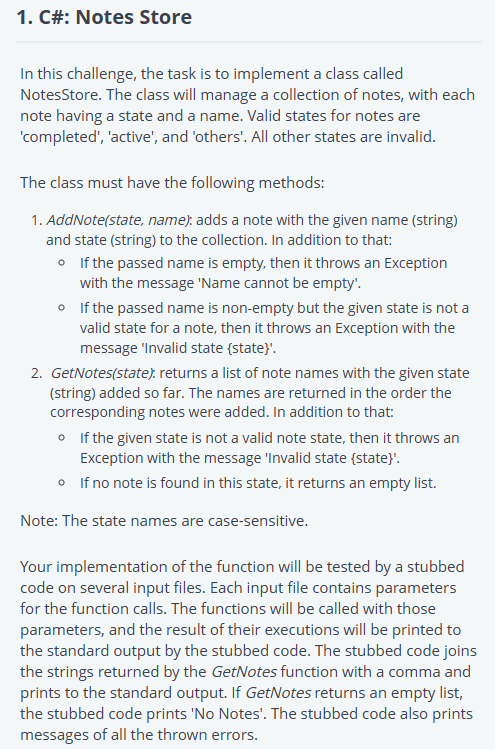
# C#: Notes Store



using System;

using System.Collections.Generic;

namespace Solution

{

    public class NotesStore

    {

        private Dictionary<string, string> notes = new Dictionary<string, string>();

        public NotesStore() { }

        public void AddNote(string state, string name)

        {

            // Check if the name is empty

            if (string.IsNullOrEmpty(name))

                throw new Exception("Name cannot be empty");

            // Check if the state is valid

            if (state != "completed" && state != "active" && state != "others")

                throw new Exception($"Invalid state {state}");

            // Add the note to the collection

            notes[name] = state;

        }

        public List<string> GetNotes(string state)

        {

            // Check if the state is valid

            if (state != "completed" && state != "active" && state != "others")

                throw new Exception($"Invalid state {state}");

            // Get the list of notes with the given state

            List<string> result = new List<string>();

            foreach (var note in notes)

            {

                if (note.Value == state)

                    result.Add(note.Key);

            }

            return result;

        }

    }

    public class Solution

    {

        public static void Main()

        {

            var notesStoreObj = new NotesStore();

            var n = int.Parse(Console.ReadLine());

            for (var i = 0; i < n; i++)

            {

                var operationInfo = Console.ReadLine().Split(' ');

                try

                {

                    if (operationInfo[0] == "AddNote")

                        notesStoreObj.AddNote(operationInfo[1], operationInfo.Length == 2 ? "" : operationInfo[2]);

                    else if (operationInfo[0] == "GetNotes")

                    {

                        var result = notesStoreObj.GetNotes(operationInfo[1]);

                        if (result.Count == 0)

                            Console.WriteLine("No Notes");

                        else

                            Console.WriteLine(string.Join(",", result));

                    }

                    else

                    {

                        Console.WriteLine("Invalid Parameter");

                    }

                }

                catch (Exception e)

                {

                    Console.WriteLine("Error: " + e.Message);

                }

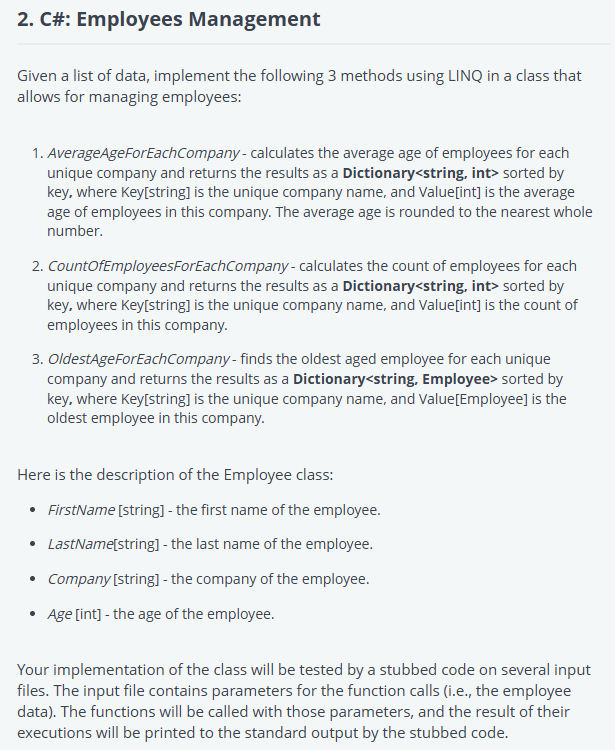
            }

        }

    }

}

# C#: Employees Management



using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Solution

{

    public class Solution

    {

        public static Dictionary<string, int> AverageAgeForEachCompany(List<Employee> employees)

        {

            return employees.GroupBy(employee => employee.Company)

                            .Select(group => new { Company = group.Key, Ages = group.Select(employee =>                                 employee.Age )})

                            .OrderBy(obj => obj.Company)

                            .ToDictionary(obj => obj.Company, obj => (int)Math.Round(obj.Ages.Average()));

        }

        public static Dictionary<string, int> CountOfEmployeesForEachCompany(List<Employee> employees)

        {

            return employees.GroupBy(employee => employee.Company)

                            .OrderBy(group => group.Key)

                            .ToDictionary(group => group.Key, group => group.Count());

