public abstract class GeneralUser

{

    public string Name { get; set; }

    public string Password { get; private set; }

    //public GeneralUser(string? name = null, string? password = null)

    public GeneralUser() { }

    public GeneralUser(string name, string password)

    {

        string encodedPassword = Utils.EncodePassword(password);

        Name = name;

        Password = encodedPassword;

    }

    public void LogOut()

    {

        System.Console.WriteLine("You have been logged out.");

        Utils.WaitForEnter();

    }

public class Manager : GeneralUser

{

    //public Manager(string? name = null, string? password = null) : base(name, password)

    public Manager(string name, string password) : base(name, password)

    {

    }

    public Manager() { }

    public void ListUsers()

    {

        if (!File.Exists(Utils.allUsersPathAndFile) || !Directory.Exists(Utils.appRootDirectoryPath))

        {

            Console.WriteLine("No registered users found.");

            Utils.WaitForEnter();

        }

        else

        {

            int index = 1;

            foreach (var row in File.ReadAllLines(Utils.allUsersPathAndFile))

            {

                var user = row.Split(';');

                System.Console.WriteLine($"User number: {index++} - {user[0]}");

            }

        }

    }

    public User? GetSelectedUser()

    {

        if (!File.Exists(Utils.allUsersPathAndFile) || !Directory.Exists(Utils.appRootDirectoryPath))

        {

            Console.WriteLine("No registered users found.");

            Utils.WaitForEnter();

            return null;

        }

        else

        {

            bool validIndex = false;

            int index;

            do

            {

                Console.Write("Input number of user to be selected: ");

                validIndex = int.TryParse(Console.ReadLine(), out index);

                if (!validIndex) Console.WriteLine("Invalid index. Try again.");

                else if (index < 0 || index > File.ReadAllLines(Utils.allUsersPathAndFile).Length)

                {

                    validIndex = false;

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

                }

                else index--; //convert to zero-based index

            } while (!validIndex);

            //Utils.WaitForEnter();

            string userLine = File.ReadAllLines(Utils.allUsersPathAndFile)[index];

            User selectedUser = new User

            {

                Name = userLine.Split(';')[0]

            };

            selectedUser.Tasks = selectedUser.GetTasksOfUser();

            return selectedUser;

        }

    }

    public void ManagerMenu()

    {

        bool endManagerMenu = false;

        do

        {

            Console.Clear();

            Console.WriteLine("---------------------------");

            Console.WriteLine("TASK TRACKER - Manager Menu");

            Console.WriteLine("---------------------------");

            Console.WriteLine("1) List Tasks");

            Console.WriteLine("2) Find Tasks");

            Console.WriteLine("3) Add Task");

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

            Console.WriteLine("5) Sign up new Manager");

            Console.WriteLine("0) Return to Main Menu");

            Console.WriteLine("---------------------------");

            Console.Write("Your choice (0-5): ");

            string choice = Console.ReadLine();

            switch (choice)

            {

                case "1":

                    {

                        ListUsers();

                        User? selectedUser = GetSelectedUser();

                        if (selectedUser != null)

                        {

                            selectedUser.ListTasks();

                            //List<Task> selectedUserTasks = selectedUser.GetTasksOfUser();

                        }

                        else System.Console.WriteLine("User not found.");

                        break;

                    }

                case "2":

                    {

                        ListUsers();

                        User? selectedUser = GetSelectedUser();

                        if (selectedUser != null)

                        {

                            selectedUser.FindTasks(); //of specific user

                        }

                        else System.Console.WriteLine("User not found.");

                        break;

                    }

                case "3":

                    {

                        AddTask();

                        break;

                    }

                case "4":

                    {

                        DeleteTask();

                        break;

                    }

                case "5":

                    {

                        Utils.SignUpManager();

                        break;

                    }

                case "0":

                    {

                        endManagerMenu = true;

                        break;

                    }

                default:

                    {

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

                        break;

                    }

            }

        } while (!endManagerMenu);

    }

    public void AddTask()

    {

        //List all users

        ListUsers();

        System.Console.WriteLine();

        //Select a user by index

        User selectedUser = GetSelectedUser();

        if (selectedUser == null)

        {

            System.Console.WriteLine("No user selected.");

            Utils.WaitForEnter();

            return;

        }

        //Get task properties

        string description;

        do

        {

            System.Console.WriteLine("Enter task description: ");

            description = Console.ReadLine();

            if (string.IsNullOrEmpty(description))

