Kauno technologijos universitetas
Informatikos fakultetas

Objektinis programavimas 2 (P175B123)

Laboratorinių darbų ataskaita

Dovydas Klimas IFD-1/1

Studentas

Doc. Lina Narbutaitė

Dėstytoja

Kaunas 2022

TURINYS

1. Rekursija (L1) 4

1.1. Darbo užduotis 4

1.2. Grafinės vartotojo sąsajos schema 4

1.3. Sąsajoje panaudotų komponentų keičiamos savybės 4

1.4. Klasių diagrama 5

1.5. Programos vartotojo vadovas 5

1.6. Programos tekstas 5

1.7. Pradiniai duomenys ir rezultatai 9

1.8. Dėstytojo pastabos 10

2. Dinaminis atminties valdymas (L2) 11

2.1. Darbo užduotis 11

2.2. Grafinės vartotojo sąsajos schema 11

2.3. Sąsajoje panaudotų komponentų keičiamos savybės 11

2.4. Klasių diagrama 11

2.5. Programos vartotojo vadovas 11

2.6. Programos tekstas 11

2.7. Pradiniai duomenys ir rezultatai 11

2.8. Dėstytojo pastabos 12

3. Bendrinės klasės ir testavimas (L3) 13

3.1. Darbo užduotis 13

3.2. Grafinės vartotojo sąsajos schema 13

3.3. Sąsajoje panaudotų komponentų keičiamos savybės 13

3.4. Klasių diagrama 13

3.5. Programos vartotojo vadovas 13

3.6. Programos tekstas 13

3.7. Pradiniai duomenys ir rezultatai 13

3.8. Dėstytojo pastabos 14

4. Polimorfizmas ir išimčių valdymas (L4) 15

4.1. Darbo užduotis 15

4.2. Grafinės vartotojo sąsajos schema 15

4.3. Sąsajoje panaudotų komponentų keičiamos savybės 15

4.4. Klasių diagrama 15

4.5. Programos vartotojo vadovas 15

4.6. Programos tekstas 15

4.7. Pradiniai duomenys ir rezultatai 15

4.8. Dėstytojo pastabos 16

5. Deklaratyvusis programavimas (L5) 17

5.1. Darbo užduotis 17

5.2. Grafinės vartotojo sąsajos schema 17

5.3. Sąsajoje panaudotų komponentų keičiamos savybės 17

5.4. Klasių diagrama 17

5.5. Programos vartotojo vadovas 17

5.6. Programos tekstas 17

5.7. Pradiniai duomenys ir rezultatai 17

5.8. Dėstytojo pastabos 18

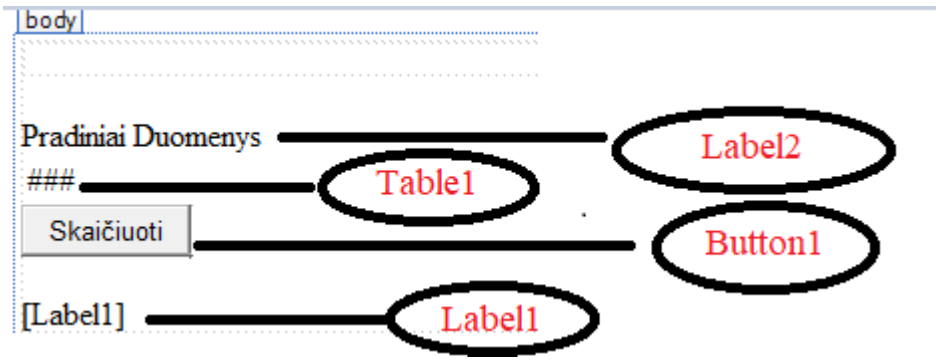
1. Rekursija (L1)

1.1. Darbo užduotis

LD_9. Kurjeris.
Studentas įsidarbino kurjeriu firmoje, turinčioje n punktų, į kuriuos jis laiku privalo pristatyti gaminius. Kurjerio algos dydis yra pastovus ir nepriklauso nuo kelionės trukmės. Norėdamas įvertinti bendrą kelionių kainą, studentas apskaičiavo kelionės kainą tarp bet kurių dviejų punktų. Užduotis – surasti tokį kelionės maršrutą, kuris praeitų tik po vieną kartą pro kiekvieną punktą ir kurio bendra punktų apvažiavimo kaina būtų mažiausia, t.y. kad studentui daugiau liktų pelno. Studento kelionės pradžia visą laiką yra pirmasis punktas. Ten jis ir turi grįžti.
Duomenys surašyti tekstiniame faile `'U3.txt'`. Pirmoje failo eilutėje parašytas sveikasis skaičius `n` (`5 ≤ n ≤ 50`), nurodantis kvadratinės matricos dydį. Toliau eilutėmis užrašyta matrica `K(n,n)`, kurioje išvardinta kelionių kaina tarp atskirų punktų, kur `K[i,j]=K[j,i]` ir reiškia kelionės kaštus tarp punktų `i` ir `j`. Išveskite **rezultatus**, kuriuose būtų išvardinti punktai jų aplankymo eilės tvarka ir kelionės kaina.

U3.txt	Rezultatų pavyzdys
5 0 1 3 4 2 1 0 4 2 6 3 4 0 7 1 4 2 7 0 7 2 6 1 7 0	1, 5, 3, 2, 4, 1 = 13

1.2. Grafinės vartotojo sąsajos schema



1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label2	Text	Pradiniai Duomenys
Button1	Text	Skaičiuoti

1.4. Klasių diagrama

1.5. Programos vartotojo vadovas

Paleidus programą, iš failo „U3.txt“ yra nuskaitomi duomenys ir jie patalpinami į sąrąšą. Paspaudus mygtuką skaičiuoti, programa rekursiškai suranda pigiausią maršrutą tarp pirminio ir kito punkto. Programai suradus pigiausią kainą tarp dviejų skirtingų punktų (Kurjeris visada nuvažiavęs į punktą, privalo sugrįžti į pradinį punktą), kviečiamas metodas susumuoja kainas. Sumą suranda sudedant pigiausių punktų kainas. Tai atlikus programa duomenis patalpina į naudotojo sąsają ir „rezultatai.txt“ failą.

1.6. Programos tekstas

InOut.cs

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

namespace LD9.Classes
{
    public class InOut
    {
        /// <summary>
        /// Method which read file
        /// </summary>
        /// <param name="path">file name</param>
        /// <returns></returns>
        public static int[,] Read(string path)
        {
            string[] text = File.ReadAllLines(path);
            int dimensions = Convert.ToInt32(text[0]);

            int[,] arr = new int[dimensions, dimensions];

            for (int i = 0; i < dimensions; i++)
            {
                string[] line = text[i+1].Split(' ');
                Console.WriteLine();
                for (int j = 0; j < dimensions; j++)
                {
                    arr[i, j] = Convert.ToInt32(line[j]);
```

```
                }

            }
            return arr;
        }
        /// <summary>
        /// Method which prints into a result .txt file
        /// </summary>
        /// <param name="arr">array of path prices</param>
        /// <param name="n">starting index</param>
        /// <param name="pieces">Table</param>
        /// <param name="path">path of result file</param>
        public static void PrintData(int[,] arr, int n, string pieces, string path)
        {
            string longline = new string('-', n * 2 + 1);
            string shortline = new string('-', 7);
            List<string> lines = new List<string>();
            lines.Add(shortline);
            lines.Add("| " + n + " |");
            lines.Add(shortline);
            lines.Add("");
            lines.Add(longline);
            for (int i = 0; i < n; i++)
            {
                string collumn = "|";
                for (int j = 0; j < n ; j++)
                {
                    collumn += arr[i, j].ToString() + "|";
                }

                lines.Add(collumn);
                lines.Add(longline);
            }
            lines.Add("");
            lines.Add(pieces);


            File.WriteAllLines(path, lines);
        }

    }
}
```

TaskUtils.cs

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

namespace LD9.Classes
{

    public class TaskUtils
    {
        /// <summary>
        /// Recursion method for finding cheapest path
        /// </summary>
        /// <param name="cheapestPath">list of cheapest path</param>
        /// <param name="arr">array of path prices</param>
        /// <param name="n">starting index </param>
        /// <param name="startingX">matrix index x(i)</param>
        /// <param name="startingY">matrix index y(j)/param>
        public static void findCheapestPath(ref List<int> cheapestPath,int[,] arr, int n,int
startingX,int startingY)
```

```
{



    cheapestPath.Add(startingX + 1);

    if (cheapestPath.Count() == n)
    {
        cheapestPath.Add(1);
        return;
    }

    int min = 10000;
    int ind = 0;
    for (int i = 0; i < n; i++)
    {
        if (arr[startingX, i] < min && !cheapestPath.Contains(i+1) && i != startingX)
        {
            min = arr[startingX, i];
            ind = i;
        }
    }

    findCheapestPath(ref cheapestPath, arr, n, ind, 0);



}
/// <summary>
/// Method which calculates trip expenses
/// </summary>
/// <param name="cheapestPath">list of cheapest path</param>
/// <param name="arr">array of path prices</param>
/// <returns></returns>
public static int CalculateTripExpenses(List<int> cheapestPath, int[,] arr)
{
    int sum = 0;
    for(int i=1;i<cheapestPath.Count();i++)
    {
        sum += arr[cheapestPath[i - 1] - 1, cheapestPath[i] - 1];
    }
    return sum;
}
/// <summary>
/// Method which converts list int to string
/// </summary>
/// <param name="list"></param>
/// <returns></returns>
public static string intListToString(List<int> list)
{

    string line = "";
        for (int i = 0; i < list.Count(); i++)
        {
            if (i == list.Count() - 1)
                line += String.Format("{0} = ", list[i]);
            else
                line += String.Format("{0}, ", list[i]);
        }
        return line;


}
```

```
    }

}
```

Form1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Form1.aspx.cs" Inherits="LD9.Form1"
%>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
        </div>
        <br />
        <asp:Label ID="Label2" runat="server" Text="Pradiniai Duomenys"></asp:Label>
        <asp:Table ID="Table1" runat="server">
        </asp:Table>
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Skaičiuoti" />
        <br />
        <br />
        <asp:Label ID="Label1" runat="server"></asp:Label>
    </form>
</body>
</html>
```

Form1.aspx.cs

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


namespace LD9
{
    public partial class Form1 : System.Web.UI.Page
    {
        int[,] arr;
        int n;
        protected void Page_Load(object sender, EventArgs e)
        {

            arr = Classes.InOut.Read(Server.MapPath("App_data/U3.txt"));
            Console.WriteLine();
            n = (int)Math.Sqrt(arr.Length);
            CreateTable(arr, n);

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            List<int> cheapestPath = new List<int>();
            Classes.TaskUtils.findCheapestPath(ref cheapestPath, arr, n, 0, 0);
            Label1.Text = Classes.TaskUtils.intListToString(cheapestPath) +
Classes.TaskUtils.CalculateTripExpenses(cheapestPath, arr);
```

```
            Classes.InOut.PrintData(arr, n, Label1.Text,
Server.MapPath("App_data/rezultatai.txt"));

        }
        /// <summary>
        /// Method which creates table
        /// </summary>
        /// <param name="arr">array of path prices</param>
        /// <param name="n">starting index</param>
        public void CreateTable(int[,] arr, int n)
        {

            for (int i = 0; i < n; i++)
            {
                TableRow row = new TableRow();
                for (int j = 0; j < n; j++)
                {
                    TableCell cell = new TableCell();
                    cell.Text = arr[i,j].ToString();
                    row.Cells.Add(cell);

                }
                Table1.Rows.Add(row);
            }
        }
    }
}
```

1.7. Pradiniai duomenys ir rezultatai

1 Variantas

U3.txt

```
5
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

rezultatai.txt

rezultatai - Notepad

File Edit Format View Help

```
-------
| 5  |
-------

-----------
|0|0|0|0|0|
-----------
|0|0|0|0|0|
-----------
|0|0|0|0|0|
-----------
|0|0|0|0|0|
-----------
|0|0|0|0|0|
-----------

1, 2, 3, 4, 5, 1 = 0
```

2 Variantas

U3.txt

```
5
0 1 3 4 2
1 0 4 2 6
3 4 0 7 1
4 2 7 0 7
2 6 1 7 0
```

rezultatai.txt

rezultatai - Notepad

File Edit Format View Help

```
-------
| 5   |
-------

-----------
|0|1|3|4|2|
-----------
|1|0|4|2|6|
-----------
|3|4|0|7|1|
-----------
|4|2|7|0|7|
-----------
|2|6|1|7|0|
-----------

1, 2, 4, 3, 5, 1 = 13
```

1.8. Dėstytojo pastabos

2. Dinaminis atminties valdymas (L2)

2.1. Darbo užduotis

LD_9. **Prenumerata.** Žmonės užsisako spaudą. Užsakymas vyksta metų ribose. Užsisakant spaudą, priskiriamas agentas, kuris pristatys užsakytą spaudą į namus. Tuo tikslu agentai gauna prenumeratorių sąrašus, kurie tikslinami kiekvieną mėnesį. Sudarykite kiekvienam agentui nurodyto mėnesio (įvedamas klaviatūra) nešiojamos prenumeratos sąrašą (prenumeratoriaus adresas, pavardė, telefono numeris, laikotarpio pradžia, laikotarpio ilgis, leidinio kodas, leidinių kiekis) ir suskaičiuokite bendrą leidinių kiekį. Agento sąrašą surikiuokite pagal prenumeratoriaus adresą ir pavardę. Sudarykite sąrašą agentų, kurie nešioja daugiau nei vidutinis kiekis nurodytam mėnesiui.

Duomenys:

- Tekstiniame faile U9a.txt yra tokia informacija apie prenumeratorius: prenumeratoriaus adresas, pavardė, telefono numeris, laikotarpio pradžia (sveikasis skaičius 1..12), laikotarpio ilgis (sveikasis skaičius 1..12), leidinio kodas, leidinių kiekis, agento kodas.
- Tekstiniame faile U9b.txt yra informacija apie agentus: agento kodas, pavardė, vardas, adresas, telefonas.

Jei yra agentų, kurių nešiojamos prenumeratos kiekis neviršija duotam mėnesiui nurodyto (įvedamas klaviatūra), pašalinkite juos iš sąrašo, jų nešiojamą prenumeratą paskirstydami kitiems agentams. Priskyrimo idėja tokia: skirkite tam agentui, kurio kiekis viršija nurodytą, bet yra mažiausias tarp agentų, tenkinančių šią sąlygą. Atspausdinkite sąrašus agentams, kuriems pasikeitė nešiojamos prenumeratos kiekiai.

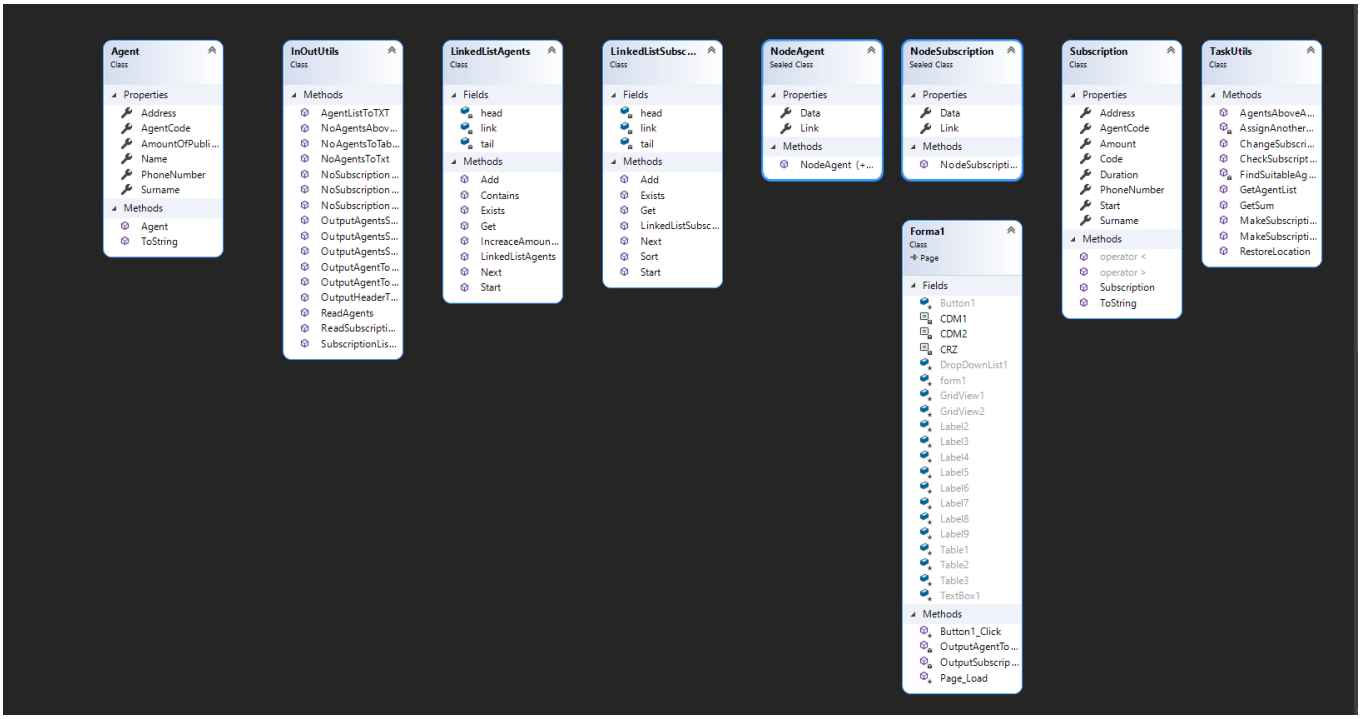
Pastaba. Laikotarpio pradžios ir laikotarpio ilgio reikšmės turi tenkinti sąlygą: laikotarpio pradžia + laikotarpio ilgis ≤ 13.

2.2. Grafinės vartotojo sąsajos schema

2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button1	Text	Skaičiuoti
Label2	Text	Vidurkis:
Label1	Text	Sarašai agentam
Label4	Text	Mėnesis
Label5	Text	Nešiojamos prenumeratos kiekis
Label7	Text	Agentai kuriem buvo pakeisti sąrašai
Label8	Text	Bendras leidinių kiekis
Label9	Text	Pradiniai duomenys:
Label3	Text	Agentų kurie nešioja daugiau nei vidurkis sąrašas:

2.4. Klasių diagrama



2.5. Programos vartotojo vadovas

Prenumeratų sąrašą reikia ikelti į App_Data aplankalą pavadinimu U9a.txt, duomenų formatas turėtu būti toks: prenumeratoriaus adresas;pavardė;telefono numeris;laikotarpio pradžia;laikotarpio ilgis; leidinio kodas;leidinių kiekis;agento kodas.

Agentų sąrašą taip pat reikia talpinti App_Data aplankale pavadinimu U9b.txt, jo duomenų formatas turėtu būti toks: agento kodas;pavardė;vardas;adresas;telefonas.

Paleidus programą reikia pasirinkti mėnesį, kurio prenumeratų sąrašus norite matyti ir į teksto lauką įvesti skaičių (prenumeratų kiekį), jei į teksto lauką galima vesti tik skaičius, kitu atveju niekas nevyks.

2.6. Programos tekstas

Agent.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public class Agent
    {
        /// <summary>
        /// Parameters of object
        /// </summary>
        public int AgentCode { get; set; }
        public string Surname { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string PhoneNumber { get; set; }
        public int AmountOfPublications { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="agentCode">agent code</param>
        /// <param name="surname">surname</param>
        /// <param name="name">name</param>
        /// <param name="address">address</param>
        /// <param name="phoneNumber">phone number</param>
        public Agent(int agentCode, string surname, string name,
            string address, string phoneNumber)
        {
            AgentCode = agentCode;
            Surname = surname;
            Name = name;
```

```
            Address = address;
            PhoneNumber = phoneNumber;
            AmountOfPublications = -1;
        }

        /// <summary>
        /// override for > operator
        /// </summary>
        /// <param name="left">one agent</param>
        /// <param name="right">another agent</param>
        /// <returns>true or false</returns>
        public static bool operator >(Agent left, Agent right)
        {
            if (left.Address.CompareTo(right.Address) == 0)
                return left.Surname.CompareTo(right.Surname) > 0;
            else
            {
                return left.Address.CompareTo(right.Address) > 0;
            }
        }

        /// <summary>
        /// override for < operator
        /// </summary>
        /// <param name="left">one agent</param>
        /// <param name="right">another agent</param>
        /// <returns>true or false</returns>
        public static bool operator <(Agent left, Agent right)
        {
            if (left.Address.CompareTo(right.Address) == 0)
                return left.Surname.CompareTo(right.Surname) < 0;
            else
            {
                return left.Address.CompareTo(right.Address) < 0;
            }
        }

        /// <summary>
        /// Override for ToString method
        /// </summary>
        /// <returns>string</returns>
        public override string ToString()
        {
            return string.Format("| {0, 12} | {1, -15} | {2, -15}" +
                " | {3, -30} | | {4,16} |", AgentCode, Surname,
                Name, Address, PhoneNumber);
        }
    }
}
```

NodeAgent.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public sealed class NodeAgent
    {
        /// <summary>
        /// Parameters of NodeAgent class
        /// </summary>
        public Agent Data { get; private set; }
        public NodeAgent Link { get; set; }
```

```
        /// <summary>
        /// Empty construcor
        /// </summary>
        public NodeAgent()
        { }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="data">agent</param>
        /// <param name="link">another segment</param>
        public NodeAgent(Agent data, NodeAgent link)
        {
            Data = data;
            Link = link;
        }
    }
}
```

InOutUtils.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
namespace LD_9_Prenumerata.App_BarCode
{
    public class InOutUtils
    {
        /// <summary>
        /// Reads subscriptions from file
        /// </summary>
        /// <param name="FileName">name of file</param>
        /// <returns>subscriptions</returns>
        public static LinkedListSubscriptions ReadSubscriptions
            (string FileName)
        {
            LinkedListSubscriptions subscriptions
                = new LinkedListSubscriptions();
            using (StreamReader reader =
                new StreamReader(FileName, Encoding.UTF8))
            {
                string line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] values = line.Split(';');
                    string address = values[0];
                    string surname = values[1];
                    string phonenumber = values[2];
                    int start = Convert.ToInt32(values[3]);
                    int duration = Convert.ToInt32(values[4]);
                    int code = Convert.ToInt32(values[5]);
                    int amount = Convert.ToInt32(values[6]);
                    int agentCode = Convert.ToInt32(values[7]);
                    subscriptions.Add(new Subscription(address,
                        surname, phonenumber, start, duration,
                        code, amount, agentCode));
                }
            }
            return subscriptions;
        }

        /// <summary>
        /// Reads agents from file
        /// </summary>
```

```
/// <param name="FileName">name of file</param>
/// <returns>agents</returns>
public static LinkedListAgents ReadAgents
    (string FileName)
{
    LinkedListAgents agents =
        new LinkedListAgents();
    using(StreamReader reader =
        new StreamReader(FileName,Encoding.UTF8))
    {
        string line;
        while (( line = reader.ReadLine()) != null)
        {
            string[] values = line.Split(';');
            int agentcode = Convert.ToInt32(values[0]);
            string surname = values[1];
            string name = values[2];
            string address = values[3];
            string phonenumber = values[4];
            agents.Add(new Agent(agentcode,
                surname, name, address, phonenumber));
        }
    }
    return agents;
}

/// <summary>
/// Outputs agent and his subscriptions to txt file
/// </summary>
/// <param name="FileName">name of file</param>
/// <param name="agent">agent</param>
/// <param name="subscriptions">agent's subscriptions</param>
public static void OutputAgentsSubscriptionsToTxt
    (string FileName, Agent agent,
    LinkedListSubscriptions subscriptions)
{
    using (StreamWriter writer =
        new StreamWriter(FileName,append: true))
    {
        writer.WriteLine("");
        writer.WriteLine(new string('-', 106));
        writer.WriteLine(string.Format("| {0, -12}" +
            " | {1, -15} | {2, -15} | {3, -30} | |" +
            " {4,-16} |", "Agento kodas", "Pavardė",
            "Vardas", "Adresas", "Telefono numeris"));
        writer.WriteLine(agent.ToString());
        writer.WriteLine(new string('-', 106));
        subscriptions.Start();
        if (subscriptions.Exists())
        {
            writer.WriteLine("Prenumeratos:");
            writer.WriteLine(new string('-', 169));
            writer.WriteLine(string.Format
                ("| {0, -30} | {1, -15} | {2, 16}" +
                " | {3, 20} | {4, 20} | {5, 15} |" +
                " {6, 16} | {7, 12} |", "Adresas",
                "Pavardė", "Telefono numeris",
                "Laikotarpio pradžia",
                "Laikotarpio ilgis", "Leidinio kodas",
                "Leidinių kiekis", "Agento kodas"));
            for (subscriptions.Start();
                subscriptions.Exists(); subscriptions.Next())
            {
                writer.WriteLine(subscriptions.Get().ToString());
            }
            writer.WriteLine(new string('-', 169));
        }
```

```
        else
        {
            writer.WriteLine("Prenumeratų nėra");
        }
    }
}

/// <summary>
/// Outputs agent and his subscriptions to table
/// </summary>
/// <param name="table">table</param>
/// <param name="agent">agent</param>
/// <param name="subscriptions">agent's subscription</param>
public static void OutputAgentsSubscriptionsToTable
    (Table table, Agent agent,
    LinkedListSubscriptions subscriptions)
{
    TableCell information = new TableCell();
    TableRow inform = new TableRow();
    information.Text = "Informacija apie agentą: ";

    inform.Cells.Add(information);
    table.Rows.Add(inform);

    TableCell tcell1 = new TableCell();
    TableCell tcell2 = new TableCell();
    TableCell tcell3 = new TableCell();
    TableCell tcell4 = new TableCell();
    TableCell tcell5 = new TableCell();
    TableCell tcell6 = new TableCell();
    TableCell tcell7 = new TableCell();
    TableRow explination = new TableRow();
    tcell1.Text = "Agento kodas";
    tcell2.Text = "Pavardė";
    tcell3.Text = "Vardas";
    tcell4.Text = "Adresas";
    tcell5.Text = "Telefono numeris";

    explination.Cells.Add(tcell1);
    explination.Cells.Add(tcell2);
    explination.Cells.Add(tcell3);
    explination.Cells.Add(tcell4);
    explination.Cells.Add(tcell5);
    table.Rows.Add(explination);

    TableRow AgentInfo = new TableRow();

    TableCell agentcode = new TableCell();
    agentcode.Text = agent.AgentCode.ToString();

    TableCell surname = new TableCell();
    surname.Text = agent.Surname;

    TableCell name = new TableCell();
    name.Text = agent.Name;

    TableCell address = new TableCell();
    address.Text = agent.Address.ToString();

    TableCell phone = new TableCell();
    phone.Text = agent.PhoneNumber.ToString();

    AgentInfo.Cells.Add(agentcode);
    AgentInfo.Cells.Add(surname);
    AgentInfo.Cells.Add(name);
    AgentInfo.Cells.Add(address);
    AgentInfo.Cells.Add(phone);
```

```
table.Rows.Add(AgentInfo);
subscriptions.Start();
if (subscriptions.Exists())
{
    TableCell buff = new TableCell();
    buff.Text = "Prenumeratų sąrašas:";
    TableRow buff1 = new TableRow();
    buff1.Cells.Add(buff);
    table.Rows.Add(buff1);

    TableCell cell1 = new TableCell();
    TableCell cell2 = new TableCell();
    TableCell cell3 = new TableCell();
    TableCell cell4 = new TableCell();
    TableCell cell5 = new TableCell();
    TableCell cell6 = new TableCell();
    TableCell cell7 = new TableCell();

    cell1.Text = "Adresas";
    cell2.Text = "Paverdė";
    cell3.Text = "Telefono numeris";
    cell4.Text = "Laikotarpio pradžia";
    cell5.Text = "Laikotarpio ilgis";
    cell6.Text = "Leidinio kodas";
    cell7.Text = "Leidinių kiekis";

    TableRow row = new TableRow();
    row.Cells.Add(cell1);
    row.Cells.Add(cell2);
    row.Cells.Add(cell3);
    row.Cells.Add(cell4);
    row.Cells.Add(cell5);
    row.Cells.Add(cell6);
    row.Cells.Add(cell7);

    table.Rows.Add(row);

    for (subscriptions.Start(); subscriptions.Exists(); subscriptions.Next())
    {
        TableCell sub1 = new TableCell();
        TableCell sub2 = new TableCell();
        TableCell sub3 = new TableCell();
        TableCell sub4 = new TableCell();
        TableCell sub5 = new TableCell();
        TableCell sub6 = new TableCell();
        TableCell sub7 = new TableCell();
        TableRow subrow = new TableRow();

        sub1.Text = subscriptions.Get().Address;
        sub2.Text = subscriptions.Get().Surname;
        sub3.Text = subscriptions.Get().PhoneNumber;
        sub4.Text = subscriptions.Get().Start.ToString();
        sub5.Text = subscriptions.Get().Duration.ToString();
        sub6.Text = subscriptions.Get().Code.ToString();
        sub7.Text = subscriptions.Get().Amount.ToString();

        subrow.Cells.Add(sub1);
        subrow.Cells.Add(sub2);
        subrow.Cells.Add(sub3);
        subrow.Cells.Add(sub4);
        subrow.Cells.Add(sub5);
        subrow.Cells.Add(sub6);
        subrow.Cells.Add(sub7);

        table.Rows.Add(subrow);
    }
```

```
        }
          else
        {
            TableCell output = new TableCell();
            output.Text = "Prenumeratų ši mėnesį agentas nenešioja";
            TableRow outputt = new TableRow();
            outputt.Cells.Add(output);
            table.Rows.Add(outputt);
        }
        TableCell Space = new TableCell();
        Space.Text = "-";
        TableRow space = new TableRow();
        space.Cells.Add(Space);
        table.Rows.Add(space);
    }

    public static void OutputAgentsSubscriptionsToTableS
        (Table table, Agent agent,
        LinkedListSubscriptions subscriptions)
    {
        TableCell information = new TableCell();
        TableRow inform = new TableRow();
        information.Text = "Informacija apie agentą: ";

        inform.Cells.Add(information);
        table.Rows.Add(inform);

        TableCell tcell1 = new TableCell();
        TableCell tcell2 = new TableCell();
        TableCell tcell3 = new TableCell();
        TableCell tcell4 = new TableCell();
        TableCell tcell5 = new TableCell();
        TableCell tcell6 = new TableCell();
        TableCell tcell7 = new TableCell();
        TableRow explination = new TableRow();
        tcell1.Text = "Agento kodas";
        tcell2.Text = "Pavardė";
        tcell3.Text = "Vardas";
        tcell4.Text = "Adresas";
        tcell5.Text = "Telefono numeris";

        explination.Cells.Add(tcell1);
        explination.Cells.Add(tcell2);
        explination.Cells.Add(tcell3);
        explination.Cells.Add(tcell4);
        explination.Cells.Add(tcell5);
        table.Rows.Add(explination);

        TableRow AgentInfo = new TableRow();

        TableCell agentcode = new TableCell();
        agentcode.Text = agent.AgentCode.ToString();

        TableCell surname = new TableCell();
        surname.Text = agent.Surname;

        TableCell name = new TableCell();
        name.Text = agent.Name;

        TableCell address = new TableCell();
        address.Text = agent.Address.ToString();

        TableCell phone = new TableCell();
        phone.Text = agent.PhoneNumber.ToString();

        AgentInfo.Cells.Add(agentcode);
        AgentInfo.Cells.Add(surname);
```

```
        AgentInfo.Cells.Add(name);
        AgentInfo.Cells.Add(address);
        AgentInfo.Cells.Add(phone);

        table.Rows.Add(AgentInfo);
    }


        /// <summary>
        /// Outputs agent to txt file
        /// </summary>
        /// <param name="fileName">name of file</param>
        /// <param name="agent">agent</param>
        public static void OutputAgentToTxt(string fileName, Agent agent)
    {
        using (StreamWriter writer = new StreamWriter(fileName, append: true))
        {
            writer.WriteLine("");
            writer.WriteLine(new string('-', 106));
            writer.WriteLine(string.Format("| {0, -12}" +
                " | {1, -15} | {2, -15} | {3, -30} |" +
                " | {4,-16} |", "Agento kodas",
                "Pavardė", "Vardas", "Adresas",
                "Telefono numeris"));
            writer.WriteLine(agent.ToString());
            writer.WriteLine(new string('-', 106));
        }
    }

    /// <summary>
    /// Outputs agent to table
    /// </summary>
    /// <param name="table">table</param>
    /// <param name="agent">agent</param>
    public static void OutputAgentToTable(Table table, Agent agent)
    {
        TableCell information = new TableCell();
        TableRow inform = new TableRow();
        information.Text = "Informacija apie agentą: ";

        inform.Cells.Add(information);
        table.Rows.Add(inform);

        TableCell tcell1 = new TableCell();
        TableCell tcell2 = new TableCell();
        TableCell tcell3 = new TableCell();
        TableCell tcell4 = new TableCell();
        TableCell tcell5 = new TableCell();
        TableCell tcell6 = new TableCell();
        TableCell tcell7 = new TableCell();
        TableRow explination = new TableRow();
        tcell1.Text = "Agento kodas";
        tcell2.Text = "Pavardė";
        tcell3.Text = "Vardas";
        tcell4.Text = "Adresas";
        tcell5.Text = "Telefono numeris";

        explination.Cells.Add(tcell1);
        explination.Cells.Add(tcell2);
        explination.Cells.Add(tcell3);
        explination.Cells.Add(tcell4);
        explination.Cells.Add(tcell5);
        table.Rows.Add(explination);

        TableRow AgentInfo = new TableRow();

        TableCell agentcode = new TableCell();
```

```
            agentcode.Text = agent.AgentCode.ToString();

            TableCell surname = new TableCell();
            surname.Text = agent.Surname;

            TableCell name = new TableCell();
            name.Text = agent.Name;

            TableCell address = new TableCell();
            address.Text = agent.Address.ToString();

            TableCell phone = new TableCell();
            phone.Text = agent.PhoneNumber.ToString();

            AgentInfo.Cells.Add(agentcode);
            AgentInfo.Cells.Add(surname);
            AgentInfo.Cells.Add(name);
            AgentInfo.Cells.Add(address);
            AgentInfo.Cells.Add(phone);

            table.Rows.Add(AgentInfo);
        }

        public static void AgentListToTXT(LinkedListAgents agents, string path, string header)
        {
            List<string> lines = new List<string>();
            string line = new string('-', 106);
            lines.Add(header);
            lines.Add(line);
            lines.Add(string.Format("| {0, 12} | {1, -15} | {2, -15}" +
                " | {3, -30} | | {4,16} |", "Agento kodas", "Pavardė",
                "Vardas", "Adresas", "Telefono numeris"));
            lines.Add(line);
            for (agents.Start(); agents.Exists(); agents.Next())
            {
                lines.Add(agents.Get().ToString());
                lines.Add(line);
            }

            lines.Add("");
            lines.Add("");

            File.AppendAllLines(path, lines);
        }

        public static void SubscriptionListToTXT(LinkedListSubscriptions subscriptions, string
path, string header)
        {
            List<string> lines = new List<string>();
            string line = new string('-', 169);
            lines.Add(header);
            lines.Add(line);
            lines.Add(string.Format
                        ("| {0, -30} | {1, -15} | {2, 16}" +
                        " | {3, 20} | {4, 20} | {5, 15} |" +
                        " {6, 16} | {7, 12} |", "Adresas",
                        "Pavardė", "Telefono numeris",
                        "Laikotarpio pradžia",
                        "Laikotarpio ilgis", "Leidinio kodas",
                        "Leidinių kiekis", "Agento kodas"));
            lines.Add(line);
            for (subscriptions.Start(); subscriptions.Exists(); subscriptions.Next())
            {
                lines.Add(subscriptions.Get().ToString());
                lines.Add(line);
            }
```

```
        lines.Add("");
        lines.Add("");

        File.AppendAllLines(path, lines);
    }

    /// <summary>
    /// Outputs that no agents are in txt file to txt file
    /// </summary>
    /// <param name="fileName">name of file</param>
    public static void NoAgentsToTxt(string fileName)
    {
        File.AppendAllText(fileName, "Duomenų faile nėra agentų");
    }

    /// <summary>
    /// Outputs that no agents are in txt file to table
    /// </summary>
    /// <param name="table">table</param>
    public static void NoAgentsToTable(Table table)
    {
        TableCell cell = new TableCell();
        cell.Text = "Duomenų faile nėra agentų";
        TableRow row = new TableRow();
        row.Cells.Add(cell);
        table.Rows.Add(row);
    }
   /// <summary>
    /// Outputs that no subscriptions are in txt file to txt
    /// </summary>
    /// <param name="fileName">name of file</param>
    public static void NoSubscriptionsToTxt
        (string fileName)
    {
        File.AppendAllText(fileName,
            "Duomenų faile nėra prenumeratų");
    }

    /// <summary>
    /// Outputs that no subscriptions are in txt file to table
    /// </summary>
    /// <param name="table">table</param>
    public static void NoSubscriptionsToTable(Table table)
    {
        TableCell cell = new TableCell();
        cell.Text = "Duomenų faile nėra prenumeratų";
        TableRow row = new TableRow();
        row.Cells.Add(cell);
        table.Rows.Add(row);
    }

    /// <summary>
    /// Outputs that no agent has subscriptions above given number to table
    /// </summary>
    /// <param name="table">table</param>
    public static void NoAgentsAboveAmount(Table table)
    {
        TableCell cell = new TableCell();
        cell.Text = "Duomenų faile nėra agentų," +
            " kurie nešiotų didesnį leidinių kiekį" +
            " nei nurodyta";
        TableRow row = new TableRow();
        row.Cells.Add(cell);
        table.Rows.Add(row);
    }

    /// <summary>
```

```
        /// Outputs that no agent has subscriptions above given number to txt
        /// </summary>
        /// <param name="fileName">name of file</param>
        public static void NoAgentsAboveAmount(string fileName)
        {
            File.AppendAllText(fileName, "Nei vienas" +
                " agentas nenešioja daugiau prenumeratos" +
                " nei nurodyta");
        }

        /// <summary>
        /// Outputs that no subscriptions are carried in that month to table
        /// </summary>
        /// <param name="table">table</param>
        public static void NoSubscriptionsInMonth(Table table)
        {
            TableCell cell = new TableCell();
            cell.Text = "Šį mėnesį žmonės neužsisakinėja" +
                " prenumeratų";
            TableRow row = new TableRow();
            row.Cells.Add(cell);
            table.Rows.Add(row);
        }

        /// <summary>
        /// Outputs that no subscriptions are carried in that month to txt
        /// </summary>
        /// <param name="fileName">name of file</param>
        public static void NoSubscriptionsInMonth(string fileName)
        {
            File.AppendAllText(fileName, "Ši mėnesį" +
                " agentai nenešioja prenumeratos");
        }

        /// <summary>
        /// OutputsHeaderToTxt
        /// </summary>
        /// <param name="fileName">name of file</param>
        /// <param name="header">header</param>
        public static void OutputHeaderTotxt(string fileName, string header)
        {
            File.AppendAllText(fileName, header);
        }
    }

}
```

