// "aplikace" která bude pracovat s uzivateli pomocí interfacu

// implementujeme interface ruznymi zpusoby – viz 1) InMemoryUserReporitory   
a 2) XmlUserRepository

class Program

{

    static void Main(string[] args)

    {

        var userRepository = new InMemoryUserReporitory(); //s vyžitím dat v paměti

        var userRepository2 = new XmlUserRepository(); //s využitím dat uložených v souboru

        App myApp = new App(userRepository);

        App myApp2 = new App(userRepository2);

        myApp.Run();

        myApp2.Run();

    }

}

public class App

{

    private IUserRepository userRepository;

    public App(IUserRepository userRepository) //constructor

    {

        this.userRepository = userRepository;

    }

    public void Run()

    {

        var users = this.userRepository.GetAllUsers();

        foreach (var user in users)

        {

            this.OutputUser(user);

        }

        this.userRepository.AddUser(new User { Name = "Martin", PhoneNumber = "+420111222333" });

        var usersAfter = this.userRepository.GetAllUsers();

        System.Console.WriteLine("Users after adding:");

        foreach (var user in usersAfter)

        {

            this.OutputUser(user);

        }

    }

    private void OutputUser(User user)

    {

        System.Console.WriteLine($"User {user.Name} has phone number {user.PhoneNumber}");

    }

}

public interface IUserRepository

{

    void AddUser(User user);

    User GetUser(string name);

    IEnumerable<User> GetAllUsers();

}

//1. způsob implementace interface

public class InMemoryUserReporitory : IUserRepository

{

    private List<User> users; //nebo prázdný seznam (pak bych nepotřebovala bezparametrický konstruktor): private List<User> users = new List<User>();

    public InMemoryUserReporitory() //bezparametrický konstruktor

    {

        this.users = new List<User>();

    }

    public void AddUser(User user)

    {

        users.Add(user);

    }

    public IEnumerable<User> GetAllUsers()

    {

        return users;

    }

    public User GetUser(string name)

    {

        return users.Where(user => user.Name == name).First();

    }

}

//2. způsob implementace interface

public class XmlUserRepository : IUserRepository

{

    public static string fileName = "userCollection.xml";

    private XmlSerializer serializer; //ne static

    public XmlUserRepository() //konstruktor

    {

        this.serializer = new XmlSerializer(typeof(List<User>));

    }

    //read the file with users

    //add the new user to the collection

    //write back

    public void AddUser(User user)

    {

        var users = this.GetAllUsers() as List<User>; //abych mohla využít metodu Add musím IEnumerable překonvertovat na List

        users.Add(user);

        using (var writer = new StreamWriter(fileName))

        {

            serializer.Serialize(writer, users);

        }

    }

    public IEnumerable<User> GetAllUsers()

    {

        //var serializer = new XmlSerializer(typeof(List<User>)); //jen pokud bych neměla private XmlSerializer ani konstruktor

        //if bych měla csv nebo txt - pomocí ReadAllLines

        if (!File.Exists(fileName))

        {

            return new List<User>();

        }

        else

        {

            using (var reader = new StreamReader(fileName))

            {

                var users = this.serializer.Deserialize(reader) as List<User>;

                return users;

            }

        }

    }

public User GetUser(string name)

    {

        //return users.Where(user => user.Name == name).First();

        var users = this.GetAllUsers();

        return users.Where(user => user.Name == name).First();

    }

}