        }

        public static Dictionary<string, Employee> OldestAgeForEachCompany(List<Employee> employees)

        {

            return employees.GroupBy(employee => employee.Company)

                .OrderBy(group => group.Key)

                .ToDictionary(group => group.Key, group => group.Where(emp => emp.Age == group.Max                          (employee => employee.Age)).First());

        }

        public static void Main()

        {

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

            var employees = new List<Employee>();

            for (int i = 0; i < countOfEmployees; i++)

            {

                string str = Console.ReadLine();

                string[] strArr = str.Split(' ');

                employees.Add(new Employee {

                    FirstName = strArr[0],

                    LastName = strArr[1],

                    Company = strArr[2],

                    Age = int.Parse(strArr[3])

                    });

            }

            foreach (var emp in AverageAgeForEachCompany(employees))

            {

                Console.WriteLine($"The average age for company {emp.Key} is {emp.Value}");

            }

            foreach (var emp in CountOfEmployeesForEachCompany(employees))

            {

                Console.WriteLine($"The count of employees for company {emp.Key} is {emp.Value}");

            }

            foreach (var emp in OldestAgeForEachCompany(employees))

            {

                Console.WriteLine($"The oldest employee of company {emp.Key} is {emp.Value.FirstName} {emp.Value.LastName} having age {emp.Value.Age}");

            }

        }

    }

    public class Employee

    {

        public string FirstName { get; set; }

        public string LastName { get; set; }

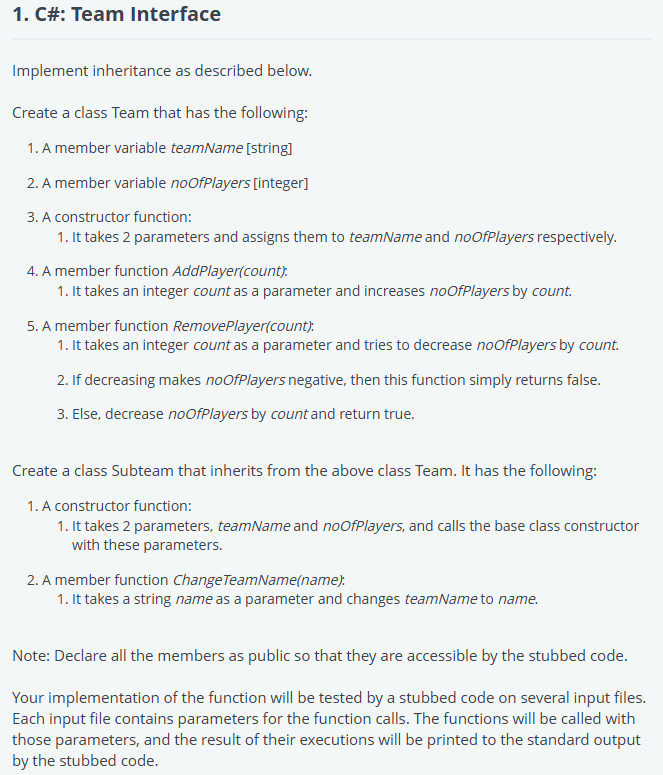
        public int Age { get; set; }

        public string Company { get; set; }

    }

}

# C#: Team Interface



using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Solution {

    public class Team {

        public string teamName;

        public int noOfPlayers;

        public Team(string teamName, int noOfPlayers) {

            this.teamName = teamName;

            this.noOfPlayers = noOfPlayers;

        }

        public void AddPlayer(int count) {

            noOfPlayers += count;

        }

        public bool RemovePlayer(int count) {

            if(noOfPlayers - count < 0) {

                return false;

            }

            noOfPlayers -= count;

            return true;

        }

    }

    public class Subteam : Team {

        public Subteam(string teamName, int noOfPlayers) : base(teamName, noOfPlayers) {}

        public void ChangeTeamName(string name) {

            this.teamName = name;

        }

    }

    class Solution {

         static void Main(string[] args) {

            if (!typeof(Subteam).IsSubclassOf(typeof(Team))) {

                throw new Exception("Subteam class should inherit from Team class");

            }

            String str = Console.ReadLine();

            String[] strArr = str.Split();

            string initialName = strArr[0];

            int count = Convert.ToInt32(strArr[1]);

            Subteam teamObj = new Subteam(initialName, count);

            Console.WriteLine("Team " + teamObj.teamName + " created");

            str = Console.ReadLine();

            count = Convert.ToInt32(str);

            Console.WriteLine("Current number of players in team " + teamObj.teamName + " is " + teamObj.noOfPlayers);

            teamObj.AddPlayer(count);

            Console.WriteLine("New number of players in team " + teamObj.teamName + " is " + teamObj.noOfPlayers);

            str = Console.ReadLine();

            count = Convert.ToInt32(str);

            Console.WriteLine("Current number of players in team " + teamObj.teamName + " is " + teamObj.noOfPlayers);

            var res = teamObj.RemovePlayer(count);

            if (res) {

                Console.WriteLine("New number of players in team " + teamObj.teamName + " is " + teamObj.noOfPlayers);

            } else {

                Console.WriteLine("Number of players in team " + teamObj.teamName + " remains same");

            }

            str = Console.ReadLine();

            teamObj.ChangeTeamName(str);

            Console.WriteLine("Team name of team " + initialName + " changed to " + teamObj.teamName);

        }

    }

}