LinkedListAgent.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public class LinkedListAgents
    {
        /// <summary>
        /// Parameters of linked list
        /// </summary>
        private NodeAgent head;
        private NodeAgent tail;
        private NodeAgent link;

        /// <summary>
        /// Empty Construcor
```

```
/// </summary>
public LinkedListAgents()
{
    head = null;
    tail = null;
}

/// <summary>
/// Start of list
/// </summary>
public void Start()
{
    link = head;
}

/// <summary>
/// Next object in list
/// </summary>
public void Next()
{
    link = link.Link;
}

/// <summary>
/// Checks if next object exists in the list
/// </summary>
/// <returns></returns>
public bool Exists()
{
    return link != null;
}

/// <summary>
/// Gets object
/// </summary>
/// <returns>Agent</returns>
public Agent Get()
{
    return link.Data;
}

/// <summary>
/// Adds object to list
/// </summary>
/// <param name="agent"></param>
public void Add(Agent agent)
{
    NodeAgent d = new NodeAgent(agent, null);
    if(head!=null)
    {
        tail.Link = d;
        tail = d;
    }
    else
    {
        head = d;
        tail = d;
    }
}

/// <summary>
/// Checks if list contains an agent
/// </summary>
/// <param name="agent">Agent</param>
/// <returns>true or false</returns>
public bool Contains(Agent agent)
{
```

```
        for (this.Start(); this.Exists(); this.Next())
        {
            if (this.Get().Name == agent.Name &&
                this.Get().Surname == agent.Surname)
                return true;
        }
        return false;
    }

    /// <summary>
    /// Adds amount of publications to agent
    /// </summary>
    /// <param name="agent">agent to add to</param>
    /// <param name="value">amount of publications</param>
    public void IncreaceAmountOfPublications(Agent agent, int value)
    {
        for (this.Start(); this.Exists(); this.Next())
        {
            if (this.Get().Name == agent.Name &&
                this.Get().Surname == agent.Surname)
                this.link.Data.AmountOfPublications += value;
        }
    }

    public void Remove(Agent agent)
    {
        if (head == null) return;

        if (head.Data.Equals(agent))
        {
            head = head.Link;
            return;
        }

        for (NodeAgent n = head; n.Link != null; n = n.Link)
        {
            if (n.Link.Data.Equals(agent))
            {
                n.Link = n.Link.Link;
                return;
            }
        }
    }

    public void Sort()
    {
        if (head == null)
            return;
        bool done = true;
        while (done)
        {
            done = false;
            var headn = head;
            while (headn.Link != null)
            {
                if (headn.Data > headn.Link.Data)
                {
                    Agent sub = headn.Data;
                    headn.Data = headn.Link.Data;
                    headn.Link.Data = sub;
                    done = true;
                }
                headn = headn.Link;
            }
        }
    }
}
```

```
}
LinkedListSubscriptions.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public class LinkedListSubscriptions
    {
        private NodeSubscription head;
        private NodeSubscription tail;
        private NodeSubscription link;

        public LinkedListSubscriptions()
        {
            this.head = null;
            this.tail = null;
        }

        public bool Exists()
        {
            return link != null;
        }
        public void Start()
        {
            link = head;
        }

        public void Next()
        {
            link = link.Link;
        }

        public Subscription Get()
        {
            return link.Data;
        }

        public void Add(Subscription subscription)
        {
            NodeSubscription d = new NodeSubscription(subscription, null);
            if (head != null)
            {
                tail.Link = d;
                tail = d;
            }
            else
            {
                head = d;
                tail = d;
            }
        }

        public void Sort()
        {
            if (head == null)
                return;
            bool done = true;
            while(done)
            {
                done = false;
                var headn = head;
                while(headn.Link!=null)
                {
```

```
                    if(headn.Data>headn.Link.Data)
                    {
                        Subscription sub = headn.Data;
                        headn.Data = headn.Link.Data;
                        headn.Link.Data = sub;
                        done = true;
                    }
                    headn = headn.Link;
                }
            }
        }
    }
}
```

NodeSubscription.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public sealed class NodeSubscription
    {
        /// <summary>
        /// Parameters of NodeSubscription class
        /// </summary>
        public Subscription Data { get; set; }
        public NodeSubscription Link { get; set; }

        /// <summary>
        /// Empty construcor
        /// </summary>
        public NodeSubscription()
        { }

        /// <summary>
        /// Construcor
        /// </summary>
        /// <param name="data">subscription</param>
        /// <param name="link">next segment</param>
        public NodeSubscription(Subscription data, NodeSubscription link)
        {
            Data = data;
            Link = link;
        }
    }
}
```

Subscription.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public class Subscription
    {
        /// <summary>
        /// Parameters of object
        /// </summary>
        public string Address { get; set; }
        public string Surname { get; set; }
        public string PhoneNumber { get; set; }
        public int Start { get; set; }
        public int Duration { get; set; }
```

```
public int Code { get; set; }
public int Amount { get; set; }
public int AgentCode { get; set; }

/// <summary>
/// Constructor
/// </summary>
/// <param name="address">address</param>
/// <param name="surname">surname</param>
/// <param name="phoneNumber">phone number</param>
/// <param name="start">start of period</param>
/// <param name="duration">duration of period</param>
/// <param name="code">code of publication</param>
/// <param name="amount">amount of publications</param>
/// <param name="agentCode">agent code</param>
public Subscription(string address, string surname,
    string phoneNumber, int start, int duration,
    int code, int amount, int agentCode)
{
    Address = address;
    Surname = surname;
    PhoneNumber = phoneNumber;
    Start = start;
    Duration = duration;
    Code = code;
    Amount = amount;
    AgentCode = agentCode;
}

/// <summary>
/// override for > operator
/// </summary>
/// <param name="left">one subscription</param>
/// <param name="right">another subscription</param>
/// <returns>true or false</returns>
public static bool operator > (Subscription left, Subscription right)
{
    if (left.Address.CompareTo(right.Address) == 0)
        return left.Surname.CompareTo(right.Surname) >0;
    else
    {
        return left.Address.CompareTo(right.Address) >0;
    }
}

/// <summary>
/// override for < operator
/// </summary>
/// <param name="left">one subscription</param>
/// <param name="right">another subscription</param>
/// <returns>true or false</returns>
public static bool operator <(Subscription left, Subscription right)
{
    if (left.Address.CompareTo(right.Address) == 0)
        return left.Surname.CompareTo(right.Surname) < 0;
    else
    {
        return left.Address.CompareTo(right.Address) < 0;
    }
}

/// <summary>
/// override for ToString method
/// </summary>
/// <returns>string</returns>
public override string ToString()
{
```

```
            return string.Format("| {0, -30} | {1, -15} | {2, 16} |" +
                " {3, 20} | {4, 20} | {5, 15} | {6, 16} | {7, 12} |",
                Address, Surname, PhoneNumber, Start, Duration, Code,
                Amount, AgentCode);
        }
    }


}
```

TaskUtils.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
namespace LD_9_Prenumerata.App_BarCode
{
    public class TaskUtils
    {
        /// <summary>
        /// Gets subscription list to agents in specific month
        /// </summary>
        /// <param name="month">month</param>
        /// <param name="subscriptions">subscriptions</param>
        /// <param name="agent">agents</param>
        /// <returns>subscriptions</returns>
        public static LinkedListSubscriptions GetAgentList
            (int month, LinkedListSubscriptions subscriptions, Agent agent)
        {
            int agentCode = agent.AgentCode;
            LinkedListSubscriptions subs =
                new LinkedListSubscriptions();
            for (subscriptions.Start();
                subscriptions.Exists();
                subscriptions.Next())
            {
                Subscription sub = subscriptions.Get();
                if (sub.AgentCode == agentCode &&
                    sub.Start == month)
                    subs.Add(sub);
            }
            subs.Sort();
            return subs;
        }

        /// <summary>
        /// Checks if there is any subscriptions in month
        /// </summary>
        /// <param name="month">month</param>
        /// <param name="subs">subscriptions</param>
        /// <returns>true or false</returns>
        public static bool CheckSubscriptionsInMonth
            (int month, LinkedListSubscriptions subs)
        {
            for(subs.Start();subs.Exists();subs.Next())
            {
                if (subs.Get().Start == month)
                    return true;
            }
            return false;
        }

        /// <summary>
        /// Gets sum of carried subscriptions
```

```
/// </summary>
/// <param name="sub">subscriptions</param>
/// <returns>sum</returns>
public static int GetSum(LinkedListSubscriptions subs)
{
    int sum = 0;
    for (subs.Start();subs.Exists();subs.Next())
    {
        sum += subs.Get().Amount;
    }
    return sum;
}

/// <summary>
/// Changes subscription for agents
/// </summary>
/// <param name="subs">subscriptions</param>
/// <param name="agents">agents</param>
/// <param name="number">number</param>
/// <param name="changed">list of agents whose subscriptions changed</param>
/// <param name="valid">valid</param>
/// <returns>subscriptions</returns>
public static LinkedListSubscriptions ChangeSubscriptions
    (LinkedListSubscriptions subs, LinkedListAgents agents,
    int number, ref LinkedListAgents changed, ref bool valid)
{
    int position = 0;
    for (agents.Start(); agents.Exists(); agents.Next())
    {
        if(agents.Get().AmountOfPublications<=number
            && agents.Get().AmountOfPublications > 0)
        {
            for(subs.Start();subs.Exists();subs.Next())
            {
                if (subs.Get().AgentCode ==
                    agents.Get().AgentCode)
                    subs.Get().AgentCode = -1;
            }

            agents.Remove(agents.Get());

            NodeAgent buff = new NodeAgent();
            subs = AssignAnotherAgent
                (subs, agents, number, changed, ref valid);

            RestoreLocation(agents, position);
        }
        position++;
    }
    return subs;
}

/// <summary>
/// Assigns another agent
/// </summary>
/// <param name="subs">subscriptions</param>
/// <param name="agents">agents</param>
/// <param name="number">number</param>
/// <param name="changed">changed</param>
/// <param name="valid">valid</param>
/// <returns>subscriptions</returns>
private static LinkedListSubscriptions AssignAnotherAgent
    (LinkedListSubscriptions subs, LinkedListAgents agents,
    int number, LinkedListAgents changed, ref bool valid)
{
    Agent agent = FindSuitableAgent(agents, number);
    if (agent != null)
```

```
        {
            for (subs.Start(); subs.Exists(); subs.Next())
            {
                if (subs.Get().AgentCode == -1)
                {
                    subs.Get().AgentCode = agent.AgentCode;
                    agents.IncreaceAmountOfPublications
                        (agent, subs.Get().Amount);
                }
            }
            if (!changed.Contains(agent))
                changed.Add(agent);
        }
        else
        {
            valid = false;
        }
        return subs;

    }

    /// <summary>
    /// Finds suitable agent
    /// </summary>
    /// <param name="agents">agents</param>
    /// <param name="number">number</param>
    /// <returns></returns>
    private static Agent FindSuitableAgent
        (LinkedListAgents agents, int number)
    {
        Agent agent = null;
        agents.Start();
        if(agents.Exists())
        {
            for(agents.Start();agents.Exists();
                agents.Next())
            {
                if(agents.Get().AmountOfPublications>number)
                {
                    if(agent == null)
                    {
                        agent = agents.Get();
                    }
                    else if(agent.AmountOfPublications>
                        agents.Get().AmountOfPublications)
                    {
                        agent = agents.Get();
                    }
                }
            }
        }
        return agent;
    }

    /// <summary>
    /// Makes subscription list
    /// </summary>
    /// <param name="agents">agents</param>
    /// <param name="subscriptions">subscriptions</param>
    /// <param name="fileName">name of file</param>
    /// <param name="month">month</param>
    /// <param name="Table">table</param>
    /// <returns>average</returns>
    public static int MakeSubscriptionList
        (LinkedListAgents agents,
        LinkedListSubscriptions subscriptions,
        string fileName, int month, Table Table)
```

```
{
    int sum = 0;
    int agentscount = 0;
    InOutUtils.OutputHeaderTotxt(fileName,
        "Agentai, kuriem buvo pakeisti sąrašai:");
    for (agents.Start(); agents.Exists();
        agents.Next())
    {
        App_BarCode.LinkedListSubscriptions buff =
            App_BarCode.TaskUtils.GetAgentList
            (month, subscriptions, agents.Get());
        App_BarCode.InOutUtils.
            OutputAgentsSubscriptionsToTxt
            (fileName, agents.Get(), buff);
        App_BarCode.InOutUtils.
            OutputAgentsSubscriptionsToTable
            (Table, agents.Get(), buff);

        agents.Get().AmountOfPublications =
            App_BarCode.TaskUtils.GetSum(buff);


        sum += agents.Get().AmountOfPublications;
        agentscount++;
    }
    decimal average = (sum / agentscount);
    average = Math.Floor(average);
    int avg = Convert.ToInt32(average);

    return avg;
}

public static int MakeSubscriptionListSUM
    (LinkedListAgents agents,
    LinkedListSubscriptions subscriptions,
    string fileName, int month, Table Table)
{
    int sum = 0;
    int agentscount = 0;

    for (agents.Start(); agents.Exists();
        agents.Next())
    {
        App_BarCode.LinkedListSubscriptions buff =
            App_BarCode.TaskUtils.GetAgentList
            (month, subscriptions, agents.Get());


        agents.Get().AmountOfPublications =
            App_BarCode.TaskUtils.GetSum(buff);


        sum += agents.Get().AmountOfPublications;
        agentscount++;
    }
    decimal average = (sum / agentscount);
    average = Math.Floor(average);
    int avg = Convert.ToInt32(average);

    return sum;
}

/// <summary>
/// Finds agents above average
/// </summary>
/// <param name="agents">agents</param>
/// <param name="avarage">average</param>
```

```
        /// <param name="table">table</param>
        /// <param name="fileName">name of file</param>
        public static void AgentsAboveAvarage
            (LinkedListAgents agents, int avarage,
            Table table, string fileName)
        {
            InOutUtils.OutputHeaderTotxt(fileName,
                "Agentai nešiojantys daugiau nei vidurkis:");
            for(agents.Start();agents.Exists();agents.Next())
            {
                if(agents.Get().AmountOfPublications>avarage)
                {
                    InOutUtils.OutputAgentToTxt(fileName, agents.Get());
                    InOutUtils.OutputAgentToTable(table, agents.Get());
                }

            }
        }

        public static void RestoreLocation(LinkedListAgents agents, int value)
        {
            agents.Start();
            for (int i = 0; i < value; i++)
            {
                agents.Next();
            }
        }
    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="LD_9_Prenumerata.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <link href="Style.css" rel="stylesheet" />
    <title></title>
</head>
<body style="height: 505px">
    <form id="form1" runat="server">
        <div style="height: 504px">
            <asp:Label ID="Label4" runat="server" Text="Mėnesis:"></asp:Label>
            <br />
            <asp:DropDownList ID="DropDownList1" runat="server">
            </asp:DropDownList>
            <br />
            <br />
            <asp:Label ID="Label5" runat="server" Text="Nešiojamos prenumeratos
kiekis:"></asp:Label>
            <br />
            <asp:TextBox ID="TextBox1" runat="server" style="width: 168px"></asp:TextBox>
            <br />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Skaičiuoti" />
            <br />
            <br />
             <br />
            <asp:Label ID="Label9" runat="server" Text="Pradiniai duomenys:"></asp:Label>
            <br />
            <br />
            <asp:GridView ID="GridView1" runat="server">
            </asp:GridView>
            <br />
```

```
            <asp:GridView ID="GridView2" runat="server">
            </asp:GridView>
            <br />
            <br />
            <asp:Label ID="Label2" runat="server" Text="Sąrašai agentam:"></asp:Label>
            <br />
            <asp:Table ID="Table1" runat="server" GridLines="Both">
            </asp:Table>
            <br />
            <asp:Label ID="Label7" runat="server" Text="Vidurkis: "></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label8" runat="server" Text="Bendras leidiniu kiekis: "></asp:Label>
            <br />
            <asp:Label ID="Label3" runat="server" Text="Agentų, kurie nešioja daugiau nei
vidurkis, sąrašas:"></asp:Label>
            <br />
            <asp:Table ID="Table2" runat="server" GridLines="Both">
            </asp:Table>
            <br />
            <asp:Label ID="Label6" runat="server" Text="Agentai kuriem buvo pakeisti
sąrašai:"></asp:Label>
            <br />
            <asp:Table ID="Table3" runat="server" GridLines="Both">
            </asp:Table>
        </div>
    </form>
</body>
</html>
```

Forma1.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Text.RegularExpressions;
using System.Data;
using LD_9_Prenumerata.App_BarCode;

namespace LD_9_Prenumerata
{
    public partial class Forma1 : System.Web.UI.Page
    {
        const string CDM1 = "U9a.txt";
        const string CDM2 = "U9b.txt";
        const string CRZ = "U9rez.txt";
        protected void Page_Load(object sender, EventArgs e)
        {
            if (DropDownList1.Items.Count == 0)
            {
                for (int i = 1; i <= 12; i++)
                {
                    DropDownList1.Items.Add(i.ToString());
                }

            }
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            var Matches = Regex.Match(TextBox1.Text, @"\D");
            if (!Matches.Success && TextBox1.Text.Length > 0)
            {
```

```
                File.Delete(Server.MapPath("App_Data/" + CRZ));

                #region Read

                string FileName = Server.MapPath("App_Data/" + CDM1);
                LinkedListSubscriptions subscriptions =
                    InOutUtils.ReadSubscriptions(FileName);
                FileName = Server.MapPath("App_Data/" + CDM2);
                LinkedListAgents agents =
                    InOutUtils.ReadAgents(FileName);


                OutputAgentToTable(agents);
                OutputSubscriptionsToTable(subscriptions);
                InOutUtils.AgentListToTXT(agents, Server.MapPath("App_Data/" + CRZ), "Pradiniai
duomenys");
                InOutUtils.SubscriptionListToTXT(subscriptions, Server.MapPath("App_Data/" +
CRZ), "");

                #endregion
                string fileName = Server.MapPath("App_Data/" + CRZ);
                subscriptions.Start();
                agents.Start();
                int month = Convert.ToInt32(DropDownList1.SelectedValue);
                if (subscriptions.Exists())
                {
                    if (agents.Exists())
                    {
                        if (TaskUtils.CheckSubscriptionsInMonth
                            (month, subscriptions))
                        {
                            #region Making subscription lists

                            int average = TaskUtils.
                                MakeSubscriptionList(agents, subscriptions,
                                fileName, month, Table1);
                            int suma = TaskUtils.
                                MakeSubscriptionListSUM(agents, subscriptions,
                                fileName, month, Table1);

                            #endregion

                            #region AgentsAboveAvarage
                            Label7.Text += average.ToString();
                            TaskUtils.AgentsAboveAvarage
                                (agents, average, Table2, fileName);
                            Label8.Text += suma.ToString();

                            #endregion

                            #region ChangeSubscriptions
                            int number = Convert.ToInt32(TextBox1.Text);
                            LinkedListAgents changed =
                                new LinkedListAgents();
                            bool valid = true;
                            subscriptions = TaskUtils.
                                ChangeSubscriptions(subscriptions,
                                agents, number, ref changed, ref valid);

                            changed.Start();
                            if (valid && changed.Exists())
                            {
                                changed.Sort();
                                TaskUtils.MakeSubscriptionList
                                (changed, subscriptions, fileName, month,
                                Table3);
                            }
```

```
                    else
                    {
                        InOutUtils.
                            NoAgentsAboveAmount(Table3);
                        InOutUtils.
                            NoAgentsAboveAmount(fileName);
                    }
                    #endregion
                }
                else
                {
                    InOutUtils.
                        NoSubscriptionsInMonth(Table1);
                    InOutUtils.
                        NoSubscriptionsInMonth(fileName);
                }
            }
            else
            {
                InOutUtils.
                    NoAgentsToTxt(fileName);
                InOutUtils.
                    NoAgentsToTable(Table1);
            }
        }
        else
        {
            InOutUtils.
                NoSubscriptionsToTxt(fileName);
            InOutUtils.
                NoSubscriptionsToTable(Table1);
        }
    }

}
void OutputAgentToTable(LinkedListAgents agents)
{
    DataTable dataTable = new DataTable();
    DataRow dataRow = null;
    dataTable.Columns.Add("Agento Kodas");
    dataTable.Columns.Add("Pavardė");
    dataTable.Columns.Add("Vardas");
    dataTable.Columns.Add("Adresas");
    dataTable.Columns.Add("Telefono numeris");

    for(agents.Start(); agents.Exists(); agents.Next())
    {
        dataRow = dataTable.NewRow();
        dataRow[0] = agents.Get().AgentCode;
        dataRow[1] = agents.Get().Surname;
        dataRow[2] = agents.Get().Name;
        dataRow[3] = agents.Get().Address;
        dataRow[4] = agents.Get().PhoneNumber;

        dataTable.Rows.Add(dataRow);
    }
    dataTable.AcceptChanges();
    GridView1.DataSource = dataTable;
    GridView1.DataBind();
}

void OutputSubscriptionsToTable(LinkedListSubscriptions subscriptions )
{
    DataTable dataTable = new DataTable();
    DataRow dataRow = null;
    dataTable.Columns.Add("Adresas");
    dataTable.Columns.Add("Pavardė");
```

```
            dataTable.Columns.Add("Telefono numeris");
            dataTable.Columns.Add("Laikotarpio pradžia");
            dataTable.Columns.Add("Laikotarpio ilgis");
            dataTable.Columns.Add("Leidinio kodas");
            dataTable.Columns.Add("Leidinių kiekis");
            dataTable.Columns.Add("Agento kodas");

            for (subscriptions.Start(); subscriptions.Exists(); subscriptions.Next())
            {
                dataRow = dataTable.NewRow();
                dataRow[0] = subscriptions.Get().Address;
                dataRow[1] = subscriptions.Get().Surname;
                dataRow[2] = subscriptions.Get().PhoneNumber;
                dataRow[3] = subscriptions.Get().Start;
                dataRow[4] = subscriptions.Get().Duration;
                dataRow[5] = subscriptions.Get().Code;
                dataRow[6] = subscriptions.Get().Amount;
                dataRow[7] = subscriptions.Get().AgentCode;

                dataTable.Rows.Add(dataRow);
            }
            dataTable.AcceptChanges();
            GridView2.DataSource = dataTable;
            GridView2.DataBind();
        }



    }
}
```

Style.cs

```
body {
    background-color:burlywood;
}
table {
    border-color:darkred;
    font-family:Arial,'Agency FB';
}
input{
    background-color:aquamarine;
    border:solid 1px black;
}
```

2.7. Pradiniai duomenys ir rezultatai

1 Variantas

Pradiniai duomenys:

U9a.txt

