**Day 3 Assignment**

**Program.cs**

using DoctorModelLibrary;

using DoctorBLLibrary;

using DoctorDALLibrary;

using System.Numerics;

namespace ClinicApplication

{

internal class Program

{

IDoctorService doctorService;

public Program() {

doctorService =new DoctorService();

}

void DisplayAdminMenu()

{

Console.WriteLine("1. Add Doctor");

Console.WriteLine("2. Modify Phone");

Console.WriteLine("3. Modify Experience");

Console.WriteLine("4. Delete Doctor");

Console.WriteLine("5. Show 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:

ShowDoctor();

break;

default:

Console.WriteLine("Invalid choice. Try again");

break;

}

} while (choice != 0);

}

private void ShowDoctor()

{

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("Please enter the doctor name");

doctor.Name = Console.ReadLine();

Console.WriteLine("Please enter the doctor's specialisation");

doctor.Specialisation = Console.ReadLine();

Console.WriteLine("Please enter the doctor's experience");

doctor.Experience = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Please enter the doctor's phone");

doctor.Phone = Convert.ToDouble(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");

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

Doctor doctor = new Doctor();

doctor.Phone = phone;

doctor.Id = id;

try

{

var result = doctorService.UpdateDoctorPhone(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.UpdateDoctorExperience(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();

}

}

}

**DoctorService.cs:**

using DoctorDALLibrary;

using DoctorModelLibrary;

using System.Diagnostics;

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 doctor = GetDoctor(id);

if (doctor != null)

{

repository.Delete(id);

return doctor;

}

throw new NoSuchDoctorException();

}

public Doctor GetDoctor(int id)

{

var result = repository.GetById(id);

//null collasing operator

return result ?? throw new NoSuchDoctorException();

}

public List<Doctor> GetDoctors()

{

var doctors = repository.GetAll();

if (doctors.Count != 0)

return doctors;

throw new NoDoctorsAvailableException();

}

public Doctor UpdateDoctorExperience(int id, int experience)

{

var doctor = GetDoctor(id);

if (doctor != null)

{

doctor.Experience= experience;

var result = repository.Update(doctor);

return result;

}

throw new NoSuchDoctorException();

}

public Doctor UpdateDoctorPhone(int id, double phone)

{

var doctor = GetDoctor(id);

if (doctor != null)

{

doctor.Phone = phone;

var result = repository.Update(doctor);

return result;

}

throw new NoSuchDoctorException();

}

}

}

**IDoctorService.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using DoctorModelLibrary;

namespace DoctorBLLibrary

{

public interface IDoctorService

{

public Doctor AddDoctor(Doctor doctor);

public Doctor UpdateDoctorPhone(int id, double phone);

public Doctor GetDoctor(int id);

public List<Doctor> GetDoctors();

public Doctor UpdateDoctorExperience(int id, int experience);

public Doctor Delete(int id);

}

}

**DoctorRepository.cs:**

using DoctorModelLibrary;

namespace DoctorDALLibrary

{

public class DoctorRepository : IRepository

{

Dictionary<int, Doctor> doctors = new Dictionary<int, Doctor>();

/// <summary>

/// Adds doctors to dictionary

/// </summary>

/// <param name="doctor">object 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;

}

private int GetTheNextId()

{

if (doctors.Count == 0)

return 1;

int id = doctors.Keys.Max();

return ++id;

}

/// <summary>

/// Deletes the doctor

/// </summary>

/// <param name="id">Id of the doctor to delete</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;

}

public Doctor Update(Doctor doctor)

{

doctors[doctor.Id] = doctor;

return doctors[doctor.Id];

}

public Doctor GetById(int id)

{

return doctors[id];

}

}

}

**IRepository.cs:**

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 Update(Doctor doctor);

public Doctor Delete(int id);

public Doctor GetById(int id);

public List<Doctor> GetAll();

}

}

**Doctor.cs:**

namespace DoctorModelLibrary

{

public class Doctor

{

public int Id { get; set; }

public string Name { get; set; }

public int Experience { get; set; }

public string Specialisation { get; set; }

public double Phone { get; set; }

public Doctor()

{

Experience = 0;

}

/// <summary>

/// Constructs the doctor object with properties

/// </summary>

/// <param name="id">Doctor id</param>

/// <param name="name">Doctor's name</param>

/// <param name="experience">Doctor's experience</param>

/// <param name="specialisation">Doctor's specialisation</param>

/// <param name="phone">Doctor's phone</param>

public Doctor(int id, string name, int experience, string specialisation, double phone)

{

Id = id;

Name = name;

Experience = experience;

Specialisation = specialisation;

Phone = phone;

}

public override string ToString()

{

return $"Doctor ID : {Id}\nName : {Name}\nSpecialisation : {Specialisation}\nExperience : {Experience} years\nPhone : {Phone}";

}

}

}

**Output:**

