List<Flights>)flights, desti, depart, calendarra);

                    break;

                }

            break;
```

Case two prints the available flights according to the information wanted by the client.

```
            case 3:

                Booking.checkForFreePlaces((List<Flights>)flights);

                break;
```

Case three for checking the available places in a specific available flight in the program.

```
            case 4:

                Booking.bookFlight((List<Flights>)flights);

                break;
```

Case four to book flight.

```
            case 5:

                Booking.calculatePrice((List<Flights>)flights);

                break;
```

Case five to calculate the price of a flight.

```
            case 6:

                Console.WriteLine("enter the flight number: ");
```

```csharp
int fn = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("enter the passenger id: ");

int id = Convert.ToInt32(Console.ReadLine());

int ff = 0;

int pp = 0;

foreach (Flights f in flights) {


    if ((f.FlightNumber == fn)) {

        foreach (Passengers p in f.Passengers) {

            if ((p.PassengerId == id)) {

                if ((p.VIP == true)) {

                    Console.WriteLine("Premium Services");

                    Console.WriteLine("Food");

                    Console.WriteLine("….");

                }

                else {

                    Console.WriteLine("Sorry, these services are available only for VIP passengers");

                }

                pp = 1;

                break;

            }

        }

        ff = 1;

        break;

    }

}

if (ff == 0)

{
```

```
                Console.WriteLine("wrong flight number");

            }

            if (pp == 0)

            {

                Console.WriteLine("wrong passengerId");

            }
```

This is used for managing VIP passengers.

```
        }

    } while (true);



}
```

The following function prints all the available flights in the system.

```
public static void availableFlights(List<Flights> flights) {

    if (string.ReferenceEquals(flights, null))

    {

        Console.WriteLine("no flight is available");

    }



    foreach (Flights flight in flights) {

        Console.WriteLine(flight.ToString());

    }



}
```

The following function shows all the available flights in the date : (DateTime c) entered .

```
public static void availableFlights(List<Flights> flights, DateTime c) {

    if (string.ReferenceEquals(flights, null)) {

    Console.WriteLine("no flight is available");

    return;
```

```
        }


        foreach (Flights flight in flights) {

            if ((flight.DepartDate == c)) {

                Console.WriteLine(flight.ToString());

            }



        }


    }
```

The following function shows all the available flights from depart place to destination place entered.

```
public static void availableFlights(List<Flights> flights, String depart, String destination) {

    if (!string.ReferenceEquals(flights, null)) {

        foreach (Flights flight in flights)

        {

            if ((flight.Depart.Equals(depart)) && flight.Destionation.Equals(destination))

            {

                Console.WriteLine(flight.ToString());

            }


        }


    }


}
```

The following function shows all the available flights from depart place to destination place entered in specific date .

```
public static void availableFlights(List<Flights> flights, String depart, String destination, DateTime c) {

    if (!string.ReferenceEquals(flights, null)) {
```

```csharp
        foreach (Flights flight in flights)

    {

       if ((flight.Depart.Equals(depart)) && flight.Destionation.Equals(destination) && (flight.DepartDate == (c)))

       {

          Console.WriteLine(flight.ToString());

       }



    }



}
```

<mark>The following function shows all the available flights to destination place entered.</mark>

```csharp
public static void availableFlights(List<Flights> flights, String Destination) {

   if (!string.ReferenceEquals(flights, null)) {


      foreach (Flights flight in flights)

    {

       if (flight.Destionation.Equals(Destination))

       {

          Console.WriteLine(flight.ToString());

       }


    }


   }


}
```

The following function shows all the available flights to destination place entered in specific date .

```
public static void availableFlights(List<Flights> flights, String Destination, DateTime cal) {

    if (!string.ReferenceEquals(flights, null)) {

        foreach (Flights flight in flights)

        {

            if ((flight.DepartDate== cal) && flight.Destionation.Equals(Destination))

            {

                Console.WriteLine(flight.ToString());

            }


        }


    }


}
```

This function checks the free places in specific flight and returns Boolean value.

```
public static bool freePlace(Flights flight) {

    if ((flight.TotalNumberOfPassengers == flight.OccupiedPlaces)) {

        return false;

    }

    else {

        return true;

    }


}
```

This function books flight for passenger in a specific flight if there is still available places in this flight.

