Kauno technologijos universitetas

Informatikos fakultetas

Objektinis programavimas 2 (P175B123)

Laboratorinių darbų ataskaita

Normantas Stankevičius IFF-1/4

Studentas

Prof. Vacius Jusas

Dėstytojas

Kaunas 2022

1. Rekursija (L1)

1.1. Darbo užduotis

LD_16.Pažintis.

Įvairių miesto mokyklų geriausi moksleiviai važiuoja į ekskursiją. Nors moksleiviai yra iš skirtingų mokyklų, tačiau yra tokių, kurie pažįsta vieni kitus. Moksleiviai nori užmegzti naujas pažintis, tačiau su nepažįstamu moksleiviu galima susipažinti tik tuomet, jeigu yra pažįstamų moksleivių grandinėlė (pirmas pažįsta antrą, antras pažįsta trečią, trečias pažįsta ketvirtą, tuomet pirmas gali susipažinti su ketvirtu), kuri veda iki nepažįstamo moksleivio. Pirmame tekstiniame faile 'U31DUOM.TXT' apie moksleivius pateikta tokia informacija: moksleivio vardas, jo pažįstamų moksleivių kiekis, pažįstamų moksleivių vardai. Kiekvienam moksleiviui tekstiniame faile yra skirta po vieną eilutę. Antrame tekstiniame faile 'U32DUOM.TXT' vienoje

eilutėje nurodyti dviejų moksleivių vardai. Tokių eilučių gali būti keletas. Abiejuose failuose moksleivių duomenys skiriami bent vienu tarpu.

Nustatykite kiekvienai moksleivių porai iš antrojo failo ar jie jau yra pažįstami, ar jie gali susipažinti (jeigu gali, reikia nurodyti visus bendrus pažįstamus moksleivius), ar jie negali susipažinti (bendro pažįstamo moksleivio neturi). Spausdinkite poros vardus, šalia nurodant atsakymą, kaip žemiau pateiktame pavyzdyje.

Pirmasis duomenų failas 'U31DUOM.TXT':

```
Rūta      1 Arnoldas
Agnė      3 Nerijus Neda Antanas
Nerijus   1 Agnė
Antanas   2 Agnė Marius
Marius    2 Antanas  Neda
Neda      3 Marius Rūta Agnė
Arnoldas 1 Rūta
```

Antrasis duomenų failas 'U32DUOM.TXT':

```
Rūta       Nerijus
Agnė       Antanas
Neda       Nerijus
```

Rezultatų failas 'U3REZ.TXT':

```
Rūta       Nerijus    negali susipažinti
Agnė       Antanas    jau pažįstami
Neda       Nerijus    bendri pažįstami: Agnė
```

1.2. Grafinės vartotojo sąsajos schema

```
body
Lab01-16   HeaderLabel

Studentų duomenys: StudentLabel
###  StudentTable

Studentų Ieškomi Junginiai:   ConnectionLabel
###  ConnectionTable

Rezultatai: OutputLabel
###  PathTable
```

1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
HeaderLabel	Text	"Lab01-16"
StudentLabel	Text	"Studentų duomenys:"
ConnectionLabel	Text	"Studentų Ieškomi Junginiai:"
OutputLabel	Text	"Rezultatai:"

1.4. Klasių diagrama
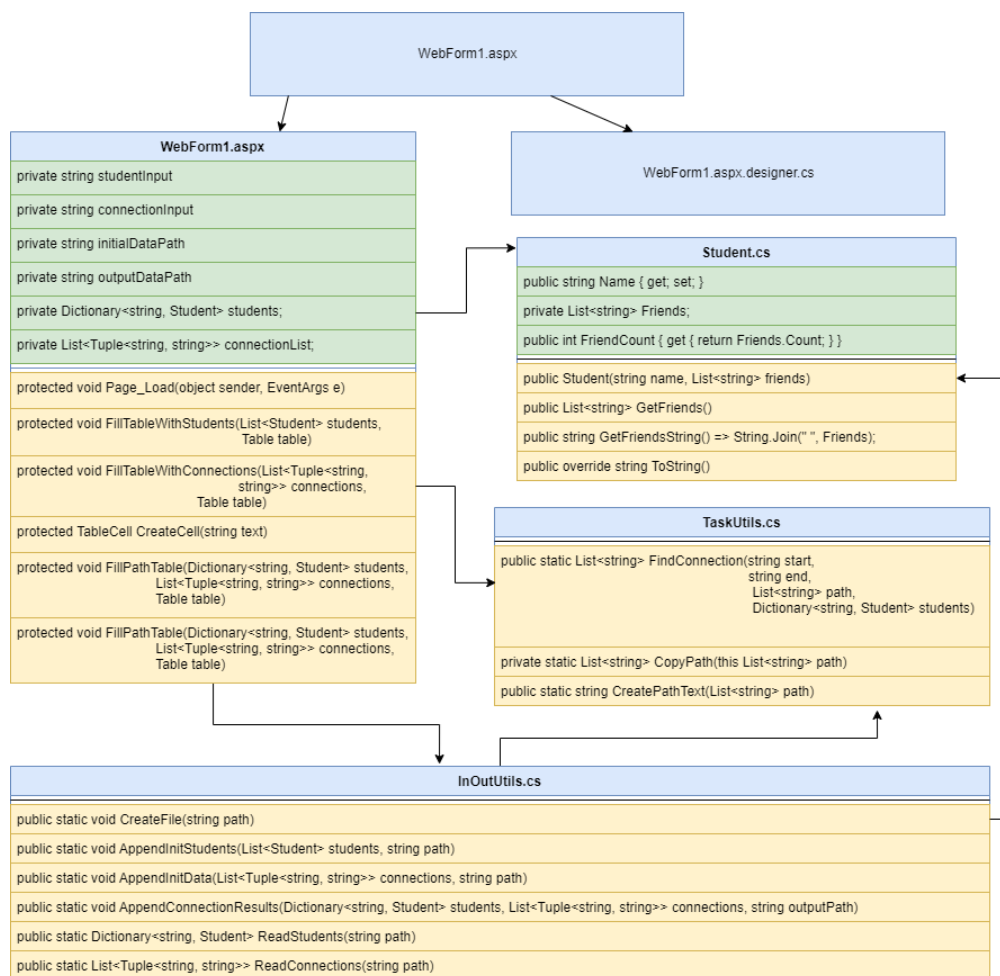
1.5. Programos vartotojo vadovas

Atsidarius programą, programa nuskaito App_Data/students.txt ir App_Data/connections.txt. Naudojant tą informaciją, parašo visą informaciją į StudentTable, ConnectionTable, PathTable su duota ir apskaičiuota informacija.

1.6. Programos tekstas

InOutUtils.cs:

```csharp
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;

namespace Lab01
{
    /// <summary>
    /// InOutUtils class for reading and writing data from/to a file
    /// </summary>
    public static class InOutUtils
    {
        /// <summary>
        /// Creates a new empty file, ready for appending data
```

```csharp
        /// </summary>
        /// <param name="path">path to the file</param>
        public static void CreateFile(string path)
        {
            using (FileStream fs = new FileStream(path, FileMode.Create))
                new StreamWriter(fs, encoding: System.Text.Encoding.UTF8).Close();
        }


        /// <summary>
        /// appends initial student data to TXT file
        /// </summary>
        /// <param name="students">List of all students (Student object)</param>
        /// <param name="path">path to the file where information will be
appended</param>
        public static void AppendInitStudents(List<Student> students, string path)
        {
            using (StreamWriter sr = new StreamWriter(path, append: true))
            {
                sr.WriteLine("Studentai ir jų draugai");
                sr.WriteLine($"{"Studentas",-20}|{"Draugų kiekis",-20}|{"Draugai:"}");
                foreach (Student student in students)
                    sr.WriteLine(student);
                sr.WriteLine();
            }
        }

        /// <summary>
        /// Appends initial connection data to output file
        /// </summary>
        /// <param name="connections">List of Tuples(string, string) that work as nodes
from student a to student b while using DFS</param>
        /// <param name="path">path to the file where to append initial data</param>
        public static void AppendInitData(List<Tuple<string, string>> connections, string
path)
        {
            using (StreamWriter sr = new StreamWriter(path, append: true))
            {
                sr.WriteLine("Studentai ir jų ieškomi draugai:");
                sr.WriteLine($"{"Studentas",-20} {"Ieškomas draugas",-20}");
                foreach (Tuple<string, string> connection in connections)
                    sr.WriteLine($"{connection.Item1,-20} {connection.Item2,-20}");
                sr.WriteLine();
            }
        }

        /// <summary>
        /// Appends output connection data to output file
        /// </summary>
        /// <param name="students">Dictionary, key -> string, name of the student, value
-> Student class object of the student</param>
        /// <param name="connections">List of tuples(string, string) that is compromised
of student names that work as nodes that are used for DFS</param>
        /// <param name="outputPath">output path to the txt file where data will be
APPENDED</param>
        public static void AppendConnectionResults(Dictionary<string, Student> students,
List<Tuple<string, string>> connections, string outputPath)
        {

            using (StreamWriter sr = new StreamWriter(outputPath))
            {
                sr.WriteLine("Draugai ir jų junginiai, bei keliai:");
                sr.WriteLine($"{"Draugas",-20}|{"Ieškomas draugas:",-20}|{"Kelias:"}");
                foreach (Tuple<string, string> connection in connections)
                {
                    List<string> studentPath = new List<string>();
                    studentPath.Add(connection.Item1);
```

```csharp
                studentPath = TaskUtils.FindConnection(connection.Item1,
connection.Item2, studentPath, students);
                string pathText = TaskUtils.CreatePathText(studentPath);
                sr.WriteLine($"{connection.Item1,-20}|{connection.Item2,-
20}|{pathText}");
            }
        }
    }

    /// <summary>
    /// Creates a name to Student class object relation dictionary
    /// </summary>
    /// <param name="path">Path to the the text file containing the data</param>
    /// <returns>Dictionary(key -> string, value -> Student class object) </returns>
    public static Dictionary<string, Student> ReadStudents(string path)
    {
        Dictionary<string, Student> students = new Dictionary<string, Student>();
        using (StreamReader sr = new StreamReader(path))
        {
            string line;
            while ((line = sr.ReadLine()) != null)
            {
                string[] elements = line.Split(' ');
                string name = elements[0];
                List<string> friends = new List<string>();
                for (int i = 2; i < elements.Length; i++)
                    friends.Add(elements[i]);

                students.Add(name, new Student(name, friends));
            }
        }
        return students;
    }

    /// <summary>
    /// Gets the connections of students
    /// </summary>
    /// <param name="path">.txt file to the input</param>
    /// <returns>List of Tupples(string, string)</returns>
    public static List<Tuple<string, string>> ReadConnections(string path)
    {
        List<Tuple<string, string>> conncetions = new List<Tuple<string, string>>();
        using (StreamReader sr = new StreamReader(path))
        {
            string line;
            while ((line = sr.ReadLine()) != null)
            {
                string[] elements = line.Split(' ');
                conncetions.Add(new Tuple<string, string>(elements[0], elements[1]));
            }
        }

        return conncetions;
    }
}
}
```

TaskUtils.cs:

```csharp
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
```

```csharp
namespace Lab01
{
    /// <summary>
    /// TaskUtils class for extra (backend) computation functions
    /// </summary>
    public static class TaskUtils
    {

        /// <summary>
        /// Recursive implementation of DFS
        /// </summary>
        /// <param name="start">Start of the person</param>
        /// <param name="end">End of the person</param>
        /// <param name="path">path to current position from initial start</param>
        /// <param name="students">Dictionary, key: string (name of the student), value
Student class object</param>
        /// <returns>List of strings, that create a path from student a to b</returns>
        public static List<string> FindConnection(string start, string end, List<string>
path, Dictionary<string, Student> students)
        {
            Student curr = students[start];
            List<string> outputPath = null;
            foreach(string next in curr.GetFriends())
            {
                if (next == end)
                    return path;

                else if (path.Contains(next)) // Checks if the current node has been
visited, so it does not loop
                    continue;

                Student nextStudent = students[next];
                List<string> pathCopy = path.CopyPath();
                pathCopy.Add(next);

                List<String> pathToEnd = FindConnection(next, end, pathCopy, students);
// Recursion Call

                if(outputPath == null || (pathToEnd != null && pathToEnd.Count <
outputPath.Count))
                    outputPath = pathToEnd;

            }

            return outputPath; // Did not found the path
        }

        /// <summary>
        /// Deep copies a string list
        /// </summary>
        /// <param name="path">string list</param>
        /// <returns>string list</returns>
        private static List<string> CopyPath(this List<string> path)
        {
            List<string> copy = new List<string>();
            foreach (string s in path)
                copy.Add(s);

            return copy;
        }

        /// <summary>
        /// Creates connection depending on the path
        /// </summary>
        /// <param name="path"> List of strings that the path is compromised of </param>
        /// <returns>a string form of the path from student a to student b</returns>
        public static string CreatePathText(List<string> path)
```

```csharp
        {
            if (path == null)
                return "negali susipažinti";
            else if (path.Count == 1)
                return "jau pažįstami";
            else
            {
                path.RemoveAt(0);
                return $"bendri pažįstami: {String.Join(" ", path)}";
            }
        }


    }
}


Student.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab01
{
    /// <summary>
    /// Student Class Data Object that stores the name and connection
    /// </summary>
    public class Student
    {
        public string Name { get; set; }
        private List<string> Friends;
        public int FriendCount { get { return Friends.Count; } }

        /// <summary>
        /// Constructor
        /// </summary>
        public Student(string name, List<string> friends)
        {
            Name = name;
            Friends = new List<string>();
            foreach (string friend in friends)
                Friends.Add(friend);
        }

        /// <summary>
        /// Copies friends
        /// </summary>
        /// <returns>Deep copy of Friends List</returns>
        public List<string> GetFriends()
        {
            List<string> friendList = new List<string>();
            foreach (string friend in Friends)
                friendList.Add(friend);

            return friendList;
        }

        /// <summary>
        /// Transforms Friends list into a string seperated by spaces
        /// </summary>
        /// <returns>string of all friends</returns>
        public string GetFriendsString() => String.Join(" ", Friends);

        /// <summary>
```

```csharp
        /// ToString Override
        /// </summary>
        /// <returns>string version of the object: Name, Friend Count, Friends</returns>
        public override string ToString()
        {
            return $"{Name,-20}|{Friends.Count,20}|{GetFriendsString()}";
        }
    }
}
```

WebForm1.aspx:

```aspx
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="Lab01.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="HeaderLabel" runat="server" Text="Lab01-16"></asp:Label>
            <br />
            <br />
            <asp:Label ID="StudentLabel" runat="server" Text="Studentų
duomenys:"></asp:Label>
            <br />
            <asp:Table ID="StudentTable" runat="server">
            </asp:Table>
            <br />
            <asp:Label ID="ConnectionLabel" runat="server" Text="Studentų Ieškomi
Junginiai:"></asp:Label>
            <br />
            <asp:Table ID="ConnectionTable" runat="server">
            </asp:Table>
            <br />
            <asp:Label ID="OutputLabel" runat="server" Text="Rezultatai:"></asp:Label>
            <br />
            <asp:Table ID="PathTable" runat="server">
            </asp:Table>
        </div>
    </form>
</body>
</html>
```

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Lab01
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        private string studentInput = @"App_Data/students.txt";
        private string connectionInput = @"App_Data/connections.txt";
        private string initialDataPath = @"App_Data/initial_data.txt";
        private string outputDataPath = @"App_Data/result.txt";
```

```csharp
        private Dictionary<string, Student> students;
        private List<Tuple<string, string>> connectionList;
        protected void Page_Load(object sender, EventArgs e)
        {
            // Initial Data
            InOutUtils.CreateFile(Server.MapPath(initialDataPath));
            students = InOutUtils.ReadStudents(Server.MapPath(studentInput));

            FillTableWithStudents(new List<Student>(students.Values),
                                  StudentTable);

            InOutUtils.AppendInitStudents(new List<Student>(students.Values),
                                          Server.MapPath(initialDataPath));

            connectionList = InOutUtils.ReadConnections(Server.MapPath(connectionInput));

            FillTableWithConnections(connectionList,
                                     ConnectionTable);

            InOutUtils.AppendInitData(connectionList,
                                      Server.MapPath(initialDataPath));


            FillPathTable(students, connectionList, PathTable);
            InOutUtils.CreateFile(Server.MapPath(outputDataPath));

            InOutUtils.AppendConnectionResults(students,
                                               connectionList,
                                               Server.MapPath(outputDataPath));

        }

        /// <summary>
        /// Used to show initial Student Data
        /// </summary>
        /// <param name="students">List Student data type</param>
        /// <param name="table">Table Object data type</param>
        protected void FillTableWithStudents(List<Student> students, Table table)
        {
            TableRow row = new TableRow();
            row.Cells.Add(CreateCell("Studentas"));
            row.Cells.Add(CreateCell("Draugų Kiekis"));
            row.Cells.Add(CreateCell("Studentų Draugai:"));
            table.Rows.Add(row);

            foreach (Student student in students)
            {
                row = new TableRow();
                row.Cells.Add(CreateCell(student.Name));
                row.Cells.Add(CreateCell(student.FriendCount.ToString()));
                row.Cells.Add(CreateCell(student.GetFriendsString()));
                table.Rows.Add(row);

            }
        }

        /// <summary>
        /// Used to show initial connection data
        /// </summary>
        /// <param name="connections">List of Tuples compromised of string, string
containing the initial node and end node to use for DFS</param>
        /// <param name="table">Table object data type</param>
        protected void FillTableWithConnections(List<Tuple<string,
                                                 string>> connections,
                                     Table table)
        {
            TableRow row = new TableRow();
```

```csharp
            row.Cells.Add(CreateCell("Draugas"));
            row.Cells.Add(CreateCell("Ieškomas Draugas"));
            table.Rows.Add(row);

            foreach (Tuple<string, string> connection in connections)
            {
                row = new TableRow();
                row.Cells.Add(CreateCell(connection.Item1));
                row.Cells.Add(CreateCell(connection.Item2));
                table.Rows.Add(row);

            }
        }

        /// <summary>
        /// Creates A cell with provided Text
        /// </summary>
        /// <param name="text">text to be added to the Cell.text param</param>
        /// <returns>TableCell object</returns>
        protected TableCell CreateCell(string text)
        {
            TableCell cell = new TableCell();
            cell.Style.Add("padding", "5px");
            cell.Text = text;
            return cell;
        }

        /// <summary>
        /// Fills the table with paths from student a to b
        /// </summary>
        /// <param name="students"> Dictionary, key -> string of the student, value ->
student object</param>
        /// <param name="connections">List of Tuples compromised of string, string
containing the initial node and end node to use for DFS</param>
        /// <param name="table">Table object where the data will be added</param>
        protected void FillPathTable(Dictionary<string, Student> students,
                                     List<Tuple<string, string>> connections,
                                     Table table)
        {
            TableRow row = new TableRow();
            row.Cells.Add(CreateCell("Draugas"));
            row.Cells.Add(CreateCell("Ieškomas Draugas"));
            row.Cells.Add(CreateCell("Kelias: "));
            table.Rows.Add(row);

            foreach (Tuple<string, string> connection in connections)
            {
                List<string> path = new List<string>();
                path.Add(connection.Item1);
                path = TaskUtils.FindConnection(connection.Item1,
                                                connection.Item2,
                                                path, students);

                string pathText = TaskUtils.CreatePathText(path);

                row = new TableRow();
                row.Cells.Add(CreateCell(connection.Item1));
                row.Cells.Add(CreateCell(connection.Item2));
                row.Cells.Add(CreateCell(pathText));
                table.Rows.Add(row);

            }
        }
    }
}
```

1.7. Pradiniai duomenys ir rezultatai

```
Pradiniai Duomenys 1:

students.txt:

Rūta 1 Arnoldas
Agnė 3 Nerijus Neda Antanas
Nerijus 1 Agnė
Antanas 2 Agnė Marius
Marius 2 Antanas Neda
Neda 3 Marius Rūta Agnė
Arnoldas 1 Rūta

connections.txt:

Rūta Nerijus
Agnė Antanas
Neda Nerijus

Rezultatai 1:
```

Vartotojo sąsaja:

Lab01-16

Studentų duomenys:

Studentas	Draugų Kiekis	Studentų Draugai:
Rūta	1	Arnoldas
Agnė	3	Nerijus Neda Antanas
Nerijus	1	Agnė
Antanas	2	Agnė Marius
Marius	2	Antanas Neda
Neda	3	Marius Rūta Agnė
Arnoldas	1	Rūta

Studentų Ieškomi Junginiai:

Draugas	Ieškomas Draugas
Rūta	Nerijus
Agnė	Antanas
Neda	Nerijus

Rezultatai:

Draugas	Ieškomas Draugas	Kelias:
Rūta	Nerijus	negali susipažinti
Agnė	Antanas	jau pažįstami
Neda	Nerijus	bendri pažįstami: Agnė

initial_data.txt:

```
Studentai ir jų draugai
Studentas               |Draugų kiekis      |Draugai:
Rūta                    |                  1|Arnoldas
Agnė                    |                  3|Nerijus Neda Antanas
Nerijus                 |                  1|Agnė
Antanas                 |                  2|Agnė Marius
Marius                  |                  2|Antanas Neda
Neda                    |                  3|Marius Rūta Agnė
Arnoldas                |                  1|Rūta