            {

                System.Console.WriteLine("Description cannot be empty.");

            }

        } while (string.IsNullOrEmpty(description));

        DateTime dueDate;

        System.Console.Write("Enter due date (dd.MM.yyyy): ");

        while (!DateTime.TryParseExact(Console.ReadLine(), Utils.supportedDateFormats, null, System.Globalization.DateTimeStyles.None, out dueDate))

        {

            System.Console.WriteLine("Invalid date format. Please use dd.MM.yyyy or dd/MM/yyyy: ");

            System.Console.Write("Enter due date (dd.MM.yyyy): ");

        }

        System.Console.Write("Is this a high priority task? (y/n): ");

        bool highPriority = Console.ReadLine().ToLower() == "y";

        //Create and add the new task, save the task

        Task newTask = new Task(description, dueDate, highPriority);

        List<Task> userTasks = selectedUser.GetTasksOfUser() ?? new List<Task>();

        userTasks.Add(newTask);

        selectedUser.SaveTasks(userTasks);

        System.Console.WriteLine("Task added successfully.");

        Utils.WaitForEnter();

    }

    public void DeleteTask()

    {

        //List users

        ListUsers();

        System.Console.WriteLine();

        //Select user by index

        User selectedUser = GetSelectedUser();

        if (selectedUser == null)

        {

            System.Console.WriteLine("No user selected.");

            Utils.WaitForEnter();

            return;

        }

        //Load and list tasks of selected user

        List<Task> userTasks = selectedUser.GetTasksOfUser();

        if (userTasks == null || userTasks.Count == 0)

        {

            System.Console.WriteLine("No tasks found for this user.");

            Utils.WaitForEnter();

            return;

        }

        System.Console.WriteLine($"Tasks of user {selectedUser.Name}:");

        for (int i = 0; i < userTasks.Count; i++)

        {

            System.Console.WriteLine($"Task number: {(i + 1).ToString().PadRight(3, ' ')} - {userTasks[i].Description.PadRight(20, '.')} Due: {userTasks[i].DueDate:dd.MM.yyyy}, High priority:{(userTasks[i].HighPriority ? "Yes" : "No")} Completed: {(userTasks[i].Completed ? "Yes" : "No")}");

        }

        //Select task to delete

        bool validIndex = false;

        int index;

        do

        {

            System.Console.WriteLine("Input number of task to delete: ");

            validIndex = int.TryParse(Console.ReadLine(), out index);

            if (!validIndex || index < 1 || index > userTasks.Count)

            {

                validIndex = false;

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

            }

            else

            {

                validIndex = true;

                index--; //convert to zero-based index;

            }

        } while (!validIndex);

        //Confirm task to delete

        System.Console.WriteLine($"Delete task {(index + 1).ToString().PadRight(3, ' ')} - {userTasks[index].Description.PadRight(20, '.')} Due: {userTasks[index].DueDate:dd.MM.yyyy}, Priority:{(userTasks[index].HighPriority ? "Yes" : "No")}? (y/n)");

        //Delete task and save

        if (Console.ReadLine().ToLower() == "y")

        {

            userTasks.RemoveAt(index);

            selectedUser.SaveTasks(userTasks);

            System.Console.WriteLine($"Task: {userTasks[index]} successfully deleted.");

            Utils.WaitForEnter();

        }

        else System.Console.WriteLine("No task deleted.");

        Utils.WaitForEnter();

    }

}

public class User : GeneralUser

{

    public List<Task> Tasks { get; set; }

    public User() { }

    //public User(string? name = null, string? password = null) : base(name, password)

    public User(string name, string password) : base(name, password)

    {

    }

    public User(string name, List<Task> tasks) : base(name) //to be corrected

    {

        Tasks = tasks;

    }

    public void UserMenu()

    {

        bool endUserMenu = false;

        do

        {

            Console.Clear();

            Console.WriteLine("------------------------");

            Console.WriteLine("TASK TRACKER - User Menu");

            Console.WriteLine("------------------------");

            Console.WriteLine("1) List Tasks");

            Console.WriteLine("2) Find Tasks");

            Console.WriteLine("3) Mark Task as Completed");

            Console.WriteLine("0) Return to Main Menu");

            Console.WriteLine("------------------------");

            Console.Write("Your choice (0-3): ");

            string choice = Console.ReadLine();

            switch (choice)

            {

                case "1":

                    {

                        ListTasks();

                        break;