```
public static void bookFlight(List<Flights> flights) {

    Booking.availableFlights(flights);
```

```
    int flightNumber = Convert.ToInt32(Console.ReadLine());

    Passengers passenger = Booking.addPassenger();

    Console.WriteLine(("Passenger id = "

            + (passenger.PassengerId + " added successfully")));

    if (!string.ReferenceEquals(flights, null)) {

        foreach (Flights flight in flights) {

            if ((flight.FlightNumber == flightNumber)) {

                flightNumber = flight.FlightNumber;

                if ((Booking.freePlace(flight) == true)) {

                    flight.addPassengers(flight.Passengers, passenger);

                    Console.WriteLine("flight booked successfully :) ");

                    int places = flight.Passengers.Count;

                    flight.OccupiedPlaces =places;

                    passenger.FlightNum = flight.FlightNumber;

                }

                else {

                    Console.WriteLine("flight book failed, there is no available place :( ");

                    passenger = null;

                }

                break;

            }


        }


    }


}

static int i = 0;
```

This function allows us to add new available flight to the system.

```
public static void addFlight(List<Flights> flights) {

    Console.WriteLine("enter the destination of the flight: ");

    String destination = Convert.ToString(Console.ReadLine());

    Console.WriteLine("enter the point depart of the flight: ");

    String depart = Convert.ToString(Console.ReadLine());

    Console.WriteLine("enter the year of the flight: ");

    int year = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter the month of the flight: ");

    int month = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter the day of the flight: ");

    int day = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter the hour of the flight: ");

    int hour = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter the minutes of the flight: ");

    int minutes = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter the seconds of the flight: ");

    int seconds = Convert.ToInt32(Console.ReadLine());

    DateTime calendar = new DateTime(year, month, day, hour, minutes, seconds);

    Console.WriteLine("enter the price of the flight: ");

    int prix = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter the total number of the passengers: ");

    int totalNumberOfPassengers = Convert.ToInt32(Console.ReadLine());

    Flights flight = new Flights(destination, depart, calendar, prix, totalNumberOfPassengers);

    flights.Add(flight);

    Console.WriteLine("Flight added successfully");

}
```

This function allow us to register new passenger and it will be called by the function booking.

```
public static Passengers addPassenger() {


    Console.WriteLine("enter the lastName of the passenger: ");

    String lastName = Convert.ToString(Console.ReadLine());

    Console.WriteLine("enter the firstName of the passenger: ");

    String fristName = Convert.ToString(Console.ReadLine());

    Console.WriteLine("enter the age of the passenger: ");

    int age = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter the weight of passenger\'s handbag in KG");

    double kiloOfHandBag = Convert.ToDouble(Console.ReadLine());

    Console.WriteLine("enter the weight of passenger\'s checked baggages");

    double kiloOfcheckedBaggage = Convert.ToDouble(Console.ReadLine());

    Packages lug = new Packages(kiloOfHandBag, kiloOfcheckedBaggage);

    Console.WriteLine("Is the passenger a VIP?\n 1: yes\n 2: no");

    int vip = Convert.ToInt32(Console.ReadLine());

    Boolean VIP = false;

    if (((vip != 1)  && (vip != 2))) {

        Console.WriteLine("Is the passenger a VIP?\n 1: yes\n 2: no");

        vip = Convert.ToInt32(Console.ReadLine());

    }

    else if ((vip == 1)) {

        VIP = true;

    }


    Passengers passenger = new Passengers(lastName, fristName, age, lug, VIP);

    return passenger;

}
```

This function to check free places in specific flight entered by the operator.

```
public static void checkForFreePlaces(List<Flights> flights) {


    Console.WriteLine("enter the flight number:");

    int flightNumber = Convert.ToInt32(Console.ReadLine());

    foreach (Flights flight in flights) {

        if ((flight.FlightNumber == flightNumber)) {

            int places = (flight.TotalNumberOfPassengers - flight.OccupiedPlaces);

            if ((places <= 0)) {

                Console.WriteLine("Unfortunatly, all the places are occupied :( ");

            }

            else {

                Console.WriteLine(("Fortunately "

                        + (places + " are still availables :D ")));

            }


            break;

        }


    }


}
```

This function calculates the price for a specific passenger in a specific flight.

