```
using DoctorModelLibrary;
using DoctorDALLibrary;
using DoctorBLLibrary;
using System.Diagnostics;
using System.Numerics;

namespace DoctorApp
{
    internal class Program
    {
        IDoctorService doctorservice;
        public Program()
        {
            doctorservice = new DoctorService();
        }
        void displayAdminMenu()
        {
            Console.WriteLine(" 1.Add Doctor");
            Console.WriteLine(" 2.Modify Doctor Phone");
            Console.WriteLine(" 3.Modify Doctor Experience");
            Console.WriteLine(" 4.Delete Doctor");
            Console.WriteLine(" 5.Print All Doctors");
            Console.WriteLine(" 0. Exit ");

        }
        void StartAdminActivities()
        {
            int choice;
            do
            {
                displayAdminMenu();
                choice = Convert.ToInt32(Console.ReadLine());
                switch (choice)
                {
                    case 0:
                        Console.WriteLine("Bye bye");
                        break;
                    case 1:
                        AddDoctor();
                        break;
                    case 2:
                        UpdatePhone();
                        break;
                    case 3:
                        UpdateExperience();
                        break;
                    case 4:
                        DeleteDoctor();
                        break;
                    case 5:
                        PrintAllDoctors();
                        break;

                    default:
                        Console.WriteLine("Invalid choice. Try again");
                        break;
                }
            } while (choice != 0);
        }
        private void PrintAllDoctors()
        {
            Console.WriteLine("**********************************");
            var doctors = doctorservice.GetDoctors();
```

```
    foreach (var item in doctors)
    {
        Console.WriteLine(item);
        Console.WriteLine("-----------------------------");
    }
    Console.WriteLine("**********************************");
}
void AddDoctor()
{
    try
    {
        Doctor doctor = TakeDoctorDetails();
        var result = doctorservice.AddDoctor(doctor);
        if (result != null)
        {
            Console.WriteLine("Doctor added");
        }
    }
    catch (FormatException e)
    {
        Console.WriteLine(e.Message);

    }
    catch (NotAddedException e)
    {
        Console.WriteLine(e.Message);
    }

}
Doctor TakeDoctorDetails()
{
    Doctor doctor = new Doctor();
    Console.WriteLine("Enter Doctor Name");
    doctor.Name = Console.ReadLine();
    Console.WriteLine("Enter Doctor Qualification");
    doctor.Qualification = Console.ReadLine();
    Console.WriteLine("Enter Doctor Specialization");
    doctor.Specialization = Console.ReadLine();
    Console.WriteLine("Enter Doctor Experience");
    doctor.Experience = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Enter Doctor Fee");
    doctor.Fee = Convert.ToDouble(Console.ReadLine());
    Console.WriteLine("Enter Doctor Mobile Number");
    doctor.Phone = Console.ReadLine();
    return doctor;
}
int GetDoctorIdFromUser()
{
    int id;
    Console.WriteLine("Please enter the doctor id");
    id = Convert.ToInt32(Console.ReadLine());
    return id;
}
private void DeleteDoctor()
{
    try
    {
        int id = GetDoctorIdFromUser();
        if (doctorservice.Delete(id) != null)
            Console.WriteLine("Doctor deleted");
    }
    catch (NoSuchDoctorException e)
    {
```

```
                Console.WriteLine(e.Message);
            }
        }
        private void UpdatePhone()
        {
            var id = GetDoctorIdFromUser();
            Console.WriteLine("Please enter the new phone number");
            string phone = Console.ReadLine();
            Doctor doctor = new Doctor();
            doctor.Phone = phone;
            doctor.Id = id;
            try
            {
                var result = doctorservice.ModifyPhoneNumber(id, phone);
                if (result != null)
                    Console.WriteLine("Update success");
            }