                    }

                case "2":

                    {

                        FindTasks();

                        break;

                    }

                case "3":

                    {

                        MarkTaskAsCompleted();

                        break;

                    }

                case "0":

                    {

                        endUserMenu = true;

                        break;

                    }

                default:

                    {

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

                        break;

                    }

            }

        } while (!endUserMenu);

    }

    public void MarkTaskAsCompleted()

    {

        ListTasks();

        bool validIndex = false;

        int index;

        do

        {

            Console.Write("Input number of task to be marked as completed: ");

            validIndex = int.TryParse(Console.ReadLine(), out index);

            if (!validIndex) Console.WriteLine("Invalid number. Try again.");

            else index--;

        } while (!validIndex);

        if (index >= 0 && index < Tasks.Count)

        {

            Tasks[index].Completed = true;

            SaveTasks(Tasks);

            Console.WriteLine("Task marked as completed.");

        }

        else

        {

            Console.WriteLine("Invalid number.");

        }

        Utils.WaitForEnter();

    }

    public void SaveTasks(List<Task> tasks)

    {

        if (tasks != null)

        {

            XmlSerializer taskSerializer = new XmlSerializer(typeof(List<Task>));

            //string appDataPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);

            //string pathToDirectory = Path.Combine(appDataPath, "TaskTracker");

            if (!Directory.Exists(Utils.appRootDirectoryPath))

            {

                Directory.CreateDirectory(Utils.appRootDirectoryPath);

            }

            string userXmlFile = $"{Name}.xml";

            string pathToXmlFileInDirectory = Path.Combine(Utils.appRootDirectoryPath, userXmlFile);

            using (StreamWriter writer = new StreamWriter(pathToXmlFileInDirectory))

            {

                taskSerializer.Serialize(writer, tasks);

            }

        }

        else System.Console.WriteLine("No tasks to be saved.");

        /\*var rows = new List<string>();

            foreach (var t in tasks)

            {

                rows.Add($"{t.Description};{t.HighPriority};{t.DueDate.ToString("dd.MM.yyyy")};{t.Completed}");

            }

            File.WriteAllLines($"{Name}.txt", rows);\*/

    }

    public List<Task> GetTasksOfUser()

    {

        if (!Directory.Exists(Utils.appRootDirectoryPath))

        {

            Directory.CreateDirectory(Utils.appRootDirectoryPath);

        }

        string userXmlFile = $"{Name}.xml";

        string pathToXmlFileInDirectory = Path.Combine(Utils.appRootDirectoryPath, userXmlFile);

        if (!File.Exists(pathToXmlFileInDirectory))

        {

            System.Console.WriteLine("Tasks not found.");

            Utils.WaitForEnter();

            return new List<Task>(); //return empty list (not null)

        }

        //check if file is empty

        var fielInfo = new FileInfo(pathToXmlFileInDirectory);

        if (fielInfo.Length == 0)

        {

            return new List<Task>();

        }

        try

        {

            XmlSerializer taskSerializer = new XmlSerializer(typeof(List<Task>));

            using (StreamReader reader = new StreamReader(pathToXmlFileInDirectory))

            {

                return taskSerializer.Deserialize(reader) as List<Task>;

            }

        }

        catch (InvalidOperationException ex)

        {

            System.Console.WriteLine("Error in XML format: " + ex.Message);

            return new List<Task>();

        }

        catch (XmlException ex)

        {

            System.Console.WriteLine("Error while parsing XML: " + ex.Message);

            return new List<Task>();

        }

        catch (IOException ex)

        {

            System.Console.WriteLine("Error while reading file: " + ex.Message);

            return new List<Task>();

        }

        catch (Exception ex)

        {

            System.Console.WriteLine("Unexpected error: " + ex.Message);

            return new List<Task>();

        }

    }

public void ListTasks()

    {

        System.Console.WriteLine($"User: {Name} - List of tasks: ");

        int index = 1;

        if (Tasks.Count == 0)

        {

            System.Console.WriteLine("No tasks found.");

            Utils.WaitForEnter();

        }

        else

        {

            GetTasksOfUser();

            foreach (Task t in Tasks)

            {

                System.Console.WriteLine($"Task number: {index++.ToString().PadRight(3, ' ')} - {t.Description.PadRight(20, '.')}; High priority: {(t.HighPriority ? "Yes" : "No")}; Due: {t.DueDate.ToString("dd.MM.yyyy")}; Completed: {(t.Completed ? "Yes" : "No")}");