```csharp
public static void calculatePrice(List<Flights> flights) {

    //Passengers pas = null;

    Console.WriteLine("enter your flightNumber: ");

    int flightNumber = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("enter your passengerId: ");

    int passengerId = Convert.ToInt32(Console.ReadLine());

    double price = 0; int i = 0;

    foreach (Flights f in flights) {

        if ((f.FlightNumber == flightNumber)) {

            foreach (Passengers p in f.Passengers) {

                if ((p.PassengerId == passengerId)) {

                    price = (f.Prix + p.Packages.Price);

                    if ((p.VIP == true))

                    {

                        price *= (f.PrixReductionVIP / 100);

                    }


                    Console.WriteLine(("the total price is: " + price + " Euro"));

                    break;

                }


            }

            if(price == 0)

            {

                Console.WriteLine("wrong passengerId or passenger not registred in tha flight: ");

            }

            i++;
```

```
                break;



        }

        if (i == 0)

        {

            Console.WriteLine("wrong flight number");

        }



    }


}

}
```

The code of Flight script:

```csharp
using System;
using System.Collections.Generic;

public class Flights
{

        private static int count = 0;
        private int flightNumber;

        private string Destionation_Conflict;

        private string Depart_Conflict;
        private int totalNumberOfPassengers;
        private int prix;
```

This is the default value of the reduction price for VIP passengers.

```csharp
        private double prixReductionVIP = 0.25;
        private DateTime dateDepart;
        private int occupiedPlaces = 0;
        private IList<Passengers> passengers = new List<Passengers>();



        public Flights() : base()
        {
        }
```

Constructor that allows us to create a new flight with the according informations: destination, departure place , price total number of passengers.

```csharp
        public Flights(string destionation, string depart, int prix, int totalNumberOfPassengers) :
base()
        {
                this.FlightNumber = ++count;
                Destionation_Conflict = destionation;
                Depart_Conflict = depart;
                this.prix = prix;
                this.totalNumberOfPassengers = totalNumberOfPassengers;
        }
```

Constructor that allows us to create a new flight with the according informations: destination, departure place , price total number of passengers, date.

```csharp
        public Flights(string destionation, string depart, DateTime departDate, int prix, int
totalNumberOfPassengers) : base()
        {
                this.FlightNumber = ++count;
                this.Destionation_Conflict = destionation;
                this.Depart_Conflict = depart;
                this.dateDepart = departDate;
                this.prix = prix;
                this.totalNumberOfPassengers = totalNumberOfPassengers;
        }
```

```csharp
        public Flights(string destionation, string depart, int prix, int totalNumberOfPassengers,
IList<Passengers> passengers) : base()
        {
                this.FlightNumber = ++count;
                Destionation_Conflict = destionation;
                Depart_Conflict = depart;
                this.prix = prix;
                this.totalNumberOfPassengers = totalNumberOfPassengers;
                this.passengers = passengers;
        }
```

```csharp
        public Flights(string destionation, string depart, DateTime departDate, int prix, int
totalNumberOfPassengers, IList<Passengers> passengers) : base()
        {
                this.FlightNumber = ++count;
                this.Destionation_Conflict = destionation;
                this.Depart_Conflict = depart;
                this.dateDepart = departDate;
                this.prix = prix;
                this.totalNumberOfPassengers = totalNumberOfPassengers;
                this.passengers = passengers;
        }
```

```csharp
        public virtual int FlightNumber
        {
                get
                {
                        return this.flightNumber;
                }
                set
                {
                        this.flightNumber = value;
                }
        }
```

```csharp
        public virtual string Destionation
        {
                get
                {
                        return Destionation_Conflict;
                }
                set
                {
                        Destionation_Conflict = value;
                }
        }
```

This function allow us to get and set the depart from the outside of the class.

```csharp
public virtual string Depart
{
    get
    {
        return Depart_Conflict;
    }
    set
    {
        Depart_Conflict = value;
    }
}
```

This function allow us to get and set flight time from the outside of the class.

```csharp
public virtual DateTime DepartDate
{
    get
    {
        return this.dateDepart;
    }
    set
    {
        this.dateDepart = value;
    }
}
```

This function allow us to get and set the total number of passengers from the outside of the class.

```csharp
public virtual int TotalNumberOfPassengers
{
    get
    {
        return this.totalNumberOfPassengers;
    }
    set
    {
        this.totalNumberOfPassengers = value;
    }
}
```

This function allow us to get and set the price from the outside of the class.