Studentai ir jų ieškomi draugai:
Studentas           Ieškomas draugas
Rūta                Nerijus
Agnė                Antanas
Neda                Nerijus
```

Result.txt:

Draugai ir jų junginiai, bei keliai:

Draugas	Ieškomas draugas:	Kelias:
Rūta	Nerijus	negali susipažinti
Agnė	Antanas	jau pažįstami
Neda	Nerijus	bendri pažįstami: Agnė

Pradiniai Duomenys 2:

students.txt:

```
a 2 g b
b 2 a c
c 2 b f
d 1 e
e 1 d
f 2 h c
g 2 a h
h 2 g f
```

connections.txt:

```
a f
a b
a e
```

```
Rezultatai 2:

Vartotojo Sąsaja:
```

Lab01-16

Studentų duomenys:

Studentas	Draugų Kiekis	Studentų Draugai:
a	2	g b
b	2	a c
c	2	b f
d	1	e
e	1	d
f	2	h c
g	2	a h
h	2	g f

Studentų Ieškomi Junginiai:

Draugas	Ieškomas Draugas
a	f
a	b
a	e

Rezultatai:

Draugas	Ieškomas Draugas	Kelias:
a	f	bendri pažįstami: g h
a	b	jau pažįstami
a	e	negali susipažinti

```
Initial_data.txt:

Studentai ir jų draugai
Studentas               |Draugų kiekis         |Draugai:
a                       |                     2|g b
b                       |                     2|a c
c                       |                     2|b f
d                       |                     1|e
e                       |                     1|d
f                       |                     2|h c
g                       |                     2|a h
h                       |                     2|g f


Studentai ir jų ieškomi draugai:
Studentas               Ieškomas draugas
a                       f
a                       b
a                       e

result.txt:

Draugai ir jų junginiai, bei keliai:
Draugas                 |Ieškomas draugas:   |Kelias:
a                       |f                    |bendri pažįstami: g h
a                       |b                    |jau pažįstami
a                       |e                    |negali susipažinti
```

1.8. Dėstytojo pastabos

1. Reiktų šiek tiek pakeisti ataskaitos įvardinimą. Jūsų grupė nėra IFF14.

2. Klasių diagramai vien tik Visual Studio įrankio neužtenka. Jis ne neatskleidžia pilnai klasės vidaus.

3. Garmatinės klaidos "su duotą ir apskaičiuotą "

4. • Įvedimo ir išvedimo metodus, veikiančius su tekstiniu failu, talpinkite į public static class InOutUtils.

5. Parametrus reikia komentuoti visiems metodams // /// appends students to TXT file /// public static void AppendInitialStudentData(List students, string path)

6. Čia tik rodyklės perrašymas:

public Student(string name, List friends)

{

Name = name;

Friends = friends;

2. Dinaminis atminties valdymas (L2)

2.1. Darbo užduotis

LD_16. **Mokesčiai.** Kiekvieną mėnesį gyventojai moka komunalinius mokesčius. Suraskite, kurį mėnesį ir kokie komunaliniai mokesčiai kainavo pigiausiai. Apskaičiuokite, kokią pinigų sumą komunaliniams mokesčiams išleido visi gyventojai. Sudarykite sąrašą gyventojų (pavardė ir vardas, adresas), kurie už komunalines paslaugas per metus mokėjo sumą, mažesnę už vidutinę. Sąrašas turi būti surikiuotas pagal gyventojų adresus, pavardes ir vardus abėcėlės tvarka.

Duomenys:

- tekstiniame faile U16a.txt yra informacija apie komunalines paslaugas: paslaugos kodas, paslaugos pavadinimas, paslaugos vieno mėnesio vieno vieneto kaina;
- tekstiniame faile U16b.txt yra informacija apie gyventojus: pavardė ir vardas, adresas, mėnuo už kurį mokama, komunalinės paslaugos kodas, sunaudotų per mėnesį vienetų kiekis.

Pašalinkite iš sąrašo gyventojus, kurie nemokėjo už nurodytą paslaugą, nurodytą mėnesį (duomenys įvedami klaviatūra).

2.2. Grafinės vartotojo sąsajos schema

2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
HeaderLabel	Text	LAB02 U16
Label1	Text	Tax InfoU16a.txt:
Label2	Text	Every Citizen Tax Data U16b:
InitTaxLabel	Text	U16a.txt Initial data:
InitCitizenLabel	Text	U16b.txt Initial data:
CitizenTaxLabel	Text	All Citizen taxes over the months
AverageTax	Text	""
TotalTaxSum	Text	""
CitizenTaxLabel0	Text	Above Average Tax:
FilterData	Text	Filtered data:
ButtonFilter	Text	Tax Code:
DataButton	Text	Month:

2.4. Klasių diagrama

Lab01Form.aspx

Lab01.Form.aspx
- taxDataInput: string
- citizenDataInput: string
- outputDataPath: string

- CitizenCalculations(in taxInfo: Tax, in citizenTaxData: CitizenTax)
- CheckFiltered(in TaxInfo: Tax, CitizenTaxData: CitizenTax)
- CreateCell(in text: string): TableCell
- GetRow(in data): TableRow

Lab01Form.aspx.designer.cs

css/styles.css

TaskUtils.cs
+ CreateCitizenData(in TaxList: Tax, in citizenTaxList: CitizenTax)

CitizenTax
- head: Node
- tail: Node
- d: Node
+ CitizenTax()
+ CitizenPayed(in taxCode: string, in month: string, in lastName: string, in firstName: string)
+ Add(in data: CitizenTaxData)

Tax.cs
- head: Node
- tail: Node
- d: Node
public Tax()
+ Begin(): {query}
+ Next(): {query}
+ Exist(): {query}
+ Get(): TaxData
+ GetPrice(): double
+ Add()

Node
- Data: TaxData
- Node: next

TaxData
+ TaxCode: string
+ TaxName: string
+ Price: double

CitizenTax
- head: Node
- tail: Node
- d: Node
+ Citizen()
+ Begin(): {query}
+ Next(): {query}
+ Exist(): {query}
+ Get(): CitizenData
+ AddMoney(in lastName: string, in firstName: string, in string: address, in double: taxSum)
+ AddMoney(in lastName: string, in firstName: string, in string: address, in double: taxSum)
+ RemoveUnderAverage()
+ Sum(): double
+ Average(): double
+ RemoveWhoDidNotPayTax(in taxCode: string, in month: string, in data: CitizenTax)
+ Sort()

Node
- next: Node
- data: CitizenData
- SwapData(in other: Node)

CitizenData
+ FirstName: string
+ LastName: string
+ Address: string
+ TaxSum: double

Node
- Node next
- Data: CitizenTaxData

CitizenTax
+ FirstName: string
+ LastName: string
+ Address: string
+ TaxCode: string
+ TaxAmount: int
+ Month: string

InOutUtils.cs
+ ReadTaxData(in fileLoc: string): Tax
+ ReadCitizenTaxData(in fileLoc: string): CitizenTax
+ WriteHeader(fileLoc: string, in header: string)
public static void CreateFile(string fileLoc)
+ CreateFile(in fileLoc: string)
+ WriteCitizenTaxData(in fileLoc: string, data: CitizenTax, in header: string)
+ WriteTaxData(in string fileLoc, in data: Tax, in header: string)

2.5. Programos vartotojo vadovas

Jeigu neranda failų visų duombazėje, programa paprašo failų. Jeigu randa tik vieną pradinį failą, rodo tik jį ir prašo likusių failų. Kai abu failai atsiranda duombazėje, užkrauna skaičiavimus. Apskaičiuoja vidutinę mokesčių kainą, sumą visų ir individualių žmonių. Tekstas yra rikiuojamas A-Z pagal: adresą, pavardę, vardą. Kodas leidžia filtruoti žmones, kurie mokėjo nurodytą mėnesį (mėnuo yra string) už nurodytus mokesčius naudojant „Tax Code" (string). Prie filtered lentelės prideda tik filtruotus duomenis.

2.6. Programos tekstas

CitizenData.cs:

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI.WebControls;

namespace Lab02
{
    /// <summary>
    /// TaskUtils static class for helper functions
    /// </summary>
    public static class TaskUtils
    {

        /// <summary>
        /// Creates Citizen class object using Tax object
        /// </summary>
        /// <param name="TaxList">Tax class object</param>
        /// <param name="citizenTaxList">CitizenTax object</param>
        /// <returns>Citizen class object</returns>
        public static Citizen CreateCitizenData(Tax TaxList, CitizenTax citizenTaxList)
        {
            Citizen citizens = new Citizen();
            for (citizenTaxList.Begin(); citizenTaxList.Exist(); citizenTaxList.Next())
            {
                CitizenTaxData citizenTaxData = citizenTaxList.Get();
                for (TaxList.Begin(); TaxList.Exist(); TaxList.Next())
                {
                    TaxData taxData = TaxList.Get();
                    if(citizenTaxData.TaxCode == taxData.TaxCode)
                    {
                        citizens.AddMoney(citizenTaxData.LastName,
citizenTaxData.FirstName, citizenTaxData.Address, (double)taxData.Price *
citizenTaxData.TaxAmount);

                    }
                }
            }

            return citizens;
        }


    }
}

Citizen.cs:

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI.WebControls;

namespace Lab02
{
    /// <summary>
    /// Citizen class object
    /// </summary>
    public class Citizen
    {
        private Node head;
        private Node tail;
        private Node d;

        /// <summary>
        /// Construcotr
```

```csharp
        /// </summary>
        /// <param name="head"></param>
        /// <param name="tail"></param>
        public Citizen()
        {
            head = null;
            tail = null;
        }


        /** Address of the head of the list is assigned */
        public void Begin()
        { d = head; }
        /** Interface variable gets address of the next entry*/
        public void Next()
        { d = d.next; }
        /** Return true, if list is empty*/
        public bool Exist()
        { return d != null; }
        //-----------------------------------------------
        /** Return data according to the interface address*/
        public CitizenData Get()
        { return d.Data; }


        /// <summary>
        /// Returns Citizen with keys
        /// </summary>
        /// <param name="lastName">Last Name of Citizen</param>
        /// <param name="firstName">First Name of Citizen</param>
        /// <param name="address">Address of citizen</param>
        /// <returns></returns>
        public CitizenData Get(string lastName, string firstName, string address)
        {
            // If Citizen exists, adds sum to his current balance
            for (Begin(); Exist(); Next())
            {
                CitizenData curr = Get();
                if (curr.LastName == lastName && curr.FirstName == firstName && curr.Address
== address)
                {
                    return curr;
                }
            }
            return null;
        }

        /// <summary>
        /// Adds CitizenData to Citizen Linked List
        /// </summary>
        /// <param name="data">CitizenData object</param>
        internal void Add(CitizenData data)
        {
            // If No citizen was found, adds the citizen to Linked List
            if (head == null)
            {
                head = new Node(data, null);
                tail = head;
            }
            else
            {
                tail.next = new Node(data, null);
                tail = tail.next;
            }
        }
```

```csharp
/// <summary>
/// Removes citiznens from linked list who payed belove average taxes
/// </summary>
public void RemoveUnderAverage()
{
    if (head == null)
        return;

    Node prev = head;
    Node curr = head.next;
    double average = GetAverage();

    while(curr != null)
    {
        if(curr.Data.TaxSum < average)
        {
            prev.next = curr.next;
            curr = curr.next;
        }
        else
        {
            curr = curr.next;
            prev = prev.next;
        }
    }

    RemoveUnderAverageHead(average);
    ResetTail();
}

/// <summary>
/// Checks if head/start of linked list is below average. If true removes
/// </summary>
/// <param name="average">Average tax sum of a citizen</param>
private void RemoveUnderAverageHead(double average)
{
    Node curr = head;
    while(curr.Data.TaxSum < average)
    {
        curr = curr.next;
    }
    head = curr;
}

/// <summary>
/// Resets tail after removing elements
/// </summary>
private void ResetTail()
{
    Node curr = head;
    if (curr == null)
    {
        tail = null;
        return;
    }

    while(curr.next != null)
    {
        curr = curr.next;
    }
    tail = curr;
}

/// <summary>
/// Returns the total amount citizens payed for taxes
```

```csharp
/// </summary>
/// <returns></returns>
public double Sum()
{
    Node curr = head;
    double sum = 0;
    while (curr != null)
    {
        sum += curr.Data.TaxSum;
        curr = curr.next;
    }
    return sum;
}

public double GetAverage()
{
    Node curr = head;
    double sum = 0;
    int i = 0;
    while (curr != null)
    {
        sum += curr.Data.TaxSum;
        i++;
        curr = curr.next;
    }

    if (i == 0)
        return 0;
    else
        return (double)sum / i;
}

/// <summary>
/// Removes citizens who did not pay taxes specified month
/// </summary>
/// <param name="taxCode"> Tax Code of the tax</param>
/// <param name="month">Specified Month </param>
/// <param name="data">CitizenTaxData to see what citizen payed what tax at the
specified month</param>
public void RemoveWhoDidNotPayTax(string taxCode, string month, CitizenTax data)
{
    {
        if (head == null)
            return;

        Node prev = head;
        Node curr = head.next;

        while (curr != null)
        {
            // Checks if the citizen has payed Taxes in CitizenTaxData on specified
Month
            if (curr != null && data.CitizenPayed(taxCode, month, curr.Data.LastName,
curr.Data.FirstName) == false)
            {
                prev.next = curr.next;
                curr = curr.next;
            }
            else
            {
                curr = curr.next;
                prev = prev.next;
            }
        }

        RemoveWhoDidNotPayTaxHead(taxCode, month, data);
```

```csharp
                ResetTail();
            }
        }

        /// <summary>
        /// Checks first/start/head element of the linked list if the tax was paid
        /// </summary>
        /// <param name="taxCode">Tax code of the specified tax</param>
        /// <param name="month">specified month to check</param>
        /// <param name="data">CitizenTaxData to check if the first element of the linked list
payed for taxes</param>
        private void RemoveWhoDidNotPayTaxHead(string taxCode, string month, CitizenTax data)
        {
            Node curr = head;
            // Checks if the citizen has payed Taxes in CitizenTaxData on specified Month
            while (curr != null && data.CitizenPayed(taxCode, month, curr.Data.LastName,
curr.Data.FirstName) == false)
            {
                curr = curr.next;
            }
            head = curr;
        }


        /// <summary>
        /// Sorts LinkedList A-Z using keys: address, last name, first name. Does data swap
instead of pointers.
        /// </summary>
        public void Sort()
        {
            Node timer = head;
            while(timer != null)
            {
                Node curr = head;
                Node next = head.next;
                while(next != null)
                {
                    if (curr.Data.CompareTo(next.Data) > 0)
                    {
                        curr.SwapData(next);
                    }
                    curr = next;
                    next = next.next;
                }
                timer = timer.next;
            }
        }


        /// <summary>
        /// Node class to be used to save every citizen seperately
        /// </summary>
        class Node
        {
            public CitizenData Data { get; set; }
            public Node next { get; set; }

            /// <summary>
            /// Constructor
            /// </summary>
            /// <param name="data">CitizenData pointer</param>
            public Node(CitizenData data, Node _next)
            {
                Data = data;
                next = _next;
            }
```

```
            /// <summary>
            /// Swaps the DATA, keeps the pointers
            /// </summary>
            /// <param name="other">Other node to be swapped with</param>
            public void SwapData(Node other)
            {
                CitizenData temp = Data;
                Data = other.Data;
                other.Data = temp;
            }
        }
    }
}
```

CitizenTaxData.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab02
{
    public class CitizenTaxData
    {

        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Address { get; set; }
        public string TaxCode { get; set; }
        public int TaxAmount { get; set; }
        public string Month { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="lastName">last name of citizen</param>
        /// <param name="firstName">first name of citizen</param>
        /// <param name="address">address of the citizen</param>
        /// <param name="month">the month the tax was paid</param>
        /// <param name="taxCode">tax code</param>
        /// <param name="taxAmount">tax amount</param>
        public CitizenTaxData(string lastName, string firstName, string address, string
month, string taxCode, int taxAmount)
        {
            FirstName = firstName;
            LastName = lastName;
            Address = address;
            TaxCode = taxCode;
            TaxAmount = taxAmount;
            Month = month;
        }

        public override string ToString()
        {
            return $"{LastName,-20} {FirstName,-20}|{Address,-20}|{Month,-15}|{TaxCode,-
20}|{TaxAmount,10}|";
        }
    }
}
```

CitizenTax.cs:

```
using System;
```

```csharp
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI.WebControls;

namespace Lab02
{
    /// <summary>
    /// Citizen class object meant to store name and how much the individual payed for taxe
    /// </summary>
    public class CitizenTax
    {
        private Node head;
        private Node tail;
        private Node d;

        /// <summary>
        /// Constructor
        /// </summary>
        public CitizenTax()
        {
            head = null;
            tail = null;
        }
        /// <summary>
        /// Adds element to Linked List
        /// </summary>
        /// <param name="lastName">Last Name</param>
        /// <param name="firstName">First Name</param>
        /// <param name="address">Address</param>
        /// <param name="month">Month</param>
        /// <param name="taxCode">Tax Code</param>
        /// <param name="taxAmount">Tax Amount</param>
        public void Add(CitizenTaxData data)
        {
            if (head == null)
            {
                head = new Node(data, null);
                tail = head;

            }
            else
            {
                tail.next = new Node(data, null);
                tail = tail.next;
            }

        }

        /** Address of the head of the list is assigned */
        public void Begin()
        { d = head; }
        /** Interface variable gets address of the next entry*/
        public void Next()
        { d = d.next; }
        /** Return true, if list is empty*/
        public bool Exist()
        { return d != null; }
        //----------------------------------------------
        /** Return data according to the interface address*/
        public CitizenTaxData Get()
        { return d.Data; }

        /// <summary>
        /// Checks of the specified citizen has payed
        /// </summary>
```

```csharp
        /// <param name="taxCode">Tax Code of the Tax Company</param>
        /// <param name="month">Month</param>
        /// <param name="lastName">Last name of the citizen</param>
        /// <param name="firstName"> First Name of the citizen</param>
        /// <returns>true if citizen has payed for specified tax on specified month, false if
the citizen did not</returns>
        public bool CitizenPayed(string taxCode, string month, string lastName, string
firstName)
        {
            Node curr = head;
            while (curr != null)
            {
                if (curr.Data.LastName == lastName && curr.Data.FirstName == firstName &&
curr.Data.Month == month && curr.Data.TaxCode == taxCode)
                    return true; // The Person paid for the month

                curr = curr.next;
            }
            return false;
        }

        /// <summary>
        /// Node class object for CitizenTaxData
        /// </summary>
        class Node
        {
            public Node next;
            public CitizenTaxData Data { get; set; }

            /// <summary>
            /// Constructor
            /// </summary>
            /// <param name="data">Pointer to CitizenTaxData object</param>
            public Node(CitizenTaxData data, Node _next)
            {
                Data = data;
                next = _next;
            }
        }
    }
}

TaxData.cs:

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab02
{
    /// <summary>
    /// TaxData object to be inherited by Tax object
    /// </summary>
    public class TaxData
    {
        public string TaxCode { get; set; }
        public string TaxName { get; set; }
        public double Price { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="taxCode"></param>
        /// <param name="taxName"></param>
        /// <param name="price"></param>
        public TaxData(string taxCode, string taxName, double price)
```

```
        {
            TaxCode = taxCode;
            TaxName = taxName;
            Price = price;
        }

        /// <summary>
        /// Returns Node in string format
        /// </summary>
        /// <returns>Node in string format</returns>
        public override string ToString()
        {
            return $"{TaxCode,-20}|{TaxName,-20}|{Price,10:f}|";
        }
    }
}
```

Tax.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI.WebControls;

namespace Lab02
{
    public class Tax
    {
        private Node head;
        private Node tail;
        private Node d;
        public Tax()
        {
            head = null;
            tail = null;
        }

        /** Address of the head of the list is assigned */
        public void Begin()
        { d = head; }
        /** Interface variable gets address of the next entry*/
        public void Next()
        { d = d.next; }
        /** Return true, if list is empty*/
        public bool Exist()
        { return d != null; }
        //-----------------------------------------------
        /** Return data according to the interface address*/
        public TaxData Get()
        { return d.Data; }

        /// <summary>
        /// Returns the price of the tax of a single use
        /// </summary>
        /// <param name="taxCode">Code to identify the type of tax</param>
        /// <returns>Double, price of a single use tax item</returns>
        public double GetPrice(string taxCode)
        {
            Node curr = head;
            while (curr != null)
            {
                if (curr.Data.TaxCode == taxCode)
                    return curr.Data.Price;
                curr = curr.next;
            }
            return 0;
```

```csharp
        }

        /// <summary>
        /// Adds Node to the tail of the LinkedList
        /// </summary>
        /// <param name="taxCode">Code of the tax</param>
        /// <param name="name"> name of the company</param>
        /// <param name="price">price of a single use</param>
        public void Add(TaxData data)
        {
            if (head == null)
            {
                head = new Node(data, null);
                tail = head;
            }
            else
            {
                tail.next = new Node(data, null);
                tail = tail.next;
            }

        }

        /// <summary>
        /// Tax Node
        /// </summary>
        class Node
        {
            public Node next;
            public TaxData Data { get; set; }
            /// <summary>
            /// Constructor
            /// </summary>
            /// <param name="data">TaxData pointer</param>
            public Node(TaxData data, Node _next)
            {
                Data = data;
                next = _next;
            }

        }

    }
}

InOutUtils.cs:

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;

namespace Lab02
{
    /// <summary>
    /// Static InOutUtils helper class for Input/Output with files
    /// </summary>
    public static class InOutUtils
    {
        /// <summary>
        /// Reads Tax Data from txt to Tax class object+
        /// </summary>
        /// <param name="fileLoc">Location of the data in .txt format</param>
        /// <returns>Tax class object</returns>
        public static Tax ReadTaxData(string fileLoc)
        {
            Tax taxes = new Tax();
```