```
Vilnius, Taikos pr. 14-22;Jonaitis;864725962;5;11;24624;4;456
Kaunas, Rusu g. 1-41;Tomutis;864725962;3;6;31421;2;123
Šilutė, Laisvės al. 4-15;Stasytis;865214867;3;9;71425;5;123
Marijampolė, Savanorių g. 7-1;Vingytė;861542789;5;2;54712;1;123
Kaunas, Taikos g. 4-156;Jonavičius;862458731;5;4;25142;6;456
Klaipėda, Ratų g. 2-13;Antanovič;864157842;3;3;24167;3;897
Pajūralis, Gėlių g. 6-1;Višinskas;869574125;5;2;85412;4;789
Visaginas, Saulės pr. 2-2;Paršaitis;861247512;5;7;62145;9;456
Vilnius, Dainavos pr. 3-4;Mikužė;862459753;3;6;74551;2;654
Jonava, Medžių g. 6-6;Vilkė;865221455;5;8;62214;1;231
Zarasai, Marių g. 7-2;Pilkys;864471524;5;6;12458;3;897
Rokiškis, Vinčų pr. 3-1;Šveistakis;866957452;5;7;33512;4;897
Palanga, Basanavičiaus g. 9-1;Lingytė;862254117;5;5;14214;5;123
Juknaičiai, Pievų g. 5-5;Vingis;863536822;3;3;44215;7;654
Mažeikiai, Bendrovės pr. 16-4;Mažeikis;866744514;5;1;33235;7;456
Šiauliai, Geležinės Lapės pr. 2-45;Kalašnikovė;865952471;5;3;22325;4;231
Rokiškis, Pieno g. 2-31;Spindulys;862241458;5;1;77452;5;897
Vilnius, Šviesos pr. 7-12;Raudienė;863323511;3;7;44751;8;231
Klaipėda, Jūros pr. 5-21;Maksimavičienė;865565124;3;6;11758;2;123
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;5;1;99425;3;456
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;3;1;99425;3;789
```

U9b.txt

```
456;Tobutis;Tomas;Vilnius, Tiesos g. 52A;867854621
123;Rukis;Jonas;Kaunas, blablala 42;867854622
789;Bukis;Petras;Vilnius, Ruklos g. 2;867854623
654;Mukis;Lukas;Visaginas, Debesies g. 4;867854624
231;Tukis;Romas;Klaipėda, blablabla g. 2;867854625
897;Vukis;Ignas;Kaunas, bagsdgaghd g. 9;867854626
```

Rezultatai:

U9rez.txt

U9rez - Notepad

File Edit Format View Help

Pradiniai duomenys

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621
123	Rukis	Jonas	Kaunas, blablala 42	867854622
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Vilnius, Taikos pr. 14-22	Jonaitis	864725962	5	11	24624	4	456
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123
Marijampolė, Savanorių g. 7-1	Vingytė	861542789	5	2	54712	1	123
Kaunas, Taikos g. 4-156	Jonavičius	862458731	5	4	25142	6	456
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	897
Pajūralis, Gėlių g. 6-1	Višinskas	869574125	5	2	85412	4	789
Visaginas, Saulės pr. 2-2	Paršaitis	861247512	5	7	62145	9	456
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2	654
Jonava, Medžių g. 6-6	Vilkė	865221455	5	8	62214	1	231
Zarasai, Marių g. 7-2	Pilkys	864471524	5	6	12458	3	897
Rokiškis, Vinčų pr. 3-1	Šveistakis	866957452	5	7	33512	4	897
Palanga, Basanavičiaus g. 9-1	Lingytė	862254117	5	5	14214	5	123
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7	654
Mažeikiai, Bendrovės pr. 16-4	Mažeikis	866744514	5	1	33235	7	456
Šiauliai, Geležinės Lapės pr. 2-45	Kalašnikovė	865952471	5	3	22325	4	231
Rokiškis, Pieno g. 2-31	Spindulys	862241458	5	1	77452	5	897
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	5	1	99425	3	456
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Agentai, kuriem buvo pakeisti sąrašai:

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621

Prenumeratų nėra

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7	654
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2	654

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	897

Agentai, kuriem buvo pakeisti sąrašai:Agentai nešiojantys daugiau nei vidurkis:

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625

Agentai, kuriem buvo pakeisti sąrašai:

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	231
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123

Vartotojo sąsaja:

Mėnesis:
3

Nešiojamos prenumeratos kiekis:
3

Skaičiuoti

Pradiniai duomenys:

Agento Kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621
123	Rukis	Jonas	Kaunas, blablala 42	867854622
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Vilnius, Taikos pr. 14-22	Jonaitis	864725962	5	11	24624	4	456
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123
Marijampolė, Savanorių g. 7-1	Vingytė	861542789	5	2	54712	1	123
Kaunas, Taikos g. 4-156	Jonavičius	862458731	5	4	25142	6	456
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	897
Pajūralis, Gėlių g. 6-1	Višinskas	869574125	5	2	85412	4	789
Visaginas, Saulės pr. 2-2	Paršaitis	861247512	5	7	62145	9	456
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2	654
Jonava, Medžių g. 6-6	Vilkė	865221455	5	8	62214	1	231
Zarasai, Marių g. 7-2	Pilkys	864471524	5	6	12458	3	897
Rokiškis, Vinčų pr. 3-1	Šveistakis	866957452	5	7	33512	4	897
Palanga, Basanavičiaus g. 9-1	Lingytė	862254117	5	5	14214	5	123
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7	654
Mažeikiai, Bendrovės pr. 16-4	Mažeikis	866744514	5	1	33235	7	456
Šiauliai, Geležinės Lapės pr. 2-45	Kalašnikovė	865952471	5	3	22325	4	231
Rokiškis, Pieno g. 2-31	Spindulys	862241458	5	1	77452	5	897
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	5	1	99425	3	456
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Sąrašai agentam:

Informacija apie agentą:				
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621
Prenumeratų ši mėnesį agentas nenešioja				
-				
Informacija apie agentą:				
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622
Prenumeratų sąrašas:				
Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5
-						
Informacija apie agentą:				
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623

789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623		
Prenumeratų sąrašas:						
Adresas	Paverdė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3
-						
Informacija apie agentą:						
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris		
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624		
Prenumeratų sąrašas:						
Adresas	Paverdė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2
-						
Informacija apie agentą:						
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris		
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625		
Prenumeratų sąrašas:						
Adresas	Paverdė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8
-						
Informacija apie agentą:						
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris		
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626		
Prenumeratų sąrašas:						
Adresas	Paverdė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3
-						

Vidurkis: 5

Bendras leidiniu kiekis: 32
Agentų, kurie nešioja daugiau nei vidurkis, sąrašas:

Informacija apie agentą:				
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622
Informacija apie agentą:				
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624
Informacija apie agentą:				
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625

Agentai kuriem buvo pakeisti sąrašai:

Informacija apie agentą:						
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris		
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625		
Prenumeratų sąrašas:						
Adresas	Paverdė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8
-						
Informacija apie agentą:						
Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris		
123	Rukis	Jonas	Kaunas, blablala 42	867854622		
Prenumeratų sąrašas:						
Adresas	Paverdė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5
-						

2 Variantas – duomenų failai yra tušti

Pradiniai duomenys:

U9a_2.txt

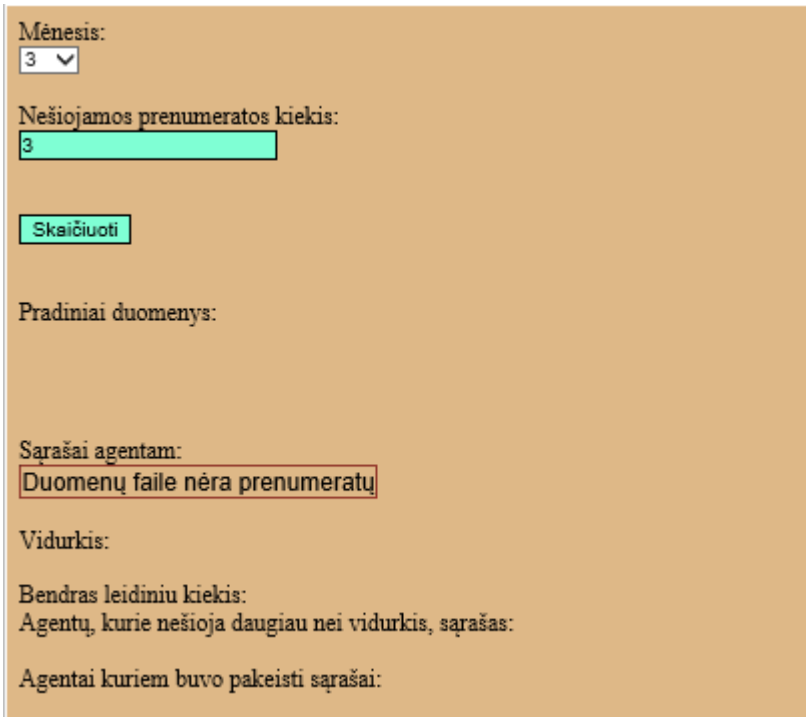
U9b_2.txt

Rezultatai:

U9rez.txt

```
Duomenų faile nėra prenumeratų
```

Vartotojo sąsaja:



Variantas 3 - pasirenkamas mėnuo kai yra nenešiojama prenumerata

U9a_3.txt

U9a_3 - Notepad

File Edit Format View Help

```
Vilnius, Taikos pr. 14-22;Jonaitis;864725962;5;11;24624;4;456
Kaunas, Rusu g. 1-41;Tomutis;864725962;3;6;31421;2;123
Šilutė, Laisvės al. 4-15;Stasytis;865214867;3;9;71425;5;123
Marijampolė, Savanorių g. 7-1;Vingytė;861542789;5;2;54712;1;123
Kaunas, Taikos g. 4-156;Jonavičius;862458731;5;4;25142;6;456
Klaipėda, Ratų g. 2-13;Antanovič;864157842;3;3;24167;3;897
Pajūralis, Gėlių g. 6-1;Višinskas;869574125;5;2;85412;4;789
Visaginas, Saulės pr. 2-2;Paršaitis;861247512;5;7;62145;9;456
Vilnius, Dainavos pr. 3-4;Mikužė;862459753;3;6;74551;2;654
Jonava, Medžių g. 6-6;Vilkė;865221455;5;8;62214;1;231
Zarasai, Marių g. 7-2;Pilkys;864471524;5;6;12458;3;897
Rokiškis, Vinčų pr. 3-1;Šveistakis;866957452;5;7;33512;4;897
Palanga, Basanavičiaus g. 9-1;Lingytė;862254117;5;5;14214;5;123
Juknaičiai, Pievų g. 5-5;Vingis;863536822;3;3;44215;7;654
Mažeikiai, Bendrovės pr. 16-4;Mažeikis;866744514;5;1;33235;7;456
Šiauliai, Geležinės Lapės pr. 2-45;Kalašnikovė;865952471;5;3;22325;4;231
Rokiškis, Pieno g. 2-31;Spindulys;862241458;5;1;77452;5;897
Vilnius, Šviesos pr. 7-12;Raudienė;863323511;3;7;44751;8;231
Klaipėda, Jūros pr. 5-21;Maksimavičienė;865565124;3;6;11758;2;123
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;5;1;99425;3;456
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;3;1;99425;3;789
```

U9b_3.txt

U9b_3 - Notepad

File Edit Format View Help

```
456;Tobutis;Tomas;Vilnius, Tiesos g. 52A;867854621
123;Rukis;Jonas;Kaunas, blablala 42;867854622
789;Bukis;Petras;Vilnius, Ruklos g. 2;867854623
654;Mukis;Lukas;Visaginas, Debesies g. 4;867854624
231;Tukis;Romas;Klaipėda, blablabla g. 2;867854625
897;Vukis;Ignas;Kaunas, bagsdgaghd g. 9;867854626
```

Rezultatai:
U9rez.txt

U9rez - Notepad

File Edit Format View Help

Pradiniai duomenys

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621
123	Rukis	Jonas	Kaunas, blablala 42	867854622
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Vilnius, Taikos pr. 14-22	Jonaitis	864725962	5	11	24624	4	456
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123
Marijampolė, Savanorių g. 7-1	Vingytė	861542789	5	2	54712	1	123
Kaunas, Taikos g. 4-156	Vingytė	862458731	5	4	25142	6	456
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	897
Pajūralis, Gėlių g. 6-1	Višinskas	869574125	5	2	85412	4	789
Visaginas, Saulės pr. 2-2	Paršaitis	861247512	5	7	62145	9	456
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2	654
Jonava, Medžių g. 6-6	Vilkė	865221455	5	8	62214	1	231
Zarasai, Marių g. 7-2	Pilkys	864471524	5	6	12458	3	897
Rokiškis, Vinčų pr. 3-1	Šveistakis	866957452	5	7	33512	4	897
Palanga, Basanavičiaus g. 9-1	Lingytė	862254117	5	5	14214	5	123
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7	654
Mažeikiai, Bendrovės pr. 16-4	Mažeikis	866744514	5	1	33235	7	456
Šiauliai, Geležinės Lapės pr. 2-45	Kalašnikovė	865952471	5	3	22325	4	231
Rokiškis, Pieno g. 2-31	Spindulys	862241458	5	1	77452	5	897
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	5	1	99425	3	456
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Ši mėnesį agentai nenešioja prenumeratos

Vartotojo sąsaja:

Mėnesis:
12

Nešiojamos prenumeratos kiekis:
5

Skaičiuoti

Pradiniai duomenys:

Agento Kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621
123	Rukis	Jonas	Kaunas, blablala 42	867854622
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Vilnius, Taikos pr. 14-22	Jonaitis	864725962	5	11	24624	4	456
Kaunas, Rusų g. 1-41	Tomutis	864725962	3	6	31421	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123
Marijampolė, Savanorių g. 7-1	Vingytė	861542789	5	2	54712	1	123
Kaunas, Taikos g. 4-156	Jonavičius	862458731	5	4	25142	6	456
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	897
Pajūralis, Gėlių g. 6-1	Višinskas	869574125	5	2	85412	4	789
Visaginas, Saulės pr. 2-2	Paršaitis	861247512	5	7	62145	9	456
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2	654
Jonava, Medžių g. 6-6	Vilkė	865221455	5	8	62214	1	231
Zarasai, Marių g. 7-2	Pilkys	864471524	5	6	12458	3	897
Rokiškis, Vinčų pr. 3-1	Šveistakis	866957452	5	7	33512	4	897
Palanga, Basanavičiaus g. 9-1	Lingytė	862254117	5	5	14214	5	123
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7	654
Mažeikiai, Bendrovės pr. 16-4	Mažeikis	866744514	5	1	33235	7	456
Šiauliai, Geležinės Lapės pr. 2-45	Kalašnikovė	865952471	5	3	22325	4	231
Rokiškis, Pieno g. 2-31	Spindulys	862241458	5	1	77452	5	897
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	5	1	99425	3	456
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Sąrašai agentam:
Šį mėnesį žmonės neužsisakinėja prenumeratų

Vidurkis:

Bendras leidiniu kiekis:
Agentų, kurie nešioja daugiau nei vidurkis, sąrašas:

Agentai kuriem buvo pakeisti sąrašai:

2.8. Dėstytojo pastabos

3. Bendrinės klasės ir testavimas (L3)

3.1. Darbo užduotis

LD_9. **Prenumerata.** Žmonės užsisako spaudą. Užsakymas vyksta metų ribose. Užsisakant spaudą, priskiriamas agentas, kuris pristatys užsakytą spaudą į namus. Tuo tikslu agentai gauna prenumeratorių sąrašus, kurie tikslinami kiekvieną mėnesį. Sudarykite kiekvienam agentui nurodyto mėnesio (įvedamas klaviatūra) nešiojamos prenumeratos sąrašą (prenumeratoriaus adresas, pavardė, telefono numeris, laikotarpio pradžia, laikotarpio ilgis, leidinio kodas, leidinių kiekis) ir suskaičiuokite bendrą leidinių kiekį. Agento sąrašą surikiuokite pagal prenumeratoriaus adresą ir pavardę. Sudarykite sąrašą agentų, kurie nešioja daugiau nei vidutinis kiekis nurodytam mėnesiui.

Duomenys:

- Tekstiniame faile U9a.txt yra tokia informacija apie prenumeratorius: prenumeratoriaus adresas, pavardė, telefono numeris, laikotarpio pradžia (sveikasis skaičius 1..12), laikotarpio ilgis (sveikasis skaičius 1..12), leidinio kodas, leidinių kiekis, agento kodas.
- Tekstiniame faile U9b.txt yra informacija apie agentus: agento kodas, pavardė, vardas, adresas, telefonas.

Jei yra agentų, kurių nešiojamos prenumeratos kiekis neviršija duotam mėnesiui nurodyto (įvedamas klaviatūra), pašalinkite juos iš sąrašo, jų nešiojamą prenumeratą paskirstydami kitiems agentams. Priskyrimo idėja tokia: skirkite tam agentui, kurio kiekis viršija nurodytą, bet yra mažiausias tarp agentų, tenkinančių šią sąlygą. Atspausdinkite sąrašus agentams, kuriems pasikeitė nešiojamos prenumeratos kiekiai.

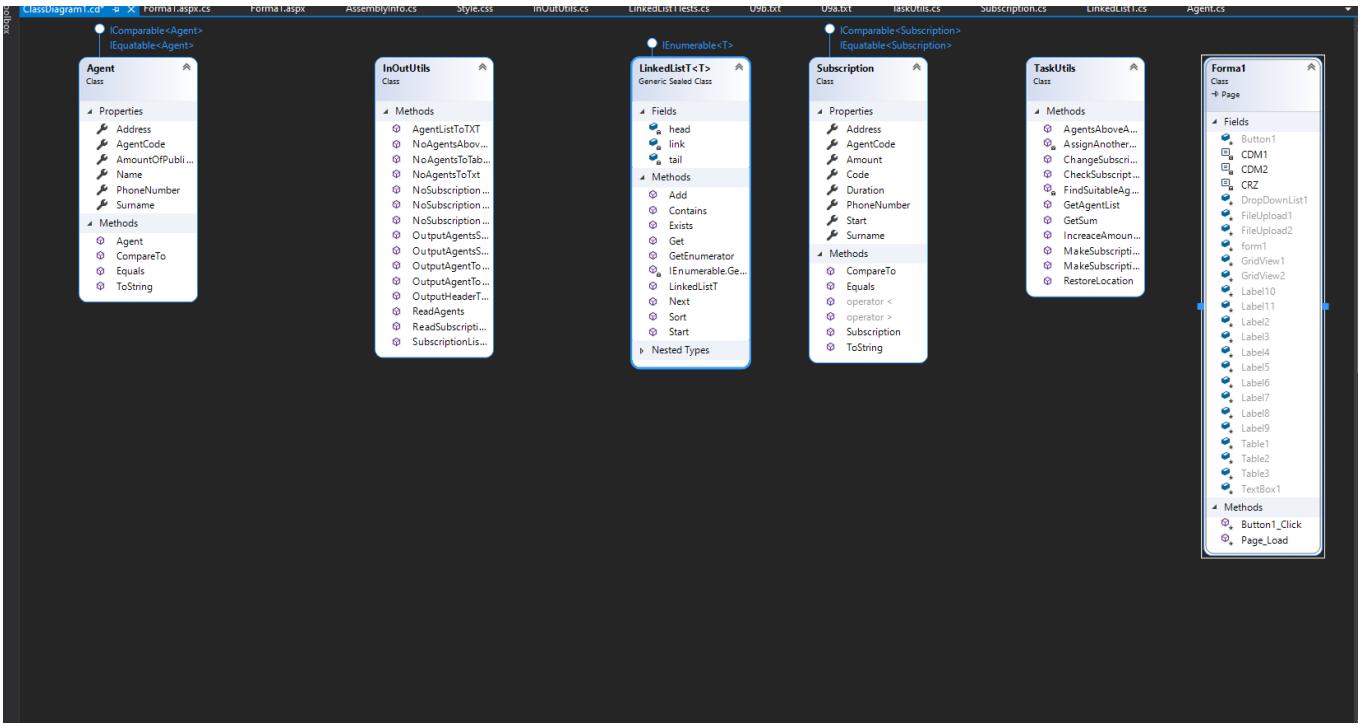
Pastaba. Laikotarpio pradžios ir laikotarpio ilgio reikšmės turi tenkinti sąlygą: laikotarpio pradžia + laikotarpio ilgis ≤ 13.

3.2. Grafinės vartotojo sąsajos schema

3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button1	Text	Skaičiuoti
Label8	Text	Pasirinketa duomenų failą, kuriame yra prenumeratos:
Label9	Text	Pasirinketa duomenų failą, kuriame yra agentai:
Label4	Text	Mėnesis:
Label5	Text	Nešiojamos prenumeratos kiekis:
Label10	Text	Pradiniai duomenys:
Label2	Text	Sąrašai agentam:
Label7	Text	Vidurkis:
Label3	Text	Agentų, kurie nešioja daugiau nei vidurkis,sąrašas:
Label6	Text	Agentai, kuriem buvo pakeisti sąrašai
Table1	GridLines	Both
Table2	GridLines	Both

3.4. Klasių diagrama



3.5. Programos vartotojo vadovas

Prenumeratų sąrašą reikia pasirinkti tam skirte laukelyje po užrašu „Pasirinkite duomenų failą, kuriame yra prenumeratos, duomenų formatas turėtu būti toks: prenumeratoriaus adresas;pavardė;telefono numeris;laikotarpio pradžia;laikotarpio ilgis; leidinio kodas;leidinių kiekis;agento kodas.

Agentų sąrašą reikia pasirinkti tam skirte laukelyje po užrašu „Pasirinkite duomenų failą, kuriame yra agentai“, jo duomenų formatas turėtu būti toks: agento kodas;pavardė;vardas;adresas;telefonas.

Paleidus programą reikia pasirinkti mėnesį, kurio prenumeratų sąrašus norite matyti ir į teksto lauką įvesti skaičių (prenumeratų kiekį), jei į teksto lauką galima vesti tik skaičius, kitu atveju niekas nevyks.

3.6. Programos tekstas

Agent.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public class Agent :
        IComparable<Agent>,IEquatable<Agent>
    {
        /// <summary>
        /// Parameters of object
        /// </summary>
        public int AgentCode { get; set; }
        public string Surname { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string PhoneNumber { get; set; }
        public int AmountOfPublications { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="agentCode">agent code</param>
        /// <param name="surname">surname</param>
        /// <param name="name">name</param>
        /// <param name="address">address</param>
        /// <param name="phoneNumber">phone number</param>
        public Agent(int agentCode, string surname, string name,
            string address, string phoneNumber)
        {
            AgentCode = agentCode;
            Surname = surname;
            Name = name;
            Address = address;
```

```
            PhoneNumber = phoneNumber;
            AmountOfPublications = -1;
        }

        /// <summary>
        /// Override for ToString method
        /// </summary>
        /// <returns>string</returns>
        public override string ToString()
        {
            return string.Format("| {0, 12} | {1, -15} | {2, -15}" +
                " | {3, -30} | | {4,16} |", AgentCode, Surname,
                Name, Address, PhoneNumber);
        }

        public int CompareTo(Agent obj)
        {
            if (this.Name.CompareTo(obj.Name) == 0
                && this.Surname.CompareTo(obj.Name) == 0)
                return 0;
            return this.Surname.CompareTo(obj.Surname);
        }

        public bool Equals(Agent obj)
        {
            if (this.Name.CompareTo(obj.Name) == 0 &&
                this.Surname.CompareTo(obj.Surname) == 0)
                return true;
            else
                return false;
        }
    }
}
```

Subscription.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public class Subscription :
        IComparable<Subscription>,
        IEquatable<Subscription>
    {
        /// <summary>
        /// Parameters of object
        /// </summary>
        public string Address { get; set; }
        public string Surname { get; set; }
        public string PhoneNumber { get; set; }
        public int Start { get; set; }
        public int Duration { get; set; }
        public int Code { get; set; }
        public int Amount { get; set; }
        public int AgentCode { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="address">address</param>
        /// <param name="surname">surname</param>
        /// <param name="phoneNumber">phone number</param>
        /// <param name="start">start of period</param>
        /// <param name="duration">duration of period</param>
        /// <param name="code">code of publication</param>
```

```
/// <param name="amount">amount of publications</param>
/// <param name="agentCode">agent code</param>
public Subscription(string address, string surname,
    string phoneNumber, int start, int duration,
    int code, int amount, int agentCode)
{
    Address = address;
    Surname = surname;
    PhoneNumber = phoneNumber;
    Start = start;
    Duration = duration;
    Code = code;
    Amount = amount;
    AgentCode = agentCode;
}

/// <summary>
/// override for > operator
/// </summary>
/// <param name="left">one subscription</param>
/// <param name="right">another subscription</param>
/// <returns>true or false</returns>
public static bool operator > (Subscription left, Subscription right)
{
    if (left.Address.CompareTo(right.Address) == 0)
        return left.Surname.CompareTo(right.Surname) >0;
    else
    {
        return left.Address.CompareTo(right.Address) >0;
    }
}

/// <summary>
/// override for < operator
/// </summary>
/// <param name="left">one subscription</param>
/// <param name="right">another subscription</param>
/// <returns>true or false</returns>
public static bool operator <(Subscription left, Subscription right)
{
    if (left.Address.CompareTo(right.Address) == 0)
        return left.Surname.CompareTo(right.Surname) < 0;
    else
    {
        return left.Address.CompareTo(right.Address) < 0;
    }
}

/// <summary>
/// override for ToString method
/// </summary>
/// <returns>string</returns>
public override string ToString()
{
    return string.Format("| {0, -30} | {1, -15} | {2, 16} |" +
        " {3, 20} | {4, 20} | {5, 15} | {6, 16} | {7, 12} |",
        Address, Surname, PhoneNumber, Start, Duration, Code,
        Amount, AgentCode);
}

public int CompareTo(Subscription sub)
{
    if (this.Address.CompareTo(sub.Address)==0)
        return this.Surname.CompareTo(sub.Surname);
    else
        return this.Address.CompareTo(sub.Address);
}
```

```
        public bool Equals(Subscription sub)
        {
            if (this.Surname.CompareTo(sub.Surname) == 0)
                return true;
            else
                return false;
        }
    }