            }

        }

        Utils.WaitForEnter();

    }

    public void FindTasks()

    {

        Console.Write("Input text to be found in task description: ");

        string findText = Console.ReadLine().ToLower();

        var foundTasks = GetTasksOfUser().FindAll(t => t.Description.ToLower().Contains(findText));

        if (foundTasks.Count == 0)

        {

            Console.WriteLine("No tasks found.");

        }

        else

        {

            foreach (var t in foundTasks)

            {

                Console.WriteLine(t);

            }

        }

        Utils.WaitForEnter();

    }

}

public class Task

{

    public string Description { get; set; }

    public bool HighPriority { get; set; }

    public DateTime DueDate { get; set; }

    public bool Completed { get; set; }

    public Task(string description, DateTime dueDate, bool highPriority = false, bool completed = false)

    {

        Description = description;

        HighPriority = highPriority;

        DueDate = dueDate;

        Completed = completed;

    }

    public Task() { } //parameterless constructor for XmlSerializer

}

public class Utils

{

    //for public static methods and var

    public static string appDataPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);

    public static string appRootDirectoryPath = Path.Combine(appDataPath, "TaskTracker");

    public static string allUsersFile = "users.txt";

    public static string allManagersFile = "managers.txt";

    public static string defaultManagerName = "firstmanager";

    public static string defaultManagerPassword = "bigboss";

    public static string allUsersPathAndFile = Path.Combine(appRootDirectoryPath, allUsersFile);

    public static string allManagersPathAndFile = Path.Combine(appRootDirectoryPath, allManagersFile);

    public static char[] invalidFileNameChar = Path.GetInvalidFileNameChars();

    public static string[] supportedDateFormats = { "dd.MM.yyyy", "dd/MM/yyyy", "dd-MM-yyyy", "dd.M.yyyy", "dd/M/yyyy", "dd-M-yyyy" };

    public static string EncodePassword(string password)

    {

        return Convert.ToBase64String(Encoding.UTF8.GetBytes(password));

    }

    public static string DecodePassword(string codedPassword)

    {

        return Encoding.UTF8.GetString(Convert.FromBase64String(codedPassword));

    }

    public static User? LogInUser()

    {

        //check if file is empty or non-existent

        var fileInfo = new FileInfo(Utils.allUsersPathAndFile);

        if (!File.Exists(allUsersPathAndFile) || !Directory.Exists(appRootDirectoryPath) || fileInfo.Length == 0)

        {

            Console.WriteLine("No registered users found.");

            Utils.WaitForEnter();

            return null;

        }

        Console.Write("Input user name: ");

        string name = Console.ReadLine();

        Console.Write("Input password: ");

        string password = Console.ReadLine();

        string codedPassword = EncodePassword(password);

        foreach (var row in File.ReadAllLines(allUsersPathAndFile))

        {

            var data = row.Split(';');

            if (data.Length >= 2 && data[0] == name && data[1] == codedPassword)

            {

                User loggedUser = new User

                {

                    Name = name

                };

                loggedUser.Tasks = loggedUser.GetTasksOfUser();

                Console.WriteLine($"Login successful, welcome {loggedUser.Name}.");

                Utils.WaitForEnter();

                return loggedUser;

            }

        }

        Console.WriteLine("Invalid credentials.");

        Utils.WaitForEnter();

        return null;

    }

    public static Manager? LogInManager()

    {

        string name;

        string password;

        string codedPassword;

        do

        {

            Console.Write("Input manager name: ");

            name = Console.ReadLine();

        } while (string.IsNullOrEmpty(name));

        do

        {

            Console.Write("Input password: ");

            password = Console.ReadLine();

            codedPassword = EncodePassword(password);

        } while (string.IsNullOrEmpty(password));

        if (!File.Exists(allManagersPathAndFile) || !Directory.Exists(appRootDirectoryPath))

        {

            Console.WriteLine("No registered managers found.");

            Utils.WaitForEnter();

            return null;

        }

        foreach (var row in File.ReadAllLines(allManagersPathAndFile))

        {

            var data = row.Split(';');

            if (data.Length >= 2 && data[0] == name && data[1] == codedPassword)

            {

                Manager loggedManager = new Manager

                {

                    Name = name

                };

                Console.WriteLine($"Login successful, welcome {loggedManager.Name}.");

                Utils.WaitForEnter();

                return loggedManager;

            }

        }

        Console.WriteLine("Invalid credentials.");