```csharp
public virtual int Prix
{
    get
    {
        return prix;
    }
    set
    {
        this.prix = value;
    }
}
```

This function allow us to get and set the price reduction for VIP from the outside of the class.

```csharp
public virtual double PrixReductionVIP
{
    get
    {
        return prixReductionVIP;
    }
    set
    {
        this.prixReductionVIP = value;
    }
}
```

This function allow us to get and set the occupied places from the outside of the class.

```csharp
public virtual int OccupiedPlaces
{
    get
    {
        if (this.passengers.Count == 0)
        {
            return 0;
        }
        this.occupiedPlaces = this.passengers.Count;
        return this.occupiedPlaces;
    }
    set
    {
        this.occupiedPlaces = value;
    }
}
```

This function allow us to get and set the list of passengers from the outside of the class.

```csharp
public virtual IList<Passengers> Passengers
{
    get
    {
        return passengers;
    }
    set
    {
        this.passengers = value;
    }
}
```

This function allow us to add a passenger to the list of passengers of the created flight from the outside of the class.

```csharp
public virtual void addPassengers(IList<Passengers> passengers, Passengers passenger)
{
    this.passengers.Add(passenger);
}
```

This function allow us to us print all the available informations of the flight that are not null from the outside of the class.

```csharp
        public override string ToString()
        {
                return "Flight [Flight Number =" + flightNumber + ", " +
(!string.ReferenceEquals(Destionation, null) ? "Destionation =" + Destionation + ", " : "") +
(!string.ReferenceEquals(Depart, null) ? "Depart =" + Depart + ", " : "") + (dateDepart != null ?
"DepartDate =" + dateDepart + ", " : "") + "Total Number Of Passengers = =" +
totalNumberOfPassengers + ", Prix =" + prix + " , Prix Reduction VIP = " + prixReductionVIP + ",
Occupied Places =" + occupiedPlaces +
                (!string.ReferenceEquals(passengers, null) ? "Passengers =" + Passengers.ToString() +
","   : "") + "]";
        }

}
```

The code of Passengers script:

```csharp
public class Passengers
{
        private static int id = 0;
        private static int passengerId;
        private string lastName;
        private string fristName;
        private int age;
        private Packages packages;
        private int flightNum;
        private bool VIP_Conflict;

        public Passengers() : base()
        {
        }
```

Constructor that allows us to create a new passenger with the according informations:last name ,
first name, age, its packages, and if it is VIP passenger or not.

```csharp
        public Passengers(string lastName, string fristName, int age, Packages packages, bool vIP) :
base()
        {
                this.PassengerId = ++id;
                this.lastName = lastName;
                this.fristName = fristName;
                this.age = age;
                this.packages = packages;
                VIP_Conflict = vIP;
        }
```

Constructor that allows us to create a new passenger with the according informations:last name ,
first name, age, its luggages, and if it is VIP passenger or not, flight number.

```csharp
        public Passengers(string lastName, string fristName, int age, Packages packages, int
flightNum, bool vIP) : base()
        {
                this.PassengerId = ++id;
                this.lastName = lastName;
                this.fristName = fristName;
                this.age = age;
                this.packages = packages;
                this.flightNum = flightNum;
                VIP_Conflict = vIP;
        }
```

This function allow us to get and set the passenger Id from the outside of the class.

```csharp
public virtual int PassengerId
{
    get
    {
        return passengerId;
    }
    set
    {
        Passengers.passengerId = value;
    }
}
```

This function allow us to get and set the last name of the passenger from the outside o the class.

```csharp
public virtual string LastName
{
    get
    {
        return lastName;
    }
    set
    {
        this.lastName = value;
    }
}
```

This function allow us to get and set the first name of the passenger from the outside of the class.

```csharp
public virtual string FristName
{
    get
    {
        return fristName;
    }
    set
    {
        this.fristName = value;
    }
}
```

This function allow us to get and set the age of the passenger from the outside of the class.