}
```

InOutUtils.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Data;
namespace LD_9_Prenumerata.App_BarCode
{
    public class InOutUtils
    {
        /// <summary>
        /// Reads subscriptions from file
        /// </summary>
        /// <param name="FileName">name of file</param>
        /// <returns>subscriptions</returns>
        public static LinkedListT<Subscription> ReadSubscriptions
            (Stream FileName)
        {
            LinkedListT<Subscription> subscriptions
                = new LinkedListT<Subscription>();
            using (StreamReader reader =
                new StreamReader(FileName, Encoding.UTF8))
            {
                string line;
                while ((line = reader.ReadLine()) != null)
                {
                    string[] values = line.Split(';');
                    string address = values[0];
                    string surname = values[1];
                    string phonenumber = values[2];
                    int start = Convert.ToInt32(values[3]);
                    int duration = Convert.ToInt32(values[4]);
                    int code = Convert.ToInt32(values[5]);
                    int amount = Convert.ToInt32(values[6]);
                    int agentCode = Convert.ToInt32(values[7]);
                    subscriptions.Add(new Subscription(address,
                        surname, phonenumber, start, duration,
                        code, amount, agentCode));
                }
            }
            return subscriptions;
        }

        /// <summary>
        /// Reads agents from file
        /// </summary>
        /// <param name="FileName">name of file</param>
        /// <returns>agents</returns>
        public static LinkedListT<Agent> ReadAgents
```

```
    (Stream FileName)
{
    LinkedListT<Agent> agents =
        new LinkedListT<Agent>();
    using(StreamReader reader =
        new StreamReader(FileName,Encoding.UTF8))
    {
        string line;
        while (( line = reader.ReadLine()) != null)
        {
            string[] values = line.Split(';');
            int agentcode = Convert.ToInt32(values[0]);
            string surname = values[1];
            string name = values[2];
            string address = values[3];
            string phonenumber = values[4];
            agents.Add(new Agent(agentcode,
                surname, name, address, phonenumber));
        }
    }
    return agents;
}

/// <summary>
/// Outputs agent and his subscriptions to txt file
/// </summary>
/// <param name="FileName">name of file</param>
/// <param name="agent">agent</param>
/// <param name="subscriptions">agent's subscriptions</param>
public static void OutputAgentsSubscriptionsToTxt
    (string FileName, Agent agent,
    IEnumerable<Subscription> subscriptions)
{
    using (StreamWriter writer =
        new StreamWriter(FileName,append: true))
    {
        writer.WriteLine("");
        writer.WriteLine(new string('-', 106));
        writer.WriteLine(string.Format("| {0, -12}" +
            " | {1, -15} | {2, -15} | {3, -30} | |" +
            " {4,-16} |", "Agento kodas", "Pavardė",
            "Vardas", "Adresas", "Telefono numeris"));
        writer.WriteLine(agent.ToString());
        writer.WriteLine(new string('-', 106));
        if (subscriptions.Any())
        {
            writer.WriteLine("Prenumeratos:");
            writer.WriteLine(new string('-', 169));
            writer.WriteLine(string.Format
                ("| {0, -30} | {1, -15} | {2, 16}" +
                " | {3, 20} | {4, 20} | {5, 15} |" +
                " {6, 16} | {7, 12} |", "Adresas",
                "Pavardė", "Telefono numeris",
                "Laikotarpio pradžia",
                "Laikotarpio ilgis", "Leidinio kodas",
                "Leidinių kiekis", "Agento kodas"));

            foreach(Subscription sub in subscriptions)
            {
                writer.WriteLine(sub.ToString());
            }
            writer.WriteLine(new string('-', 169));
        }
        else
        {
            writer.WriteLine("Prenumeratų nėra");
        }
```

```
        }
    }

    /// <summary>
    /// Outputs agent and his subscriptions to table
    /// </summary>
    /// <param name="table">table</param>
    /// <param name="agent">agent</param>
    /// <param name="subscriptions">agent's subscription</param>
    public static void OutputAgentsSubscriptionsToTable
        (Table table, Agent agent,
        IEnumerable<Subscription> subscriptions)
    {
        TableCell information = new TableCell();
        TableRow inform = new TableRow();
        information.Text = "Informacija apie agentą: ";

        inform.Cells.Add(information);
        table.Rows.Add(inform);

        TableCell tcell1 = new TableCell();
        TableCell tcell2 = new TableCell();
        TableCell tcell3 = new TableCell();
        TableCell tcell4 = new TableCell();
        TableCell tcell5 = new TableCell();
        TableCell tcell6 = new TableCell();
        TableCell tcell7 = new TableCell();
        TableRow explination = new TableRow();
        tcell1.Text = "Agento kodas";
        tcell2.Text = "Pavardė";
        tcell3.Text = "Vardas";
        tcell4.Text = "Adresas";
        tcell5.Text = "Telefono numeris";

        explination.Cells.Add(tcell1);
        explination.Cells.Add(tcell2);
        explination.Cells.Add(tcell3);
        explination.Cells.Add(tcell4);
        explination.Cells.Add(tcell5);
        table.Rows.Add(explination);

        TableRow AgentInfo = new TableRow();

        TableCell agentcode = new TableCell();
        agentcode.Text = agent.AgentCode.ToString();

        TableCell surname = new TableCell();
        surname.Text = agent.Surname;

        TableCell name = new TableCell();
        name.Text = agent.Name;

        TableCell address = new TableCell();
        address.Text = agent.Address.ToString();

        TableCell phone = new TableCell();
        phone.Text = agent.PhoneNumber.ToString();

        AgentInfo.Cells.Add(agentcode);
        AgentInfo.Cells.Add(surname);
        AgentInfo.Cells.Add(name);
        AgentInfo.Cells.Add(address);
        AgentInfo.Cells.Add(phone);

        table.Rows.Add(AgentInfo);
        if (subscriptions.Any())
        {
```

```
        TableCell buff = new TableCell();
        buff.Text = "Prenumeratų sąrašas:";
        TableRow buff1 = new TableRow();
        buff1.Cells.Add(buff);
        table.Rows.Add(buff1);

        TableCell cell1 = new TableCell();
        TableCell cell2 = new TableCell();
        TableCell cell3 = new TableCell();
        TableCell cell4 = new TableCell();
        TableCell cell5 = new TableCell();
        TableCell cell6 = new TableCell();
        TableCell cell7 = new TableCell();

        cell1.Text = "Adresas";
        cell2.Text = "Paverdė";
        cell3.Text = "Telefono numeris";
        cell4.Text = "Laikotarpio pradžia";
        cell5.Text = "Laikotarpio ilgis";
        cell6.Text = "Leidinio kodas";
        cell7.Text = "Leidinių kiekis";

        TableRow row = new TableRow();
        row.Cells.Add(cell1);
        row.Cells.Add(cell2);
        row.Cells.Add(cell3);
        row.Cells.Add(cell4);
        row.Cells.Add(cell5);
        row.Cells.Add(cell6);
        row.Cells.Add(cell7);

        table.Rows.Add(row);

        foreach(Subscription subscription in subscriptions)
        {
            TableCell sub1 = new TableCell();
            TableCell sub2 = new TableCell();
            TableCell sub3 = new TableCell();
            TableCell sub4 = new TableCell();
            TableCell sub5 = new TableCell();
            TableCell sub6 = new TableCell();
            TableCell sub7 = new TableCell();
            TableRow subrow = new TableRow();

            sub1.Text = subscription.Address;
            sub2.Text = subscription.Surname;
            sub3.Text = subscription.PhoneNumber;
            sub4.Text = subscription.Start.ToString();
            sub5.Text = subscription.Duration.ToString();
            sub6.Text = subscription.Code.ToString();
            sub7.Text = subscription.Amount.ToString();

            subrow.Cells.Add(sub1);
            subrow.Cells.Add(sub2);
            subrow.Cells.Add(sub3);
            subrow.Cells.Add(sub4);
            subrow.Cells.Add(sub5);
            subrow.Cells.Add(sub6);
            subrow.Cells.Add(sub7);

            table.Rows.Add(subrow);
        }
    }
    else
    {
        TableCell output = new TableCell();
        output.Text = "Prenumeratų ši mėnesį agentas nenešioja";
```

```
                TableRow outputt = new TableRow();
                outputt.Cells.Add(output);
                table.Rows.Add(outputt);
            }
            TableCell Space = new TableCell();
            Space.Text = "-";
            TableRow space = new TableRow();
            space.Cells.Add(Space);
            table.Rows.Add(space);
        }

        public static void AgentListToTXT(IEnumerable<Agent> agents, string path, string header)
        {
            List<string> lines = new List<string>();
            string line = new string('-', 106);
            lines.Add(header);
            lines.Add(line);
            lines.Add(string.Format("| {0, 12} | {1, -15} | {2, -15}" +
                " | {3, -30} | | {4,16} |", "Agento kodas", "Pavardė",
                "Vardas", "Adresas", "Telefono numeris"));
            lines.Add(line);
            foreach (Agent agent in agents)
            {
                lines.Add(agent.ToString());
                lines.Add(line);
            }

            lines.Add("");
            lines.Add("");

            File.AppendAllLines(path, lines);
        }

        public static void SubscriptionListToTXT(IEnumerable<Subscription> subscriptions, string
path, string header)
        {
            List<string> lines = new List<string>();
            string line = new string('-', 169);
            lines.Add(header);
            lines.Add(line);
            lines.Add(string.Format
                        ("| {0, -30} | {1, -15} | {2, 16}" +
                        " | {3, 20} | {4, 20} | {5, 15} |" +
                        " {6, 16} | {7, 12} |", "Adresas",
                        "Pavardė", "Telefono numeris",
                        "Laikotarpio pradžia",
                        "Laikotarpio ilgis", "Leidinio kodas",
                        "Leidinių kiekis", "Agento kodas"));
            lines.Add(line);
            foreach (Subscription subscription in subscriptions)
            {
                lines.Add(subscription.ToString());
                lines.Add(line);
            }

            lines.Add("");
            lines.Add("");

            File.AppendAllLines(path, lines);
        }



        /// <summary>
        /// Outputs agent to txt file
        /// </summary>
        /// <param name="fileName">name of file</param>
```

```
/// <param name="agent">agent</param>
public static void OutputAgentToTxt(string fileName, Agent agent)
{
    using (StreamWriter writer = new StreamWriter(fileName, append: true))
    {
        writer.WriteLine("");
        writer.WriteLine(new string('-', 106));
        writer.WriteLine(string.Format("| {0, -12}" +
            " | {1, -15} | {2, -15} | {3, -30} |" +
            " | {4,-16} |", "Agento kodas",
            "Pavardė", "Vardas", "Adresas",
            "Telefono numeris"));
        writer.WriteLine(agent.ToString());
        writer.WriteLine(new string('-', 106));
    }
}

/// <summary>
/// Outputs agent to table
/// </summary>
/// <param name="table">table</param>
/// <param name="agent">agent</param>
public static void OutputAgentToTable(Table table, Agent agent)
{
    TableCell information = new TableCell();
    TableRow inform = new TableRow();
    information.Text = "Informacija apie agentą: ";

    inform.Cells.Add(information);
    table.Rows.Add(inform);

    TableCell tcell1 = new TableCell();
    TableCell tcell2 = new TableCell();
    TableCell tcell3 = new TableCell();
    TableCell tcell4 = new TableCell();
    TableCell tcell5 = new TableCell();
    TableCell tcell6 = new TableCell();
    TableCell tcell7 = new TableCell();
    TableRow explination = new TableRow();
    tcell1.Text = "Agento kodas";
    tcell2.Text = "Pavardė";
    tcell3.Text = "Vardas";
    tcell4.Text = "Adresas";
    tcell5.Text = "Telefono numeris";

    explination.Cells.Add(tcell1);
    explination.Cells.Add(tcell2);
    explination.Cells.Add(tcell3);
    explination.Cells.Add(tcell4);
    explination.Cells.Add(tcell5);
    table.Rows.Add(explination);

    TableRow AgentInfo = new TableRow();

    TableCell agentcode = new TableCell();
    agentcode.Text = agent.AgentCode.ToString();

    TableCell surname = new TableCell();
    surname.Text = agent.Surname;

    TableCell name = new TableCell();
    name.Text = agent.Name;

    TableCell address = new TableCell();
    address.Text = agent.Address.ToString();

    TableCell phone = new TableCell();
```

```
        phone.Text = agent.PhoneNumber.ToString();

        AgentInfo.Cells.Add(agentcode);
        AgentInfo.Cells.Add(surname);
        AgentInfo.Cells.Add(name);
        AgentInfo.Cells.Add(address);
        AgentInfo.Cells.Add(phone);

        table.Rows.Add(AgentInfo);
    }

    /// <summary>
    /// Outputs that no agents are in txt file to txt file
    /// </summary>
    /// <param name="fileName">name of file</param>
    public static void NoAgentsToTxt(string fileName)
    {
        File.AppendAllText(fileName, "Duomenų faile nėra agentų");
    }

    /// <summary>
    /// Outputs that no agents are in txt file to table
    /// </summary>
    /// <param name="table">table</param>
    public static void NoAgentsToTable(Table table)
    {
        TableCell cell = new TableCell();
        cell.Text = "Duomenų faile nėra agentų";
        TableRow row = new TableRow();
        row.Cells.Add(cell);
        table.Rows.Add(row);
    }

    /// <summary>
    /// Outputs that no subscriptions are in txt file to txt
    /// </summary>
    /// <param name="fileName">name of file</param>
    public static void NoSubscriptionsToTxt
        (string fileName)
    {
        File.AppendAllText(fileName,
            "Duomenų faile nėra prenumeratų");
    }

    /// <summary>
    /// Outputs that no subscriptions are in txt file to table
    /// </summary>
    /// <param name="table">table</param>
    public static void NoSubscriptionsToTable(Table table)
    {
        TableCell cell = new TableCell();
        cell.Text = "Duomenų faile nėra prenumeratų";
        TableRow row = new TableRow();
        row.Cells.Add(cell);
        table.Rows.Add(row);
    }

    /// <summary>
    /// Outputs that no agent has subscriptions above given number to table
    /// </summary>
    /// <param name="table">table</param>
    public static void NoAgentsAboveAmount(Table table)
    {
        TableCell cell = new TableCell();
        cell.Text = "Duomenų faile nėra agentų," +
            " kurie nešiotų didesnį leidinių kiekį" +
            " nei nurodyta";
```

```
            TableRow row = new TableRow();
            row.Cells.Add(cell);
            table.Rows.Add(row);
        }

        /// <summary>
        /// Outputs that no agent has subscriptions above given number to txt
        /// </summary>
        /// <param name="fileName">name of file</param>
        public static void NoAgentsAboveAmount(string fileName)
        {
            File.AppendAllText(fileName, "Nei vienas" +
                " agentas nenešioja daugiau prenumeratos" +
                " nei nurodyta");
        }

        /// <summary>
        /// Outputs that no subscriptions are carried in that month to table
        /// </summary>
        /// <param name="table">table</param>
        public static void NoSubscriptionsInMonth(Table table)
        {
            TableCell cell = new TableCell();
            cell.Text = "Šį mėnesį žmonės neužsisakinėja" +
                " prenumeratų";
            TableRow row = new TableRow();
            row.Cells.Add(cell);
            table.Rows.Add(row);
        }

        /// <summary>
        /// Outputs that no subscriptions are carried in that month to txt
        /// </summary>
        /// <param name="fileName">name of file</param>
        public static void NoSubscriptionsInMonth(string fileName)
        {
            File.AppendAllText(fileName, "Ši mėnesį" +
                " agentai nenešioja prenumeratos");
        }

        /// <summary>
        /// OutputsHeaderToTxt
        /// </summary>
        /// <param name="fileName">name of file</param>
        /// <param name="header">header</param>
        public static void OutputHeaderTotxt(string fileName, string header)
        {
            File.AppendAllText(fileName, header);
        }
    }

}
```

LinkedListT.cs

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

namespace LD_9_Prenumerata.App_BarCode
{
    public sealed class LinkedListT<T> :IEnumerable<T> where T : IComparable<T>, IEquatable<T>
    {
        private sealed class NodeT <T>
        {
            public T Data { get; set; }
```

```
        public NodeT<T> Link { get; set; }

        public NodeT() { }

        public NodeT(T obj, NodeT<T> link)
        {
            Data = obj;
            Link = link;
        }
    }

    private NodeT<T> head;
    private NodeT<T> tail;
    private NodeT<T> link;

    public IEnumerator<T> GetEnumerator()
    {
        for(NodeT<T> dd = head; dd!= null; dd=dd.Link)
        {
            yield return dd.Data;
        }
    }

    IEnumerator IEnumerable.GetEnumerator()
    {
        return GetEnumerator();
    }

    public LinkedListT()
    {
        this.head = null;
        this.tail = null;
        this.link = null;
    }

    public bool IsEmpty() { return head == null; }

    public void Start()
    {
        link = head;
    }

    public void Next()
    {
        link = link.Link;
    }

    public bool Exists()
    {
        return link != null;
    }

    public T Get()
    {
        return link.Data;
    }

    public void Add(T data)
    {
        var dd = new NodeT<T>(data, null);
        if(head!=null)
        {
            tail.Link = dd;
            tail = dd;
        }
        else
        {
```

```
                head = dd;
                tail = dd;
            }
        }

        public bool Contains (T data)
        {
            for (this.Start(); this.Exists(); this.Next())
            {
                if (this.Get().CompareTo(data) == 0)
                    return true;
            }
            return false;
        }

        public void Remove(Agent agent)
        {
            if (head == null) return;

            if (head.Data.Equals(agent))
            {
                head = head.Link;
                return;
            }

            for (NodeT<T> n = head; n.Link != null; n = n.Link)
            {
                if (n.Link.Data.Equals(agent))
                {
                    n.Link = n.Link.Link;
                    return;
                }
            }
        }

        public void Sort()
        {
            for(NodeT<T> d1 = head; d1!= null; d1= d1.Link)
            {
                NodeT<T> maxv = d1;
                for(NodeT<T> d2 = d1; d2!=null; d2=d2.Link)
                {
                    if (d2.Data.CompareTo(maxv.Data) == -1)
                        maxv = d2;
                }
                T St = d1.Data;
                d1.Data = maxv.Data;
                maxv.Data = St;
            }
        }
    }
}
```

TaskUtils.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
namespace LD_9_Prenumerata.App_BarCode
{
    public class TaskUtils
    {
        /// <summary>
        /// Gets subscription list to agents in specific month
```

```
/// </summary>
/// <param name="month">month</param>
/// <param name="subscriptions">subscriptions</param>
/// <param name="agent">agents</param>
/// <returns>subscriptions</returns>
public static LinkedListT<Subscription> GetAgentList
    (int month, LinkedListT<Subscription> subscriptions, Agent agent)
{
    int agentCode = agent.AgentCode;
    LinkedListT<Subscription> subs =
        new LinkedListT<Subscription>();
    for (subscriptions.Start();
        subscriptions.Exists();
        subscriptions.Next())
    {
        Subscription sub = subscriptions.Get();
        if (sub.AgentCode == agentCode &&
            sub.Start == month)
            subs.Add(sub);
    }
    subs.Sort();
    return subs;
}

/// <summary>
/// Checks if there is any subscriptions in month
/// </summary>
/// <param name="month">month</param>
/// <param name="subs">subscriptions</param>
/// <returns>true or false</returns>
public static bool CheckSubscriptionsInMonth
    (int month, LinkedListT<Subscription> subs)
{
    for(subs.Start();subs.Exists();subs.Next())
    {
        if (subs.Get().Start == month)
            return true;
    }
    return false;
}

/// <summary>
/// Gets sum of carried subscriptions
/// </summary>
/// <param name="sub">subscriptions</param>
/// <returns>sum</returns>
public static int GetSum(LinkedListT<Subscription> sub)
{
    int sum = 0;
    for (sub.Start();sub.Exists();sub.Next())
    {
        sum += sub.Get().Amount;
    }
    return sum;
}

/// <summary>
/// Changes subscription for agents
/// </summary>
/// <param name="subs">subscriptions</param>
/// <param name="agents">agents</param>
/// <param name="number">number</param>
/// <param name="changed">list of agents whose subscriptions changed</param>
/// <param name="valid">valid</param>
/// <returns>subscriptions</returns>
public static LinkedListT<Subscription> ChangeSubscriptions
    (LinkedListT<Subscription> subs, LinkedListT<Agent> agents,
```

```
    int number, ref LinkedListT<Agent> changed, ref bool valid)
{
    int position = 0;
    for (agents.Start(); agents.Exists(); agents.Next())
    {
        if(agents.Get().AmountOfPublications<=number
            && agents.Get().AmountOfPublications > 0)
        {
            for(subs.Start();subs.Exists();subs.Next())
            {
                if (subs.Get().AgentCode ==
                    agents.Get().AgentCode)
                    subs.Get().AgentCode = -1;
            }
            agents.Remove(agents.Get());

            subs = AssignAnotherAgent
                (subs, agents, number, changed,ref valid);
            RestoreLocation(agents, position);
        }
        position++;
    }
    return subs;
}

/// <summary>
/// Assigns another agent
/// </summary>
/// <param name="subs">subscriptions</param>
/// <param name="agents">agents</param>
/// <param name="number">number</param>
/// <param name="changed">changed</param>
/// <param name="valid">valid</param>
/// <returns>subscriptions</returns>
private static LinkedListT<Subscription> AssignAnotherAgent
    (LinkedListT<Subscription> subs, LinkedListT<Agent> agents,
    int number, LinkedListT<Agent> changed, ref bool valid)
{
    Agent agent = FindSuitableAgent(agents, number);
    if (agent != null)
    {
        for (subs.Start(); subs.Exists(); subs.Next())
        {
            if (subs.Get().AgentCode == -1)
            {
                subs.Get().AgentCode = agent.AgentCode;
                IncreaceAmountOfPublications
                    (agents, subs.Get().Amount, agent);
            }
        }
        if (!changed.Contains(agent))
            changed.Add(agent);
    }
    else
    {
        valid = false;
    }
    return subs;

}

/// <summary>
/// Finds suitable agent
/// </summary>
/// <param name="agents">agents</param>
/// <param name="number">number</param>
/// <returns></returns>
```

```
private static Agent FindSuitableAgent
    (LinkedListT<Agent> agents, int number)
{
    Agent agent = null;
    agents.Start();
    if(agents.Exists())
    {
        for(agents.Start();agents.Exists();
            agents.Next())
        {
            if(agents.Get().AmountOfPublications>number)
            {
                if(agent == null)
                {
                    agent = agents.Get();
                }
                else if(agent.AmountOfPublications>
                    agents.Get().AmountOfPublications)
                {
                    agent = agents.Get();
                }
            }
        }
    }
    return agent;
}

/// <summary>
/// Makes subscription list
/// </summary>
/// <param name="agents">agents</param>
/// <param name="subscriptions">subscriptions</param>
/// <param name="fileName">name of file</param>
/// <param name="month">month</param>
/// <param name="Table">table</param>
/// <returns>average</returns>
public static int MakeSubscriptionList
    (LinkedListT<Agent> agents,
    LinkedListT<Subscription> subscriptions,
    string fileName, int month, Table Table)
{
    int sum = 0;
    int agentscount = 0;
    InOutUtils.OutputHeaderTotxt(fileName,
        "Agentai, kuriem buvo pakeisti sąrašai:");
    for (agents.Start(); agents.Exists();
        agents.Next())
    {
        LinkedListT<Subscription> buff =
            TaskUtils.GetAgentList
            (month, subscriptions, agents.Get());
        InOutUtils.
            OutputAgentsSubscriptionsToTxt
            (fileName, agents.Get(), buff);
        InOutUtils.
            OutputAgentsSubscriptionsToTable
            (Table, agents.Get(), buff);

        agents.Get().AmountOfPublications =
            TaskUtils.GetSum(buff);


        sum += agents.Get().AmountOfPublications;
        agentscount++;
    }
    decimal average = (sum / agentscount);
    average = Math.Floor(average);
```

```
        int avg = Convert.ToInt32(average);

        return avg;
    }

    public static int MakeSubscriptionListSUM
        (LinkedListT<Agent> agents,
        LinkedListT<Subscription> subscriptions,
        string fileName, int month, Table Table)
    {
        int sum = 0;
        int agentscount = 0;

        for (agents.Start(); agents.Exists();
            agents.Next())
        {
            LinkedListT<Subscription> buff =
                TaskUtils.GetAgentList
                (month, subscriptions, agents.Get());


            agents.Get().AmountOfPublications =
                TaskUtils.GetSum(buff);


            sum += agents.Get().AmountOfPublications;
            agentscount++;
        }
        decimal average = (sum / agentscount);
        average = Math.Floor(average);
        int avg = Convert.ToInt32(average);

        return sum;
    }
   /// <summary>
    /// Finds agents above average
    /// </summary>
    /// <param name="agents">agents</param>
    /// <param name="avarage">average</param>
    /// <param name="table">table</param>
    /// <param name="fileName">name of file</param>
    public static void AgentsAboveAvarage
        (LinkedListT<Agent> agents, int avarage,
        Table table, string fileName)
    {
        InOutUtils.OutputHeaderTotxt(fileName,
            "Agentai nešiojantys daugiau nei vidurkis:");
        for(agents.Start();agents.Exists();agents.Next())
        {
            if(agents.Get().AmountOfPublications>avarage)
            {
                InOutUtils.OutputAgentToTxt(fileName, agents.Get());
                InOutUtils.OutputAgentToTable(table, agents.Get());
            }
        }
    }

    public static void RestoreLocation(LinkedListT<Agent> agents, int value)
    {
        agents.Start();
        for (int i = 0; i < value; i++)
        {
            agents.Next();
        }
    }
```

```
        public static void IncreaceAmountOfPublications(LinkedListT<Agent> agents, int value,
Agent agent)
        {
            for(agents.Start(); agents.Exists(); agents.Next())
            {
                if(agent.Equals(agents.Get()))
                {
                    agents.Get().AmountOfPublications += value;
                    break;
                }
            }
        }
    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="LD_9_Prenumerata.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <link href="Style.css" rel="stylesheet" />
    <title></title>
</head>
<body style="height: 505px">
    <form id="form1" runat="server">
        <div style="height: 781px">
            <asp:Label ID="Label8" runat="server" Text="Pasirinkite duomenų failą, kuriame yra
prenumeratos:"></asp:Label>
            <br />
            <asp:FileUpload ID="FileUpload1" runat="server" />
            <br />
            <br />
            <asp:Label ID="Label9" runat="server" Text="Pasirinkite duomenų failą, kuriame yra
agentai:"></asp:Label>
            <br />
            <asp:FileUpload ID="FileUpload2" runat="server" />
            <br />
            <br />
            <asp:Label ID="Label4" runat="server" Text="Mėnesis:"></asp:Label>
            <br />
            <asp:DropDownList ID="DropDownList1" runat="server">
            </asp:DropDownList>
            <br />
            <br />
            <asp:Label ID="Label5" runat="server" Text="Nešiojamos prenumeratos
kiekis:"></asp:Label>
            <br />
            <asp:TextBox ID="TextBox1" runat="server" style="width: 168px"></asp:TextBox>
            <br />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Skaičiuoti" />
            <br />
            <br />
            <asp:Label ID="Label10" runat="server" Text="Pradiniai duomenys:"></asp:Label>
            <br />
            <asp:GridView ID="GridView1" runat="server">
            </asp:GridView>
            <asp:GridView ID="GridView2" runat="server">
            </asp:GridView>
            <br />
            <br />
            <br />
```

```
            <asp:Label ID="Label2" runat="server" Text="Sąrašai agentam:"></asp:Label>
            <br />
            <asp:Table ID="Table1" runat="server" GridLines="Both">
            </asp:Table>
            <br />
            <asp:Label ID="Label11" runat="server" Text="Bendras leidinių kiekis:"></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label7" runat="server" Text="Vidurkis: "></asp:Label>
            <br />
            <asp:Label ID="Label3" runat="server" Text="Agentų, kurie nešioja daugiau nei
vidurkis, sąrašas:"></asp:Label>
            <br />
            <asp:Table ID="Table2" runat="server" GridLines="Both">
            </asp:Table>
            <br />
            <asp:Label ID="Label6" runat="server" Text="Agentai kuriem buvo pakeisti
sąrašai:"></asp:Label>
            <br />
            <asp:Table ID="Table3" runat="server" GridLines="Both">
            </asp:Table>
        </div>
    </form>
</body>
</html>
```

Forma1.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Text.RegularExpressions;
using System.Data;
using LD_9_Prenumerata.App_BarCode;
namespace LD_9_Prenumerata
{
    public partial class Forma1 : System.Web.UI.Page
    {
        const string CDM1 = "U9a.txt";
        const string CDM2 = "U9b.txt";
        const string CRZ = "U9rez.txt";
        protected void Page_Load(object sender, EventArgs e)
        {
            if (DropDownList1.Items.Count == 0)
            {
                for (int i = 1; i <= 12; i++)
                {
                    DropDownList1.Items.Add(i.ToString());
                }

            }
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            var Matches = Regex.Match(TextBox1.Text, @"\D");

            if (!Matches.Success && TextBox1.Text.Length > 0)
            {
                File.Delete(Server.MapPath("App_Data/" + CRZ));

                if (FileUpload1.HasFile && FileUpload1.FileName.EndsWith(".txt") &&
```

```
                FileUpload2.HasFile && FileUpload2.FileName.EndsWith(".txt"))
            {
                #region Read

                App_BarCode.LinkedListT<App_BarCode.Subscription> subscriptions =
                    App_BarCode.InOutUtils.ReadSubscriptions
                    (FileUpload1.PostedFiles[0].InputStream);
                App_BarCode.LinkedListT<App_BarCode.Agent> agents =
                    App_BarCode.InOutUtils.ReadAgents
                    (FileUpload2.PostedFiles[0].InputStream);

                #endregion

                string fileName = Server.MapPath("App_Data/" + CRZ);
                subscriptions.Start();
                agents.Start();
                int month = Convert.ToInt32(DropDownList1.SelectedValue);
                if (subscriptions.Exists())
                {
                    if (agents.Exists())
                    {
                        if (App_BarCode.TaskUtils.CheckSubscriptionsInMonth
                            (month, subscriptions))
                        {
                            #region Making subscription lists
                            OutputAgentToTable(agents);
                            OutputSubscriptionsToTable(subscriptions);
                            App_BarCode.InOutUtils.AgentListToTXT(agents,
Server.MapPath("App_Data/" + CRZ), "Pradiniai duomenys");
                            App_BarCode.InOutUtils.SubscriptionListToTXT(subscriptions,
Server.MapPath("App_Data/" + CRZ), "");
                            int average = App_BarCode.TaskUtils.
                                MakeSubscriptionList(agents, subscriptions,
                                fileName, month, Table1);
                            int suma = App_BarCode.TaskUtils.
                            MakeSubscriptionListSUM(agents, subscriptions,
                            fileName, month, Table1);

                            #endregion

                            #region AgentsAboveAvarage
                            Label7.Text += average.ToString();
                            App_BarCode.TaskUtils.AgentsAboveAvarage
                                (agents, average, Table2, fileName);
                            Label11.Text += suma.ToString();

                            #endregion

                            #region ChangeSubscriptions
                            int number = Convert.ToInt32(TextBox1.Text);
                            App_BarCode.LinkedListT<App_BarCode.Agent> changed =
                                new App_BarCode.LinkedListT<App_BarCode.Agent>();
                            bool valid = true;
                            subscriptions = App_BarCode.TaskUtils.
                                ChangeSubscriptions(subscriptions,
                                agents, number, ref changed, ref valid);
                            if (valid)
                                App_BarCode.TaskUtils.MakeSubscriptionList
                                    (changed, subscriptions, fileName, month,
                                    Table3);
                            else
                            {
                                App_BarCode.InOutUtils.
                                    NoAgentsAboveAmount(Table3);
                                App_BarCode.InOutUtils.
                                    NoAgentsAboveAmount(fileName);
                            }
```

```
                        #endregion
                    }
                    else
                    {
                        App_BarCode.InOutUtils.
                            NoSubscriptionsInMonth(Table1);
                        App_BarCode.InOutUtils.
                            NoSubscriptionsInMonth(fileName);
                    }
                }
                else
                {
                    App_BarCode.InOutUtils.
                        NoAgentsToTxt(fileName);
                    App_BarCode.InOutUtils.
                        NoAgentsToTable(Table1);
                }
            }
            else
            {
                App_BarCode.InOutUtils.
                    NoSubscriptionsToTxt(fileName);
                App_BarCode.InOutUtils.
                    NoSubscriptionsToTable(Table1);
            }
        }
    }
    void OutputAgentToTable(App_BarCode.LinkedListT<App_BarCode.Agent> agents)
    {
        DataTable dataTable = new DataTable();
        DataRow dataRow = null;
        dataTable.Columns.Add("Agento Kodas");
        dataTable.Columns.Add("Pavardė");
        dataTable.Columns.Add("Vardas");
        dataTable.Columns.Add("Adresas");
        dataTable.Columns.Add("Telefono numeris");

        for (agents.Start(); agents.Exists(); agents.Next())
        {
            dataRow = dataTable.NewRow();
            dataRow[0] = agents.Get().AgentCode;
            dataRow[1] = agents.Get().Surname;
            dataRow[2] = agents.Get().Name;
            dataRow[3] = agents.Get().Address;
            dataRow[4] = agents.Get().PhoneNumber;

            dataTable.Rows.Add(dataRow);
        }
        dataTable.AcceptChanges();
        GridView1.DataSource = dataTable;
        GridView1.DataBind();
    }

    void OutputSubscriptionsToTable(App_BarCode.LinkedListT<App_BarCode.Subscription>
subscriptions)
    {
        DataTable dataTable = new DataTable();
        DataRow dataRow = null;
        dataTable.Columns.Add("Adresas");
        dataTable.Columns.Add("Pavardė");
        dataTable.Columns.Add("Telefono numeris");
        dataTable.Columns.Add("Laikotarpio pradžia");
        dataTable.Columns.Add("Laikotarpio ilgis");
        dataTable.Columns.Add("Leidinio kodas");
        dataTable.Columns.Add("Leidinių kiekis");
        dataTable.Columns.Add("Agento kodas");
```