        Utils.WaitForEnter();

        return null;

    }

    public static void SignUpUser()

    {

        int minLength = 3;

        bool repeatInput;

        string name;

        string password;

        string codedPassword;

        do

        {

            repeatInput = false;

            Console.Write("Input new user name: ");

            name = Console.ReadLine();

            if (!hasRequiredMinLength(name, minLength))

            {

                System.Console.WriteLine($"User name must have at least {minLength} characters. Try again.");

                repeatInput = true;

            }

            if (ExistsUserName(name) || ExistsUserNameFile(name))

            {

                System.Console.WriteLine($"User name already exists. Try again.");

                repeatInput = true;

            }

            foreach (char i in name)

                foreach (char j in invalidFileNameChar)

                {

                    if (i == j)

                    {

                        repeatInput = true;

                        System.Console.Write($"User name must not contain characters: ");

                        foreach (char item in invalidFileNameChar)

                        {

                            Console.Write($"{item} ");

                        }

                        System.Console.WriteLine();

                    }

                }

        } while (repeatInput);

        do

        {

            repeatInput = false;

            Console.Write($"Input password (min {minLength} characters): ");

            password = Console.ReadLine();

            if (!hasRequiredMinLength(password, minLength))

            {

                System.Console.WriteLine($"Password must have at least {minLength} characters. Try again.");

                repeatInput = true;

                continue;

            }

            Console.Write($"Input password for verification: ");

            string password2 = Console.ReadLine();

            if (password != password2)

            {

                repeatInput = true;

                System.Console.WriteLine("Password does not match. Try again.");

            }

        } while (repeatInput);

        codedPassword = EncodePassword(password);

        string userAndCodedPassword = $"{name};{codedPassword}\n";

        string userFile = $"{name}.xml";

        string userPathAndFile = Path.Combine(appRootDirectoryPath, userFile);

        if (!ExistsUserNameFile(name))

        {

            File.AppendAllText(allUsersPathAndFile, userAndCodedPassword);

            File.Create(userPathAndFile).Close();

            List<Task> emptyTasks = new List<Task>();

            XmlSerializer serializer = new XmlSerializer(typeof(List<Task>));

            using (var stream = new FileStream(userPathAndFile, FileMode.Create))

            {

                serializer.Serialize(stream, emptyTasks);

            }

            Console.WriteLine($"Sign up of user {name} successfully completed.");

        }

        else

        {

            System.Console.WriteLine($"Warning: Sign up of user {name} failed.");

        }

    }

    public static void SignUpManager()

    {

        int minLength = 3;

        bool repeatInput;

        string name;

        string password;

        string codedPassword;

        do

        {

            repeatInput = false;

            Console.Write("Input new manager name: ");

            name = Console.ReadLine();

            if (!hasRequiredMinLength(name, minLength))

            {

                System.Console.WriteLine($"User name must have at least {minLength} characters. Try again.");

                repeatInput = true;

            }

            if (Utils.ExistsManagerName(name))

            {

                System.Console.WriteLine($"Manager name already exists. Try again.");

                repeatInput = true;

            }

            foreach (char i in name)

                foreach (char j in invalidFileNameChar)

                {

                    if (i == j)

                    {

                        repeatInput = true;

                        System.Console.Write($"Manager name must not contain characters: ");

                        foreach (char item in invalidFileNameChar)

                        {

                            Console.Write($"{item} ");

                        }

                        System.Console.WriteLine();

                    }

                }

        } while (repeatInput);

        do

        {

            repeatInput = false;

            Console.Write($"Input password (min {minLength} characters): ");

            password = Console.ReadLine();

            if (!hasRequiredMinLength(password, minLength))

            {

                System.Console.WriteLine($"Password must have at least {minLength} characters. Try again.");

                repeatInput = true;

                continue;

            }

            Console.Write($"Input password for verification): ");

            string password2 = Console.ReadLine();

            if (password != password2)

            {

                repeatInput = true;

                System.Console.WriteLine("Password does not match. Try again.");

            }

        } while (repeatInput);

        codedPassword = EncodePassword(password);

        string managerAndCodedPassword = $"{name};{codedPassword}\n";

        if (ExistsAllManagersFile() && File.ReadAllLines(allManagersPathAndFile)[0].Split(';')[0] == defaultManagerName)

        {

            File.WriteAllText(allManagersPathAndFile, managerAndCodedPassword);