```csharp
public virtual int Age
{
    get
    {
        return age;
    }
    set
    {
        this.age = value;
    }
}
```

==This function allow us to get and set the packages of the passenger from the outside of the class.==

```csharp
        public virtual Packages Packages
        {
                get
                {
                        return packages;
                }
                set
                {
                        this.packages = value;
                }
        }
```

==This function allow us to get and set the flight number e of the passenger from the outside of the class.==

```csharp
        public virtual int FlightNum
        {
                get
                {
                        return flightNum;
                }
                set
                {
                        this.flightNum = value;
                }
        }
```

==This function allow us to get and set the true value of the Passenger Id from the outside of the class.==

```csharp
        public virtual bool VIP
        {
                get
                {
                        return VIP_Conflict;
                }
                set
                {
                        VIP_Conflict = value;
                }
        }
```

==This function allow us to us print all the available informations of the passenger that are not null from the outside of the class.==

```csharp
        public override string ToString()
        {
                return "Passengers [PassengerId =" + PassengerId + ", " +
(!string.ReferenceEquals(LastName, null) ? "Last Name =" + LastName + ", " : "") +
(!string.ReferenceEquals(FristName, null) ? "Frist Name =" + FristName + ", " : "") + "getAge =" +
Age + ", " + (Packages != null ? "getPackages =" + Packages + ", " : "") + "getFlightNum =" +
FlightNum + ", isVIP =" + VIP + "]";
        }

}
```

The code of Packages script:

```csharp
using System;

public class Packages
{

        private double kilo_HandBag = 10;
        private double kilo_CheckedBaggages = 30;
        private double price = 10;

        public Packages() : base()
        {
        }
```

<mark>Constructor that allows us to create a new package for the user with the according informations: weight in Kilo of the hand bag and the weight in kilo of the checked luggages.</mark>

```csharp
        public Packages(double kilo_HandBag, double kilo_CheckedBaggages) : base()
        {
                if (kilo_HandBag > 10)
                {
                        Console.WriteLine("kilo of handbag baggages more than the allowed kilos (10 kg)");

                        return;
                }
                this.kilo_HandBag = kilo_HandBag;
                this.kilo_CheckedBaggages = kilo_CheckedBaggages;
        }
```

<mark>This function allow us to get and set the weight in Kilo for the hand bag of a specific passenger from the outside of the class.</mark>

```csharp
        public virtual double Kilo_HandBag
        {
                get
                {
                        return kilo_HandBag;
                }
                set
                {
                        this.kilo_HandBag = value;
                }
        }
```

<mark>This function allow us to get and set the weight in Kilo for the checked luggages of a specific passenger from the outside of the class.</mark>

```csharp
        public virtual double Kilo_CheckedBaggages
        {
                get
                {
                        return kilo_CheckedBaggages;
                }
                set
                {
                        this.kilo_CheckedBaggages = value;
                }
        }
```

This function allow us to get and set the price of the hand bag and the checked luggages of a specific passenger from the outside of the class.

```csharp
public virtual double Price
{
    get
    {
        if (this.Kilo_CheckedBaggages > 30)
        {
            this.price = (this.Kilo_CheckedBaggages - 30) * (this.price);
        }
        return price;
    }
    set
    {
        this.price = value;
    }
}
```

This function allow us to us print all the informations of the luggages rom the outside of the class.

```csharp
public override string ToString()
{
    return "Packages [Kilo_HandBag =" + Kilo_HandBag + ", Kilo_CheckedBaggages =" +
Kilo_CheckedBaggages + ", Price =" + Price + "]";
}
```

Running the Program :

In order to run the program correctly the operator should first register a flight in the system than check, add a passenger and check prices or any service wanted by the user.

I join pictures for an example application for the program.

```
C:\Users\MULTI_TECH\Desktop\M2\Development of software app\so\BookingFlightsManagerC\projet\bin\Debug\ConsoleApplication.exe
***********************
Flight Booking Manager
***********************
Please type '1' to add a flight
Please type '2' to check the available flights
Please type '3' to check the available free places in a specific flight
Please type '4' to book flight for a passenger
Please type '5' to calculate the price of the booking flight for a passenger
Please type '6' to check the services available for the VIP passengers
1
enter the destination of the flight:
Spain
enter the point depart of the flight:
Hungary
enter the year of the flight:
2020
enter the month of the flight:
07
enter the day of the flight:
5
enter the hour of the flight:
12
enter the minutes of the flight:
00
enter the seconds of the flight:
00
enter the price of the flight:
100
enter the total number of the passengers:
15
Flight added successfully
***********************
Flight Booking Manager
***********************
Please type '1' to add a flight
Please type '2' to check the available flights
Please type '3' to check the available free places in a specific flight
Please type '4' to book flight for a passenger
Please type '5' to calculate the price of the booking flight for a passenger
Please type '6' to check the services available for the VIP passengers
2
To print all the available flights type: 1
To print all the available flights for a specific destination type: 2
To print all the available flights for a specific destination from a specific point of depart type: 3
```
Activer Windows
Accédez aux paramètres pour activer Windows.