31

```csharp
            string[] lines = File.ReadAllLines(fileLoc);
            foreach (string line in lines)
            {
                string[] elements = line.Split(';');
                taxes.Add(new TaxData(elements[0], elements[1],
double.Parse(elements[2])));
            }
            return taxes;
        }

        /// <summary>
        /// Creates CitizenTaxData from .txt file
        /// </summary>
        /// <param name="fileLoc">Location of .txt file</param>
        /// <returns>CitizenTaxData class object</returns>
        public static CitizenTax ReadCitizenTaxData(string fileLoc)
        {
            CitizenTax data = new CitizenTax();
            string[] lines = File.ReadAllLines(fileLoc);
            foreach (string line in lines)
            {
                string[] elements = line.Split(';');
                CitizenTaxData temp = new CitizenTaxData(elements[1], elements[0],
elements[2], elements[3], elements[4], int.Parse(elements[5]));
                data.Add(temp);
            }
            return data;
        }

        /// <summary>
        /// Appends a header to a file
        /// </summary>
        /// <param name="fileLoc">Name/location of the file</param>
        /// <param name="header">text to be appended</param>
        public static void WriteHeader(string fileLoc, string header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append: true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
            }
        }

        /// <summary>
        /// Creates a new or wipes a file
        /// </summary>
        /// <param name="fileLoc">Location of the file</param>
        public static void CreateFile(string fileLoc)
        {
            using (FileStream fs = new FileStream(fileLoc, FileMode.Create))
                new StreamWriter(fs, encoding: System.Text.Encoding.UTF8).Close();
        }

        /// <summary>
        /// Appends CitizenTaxData to a file
        /// </summary>
        /// <param name="fileLoc">Location/name of the file</param>
        /// <param name="data">data to append to the .txt file</param>
        /// <param name="header">Header text of the data file</param>
        public static void WriteCitizenTaxData(string fileLoc, CitizenTax data, string
header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append:true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
```

```csharp
            writer.WriteLine($"{"LastName",-20} {"FirstName",-20}|{"Address",-
20}|{"Month",-15}|{"TaxCode",-20}|{"TaxAmount",10}|");
                for (data.Begin(); data.Exist(); data.Next())
                {
                    CitizenTaxData temp = data.Get();
                    writer.WriteLine(temp.ToString());
                }
                writer.WriteLine();
            }
        }

        /// <summary>
        /// appends Citizen class object data to text file
        /// </summary>
        /// <param name="fileLoc">location/name of the file</param>
        /// <param name="data">data to append to the file</param>
        /// <param name="header">Header of the file</param>
        public static void WriteCitizenData(string fileLoc, Citizen data, string header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append: true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
                writer.WriteLine($"{"LastName",-20} {"FirstName",-20}|{"Address",-
20}|{"TaxSum",-10}|");
                for (data.Begin(); data.Exist(); data.Next())
                {
                    CitizenData temp = data.Get();
                    writer.WriteLine(temp.ToString());
                }
                writer.WriteLine();
            }
        }

        /// <summary>
        /// Appends Tax data to a .txt file
        /// </summary>
        /// <param name="fileLoc">Location/name of the file</param>
        /// <param name="data">data to append to the .txt file</param>
        /// <param name="header">header to be added to the file</param>
        public static void WriteTaxData(string fileLoc, Tax data, string header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append: true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
                writer.WriteLine($"{"TaxCode",-20}|{"TaxName",-20}|{"Price",10:2f}|");
                for (data.Begin(); data.Exist(); data.Next())
                {
                    TaxData temp = data.Get();
                    writer.WriteLine(temp.ToString());
                }
                writer.WriteLine();
            }
        }
    }
}

css/styles.css:

body {
    color:white;
    background:black;
}
td
{
    padding:5px;
```

```
}

Lab01Form.aspx:

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Lab01Form.aspx.cs"
Inherits="Lab02.Lab01Form" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <link rel="stylesheet" runat="server" media="screen" href="~/css/styles.css" />
    <title>Lab02 U16</title>
</head>
<body>
    <form id="form1" runat="server">
        <div id="body">
            <asp:Label ID="HeaderLabel" runat="server" Text="LAB02 U16"></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Tax Info U16a.txt:"></asp:Label>
            <br />
            <asp:FileUpload ID="FileUpload1" runat="server" />
            <br />
            <br />
            <asp:Label ID="Label2" runat="server" Text="Every Citizen Tax Data U16b.txt:
"></asp:Label>
            <br />
            <asp:FileUpload ID="FileUpload2" runat="server" />
            <br />
            <asp:Button ID="DataButton" runat="server" Text="Submit New Data"
OnClick="DataButton_Click" />
            <br />
            <br />
            <asp:Label ID="InitTaxLabel" runat="server" Text="U16a.txt Initial
data:"></asp:Label>
            <asp:Table ID="InitTaxTable" runat="server">
            </asp:Table>
            <br />
            <asp:Label ID="InitCitizenLabel" runat="server" Text="U16b.txt Initial
data:"></asp:Label>
            <asp:Table ID="InitCitizenTable" runat="server">
            </asp:Table>
            <br />
            <asp:Panel ID="CalculationsPanel" runat="server">
                <asp:Label ID="CitizenTaxLabel" runat="server" Text="All Citizen taxes
over the months"></asp:Label>
                <asp:Table ID="CitizenTaxTable" runat="server">
                </asp:Table>
                <br />
                <asp:Label ID="AverageTax" runat="server"></asp:Label>
                <br />
                <asp:Label ID="TotalTaxSum" runat="server"></asp:Label>
                <br />
                <br />
                <asp:Label ID="CitizenTaxLabel0" runat="server" Text="Above Average
Tax:"></asp:Label>
                <asp:Table ID="AboveAverageTable" runat="server">
                </asp:Table>
                <br />
                <asp:Label ID="FilterData" runat="server" Text="Filtered
data:"></asp:Label>
                <asp:Table ID="FilterTable" runat="server">
                </asp:Table>
                <br />
                Tax Code:<br />
                <asp:TextBox ID="TaxCodeTextBox" runat="server"></asp:TextBox>
```

```html
                <br />
                Month:<br />
                <asp:TextBox ID="TaxMonthTextBox" runat="server"></asp:TextBox>
                <br />
                <asp:Button ID="ButtonFilter" runat="server" Text="Submit"
OnClick="ButtonFilter_Click" />
            </asp:Panel>
            <br />
        </div>
    </form>
</body>
</html>
```

Lab01Form.aspx.cs:

```csharp
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Lab02
{
    public partial class Lab01Form : System.Web.UI.Page
    {
        private string taxDataInput = @"App_Data/U16a.txt";
        private string citizenDataInput = @"App_Data/U16b.txt";
        private string outputDataPath = @"App_Data/U16result.txt";
        protected void Page_Load(object sender, EventArgs e)
        {
            CitizenTax citizenTaxData = null;
            Tax taxInfo = null;
            InOutUtils.CreateFile(Server.MapPath(outputDataPath));
            if (File.Exists(Server.MapPath(taxDataInput)))
            {
                taxInfo = InOutUtils.ReadTaxData(Server.MapPath(taxDataInput));
                InOutUtils.WriteTaxData(Server.MapPath(outputDataPath), taxInfo, "Initial
Tax Company Data:");
                FillTaxDataTable(taxInfo, InitTaxTable);
            }
            else
            {
                InitTaxLabel.Text = "";
            }

            if (File.Exists(Server.MapPath(citizenDataInput)))
            {
                citizenTaxData =
InOutUtils.ReadCitizenTaxData(Server.MapPath(citizenDataInput));
                InOutUtils.WriteCitizenTaxData(Server.MapPath(outputDataPath),
citizenTaxData, "Initial Citizen Tax Data:");
                FillCitizenTaxDataTable(citizenTaxData, InitCitizenTable);
            }
            else
            {
                InitCitizenLabel.Text = "";
            }

            if (citizenTaxData != null && taxInfo != null)
            {
                // Reads Initial Data and Outputs the Initial Data To WebForm and to text

                CitizenCalculations(taxInfo, citizenTaxData);
```

```csharp
                    CheckFiltered(taxInfo, citizenTaxData);
                }
                else
                {
                    HeaderLabel.Text = "Plaese Upload remaining data files";
                    CalculationsPanel.Visible = false;
                }
            }

            /// <summary>
            /// Does calculations from Tax and CitizenTax object
            /// </summary>
            /// <param name="taxInfo">Tax object</param>
            /// <param name="citizenTaxData">CitizenTax object</param>
            protected void CitizenCalculations(Tax taxInfo, CitizenTax citizenTaxData)
            {
                Citizen citizensAverage = TaskUtils.CreateCitizenData(taxInfo,
citizenTaxData); // For Above Average
                InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath), citizensAverage,
"Tax Sum of all citizens:");

                citizensAverage.Sort();
                InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath), citizensAverage,
"Tax Sum of all citizens SORTED A-Z:");
                FillCitizenTable(citizensAverage, CitizenTaxTable);

                double sum = citizensAverage.Sum();
                double average = citizensAverage.GetAverage();
                InOutUtils.WriteHeader(Server.MapPath(outputDataPath), $"All Citizen TOTAL
Tax Sum: {sum:f}");
                InOutUtils.WriteHeader(Server.MapPath(outputDataPath), $"Average Tax Sum:
{average:f}");
                AverageTax.Text = $"Average tax per citizen: {average}";
                TotalTaxSum.Text = $"Total tax sum: {sum}";

                citizensAverage.RemoveUnderAverage();
                InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath), citizensAverage,
"Citizens who paid above average:");
                FillCitizenTable(citizensAverage, AboveAverageTable);
            }
            /// <summary>
            /// Updates filtered data
            /// </summary>
            /// <param name="taxInfo">Tax Object</param>
            /// <param name="citizenTaxData">CitizenTax object</param>
            protected void CheckFiltered(Tax taxInfo, CitizenTax citizenTaxData)
            {
                if (Session["TaxCode"] != null && Session["Month"] != null)
                {
                    Citizen citizensFiltered = TaskUtils.CreateCitizenData(taxInfo,
citizenTaxData); // For Filter
                    citizensFiltered.Sort();
                    citizensFiltered.RemoveWhoDidNotPayTax(Session["TaxCode"].ToString(),
Session["Month"].ToString(), citizenTaxData);
                    InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath),
citizensFiltered, $"Citizens who paid TaxCode: \"{Session["TaxCode"]}\" on Month:
\"{Session["Month"]}\"");
                    FillCitizenTable(citizensFiltered, FilterTable);
                }
                else
                {
                    FilterData.Text = "No Filter provided";
                }

                Session["TaxCode"] = null;
                Session["Month"] = null;
            }
```

2.7. Pradiniai duomenys ir rezultatai

```
U16a.txt:

11; Elektra; 0.12
21; Dujos; 0.58
31; Benzinas; 1.78
32; Diezelis; 1.90

U16b.txt:

pavardė1; vardas1; adresas1;1;22;28;
pavardė1; vardas1; adresas1;5;22;20;
pavardė1; vardas1; adresas1;1;32;100;
pavardė1; vardas1; adresas1;2;32;97;
pavardė1; vardas1; adresas1;3;32;63;
pavardė1; vardas1; adresas1;2;22;25;
pavardė1; vardas1; adresas1;3;22;29;
pavardėAA; vardasAA; adresasAA;1;21;13;
pavardėAA; vardasAA; adresasAA;2;21;84;
pavardėAA; vardasAA; adresasAA;3;21;76;
pavardė1; vardas1; adresas1;4;22;39;
pavardė2; vardas2; adresas2;3;31;67;
pavardė2; vardas2; adresas2;4;31;98;
pavardė0; vardas2; adresas2;5;31;125;
pavardė0; vardas0; adresas3;1;11;31;
pavardė1; vardas1; adresas1;4;32;39;
pavardė1; vardas1; adresas1;5;32;20;
pavardė1; vardas1; adresas1;3;11;80;
pavardė1; vardas1; adresas1;4;11;39;
pavardė1; vardas1; adresas1;1;11;120;
pavardė1; vardas1; adresas1;2;11;100;
pavardė1; vardas1; adresas1;5;11;139;
pavardė2; vardas2; adresas2;1;31;31;
pavardė2; vardas2; adresas2;2;31;48;
pavardė0; vardas0; adresas3;2;11;48;
pavardė0; vardas0; adresas3;3;11;67;
pavardė0; vardas0; adresas3;4;11;98;
pavardė0; vardas0; adresas3;5;11;125;
pavardėAA; vardasAA; adresasAA;4;21;8;
pavardėAA; vardasAA; adresasAA;5;21;25;

Rezultatai 1:

Vartotojo sąsaja:
```

Duombazėje nerado jokio failo:

Plaese Upload remaining data files

Tax Info U16a.txt:
[Choose File] No file chosen

Every Citizen Tax Data U16b.txt:
[Choose File] No file chosen
[Submit New Data]

Prikabinamo U16a.txt failą:

Plaese Upload remaining data files

Tax Info U16a.txt:
[Choose File] No file chosen

Every Citizen Tax Data U16b.txt:
[Choose File] No file chosen
[Submit New Data]

U16a.txt Initial data:

Tax Code	Tax Company Name:	Price:
11	Elektra	0.12
21	Dujos	0.58
31	Benzinas	1.78
32	Diezelis	1.9

Prikabinome U16b.txt:

LAB02 U16

Tax Info U16a.txt:
Choose File | No file chosen

Every Citizen Tax Data U16b.txt:
Choose File | No file chosen
Submit New Data

U16a.txt Initial data:

Tax Code	Tax Company Name:	Price:
11	Elektra	0.12
21	Dujos	0.58
31	Benzinas	1.78
32	Diezelis	1.9

U16b.txt Initial data:

Last Name	First Name	Address	Month	Tax Code	Amount
pavardė1	vardas1	adresas1	1	22	28
pavardė1	vardas1	adresas1	5	22	20
pavardė1	vardas1	adresas1	1	32	100
pavardė1	vardas1	adresas1	2	32	97
pavardė1	vardas1	adresas1	3	32	63
pavardė1	vardas1	adresas1	2	22	25
pavardė1	vardas1	adresas1	3	22	29
pavardėAA	vardasAA	adresasAA	1	21	13
pavardėAA	vardasAA	adresasAA	2	21	84
pavardėAA	vardasAA	adresasAA	3	21	76
pavardė1	vardas1	adresas1	4	22	39
pavardė2	vardas2	adresas2	3	31	67
pavardė2	vardas2	adresas2	4	31	98
pavardė0	vardas2	adresas2	5	31	125
pavardė0	vardas0	adresas3	1	11	31
pavardė1	vardas1	adresas1	4	32	39
pavardė1	vardas1	adresas1	5	32	20
pavardė1	vardas1	adresas1	3	11	80
pavardė1	vardas1	adresas1	4	11	39
pavardė1	vardas1	adresas1	1	11	120
pavardė1	vardas1	adresas1	2	11	100
pavardė1	vardas1	adresas1	5	11	139
pavardė2	vardas2	adresas2	1	31	31
pavardė2	vardas2	adresas2	2	31	48
pavardė0	vardas0	adresas3	2	11	48
pavardė0	vardas0	adresas3	3	11	67
pavardė0	vardas0	adresas3	4	11	98
pavardė0	vardas0	adresas3	5	11	125
pavardėAA	vardasAA	adresasAA	4	21	8
pavardėAA	vardasAA	adresasAA	5	21	25

All Citizen taxes over the months

Last Name	First Name	Address	Tax Sum
pavardėAA	vardasAA	adresasAA	119.48
pavardė0	vardas0	adresas3	44.28
pavardė2	vardas2	adresas2	434.32
pavardė0	vardas2	adresas2	222.5
pavardė1	vardas1	adresas1	663.46

Average tax per citizen: 296.808
Total tax sum: 1484.04

Above Average Tax:

Last Name	First Name	Address	Tax Sum
pavardė2	vardas2	adresas2	434.32
pavardė1	vardas1	adresas1	663.46

No Filter provided

Tax Code:

Month:

Submit

Prafiltravome duomenis pagal kodą: „11", mėnesį: „2". Prisidėjo lentelė papildoma.

All Citizen taxes over the months

Last Name	First Name	Address	Tax Sum
pavardėAA	vardasAA	adresasAA	119.48
pavardė0	vardas0	adresas3	44.28
pavardė2	vardas2	adresas2	434.32
pavardė0	vardas2	adresas2	222.5
pavardė1	vardas1	adresas1	663.46

Average tax per citizen: 296.808
Total tax sum: 1484.04

Above Average Tax:

Last Name	First Name	Address	Tax Sum
pavardė2	vardas2	adresas2	434.32
pavardė1	vardas1	adresas1	663.46

Filtered data:

Last Name	First Name	Address	Tax Sum
pavardė0	vardas0	adresas3	44.28
pavardė1	vardas1	adresas1	663.46

Tax Code:

Month:

Submit

```
U16result.txt:

Tax Sum of all citizens:

LastName            FirstName       |Address         |TaxSum    |
pavardė1             vardas1        | adresas1       |    663.46|
pavardėAA            vardasAA       | adresasAA      |    119.48|
pavardė2             vardas2        | adresas2       |    434.32|
pavardė0             vardas2        | adresas2       |    222.50|
pavardė0             vardas0        | adresas3       |     44.28|

Tax Sum of all citizens SORTED A-Z:

LastName            FirstName       |Address         |TaxSum    |
pavardėAA            vardasAA       | adresasAA      |    119.48|
pavardė0             vardas0        | adresas3       |     44.28|
pavardė2             vardas2        | adresas2       |    434.32|
pavardė0             vardas2        | adresas2       |    222.50|
pavardė1             vardas1        | adresas1       |    663.46|

All Citizen TOTAL Tax Sum: 1484.04

Average Tax Sum: 296.81

Citizens who paid above average:

LastName            FirstName       |Address         |TaxSum    |
pavardė2             vardas2        | adresas2       |    434.32|
pavardė1             vardas1        | adresas1       |    663.46|

Citizens who paid TaxCode: "11" on Month: "2"

LastName            FirstName       |Address         |TaxSum    |
pavardė0             vardas0        | adresas3       |     44.28|
pavardė1             vardas1        | adresas1       |    663.46|

Duomenys 2:

U16a.txt:

VAND; Vanduo; 0.07
KVND; Karštas vanduo; 0.20
LH20; Ledinis Vanduo; 0.10

U16b.txt:

Pavardauskis; Vardenis; Adresatas;Vasaris;Benzinas;28;
Pavardauskis; Vardenis; Adresatas;Vasaris;VAND;14;
Pavardauskis; Vardenis; Adresatas;Kovas;KVND;20;
Pavardauskis; Vardenis; Adresatas;Kovas;LH20;30;
Pavardauskis; Vardenis; Adresatas;Kovas;VAND;15;
Pavardauskis; Vardenis; Adresatas;Balandis;VAND;99;
Tomas; Tomukas; Tomo namas 1;Kovas;VAND;97;
Tomas; Tomukas; Tomo namas 1;Balandis;VAND;156;
Tomas; Tomukas; Tomo namas 1;Rugsėjis;VAND;20;

Rezultatai:
```

Vartotojo sąsaja
Naudojant mėnesį: Balandis ir mokesčių kodą: VAND:

LAB02 U16

Tax Info U16a.txt:
[Choose File] No file chosen

Every Citizen Tax Data U16b.txt:
[Choose File] No file chosen
[Submit New Data]

U16a.txt Initial data:

Tax Code	Tax Company Name:	Price:
VAND	Vanduo	0.07
KVND	Karštas vanduo	0.2
LH20	Ledinis Vanduo	0.1

U16b.txt Initial data:

Last Name	First Name	Address	Month	Tax Code	Amount
Pavardauskis	Vardenis	Adresatas	Vasaris	Benzinas	28
Pavardauskis	Vardenis	Adresatas	Vasaris	VAND	14
Pavardauskis	Vardenis	Adresatas	Kovas	KVND	20
Pavardauskis	Vardenis	Adresatas	Kovas	LH20	30
Pavardauskis	Vardenis	Adresatas	Kovas	VAND	15
Pavardauskis	Vardenis	Adresatas	Balandis	VAND	99
Tomas	Tomukas	Tomo namas 1	Kovas	VAND	97
Tomas	Tomukas	Tomo namas 1	Balandis	VAND	156
Tomas	Tomukas	Tomo namas 1	Rugsėjis	VAND	20