        }

        else if (ExistsAllManagersFile() && File.ReadAllLines(allManagersPathAndFile)[0].Split(';')[0] != defaultManagerName)

        {

            File.AppendAllText(allManagersPathAndFile, managerAndCodedPassword);

            Console.WriteLine($"Sign up of manager {name} successfully completed.");

        }

        else

        {

            System.Console.WriteLine($"Warning: Sign up of manager {name} failed.");

        }

    }

    public static bool hasRequiredMinLength(string text, int minLength)

    {

        if (text.Length >= minLength) return true;

        else return false;

    }

    public static bool ExistsDirectory()

    {

        if (Directory.Exists(appRootDirectoryPath)) return true;

        return false;

    }

    public static bool ExistsAllUsersFile()

    {

        if (File.Exists(allUsersPathAndFile)) return true;

        return false;

    }

    public static bool ExistsAllManagersFile()

    {

        if (File.Exists(allManagersPathAndFile)) return true;

        return false;

    }

    public static bool ExistsUserNameFile(string userName)

    {

        string userFile = $"{userName}.xml";

        string userPathAndFile = Path.Combine(appRootDirectoryPath, userFile);

        if (File.Exists(userPathAndFile)) return true;

        return false;

    }

    public static bool ExistsUserName(string userName)

    {

        if (!File.Exists(allUsersPathAndFile) || !Directory.Exists(appRootDirectoryPath)) return false;

        foreach (var row in File.ReadAllLines(allUsersPathAndFile))

        {

            var data = row.Split(';');

            if (data.Length >= 1 && data[0] == userName) return true;

        }

        return false;

    }

    public static bool ExistsManagerName(string managerName)

    {

        if (!File.Exists(allManagersPathAndFile) || !Directory.Exists(appRootDirectoryPath)) return false;

        foreach (var row in File.ReadAllLines(allManagersPathAndFile))

        {

            var data = row.Split(';');

            if (data.Length >= 1 && data[0] == managerName) return true;

        }

        return false;

    }

    public static void WaitForEnter()

    {

        Console.WriteLine("Press Enter to continue...");

        Console.ReadLine();

    }

}

class Program

{

    static void Main(string[] args)

    {

        User? loggedUser = null;

        Manager? loggedManager = null;

        if (!Utils.ExistsDirectory()) Directory.CreateDirectory(Utils.appRootDirectoryPath);

        if (!Utils.ExistsAllUsersFile()) File.Create(Utils.allUsersPathAndFile).Close();

        if (!Utils.ExistsAllManagersFile())

        {

            string managerAndCodedPassword = $"{Utils.defaultManagerName};{Utils.EncodePassword(Utils.defaultManagerPassword)}\n";

            File.WriteAllText(Utils.allManagersPathAndFile, managerAndCodedPassword);

        }

        do

        {

            if (loggedUser == null && loggedManager == null)

            {

                Console.Clear();

                Console.WriteLine("========================");

                Console.WriteLine("TASK TRACKER - Main Menu");

                Console.WriteLine("========================");

                Console.WriteLine("1) Log in as User");

                Console.WriteLine("2) Log in as Manager");

                Console.WriteLine("3) Sign up as new User");

                Console.WriteLine("4) Log out");

                Console.WriteLine("0) End program");

                Console.WriteLine("------------------------");

                Console.Write("Your choice (0-4): ");

                string choice = Console.ReadLine();

                switch (choice)

                {

                    case "1":

                        {

                            loggedUser = Utils.LogInUser();

                            break;

                        }

                    case "2":

                        {

                            loggedManager = Utils.LogInManager();

                            break;

                        }

                    case "3":

                        {

                            Utils.SignUpUser();

                            break;

                        }

                    case "4":

                        {

                            if (loggedUser != null) loggedUser.LogOut(); //inherited from GeneralUser

                            if (loggedManager != null) loggedManager.LogOut();

                            break;

                        }

                    case "0":

                        {

                            System.Console.WriteLine("Ending program.");

                            Console.WriteLine("Press Enter to continue...");

                            Console.ReadLine();

                            return;

                        }

                    default:

                        {

                            System.Console.WriteLine("Invalid input. Try again.");

                            break;

                        }

                }

            }

            else if (loggedUser != null)

            {

                loggedUser.UserMenu();

                loggedUser = null;

            }

            else if (loggedManager != null)

            {

                loggedManager.ManagerMenu();

                loggedManager = null;

            }

        } while (true);

    }

}