```
C:\Users\MULTI_TECH\Desktop\M2\Development of software app\so\BookingFlightsManagerC\projet\bin\Debug\ConsoleApplication.exe
100
enter the total number of the passengers:
15
Flight added successfully
***********************
Flight Booking Manager
***********************
Please type '1' to add a flight
Please type '2' to check the available flights
Please type '3' to check the available free places in a specific flight
Please type '4' to book flight for a passenger
Please type '5' to calculate the price of the booking flight for a passenger
Please type '6' to check the services available for the VIP passengers
2
To print all the available flights type: 1
To print all the available flights for a specific destination type: 2
To print all the available flights for a specific destination from a specific point of depart type: 3
To print all the available flights for a specific date: 4
To print all the available flights for a specific destination from a specific point of depart type for a specific date: 5
To print all the available flights for a specific destination from a specific point of depart type for a specific date: 6
1
Flight [Flight Number =1, Destionation =Spain, Depart =Hungary, DepartDate =05/07/2020 12:00:00, Total Number Of Passengers = =15, Prix =100 , Prix Reduction VIP = 0,25
, Occupied Places =0Passengers =System.Collections.Generic.List`1[Passengers],]
***********************
Flight Booking Manager
***********************
Please type '1' to add a flight
Please type '2' to check the available flights
Please type '3' to check the available free places in a specific flight
Please type '4' to book flight for a passenger
Please type '5' to calculate the price of the booking flight for a passenger
Please type '6' to check the services available for the VIP passengers
3
enter the flight number:
1
Fortunately 15 are still availables :D
***********************
Flight Booking Manager
***********************
Please type '1' to add a flight
Please type '2' to check the available flights
Please type '3' to check the available free places in a specific flight
Please type '4' to book flight for a passenger
Please type '5' to calculate the price of the booking flight for a passenger
```
Activer Windows
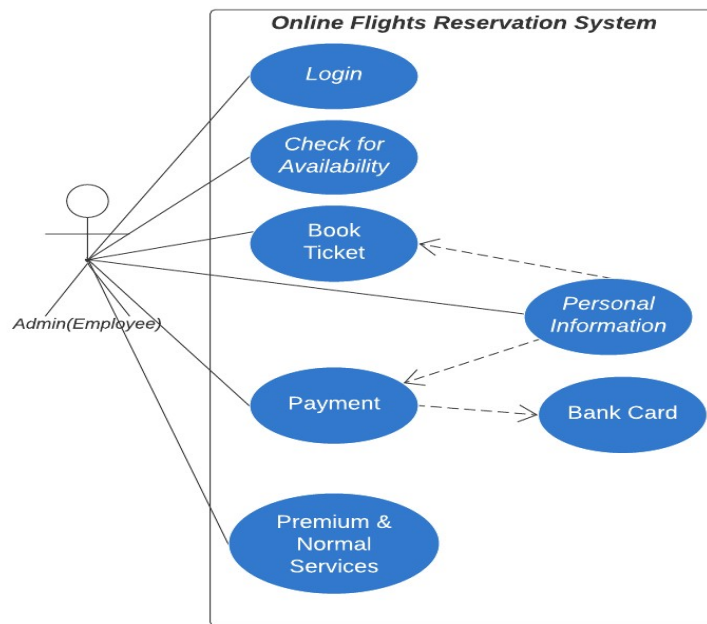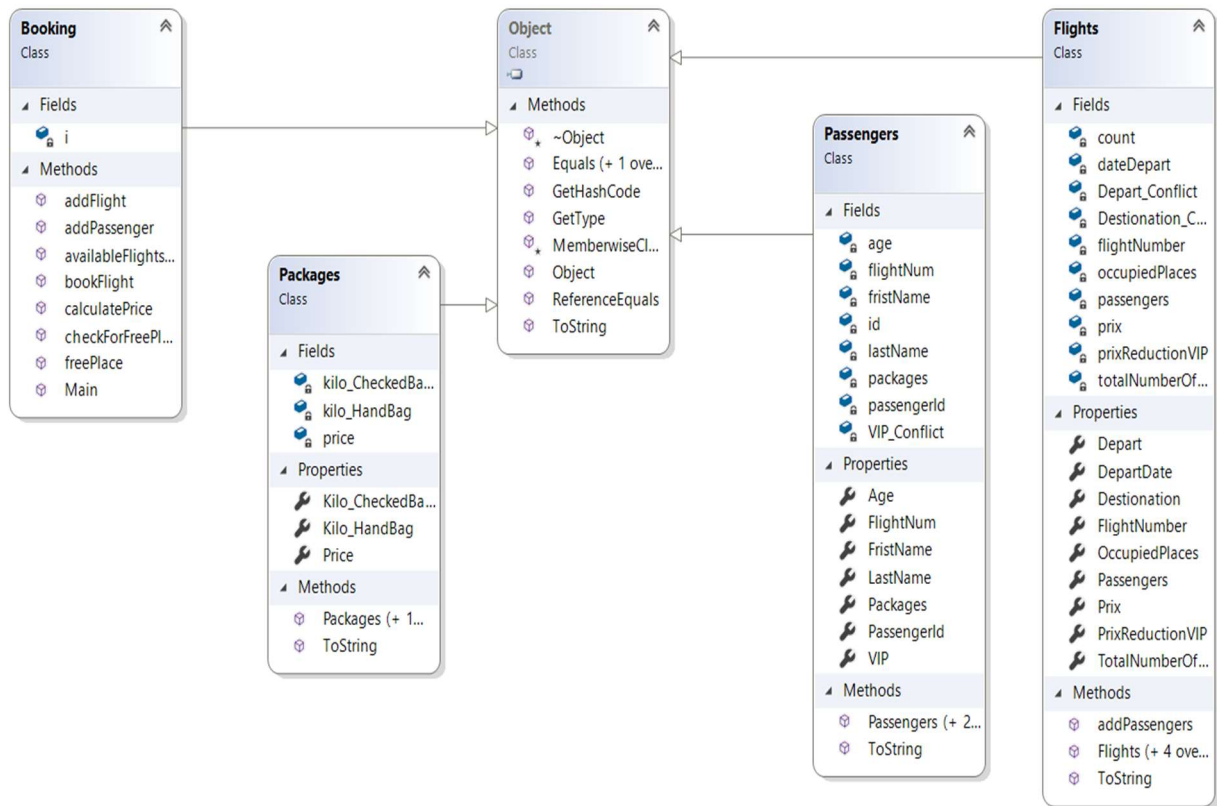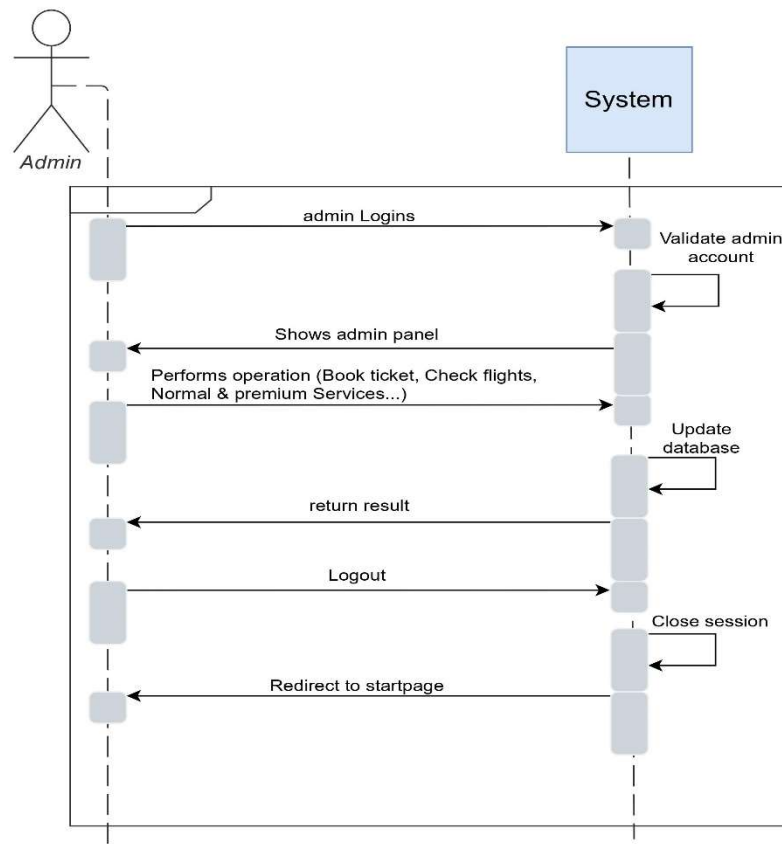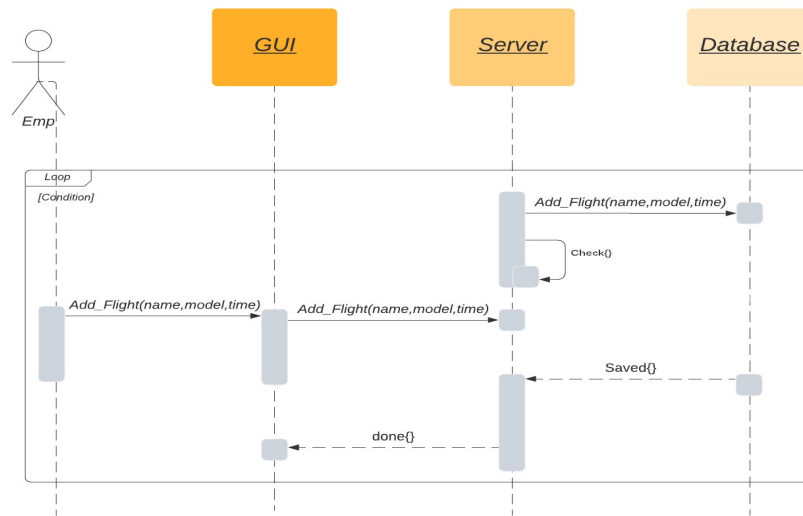Accédez aux paramètres pour activer Windows.

UML DIAGRAMS

1) UML USE CASE DIAGRAM

2) UML CLASS DIAGRAM

Booking
Class

▲ Fields
- 🔒 i

▲ Methods
- ⬡ addFlight