```
                for (subscriptions.Start(); subscriptions.Exists(); subscriptions.Next())
                {
                    dataRow = dataTable.NewRow();
                    dataRow[0] = subscriptions.Get().Address;
                    dataRow[1] = subscriptions.Get().Surname;
                    dataRow[2] = subscriptions.Get().PhoneNumber;
                    dataRow[3] = subscriptions.Get().Start;
                    dataRow[4] = subscriptions.Get().Duration;
                    dataRow[5] = subscriptions.Get().Code;
                    dataRow[6] = subscriptions.Get().Amount;
                    dataRow[7] = subscriptions.Get().AgentCode;

                    dataTable.Rows.Add(dataRow);
                }
                dataTable.AcceptChanges();
                GridView2.DataSource = dataTable;
                GridView2.DataBind();
            }
        }
    }
}
```

LinkedListTTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using LD_9_Prenumerata.App_BarCode;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LD_9_Prenumerata.Tests
{
    public abstract class LinkedListTTests<T> where T : IComparable<T>,
IEquatable<T>
    {
        protected abstract T CreateSampleValue1();
        protected abstract T CreateSampleValue2();
        protected abstract T CreateSampleValue3();
        protected abstract T CreateSampleValue4();

        [TestMethod()]
        public void IsEmpty_EmptyLinkedListT_ReturnsTrue()
        {
            LinkedListT<T> list = new LinkedListT<T>();
            Assert.IsTrue(list.IsEmpty());
        }

        [TestMethod()]
        public void IsEmpty_FilledLinkedListT_ReturnsFalse()
        {
            T value1 = this.CreateSampleValue1();
            LinkedListT<T> list = new LinkedListT<T>();
            list.Add(value1);
            Assert.IsFalse(list.IsEmpty());
        }

        [TestMethod()]
        public void Add_LinkedListT_CorrectOrder()
        {
            T value1 = this.CreateSampleValue1();
            T value2 = this.CreateSampleValue2();
            T value3 = this.CreateSampleValue3();
            T value4 = this.CreateSampleValue4();
            LinkedListT<T> list = new LinkedListT<T>();
            list.Add(value1);
```

```
        list.Add(value2);
        list.Add(value3);
        T[] expected = { value1, value2, value3, value4 };
        int i = 0;
        foreach (T value in list)
        {
            Assert.AreEqual(expected[i], value);
            i++;
        }
    }

    [TestMethod()]
    public void Sort_EmptyLinkedListT_NotCrashes()
    {
        LinkedListT<T> list = new LinkedListT<T>();
        list.Sort();
    }

    [TestMethod()]
    public void Sort_OneElementInLinkedListT_RemainsUnchanged()
    {
        T value1 = this.CreateSampleValue1();
        LinkedListT<T> list = new LinkedListT<T>();
        list.Add(value1);
        list.Sort();
        T[] expected = { value1 };
        int i = 0;
        foreach (T value in list)
        {
            Assert.AreEqual(expected[i], value);
            i++;
        }
    }

    [TestMethod()]
    public void Sort_SortedLinkedListT_RemainsUnchanged()
    {
        T value1 = this.CreateSampleValue1();
        T value2 = this.CreateSampleValue2();
        T value3 = this.CreateSampleValue3();
        T value4 = this.CreateSampleValue4();
        LinkedListT<T> list = new LinkedListT<T>();
        list.Add(value1);
        list.Add(value2);
        list.Add(value3);
        list.Add(value4);
        list.Sort();
        T[] expected = { value1, value2, value3, value4 };
        int i = 0;
        foreach (T value in list)
        {
            Assert.AreEqual(expected[i], value);
            i++;
        }
    }

    [TestMethod()]
    public void Sort_UnsortedLinkedListT_SortedCorrectly()
    {
        T value1 = this.CreateSampleValue1();
        T value2 = this.CreateSampleValue2();
        T value3 = this.CreateSampleValue3();
        T value4 = this.CreateSampleValue4();
        LinkedListT<T> list = new LinkedListT<T>();
        list.Add(value3);
        list.Add(value4);
        list.Add(value2);
```

```
            list.Add(value1);
            list.Sort();
            T[] expected = { value1, value2, value3, value4 };
            int i = 0;
            foreach (T value in list)
            {
                Assert.AreEqual(expected[i], value);
                i++;
            }
        }
    }

    [TestClass()]
    public class LinkedListTRefTests : LinkedListTTests<Agent>
    {
        protected override Agent CreateSampleValue1() => new Agent(789,
            "Bukis", "Petras", "Vilnius, Ruklos g. 2", "867854623");
        protected override Agent CreateSampleValue2() => new Agent(654,
            "Mukis", "Lukas", "Visaginas, Debesies g. 4", "867854624");
        protected override Agent CreateSampleValue3() => new Agent(123,
            "Rukis", "Jonas", "Kaunas, blablala 42", "867854622");
        protected override Agent CreateSampleValue4() => new Agent(456,
            "Tobutis", "Tomas", "Vilnius, Tiesos g. 52A", "867854621");
    }

    [TestClass()]
    public class LinkedListTValTests : LinkedListTTests<int>
    {
        protected override int CreateSampleValue1() => -96;
        protected override int CreateSampleValue2() => 0;
        protected override int CreateSampleValue3() => 42;
        protected override int CreateSampleValue4() => 1693;
    }
}

Style.css
body {
    background-color:burlywood;
}
table {
    border-color:darkred;
    font-family:Arial,'Agency FB';
}
input{
    background-color:aquamarine;
    border:solid 1px black;
}
```

3.7. Pradiniai duomenys ir rezultatai

1 Variantas

Pradiniai duomenys:
U9a.txt

```
Vilnius, Taikos pr. 14-22;Jonaitis;864725962;5;11;24624;4;456
Kaunas, Rusu g. 1-41;Tomutis;864725962;3;6;31421;2;123
Šilutė, Laisvės al. 4-15;Stasytis;865214867;3;9;71425;5;123
Marijampolė, Savanorių g. 7-1;Vingytė;861542789;5;2;54712;1;123
Kaunas, Taikos g. 4-156;Jonavičius;862458731;5;4;25142;6;456
Klaipėda, Ratų g. 2-13;Antanovič;864157842;3;3;24167;3;897
Pajūralis, Gėlių g. 6-1;Višinskas;869574125;5;2;85412;4;789
Visaginas, Saulės pr. 2-2;Paršaitis;861247512;5;7;62145;9;456
Vilnius, Dainavos pr. 3-4;Mikužė;862459753;3;6;74551;2;654
Jonava, Medžių g. 6-6;Vilkė;865221455;5;8;62214;1;231
Zarasai, Marių g. 7-2;Pilkys;864471524;5;6;12458;3;897
Rokiškis, Vinčų pr. 3-1;Šveistakis;866957452;5;7;33512;4;897
Palanga, Basanavičiaus g. 9-1;Lingytė;862254117;5;5;14214;5;123
Juknaičiai, Pievų g. 5-5;Vingis;863536822;3;3;44215;7;654
Mažeikiai, Bendrovės pr. 16-4;Mažeikis;866744514;5;1;33235;7;456
Šiauliai, Geležinės Lapės pr. 2-45;Kalašnikovė;865952471;5;3;22325;4;231
Rokiškis, Pieno g. 2-31;Spindulys;862241458;5;1;77452;5;897
Vilnius, Šviesos pr. 7-12;Raudienė;863323511;3;7;44751;8;231
Klaipėda, Jūros pr. 5-21;Maksimavičienė;865565124;3;6;11758;2;123
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;5;1;99425;3;456
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;3;1;99425;3;789
```

U9b.txt

```
456;Tobutis;Tomas;Vilnius, Tiesos g. 52A;867854621
123;Rukis;Jonas;Kaunas, blablala 42;867854622
789;Bukis;Petras;Vilnius, Ruklos g. 2;867854623
654;Mukis;Lukas;Visaginas, Debesies g. 4;867854624
231;Tukis;Romas;Klaipėda, blablabla g. 2;867854625
897;Vukis;Ignas;Kaunas, bagsdgaghd g. 9;867854626
```

Rezultatai:

U9rez.txt

U9rez - Notepad

File Edit Format View Help

Pradiniai duomenys

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621
123	Rukis	Jonas	Kaunas, blablala 42	867854622
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Vilnius, Taikos pr. 14-22	Jonaitis	864725962	5	11	24624	4	456
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123
Marijampolė, Savanorių g. 7-1	Vingytė	861542789	5	2	54712	1	123
Kaunas, Taikos g. 4-156	Jonavičius	862458731	5	4	25142	6	456
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	897
Pajūralis, Gėlių g. 6-1	Višinskas	869574125	5	2	85412	4	789
Visaginas, Saulės pr. 2-2	Paršaitis	861247512	5	7	62145	9	456
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2	654
Jonava, Medžių g. 6-6	Vilkė	865221455	5	8	62214	1	231
Zarasai, Marių g. 7-2	Pilkys	864471524	5	6	12458	3	897
Rokiškis, Vinčų pr. 3-1	Šveistakis	866957452	5	7	33512	4	897
Palanga, Basanavičiaus g. 9-1	Lingytė	862254117	5	5	14214	5	123
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7	654
Mažeikiai, Bendrovės pr. 16-4	Mažeikis	866744514	5	1	33235	7	456
Šiauliai, Geležinės Lapės pr. 2-45	Kalašnikovė	865952471	5	3	22325	4	231
Rokiškis, Pieno g. 2-31	Spindulys	862241458	5	1	77452	5	897
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	5	1	99425	3	456
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Agentai, kuriem buvo pakeisti sąrašai:

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621

Prenumeratų nėra

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

U9rez - Notepad

File Edit Format View Help

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
456	Tobutis	Tomas	Vilnius, Tiesos g. 52A	867854621

Prenumeratų nėra

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
789	Bukis	Petras	Vilnius, Ruklos g. 2	867854623

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	789

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Juknaičiai, Pievų g. 5-5	Vingis	863536822	3	3	44215	7	654
Vilnius, Dainavos pr. 3-4	Mikužė	862459753	3	6	74551	2	654

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
897	Vukis	Ignas	Kaunas, bagsdgaghd g. 9	867854626

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	897

Agentai nešiojantys daugiau nei vidurkis:

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
654	Mukis	Lukas	Visaginas, Debesies g. 4	867854624

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625

Agentai, kuriem buvo pakeisti sąrašai:

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
231	Tukis	Romas	Klaipėda, blablabla g. 2	867854625

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Vytauto pr. 3-11	Žalytė	862124371	3	1	99425	3	231
Vilnius, Šviesos pr. 7-12	Raudienė	863323511	3	7	44751	8	231

Agento kodas	Pavardė	Vardas	Adresas	Telefono numeris
123	Rukis	Jonas	Kaunas, blablala 42	867854622

Prenumeratos:

Adresas	Pavardė	Telefono numeris	Laikotarpio pradžia	Laikotarpio ilgis	Leidinio kodas	Leidinių kiekis	Agento kodas
Kaunas, Rusu g. 1-41	Tomutis	864725962	3	6	31421	2	123
Klaipėda, Jūros pr. 5-21	Maksimavičienė	865565124	3	6	11758	2	123
Klaipėda, Ratų g. 2-13	Antanovič	864157842	3	3	24167	3	123
Šilutė, Laisvės al. 4-15	Stasytis	865214867	3	9	71425	5	123

2 Variantas – duomenų failai tušti

Pradiniai duomenys:

U9a_2.txt

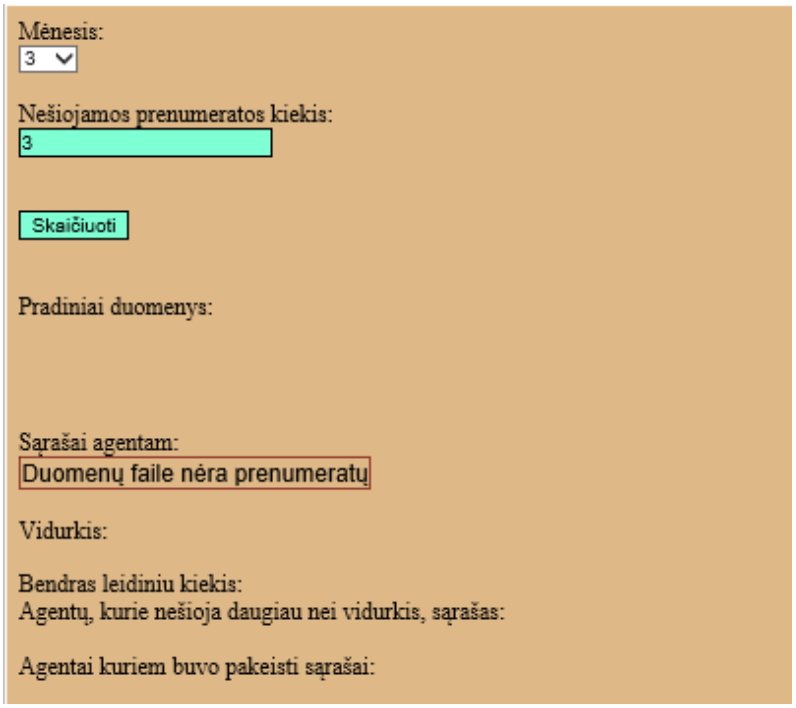
U9b_2.txt

Rezultatai:

U9rez.txt

```
Duomenų faile nėra prenumeratų
```

Vartotojo sąsaja:



Variantas 3 – pasirenkamas menuo kai yra nenešiojama prenumerata

Duomenys:

U9a_3.txt

U9a_3 - Notepad

File Edit Format View Help

```
Vilnius, Taikos pr. 14-22;Jonaitis;864725962;5;11;24624;4;456
Kaunas, Rusu g. 1-41;Tomutis;864725962;3;6;31421;2;123
Šilutė, Laisvės al. 4-15;Stasytis;865214867;3;9;71425;5;123
Marijampolė, Savanorių g. 7-1;Vingytė;861542789;5;2;54712;1;123
Kaunas, Taikos g. 4-156;Jonavičius;862458731;5;4;25142;6;456
Klaipėda, Ratų g. 2-13;Antanovič;864157842;3;3;24167;3;897
Pajūralis, Gėlių g. 6-1;Višinskas;869574125;5;2;85412;4;789
Visaginas, Saulės pr. 2-2;Paršaitis;861247512;5;7;62145;9;456
Vilnius, Dainavos pr. 3-4;Mikužė;862459753;3;6;74551;2;654
Jonava, Medžių g. 6-6;Vilkė;865221455;5;8;62214;1;231
Zarasai, Marių g. 7-2;Pilkys;864471524;5;6;12458;3;897
Rokiškis, Vinčų pr. 3-1;Šveistakis;866957452;5;7;33512;4;897
Palanga, Basanavičiaus g. 9-1;Lingytė;862254117;5;5;14214;5;123
Juknaičiai, Pievų g. 5-5;Vingis;863536822;3;3;44215;7;654
Mažeikiai, Bendrovės pr. 16-4;Mažeikis;866744514;5;1;33235;7;456
Šiauliai, Geležinės Lapės pr. 2-45;Kalašnikovė;865952471;5;3;22325;4;231
Rokiškis, Pieno g. 2-31;Spindulys;862241458;5;1;77452;5;897
Vilnius, Šviesos pr. 7-12;Raudienė;863323511;3;7;44751;8;231
Klaipėda, Jūros pr. 5-21;Maksimavičienė;865565124;3;6;11758;2;123
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;5;1;99425;3;456
Kaunas, Vytauto pr. 3-11;Žalytė;862124371;3;1;99425;3;789
```

U9b_3.txt

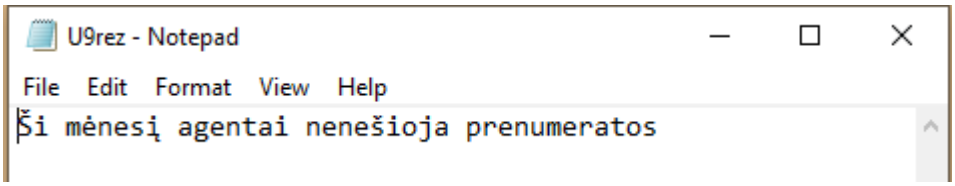
U9b_3 - Notepad

File Edit Format View Help

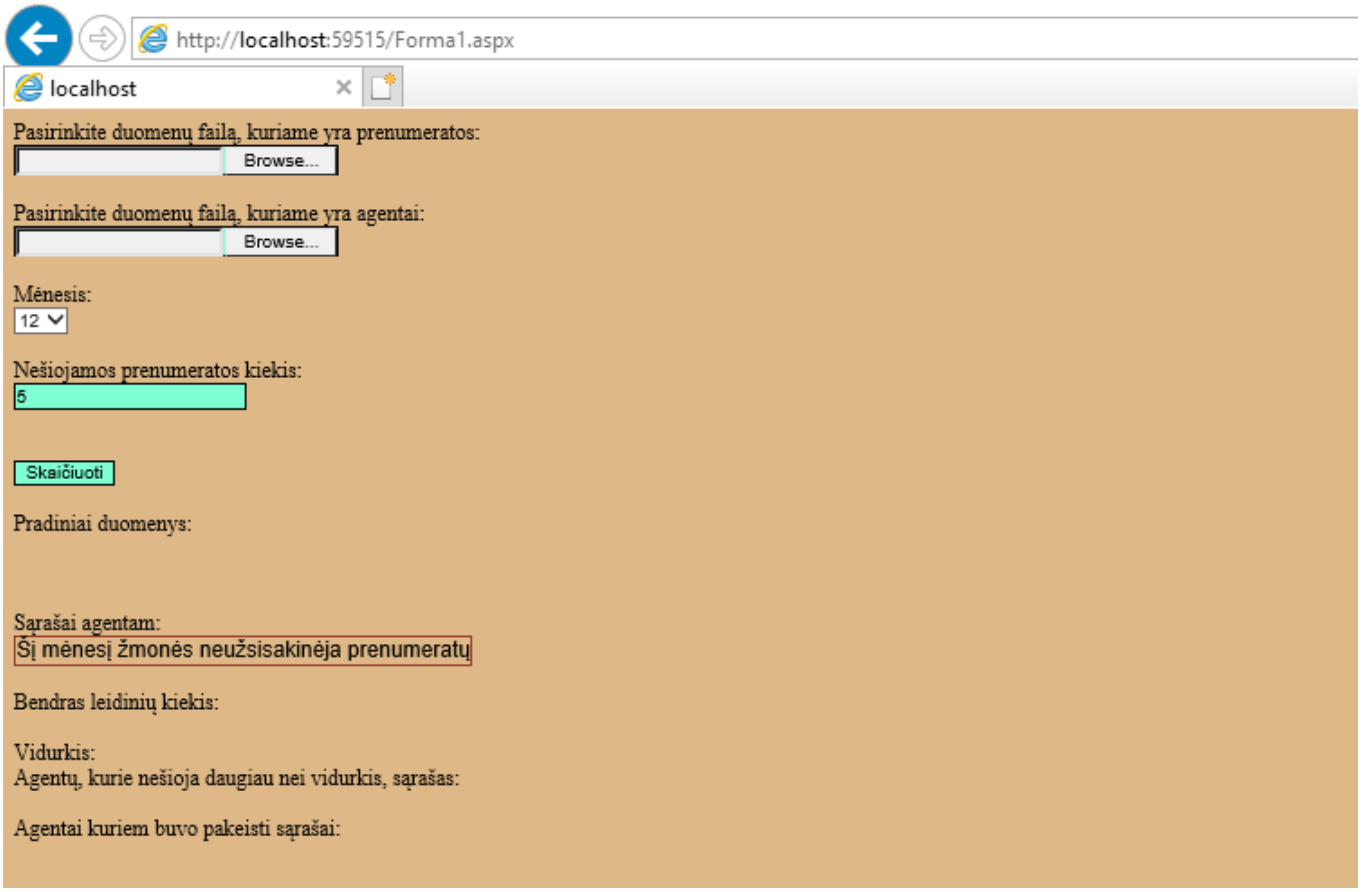
```
456;Tobutis;Tomas;Vilnius, Tiesos g. 52A;867854621
123;Rukis;Jonas;Kaunas, blablala 42;867854622
789;Bukis;Petras;Vilnius, Ruklos g. 2;867854623
654;Mukis;Lukas;Visaginas, Debesies g. 4;867854624
231;Tukis;Romas;Klaipėda, blablabla g. 2;867854625
897;Vukis;Ignas;Kaunas, bagsdgaghd g. 9;867854626
```

Rezultatai:

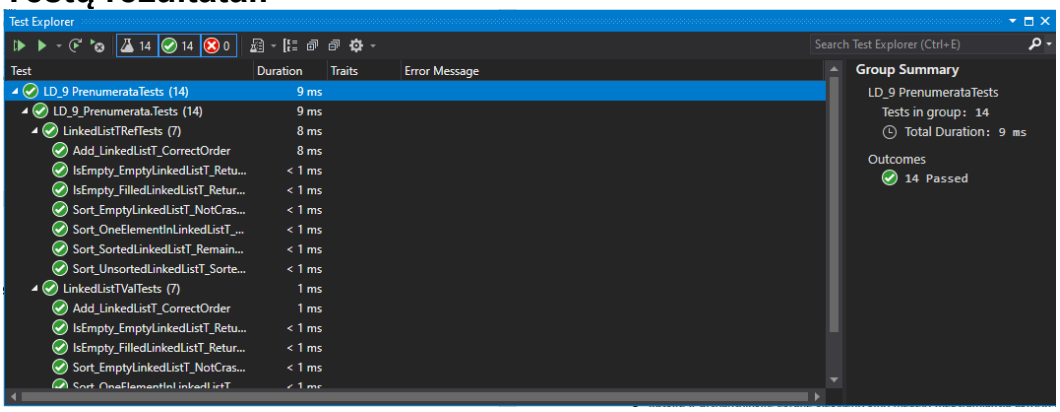
U9rez.txt



Vartotojo sąsaja:



Testų rezultatai:



3.8. Dėstytojo pastabos

4. Polimorfizmas ir išimčių valdymas (L4)

4.1. Darbo užduotis

U4_9. IMBD. Turite keleto kinomanų mėgėjų peržiūrėtus filmų ir serialų sąrašus. Pirmoje eilutėje yra kino mėgėjo vardas pavardė, antroje – gimimo metai, trečioje – miestas. Sudarykite abstrakčiąją klasę „Record“ (savybės - pavadinimas, žanras, kino studija, du pagrindiniai aktoriai), kurią paveldės klasės „Film“ (savybės – leidimo metai, režisierius, pajamos) ir „TVseries“ (savybės – pradžios metai, serijų kiekis, pabaigos metai (jei yra), požymis „ar tęsiasi“).

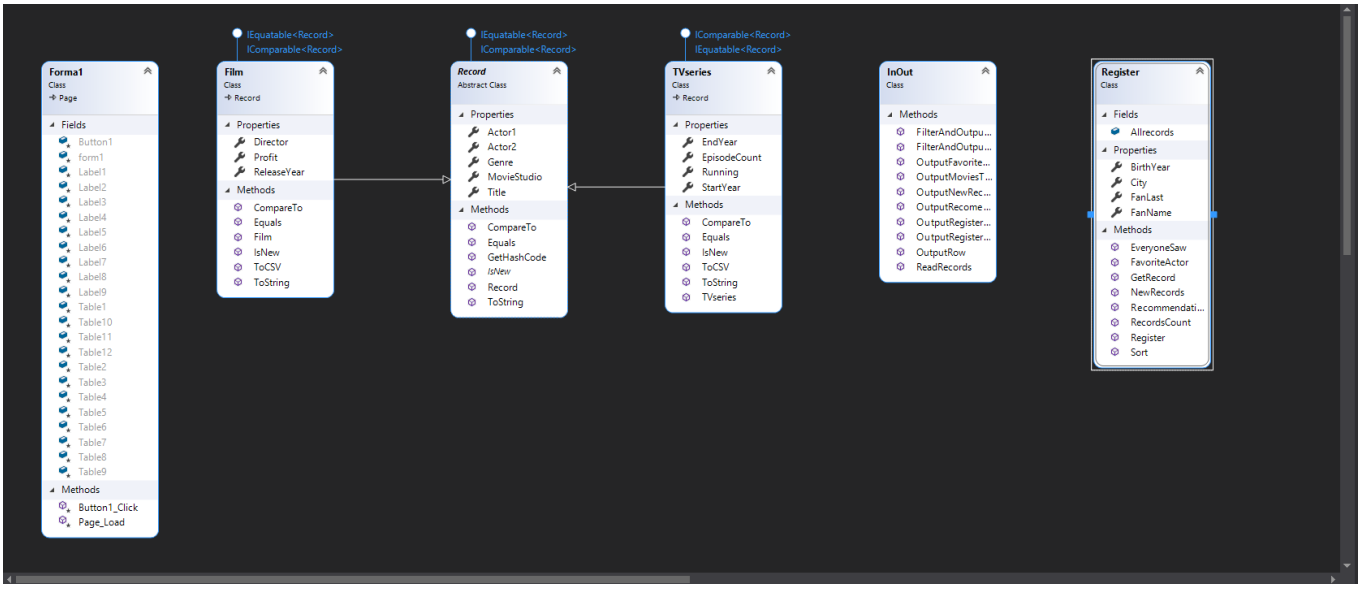
- Raskite ir atspausdinkite ekrane kiekvieno kino mėgėjo mėgstamiausią aktorių (tai aktorius, kuris atliko daugiausiai vaidmenų peržiūrėtuose filmuose ir serialuose).
- Sudarykite filmų ir serialų, kuriuos peržiūrėjo visi kino mėgėjai, sąrašą. Visus duomenis apie juos įrašykite į failą „MatėVisi.csv“.
- Kiekvienam kino mėgėjui sudarykite rekomenduojamų peržiūrėti filmų ir serialų sąrašą, į kurį įtraukite filmus ir serialus, kurių jis nematė, tačiau matė kiti kino mėgėjai. Rekomendacijų sąrašus įrašykite į failus „Rekomendacija_vardas_pavardė.csv“.
- Sudarykite ir surikiuokite naujų filmų ir serialų sąrašą, pateikdami pilną informaciją apie juos. Filmas yra naujas, jei nuo leidimo metų prabėgo mažiau, nei 2 metai. Serialas yra naujas, jei nuo pradžios metų prabėgo mažiau nei metai. Filmus rikiuokite pagal pajamas, serialus – pagal pradžios ir pabaigos metus. Rezultatus įrašykite į failą „Nauji.csv“.

4.2. Grafinės vartotojo sąsajos schema

4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button1	Text	Skaičiuoti
Table1	GridLines	Both
Label1	Text	Originalūs duomenys:
Label2	Text	Mėgiamiausi aktoriai:
Label4	Text	Nematyti filmai ir serialai
Table2	GridLines	Both
Table3	GridLines	Both
Label5	Text	Rekomendacijos
Label7	Text	Pirmo fano rekomendacijos
Label8	Text	Antro fano rekomendacijos
Label9	Text	Trečio fano rekomendacijos
Label6	Text	Nauji filmai ir serialai

4.4. Klasių diagrama



4.5. Programos vartotojo vadovas

Duomenų failus reikia įkelti į App_Data katalogą, duomenų failų pavadinimai nėra svarbūs. Filmų formatas duomenų faile: pavadinimas;žanras;kino studija;pirmas pagrindinis aktorius;antras pagrindinis aktorius;leidimo metai;režisierius;pajamos. Serialo duomenų formatas: pavadinimas;žanras;kino

studija;pirmas pagrindinis aktorius;antras pagrindinis aktorius;pradžios metai;serijų kiekis;pabaigos metai (jei yra);požymis. Vartotojo sąsajoje paspaudus mygtuką skaičiuoti programa pradeda veikti ir išveda į vartotojo sąsają pradinius duomenis bei mėgiamiausius aktorius. Likusieji rezultatai bus sukurtuose .csv failuose projekto aplankale.

4.6. Programos tekstas

Film.cs

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

namespace L4.App_BarCode
{
    /// <summary>
    /// Derrivative class Film
    /// </summary>
    class Film : Record, IEquatable<Record>, IComparable<Record>
    {
        public int ReleaseYear { get; set; }
        public string Director { get; set; }
        public decimal Profit { get; set; }

        /// <summary>
        /// Constructor for Film object
        /// </summary>
        /// <param name="title"></param>
        /// <param name="genre"></param>
        /// <param name="movieStudio"></param>
        /// <param name="actor1"></param>
        /// <param name="actor2"></param>
        /// <param name="releaseYear"></param>
        /// <param name="director"></param>
        /// <param name="profit"></param>
        public Film(string title, string genre, string movieStudio,
            string actor1, string actor2,
            int releaseYear, string director, decimal profit)
            : base(title, genre, movieStudio, actor1, actor2)
        {
            this.ReleaseYear = releaseYear;
            this.Director = director;
            this.Profit = profit;
        }

        /// <summary>
        /// Formats string for outputing to CSV file
        /// </summary>
        /// <returns></returns>
        public string ToCSV()
        {
            return string.Format("{0}; {1}; {2}; {3};" +
                                 " {4}; {5}; {6}; {7};", this.Title,
                                 this.Genre, this.MovieStudio,
                                 this.Actor1, this.Actor2,
                                 this.ReleaseYear, this.Director,
                                 this.Profit);
        }
```

```
        /// <summary>
        /// Formats string to a table format
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            return string.Format("| {0, -25}| {1, -14}| " +
                                 "{2, -20}| {3, -30}|" +
                                 " {4, -30}| {5, -8}|" +
                                 " {6, -25}| {7, -10}| {8, 10} |",
                                  this.Title, this.Genre, this.MovieStudio,
                                  this.Actor1,this.Actor2, this.ReleaseYear,
                                  this.Director,this.Profit, "");
        }

        /// <summary>
        /// Compes two records
        /// </summary>
        /// <param name="other">other record</param>
        /// <returns>result</returns>
        public int CompareTo(Film other)
        {


            Film temp = (other as Film);
            return temp.Profit.CompareTo(this.Profit);