            catch (NoSuchDoctorException e)
            {
                Console.WriteLine(e.Message);
            }
        }
        private void UpdateExperience()
        {
            var id = GetDoctorIdFromUser();
            Console.WriteLine("Please enter the new experience");
            int experience = Convert.ToInt32(Console.ReadLine());
            Doctor doctor = new Doctor();
            doctor.Experience = experience;
            doctor.Id = id;
            try
            {
                var result = doctorservice.ModifyExperience(id, experience);
                if (result != null)
                    Console.WriteLine("Update success");
            }
            catch (NoSuchDoctorException e)
            {
                Console.WriteLine(e.Message);
            }
        }



        static void Main(string[] args)
        {
            Program program = new Program();
            program.StartAdminActivities();
            Console.WriteLine("Hello, World!");
        }
    }
}
```

```
namespace DoctorModelLibrary
{
    public class Doctor
    {
        public int Id { get; set; }
        public string Name { get; set; }=string.Empty;
        public string Qualification { get; set; } = string.Empty;
        public string Specialization { get; set; } = string.Empty;
        public int Experience { get; set; }
        public string Phone { get; set; } = string.Empty;
```

```
        public double Fee { get; set; }

        public Doctor(int id, string name, string qualification, string
specialization, int experience, string phone, double fee)
        {
            Id = id;
            Name = name;
            Qualification = qualification;
            Specialization = specialization;
            Experience = experience;
            Phone = phone;
            Fee = fee;
        }

        public Doctor()
        {
            Experience = 0;



        }
        /// <summary>
        ///
        /// </summary>
        /// <returns></returns>

        public override string ToString()
        {
            return $"Id:{Id}\nName: {Name}\nQualification:
{Qualification}\nSpecialization: {Specialization}\n" +
                $"Experience: {Experience}\nFee: {Fee}\nPhone: {Phone}";
        }


    }
}
```

```
using DoctorModelLibrary;
namespace DoctorDALLibrary
{
    public class DoctorRepository : IRepository
    {
        Dictionary<int, Doctor> doctors = new Dictionary<int, Doctor>();
        /// <summary>
        ///
        /// </summary>
        /// <param name="doctor">Doctor object that has to be added</param>
        /// <returns></returns>
        public Doctor Add(Doctor doctor)
        {
            int id = GetTheNextId();
            try
            {
                doctor.Id = id;
                doctors.Add(doctor.Id, doctor);
                return doctor;
            }
            catch (ArgumentException e)
            {
                Console.WriteLine("The doctor Id already exists");
                Console.WriteLine(e.Message);
            }
            return null;
```

```
}
/// <summary>
///
///
/// </summary>
/// <returns></returns>
private int GetTheNextId()
{
    if (doctors.Count == 0)
        return 1;
    int id = doctors.Keys.Max();
    return ++id;
}
/// <summary>
///
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
public Doctor Delete(int id)
{
    var doctor = doctors[id];
    doctors.Remove(id);
    return doctor;
}
public List<Doctor> GetAll()
{
    var doctorList = doctors.Values.ToList();
    return doctorList;
}
/// <summary>
///
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
public Doctor GetById(int id)
{
    if (doctors.ContainsKey(id))
        return doctors[id];
    return null;

}
/// <summary>
///
/// </summary>
/// <param name="doctors"></param>
/// <returns></returns>
public Doctor ModifyPhone(Doctor doctor)
{
    doctors[doctor.Id] = doctor;
    return doctors[doctor.Id];
}
/// <summary>
///
/// </summary>
/// <param name="doctors"></param>
/// <returns></returns>
public Doctor ModifyExperience(Doctor doctor)
{
    doctors[doctor.Id] = doctor;
    return doctors[doctor.Id];
}
```

```
    }
}
```