- ⬡ addPassenger
- ⬡ availableFlights...
- ⬡ bookFlight
- ⬡ calculatePrice
- ⬡ checkForFreePl...
- ⬡ freePlace
- ⬡ Main

Packages
Class

▲ Fields
- 🔒 kilo_CheckedBa...
- 🔒 kilo_HandBag
- 🔒 price

▲ Properties
- 🔧 Kilo_CheckedBa...
- 🔧 Kilo_HandBag
- 🔧 Price

▲ Methods
- ⬡ Packages (+ 1...
- ⬡ ToString

Object
Class

▲ Methods
- ⬡ ~Object
- ⬡ Equals (+ 1 ove...
- ⬡ GetHashCode
- ⬡ GetType
- ⬡ MemberwiseCl...
- ⬡ Object
- ⬡ ReferenceEquals
- ⬡ ToString

Passengers
Class

▲ Fields
- 🔒 age
- 🔒 flightNum
- 🔒 fristName
- 🔒 id
- 🔒 lastName
- 🔒 packages
- 🔒 passengerId
- 🔒 VIP_Conflict

▲ Properties
- 🔧 Age
- 🔧 FlightNum
- 🔧 FristName
- 🔧 LastName
- 🔧 Packages
- 🔧 PassengerId
- 🔧 VIP

▲ Methods
- ⬡ Passengers (+ 2...
- ⬡ ToString

Flights
Class

▲ Fields
- 🔒 count
- 🔒 dateDepart
- 🔒 Depart_Conflict
- 🔒 Destionation_C...
- 🔒 flightNumber
- 🔒 occupiedPlaces
- 🔒 passengers
- 🔒 prix
- 🔒 prixReductionVIP
- 🔒 totalNumberOf...

▲ Properties
- 🔧 Depart
- 🔧 DepartDate
- 🔧 Destionation
- 🔧 FlightNumber
- 🔧 OccupiedPlaces
- 🔧 Passengers
- 🔧 Prix
- 🔧 PrixReductionVIP
- 🔧 TotalNumberOf...

▲ Methods
- ⬡ addPassengers
- ⬡ Flights (+ 4 ove...
- ⬡ ToString

3) UML SEQUENCE DIAGRAMS

4) EXTRA WORK

https://github.com/sanahaddou/Homework-of-Developpement-of-Software-applications