using System;

using System.Collections.Generic;

// Define CRUD Interfaces for PersonalDetails

public interface ICreatePersonalDetails

{

    void Create(PersonalDetails obj);

}

public interface IReadPersonalDetails

{

    PersonalDetails Read(int id);

}

public interface IUpdatePersonalDetails

{

    void Update(PersonalDetails obj);

}

public interface IDeletePersonalDetails

{

    void Delete(int id);

}

// Define CRUD Interfaces for Marks

public interface ICreateMarks

{

    void Create(Marks obj);

}

public interface IReadMarks

{

    Marks Read(int id);

}

public interface IUpdateMarks

{

    void Update(Marks obj);

}

public interface IDeleteMarks

{

    void Delete(int id);

}

// Define CRUD Interfaces for Attendance

public interface ICreateAttendance

{

    void Create(Attendance obj);

}

public interface IReadAttendance

{

    Attendance Read(int id);

}

public interface IUpdateAttendance

{

    void Update(Attendance obj);

}

public interface IDeleteAttendance

{

    void Delete(int id);

}

// Entities

public class PersonalDetails

{

    public int Id { get; set; }

    public string Name { get; set; }

    public string Email { get; set; }

}

public class Marks

{

    public int Id { get; set; }

    public string Subject { get; set; }

    public int Score { get; set; }

}

public class Attendance

{

    public int Id { get; set; }

    public DateTime Date { get; set; }

    public string Status { get; set; }

}

// Repositories

public class PersonalDetailsRepository : ICreatePersonalDetails, IReadPersonalDetails, IUpdatePersonalDetails, IDeletePersonalDetails

{

    private readonly Dictionary<int, PersonalDetails> \_data = new();

    public void Create(PersonalDetails obj)

    {

        \_data[obj.Id] = obj;

        Console.WriteLine("PersonalDetails created.");

    }

    public PersonalDetails Read(int id)

    {

        return \_data.ContainsKey(id) ? \_data[id] : null;

    }

    public void Update(PersonalDetails obj)

    {

        if (\_data.ContainsKey(obj.Id))

        {

            \_data[obj.Id] = obj;

            Console.WriteLine("PersonalDetails updated.");

        }

    }

    public void Delete(int id)

    {

        if (\_data.ContainsKey(id))

        {

            \_data.Remove(id);

            Console.WriteLine("PersonalDetails deleted.");

        }

    }

}

public class MarksRepository : ICreateMarks, IReadMarks, IUpdateMarks, IDeleteMarks

{

    private readonly Dictionary<int, Marks> \_data = new();

    public void Create(Marks obj)

    {

        \_data[obj.Id] = obj;

        Console.WriteLine("Marks created.");

    }

    public Marks Read(int id)

    {

        return \_data.ContainsKey(id) ? \_data[id] : null;

    }

    public void Update(Marks obj)

    {

        if (\_data.ContainsKey(obj.Id))

        {

            \_data[obj.Id] = obj;

            Console.WriteLine("Marks updated.");

        }

    }

    public void Delete(int id)

    {

        if (\_data.ContainsKey(id))

        {

            \_data.Remove(id);

            Console.WriteLine("Marks deleted.");

        }

    }

}

public class AttendanceRepository : ICreateAttendance, IReadAttendance, IUpdateAttendance, IDeleteAttendance

{

    private readonly Dictionary<int, Attendance> \_data = new();

    public void Create(Attendance obj)

    {

        \_data[obj.Id] = obj;

        Console.WriteLine("Attendance created.");

    }

    public Attendance Read(int id)

    {

        return \_data.ContainsKey(id) ? \_data[id] : null;

    }

    public void Update(Attendance obj)

    {

        if (\_data.ContainsKey(obj.Id))

        {

            \_data[obj.Id] = obj;

            Console.WriteLine("Attendance updated.");

        }

    }

    public void Delete(int id)

    {

        if (\_data.ContainsKey(id))

        {

            \_data.Remove(id);

            Console.WriteLine("Attendance deleted.");

        }

    }

}

// Menu-Driven Program

class Program

{

    static void Main(string[] args)

    {

        var personalDetailsRepo = new PersonalDetailsRepository();

        var marksRepo = new MarksRepository();

        var attendanceRepo = new AttendanceRepository();

        while (true)

        {

            Console.WriteLine("\n=== Menu ===");

            Console.WriteLine("1. PersonalDetails CRUD");

            Console.WriteLine("2. Marks CRUD");

            Console.WriteLine("3. Attendance CRUD");

            Console.WriteLine("4. Exit");

            Console.Write("Choose an option: ");

            var choice = Console.ReadLine();

            if (choice == "4") break;

            switch (choice)

            {

                case "1":

                    HandlePersonalDetails(personalDetailsRepo);

                    break;

                case "2":

                    HandleMarks(marksRepo);

                    break;

                case "3":

                    HandleAttendance(attendanceRepo);

                    break;

                default:

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

                    break;

            }

        }

    }

    static void HandlePersonalDetails(PersonalDetailsRepository repo)

    {

        Console.WriteLine("\n--- PersonalDetails CRUD ---");

        Console.WriteLine("1. Create");

        Console.WriteLine("2. Read");

        Console.WriteLine("3. Update");

        Console.WriteLine("4. Delete");

        Console.Write("Choose an option: ");

        var choice = Console.ReadLine();

        switch (choice)

        {

            case "1":

                Console.Write("Enter ID: ");

                int id = int.Parse(Console.ReadLine());

                Console.Write("Enter Name: ");

                string name = Console.ReadLine();

                Console.Write("Enter Email: ");

                string email = Console.ReadLine();

                repo.Create(new PersonalDetails { Id = id, Name = name, Email = email });

                break;

            case "2":

                Console.Write("Enter ID to Read: ");

                id = int.Parse(Console.ReadLine());

                var details = repo.Read(id);

                if (details != null)

                    Console.WriteLine($"ID: {details.Id}, Name: {details.Name}, Email: {details.Email}");

                else

                    Console.WriteLine("Record not found.");

                break;

            case "3":

                Console.Write("Enter ID: ");

                id = int.Parse(Console.ReadLine());

                Console.Write("Enter Updated Name: ");

                name = Console.ReadLine();

                Console.Write("Enter Updated Email: ");

                email = Console.ReadLine();

                repo.Update(new PersonalDetails { Id = id, Name = name, Email = email });

                break;

            case "4":

                Console.Write("Enter ID to Delete: ");

                id = int.Parse(Console.ReadLine());

                repo.Delete(id);

                break;

            default:

                Console.WriteLine("Invalid option.");

                break;

        }

    }

    static void HandleMarks(MarksRepository repo)

    {

        Console.WriteLine("\n--- Marks CRUD ---");

        // Similar to HandlePersonalDetails

    }

    static void HandleAttendance(AttendanceRepository repo)

    {

        Console.WriteLine("\n--- Attendance CRUD ---");

        // Similar to HandlePersonalDetails

    }

}