All Citizen taxes over the months

Last Name	First Name	Address	Tax Sum
Tomas	Tomukas	Tomo namas 1	19.11
Pavardauskis	Vardenis	Adresatas	15.96

Average tax per citizen: 17.535
Total tax sum: 35.07

Above Average Tax:

Last Name	First Name	Address	Tax Sum
Tomas	Tomukas	Tomo namas 1	19.11

Filtered data:

Last Name	First Name	Address	Tax Sum
Tomas	Tomukas	Tomo namas 1	19.11
Pavardauskis	Vardenis	Adresatas	15.96

Tax Code:
[   ]
Month:
[   ]
[Submit]

```
U16result.txt:

Tax Sum of all citizens:

LastName               FirstName          |Address               |TaxSum    |
Pavardauskis           Vardenis           | Adresatas            |    15.96|
Tomas                  Tomukas            | Tomo namas 1         |    19.11|


Tax Sum of all citizens SORTED A-Z:

LastName               FirstName          |Address               |TaxSum    |
Tomas                  Tomukas            | Tomo namas 1         |    19.11|
Pavardauskis           Vardenis           | Adresatas            |    15.96|


All Citizen TOTAL Tax Sum: 35.07

Average Tax Sum: 17.54

Citizens who paid above average:

LastName               FirstName          |Address               |TaxSum    |
Tomas                  Tomukas            | Tomo namas 1         |    19.11|


Citizens who paid TaxCode: "VAND" on Month: "Balandis"

LastName               FirstName          |Address               |TaxSum    |
Tomas                  Tomukas            | Tomo namas 1         |    19.11|
Pavardauskis           Vardenis           | Adresatas            |    15.96|
```

2.8. Dėstytojo pastabos

1. Klasių diagramoje nebūna žodžių private ar public. Tam yra spec. simboliai.
2. Negalima private int count;
3. Kam žodis internal, internal class Node?
4. Negalim painioti su sąsaja:
   public TableRow GetRow(int index
5. Taip neturi būti:
   public string TaxCode { get; set; }
   public string TaxName { get; set; }
   public double Price { get; set; }
   public Node next;
6. Kur klasės Tax sąsajos metodai?
   Perdaryti.
7. Negaliu sutikti su tokia klase Node:
   public Node(string lastName, string firstName, string address)
   {
   Data = new CitizenData(lastName, firstName, address);
   }

44

```csharp
/// <summary>
/// Swaps the DATA, keeps the pointers
/// </summary>
/// <param name="other">Other node to be swapped with</param>
public void SwapData(Node other)
{
CitizenData temp = Data;
Data = other.Data;
other.Data = temp;
}
```

8. Nėra klasės InoutUtils.
9. AddMoney(string   lastName,   string   firstName,   string   address,   double
   Kodėl ne citizen?
10. Kodėl nėra Next antraštėje?
    `public Node(CitizenData data)`

3. Bendrinės klasės ir testavimas (L3)

3.1. Darbo užduotis

LD_16. **Mokesčiai.** Kiekvieną mėnesį gyventojai moka komunalinius mokesčius. Suraskite, kurį mėnesį ir kokie komunaliniai mokesčiai kainavo pigiausiai. Apskaičiuokite, kokią pinigų sumą komunaliniams mokesčiams išleido visi gyventojai. Sudarykite sąrašą gyventojų (pavardė ir vardas, adresas), kurie už komunalines paslaugas per metus mokėjo sumą, mažesnę už vidutinę. Sąrašas turi būti surikiuotas pagal gyventojų adresus, pavardes ir vardus abėcėlės tvarka.

Duomenys:

- tekstiniame faile U16a.txt yra informacija apie komunalines paslaugas: paslaugos kodas, paslaugos pavadinimas, paslaugos vieno mėnesio vieno vieneto kaina;
- tekstiniame faile U16b.txt yra informacija apie gyventojus: pavardė ir vardas, adresas, mėnuo už kurį mokama, komunalinės paslaugos kodas, sunaudotų per mėnesį vienetų kiekis.

Pašalinkite iš sąrašo gyventojus, kurie nemokėjo už nurodytą paslaugą, nurodytą mėnesį (duomenys įvedami klaviatūra).

3.2. Grafinės vartotojo sąsajos schema

3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
HeaderLabel	Text	LAB02 U16
Label1	Text	Tax InfoU16a.txt:
Label2	Text	Every Citizen Tax Data U16b:
InitTaxLabel	Text	U16a.txt Initial data:
InitCitizenLabel	Text	U16b.txt Initial data:
CitizenTaxLabel	Text	All Citizen taxes over the months

AverageTax	Text	""
TotalTaxSum	Text	""
CitizenTaxLabel0	Text	Above Average Tax:
FilterData	Text	Filtered data:
ButtonFilter	Text	Tax Code:
DataButton	Text	Month:

3.4. Klasių diagrama
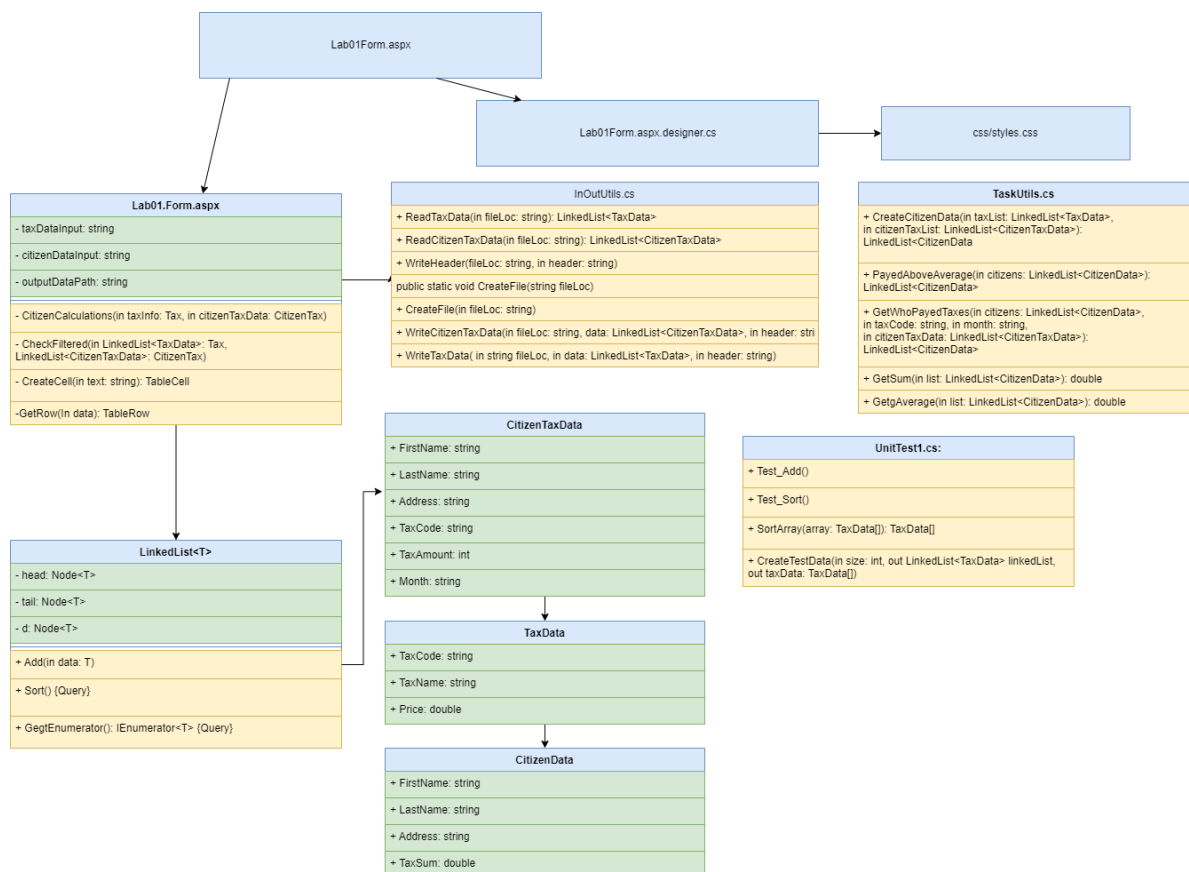
3.5. Programos vartotojo vadovas

Jeigu neranda failų visų duombazėje, programa paprašo failų. Jeigu randa tik vieną pradinį failą, rodo tik jį ir prašo likusių failų. Kai abu failai atsiranda duombazėje, užkrauna skaičiavimus. Apskaičiuoja vidutinę mokesčių kainą, sumą visų ir individualių žmonių. Tekstas yra rikiuojamas A-Z pagal: adresą, pavardę, vardą. Kodas leidžia filtruoti žmones, kurie mokėjo nurodytą mėnesį (mėnuo yra string) už nurodytus mokesčius naudojant „Tax Code" (string). Prie filtered lentelės prideda tik filtruotus duomenis.

3.6. Programos tekstas

CitizenData.cs:

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab02
{
    /// <summary>
    /// CitizenData class object to be used by class Citizen
    /// </summary>
    public class CitizenData : IComparable<CitizenData>, IEquatable<CitizenData>
    {

        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Address { get; set; }

        public double TaxSum { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="lastName">Last name of the citizen</param>
        /// <param name="firstName">First Name of the citizen</param>
        /// <param name="address">Address of the citizen</param>
        public CitizenData(string lastName, string firstName, string address, double
taxSum)
        {
            LastName = lastName;
            FirstName = firstName;
            Address = address;
            TaxSum = taxSum;
        }


        /// <summary>
        /// To String override
        /// </summary>
        /// <returns>stringg format of the citizen</returns>
        public override string ToString()
        {
            return $"{LastName,-20} {FirstName,-20}|{Address,-20}|{TaxSum,10:f}|";
        }

        /// <summary>
        /// Compares to other Node of citizen type
        /// </summary>
        /// <param name="other"></param>
        /// <returns>Integer</returns>
        public int CompareTo(CitizenData other)
        {
            int comparison = other.Address.CompareTo(Address);
            if (comparison == 0)
            {
                comparison = other.LastName.CompareTo(LastName);
                if (comparison == 0)
                {
                    comparison = other.FirstName.CompareTo(FirstName);
                }
            }
```

```csharp
            return comparison;
        }

        /// <summary>
        /// IEquatable iomplementation
        /// </summary>
        /// <param name="other">Comparison object</param>
        /// <returns>Boolean</returns>
        public bool Equals(CitizenData other)
        {
            if (FirstName == other.FirstName && LastName == other.LastName && Address ==
other.Address)
                return true;
            return false;
        }

        /// <summary>
        /// IEquatable iomplementation
        /// </summary>
        /// <param name="other">Comparison object</param>
        /// <returns>Boolean</returns>
        public bool Equals(CitizenTaxData other)
        {
            if (FirstName == other.FirstName && LastName == other.LastName && Address ==
other.Address)
                return true;
            return false;
        }
    }
}


CitizenTaxData.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab02
{
    public class CitizenTaxData : IComparable<CitizenTaxData>, IEquatable<CitizenTaxData>
    {

        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Address { get; set; }
        public string TaxCode { get; set; }
        public int TaxAmount { get; set; }
        public string Month { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="lastName">last name of citizen</param>
        /// <param name="firstName">first name of citizen</param>
        /// <param name="address">address of the citizen</param>
        /// <param name="month">the month the tax was paid</param>
        /// <param name="taxCode">tax code</param>
        /// <param name="taxAmount">tax amount</param>
        public CitizenTaxData(string lastName, string firstName, string address, string
month, string taxCode, int taxAmount)
        {
            FirstName = firstName;
            LastName = lastName;
            Address = address;
```

```csharp
            TaxCode = taxCode;
            TaxAmount = taxAmount;
            Month = month;
        }

        /// <summary>
        /// ToString implementation
        /// </summary>
        /// <returns>String</returns>
        public override string ToString()
        {
            return $"{LastName,-20} {FirstName,-20}|{Address,-20}|{Month,-15}|{TaxCode,-20}|{TaxAmount,10}|";
        }

        /// <summary>
        /// IEquatable implementation
        /// </summary>
        /// <param name="other">Comparison object</param>
        /// <returns>Boolean</returns>
        public bool Equals(CitizenTaxData other)
        {
            if (FirstName == other.FirstName && LastName == other.LastName && Address == other.Address)
                return true;
            return false;
        }

        /// <summary>
        /// IComparable Implementation
        /// </summary>
        /// <param name="other">Comparison object</param>
        /// <returns>Integer</returns>
        public int CompareTo(CitizenTaxData other)
        {
            int comparison = LastName.CompareTo(other.LastName);
            if (comparison == 0)
            {
                comparison = FirstName.CompareTo(other.FirstName);
                if (comparison == 0)
                    comparison = TaxAmount.CompareTo(other.TaxAmount);
            }

            return comparison;
        }
    }
}
```

TaxData.cs:

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab02
{
    /// <summary>
    /// TaxData object to be inherited by Tax object
    /// </summary>
    public class TaxData : IComparable<TaxData>, IEquatable<TaxData>
    {
        public string TaxCode { get; set; }
        public string TaxName { get; set; }
        public double Price { get; set; }
```

```csharp
        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="taxCode"></param>
        /// <param name="taxName"></param>
        /// <param name="price"></param>
        public TaxData(string taxCode, string taxName, double price)
        {
            TaxCode = taxCode;
            TaxName = taxName;
            Price = price;
        }

        /// <summary>
        /// Returns Node in string format
        /// </summary>
        /// <returns>Node in string format</returns>
        public override string ToString()
        {
            return $"{TaxCode,-20}|{TaxName,-20}|{Price,10:f}|";
        }

        /// <summary>
        /// IComparable implementation
        /// </summary>
        /// <param name="other">Comparison object</param>
        /// <returns>Integer</returns>
        public int CompareTo(TaxData other)
        {
            int comparison = Price.CompareTo(other.Price);
            return comparison;
        }

        /// <summary>
        /// IEquatable implementation
        /// </summary>
        /// <param name="other"> comparison object </param>
        /// <returns>boolean</returns>
        public bool Equals(TaxData other)
        {
            if (TaxCode == other.TaxCode)
                return true;
            return false;
        }
    }
}


LinkedList.cs:


using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab02
{
    /// <summary>
    /// Linked List Class Object
    /// </summary>
    /// <typeparam name="T">Object Type</typeparam>
    public class LinkedList<T> : IEnumerable<T> where T : IComparable<T>, IEquatable<T>
    {
        private Node<T> head;
        private Node<T> tail;
```

```csharp
    private Node<T> d;

    /// <summary>
    /// Construcotr
    /// </summary>
    public LinkedList()
    {
        head = null;
        tail = null;
    }

    //Deprecated, Use Foreach with IEnumerable
    /** Address of the head of the list is assigned */
    public void Begin()
    { d = head; }
    /** Interface variable gets address of the next entry*/
    public void Next()
    { d = d.next; }
    /** Return true, if list is empty*/
    public bool Exist()
    { return d != null; }
    //------------------------------------------------
    /** Return data according to the interface address*/
    public T Get()
    { return d.Data; }

    /// <summary>
    /// Adds T object to  Linked List
    /// </summary>
    /// <param name="data"> <T> Type Object</param>
    public void Add(T data)
    {
        // If No citizen was found, adds the citizen to Linked List
        if (head == null)
        {
            head = new Node<T>(data, null);
            tail = head;
        }
        else
        {
            tail.next = new Node<T>(data, null);
            tail = tail.next;
        }
    }

    /// <summary>
    /// Sort Function using iComprable
    /// </summary>
    public void Sort()
    {

        Node<T> timer = head;
        while (timer != null)
        {
            Node<T> curr = head;
            Node<T> next = head.next;
            while (next != null)
            {
                if (curr.Data.CompareTo(next.Data) > 0)
                {
                    curr.SwapData(next);
                }
                curr = next;
                next = next.next;
            }
            timer = timer.next;
        }
```

```csharp
            }

            /// <summary>
            /// IEnumerable implementation
            /// </summary>
            /// <returns>yield of T data</returns>
            public IEnumerator<T> GetEnumerator()
            {
                for (Node<T> dd = head; dd != null; dd = dd.next)
                {
                    yield return dd.Data;
                }
            }

            /// <summary>
            /// Obligatory, since IEnumerable<T> inherits IEnumerable
            /// </summary>
            /// <returns>none</returns>
            /// <exception cref="NotImplementedException">Not Implemented</exception>
            IEnumerator IEnumerable.GetEnumerator()
            {
                throw new NotImplementedException();
            }

            /// <summary>
            /// Node class to be used to save every citizen seperately
            /// </summary>
            class Node<T>
            {
                public T Data { get; set; }
                public Node<T> next { get; set; }

                /// <summary>
                /// Constructor
                /// </summary>
                /// <param name="data">CitizenData pointer</param>
                public Node(T data, Node<T> link)
                {
                    Data = data;
                    next = link;
                }

                /// <summary>
                /// Swaps the DATA, keeps the pointers
                /// </summary>
                /// <param name="other">Other node to be swapped with</param>
                public void SwapData(Node<T> other)
                {
                    T temp = Data;
                    Data = other.Data;
                    other.Data = temp;
                }
            }
        }
    }
}


TaskUtils.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI.WebControls;

namespace Lab02
```

```csharp
{
    /// <summary>
    /// TaskUtils static class for helper functions
    /// </summary>
    public static class TaskUtils
    {

        /// <summary>
        /// Creates Citizen class object using Tax object
        /// </summary>
        /// <param name="TaxList">Tax class object</param>
        /// <param name="citizenTaxList">CitizenTax object</param>
        /// <returns>Citizen class object</returns>
        public static LinkedList<CitizenData> CreateCitizenData(LinkedList<TaxData> taxList, LinkedList<CitizenTaxData> citizenTaxList)
        {
            LinkedList<CitizenData> citizens = new LinkedList<CitizenData>();
            // Goes through every tax data record
            foreach (CitizenTaxData citizenTaxData in citizenTaxList)
            {
                // Finds the the tax code and returns price
                foreach(TaxData taxData in taxList)
                {
                    if(citizenTaxData.TaxCode == taxData.TaxCode)
                    {
                        CitizenData temp = null;
                        // Finds the citizen if already exists
                        foreach (CitizenData citizen in citizens)
                        {
                            // Finds the citizen if it already exists
                            if(citizen.Equals(citizenTaxData))
                            {
                                temp = citizen;
                                break;
                            }
                        }

                        // Creates a new citizen or appends the data
                        if (temp != null)
                        {
                            temp.TaxSum += (double)taxData.Price * citizenTaxData.TaxAmount;
                        }
                        else
                        {
                            temp = new CitizenData(citizenTaxData.LastName, citizenTaxData.FirstName, citizenTaxData.Address, (double)taxData.Price * citizenTaxData.TaxAmount);
                            citizens.Add(temp);
                        }

                    }
                }
            }

            return citizens;
        }

        /// <summary>
        /// Returns a list for people who payed above average
        /// </summary>
        /// <param name="citizens">CitizenData Linked List</param>
        /// <returns>Citizen Data Linked List</returns>
        public static LinkedList<CitizenData> PayedAboveAverage(LinkedList<CitizenData> citizens)
        {
            double average = GetAverage(citizens);
```

```csharp
            LinkedList<CitizenData> output = new LinkedList<CitizenData>();

            foreach (CitizenData citizen in citizens)
            {
                if (citizen.TaxSum >= average)
                {
                    output.Add(citizen);
                }
            }

            return output;
        }

        /// <summary>
        /// Creates a new list who payed taxes specified tax, month
        ///
        /// </summary>
        /// <param name="citizens">CitizenData Linked List</param>
        /// <param name="taxCode">Tax Code to filter by</param>
        /// <param name="month">Month to filter by</param>
        /// <param name="citizenTaxData">CitizenTaxData Linked List</param>
        /// <returns></returns>
        public static LinkedList<CitizenData> GetWhoPayedTaxes(LinkedList<CitizenData> citizens, string taxCode, string month, LinkedList<CitizenTaxData> citizenTaxData)
        {
            LinkedList<CitizenData> output = new LinkedList<CitizenData>();
            foreach (CitizenData citizen in citizens)
            {
                foreach (CitizenTaxData citizenTax in citizenTaxData)
                {
                    if(citizenTax.TaxCode == taxCode && citizenTax.Month == month && citizen.Equals(citizenTax))
                    {
                        output.Add(citizen);
                        break;
                    }
                }
            }
            return output;
        }

        /// <summary>
        /// Returns Sum
        /// </summary>
        /// <param name="list">CitizenData LinkedList</param>
        /// <returns>Double</returns>
        public static double GetSum(LinkedList<CitizenData> list)
        {
            double sum = 0;
            foreach (CitizenData citizen in list)
                sum += citizen.TaxSum;

            return sum;
        }

        /// <summary>
        /// Gets Average
        /// </summary>
        /// <param name="list">CitizenData Linked List</param>
        /// <returns>Double</returns>
        public static double GetAverage(LinkedList<CitizenData> list)
        {
            double sum = 0;
            int i = 0;
            foreach (CitizenData citizen in list)
            {
                sum += citizen.TaxSum;
```

```csharp
                i++;
            }

            return (double)(i > 0 ? sum / i : 0);
        }


    }
}


InOutUtils.cs:


using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;

namespace Lab02
{
    /// <summary>
    /// Static InOutUtils helper class for Input/Output with files
    /// </summary>
    public static class InOutUtils
    {
        /// <summary>
        /// Reads Tax Data from txt to Tax class object+
        /// </summary>
        /// <param name="fileLoc">Location of the data in .txt format</param>
        /// <returns>Linked ListTaxData class object</returns>
        public static LinkedList<TaxData> ReadTaxData(string fileLoc)
        {
            LinkedList<TaxData> taxes = new LinkedList<TaxData>();
            string[] lines = File.ReadAllLines(fileLoc);
            foreach (string line in lines)
            {
                string[] elements = line.Split(';');
                taxes.Add(new TaxData(elements[0], elements[1],
double.Parse(elements[2])));
            }
            return taxes;
        }

        /// <summary>
        /// Creates CitizenTaxData from .txt file
        /// </summary>
        /// <param name="fileLoc">Location of .txt file</param>
        /// <returns>LinkedList CitizenTaxData class object</returns>
        public static LinkedList<CitizenTaxData> ReadCitizenTaxData(string fileLoc)
        {
            LinkedList<CitizenTaxData> data = new LinkedList<CitizenTaxData>();
            string[] lines = File.ReadAllLines(fileLoc);
            foreach (string line in lines)
            {
                string[] elements = line.Split(';');
                CitizenTaxData temp = new CitizenTaxData(elements[1], elements[0],
elements[2], elements[3], elements[4], int.Parse(elements[5]));
                data.Add(temp);
            }
            return data;
        }

        /// <summary>
        /// Appends a header to a file
        /// </summary>
```

```csharp
        /// <param name="fileLoc">Name/location of the file</param>
        /// <param name="header">text to be appended</param>
        public static void WriteHeader(string fileLoc, string header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append: true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
            }
        }

        /// <summary>
        /// Creates a new or wipes a file
        /// </summary>
        /// <param name="fileLoc">Location of the file</param>
        public static void CreateFile(string fileLoc)
        {
            using (FileStream fs = new FileStream(fileLoc, FileMode.Create))
                new StreamWriter(fs, encoding: System.Text.Encoding.UTF8).Close();
        }

        /// <summary>
        /// Appends CitizenTaxData to a file
        /// </summary>
        /// <param name="fileLoc">Location/name of the file</param>
        /// <param name="data">data to append to the .txt file</param>
        /// <param name="header">Header text of the data file</param>
        public static void WriteCitizenTaxData(string fileLoc, LinkedList<CitizenTaxData>
data, string header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append:true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
                writer.WriteLine($"{"LastName",-20} {"FirstName",-20}|{"Address",-
20}|{"Month",-15}|{"TaxCode",-20}|{"TaxAmount",10}|");
                foreach (CitizenTaxData taxData in data)
                {
                    writer.WriteLine(taxData.ToString());
                }
                writer.WriteLine();
            }
        }

        /// <summary>
        /// appends Citizen class object data to text file
        /// </summary>
        /// <param name="fileLoc">location/name of the file</param>
        /// <param name="data">data to append to the file</param>
        /// <param name="header">Header of the file</param>
        public static void WriteCitizenData(string fileLoc, LinkedList<CitizenData> data,
string header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append: true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
                writer.WriteLine($"{"LastName",-20} {"FirstName",-20}|{"Address",-
20}|{"TaxSum",-10}|");
                foreach (CitizenData taxData in data)
                {
                    writer.WriteLine(taxData.ToString());
                }
                writer.WriteLine();
            }
        }
```

```csharp
        /// <summary>
        /// Appends Tax data to a .txt file
        /// </summary>
        /// <param name="fileLoc">Location/name of the file</param>
        /// <param name="data">data to append to the .txt file</param>
        /// <param name="header">header to be added to the file</param>
        public static void WriteTaxData(string fileLoc, LinkedList<TaxData> data, string
header)
        {
            using (StreamWriter writer = new StreamWriter(fileLoc, append: true))
            {
                writer.WriteLine(header);
                writer.WriteLine();
                writer.WriteLine($"{"TaxCode",-20}|{"TaxName",-20}|{"Price",10:2f}|");
                foreach (TaxData taxData in data)
                {
                    writer.WriteLine(taxData.ToString());
                }
                writer.WriteLine();
            }
        }
    }
}
```