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

namespace DoctorDALLibrary
{
    public interface IRepository
    {
        public Doctor Add(Doctor doctor);
        public Doctor ModifyPhone(Doctor doctor);
        public Doctor ModifyExperience(Doctor doctor);
        public Doctor Delete(int id);
        public Doctor GetById(int id);
        public List<Doctor> GetAll();
    }
}
```

```
using DoctorDALLibrary;
using DoctorModelLibrary;
namespace DoctorBLLibrary
{
    public class DoctorService : IDoctorService
    {
        IRepository repository;
        public  DoctorService()
        {
            repository = new DoctorRepository();
        }
        public Doctor AddDoctor(Doctor doctor)
        {
            var result = repository.Add(doctor);
            if (result != null)
                return result;
            throw new NotAddedException();
        }
        public Doctor Delete(int id)
        {
            var product = GetDoctor(id);
            if (product != null)
            {
                repository.Delete(id);
                return product;
            }
            throw new NoSuchDoctorException();
        }
        public Doctor GetDoctor(int id)
        {
            var result = repository.GetById(id);
            //if (result != null)
            //    return result;
            //throw new NoSuchProductException();

            //null collasing operator
            //return result ?? throw new NoSuchProductException();

            return result == null ? throw new NoSuchDoctorException() : result;
```

```
        }
        public List<Doctor> GetDoctors()
        {
            var doctors = repository.GetAll();
            if (doctors.Count != 0)
                return doctors;
            throw new NoDoctorsAvailableException();
        }
        public Doctor ModifyPhoneNumber(int id, string Phone)
        {
            var doctor = GetDoctor(id);
            if (doctor != null)
            {
                doctor.Phone = Phone;
                var result = repository.ModifyPhone(doctor);
                return result;
            }
            throw new NoSuchDoctorException();
        }
        public Doctor ModifyExperience(int id, int Experience)
        {
            var doctor = GetDoctor(id);
            if (doctor != null)
            {
                doctor.Experience = Experience;
                var result = repository.ModifyExperience(doctor);
                return result;
            }
            throw new NoSuchDoctorException();
        }




    }
}
```

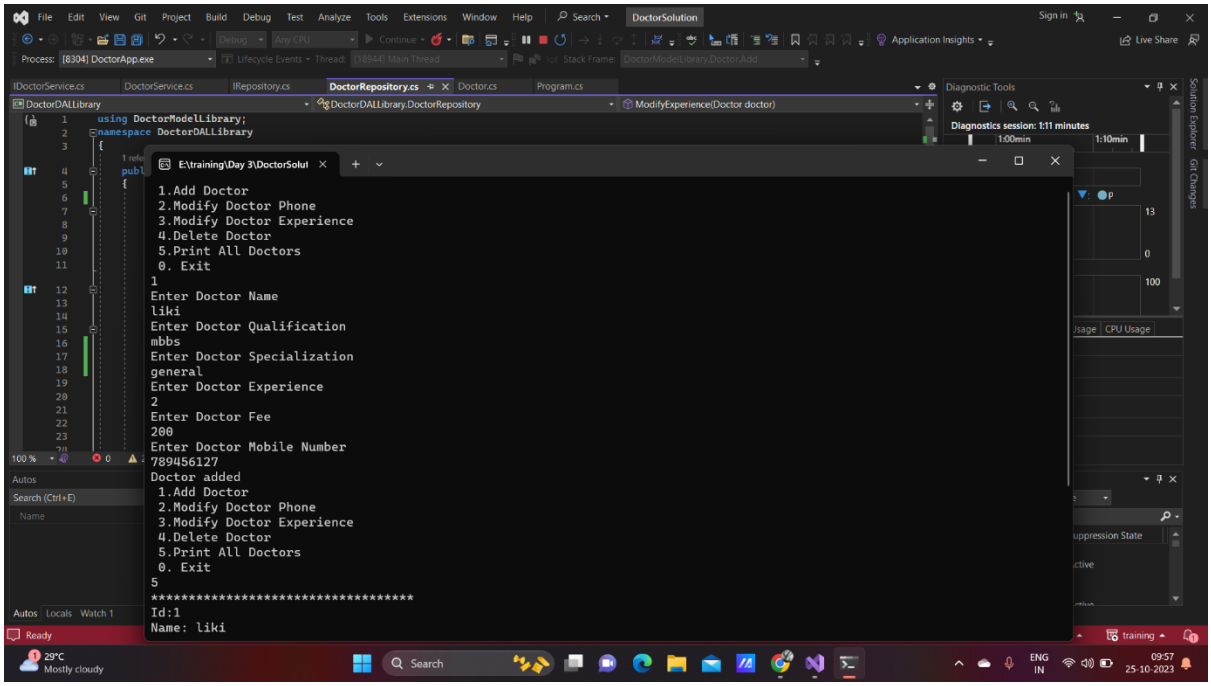
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using DoctorDALLibrary;
using DoctorModelLibrary;

namespace DoctorBLLibrary
{
    public interface IDoctorService
    {
        public Doctor AddDoctor(Doctor doctor);
        public Doctor ModifyPhoneNumber(int id, string Experience);
        public Doctor ModifyExperience(int id, int Experience);
        public Doctor GetDoctor(int id);
        public List<Doctor> GetDoctors();
        public Doctor Delete(int id);

    }
}
```

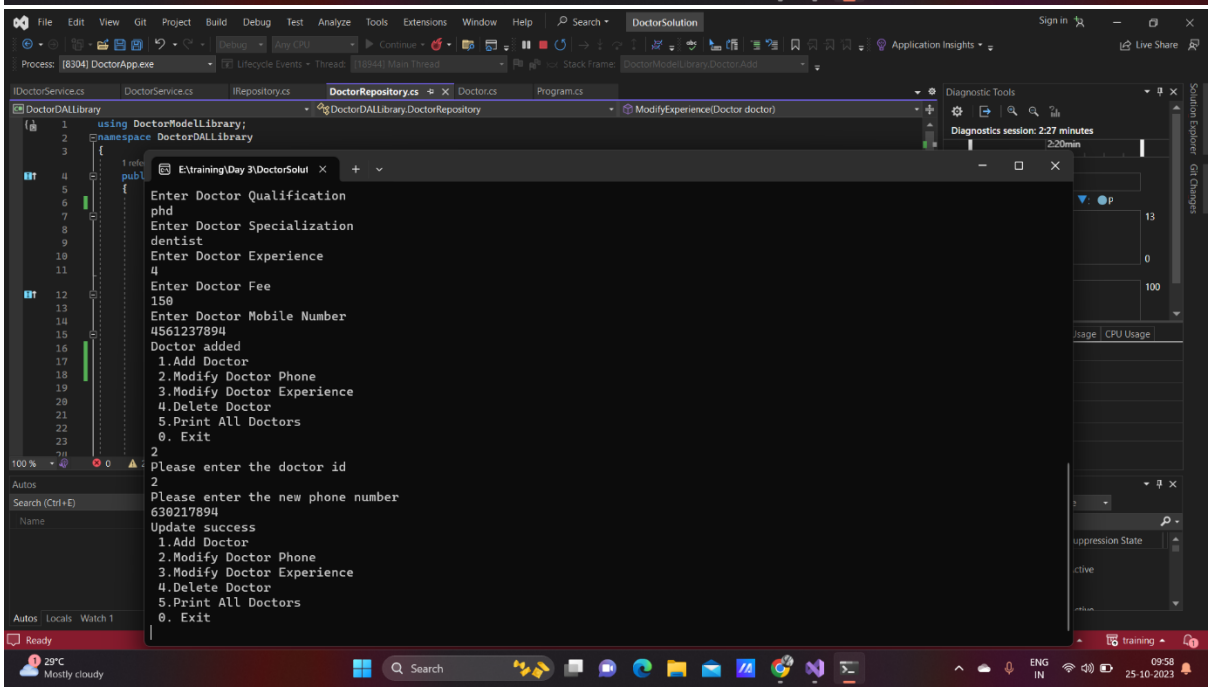
```
using System;
using System.Collections.Generic;
```