UnitTest1.cs:

```csharp
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using Lab02;

namespace UnitTest
{
    [TestClass]
    public class UnitTest1
    {
        /// <summary>
        /// Compares CitizenData object with same parameters. Comparison should return 0
        /// </summary>
        [TestMethod]
        public void CompareTo_CitizenDataSame_Returns0()
        {
            CitizenData cData1 = new CitizenData("lastname", "firstname", "address", 0);
            CitizenData cData2 = new CitizenData("lastname", "firstname", "address", 0);
            Assert.AreEqual(cData1.CompareTo(cData2),0);
        }

        /// <summary>
        /// Compares lhs CitizenData object with alphabetically higher parameters.
Comparison should return 1
        /// </summary>
        [TestMethod]
        public void CompareTo_CitizenDataSame_Returns1()
        {
            CitizenData cData1 = new CitizenData("a", "a", "a", 0);
            CitizenData cData2 = new CitizenData("b", "b", "b", 0);
            Assert.AreEqual(cData1.CompareTo(cData2), 1);
        }

        /// <summary>
        /// Compares lhs CitizenData object with alphabetically lower parameters.
Comparison should return -1
        /// </summary>
        [TestMethod]
        public void CompareTo_CitizenDataSame_ReturnsMinus1()
        {
            CitizenData cData1 = new CitizenData("b", "b", "b", 0);
            CitizenData cData2 = new CitizenData("a", "a", "a", 0);
            Assert.AreEqual(cData1.CompareTo(cData2), -1);
        }

        /// <summary>
        /// Tests CitizenTax and CitizenTaxData object comparison with different
parameters
        /// </summary>
        [TestMethod]
        public void Equals_CitizenDataCitizenTaxDataDifferentParameters_False()
        {
            CitizenData cData = new CitizenData("lastname1", "firstname1", "address1",
0);
            CitizenTaxData cTaxData = new CitizenTaxData("lastname0", "firstname0",
"address0", "April", "0", 0);
            Assert.IsFalse(cData.Equals(cTaxData));
        }

        /// <summary>
        /// Tests CitizenTax and CitizenTaxData object comparison with same parameters
        /// </summary>
        [TestMethod]
        public void Equals_CitizenDataCitizenTaxDataSameParameters_True()
        {
            CitizenData cData = new CitizenData("lastname", "firstname", "address", 0);
```

```csharp
        CitizenTaxData cTaxData = new CitizenTaxData("lastname", "firstname",
"address", "April", "0", 0);
        Assert.IsTrue(cData.Equals(cTaxData));
    }

    /// <summary>
    /// Tests 2 Citizen Data Comparison with same parameters. Should Return True.
    /// </summary>
    [TestMethod]
    public void Equals_CitizenDataSameParameters_True()
    {
        CitizenData cData1 = new CitizenData("lastname1", "firstname1", "address1",
0);
        CitizenData cData2 = new CitizenData("lastname1", "firstname1", "address1",
0);

        Assert.IsTrue(cData1.Equals(cData2));
    }

    /// <summary>
    /// Tests 2 Citizen Data Comparison with different Keys. Should Return false.
    /// </summary>
    [TestMethod]
    public void Equals_CitizenDataDifferentParameters_False()
    {
        CitizenData cData1 = new CitizenData("lastname1", "firstname1", "address1",
0);
        CitizenData cData2 = new CitizenData("lastname2", "firstname2", "address2",
0);

        Assert.IsFalse(cData1.Equals(cData2));
    }

    /// <summary>
    /// Tets add function and compares to array
    /// </summary>
    [TestMethod]
    public void Add_LinkedListArrayEquality_True()
    {
        TaxData[] testArray;
        LinkedList<TaxData> list;
        CreateTestData(10, out list, out testArray);

        int index = 0;
        foreach (TaxData taxData in list)
        {
            Assert.IsTrue(taxData.Equals(testArray[index]));
            index++;
        }
    }

    /// <summary>
    /// Tests LinkedList Sort() function and compares to array funcction
    /// </summary>
    [TestMethod]
    public void Sort_LinkedListArrayEquality_True()
    {
        TaxData[] testArray;
        LinkedList<TaxData> list;
        CreateTestData(10, out list, out testArray);

        list.Sort();
        testArray = SortArray(testArray);
        int index = 0;
        foreach (TaxData taxData in list)
        {
            Assert.IsTrue(taxData.Equals(testArray[index]));
```

```csharp
                index++;
            }
        }

        /// <summary>
        /// Sorts TaxData[] object to compare to LinkedList object
        /// </summary>
        /// <param name="array">Unsorted object</param>
        /// <returns>sorted array</returns>
        public TaxData[] SortArray(TaxData[] array)
        {
            for (int i = 0; i < array.Length - 1; i++)
            {
                for (int j = 0; j < array.Length - 1 - i; j++)
                {
                    if(array[j].CompareTo(array[j+1]) > 0)
                    {
                        // SWAP
                        TaxData temp = array[j];
                        array[j] = array[j+1];
                        array[j+1] = temp;
                    }
                }
            }
            return array;
        }

        /// <summary>
        /// Creates test data with the same objects to compare while doing testsgggg
        /// </summary>
        /// <param name="size">amount of elements</param>
        /// <param name="linkedList">OUT Lab03 implementation of linked list</param>
        /// <param name="taxData">OUT Array</param>
        public void CreateTestData(int size, out LinkedList<TaxData> linkedList, out
TaxData[] taxData)
        {
            linkedList = new LinkedList<TaxData>();
            taxData = new TaxData[size];
            for (int i = 0; i < size; i++)
            {
                Random rng = new Random();
                TaxData temp = new TaxData(rng.Next(100).ToString(), i.ToString(),
(double)i/10);
                linkedList.Add(temp);
                taxData[i] = temp;
            }
        }

    }
}
```

Lab01Form.aspx:

```aspx
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Lab01Form.aspx.cs"
Inherits="Lab02.Lab01Form" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <link rel="stylesheet" runat="server" media="screen" href="~/css/styles.css" />
    <title>Lab02 U16</title>
</head>
<body>
    <form id="form1" runat="server">
        <div id="body">
            <asp:Label ID="HeaderLabel" runat="server" Text="LAB02 U16"></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Tax Info U16a.txt:"></asp:Label>
            <br />
            <asp:FileUpload ID="FileUpload1" runat="server" />
            <br />
            <br />
            <asp:Label ID="Label2" runat="server" Text="Every Citizen Tax Data U16b.txt:
"></asp:Label>
            <br />
            <asp:FileUpload ID="FileUpload2" runat="server" />
            <br />
            <asp:Button ID="DataButton" runat="server" Text="Submit New Data"
OnClick="DataButton_Click" />
            <br />
            <br />
            <asp:Label ID="InitTaxLabel" runat="server" Text="U16a.txt Initial
data:"></asp:Label>
            <asp:Table ID="InitTaxTable" runat="server">
            </asp:Table>
            <br />
            <asp:Label ID="InitCitizenLabel" runat="server" Text="U16b.txt Initial
data:"></asp:Label>
            <asp:Table ID="InitCitizenTable" runat="server">
            </asp:Table>
            <br />
            <asp:Panel ID="CalculationsPanel" runat="server">
                <asp:Label ID="CitizenTaxLabel" runat="server" Text="All Citizen taxes
over the months"></asp:Label>
                <asp:Table ID="CitizenTaxTable" runat="server">
                </asp:Table>
                <br />
                <asp:Label ID="AverageTax" runat="server"></asp:Label>
                <br />
                <asp:Label ID="TotalTaxSum" runat="server"></asp:Label>
                <br />
                <br />
                <asp:Label ID="CitizenTaxLabel0" runat="server" Text="Above Average
Tax:"></asp:Label>
                <asp:Table ID="AboveAverageTable" runat="server">
                </asp:Table>
                <br />
                <asp:Label ID="FilterData" runat="server" Text="Filtered
data:"></asp:Label>
                <asp:Table ID="FilterTable" runat="server">
                </asp:Table>
                <br />
                Tax Code:<br />
                <asp:TextBox ID="TaxCodeTextBox" runat="server"></asp:TextBox>
                <br />
```

```html
                Month:<br />
                <asp:TextBox ID="TaxMonthTextBox" runat="server"></asp:TextBox>
                <br />
                <asp:Button ID="ButtonFilter" runat="server" Text="Submit"
OnClick="ButtonFilter_Click" />
            </asp:Panel>
            <br />
        </div>
    </form>
</body>
</html>
```

```csharp
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Lab02
{
    public partial class Lab01Form : System.Web.UI.Page
    {
        private string taxDataInput = @"App_Data/U16a.txt";
        private string citizenDataInput = @"App_Data/U16b.txt";
        private string outputDataPath = @"App_Data/U16result.txt";
        protected void Page_Load(object sender, EventArgs e)
        {
            LinkedList<CitizenTaxData> citizenTaxData = null;
            LinkedList<TaxData> taxInfo = null;
            InOutUtils.CreateFile(Server.MapPath(outputDataPath));
            if (File.Exists(Server.MapPath(taxDataInput)))
            {
                taxInfo = InOutUtils.ReadTaxData(Server.MapPath(taxDataInput));
                InOutUtils.WriteTaxData(Server.MapPath(outputDataPath), taxInfo, "Initial
Tax Company Data:");
                FillTaxDataTable(taxInfo, InitTaxTable);
            }
            else
            {
                InitTaxLabel.Text = "";
            }

            if (File.Exists(Server.MapPath(citizenDataInput)))
            {
                citizenTaxData =
InOutUtils.ReadCitizenTaxData(Server.MapPath(citizenDataInput));
                //citizenTaxData.Sort(); Test
                InOutUtils.WriteCitizenTaxData(Server.MapPath(outputDataPath),
citizenTaxData, "Initial Citizen Tax Data:");
                FillCitizenTaxDataTable(citizenTaxData, InitCitizenTable);
            }
            else
            {
                InitCitizenLabel.Text = "";
            }

            if (citizenTaxData != null && taxInfo != null)
            {
                // Reads Initial Data and Outputs the Initial Data To WebForm and to text

                CitizenCalculations(taxInfo, citizenTaxData);
                CheckFiltered(taxInfo, citizenTaxData);
```

```csharp
            }
            else
            {
                HeaderLabel.Text = "Plaese Upload remaining data files";
                CalculationsPanel.Visible = false;
            }
        }

        /// <summary>
        /// Does calculations from Tax and CitizenTax object
        /// </summary>
        /// <param name="taxInfo">Tax object</param>
        /// <param name="citizenTaxData">CitizenTax object</param>
        protected void CitizenCalculations(LinkedList<TaxData> taxInfo,
LinkedList<CitizenTaxData> citizenTaxData)
        {
            LinkedList<CitizenData> citizensAverage =
TaskUtils.CreateCitizenData(taxInfo, citizenTaxData); // For Above Average
            InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath), citizensAverage,
"Tax Sum of all citizens:");

            citizensAverage.Sort();
            InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath), citizensAverage,
"Tax Sum of all citizens SORTED A-Z:");
            FillCitizenTable(citizensAverage, CitizenTaxTable);

            double sum = TaskUtils.GetSum(citizensAverage);
            double average = TaskUtils.GetAverage(citizensAverage);
            InOutUtils.WriteHeader(Server.MapPath(outputDataPath), $"All Citizen TOTAL
Tax Sum: {sum:f}");
            InOutUtils.WriteHeader(Server.MapPath(outputDataPath), $"Average Tax Sum:
{average:f}");
            AverageTax.Text = $"Average tax per citizen: {average}";
            TotalTaxSum.Text = $"Total tax sum: {sum}";

            citizensAverage = TaskUtils.PayedAboveAverage(citizensAverage);
            InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath), citizensAverage,
"Citizens who paid above average:");
            FillCitizenTable(citizensAverage, AboveAverageTable);
        }
        /// <summary>
        /// Updates filtered data
        /// </summary>
        /// <param name="taxInfo">Tax Object</param>
        /// <param name="citizenTaxData">CitizenTax object</param>
        protected void CheckFiltered(LinkedList<TaxData> taxInfo,
LinkedList<CitizenTaxData> citizenTaxData)
        {
            if (Session["TaxCode"] != null && Session["Month"] != null)
            {
                LinkedList<CitizenData> citizensFiltered =
TaskUtils.CreateCitizenData(taxInfo, citizenTaxData); // For Filter
                citizensFiltered = TaskUtils.GetWhoPayedTaxes(citizensFiltered,
Session["TaxCode"].ToString(), Session["Month"].ToString(), citizenTaxData);
                citizensFiltered.Sort();
                InOutUtils.WriteCitizenData(Server.MapPath(outputDataPath),
citizensFiltered, $"Citizens who paid TaxCode: \"{Session["TaxCode"]}\" on Month:
\"{Session["Month"]}\"");
                FillCitizenTable(citizensFiltered, FilterTable);
            }
            else
            {
                FilterData.Text = "No Filter provided";
                FilterTable.Visible = false;
            }

            Session["TaxCode"] = null;
```

```csharp
            Session["Month"] = null;
        }

        /// <summary>
        /// Fills Table from CitizenTax object
        /// </summary>
        /// <param name="citizenTaxes">CitizenTaxData LinkedList</param>
        /// <param name="table">Table UI object</param>
        protected void FillCitizenTaxDataTable(LinkedList<CitizenTaxData> citizenTaxes,
Table table)
        {
            TableRow headerRow = new TableRow();
            headerRow.Cells.Add(CreateCell("Last Name"));
            headerRow.Cells.Add(CreateCell("First Name"));
            headerRow.Cells.Add(CreateCell("Address"));
            headerRow.Cells.Add(CreateCell("Month"));
            headerRow.Cells.Add(CreateCell("Tax Code"));
            headerRow.Cells.Add(CreateCell("Amount"));
            table.Rows.Add(headerRow);
            foreach (CitizenTaxData data in citizenTaxes)
            {
                table.Rows.Add(GetRow(data));
            }
        }

        /// <summary>
        /// Fills Table from Tax object
        /// </summary>
        /// <param name="taxes">LinkedList TaxData object</param>
        /// <param name="table">UI Table object</param>
        protected void FillTaxDataTable(LinkedList<TaxData> taxes, Table table)
        {
            TableRow headerRow = new TableRow();
            headerRow.Cells.Add(CreateCell("Tax Code"));
            headerRow.Cells.Add(CreateCell("Tax Company Name:"));
            headerRow.Cells.Add(CreateCell("Price:"));;
            table.Rows.Add(headerRow);
            foreach (TaxData data in taxes)
            {
                table.Rows.Add(GetRow(data));
            }
        }

        /// <summary>
        /// Fills citizen table
        /// </summary>
        /// <param name="citizens">Citizen object</param>
        /// <param name="table">UI.Table object</param>
        protected void FillCitizenTable(LinkedList<CitizenData> citizens, Table table)
        {
            TableRow headerRow = new TableRow();
            headerRow.Cells.Add(CreateCell("Last Name"));
            headerRow.Cells.Add(CreateCell("First Name"));
            headerRow.Cells.Add(CreateCell("Address"));
            headerRow.Cells.Add(CreateCell("Tax Sum"));
            table.Rows.Add(headerRow);
            foreach (CitizenData data in citizens)
            {
                table.Rows.Add(GetRow(data));
            }
        }

        protected void ButtonFilter_Click(object sender, EventArgs e)
        {
            string taxCode = TaxCodeTextBox.Text;
            string month = TaxMonthTextBox.Text;
            if (month != "" && taxCode != null)
```

```csharp
        {
            Session["TaxCode"] = TaxCodeTextBox.Text;
            Session["Month"] = TaxMonthTextBox.Text;
        }
        Response.Redirect("Lab01Form.aspx");
    }

    protected void DataButton_Click(object sender, EventArgs e)
    {
        if(FileUpload1.HasFile && FileUpload1.FileName.EndsWith(".txt"))
        {
            FileUpload1.SaveAs(Server.MapPath(taxDataInput));
        }
        if (FileUpload2.HasFile && FileUpload2.FileName.EndsWith(".txt"))
        {
            FileUpload2.SaveAs(Server.MapPath(citizenDataInput));
        }
        Response.Redirect("Lab01Form.aspx");
    }

    /// <summary>
    /// Creates TableCell from text to speed up TableCell creation
    /// </summary>
    /// <param name="text">string text to add to the table cell</param>
    /// <returns>TableCell class object</returns>
    protected static TableCell CreateCell(string text)
    {
        TableCell cell = new TableCell();
        cell.Text = text;
        return cell;
    }

    /// <summary>
    /// Creates TableRow from TaxData object
    /// </summary>
    /// <param name="data">TaxData object</param>
    /// <returns>TableRow object</returns>
    public TableRow GetRow(TaxData data)
    {
        TableRow row = new TableRow();
        row.Cells.Add(CreateCell(data.TaxCode));
        row.Cells.Add(CreateCell(data.TaxName));
        row.Cells.Add(CreateCell(data.Price.ToString()));
        return row;

    }

    /// <summary>
    /// Creates a row from CitizenTaxData
    /// </summary>
    /// <param name="data">CitizenTaxData class object</param>
    /// <returns>TableRow object</returns>
    public TableRow GetRow(CitizenTaxData data)
    {
        TableRow row = new TableRow();
        row.Cells.Add(CreateCell(data.LastName));
        row.Cells.Add(CreateCell(data.FirstName));
        row.Cells.Add(CreateCell(data.Address));
        row.Cells.Add(CreateCell(data.Month));
        row.Cells.Add(CreateCell(data.TaxCode));
        row.Cells.Add(CreateCell(data.TaxAmount.ToString()));
        return row;
    }
```

```
/// <summary>
/// Returns cictizen in TableRow format for the specified citizen
/// </summary>
/// <param name="data">data of the citizen</param>
/// <returns>TableRow format of the specified citizen</returns>
public TableRow GetRow(CitizenData data)
{
    TableRow row = new TableRow();
    row.Cells.Add(CreateCell(data.LastName));
    row.Cells.Add(CreateCell(data.FirstName));
    row.Cells.Add(CreateCell(data.Address));
    row.Cells.Add(CreateCell(data.TaxSum.ToString()));
    return row;
}
}
}
```
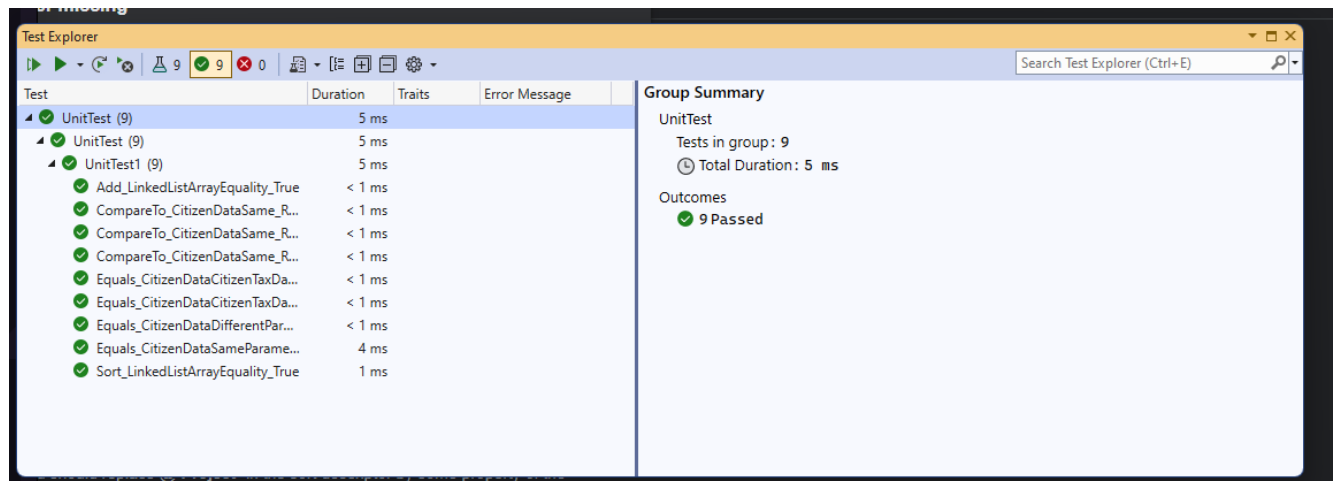
3.7. Pradiniai duomenys ir rezultatai