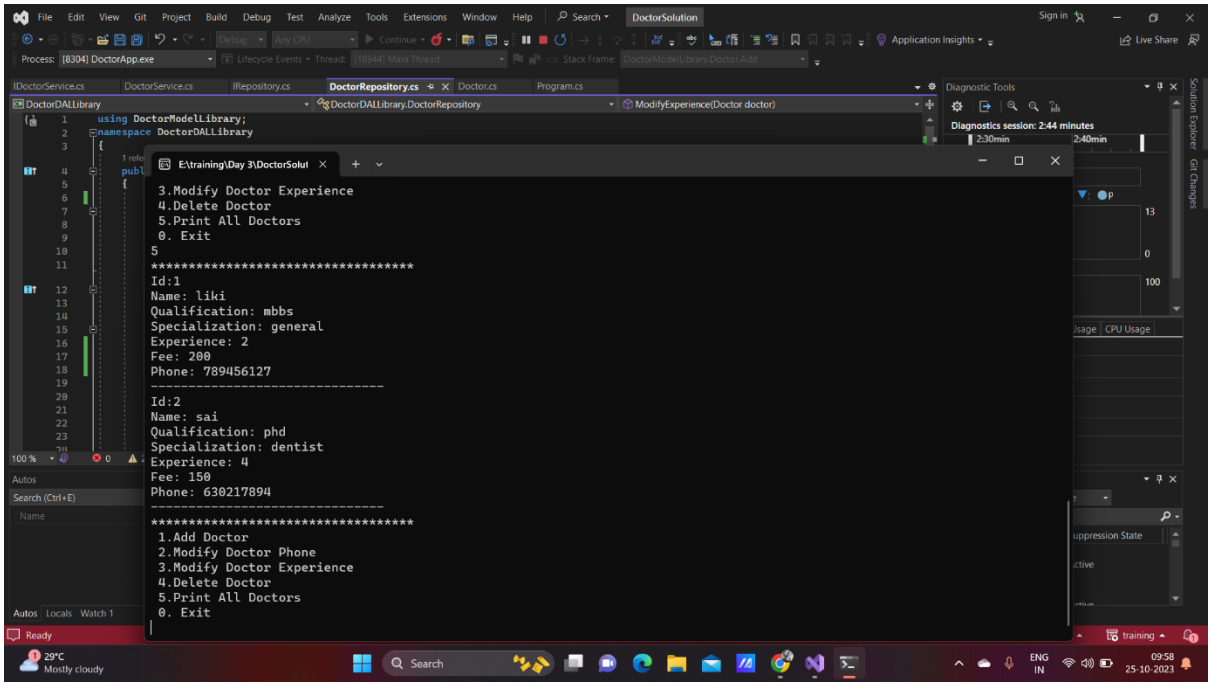
```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DoctorBLLibrary
{
    public class NoDoctorsAvailableException : Exception
    {
        string message;
        public NoDoctorsAvailableException()
        {
            message = "No Doctors are available currently";
        }

        public override string Message => message;


    }
}
```

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


namespace DoctorBLLibrary
{
    public class NoSuchDoctorException : Exception
    {
        string message;
        public NoSuchDoctorException()
        {
            message = "The doctor with teh given id is not present";
        }
        public override string Message => message;

    }
}
```

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

namespace DoctorBLLibrary
{
    public class NotAddedException : Exception
    {
        string message;
        public NotAddedException()
        {
            message = "Doctor was not addedd.";
        }
        public override string Message => message;

    }
}
```


E:\training\Day 3\DoctorSolut
1.Add Doctor
2.Modify Doctor Phone
3.Modify Doctor Experience
4.Delete Doctor
5.Print All Doctors
0. Exit
1
Enter Doctor Name
liki
Enter Doctor Qualification
mbbs
Enter Doctor Specialization
general
Enter Doctor Experience
2
Enter Doctor Fee
200
Enter Doctor Mobile Number
789456127
Doctor added
1.Add Doctor
2.Modify Doctor Phone
3.Modify Doctor Experience
4.Delete Doctor
5.Print All Doctors
0. Exit
5
****
Id:1
Name: liki


E:\training\Day 3\DoctorSolut
Enter Doctor Qualification
phd
Enter Doctor Specialization
dentist
Enter Doctor Experience
4
Enter Doctor Fee
150
Enter Doctor Mobile Number
4561237894
Doctor added
1.Add Doctor
2.Modify Doctor Phone
3.Modify Doctor Experience
4.Delete Doctor
5.Print All Doctors
0. Exit
2
Please enter the doctor id
2
Please enter the new phone number
630217894
Update success
1.Add Doctor
2.Modify Doctor Phone
3.Modify Doctor Experience
4.Delete Doctor
5.Print All Doctors
0. Exit


E:\training\Day 3\DoctorSolut
3.Modify Doctor Experience
4.Delete Doctor
5.Print All Doctors
0. Exit
5
****
Id:1
Name: liki
Qualification: mbbs
Specialization: general
Experience: 2
Fee: 200
Phone: 789456127

Id:2
Name: sai
Qualification: phd
Specialization: dentist
Experience: 4
Fee: 150
Phone: 630217894

****
1.Add Doctor
2.Modify Doctor Phone
3.Modify Doctor Experience
4.Delete Doctor
5.Print All Doctors
0. Exit