Testavimo rezultatai:

Pradiniai duomenys 1:

U16a.txt:

11; Elektra; 0.12
21; Dujos; 0.58
31; Benzinas; 1.78
32; Diezelis; 1.90

U16b.txt:

pavardė1; vardas1; adresas1;1;22;28;
pavardė1; vardas1; adresas1;5;22;20;
pavardė1; vardas1; adresas1;1;32;100;
pavardė1; vardas1; adresas1;2;32;97;
pavardė1; vardas1; adresas1;3;32;63;
pavardė1; vardas1; adresas1;2;22;25;
pavardė1; vardas1; adresas1;3;22;29;
pavardėAA; vardasAA; adresasAA;1;21;13;

```
pavardėAA; vardasAA; adresasAA;2;21;84;
pavardėAA; vardasAA; adresasAA;3;21;76;
pavardė1; vardas1; adresas1;4;22;39;
pavardė2; vardas2; adresas2;3;31;67;
pavardė2; vardas2; adresas2;4;31;98;
pavardė0; vardas2; adresas2;5;31;125;
pavardė0; vardas0; adresas3;1;11;31;
pavardė1; vardas1; adresas1;4;32;39;
pavardė1; vardas1; adresas1;5;32;20;
pavardė1; vardas1; adresas1;3;11;80;
pavardė1; vardas1; adresas1;4;11;39;
pavardė1; vardas1; adresas1;1;11;120;
pavardė1; vardas1; adresas1;2;11;100;
pavardė1; vardas1; adresas1;5;11;139;
pavardė2; vardas2; adresas2;1;31;31;
pavardė2; vardas2; adresas2;2;31;48;
pavardė0; vardas0; adresas3;2;11;48;
pavardė0; vardas0; adresas3;3;11;67;
pavardė0; vardas0; adresas3;4;11;98;
pavardė0; vardas0; adresas3;5;11;125;
pavardėAA; vardasAA; adresasAA;4;21;8;
pavardėAA; vardasAA; adresasAA;5;21;25;
```

Rezultatai 1:

Pradinė vartotojo sąsaja:
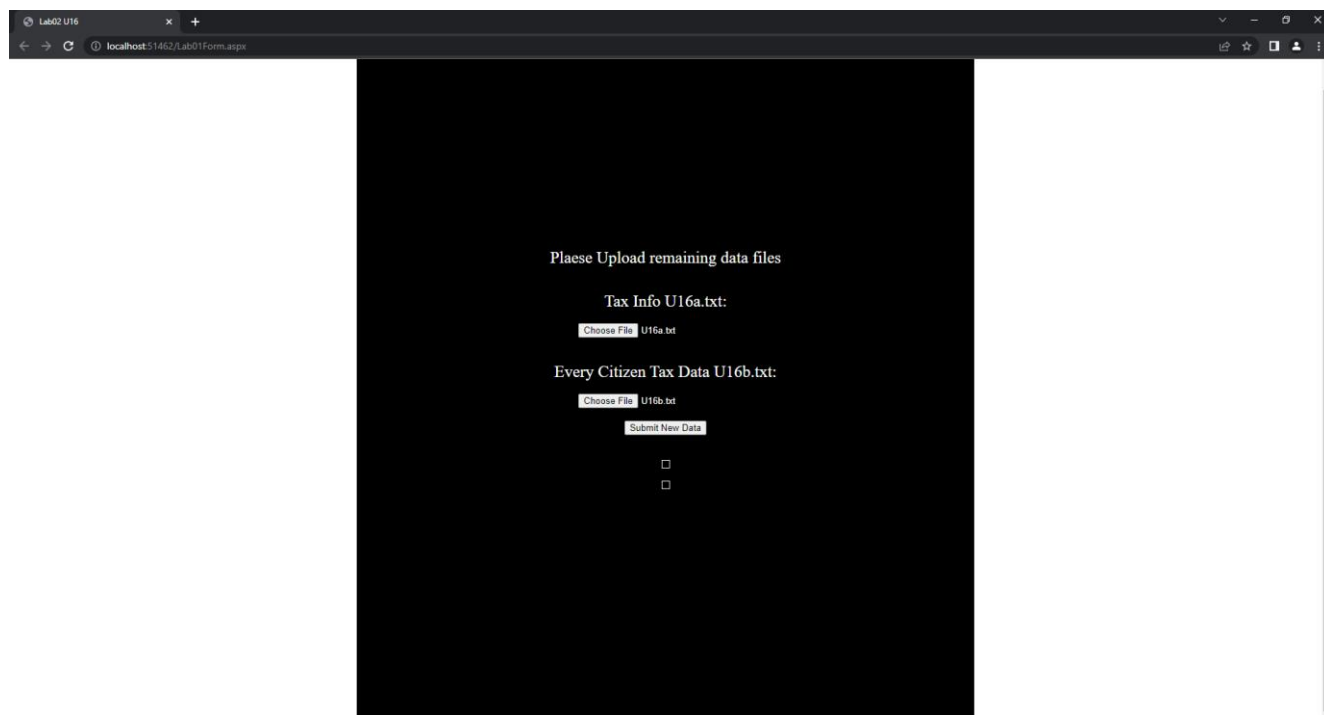
Įkeliam U16a.txt ir U16b.txt:

LAB02 U16

Tax Info U16a.txt:

[Choose File] No file chosen

Every Citizen Tax Data U16b.txt:

[Choose File] No file chosen

[Submit New Data]

U16a.txt Initial data:

Tax Code	Tax Company Name:	Price:
11	Elektra	0.12
21	Dujos	0.58
31	Benzinas	1.78
32	Diezelis	1.9

U16b.txt Initial data:

Last Name	First Name	Address	Month	Tax Code	Amount
vardas1	pavardė1	adresas1	1	22	28
vardas1	pavardė1	adresas1	5	22	20
vardas1	pavardė1	adresas1	1	32	100
vardas1	pavardė1	adresas1	2	32	97
vardas1	pavardė1	adresas1	3	32	63
vardas1	pavardė1	adresas1	2	22	25
vardas1	pavardė1	adresas1	3	22	29
vardasAA	pavardėAA	adresasAA	1	21	13
vardasAA	pavardėAA	adresasAA	2	21	84
vardasAA	pavardėAA	adresasAA	3	21	76
vardas1	pavardė1	adresas1	4	22	39
vardas2	pavardė2	adresas2	3	31	67
vardas2	pavardė2	adresas2	4	31	98
vardas2	pavardė0	adresas2	5	31	125
vardas0	pavardė0	adresas3	1	11	31
vardas1	pavardė1	adresas1	4	32	39
vardas1	pavardė1	adresas1	5	32	20
vardas1	pavardė1	adresas1	3	11	80
vardas1	pavardė1	adresas1	4	11	39
vardas1	pavardė1	adresas1	1	11	120
vardas1	pavardė1	adresas1	2	11	100
vardas1	pavardė1	adresas1	5	11	139
vardas2	pavardė2	adresas2	1	31	31
vardas2	pavardė2	adresas2	2	31	48
vardas0	pavardė0	adresas3	2	11	48
vardas0	pavardė0	adresas3	3	11	67
vardas0	pavardė0	adresas3	4	11	98
vardas0	pavardė0	adresas3	5	11	125
vardasAA	pavardėAA	adresasAA	4	21	8
vardasAA	pavardėAA	adresasAA	5	21	25

All Citizen taxes over the months

Last Name	First Name	Address	Tax Sum
vardasAA	pavardėAA	adresasAA	119.48
vardas0	pavardė0	adresas3	44.28
vardas2	pavardė2	adresas2	434.32
vardas2	pavardė0	adresas2	222.5
vardas1	pavardė1	adresas1	663.46

Average tax per citizen: 296.808
Total tax sum: 1484.04

Above Average Tax:

Last Name	First Name	Address	Tax Sum
vardas2	pavardė2	adresas2	434.32
vardas1	pavardė1	adresas1	663.46

No Filter provided

Tax Code:

Month:

[Submit]

Filtruojame duomenis pagal TaxCode: 11, Month: 1:

U16result.txt:

```
Initial Tax Company Data:

TaxCode         |TaxName        |       Price|
11              | Elektra       |        0.12|
21              | Dujos         |        0.58|
31              | Benzinas      |        1.78|
32              | Diezelis      |        1.90|

Initial Citizen Tax Data:

LastName         FirstName       |Address        |Month      |TaxCode        |TaxAmount|
 vardas1         pavardė1        | adresas1      |1          |22             |       28|
 vardas1         pavardė1        | adresas1      |5          |22             |       20|
 vardas1         pavardė1        | adresas1      |1          |32             |      100|
 vardas1         pavardė1        | adresas1      |2          |32             |       97|
 vardas1         pavardė1        | adresas1      |3          |32             |       63|
 vardas1         pavardė1        | adresas1      |2          |22             |       25|
 vardas1         pavardė1        | adresas1      |3          |22             |       29|
 vardasAA        pavardėAA       | adresasAA     |1          |21             |       13|
 vardasAA        pavardėAA       | adresasAA     |2          |21             |       84|
 vardasAA        pavardėAA       | adresasAA     |3          |21             |       76|
 vardas1         pavardė1        | adresas1      |4          |22             |       39|
 vardas2         pavardė2        | adresas2      |3          |31             |       67|
 vardas2         pavardė2        | adresas2      |4          |31             |       98|
 vardas2         pavardė0        | adresas2      |5          |31             |      125|
 vardas0         pavardė0        | adresas3      |1          |11             |       31|
 vardas1         pavardė1        | adresas1      |4          |32             |       39|
 vardas1         pavardė1        | adresas1      |5          |32             |       20|
 vardas1         pavardė1        | adresas1      |3          |11             |       80|
 vardas1         pavardė1        | adresas1      |4          |11             |       39|
 vardas1         pavardė1        | adresas1      |1          |11             |      120|
 vardas1         pavardė1        | adresas1      |2          |11             |      100|
 vardas1         pavardė1        | adresas1      |5          |11             |      139|
 vardas2         pavardė2        | adresas2      |1          |31             |       31|
 vardas2         pavardė2        | adresas2      |2          |31             |       48|
 vardas0         pavardė0        | adresas3      |2          |11             |       48|
 vardas0         pavardė0        | adresas3      |3          |11             |       67|
 vardas0         pavardė0        | adresas3      |4          |11             |       98|
 vardas0         pavardė0        | adresas3      |5          |11             |      125|
 vardasAA        pavardėAA       | adresasAA     |4          |21             |        8|
 vardasAA        pavardėAA       | adresasAA     |5          |21             |       25|

Tax Sum of all citizens:

LastName         FirstName       |Address        |TaxSum     |
 vardas1         pavardė1        | adresas1      |    663.46|
 vardasAA        pavardėAA       | adresasAA     |    119.48|
 vardas2         pavardė2        | adresas2      |    434.32|
 vardas2         pavardė0        | adresas2      |    222.50|
 vardas0         pavardė0        | adresas3      |     44.28|

Tax Sum of all citizens SORTED A-Z:

LastName         FirstName       |Address        |TaxSum     |
 vardasAA        pavardėAA       | adresasAA     |    119.48|
 vardas0         pavardė0        | adresas3      |     44.28|
 vardas2         pavardė2        | adresas2      |    434.32|
 vardas2         pavardė0        | adresas2      |    222.50|
 vardas1         pavardė1        | adresas1      |    663.46|

All Citizen TOTAL Tax Sum: 1484.04

Average Tax Sum: 296.81

Citizens who paid above average:

LastName         FirstName       |Address        |TaxSum     |
 vardas2         pavardė2        | adresas2      |    434.32|
 vardas1         pavardė1        | adresas1      |    663.46|

Citizens who paid TaxCode: "11" on Month: "1"

LastName         FirstName       |Address        |TaxSum     |
 vardas0         pavardė0        | adresas3      |     44.28|
 vardas1         pavardė1        | adresas1      |    663.46|
```

Pradiniai duomenys 2:

U16a.txt:

VAND; Vanduo; 0.07
KVND; Karštas vanduo; 0.20
LH20; Ledinis Vanduo; 0.10

```
U16b.txt:

Pavardauskis; Vardenis; Adresatas;Vasaris;Benzinas;28;
Pavardauskis; Vardenis; Adresatas;Vasaris;VAND;14;
Pavardauskis; Vardenis; Adresatas;Kovas;KVND;20;
Pavardauskis; Vardenis; Adresatas;Kovas;LH20;30;
Pavardauskis; Vardenis; Adresatas;Kovas;VAND;15;
Pavardauskis; Vardenis; Adresatas;Balandis;VAND;99;
Tomas; Tomukas; Tomo namas 1;Kovas;VAND;97;
Tomas; Tomukas; Tomo namas 1;Balandis;VAND;156;
Tomas; Tomukas; Tomo namas 1;Rugsėjis;VAND;20;
```

Rezultatai 2:
Vartotojo sąsaja:
Naudojant filtrus: TaxCode: Vand, Month: Kovas

LAB02 U16

Tax Info U16a.txt:

Choose File | No file chosen

Every Citizen Tax Data U16b.txt:

Choose File | No file chosen

Submit New Data

U16a.txt Initial data:

Tax Code	Tax Company Name:	Price:
VAND	Vanduo	0.07
KVND	Karštas vanduo	0.2
LH20	Ledinis Vanduo	0.1

U16b.txt Initial data:

Last Name	First Name	Address	Month	Tax Code	Amount
Vardenis	Pavardauskis	Adresatas	Vasaris	Benzinas	28
Vardenis	Pavardauskis	Adresatas	Vasaris	VAND	14
Vardenis	Pavardauskis	Adresatas	Kovas	KVND	20
Vardenis	Pavardauskis	Adresatas	Kovas	LH20	30
Vardenis	Pavardauskis	Adresatas	Kovas	VAND	15
Vardenis	Pavardauskis	Adresatas	Balandis	VAND	99
Tomukas	Tomas	Tomo namas 1	Kovas	VAND	97
Tomukas	Tomas	Tomo namas 1	Balandis	VAND	156
Tomukas	Tomas	Tomo namas 1	Rugsėjis	VAND	20

All Citizen taxes over the months

Last Name	First Name	Address	Tax Sum
Tomukas	Tomas	Tomo namas 1	19.11
Vardenis	Pavardauskis	Adresatas	15.96

U16Result.txt:

css/styles.css:

```css
body {
    color: white;
    background: white;
    padding: 0;
    margin: 0;
    display: flex;
```

```
        justify-content: center;
}

#body {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-align: center;
    padding: 15px 280px;
    background-color: black;
    min-height:100vh;
}

td {
    background-color: white;
}

table {
    border: 1px solid;
    border-color:white;
    padding: 5px;
}

td {
    color: black;
    padding:10px;
}

span {
    font-size: 1.5em;
}
```

Lab01Form.aspx:

3.8. Dėstytojo pastabos

Ačiū, puikus darbas, tik:

1. Testiniai variantai neturi tikslų.
2. Vienetų testavimas yra minimalus.
3. Vienetų testavimo rezultatai nėra įtraukti į ataskaitą.

4. Polimorfizmas ir išimčių valdymas (L4)

4.1. Darbo užduotis

„Buvo.csv".

U4_16. Biblioteka. Turite visų KTU bibliotekų padalinių duomenis. Pirmoje eilutėje – pavadinimas, antroje – adresas, trečioje – telefonas. Bibliotekoje galima rasti įvairių leidinių – knygų, žurnalų ir laikraščių. Sukurkite abstrakčią klasę „Publication" (savybės - pavadinimas, tipas, leidykla, išleidimo metai, puslapių skaičius, tiražas), kurią paveldės klasės "Book" (savybės - ISBN, autorius(-iai)), "Journal" (savybės – ISBN, numeris) ir "Newspaper" (savybės – data, numeris).

- Suskaičiuokite, kiek leidinių, senesnių nei 2 metų, yra kiekviename filiale. Rezultatą atspausdinkite ekrane.
- Sudarykite visų leidinių, kurių tipas yra „mokslinis" sąrašą. Visą informaciją apie juos atspausdinkite ekrane.
- Sudarykite ir surikiuokite nenaujų leidinių sąrašą, pateikdami pilną informaciją apie juos. Knyga yra nenauja, jei nuo išleidimo prabėgo daugiau, nei metai. Žurnalas yra nenaujas, jei nuo išleidimo prabėgo, daugiau nei mėnesis. Laikraštis yra nenaujas, jei nuo išleidimo prabėgo daugiau, nei savaitė. Knygas rikiuokite pagal išleidimo metus, žurnalus – pagal išleidimo metus ir mėnesius, o laikraščius – pagal išleidimo metus, mėnesius ir dienas. Rezultatus įrašykite į failą „Nenauji.csv".
- Sudarykite visų leidinių, kurių tiražas didesnis nei 10 000 vnt., sąrašą, surašykite šių leidinių pavadinimus ir tiražus į failą „PopuliarūsLeidiniai.csv".

4.2. Grafinės vartotojo sąsajos schema
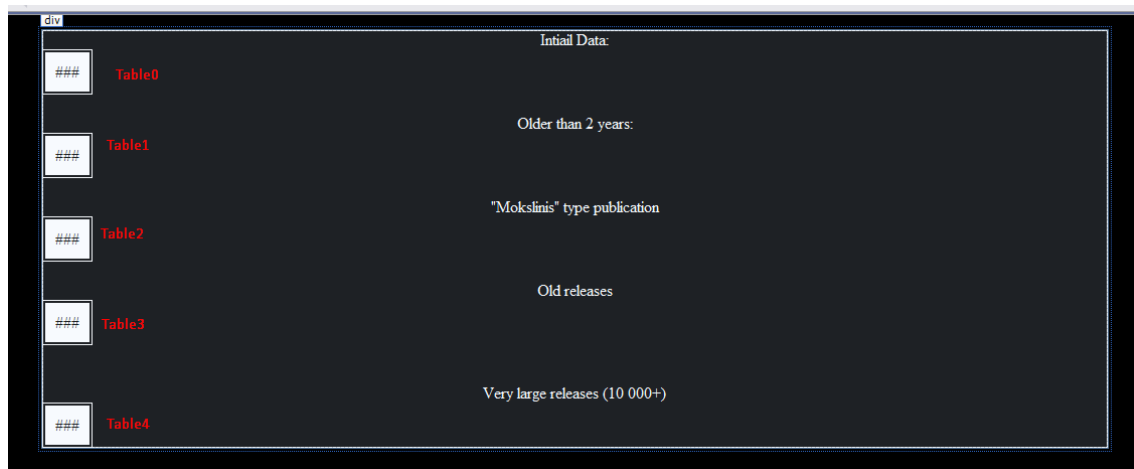
4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

Sąsajoje nebuvo panaudotų komponentų savybės keičiamos

4.4. Klasių diagrama
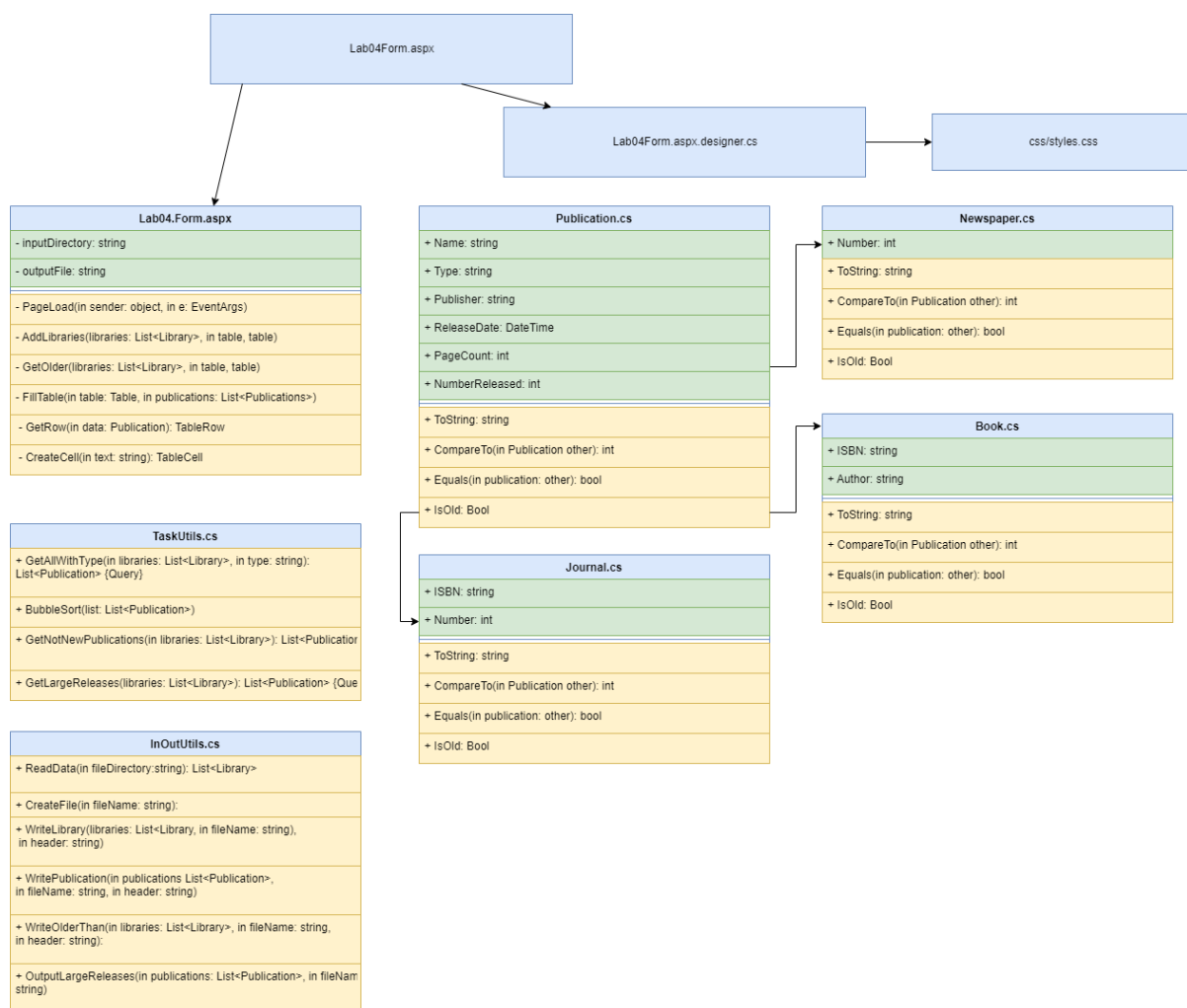
4.5. Programos vartotojo vadovas

Programa įjungiama. Užkrauną visus duomenų failus rastus App_Data/Data folderyję ir parodo vartotojui.

4.6. Programos tekstas

Publication.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab04
```

```csharp
{
    public abstract class Publication : IComparable<Publication>, IEquatable<Publication>
    {
        public string Name { get; set; }
        public string Type { get; set; }
        public string Publisher { get; set; }
        public DateTime ReleaseDate { get; set; }
        public int PageCount { get; set; }
        public int NumberReleased { get; set; }

        /// <summary>
        /// Custructor
        /// </summary>
        /// <param name="name"> Name of the Publication </param>
        /// <param name="type"> Type of the Publication </param>
        /// <param name="publisher"> Publisher of the publication </param>
        /// <param name="realeseYear"> Year in which the publication was released </param>
        /// <param name="pageCount"> Page Count </param>
        /// <param name="numberReleased"> Number of publications released</param>
        public Publication(string name, string type, string publisher, DateTime releaseDate, int pageCount, int numberReleased)
        {
            Name = name;
            Type = type;
            Publisher = publisher;
            ReleaseDate = releaseDate;
            PageCount = pageCount;
            NumberReleased = numberReleased;
        }

        public override string ToString()
        {
            return $"{Name,-25} |{Type,-15} |{Publisher,-25} |{ReleaseDate,-20} |{PageCount,10} |{NumberReleased,15} |";
        }

        /// <summary>
        /// CompareTo Implementation
        /// </summary>
        /// <param name="other"> Other object to compare to</param>
        /// <returns>integer</returns>
        public virtual int CompareTo(Publication other)
        {
            return ReleaseDate.CompareTo(other.ReleaseDate);
        }

        /// <summary>
        /// Equals Override
        /// </summary>
        /// <param name="other"> Other Publication to compare to</param>
        /// <returns> Boolean </returns>
        public virtual bool Equals(Publication other)
        {
            return Name == other.Name;
        }

        /// <summary>
        /// IsOld Function. Checks if the publication is old
        /// </summary>
        public virtual bool IsOld()
        {
            if(DateTime.Now.AddYears(-1).CompareTo(ReleaseDate) < 0)
                return true;
            return false;
        }
```

```
        }
}


Book.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab04
{
    public class Book : Publication, IComparable<Publication>, IEquatable<Publication>
    {
        public Book(string name, string type, string publisher, DateTime releaseDate, int
pageCount, int numberReleased, string isbn, string author) : base(name, type, publisher,
releaseDate, pageCount, numberReleased)
        {
            ISBN = isbn;
            Author = author;
        }

        public string ISBN { get; set; }
        public string Author { get; set; }

        /// <summary>
        /// To String Implementation
        /// </summary>
        /// <returns> string </returns>
        public override string ToString()
        {
            return base.ToString() + $"{ISBN,-15} |{Author,-20}";
        }

        /// <summary>
        /// CompareTo override
        /// </summary>
        /// <param name="other">  Other Publication to compare to </param>
        /// <returns> Publication </returns>
        public override int CompareTo(Publication other)
        {
            int comparison = ReleaseDate.Year.CompareTo(other.ReleaseDate.Year);
            return comparison;
        }

        /// <summary>
        /// iEquatable override
        /// </summary>
        /// <param name="other"> Other Publication to compare to</param>
        /// <returns> Bool </returns>
        public override bool Equals(Publication other)
        {
            return Name == other.Name;
        }

        /// <summary>
        /// IsOld Function. Checks if the publication is old
        /// </summary>
        public override bool IsOld()
        {
            if (DateTime.Now.AddYears(-1).CompareTo(ReleaseDate) > 0)
                return true;
            return false;
        }
    }
```

```
}


Journal.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab04
{
    public class Journal : Publication, IComparable<Publication>, IEquatable<Publication>
    {
        public Journal(string name, string type, string publisher, DateTime releaseDate,
int pageCount, int numberReleased, string isbn, int number) : base(name, type, publisher,
releaseDate, pageCount, numberReleased)
        {
            ISBN = isbn;
            Number = number;
        }

        public string ISBN { get; set; }

        public int Number { get; set; }

        /// <summary>
        /// To String Implementation
        /// </summary>
        /// <returns> string </returns>
        public override string ToString()
        {
            return base.ToString() + $"{ISBN,-15} |{Number,20}";
        }

        /// <summary>
        /// CompareTo override
        /// </summary>
        /// <param name="other">  Other Publication to compare to </param>
        /// <returns> Publication </returns>
        public override int CompareTo(Publication other)
        {
            int comparison = ReleaseDate.Year.CompareTo(other.ReleaseDate.Year);
            return comparison;
        }

        /// <summary>
        /// iEquatable override
        /// </summary>
        /// <param name="other"> Other Publication to compare to</param>
        /// <returns> Bool </returns>
        public override bool Equals(Publication other)
        {
            return Name == other.Name;
        }

        /// <summary>
        /// IsOld Function. Checks if the publication is old
        /// </summary>
        public override bool IsOld()
        {
            if (DateTime.Now.AddDays(-30).CompareTo(ReleaseDate) < 0)
                return true;
            return false;
        }
```

```
        }
}


Newspaper.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab04.App_Code
{
    public class Newspaper : Publication, IComparable<Publication>,
IEquatable<Publication>
    {
        public Newspaper(string name, string type, string publisher, DateTime
releaseDate, int pageCount, int numberReleased, int number) : base(name, type, publisher,
releaseDate, pageCount, numberReleased)
        {
            Number = number;
        }
        public int Number { get; set; }

        /// <summary>
        /// To String Implementation
        /// </summary>
        /// <returns> string </returns>
        public override string ToString()
        {
            return base.ToString() + $"{Number,15} |";
        }

        /// <summary>
        /// CompareTo override
        /// </summary>
        /// <param name="other">  Other Publication to compare to </param>
        /// <returns> Publication </returns>
        public override int CompareTo(Publication other)
        {
            int comparison = ReleaseDate.Year.CompareTo(other.ReleaseDate.Year);
            if (comparison != 0)
            {
                comparison = ReleaseDate.Month.CompareTo(other.ReleaseDate.Month);
            }
            return comparison;
        }

        /// <summary>
        /// iEquatable override
        /// </summary>
        /// <param name="other"> Other Publication to compare to</param>
        /// <returns> Bool </returns>
        public override bool Equals(Publication other)
        {
            return Name == other.Name;
        }

        /// <summary>
        /// IsOld Function. Checks if the publication is old
        /// </summary>
        public override bool IsOld()
        {
            if (DateTime.Now.AddDays(-7).CompareTo(ReleaseDate) > 0)
                return true;
            return false;
```

```
            }
        }
    }


Library.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab04.App_Code
{
    public class Library
    {
        public string Name { get; set; }
        public string Address { get; set; }
        public string PhoneNumber { get; set; }

        private List<Publication> publications;

        public int Count { get { return publications.Count;  } }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="name"> Name of the library</param>
        /// <param name="adress"> Adress of the library</param>
        /// <param name="phoneNumber"></param>
        public Library(string name, string address, string phoneNumber)
        {
            Name = name;
            Address = address;
            PhoneNumber = phoneNumber;
            publications = new List<Publication>();
        }

        /// <summary>
        /// Adds a publication
        /// </summary>
        /// <param name="publication"> Publication Data Type</param>
        public void Add(Publication publication)
        {
            publications.Add(publication);
        }

        /// <summary>
        /// Returns Items with selected index. If out of bounds error is thrown, returns
null-
        /// </summary>
        /// <param name="index"> index of the item to return</param>
        /// <returns></returns>
        public Publication Get(int index)
        {
            try
            {
                return publications[index];
            }
            catch
            {
                return null;
            }
        }

        /// <summary>
```

```csharp
        /// Gets The count of publications older than specified num ber
        /// </summary>
        /// <param name="year"> Years to check for older count</param>
        /// <returns> Older Count</returns>
        /// <exception cref="Exception"> Failed to compare the objects, Publication type
error</exception>
        public int OlderThanCount(int year)
        {
            int count = 0;
            try
            {
                foreach (Publication publication in publications)
                {
                    if (publication.ReleaseDate.CompareTo(DateTime.Now.AddYears(-year)) <
0)
                    {
                        count++;
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
ex.TargetSite, ex.Message, ex.Source));
            }

            return count;
        }
    }
}


TaskUtils.cs:


using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Lab04.App_Code
{
    public static class TaskUtils
    {
        /// <summary>
        /// Gets all Publications with specified Typer
        /// </summary>
        /// <param name="libraries"> List Library class object</param>
        /// <param name="type"> string, type of the object</param>
        /// <returns></returns>
        public static List<Publication> GetAllWithType(List<Library> libraries, string
type)
        {
            List<Publication> list = new List<Publication>();

            try
            {
                foreach (Library library in libraries)
                {
                    for (int i = 0; i < library.Count; i++)
                    {
                        Publication publication = library.Get(i);
                        if(publication.Type == type)
                            list.Add(publication);
                    }
                }
            }
```

```csharp
            catch (Exception ex)
            {
                throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
        ex.TargetSite, ex.Message, ex.Source));
            }

            return list;
        }

        public static void BubbleSort( this List<Publication> list)
        {
            for (int i = 0; i < list.Count - 1; i++)
            {
                for (int j = 0; j < list.Count - 1 - i; j++)
                {
                    if(list[j].CompareTo(list[j]) < 0)
                    {
                        Publication temp = list[j];
                        list[j] = list[j + 1];
                        list[j + 1] = temp;
                    }
                }
            }
        }

        /// <summary>
        /// Gets Not New Publications
        /// </summary>
        /// <param name="libraries"> All Library datas</param>
        /// <returns> List Publications of not new publicaitopns</returns>
        /// <exception cref="Exception"></exception>
        public static List<Publication> GetNotNewPublications(List<Library> libraries)
        {
            List<Publication> list = new List<Publication>();

            try
            {
                foreach (Library library in libraries)
                {
                    for (int i = 0; i < library.Count; i++)
                    {
                        Publication publication = library.Get(i);
                        if(publication.IsOld())
                        {
                            list.Add(publication);
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
        ex.TargetSite, ex.Message, ex.Source));
            }

            return list;
        }

        /// <summary>
        /// Gets LargeReleases
        /// </summary>
        /// <param name="libraries"> All Libraries to check data for</param>
        /// <returns></returns>
        /// <exception cref="Exception"></exception>
        public static List<Publication> GetLargeReleases(List<Library> libraries)
        {
            List<Publication> list = new List<Publication>();
```

```csharp
            try
            {
                foreach (Library library in libraries)
                {
                    for (int i = 0; i < library.Count; i++)
                    {
                        Publication pub = library.Get(i);
                        if(pub.NumberReleased >= 10000)
                            list.Add(pub);
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
ex.TargetSite, ex.Message, ex.Source));
            }

            return list;
        }
    }
}


InOutUtils.cs:


using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Web;

namespace Lab04.App_Code
{
    public static class InOutUtils
    {
        /// <summary>
        /// Reads Data from file directory
        /// </summary>
        /// <param name="fileDirectory"> file directory</param>
        /// <returns></returns>
        /// <exception cref="Exception"></exception>
        public static List<Library> ReadData(string fileDirectory)
        {
            List<Library> list = new List<Library>();
            foreach (string filePath in Directory.GetFiles(fileDirectory))
            {
                string[] lines = File.ReadAllLines(filePath);
                Library library = new Library(lines[0], lines[1], lines[2]);
                for (int i = 3; i < lines.Length; i++)
                {
                    string[] elements = lines[i].Split(';');
                    string name = elements[0];
                    string publisherType = elements[1];
                    string type = elements[2];
                    string publisher = elements[3];
                    DateTime releaseDate;
                    int pageCount;
                    int numberReleased;

                    try
                    {
                        releaseDate = DateTime.Parse(elements[4]);
                        pageCount = int.Parse(elements[5]);
                        numberReleased = int.Parse(elements[6]);
```

```csharp
                }
                catch (Exception ex)
                {
                    throw new Exception(string.Format(" Method {0}, Message {1},
Source {2}", ex.TargetSite, ex.Message, ex.Source));
                }
                try
                {
                    switch (publisherType)
                    {
                        case "Book":
                            string isbnBook = elements[7];
                            string author = elements[8];
                            Book book = new Book(name, type, publisher, releaseDate,
pageCount, numberReleased, isbnBook, author);
                            library.Add(book);
                            break;

                        case "Journal":
                            string isbnJournal = elements[7];
                            int number = int.Parse(elements[8]);
                            Journal journal = new Journal(name, type, publisher,
releaseDate, pageCount, numberReleased, isbnJournal, number);
                            library.Add(journal);
                            break;

                        case "Newspaper":
                            int numberPaper = int.Parse(elements[7]);
                            Newspaper paper = new Newspaper(name, type, publisher,
releaseDate, pageCount, numberReleased, numberPaper);
                            library.Add(paper);
                            break;
                    }
                }
                catch (Exception ex)
                {
                    throw new Exception(string.Format(" Method {0}, Message {1},
Source {2}", ex.TargetSite, ex.Message, ex.Source));
                }
            }
            list.Add(library);
        }
        return list;
    }

    public static void CreateFile(string fileName)
    {
        new StreamWriter(fileName).Close();
    }
    public static void WriteLibrary(List<Library> libraries, string fileName, string
header)
    {
        try
        {
            using (StreamWriter sw = new StreamWriter(fileName, append: true))
            {
                sw.WriteLine(header);
                sw.WriteLine();
                foreach (Library library in libraries)
                {
                    sw.WriteLine(library.Name);
                    sw.WriteLine(library.Address);
                    sw.WriteLine(library.PhoneNumber);
                    sw.WriteLine(new String('-', 100));
                    sw.WriteLine($"{"Name",-25} |{"Type",-15} |{"Publisher",-25}
|{"ReleaseDate",-20} |{"PageCount",-10} |{"NumberReleased",-15} |{"ISBN/Number",-15}
|{"Author/Number",-20}");
```

```csharp
                        for (int i = 0; i < library.Count; i++)
                        {
                            sw.WriteLine(library.Get(i));
                        }

                        sw.WriteLine();
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
ex.TargetSite, ex.Message, ex.Source));
            }
        }

        /// <summary>
        /// Writes all publications from List Publicatrion object
        /// </summary>
        /// <param name="publications"> List Publication object</param>
        /// <param name="fileName"> Output file</param>
        /// <param name="header"> Headerd</param>
        public static void WritePublication(List<Publication> publications, string
fileName, string header)
        {
            using (StreamWriter sw = new StreamWriter(fileName, append: true))
            {
                sw.WriteLine(header);
                sw.WriteLine();
                sw.WriteLine($"{"Name",-25} |{"Type",-15} |{"Publisher",-25}
|{"ReleaseDate",-20} |{"PageCount",-10} |{"NumberReleased",-15} |{"ISBN/Number",-15}|
{"Author/Number",-20}");
                foreach (Publication publication in publications)
                {
                    sw.WriteLine(publication);
                }
                sw.WriteLine();
            }
        }

        /// <summary>
        /// Writes Older than 2 years List to txt file
        /// </summary>
        /// <param name="libraries"> List Library file </param>
        /// <param name="fileName"> Filename to input file </param>
        /// <param name="header"> Header of the text</param>
        public static void WriteOlderThan(List<Library> libraries, string fileName,
string header)
        {
            using (StreamWriter sw = new StreamWriter(fileName, append: true))
            {
                sw.WriteLine(header);
                sw.WriteLine();
                foreach (Library library in libraries)
                {
                    sw.WriteLine($"{library.Name} has {library.OlderThanCount(2)}
publications older than 2 years");
                    sw.WriteLine();
                }
                sw.WriteLine();
            }
        }

        public static void OutputLargeReleases(List<Publication> publications, string
fileName)
        {
            using (StreamWriter sw = new StreamWriter(fileName))
```

```csharp
                {
                    sw.WriteLine("Name;Release Number");
                    foreach (Publication publication in publications)
                    {
                        sw.WriteLine($"{publication.Name};{publication.NumberReleased}");
                    }
                }
            }
        }
    }
}
```

Lab04Form.aspx:

```asp
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Lab04Form.aspx.cs"
Inherits="Lab04.Lab04Form" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <link rel="stylesheet" runat="server" media="screen" href="~/css/stylesheet.css" />
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Intiail Data:<br />
            <asp:Table ID="Table0" runat="server">
            </asp:Table>
            <br />
            Older than 2 years:<br />
            <asp:Table ID="Table1" runat="server">
            </asp:Table>
            <br />
            &quot;Mokslinis&quot; type publication<br />
            <asp:Table ID="Table2" runat="server">
            </asp:Table>
            <br />
            Old releases<br />
            <asp:Table ID="Table3" runat="server">
            </asp:Table>
            <br />
            <br />
            Very large releases (10 000+)<br />
            <asp:Table ID="Table4" runat="server">
            </asp:Table>
        </div>
    </form>
</body>
</html>
```

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using Lab04.App_Code;

namespace Lab04
{
    public partial class Lab04Form : System.Web.UI.Page
    {
```

```csharp
        private string inputDirectory = @"App_Data/Data1";
        private string outputFile = @"App_Data/Output.txt";
        protected void Page_Load(object sender, EventArgs e)
        {
            // Starting Data
            List<Library> libraries =
InOutUtils.ReadData(Server.MapPath(inputDirectory));
            InOutUtils.CreateFile(Server.MapPath(outputFile));
            InOutUtils.WriteLibrary(libraries, Server.MapPath(outputFile), "Initial
Data:");
            AddLibraries(libraries, Table0);

            // Task 1
            InOutUtils.WriteOlderThan(libraries, Server.MapPath(outputFile), "Older Than
2 Years Publications in specific libraries");
            GetOlder(Table1, libraries);

            // Task 2
            List<Publication> withSelectedType = TaskUtils.GetAllWithType(libraries,
"Mokslinis");
            InOutUtils.WritePublication(withSelectedType, Server.MapPath(outputFile),
"\"Mokslinis\" type publication:");
            FillTable(Table2, withSelectedType);

            // Task 3
            List<Publication> notNewPublications =
TaskUtils.GetNotNewPublications(libraries);
            notNewPublications.BubbleSort();
            InOutUtils.WritePublication(notNewPublications, Server.MapPath(outputFile),
"Old Publications");
            FillTable(Table3, notNewPublications);

            // Task 4
            List<Publication> largePublications = TaskUtils.GetLargeReleases(libraries);
            InOutUtils.OutputLargeReleases(largePublications,
Server.MapPath("App_Data/PopuliarusLeidiniai.csv"));
            FillTable(Table4, largePublications);

        }

        public void AddLibraries(List<Library> libraries, Table table)
        {
            try
            {
                foreach (Library library in libraries)
                {
                    TableRow row = new TableRow();
                    row.Cells.Add(CreateCell(library.Name));
                    row.Cells.Add(CreateCell(library.Address));
                    row.Cells.Add(CreateCell(library.PhoneNumber));
                    table.Rows.Add(row);
                    row = new TableRow();
                    row.Cells.Add(CreateCell("Name"));
                    row.Cells.Add(CreateCell("Type"));
                    row.Cells.Add(CreateCell("Publisher"));
                    row.Cells.Add(CreateCell("ReleaseDate"));
                    row.Cells.Add(CreateCell("PageCount"));
                    row.Cells.Add(CreateCell("NumberReleased"));
                    row.Cells.Add(CreateCell("ISBN/Number"));
                    row.Cells.Add(CreateCell("Author/Number"));
                    table.Rows.Add(row);
                    for (int i = 0; i < library.Count; i++)
                    {
                        Publication publication = library.Get(i);
                        table.Rows.Add(GetRow(publication));
                    }
                }
```

```csharp
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
ex.TargetSite, ex.Message, ex.Source));
        }
    }

    /// <summary>
    /// Gets Older than 2 years publications
    /// </summary>
    /// <param name="table"> Input Table</param>
    /// <param name="libraries">List<Library> libraries</Library></param>
    /// <exception cref="Exception"></exception>
    public void GetOlder(Table table, List<Library> libraries)
    {
        try
        {
            TableRow row = new TableRow();
            foreach (Library library in libraries)
            {
                TableRow tempRow = new TableRow();
                tempRow.Cells.Add(CreateCell($"{library.Name} has
{library.OlderThanCount(2)} publications older than 2 years"));
                table.Rows.Add(tempRow);
            }
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
ex.TargetSite, ex.Message, ex.Source));
        }
    }

    /// <summary>
    /// Fills Table
    /// </summary>
    /// <param name="table"></param>
    /// <param name="publications"></param>
    /// <exception cref="Exception"></exception>
    public void FillTable(Table table, List<Publication> publications)
    {
        try
        {
            TableRow row = new TableRow();
            row.Cells.Add(CreateCell("Name"));
            row.Cells.Add(CreateCell("Type"));
            row.Cells.Add(CreateCell("Publisher"));
            row.Cells.Add(CreateCell("ReleaseDate"));
            row.Cells.Add(CreateCell("PageCount"));
            row.Cells.Add(CreateCell("NumberReleased"));
            row.Cells.Add(CreateCell("ISBN/Number"));
            row.Cells.Add(CreateCell("Author/Number"));
            table.Rows.Add(row);

            foreach (Publication publication in publications)
            {
                table.Rows.Add(GetRow(publication));
            }
        }
        catch (Exception ex)
        {
            throw new Exception(string.Format(" Method {0}, Message {1}, Source {2}",
ex.TargetSite, ex.Message, ex.Source));
        }
    }
```

```csharp
        /// <summary>
        /// Creates TableRow from Book object
        /// </summary>
        /// <param name="data">Book object</param>
        /// <returns>TableRow object</returns>
        public TableRow GetRow(Publication data)
        {
            TableRow row = new TableRow();
            string[] elements = data.ToString().Split('|');
            foreach (string element in elements)
                row.Cells.Add(CreateCell(element.Trim()));

            return row;
        }


        /// <summary>
        /// Creates TableCell from text to speed up TableCell creation
        /// </summary>
        /// <param name="text">string text to add to the table cell</param>
        /// <returns>TableCell class object</returns>
        protected static TableCell CreateCell(string text)
        {
            TableCell cell = new TableCell();
            cell.Text = text;
            return cell;
        }


    }
}
```

4.7. Pradiniai duomenys ir rezultatai

Pradiniai Duomenys 1:

KTU.txt:

Kauno Biblioteka
Kauno adresato g. 1
8675946855
Knyga 1;Book;Mokslinis;Kauno knygos;5/3/2020;125;1000;123546;Visų Rašytojas;
Laikraštis 1918;Newspaper;Istorinis;Istoriniai laikraščiai;5/3/2022;25;200;124
Knyga 3;Book;Pramoginis;Stumbro Lapai;5/5/2022;253;250;111111;Rašytojas 1
Žurnalas Pasaulis;Journal;Geografinis;Pasaulio
žemėlapiai;1/1/2000;64;15000;555555;1
Žurnalas Mokslas;Journal;Mokslinis;Mokslu susidomėja;5/3/2022;32;3000;444444;45

VDU.txt:

VDU Biblioteka
Mickevičiaus g. 10
86666666666
Laikraštis Įvairovė;Newspaper;Viskas;Leidykla Supratimas;10/10/2018;15;20000;15
Žurnalas Pasaulis;Journal;Geografinis;Pasaulio
žemėlapiai;1/1/2000;64;1500;123456789;25
Žurnalas
Informatika;Journal;Mokslinis;Informatikai;9/8/1995;120;25000;987654321;13

Rezultatai 1:

Vartotojo sąsaja:

Intial Data:

Kauno Biblioteka	Kauno adresato g. 1	8675946855					
Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Knyga 1	Mokslinis	Kauno knygos	5/3/2020 12:00:00 AM	125	1000	123546	Visų Rašytojas
Laikraštis 1918	Istorinis	Istoriniai laikraščiai	5/3/2022 12:00:00 AM	25	200	124	
Knyga 3	Pramoginis	Stumbro Lapai	5/5/2022 12:00:00 AM	253	250	111111	Rašytojas 1
Žurnalas Pasaulis	Geografinis	Pasaulio žemėlapiai	1/1/2000 12:00:00 AM	64	15000	555555	1
Žurnalas Mokslas	Mokslinis	Mokslu susidomėja	5/3/2022 12:00:00 AM	32	3000	444444	45
VDU Biblioteka	Mickevičiaus g. 10	86666666666					
Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Laikraštis Įvairovė	Viskas	Leidykla Supratimas	10/10/2018 12:00:00 AM	15	20000	15	
Žurnalas Pasaulis	Geografinis	Pasaulio žemėlapiai	1/1/2000 12:00:00 AM	64	1500	123456789	25
Žurnalas Informatika	Mokslinis	Informatikai	9/8/1995 12:00:00 AM	120	25000	987654321	13

Older than 2 years:

Kauno Biblioteka has 2 publications older than 2 years
VDU Biblioteka has 3 publications older than 2 years

"Mokslinis" type publication

Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Knyga 1	Mokslinis	Kauno knygos	5/3/2020 12:00:00 AM	125	1000	123546	Visų Rašytojas
Žurnalas Mokslas	Mokslinis	Mokslu susidomėja	5/3/2022 12:00:00 AM	32	3000	444444	45
Žurnalas Informatika	Mokslinis	Informatikai	9/8/1995 12:00:00 AM	120	25000	987654321	13

Old releases

Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Knyga 1	Mokslinis	Kauno knygos	5/3/2020 12:00:00 AM	125	1000	123546	Visų Rašytojas
Žurnalas Mokslas	Mokslinis	Mokslu susidomėja	5/3/2022 12:00:00 AM	32	3000	444444	45
Laikraštis Įvairovė	Viskas	Leidykla Supratimas	10/10/2018 12:00:00 AM	15	20000	15	

Very large releases (10 000+)

Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Žurnalas Pasaulis	Geografinis	Pasaulio žemėlapiai	1/1/2000 12:00:00 AM	64	15000	555555	1
Laikraštis Įvairovė	Viskas	Leidykla Supratimas	10/10/2018 12:00:00 AM	15	20000	15	
Žurnalas Informatika	Mokslinis	Informatikai	9/8/1995 12:00:00 AM	120	25000	987654321	13

Output.txt:

```
Initial Data:

Kauno Biblioteka
Kauno adresato g. 1
8675946855
----------------------------------------------------------------------------------------------------------------------
Name                    |Type         |Publisher              |ReleaseDate            |PageCount  |NumberReleased |ISBN/Number    |Author/Number
Knyga 1                 |Mokslinis    |Kauno knygos           |5/3/2020 12:00:00 AM |      125 |          1000 |123546         |Visų Rašytojas
Laikraštis 1918         |Istorinis    |Istoriniai laikraščiai |5/3/2022 12:00:00 AM |       25 |           200 |           124 |
Knyga 3                 |Pramoginis   |Stumbro Lapai          |5/5/2022 12:00:00 AM |      253 |           250 |111111         |Rašytojas 1
Žurnalas Pasaulis       |Geografinis  |Pasaulio žemėlapiai    |1/1/2000 12:00:00 AM |       64 |         15000 |555555         |             1
Žurnalas Mokslas        |Mokslinis    |Mokslu susidomėja      |5/3/2022 12:00:00 AM |       32 |          3000 |444444         |            45

VDU Biblioteka
Mickevičiaus g. 10
86666666666
----------------------------------------------------------------------------------------------------------------------
Name                    |Type         |Publisher              |ReleaseDate             |PageCount  |NumberReleased |ISBN/Number    |Author/Number
Laikraštis Įvairovė      |Viskas       |Leidykla Supratimas    |10/10/2018 12:00:00 AM |       15 |         20000 |            15 |
Žurnalas Pasaulis       |Geografinis  |Pasaulio žemėlapiai    |1/1/2000 12:00:00 AM   |       64 |          1500 |123456789      |            25
Žurnalas Informatika    |Mokslinis    |Informatikai           |9/8/1995 12:00:00 AM   |      120 |         25000 |987654321      |            13

Older Than 2 Years Publications in specific libraries
|
Kauno Biblioteka has 2 publications older than 2 years

VDU Biblioteka has 3 publications older than 2 years


"Mokslinis" type publication:

Name                    |Type         |Publisher              |ReleaseDate            |PageCount  |NumberReleased |ISBN/Number    | Author/Number
Knyga 1                 |Mokslinis    |Kauno knygos           |5/3/2020 12:00:00 AM |      125 |          1000 |123546         |Visų Rašytojas
Žurnalas Mokslas        |Mokslinis    |Mokslu susidomėja      |5/3/2022 12:00:00 AM |       32 |          3000 |444444         |            45
Žurnalas Informatika    |Mokslinis    |Informatikai           |9/8/1995 12:00:00 AM |      120 |         25000 |987654321      |            13

Old Publications

Name                    |Type         |Publisher              |ReleaseDate            |PageCount  |NumberReleased |ISBN/Number    | Author/Number
Knyga 1                 |Mokslinis    |Kauno knygos           |5/3/2020 12:00:00 AM |      125 |          1000 |123546         |Visų Rašytojas
Žurnalas Mokslas        |Mokslinis    |Mokslu susidomėja      |5/3/2022 12:00:00 AM |       32 |          3000 |444444         |            45
Laikraštis Įvairovė      |Viskas       |Leidykla Supratimas    |10/10/2018 12:00:00 AM |       15 |         20000 |            15 |
```

PopuliarusLeidiniai.csv:

	A	B	C	D	E	F	G	H	I
1	Name	Release Number							
2	Žurnalas Pasaulis	15000							
3	Laikraštis Įvairovė	20000							
4	Žurnalas Informatika	25000							
5									
6									
7									

Duomenys 2:

SMK.txt:

SMK Biblioteka
Aleksoto g. 26
8656546984896
Žurnalas Draugas;Journal;Psichologija;Psichologijos
darbai;1/1/2025;25;100;123456789;25
Žurnalas
Mokslinčius;Journal;Mokslinis;Mokslininkai;6/7/2021;120;25000;987654321;13

VGTU.txt:

VGTU Biblioteka
Mickevičiaus g. 356
86666111111

```
Laikraštis Tikslas;Newspaper;Mokslinis;Leidykla Tiksliukai;5/10/2021;25;200;30
Laikraštis Abstraktas;Newspaper;Filosofija;Leidykla
Filosofantas;1/1/2022;15;10000;15

VU.txt:

VU Mažoji Biblioteka
Vilniaus g. 20
8764564694
Katekizmas;Book;Mokslinis;Kauno knygos;5/3/1547;79;27;123546;Mažvydas;
Raganius;Book;Pramoginis;Andrejus Sapovskis;5/5/1991;423;250000;111111;Fantastikos
Rašytojas
```

Rezultatai 2:

Vartotojo sąsaja:

Intiail Data:

SMK Biblioteka	Aleksoto g. 26	8656546984896					
Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Žurnalas Draugas	Psichologija	Psichologijos darbai	1/1/2025 12:00:00 AM	25	100	123456789	25
Žurnalas Mokslinčius	Mokslinis	Mokslininkai	6/7/2021 12:00:00 AM	120	25000	987654321	13
VGTU Biblioteka	Mickevičiaus g. 356	86666111111					
Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Laikraštis Tikslas	Mokslinis	Leidykla Tiksliukai	5/10/2021 12:00:00 AM	25	200	30	
Laikraštis Abstraktas	Filosofija	Leidykla Filosofantas	1/1/2022 12:00:00 AM	15	10000	15	
VU Mažoji Biblioteka	Vilniaus g. 20	8764564694					
Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Katekizmas	Mokslinis	Kauno knygos	5/3/1547 12:00:00 AM	79	27	123546	Mažvydas
Raganius	Pramoginis	Andrejus Sapovskis	5/5/1991 12:00:00 AM	423	250000	111111	Fantastikos Rašytojas

Older than 2 years:

SMK Biblioteka has 0 publications older than 2 years
VGTU Biblioteka has 0 publications older than 2 years
VU Mažoji Biblioteka has 2 publications older than 2 years

"Mokslinis" type publication

Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Žurnalas Mokslinčius	Mokslinis	Mokslininkai	6/7/2021 12:00:00 AM	120	25000	987654321	13
Laikraštis Tikslas	Mokslinis	Leidykla Tiksliukai	5/10/2021 12:00:00 AM	25	200	30	
Katekizmas	Mokslinis	Kauno knygos	5/3/1547 12:00:00 AM	79	27	123546	Mažvydas

Old releases

Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Žurnalas Draugas	Psichologija	Psichologijos darbai	1/1/2025 12:00:00 AM	25	100	123456789	25
Laikraštis Tikslas	Mokslinis	Leidykla Tiksliukai	5/10/2021 12:00:00 AM	25	200	30	
Laikraštis Abstraktas	Filosofija	Leidykla Filosofantas	1/1/2022 12:00:00 AM	15	10000	15	
Katekizmas	Mokslinis	Kauno knygos	5/3/1547 12:00:00 AM	79	27	123546	Mažvydas
Raganius	Pramoginis	Andrejus Sapovskis	5/5/1991 12:00:00 AM	423	250000	111111	Fantastikos Rašytojas

Very large releases (10 000+)

Name	Type	Publisher	ReleaseDate	PageCount	NumberReleased	ISBN/Number	Author/Number
Žurnalas Mokslinčius	Mokslinis	Mokslininkai	6/7/2021 12:00:00 AM	120	25000	987654321	13
Laikraštis Abstraktas	Filosofija	Leidykla Filosofantas	1/1/2022 12:00:00 AM	15	10000	15	
Raganius	Pramoginis	Andrejus Sapovskis	5/5/1991 12:00:00 AM	423	250000	111111	Fantastikos Rašytojas

Output.txt:

```
Initial Data:

SMK Biblioteka
Aleksoto g. 26
8656546984896
---------------------------------------------------------------------
Name               |Type         |Publisher            |ReleaseDate          |PageCount  |NumberReleased  |ISBN/Number  |Author/Number
Žurnalas Draugas   |Psichologija |Psichologijos darbai |1/1/2025 12:00:00 AM |      25 |            100 |123456789    |              25
Žurnalas Mokslinčius|Mokslinis   |Mokslininkai         |6/7/2021 12:00:00 AM |     120 |          25000 |987654321    |              13

VGTU Biblioteka
Mickevičiaus g. 356
86666111111
---------------------------------------------------------------------
Name               |Type         |Publisher            |ReleaseDate          |PageCount  |NumberReleased  |ISBN/Number  |Author/Number
Laikraštis Tikslas |Mokslinis    |Leidykla Tiksliukai  |5/10/2021 12:00:00 AM|      25 |            200 |        30 |
Laikraštis Abstraktas|Filosofija |Leidykla Filosofantas|1/1/2022 12:00:00 AM |      15 |          10000 |        15 |

VU Mažoji Biblioteka
Vilniaus g. 20
8764564694
---------------------------------------------------------------------
Name               |Type         |Publisher            |ReleaseDate          |PageCount  |NumberReleased  |ISBN/Number  |Author/Number
Katekizmas         |Mokslinis    |Kauno knygos         |5/3/1547 12:00:00 AM |      79 |             27 |123546       |Mažvydas
Raganius           |Pramoginis   |Andrejus Sapovskis   |5/5/1991 12:00:00 AM |     423 |         250000 |111111       |Fantastikos Rašytojas

Older Than 2 Years Publications in specific libraries

SMK Biblioteka has 0 publications older than 2 years

VGTU Biblioteka has 0 publications older than 2 years

VU Mažoji Biblioteka has 2 publications older than 2 years


"Mokslinis" type publication:

Name               |Type         |Publisher            |ReleaseDate          |PageCount  |NumberReleased  |ISBN/Number  | Author/Number
Žurnalas Mokslinčius|Mokslinis   |Mokslininkai         |6/7/2021 12:00:00 AM |     120 |          25000 |987654321    |              13
Laikraštis Tikslas |Mokslinis    |Leidykla Tiksliukai  |5/10/2021 12:00:00 AM|      25 |            200 |        30 |
Katekizmas         |Mokslinis    |Kauno knygos         |5/3/1547 12:00:00 AM |      79 |             27 |123546       |Mažvydas

Old Publications

Name               |Type         |Publisher            |ReleaseDate          |PageCount  |NumberReleased  |ISBN/Number  | Author/Number
Žurnalas Draugas   |Psichologija |Psichologijos darbai |1/1/2025 12:00:00 AM |      25 |            100 |123456789    |              25
Laikraštis Tikslas |Mokslinis    |Leidykla Tiksliukai  |5/10/2021 12:00:00 AM|      25 |            200 |        30 |
Laikraštis Abstraktas|Filosofija |Leidykla Filosofantas|1/1/2022 12:00:00 AM |      15 |          10000 |        15 |
Katekizmas         |Mokslinis    |Kauno knygos         |5/3/1547 12:00:00 AM |      79 |             27 |123546       |Mažvydas
Raganius           |Pramoginis   |Andrejus Sapovskis   |5/5/1991 12:00:00 AM |     423 |         250000 |111111       |Fantastikos Rašytojas
```

PopuliarusLeidiniai.csv:

	A	B	C	D	E	F	G	H
1	Name	Release Number						
2	ÅŽurnalas Mokslinčius	25000						
3	Laikraštis Abstraktas	10000						
4	Raganius	250000						
5								
6								
7								
8								
9								
10								

4.8. Dėstytojo pastabos

5. Deklaratyvusis programavimas (L5)

5.1. Darbo užduotis

5.2. Grafinės vartotojo sąsajos schema

5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

5.4. Klasių diagrama

5.5. Programos vartotojo vadovas

5.6. Programos tekstas

5.7. Pradiniai duomenys ir rezultatai

5.8. Dėstytojo pastabos