        }

        public override bool Equals(Record other)
        {
            if (other is Film)
            {
                if (this.Title == other.Title &&
                    this.MovieStudio == other.MovieStudio)
                    return true;
                else
                    return false;
            }
            else
                return false;
        }

        public override bool IsNew()
        {
            return DateTime.Now.Year - ReleaseYear < 2;
        }
    }
}
```

InOut.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Globalization;
using System.Text;
using System.Web.DynamicData;

namespace L4.App_BarCode
{
    class InOut
```

```
{
    /// <summary>
    /// Reads data from specified file
    /// </summary>
    /// <param name="fileName"> file </param>
    /// <returns> The data to a register format </returns>
    public static Register ReadRecords(string fileName)
    {
        try
        {
            List<Record> records = new List<Record>();
            string[] lines = File.ReadAllLines(fileName, Encoding.UTF8);
            string[] namevalues = lines[0].Split(' ');
            string FanName = namevalues[0];
            string FanLastName = namevalues[1];
            int year = int.Parse(lines[1]);
            string city = lines[2];
            for (int i = 3; i < lines.Length; i++)
            {
                string[] values = lines[i].Split(';');
                string type = values[0];
                string title = values[1];
                string genre = values[2];
                string movieStudio = values[3];
                string actor1 = values[4];
                string actor2 = values[5];


                switch (type)
                {
                    case "FILM":
                        int releaseYear = int.Parse(values[6]);
                        string director = values[7];
                        decimal profit = 0;
                        decimal.TryParse(values[8], out profit);
                        Film film = new Film(title, genre, movieStudio,
                            actor1, actor2, releaseYear, director, profit);
                        records.Add(film);
                        break;

                    case "TVSERIES":
                        int startYear = int.Parse(values[6]);
                        int episodeCount = int.Parse(values[7]);
                        string endyear = "";
                        bool running = false;
                        if (values.Length < 10)
                        {
                            endyear = "-";
                            if (values[8] == "Taip") running = true;
                            if (values[8] == "Ne") running = false;
                        }
                        else
                        {
                            endyear = values[8];
                            if (values[9] == "Taip") running = true;
                            if (values[9] == "Ne") running = false;
                        }
                        TVseries serial = new TVseries(title, genre,
                            movieStudio, actor1, actor2,
                            startYear, episodeCount, endyear, running);
                        records.Add(serial);
                        break;

                    default:
                        break;
                }
```

```
                }

                Register register = new Register(records,
                    FanName, FanLastName, year, city);
                return register;
            }
            catch
            {
                throw new Exception("Duomenys netinkamo formato/tušti/netinkamoi vietoj");
            }
        }

        /// <summary>
        /// Outputs favorite actor of movie fan
        /// </summary>
        /// <param name="actor"> Favorite actor </param>
        /// <param name="register"> register with the data about the fan </param>
        public static void OutputFavoriteActor(Table Table,
            string actor, Register register)
        {
            OutputRow(Table, new object[] { $"{register.FanName} {register.FanLast}", actor });
        }

        public static void FilterAndOutputFilms(Table Table, List<Record> records)
        {
            List<Film> films = records.OfType<Film>().ToList();

            if (films.Count > 0)
            {
                OutputRow(Table, new object[] { "Filmai" });
                OutputRow(Table, new object[] { "Pavadinimas", "Žanras", "Kino studija", "Pagr.
aktorius 1", "Pagr. aktorius 2", "Leidimo metai", "Režisierius", "Pajamos" });
                foreach (Film film in films)
                {
                            OutputRow(Table, new object[] { film.Title, film.Genre,
       film.MovieStudio, film.Actor1, film.Actor2, film.ReleaseYear, film.Director, film.Profit
       });
                }
            }
            else
            {
                OutputRow(Table, new object[] { "Nėra filmų!" });
            }
        }

        public static void FilterAndOutputTVseries(Table Table, List<Record> records)
        {
            List<TVseries> tvseries = records.OfType<TVseries>().ToList();

            if (tvseries.Count > 0)
            {
                OutputRow(Table, new object[] { "Serialai" });
                OutputRow(Table, new object[] { "Pavadinimas", "Žanras", "Kino studija", "Pagr.
aktorius 1", "Pagr. aktorius 2", "Pradžios metai", "Serijų kiekis", "Pabaigos metai", "Ar
tęsiasi" });
                foreach (TVseries series in tvseries)
                {
                    OutputRow(Table, new object[] { series.Title, series.Genre,
series.MovieStudio,
                    series.Actor1, series.Actor2, series.StartYear, series.EpisodeCount,
series.EndYear, series.Running });
                }
            }
            else
            {
                OutputRow(Table, new object[] { "Nėra serialų!" });
            }
```

```
    }

    public static void OutputRow(Table Table, object[] cols)
    {
        TableRow row = new TableRow();

        foreach (object col in cols)
        {
            TableCell cell = new TableCell();
            cell.Text = col.ToString();
            row.Cells.Add(cell);
        }

        Table.Rows.Add(row);
    }

    /// <summary>
    /// Outputs all the movies that everyone saw
    /// </summary>
    /// <param name="fileName"></param>
    /// <param name="records"></param>
   public static void OutputMoviesThatEveryoneSaw(string fileName,
        List<Record> records)
    {
        if (records.Count == 0)
        {
            File.WriteAllText(fileName,"Filmų, kuriuos matė visi nėra!");
        }
        else
        {
            List<string> data = new List<string>();
            for (int i = 0; i < records.Count; i++)
            {
                Record record = records[i];
                if (record is Film)
                {

                    Film film = (record as Film);
                    data.Add(film.ToCSV());
                }
                if (record is TVseries)
                {
                    TVseries serial = (record as TVseries);
                    data.Add(serial.ToCSV());
                }

            }
            File.WriteAllLines(fileName, data, Encoding.UTF8);
        }

    }

    /// <summary>
    /// Outputs recommendations to a movie fan
    /// </summary>
    /// <param name="fileName"> file to write in </param>
    /// <param name="recomendations"> all the recommendations </param>
    public static void OutputRecomendations(string fileName,
        List<Record> recomendations)
    {
        List<string> data = new List<string>();

        if (recomendations.Count == 0)
        {
            data.Add("Rekomendacijų nėra");
        }
        else
```

```
        {
            for (int i = 0; i < recomendations.Count; i++)
            {
                try
                {
                    Record recomendation = recomendations[i];
                    if (recomendation is Film)
                    {

                        Film film = (recomendation as Film);
                        data.Add(film.ToCSV());
                    }
                    if (recomendation is TVseries)
                    {
                        TVseries serial = (recomendation as TVseries);
                        data.Add(serial.ToCSV());
                    }
                    else if(i >= recomendations.Count - 1)
                    {
                        data.Add("Nera serialu");
                    }
                }
                catch
                {
                    throw new Exception("Rekomendaciju nera");
                }

            }

        }
        File.WriteAllLines(fileName, data, Encoding.UTF8);
    }

    /// <summary>
    /// Outputs original data
    /// </summary>
    /// <param name="fileName"> file to write in</param>
    /// <param name="register"> data </param>
    public static void OutputRegisterToTxt(string fileName,
        Register register)
    {
        const string FormatFanData = "| {0, -37}|";

        List<string> data = new List<string>();
        data.Add(new string('-', 40));
        data.Add(string.Format(FormatFanData,
                 "Kinomano mėgėjo vardas ir pavardė:"));
        data.Add(string.Format(FormatFanData, register.FanName +
            " " + register.FanLast));
        data.Add(new string('-', 40));
        data.Add(string.Format(FormatFanData,
                "Kinomano mėgėjo gimimo metai: "));
        data.Add(string.Format(FormatFanData, register.BirthYear.ToString()));
        data.Add(new string('-', 40));
        data.Add(string.Format(FormatFanData, "Kinomano mėgėjo miestas:"));
        data.Add(string.Format(FormatFanData, register.City));
        data.Add(new string('-', 192));
        for (int i = 0; i < register.Allrecords.Count; i++)
        {
            Record record = register.Allrecords[i];
            if (record is Film)
            {
                Film film = (record as Film);
                data.Add(film.ToString());
            }
            if (record is TVseries)
            {
```

```
            TVseries serial = (record as TVseries);
            data.Add(serial.ToString());
        }
    }
    data.Add(new string('-', 192));
    data.Add(new string(' ', 192));


    File.AppendAllLines(fileName, data, Encoding.UTF8);
}

/// <summary>
/// Outputs register to table
/// </summary>
/// <param name="Table">table to output</param>
/// <param name="register">register</param>
public static void OutputRegisterToTable(Table Table,
    Register register)
{
    TableCell fanname = new TableCell();
    TableCell fanlast = new TableCell();
    fanname.Text = register.FanName;
    fanlast.Text = register.FanLast;
    TableRow namesurname = new TableRow();
    namesurname.Cells.Add(fanname);
    namesurname.Cells.Add(fanlast);
    Table.Rows.Add(namesurname);

    TableCell birthdate = new TableCell();
    birthdate.Text = register.BirthYear.ToString();
    TableRow date = new TableRow();
    date.Cells.Add(birthdate);
    Table.Rows.Add(date);

    TableCell city = new TableCell();
    city.Text = register.City;
    TableRow City = new TableRow();
    City.Cells.Add(city);
    Table.Rows.Add(City);

    for(int i=0; i<register.RecordsCount();i++)
    {
        if(register.GetRecord(i) is Film)
        {
            TableCell cell1 = new TableCell();
            TableCell cell2 = new TableCell();
            TableCell cell3 = new TableCell();
            TableCell cell4 = new TableCell();
            TableCell cell5 = new TableCell();
            TableCell cell6 = new TableCell();
            TableCell cell7 = new TableCell();
            TableCell cell8 = new TableCell();
            Film temp = (register.GetRecord(i) as Film);
            cell1.Text = temp.Title;
            cell2.Text = temp.Genre;
            cell3.Text = temp.MovieStudio;
            cell4.Text = temp.Actor1;
            cell5.Text = temp.Actor2;
            cell6.Text = temp.ReleaseYear.ToString();
            cell7.Text = temp.Director;
            cell8.Text = temp.Profit.ToString();
            TableRow row = new TableRow();
            row.Cells.Add(cell1);
            row.Cells.Add(cell2);
            row.Cells.Add(cell3);
            row.Cells.Add(cell4);
            row.Cells.Add(cell5);
```

```
            row.Cells.Add(cell6);
            row.Cells.Add(cell7);
            row.Cells.Add(cell8);
            Table.Rows.Add(row);
        }
        if(register.GetRecord(i) is TVseries)
        {
            TableCell cell1 = new TableCell();
            TableCell cell2 = new TableCell();
            TableCell cell3 = new TableCell();
            TableCell cell4 = new TableCell();
            TableCell cell5 = new TableCell();
            TableCell cell6 = new TableCell();
            TableCell cell7 = new TableCell();
            TableCell cell8 = new TableCell();
            TableCell cell9 = new TableCell();
            TVseries temp = (register.GetRecord(i) as TVseries);
            cell1.Text = temp.Title;
            cell2.Text = temp.Genre;
            cell3.Text = temp.MovieStudio;
            cell4.Text = temp.Actor1;
            cell5.Text = temp.Actor2;
            cell6.Text = temp.StartYear.ToString();
            cell7.Text = temp.EpisodeCount.ToString();
            cell8.Text = temp.EndYear.ToString();
            if (temp.Running == true)
                cell9.Text = "Tęsiasi";
            else
                cell9.Text = "Nesitesi";
            TableRow row = new TableRow();
            row.Cells.Add(cell1);
            row.Cells.Add(cell2);
            row.Cells.Add(cell3);
            row.Cells.Add(cell4);
            row.Cells.Add(cell5);
            row.Cells.Add(cell6);
            row.Cells.Add(cell7);
            row.Cells.Add(cell8);
            row.Cells.Add(cell9);
            Table.Rows.Add(row);
        }
    }
}

/// <summary>
/// Outputs new records to csv file
/// </summary>
/// <param name="records">new records</param>
/// <param name="fileName">file to output</param>
public static void OutputNewRecords(List<Record> records, string fileName)
{
    try
    {
        List<string> data = new List<string>();
        if (records.Count > 0)
        {
            for (int i = 0; i < records.Count; i++)
            {
                if (records[i] is Film)
                {
                    data.Add((records[i] as Film).ToCSV());
                }
                if (records[i] is TVseries)
                {
                    data.Add((records[i] as TVseries).ToCSV());
                }
```

```
                }
                File.WriteAllLines(fileName, data, Encoding.UTF8);
            }
            else
                File.WriteAllText(fileName, "Nera tokiu filmu/serialu");
        }
        catch
        {
            throw new Exception("Nera nauju filmu ar serialu");
        }
    }
}
```

Record.cs

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

namespace L4.App_BarCode
{
    /// <summary>
    /// Base class
    /// </summary>
    abstract class Record : IEquatable<Record>, IComparable<Record>
    {
        /// <summary>
        /// Constructor for Record
        /// </summary>
        /// <param name="title"></param>
        /// <param name="genre"></param>
        /// <param name="movieStudio"></param>
        /// <param name="actor1"></param>
        /// <param name="actor2"></param>
        public Record(string title, string genre,
            string movieStudio, string actor1, string actor2)
        {
            Title = title;
            Genre = genre;
            MovieStudio = movieStudio;
            this.Actor1 = actor1;
            this.Actor2 = actor2;
        }

        public string Title { get; set; }
        public string Genre { get; set; }
        public string MovieStudio { get; set; }
        public string Actor1 { get; set; }
        public string Actor2 { get; set; }

        /// <summary>
        /// Compares records
        /// </summary>
        /// <param name="other"></param>
        /// <returns></returns>
        public virtual int CompareTo(Record other)
        {
            if (this.Title == other.Title)
            {
                return MovieStudio.CompareTo(other.MovieStudio);
            }
            else
                return this.Title.CompareTo(other.Title);
        }
```

```
        /// <summary>
        /// Checks if records are equal
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public virtual bool Equals(Record other)
        {
            return other is Record record &&
                   Title == record.Title &&
                   Genre == record.Genre &&
                   MovieStudio == record.MovieStudio &&
                   Actor1 == record.Actor1 &&
                   Actor2 == record.Actor2;
        }

        /// <summary>
        /// Returns HashCode
        /// </summary>
        /// <returns>HashCode</returns>
        public override int GetHashCode()
        {
            return base.GetHashCode();
        }

        /// <summary>
        /// Formats string
        /// </summary>
        /// <returns> Formated string </returns>
        public override string ToString()
        {
            return string.Format("{0} {1} {2} {3} {4}",
                this.Title, this.Genre, this.MovieStudio,
                this.Actor1, this.Actor2);
        }

        public abstract bool IsNew();
    }
}
```

Register.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Text.RegularExpressions;
namespace L4.App_BarCode
{
    /// <summary>
    /// Register class for preforming tasks
    /// </summary>
    class Register
    {
        public List<Record> Allrecords;

        public string FanName { get; set; }

        public string FanLast { get; set; }
        public int BirthYear { get; set; }
        public string City { get; set; }

        /// <summary>
        /// Register constructor
        /// </summary>
        /// <param name="allrecords"></param>
        /// <param name="fanName"></param>
```

```
/// <param name="fanlastName"></param>
/// <param name="birthYear"></param>
/// <param name="city"></param>
public Register(List<Record> allrecords,
 string fanName, string fanlastName, int birthYear, string city)
{
    Allrecords = allrecords;
    FanName = fanName;
    FanLast = fanlastName;
    BirthYear = birthYear;
    City = city;
}

/// <summary>
/// Count of records
/// </summary>
/// <returns> The number of records </returns>
public int RecordsCount()
{
    return this.Allrecords.Count;
}

/// <summary>
/// Gets record by index
/// </summary>
/// <param name="index"></param>
/// <returns> The needed record </returns>
public Record GetRecord(int index)
{
    return this.Allrecords[index];
}

#region Task1

/// <summary>
/// Finds favorite actor of movie fan
/// </summary>
/// <returns> the actor </returns>
public string FavoriteActor()
{

    int count1 = 0;
    int count2 = 0;
    int maxcount = 0;
    string actor1 = "";
    string actor2 = "";
    string maxactor = "";
    for (int i = 0; i < Allrecords.Count; i++)
    {
        for (int j = 0; j < Allrecords.Count; j++)
        {
            var matches1 = Regex.Matches(Allrecords[j].ToString(),
                Allrecords[i].Actor1.ToString());


            count1 += matches1.Count;
            actor1 = Allrecords[i].Actor1.ToString();


            var matches2 = Regex.Matches(Allrecords[j].ToString(),
                Allrecords[i].Actor2.ToString());

            count2 += matches2.Count;
            actor2 = Allrecords[i].Actor2.ToString();


        }
```

```
        if (maxcount < Math.Max(count1, count2))
        {
            maxcount = Math.Max(count1, count2);
            if (count1 >= count2)
            {
                maxactor = Allrecords[i].Actor1.ToString();
            }
            if (count1 <= count2)
            {
                maxactor = Allrecords[i].Actor2.ToString();
            }

        }

        count1 = 0;
        count2 = 0;
    }


    return maxactor;
}



#endregion

#region Task2

/// <summary>
/// Searches for movies that everyone saw
/// </summary>
/// <param name="register1"> first register </param>
/// <param name="register2"> second register </param>
/// <returns></returns>
public List<Record> EveryoneSaw(Register register1, Register register2)
{
    List<Record> Allsaw = new List<Record>();
    for (int i = 0; i < this.Allrecords.Count; i++)
    {
        if (register1.Allrecords.Contains(Allrecords[i]) &&
            register2.Allrecords.Contains(Allrecords[i]) &&
            !Allsaw.Contains(Allrecords[i]))
        {
            Allsaw.Add(Allrecords[i]);
        }
    }
    return Allsaw;
}
#endregion

#region Task3
/// <summary>
/// Finds recomendations for movie fan
/// </summary>
/// <param name="register1"></param>
/// <param name="register2"></param>
/// <returns> A list of recommendations </returns>
public List<Record> Recommendations(Register register1,
    Register register2)
{
    List<Record> recommendationsList = new List<Record>();
    for (int i = 0; i < this.Allrecords.Count; i++)
    {
        if (register1.Allrecords.Contains(register2.Allrecords[i])
            && !Allrecords.Contains(register2.Allrecords[i]))
        {
            recommendationsList.Add(register2.Allrecords[i]);
```

```
        }
    }
    return recommendationsList;
}
#endregion

#region task4

/// <summary>
/// Finds all new records
/// </summary>
/// <param name="NewFilms">list to add new records</param>
/// <returns>new record list</returns>
public List<Record> NewRecords (List<Record> NewFilms)
{
    try
    {
        for (int i = 0; i < RecordsCount(); i++)
        {
            Record curRecord = GetRecord(i);

            if (curRecord.IsNew() && !NewFilms.Contains(curRecord))
            {
                NewFilms.Add(curRecord);
            }
        }
        return NewFilms;
    }
    catch
    {
        throw new Exception("Nera nauju filmu");
    }
}

/// <summary>
/// Sorts new records
/// </summary>
/// <param name="records">new records</param>
/// <returns>sorted list</returns>
public List<Record> Sort(List<Record> records)
{
    try
    {
        if (records.Count > 0)
        {
            bool flag = true;
            while (flag)
            {
                flag = false;
                for (int i = 0; i < records.Count - 1; i++)
                {
                    if (records[i].CompareTo(records[i + 1]) > 0)
                    {
                        Record a = records[i];
                        Record b = records[i + 1];
                        records[i] = b;
                        records[i + 1] = a;
                        flag = true;
                    }
                }
            }
        }
        return records;
    }
    catch
    {
        throw new Exception("nera ka rikiuoti");
```

```
            }
        }

        #endregion

    }
}

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

namespace L4.App_BarCode
{
    /// <summary>
    /// Derrivative class
    /// </summary>
    class TVseries : Record, IComparable<Record>, IEquatable<Record>
    {
        public int StartYear { get; set; }
        public int EpisodeCount { get; set; }
        public string EndYear { get; set; }
        public bool Running { get; set; }

        /// <summary>
        /// Constructor for a serial object
        /// </summary>
        /// <param name="title"></param>
        /// <param name="genre"></param>
        /// <param name="movieStudio"></param>
        /// <param name="actor1"></param>
        /// <param name="actor2"></param>
        /// <param name="startyear"></param>
        /// <param name="episodeCount"></param>
        /// <param name="endYear"></param>
        /// <param name="running"></param>
        public TVseries(string title, string genre, string movieStudio,
            string actor1, string actor2, int startyear,
            int episodeCount, string endYear, bool running)
            : base(title, genre, movieStudio, actor1, actor2)
        {
            this.StartYear = startyear;
            this.EpisodeCount = episodeCount;
            this.EndYear = endYear;
            this.Running = running;
        }

        /// <summary>
        /// Method that formats string to be outputed to CSV file
        /// </summary>
        /// <returns></returns>
        public string ToCSV()
        {
            return string.Format("{0}; {1}; {2}; {3};" +
                                 " {4}; {5}; {6}; {7}; {8};",
               this.Title, this.Genre, this.MovieStudio,
               this.Actor1, this.Actor2, this.StartYear,
               this.EpisodeCount, this.EndYear, this.Running);
        }

        /// <summary>
        /// Formats a string to a table format
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
```

```
            return string.Format("| {0, -25}| {1, -14}|" +
                " {2, -20}| {3, -30}|" +
                            " {4, -30}| {5, -8}| {6, -25}| " +
                            "{7, -10}| {8, 10} |",
                            this.Title, this.Genre, this.MovieStudio,
                            this.Actor1,this.Actor2, this.StartYear,
                            this.EpisodeCount,this.EndYear, this.Running);
        }

        /// <summary>
        /// Compares two records
        /// </summary>
        /// <param name="other">other record</param>
        /// <returns></returns>
        public int CompareTo(TVseries other)
        {


            TVseries temp = (other as TVseries);
            if (this.StartYear == temp.StartYear)
            {
                return temp.EndYear.CompareTo(this.EndYear);
            }
            else
                return 0;

        }

        /// <summary>
        /// Checks if two records are equal
        /// </summary>
        /// <param name="other">other record</param>
        /// <returns></returns>
        public override bool Equals(Record other)
        {
            if (other is TVseries)
            {
                TVseries temp = (other as TVseries);
                if (this.Title.CompareTo(other.Title) == 0
                    && this.MovieStudio.CompareTo(temp.MovieStudio) == 0)
                    return true;
                else
                    return false;
            }
            else
                return false;
        }

        public override bool IsNew()
        {
            return DateTime.Now.Year - StartYear < 1;
        }
    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs" Inherits="L4.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <link href="Style.css" rel="stylesheet" />
    <title></title>
</head>
<body>
```

```
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label3" runat="server"></asp:Label>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Skaičiuoti" />
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Orginalūs duomenys:"></asp:Label>
            <br />
            <asp:Table ID="Table1" runat="server" GridLines="Both">
            </asp:Table>
            <br />
            <br />
            <asp:Label ID="Label2" runat="server" Text="Mėgiamiausi aktoriai:" Font-
Bold="True"></asp:Label>
            <asp:Table ID="Table2" runat="server" GridLines="Both">
            </asp:Table>
            <br />
            <br />
            <asp:Label ID="Label4" runat="server" Text="Nematyti filmai ir serialai" Font-
Bold="True"></asp:Label>
            <asp:Table ID="Table3" runat="server" GridLines="Both"></asp:Table>
            <br />
            <asp:Table ID="Table4" runat="server" GridLines="Both"></asp:Table>
            <br />
            <br />
            <asp:Label ID="Label5" runat="server" Text="Rekomendacijos" Font-
Bold="True"></asp:Label>
            <br />
            <asp:Label ID="Label7" runat="server" Text="Pirmo fano rekomendacijos"></asp:Label>
            <asp:Table ID="Table5" runat="server" GridLines="Both"></asp:Table>
            <br />
            <asp:Table ID="Table6" runat="server" GridLines="Both"></asp:Table>
            <br />
            <asp:Label ID="Label8" runat="server" Text="Antro fano rekomendacijos"></asp:Label>
            <asp:Table ID="Table9" runat="server" GridLines="Both"></asp:Table>
            <br />
            <asp:Table ID="Table10" runat="server" GridLines="Both"></asp:Table>
            <br />
            <asp:Label ID="Label9" runat="server" Text="Trečio fano rekomendacijos"></asp:Label>
            <asp:Table ID="Table11" runat="server" GridLines="Both"></asp:Table>
            <br />
            <asp:Table ID="Table12" runat="server" GridLines="Both"></asp:Table>
            <br />

            <br />
            <br />
            <asp:Label ID="Label6" runat="server" Text="Nauji filmai ir serialai" Font-
Bold="True"></asp:Label>
            <asp:Table ID="Table7" runat="server" GridLines="Both"></asp:Table>
            <br />
            <asp:Table ID="Table8" runat="server" GridLines="Both"></asp:Table>
        </div>
    </form>
</body>
</html>
```

Forma1.aspx.cs

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

```
using System.Text;
using System.IO;
namespace L4
{
    public partial class Forma1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                Console.OutputEncoding = Encoding.UTF8;

                #region DataAndFiles
                const string everyonesaw = "MateVisi.csv";
                const string originalData = "OriginalData.txt";
                const string newrecords = "Nauji.csv";

                File.Delete(Server.MapPath(originalData));
                File.Delete(Server.MapPath(newrecords));
                File.Delete(Server.MapPath(everyonesaw));

                string[] files = Directory.GetFiles
                    (Server.MapPath("App_Data/Case1"));
                string firstfile = files[0];
                string secondfile = files[1];
                string thirdfile = files[2];
                #endregion

                App_BarCode.Register register1 =
                    App_BarCode.InOut.ReadRecords(firstfile);
                App_BarCode.InOut.OutputRegisterToTxt
                    (Server.MapPath(originalData), register1);
                App_BarCode.InOut.OutputRegisterToTable
                    (Table1, register1);

                App_BarCode.Register register2 =
                    App_BarCode.InOut.ReadRecords(secondfile);
                App_BarCode.InOut.OutputRegisterToTxt
                    (Server.MapPath(originalData), register2);
                App_BarCode.InOut.OutputRegisterToTable
                    (Table1, register2);

                App_BarCode.Register register3 =
                    App_BarCode.InOut.ReadRecords(thirdfile);
                App_BarCode.InOut.OutputRegisterToTxt
                    (Server.MapPath(originalData), register3);
                App_BarCode.InOut.OutputRegisterToTable
                    (Table1, register3);

                #region Task1
                App_BarCode.InOut.OutputRow(Table2, new object[] { "Kino mėgėjas",
"Mėgstamiausias aktorius" });

                string firstfavorite =
                    register1.FavoriteActor();
                App_BarCode.InOut.OutputFavoriteActor
                    (Table2, firstfavorite, register1);

                string secondfavorite =
                    register2.FavoriteActor();
                App_BarCode.InOut.OutputFavoriteActor
                    (Table2, secondfavorite, register2);
```

```
string thirdfavorite =
    register3.FavoriteActor();
App_BarCode.InOut.OutputFavoriteActor
    (Table2, thirdfavorite, register3);

#endregion

#region Task2
List<App_BarCode.Record> EveryoneSaw =
    register1.EveryoneSaw(register2,
    register3);
App_BarCode.InOut.OutputMoviesThatEveryoneSaw
    (Server.MapPath(everyonesaw), EveryoneSaw);

App_BarCode.InOut.FilterAndOutputFilms(Table3, EveryoneSaw);
App_BarCode.InOut.FilterAndOutputTVseries(Table4, EveryoneSaw);

#endregion

#region Task3


List<App_BarCode.Record> Reccomendations1 =
    register1.Recommendations(register3, register2);
string recomendationsFile1 = string.Format
    ("Recomendacija_{0}_{1}.csv",
    register1.FanName, register1.FanLast);
App_BarCode.InOut.OutputRecomendations
    (Server.MapPath(recomendationsFile1), Reccomendations1);

Label7.Text = $"{register1.FanName} {register1.FanLast} rekomenduoja:";
App_BarCode.InOut.FilterAndOutputFilms(Table5, Reccomendations1);
App_BarCode.InOut.FilterAndOutputTVseries(Table6, Reccomendations1);


List<App_BarCode.Record> Reccomendations2 =
    register2.Recommendations(register1, register3);
string recomendationsFile2 = string.Format
    ("Recomendacija_{0}_{1}.csv", register2.FanName,
    register2.FanLast);
App_BarCode.InOut.OutputRecomendations
    (Server.MapPath(recomendationsFile2), Reccomendations2);

Label8.Text = $"{register2.FanName} {register2.FanLast} rekomenduoja:";
App_BarCode.InOut.FilterAndOutputFilms(Table9, Reccomendations2);
App_BarCode.InOut.FilterAndOutputTVseries(Table10, Reccomendations2);


List<App_BarCode.Record> Reccomendations3 =
    register3.Recommendations(register1, register2);
string recomendationsFile3 = string.Format
    ("Recomendacija_{0}_{1}.csv", register3.FanName,
    register3.FanLast);
App_BarCode.InOut.OutputRecomendations
    (Server.MapPath(recomendationsFile3), Reccomendations3);

Label9.Text = $"{register3.FanName} {register3.FanLast} rekomenduoja:";
App_BarCode.InOut.FilterAndOutputFilms(Table11, Reccomendations3);
App_BarCode.InOut.FilterAndOutputTVseries(Table12, Reccomendations3);

#endregion

#region Task4

List<App_BarCode.Record> NewRecords =
    new List<App_BarCode.Record>();
```

```
                NewRecords = register1.NewRecords(NewRecords);
                NewRecords = register2.NewRecords(NewRecords);
                NewRecords = register3.NewRecords(NewRecords);

                NewRecords = register1.Sort(NewRecords);

                App_BarCode.InOut.OutputNewRecords
                    (NewRecords, Server.MapPath(newrecords));

                App_BarCode.InOut.FilterAndOutputFilms(Table7, NewRecords);
                App_BarCode.InOut.FilterAndOutputTVseries(Table8, NewRecords);

                #endregion
            }
            catch
            {
                Label3.Text = "Duomenų failai yra blogame formate" +
                    "/tušti/netinkamoje vietoje/ar per mažai duomenų failų";
            }
        }
    }
}
```

Style.css

```
body {
    background-color: burlywood;
}

table {
    border-color: darkred;
    font-family: Arial,'Agency FB';
}

input {
    background-color: aquamarine;
    border: solid 1px black;
}
```

4.7. Pradiniai duomenys ir rezultatai

Testas 1 – duoti duomenys reikalingi užpildyti visus punktus

Duomenys:
Pirmasis failas:

```
Vytautas Kervedis
1912
Vilnius
FILM;Pjuklo Ketera;Trileris;Electric Studio;Nicolas Cage;Scarlet Johanson;2021;Arvydas Lempa;220
FILM;Geriausi mūsų metai;Drama;Italian Cinema;Harrisonas Fordas;Morganas Freemanas;1999;Arvydas Lempa;220
FILM;Kaimynai;Komedija;Pasakos Studija;Eddie Murphy;Tomas Cruise'as;1999;Arvydas Lempa;220
FILM;Raudonkepuraitė;Drama;Marvel studios;Robertas Downey jaunesnysis;Kestas Blalaba;1999;Arvydas Lempa;1000
FILM;Naktis Muziejuje;Komędija;WB production;Robinas Williamsas;Nicolas Cage;1999;Arvydas Lempa;200
FILM;Matrica;Veiksmo;Kiškio studija;Nicolas Cage;Jennifer Lopez;1999;Arvydas Lempa;50
TVSERIES;Žuviukas Nemo;Animacinis;Pixar Studija;Aistė Valdžytė;Nicolas Cage;2022;168;2022;Ne
FILM;Ratai 2;Animacinis;Pixar Studija;Augustas Višinskas;Gabija Arnašiūtė;1999;Arvydas Lempa;800
```

Antrasis failas:

```
Vinas Dyzelis
2015
Kaunas
FILM;Matrica;Veiksmo;Kiškio studija;Nicolas Cage;Jennifer Lopez;1999;Arvydas Lempa;50
FILM;Išrinktasis;Dramaturgas;Batuotas Katinas;Connor Mcgregor;Viktoras Medis;1999;Arvydas Lempa;120
FILM;Geriausi mūsų metai;Drama;Italian Cinema;Harrisonas Fordas;Jennifer Lopez;1999;Arvydas Lempa;220
FILM;Raudonkepuraita;Drama;Marvel studios;Jennifer Lopez;Jennifer Lopez;1999;Arvydas Lempa;1000
FILM;Naktis Muziejuje;Komędija;WB production;Robinas Williamsas;Nicolas Cage;1999;Arvydas Lempa;200
FILM;Ratai 2;Animacinis;Pixar Studija;Augustas Višinskas;Gabija Arnašiūtė;2022;Arvydas Lempa;800
FILM;Žuviukas Nemo;Animacinis;Pixar Studija;Aistė Valdžytė;Arūnas Beržas;1999;Arvydas Lempa;999
TVSERIES;Kurmis;Animacinis;Pixar Studija;Aistė Valdžytė;Nicolas Cage;2022;168;Taip
```

Trečiasis failas:

```
Alvydas Ežerinskis
1988
Šilutė
FILM;Matrica;Veiksmo;Kiškio studija;Nicolas Cage;Jennifer Lopez;1999;Arvydas Lempa;50
FILM;Išrinktasis;Dramaturgas;Batuotas Katinas;Connor Mcgregor;Viktoras Medis;1999;Arvydas Lempa;120
FILM;Geriausi mūsų metai;Drama;Italian Cinema;Harrisonas Fordas;Morganas Freemanas;2022;Arvydas Lempa;220
FILM;Raudonkepuraitė;Drama;Marvel studios;JNicolas Cage;Robertas Downey jaunesnysis;1999;Arvydas Lempa;1000
FILM;Naktis Muziejuje;OoogaBooga;WB production;Robinas Williamsas;Nicolas Cage;1999;Arvydas Lempa;200
FILM;Ratai 2;Animacinis;Pixar Studija;Augustas Višinskas;Gabija Arnašiūtė;1999;Arvydas Lempa;800
FILM;Žuviukas Nemo;Animacinis;Pixar Studija;Aistė Valdžytė;Arūnas Beržas;1999;Arvydas Lempa;999
TVSERIES;Tomis;Animacinis;WB production;Aistė Valdžytė;Nicolas Cage;2022;168;2022;Ne
```

Rezultatai:

OriginalData.txt

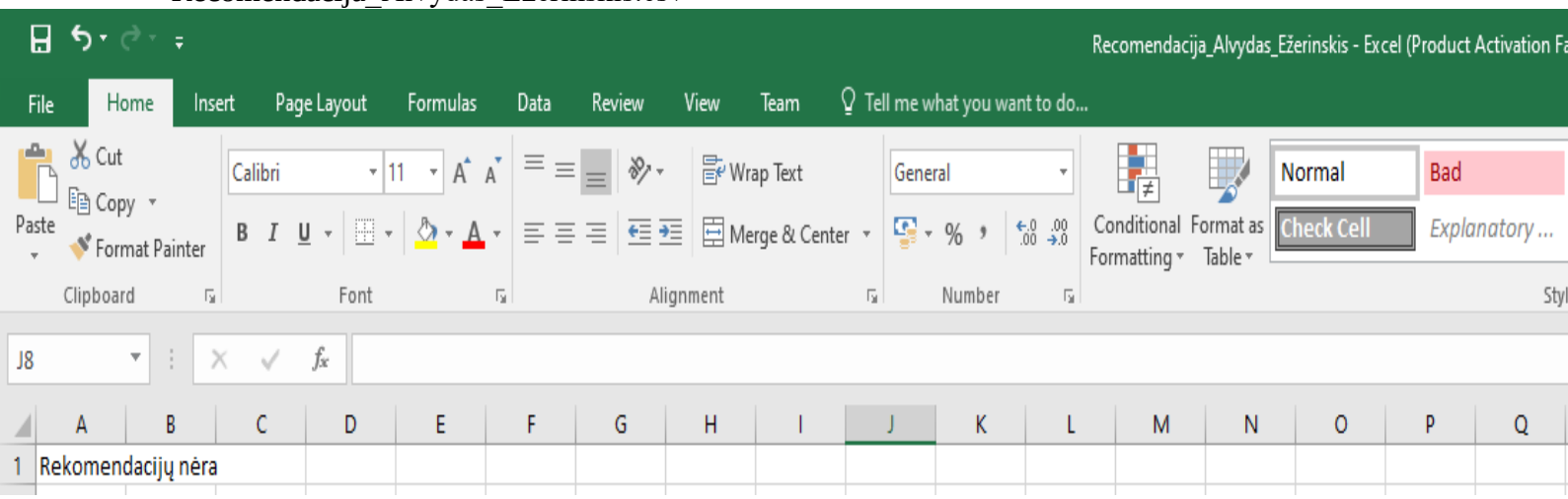
OriginalData - Notepad

File Edit Format View Help

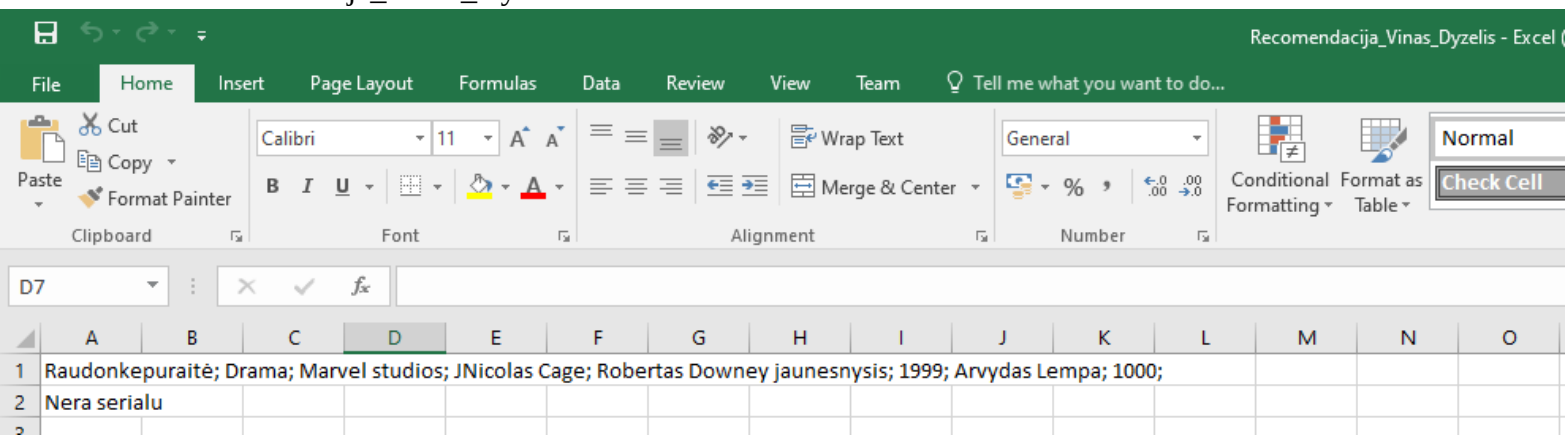
```
----------------------------------------
| Kinomano mėgėjo vardas ir pavardė:   |
| Vytautas Kervedis                    |
----------------------------------------
| Kinomano mėgėjo gimimo metai:        |
| 1912                                 |
----------------------------------------
| Kinomano mėgėjo miestas:             |
| Vilnius                              |
----------------------------------------
| Pjuklo Ketera          | Trileris      | Electric Studio      | Nicolas Cage                 | Scarlet Johanson             | 2021 | Arvydas Lempa | 220 |          |
| Geriausi mūsų metai    | Drama         | Italian Cinema       | Harrisonas Fordas            | Morganas Freemanas           | 1999 | Arvydas Lempa | 220 |          |
| Kaimynai               | Komedija      | Pasakos Studija      | Eddie Murphy                 | Tomas Cruise'as              | 1999 | Arvydas Lempa | 220 |          |
| Raudonkepuraitė        | Drama         | Marvel studios       | Robertas Downey jaunesnysis  | Kestas Blalaba               | 1999 | Arvydas Lempa | 1000 |         |
| Naktis Muziejuje       | Komędija      | WB production        | Robinas Williamsas           | Nicolas Cage                 | 1999 | Arvydas Lempa | 200 |          |
| Matrica                | Veiksmo       | Kiškio studija       | Nicolas Cage                 | Jennifer Lopez               | 1999 | Arvydas Lempa | 50 |          |
| Žuviukas Nemo          | Animacinis    | Pixar Studija        | Aistė Valdžytė               | Nicolas Cage                 | 2022 | 168           | 2022 |      False |
| Ratai 2                | Animacinis    | Pixar Studija        | Augustas Višinskas           | Gabija Arnašiūtė             | 1999 | Arvydas Lempa | 800 |          |
----------------------------------------

----------------------------------------
| Kinomano mėgėjo vardas ir pavardė:   |
| Vinas Dyzelis                        |
----------------------------------------
| Kinomano mėgėjo gimimo metai:        |
| 2015                                 |
----------------------------------------
| Kinomano mėgėjo miestas:             |
| Kaunas                               |
----------------------------------------
| Matrica                | Veiksmo       | Kiškio studija       | Nicolas Cage                 | Jennifer Lopez               | 1999 | Arvydas Lempa | 50 |          |
| Išrinktasis            | Dramaturgas   | Batuotas Katinas     | Connor Mcgregor              | Viktoras Medis               | 1999 | Arvydas Lempa | 120 |          |
| Geriausi mūsų metai    | Drama         | Italian Cinema       | Harrisonas Fordas            | Jennifer Lopez               | 1999 | Arvydas Lempa | 220 |          |
| Raudonkepuraita        | Drama         | Marvel studios       | Jennifer Lopez               | Jennifer Lopez               | 1999 | Arvydas Lempa | 1000 |         |
| Naktis Muziejuje       | Komędija      | WB production        | Robinas Williamsas           | Nicolas Cage                 | 1999 | Arvydas Lempa | 200 |          |
| Ratai 2                | Animacinis    | Pixar Studija        | Augustas Višinskas           | Gabija Arnašiūtė             | 2022 | Arvydas Lempa | 800 |          |
| Žuviukas Nemo          | Animacinis    | Pixar Studija        | Aistė Valdžytė               | Arūnas Beržas                | 1999 | Arvydas Lempa | 999 |          |
| Kurmis                 | Animacinis    | Pixar Studija        | Aistė Valdžytė               | Nicolas Cage                 | 2022 | 168           | - |        True |
----------------------------------------

----------------------------------------
| Kinomano mėgėjo vardas ir pavardė:   |
| Alvydas Ežerinskis                   |
----------------------------------------
| Kinomano mėgėjo gimimo metai:        |
| 1988                                 |
----------------------------------------
| Kinomano mėgėjo miestas:             |
| Šilutė                               |
----------------------------------------
| Matrica                | Veiksmo       | Kiškio studija       | Nicolas Cage                 | Jennifer Lopez               | 1999 | Arvydas Lempa | 50 |          |
| Išrinktasis            | Dramaturgas   | Batuotas Katinas     | Connor Mcgregor              | Viktoras Medis               | 1999 | Arvydas Lempa | 120 |          |
| Geriausi mūsų metai    | Drama         | Italian Cinema       | Harrisonas Fordas            | Morganas Freemanas           | 2022 | Arvydas Lempa | 220 |          |
| Raudonkepuraitė        | Drama         | Marvel studios       | JNicolas Cage                | Robertas Downey jaunesnysis  | 1999 | Arvydas Lempa | 1000 |         |
| Naktis Muziejuje       | OoogaBooga    | WB production        | Robinas Williamsas           | Nicolas Cage                 | 1999 | Arvydas Lempa | 200 |          |
| Ratai 2                | Animacinis    | Pixar Studija        | Augustas Višinskas           | Gabija Arnašiūtė             | 1999 | Arvydas Lempa | 800 |          |
| Žuviukas Nemo          | Animacinis    | Pixar Studija        | Aistė Valdžytė               | Arūnas Beržas                | 1999 | Arvydas Lempa | 999 |          |
| Tomis                  | Animacinis    | WB production        | Aistė Valdžytė               | Nicolas Cage                 | 2022 | 168           | 2022 |      False |
----------------------------------------
```

Rekomendacijos:

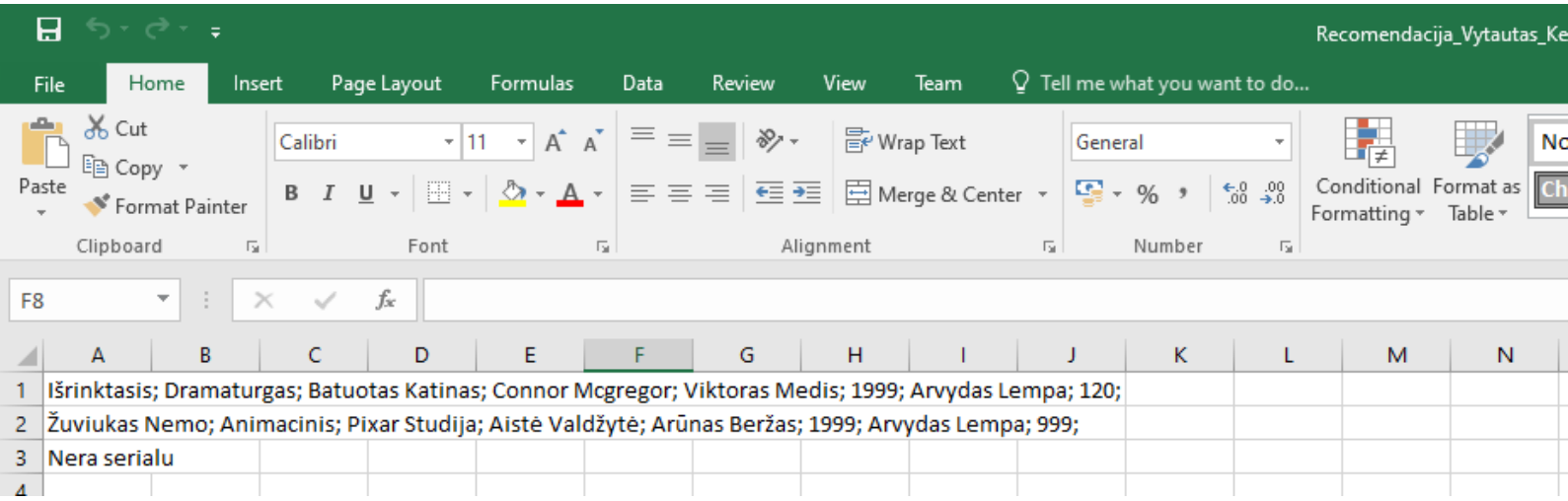
Recomendacija_Alvydas_Ežerinskis.csv



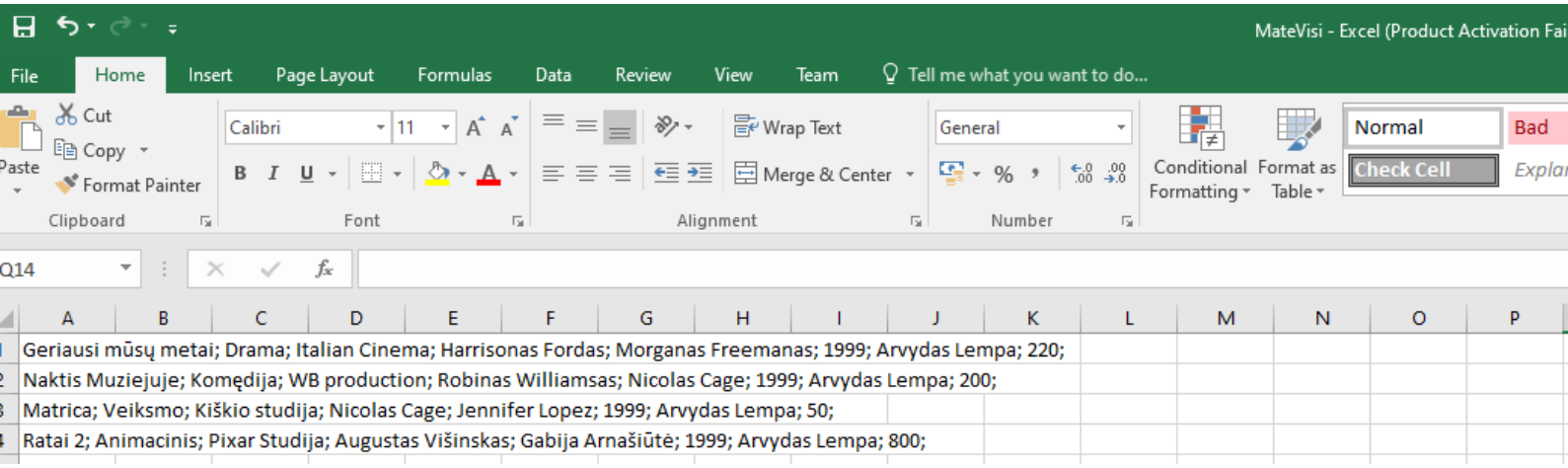
Recomendacija_Vinas_Dyzelis.csv



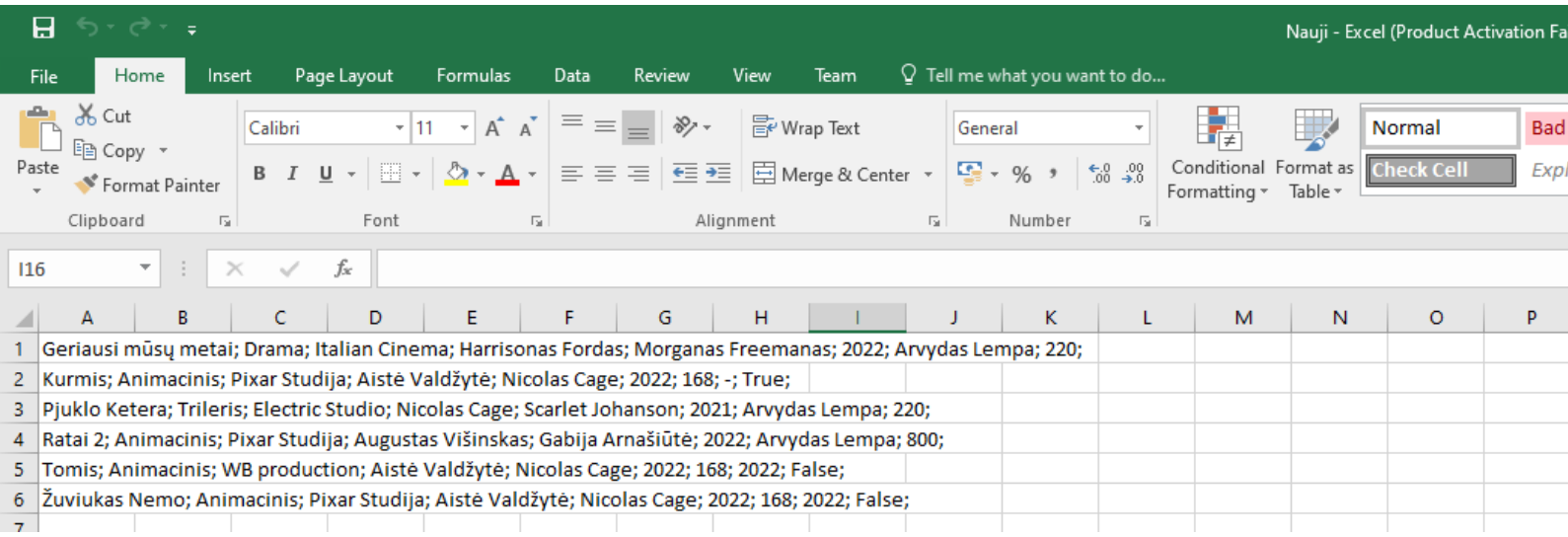
Recomendacija_Vytautas_Kervedis.csv



MateVisi.csv



Nauji.csv



Vartotojo sąsaja:

http://localhost:59570/Forma1.aspx

localhost

Skaičiuoti

Orginalūs duomenys:

Vytautas	Kervedis							
1912								
Vilnius								
Pjuklo Ketera	Trileris	Electric Studio	Nicolas Cage	Scarlet Johanson	2021	Arvydas Lempa	220	
Geriausi mūsų metai	Drama	Italian Cinema	Harrisonas Fordas	Morganas Freemanas	1999	Arvydas Lempa	220	
Kaimynai	Komedija	Pasakos Studija	Eddie Murphy	Tomas Cruise'as	1999	Arvydas Lempa	220	
Raudonkepuraitė	Drama	Marvel studios	Robertas Downey jaunesnysis	Kestas Blalaba	1999	Arvydas Lempa	1000	
Naktis Muziejuje	Komędija	WB production	Robinas Williamsas	Nicolas Cage	1999	Arvydas Lempa	200	
Matrica	Veiksmo	Kiškio studija	Nicolas Cage	Jennifer Lopez	1999	Arvydas Lempa	50	
Žuviukas Nemo	Animacinis	Pixar Studija	Aistė Valdžytė	Nicolas Cage	2022	168	2022	Nesitesi
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Gabija Arnašiūtė	1999	Arvydas Lempa	800	
Vinas	Dyzelis							
2015								
Kaunas								
Matrica	Veiksmo	Kiškio studija	Nicolas Cage	Jennifer Lopez	1999	Arvydas Lempa	50	
Išrinktasis	Dramaturgas	Batuotas Katinas	Connor Mcgregor	Viktoras Medis	1999	Arvydas Lempa	120	
Geriausi mūsų metai	Drama	Italian Cinema	Harrisonas Fordas	Jennifer Lopez	1999	Arvydas Lempa	220	
Raudonkepuraita	Drama	Marvel studios	Jennifer Lopez	Jennifer Lopez	1999	Arvydas Lempa	1000	
Naktis Muziejuje	Komędija	WB production	Robinas Williamsas	Nicolas Cage	1999	Arvydas Lempa	200	
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Gabija Arnašiūtė	2022	Arvydas Lempa	800	
Žuviukas Nemo	Animacinis	Pixar Studija	Aistė Valdžytė	Arūnas Beržas	1999	Arvydas Lempa	999	
Kurmis	Animacinis	Pixar Studija	Aistė Valdžytė	Nicolas Cage	2022	168	-	Tęsiasi
Alvydas	Ežerinskis							
1988								
Šilutė								
Matrica	Veiksmo	Kiškio studija	Nicolas Cage	Jennifer Lopez	1999	Arvydas Lempa	50	
Išrinktasis	Dramaturgas	Batuotas Katinas	Connor Mcgregor	Viktoras Medis	1999	Arvydas Lempa	120	
Geriausi mūsų metai	Drama	Italian Cinema	Harrisonas Fordas	Morganas Freemanas	2022	Arvydas Lempa	220	
Raudonkepuraitė	Drama	Marvel studios	JNicolas Cage	Robertas Downey jaunesnysis	1999	Arvydas Lempa	1000	
Naktis Muziejuje	OoogaBooga	WB production	Robinas Williamsas	Nicolas Cage	1999	Arvydas Lempa	200	
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Gabija Arnašiūtė	1999	Arvydas Lempa	800	
Žuviukas Nemo	Animacinis	Pixar Studija	Aistė Valdžytė	Arūnas Beržas	1999	Arvydas Lempa	999	
Tomis	Animacinis	WB production	Aistė Valdžytė	Nicolas Cage	2022	168	2022	Nesitesi

Mėgiamiausi aktoriai:

Kino mėgėjas	Mėgstamiausias aktorius
Vytautas Kervedis	Nicolas Cage
Vinas Dyzelis	Jennifer Lopez
Alvydas Ežerinskis	Nicolas Cage

Nematyti filmai ir serialai

Filmai

Pavadinimas	Žanras	Kino studija	Pagr. aktorius 1	Pagr. aktorius 2	Leidimo metai	Režisierius	Pajamos
Geriausi mūsų metai	Drama	Italian Cinema	Harrisonas Fordas	Morganas Freemanas	1999	Arvydas Lempa	220
Naktis Muziejuje	Komędija	WB production	Robinas Williamsas	Nicolas Cage	1999	Arvydas Lempa	200
Matrica	Veiksmo	Kiškio studija	Nicolas Cage	Jennifer Lopez	1999	Arvydas Lempa	50
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Gabija Arnašiūtė	1999	Arvydas Lempa	800

Nėra serialų!

Rekomendacijos

Vytautas Kervedis rekomenduoja:

Filmai

Pavadinimas	Žanras	Kino studija	Pagr. aktorius 1	Pagr. aktorius 2	Leidimo metai	Režisierius	Pajamos
Išrinktasis	Dramaturgas	Batuotas Katinas	Connor Mcgregor	Viktoras Medis	1999	Arvydas Lempa	120
Žuviukas Nemo	Animacinis	Pixar Studija	Aistė Valdžytė	Arūnas Beržas	1999	Arvydas Lempa	999

Nėra serialų!

Vinas Dyzelis rekomenduoja:

Filmai

Pavadinimas	Žanras	Kino studija	Pagr. aktorius 1	Pagr. aktorius 2	Leidimo metai	Režisierius	Pajamos
Raudonkepuraitė	Drama	Marvel studios	JNicolas Cage	Robertas Downey jaunesnysis	1999	Arvydas Lempa	1000

Nėra serialų!

Alvydas Ežerinskis rekomenduoja:

Nėra filmų!

Nėra serialų!

Nauji filmai ir serialai

Filmai

Pavadinimas	Žanras	Kino studija	Pagr. aktorius 1	Pagr. aktorius 2	Leidimo metai	Režisierius	Pajamos
Geriausi mūsų metai	Drama	Italian Cinema	Harrisonas Fordas	Morganas Freemanas	2022	Arvydas Lempa	220
Pjuklo Ketera	Trileris	Electric Studio	Nicolas Cage	Scarlet Johanson	2021	Arvydas Lempa	220
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Gabija Arnašiūtė	2022	Arvydas Lempa	800

Serialai

Pavadinimas	Žanras	Kino studija	Pagr. aktorius 1	Pagr. aktorius 2	Pradžios metai	Serijų kiekis	Pabaigos metai	Ar tęsiasi
Kurmis	Animacinis	Pixar Studija	Aistė Valdžytė	Nicolas Cage	2022	168	-	True
Tomis	Animacinis	WB production	Aistė Valdžytė	Nicolas Cage	2022	168	2022	False
Žuviukas Nemo	Animacinis	Pixar Studija	Aistė Valdžytė	Nicolas Cage	2022	168	2022	False

Testas 2 – nėra naujų filmų ar serialų

Duomenys:

Pirmas duomenų failas:

```
Mahatma Gandis
1912
Vilnius
FILM;Pjuklo Ketera;Trileris;Electric Studio;Nicolas Cage;Scarlet Johanson;1999;Aistė Valdžytė;220
FILM;Geriausi mūsų metai;Drama;Italian Cinema;Harrisonas Fordas;Aistė Valdžytė;1999;Arvydas Lempa;220
FILM;Kaimynai;Komedija;Pasakos Studija;Eddie Murphy;Aistė Valdžytė;1999;Arvydas Lempa;220
FILM;Raudonkepuraitė;Drama;Marvel studios;Robertas Downey jaunesnysis;Kestas Blalaba;1999;Arvydas Lempa;1000
FILM;Naktis Muziejuje;Komędija;WB production;Aistė Valdžytė;Nicolas Cage;1999;Arvydas Lempa;200
FILM;Matrica;Veiksmo;Kiškio studija;Nicolas Cage;Jennifer Lopez;1999;Arvydas Lempa;50
TVSERIES;Žuviukas Nemo;Animacinis;Pixar Studija;Aistė Valdžytė;Nicolas Cage;2000;168;2021;Ne
FILM;Ratai 2;Animacinis;Pixar Studija;Augustas Višinskas;Aistė Valdžytė;1999;Arvydas Lempa;800
```

Antras duomenų failas:

```
Kęstas Pajūris
2015
Kaunas
FILM;Matrica;Veiksmo;Kiškio studija;Nicolas Cage;Connor Mcgregor;1999;Arvydas Lempa;50
FILM;Išrinktasis;Dramaturgas;Batuotas Katinas;Connor Mcgregor;Viktoras Medis;1999;Arvydas Lempa;120
FILM;Geriausi mūsų metai;Drama;Italian Cinema;Connor Mcgregor;Jennifer Lopez;1999;Arvydas Lempa;220
FILM;Raudonkepuraita;Drama;Marvel studios;Jennifer Lopez;Connor Mcgregor;1999;Arvydas Lempa;1000
FILM;Naktis Muziejuje;Komędija;WB production;Robinas Williamsas;Nicolas Cage;1999;Arvydas Lempa;200
FILM;Ratai 2;Animacinis;Pixar Studija;Augustas Višinskas;Connor Mcgregor;1999;Arvydas Lempa;800
FILM;Žuviukas Nemo;Animacinis;Pixar Studija;Aistė Valdžytė;Connor Mcgregor;1999;Arvydas Lempa;999
TVSERIES;Kurmis;Animacinis;Pixar Studija;Aistė Valdžytė;Connor Mcgregor;2000;168;2021;Ne
```

Trečias duomenų failas:

```
Vinas Benzinas
1988
Šilutė
FILM;Namas;Veiksmo;Kiškio studija;Nicolas Cage;Jennifer Lopez;1999;Arvydas Lempa;50
FILM;Ratas;Dramaturgas;Batuotas Katinas;Connor Mcgregor;Viktoras Medis;1999;Arvydas Lempa;120
FILM;Kęlmas;Drama;Italian Cinema;Harrisonas Fordas;Morganas Freemanas;1999;Arvydas Lempa;220
FILM;Raudonkepuraitis;Drama;Marvel studios;JNicolas Cage;Augustas Višinskas;1999;Arvydas Lempa;1000
FILM;Naktis Paryžiuje;OoogaBooga;WB production;Robinas Williamsas;Augustas Višinskas;1999;Arvydas Lempa;200
FILM;Ratai 6;Animacinis;Pixar Studija;Augustas Višinskas;Gabija Arnašiūtė;1999;Arvydas Lempa;800
FILM;Žuvytė Dorė;Animacinis;Pixar Studija;Augustas Višinskas;Arūnas Beržas;1999;Arvydas Lempa;999
TVSERIES;Lietuva;Animacinis;Pixar Studija;Augustas Višinskas;Nicolas Cage;2000;168;2021;Ne
```

Rezultatai:

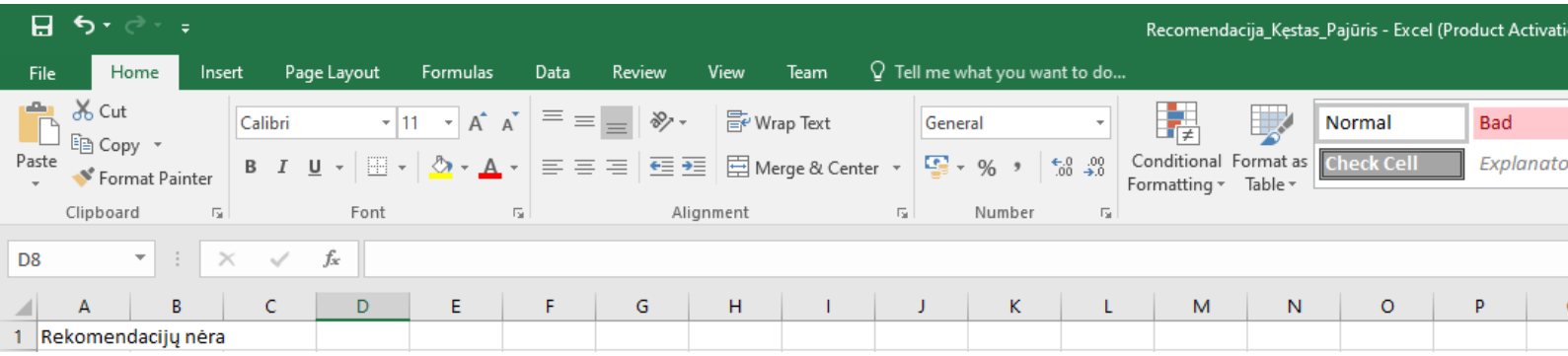
OriginalData.txt

OriginalData - Notepad

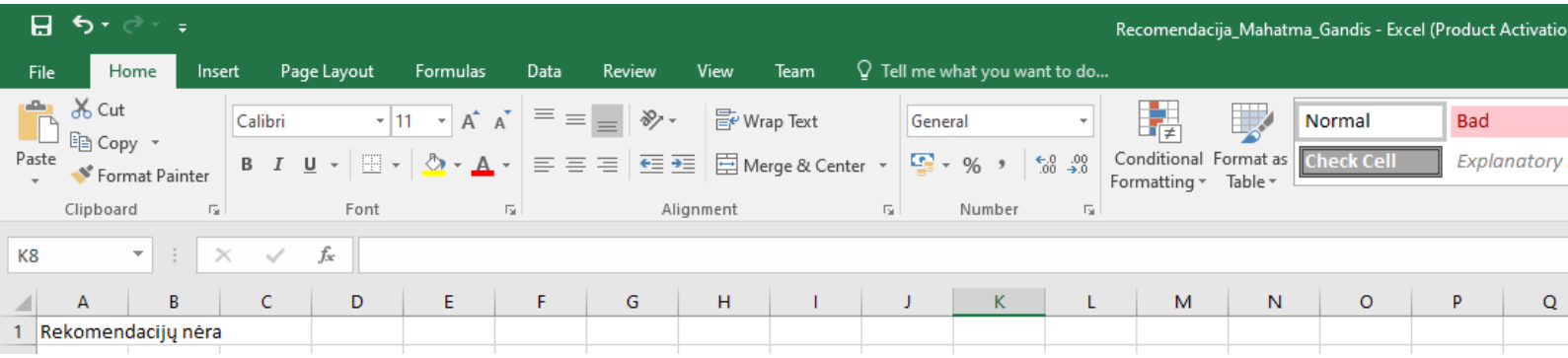
File Edit Format View Help

```
| Kinomano mėgėjo vardas ir pavardė:   |
| Mahatma Gandis                       |
| Kinomano mėgėjo gimimo metai:        |
| 1912                                 |
| Kinomano mėgėjo miestas:             |
| Vilnius                              |

| Pjuklo Ketera          | Trileris     | Electric Studio     | Nicolas Cage                | Scarlet Johanson            | 1999 | Aistė Valdžytė          | 220  |          |
| Geriausi mūsų metai    | Drama        | Italian Cinema      | Harrisonas Fordas           | Aistė Valdžytė              | 1999 | Arvydas Lempa           | 220  |          |
| Kaimynai               | Komedija     | Pasakos Studija     | Eddie Murphy                | Aistė Valdžytė              | 1999 | Arvydas Lempa           | 220  |          |
| Raudonkepuraitė        | Drama        | Marvel studios      | Robertas Downey jaunesnysis | Kestas Blalaba              | 1999 | Arvydas Lempa           | 1000 |          |
| Naktis Muziejuje       | Komėdija     | WB production       | Aistė Valdžytė              | Nicolas Cage                | 1999 | Arvydas Lempa           | 200  |          |
| Matrica                | Veiksmo      | Kiškio studija      | Nicolas Cage                | Jennifer Lopez              | 1999 | Arvydas Lempa           | 50   |          |
| Žuviukas Nemo          | Animacinis   | Pixar Studija       | Aistė Valdžytė              | Nicolas Cage                | 2000 | 168                     | 2021 | False    |
| Ratai 2                | Animacinis   | Pixar Studija       | Augustas Višinskas          | Aistė Valdžytė              | 1999 | Arvydas Lempa           | 800  |          |

| Kinomano mėgėjo vardas ir pavardė:   |
| Kęstas Pajūris                       |
| Kinomano mėgėjo gimimo metai:        |
| 2015                                 |
| Kinomano mėgėjo miestas:             |
| Kaunas

| Matrica                | Veiksmo      | Kiškio studija      | Nicolas Cage                | Connor Mcgregor             | 1999 | Arvydas Lempa           | 50   |          |
| Išrinktasis            | Dramaturgas  | Batuotas Katinas    | Connor Mcgregor             | Viktoras Medis              | 1999 | Arvydas Lempa           | 120  |          |
| Geriausi mūsų metai    | Drama        | Italian Cinema      | Connor Mcgregor             | Jennifer Lopez              | 1999 | Arvydas Lempa           | 220  |          |
| Raudonkepuraita        | Drama        | Marvel studios      | Jennifer Lopez              | Connor Mcgregor             | 1999 | Arvydas Lempa           | 1000 |          |
| Naktis Muziejuje       | Komėdija     | WB production       | Robinas Williamsas          | Nicolas Cage                | 1999 | Arvydas Lempa           | 200  |          |
| Ratai 2                | Animacinis   | Pixar Studija       | Augustas Višinskas          | Connor Mcgregor             | 1999 | Arvydas Lempa           | 800  |          |
| Žuviukas Nemo          | Animacinis   | Pixar Studija       | Aistė Valdžytė              | Connor Mcgregor             | 1999 | Arvydas Lempa           | 999  |          |
| Kurmis                 | Animacinis   | Pixar Studija       | Aistė Valdžytė              | Connor Mcgregor             | 2000 | 168                     | 2021 | False    |

| Kinomano mėgėjo vardas ir pavardė:   |
| Vinas Benzinas                       |
| Kinomano mėgėjo gimimo metai:        |
| 1988                                 |
| Kinomano mėgėjo miestas:             |
| Šilutė

| Namas                  | Veiksmo      | Kiškio studija      | Nicolas Cage                | Jennifer Lopez              | 1999 | Arvydas Lempa           | 50   |          |
| Ratas                  | Dramaturgas  | Batuotas Katinas    | Connor Mcgregor             | Viktoras Medis              | 1999 | Arvydas Lempa           | 120  |          |
| Kėlmas                 | Drama        | Italian Cinema      | Harrisonas Fordas           | Morganas Freemanas          | 1999 | Arvydas Lempa           | 220  |          |
| Raudonkepuraitis       | Drama        | Marvel studios      | JNicolas Cage               | Augustas Višinskas          | 1999 | Arvydas Lempa           | 1000 |          |
| Naktis Paryžiuje       | OoogaBooga   | WB production       | Robinas Williamsas          | Augustas Višinskas          | 1999 | Arvydas Lempa           | 200  |          |
| Ratai 6                | Animacinis   | Pixar Studija       | Augustas Višinskas          | Gabija Arnašiūtė            | 1999 | Arvydas Lempa           | 800  |          |
| Žuvytė Dorė            | Animacinis   | Pixar Studija       | Augustas Višinskas          | Arūnas Beržas               | 1999 | Arvydas Lempa           | 999  |          |
| Lietuva                | Animacinis   | Pixar Studija       | Augustas Višinskas          | Nicolas Cage                | 2000 | 168                     | 2021 | False    |
```

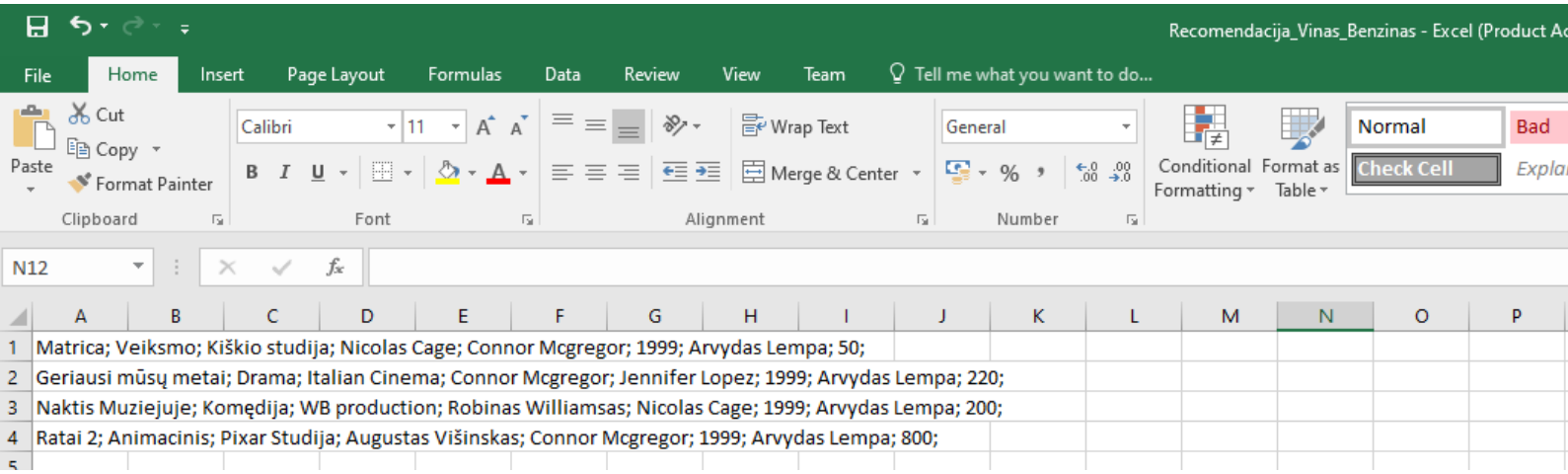
Recomendacija_Kęstas_Pajūris.csv



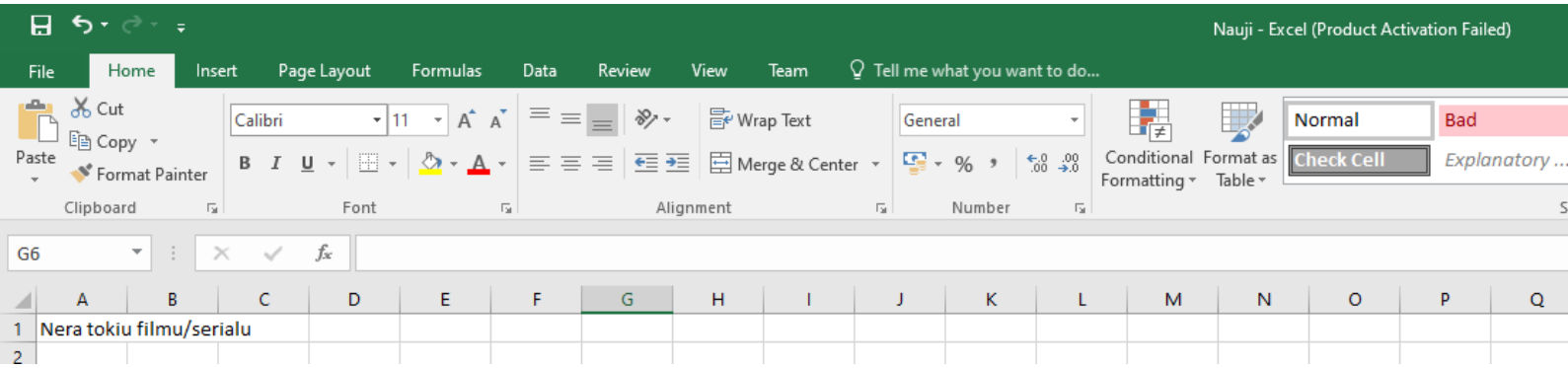
Recomendacija_Mahatma_Gandis.csv



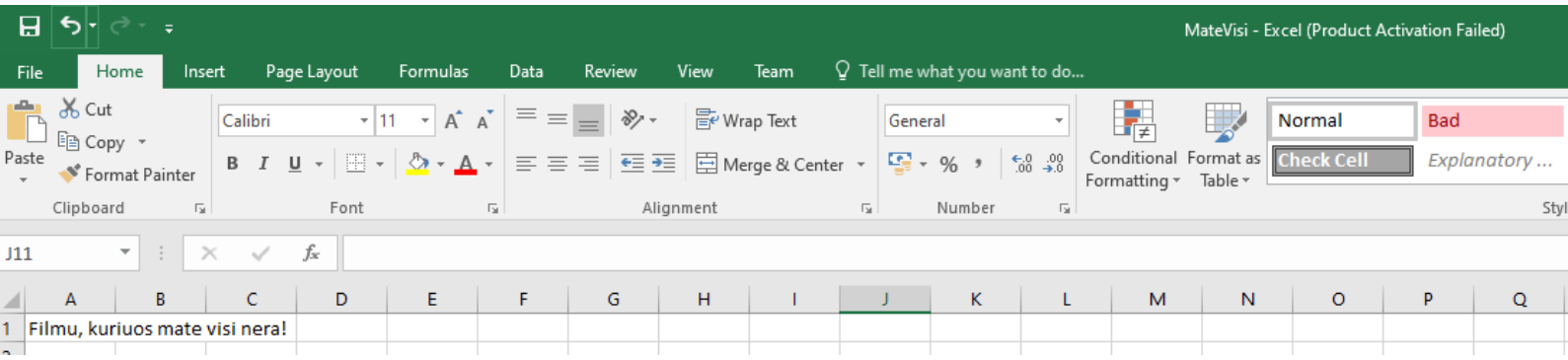
Recomendacija_Vinas_Benzinas.csv



Nauji.csv



MateVisi.csv



Vartotojo sasaja:

http://localhost:59570/Forma1.aspx

localhost

Skaičiuoti

Orginalūs duomenys:

Mahatma	Gandis							
1912								
Vilnius								
Pjuklo Ketera	Trileris	Electric Studio	Nicolas Cage	Scarlet Johanson	1999	Aistė Valdžytė	220	
Geriausi mūsų metai	Drama	Italian Cinema	Harrisonas Fordas	Aistė Valdžytė	1999	Arvydas Lempa	220	
Kaimynai	Komedija	Pasakos Studija	Eddie Murphy	Aistė Valdžytė	1999	Arvydas Lempa	220	
Raudonkepuraitė	Drama	Marvel studios	Robertas Downey jaunesnysis	Kestas Blalaba	1999	Arvydas Lempa	1000	
Naktis Muziejuje	Komędija	WB production	Aistė Valdžytė	Nicolas Cage	1999	Arvydas Lempa	200	
Matrica	Veiksmo	Kiškio studija	Nicolas Cage	Jennifer Lopez	1999	Arvydas Lempa	50	
Žuviukas Nemo	Animacinis	Pixar Studija	Aistė Valdžytė	Nicolas Cage	2000	168	2021	Nesitesi
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Aistė Valdžytė	1999	Arvydas Lempa	800	
Kęstas	Pajūris							
2015								
Kaunas								
Matrica	Veiksmo	Kiškio studija	Nicolas Cage	Connor Mcgregor	1999	Arvydas Lempa	50	
Išrinktasis	Dramaturgas	Batuotas Katinas	Connor Mcgregor	Viktoras Medis	1999	Arvydas Lempa	120	
Geriausi mūsų metai	Drama	Italian Cinema	Connor Mcgregor	Jennifer Lopez	1999	Arvydas Lempa	220	
Raudonkepuraita	Drama	Marvel studios	Jennifer Lopez	Connor Mcgregor	1999	Arvydas Lempa	1000	
Naktis Muziejuje	Komędija	WB production	Robinas Williamsas	Nicolas Cage	1999	Arvydas Lempa	200	
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Connor Mcgregor	1999	Arvydas Lempa	800	
Žuviukas Nemo	Animacinis	Pixar Studija	Aistė Valdžytė	Connor Mcgregor	1999	Arvydas Lempa	999	
Kurmis	Animacinis	Pixar Studija	Aistė Valdžytė	Connor Mcgregor	2000	168	2021	Nesitesi
Vinas	Benzinas							
1988								
Šilutė								
Namas	Veiksmo	Kiškio studija	Nicolas Cage	Jennifer Lopez	1999	Arvydas Lempa	50	
Ratas	Dramaturgas	Batuotas Katinas	Connor Mcgregor	Viktoras Medis	1999	Arvydas Lempa	120	
Kęlmas	Drama	Italian Cinema	Harrisonas Fordas	Morganas Freemanas	1999	Arvydas Lempa	220	
Raudonkepuraitis	Drama	Marvel studios	JNicolas Cage	Augustas Višinskas	1999	Arvydas Lempa	1000	
Naktis Paryžiuje	OoogaBooga	WB production	Robinas Williamsas	Augustas Višinskas	1999	Arvydas Lempa	200	
Ratai 6	Animacinis	Pixar Studija	Augustas Višinskas	Gabija Arnašiūtė	1999	Arvydas Lempa	800	
Žuvytė Dorė	Animacinis	Pixar Studija	Augustas Višinskas	Arūnas Beržas	1999	Arvydas Lempa	999	
Lietuva	Animacinis	Pixar Studija	Augustas Višinskas	Nicolas Cage	2000	168	2021	Nesitesi

Mėgiamiausi aktoriai:

Kino mėgėjas	Mėgstamiausias aktorius
Mahatma Gandis	Aistė Valdžytė
Kęstas Pajūris	Connor Mcgregor
Vinas Benzinas	Augustas Višinskas

Nematyti filmai ir serialai

Nėra filmų!

Nėra serialų!

Rekomendacijos

Mahatma Gandis rekomenduoja:

Nėra filmų!

Nėra serialų!

Kęstas Pajūris rekomenduoja:

Nėra filmų!

Nėra serialų!

Vinas Benzinas rekomenduoja:

Filmai							
Pavadinimas	Žanras	Kino studija	Pagr. aktorius 1	Pagr. aktorius 2	Leidimo metai	Režisierius	Pajamos
Matrica	Veiksmo	Kiškio studija	Nicolas Cage	Connor Mcgregor	1999	Arvydas Lempa	50
Geriausi mūsų metai	Drama	Italian Cinema	Connor Mcgregor	Jennifer Lopez	1999	Arvydas Lempa	220
Naktis Muziejuje	Komędija	WB production	Robinas Williamsas	Nicolas Cage	1999	Arvydas Lempa	200
Ratai 2	Animacinis	Pixar Studija	Augustas Višinskas	Connor Mcgregor	1999	Arvydas Lempa	800

Nėra serialų!

Nauji filmai ir serialai

Nėra filmų!

Nėra serialų!

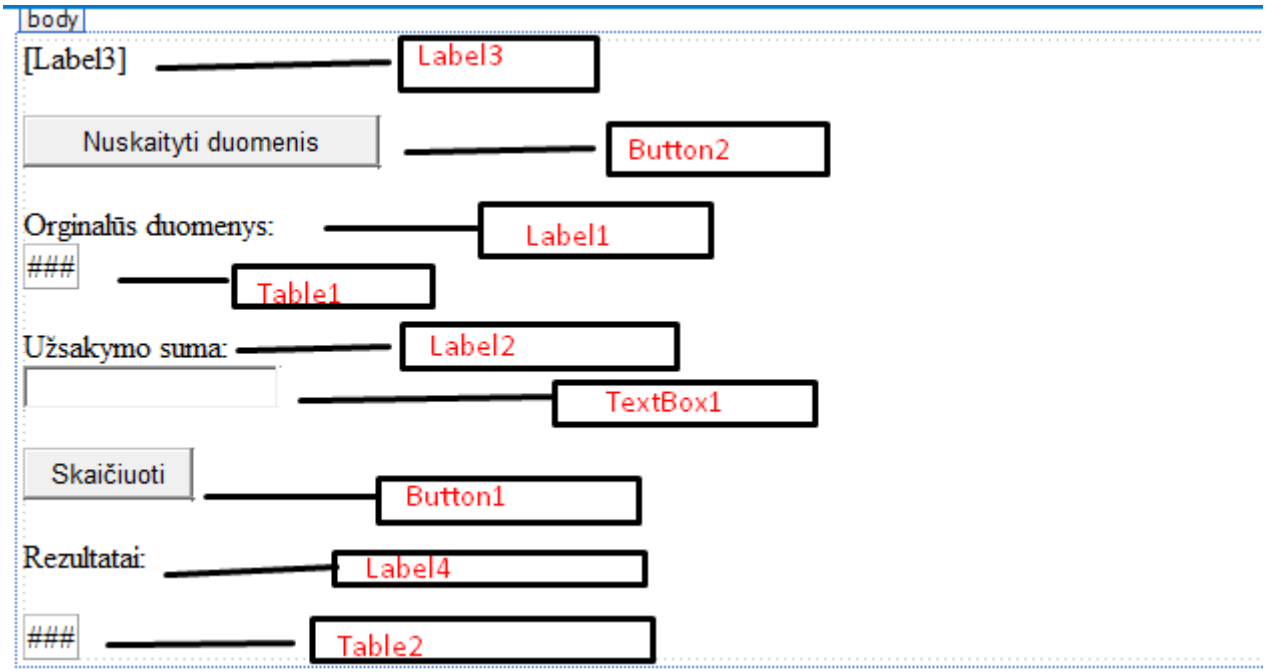
4.8. Dėstytojo pastabos

5. Deklaratyvusis programavimas (L5)

5.1. Darbo užduotis

LDD_9. **Neišpildytas užsakymas.** Pirmoje failo eilutėje nurodytas sandėlio numeris (failų daug). Faile yra duomenys apie sandėlyje esančias prekes. Eilutėje yra tokie duomenys: vardas, kiekis, kaina. Atskirame tekstiniame faile yra užsakymas: vardas, kiekis, prekės kainos riba. Prekės sandėliuose gali kartotis, bet skirtingomis kainomis. Rinkti prekę, kurios kaina artimiausia nurodytai, bet jos neviršija. Užsakymo suma yra ribota (įvedama klaviatūra). Jei užsakymo suma viršija nurodytą ribą, tai iš užsakymo šalinti brangiausias prekes po vieną, kol suma nebus viršyta. Atspausdinti neįvykdyto užsakymo prekes, surikiuotas pagal kainą ir pavadinimą, nurodant priežastį (nebuvo sandėliuose, per brangu, netilpo į užsakymą).

5.2. Grafinės vartotojo sąsajos schema



5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button2	Text	Nuskaityti duomenis
Label1	Text	Originalūs duomenys
Table1	GridLines	Both
Label2	Text	Uzsakymo suma
Button1	Text	Skaičiuoti

Label4	Text	Rezultatai:
Table2	GridLines	Both

5.4. Klasių diagrama

5.5. Programos vartotojo vadovas

Duomenų failus reikia įkelti į App_Dataa aplankalą. Informacija apie sandelius reikia kelti į Warehouses aplankalą esanti App_Dataa aplankale. Sandelio informacijos failas turėtu pirmoje eilutėje turėti sandelio numerį, o sekančiose, prekės pavadinimas, kaina, kiekis. Užsakymo duomenų failas turėtu būti įkeltas į App_Dataa/Order aplankalą. Užsakymo duomenų failas turėtu turėti tik vieną eilutę teksto, prekės pavadinimas, kiekis, kainos riba. Norint suskaičiuoti rezultatą, vartotojo sąsajoje reikia pirmiausia paspausti mygtuką „Nuskaityti duomenis“, o vėliau „Skaičiuoti“

5.6. Programos tekstas

Good.cs

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

namespace L5.App_Codee
{
    /// <summary>
    /// Class for Good object
    /// </summary>
    public class Good
    {
        /// <summary>
        /// Properties
        /// </summary>
        public string Name { get; set; }
        public int Amount { get; set; }
        public decimal PricePerUnit { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="name">name of good</param>
        /// <param name="amount">amount</param>
        /// <param name="pricePerUnit">price</param>
        public Good(string name, int amount, decimal pricePerUnit)
        {
            Name = name;
```

```
            Amount = amount;
            PricePerUnit = pricePerUnit;
        }

        /// <summary>
        /// Equals
        /// </summary>
        /// <param name="obj">other Good object</param>
        /// <returns>true or false</returns>
        public bool Equals(Good obj)
        {
            if (this.Name.Equals(obj.Name)
                && this.PricePerUnit.Equals(obj.PricePerUnit))
                return true;
            else
                return false;
        }

        /// <summary>
        /// ToString() override
        /// </summary>
        /// <returns>string</returns>
        public override string ToString()
        {
            return string.Format("| {0,-15} | {1,6} | {2,8} |", Name,
                Amount, PricePerUnit);
        }
    }
}
```

InOut.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.IO;
using System.Web.UI.WebControls;
namespace L5.App_Codee
{
    public class InOut
    {
        /// <summary>
        /// Reads all warehouses
        /// </summary>
        /// <param name="files">files of warehouses</param>
        /// <returns>warehouses</returns>
        public static List<Warehouse> ReadWarehouses(string[] files, List<Exception> errs)
        {
            List<Warehouse> warehouses = new List<Warehouse>();
            foreach (string fileName in files)
            {
                string[] lines = File.ReadAllLines(fileName);
                string nameofwarehouse = lines[0];
                List<string> name = new List<string>();
                List<int> amount = new List<int>();
                List<decimal> price = new List<decimal>();

                for (int i = 1; i < lines.Length; i++)
                {
                    string[] values = lines[i].Split(';');

                    string nameT;
                    int amountT;
                    decimal priceT;

                    try
```

```
                    {
                        nameT = values[0];
                    }
                    catch
                    {
                        errs.Add(new Exception($"{fileName} failas, {i + 1} eilutė: Blogas
pavadinimas"));
                        continue;
                    }

                    try
                    {
                        amountT = Convert.ToInt32(values[1]);
                    }
                    catch
                    {
                        errs.Add(new Exception($"{fileName} failas, {i + 1} eilutė: Blogas
kiekis"));
                        continue;
                    }

                    try
                    {
                        priceT = Convert.ToDecimal(values[2]);
                    }
                    catch
                    {
                        errs.Add(new Exception($"{fileName} failas, {i + 1} eilutė: Bloga
kaina"));
                        continue;
                    }

                    name.Add(nameT);
                    amount.Add(amountT);
                    price.Add(priceT);
                }
                warehouses.Add(new Warehouse(nameofwarehouse,
                    name, amount, price));
            }
            return warehouses;
        }

        /// <summary>
        /// Reads order
        /// </summary>
        /// <param name="fileName">name of file</param>
        /// <returns>order</returns>
        public static Order ReadOrder(string fileName)
        {
            try
            {
                string line = File.ReadAllLines(fileName)[0];
                string[] values = line.Split(';');
                string name = values[0];
                int amount = Convert.ToInt32(values[1]);
                decimal price = Convert.ToDecimal(values[2]);

                return new Order(name, amount, price);
            }
            catch
            {
                throw new NotImplementedException("Blogas uzsakymo failas");
            }
        }

        /// <summary>
        /// Outputs original data
```

```
/// </summary>
/// <param name="table">table to output</param>
/// <param name="warehouses">warehouses</param>
/// <param name="order">order</param>
public static void OutputOriginalData(Table table,
    List<Warehouse> warehouses, Order order)
{
    foreach (Warehouse warehouse in warehouses)
    {
        TableCell explanation = new TableCell();
        explanation.Text = "Sandėlio numeris:";
        TableRow row1 = new TableRow();
        row1.Cells.Add(explanation);
        table.Rows.Add(row1);
        TableRow name = new TableRow();
        TableCell wnumber = new TableCell();
        wnumber.Text = warehouse.NameOfWarehouse;
        name.Cells.Add(wnumber);
        table.Rows.Add(name);

        TableCell exp1 = new TableCell();
        TableCell exp2 = new TableCell();
        TableCell exp3 = new TableCell();
        exp1.Text = "Prekės pavadinimas";
        exp2.Text = "Kiekis";
        exp3.Text = "Kaina";
        TableRow exp = new TableRow();
        exp.Cells.Add(exp1);
        exp.Cells.Add(exp2);
        exp.Cells.Add(exp3);
        table.Rows.Add(exp);
        for (int i = 0; i < warehouse.GetNameList().
            Count(); i++)
        {
            TableCell goodname = new TableCell();
            TableCell amount = new TableCell();
            TableCell price = new TableCell();
            goodname.Text = warehouse.GetNameList()[i];
            amount.Text = warehouse.GetAmountList()[i].ToString();
            price.Text = warehouse.GetPriceList()[i].ToString();
            TableRow row = new TableRow();
            row.Cells.Add(goodname);
            row.Cells.Add(amount);
            row.Cells.Add(price);
            table.Rows.Add(row);
        }
    }
    TableCell orderr = new TableCell();
    orderr.Text = "Užsakymas:";
    TableRow row2 = new TableRow();
    row2.Cells.Add(orderr);
    table.Rows.Add(row2);

    TableCell e1 = new TableCell();
    TableCell e2 = new TableCell();
    TableCell e3 = new TableCell();
    e1.Text = "Prekės pavadinimas";
    e2.Text = "Kiekis";
    e3.Text = "Kaina";
    TableRow ep = new TableRow();
    ep.Cells.Add(e1);
    ep.Cells.Add(e2);
    ep.Cells.Add(e3);
    table.Rows.Add(ep);

    TableCell ordername = new TableCell();
    TableCell orderamount = new TableCell();
```

```
        TableCell priceperunit = new TableCell();
        ordername.Text = order.NameOfGood;
        orderamount.Text = order.Amount.ToString();
        priceperunit.Text = order.PricePerUnit.ToString();
        TableRow orderinfo = new TableRow();
        orderinfo.Cells.Add(ordername);
        orderinfo.Cells.Add(orderamount);
        orderinfo.Cells.Add(priceperunit);
        table.Rows.Add(orderinfo);
    }

    /// <summary>
    /// Outputs original data
    /// </summary>
    /// <param name="fileName">name of file</param>
    /// <param name="warehouses">warehouses</param>
    /// <param name="order">order</param>
    public static void OutputOriginalData(string fileName,
        List<Warehouse> warehouses, Order order)
    {
        List<string> Data = new List<string>();
        string format = "| {0,-15} | {1, -6} | {2, -8} |";
        foreach (Warehouse warehouse in warehouses)
        {
            Data.Add("Sandelio numeris:" +warehouse.ToString());
            Data.Add(new string('-', 39));
            if (warehouse.GetAmountList().Count > 0)
            {
                Data.Add(string.Format(format, "Pavadinimas",
                    "Kiekis", "Kaina"));
                Data.Add(new string('-', 39));
            }
            else
            {
                Data.Add("Nėra prekių sandelyje");
                Data.Add(new string('-', 39));
            }

            for (int i = 0; i < warehouse.GetNameList().Count(); i++)
            {
                Data.Add(string.Format("| {0,-15} | {1, 6} | {2, 8} |",
                    warehouse.GetNameList()[i],
                    warehouse.GetAmountList()[i],
                    warehouse.GetPriceList()[i]));
                Data.Add(new string('-', 39));
            }
        }
        Data.Add("Užsakymas:");
        Data.Add(new string('-', 39));
        Data.Add(string.Format(format, "Pavadinimas",
            "Kiekis", "Kaina"));
        Data.Add(new string('-', 39));
        Data.Add(order.ToString());
        Data.Add(new string('-', 39));
        File.WriteAllLines(fileName, Data);
    }

    /// <summary>
    /// Outputs rezults
    /// </summary>
    /// <param name="table">table to output</param>
    /// <param name="explanations">explanations</param>
    /// <param name="goods">goods</param>
    public static void OutputRezults(Table table,
        List<string> explanations, List<Good> goods)
    {
        if (!goods.Any())
```

```
            {
                TableRow row = new TableRow();
                TableCell cell = new TableCell();
                cell.Text = "Uzsakymas pavyko";
                row.Cells.Add(cell);
                table.Rows.Add(row);
                return;
            }

            TableRow exp = new TableRow();
            TableCell exp1 = new TableCell();
            TableCell exp2 = new TableCell();
            TableCell exp3 = new TableCell();
            TableCell exp4 = new TableCell();
            exp1.Text = "Prekė";
            exp2.Text = "Kiekis";
            exp3.Text = "Kaina";
            exp4.Text = "Paaiškinimas";
            exp.Cells.Add(exp1);
            exp.Cells.Add(exp2);
            exp.Cells.Add(exp3);
            exp.Cells.Add(exp4);
            table.Rows.Add(exp);
            for (int i = 0; i < goods.Count; i++)
            {
                TableRow row = new TableRow();
                TableCell name = new TableCell();
                TableCell amount = new TableCell();
                TableCell price = new TableCell();
                TableCell explanation = new TableCell();
                name.Text = goods[i].Name;
                amount.Text = goods[i].Amount.ToString();
                price.Text = goods[i].PricePerUnit.ToString();
                explanation.Text = explanations[i];
                row.Cells.Add(name);
                row.Cells.Add(amount);
                row.Cells.Add(price);
                row.Cells.Add(explanation);
                table.Rows.Add(row);
            }
        }
    }
}
```

Warehouse.cs

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

namespace L5.App_Codee
{
    /// <summary>
    /// Class for Warehouse object
    /// </summary>
    public class Warehouse
    {
        /// <summary>
        /// Properties
        /// </summary>
        public string NameOfWarehouse { get; set; }
        private List<string> GoodName;
        private List<int> Amount;
        private List<decimal> Price;

        /// <summary>
        /// Constructor
```

```
        /// </summary>
        /// <param name="nameOfWarehouse">name of warehouse</param>
        /// <param name="goodName">names of goods</param>
        /// <param name="amount">amount of goods</param>
        /// <param name="price">price of goods</param>
        public Warehouse(string nameOfWarehouse,
            List<string> goodName, List<int> amount, List<decimal> price)
        {
            NameOfWarehouse = nameOfWarehouse;
            GoodName = goodName;
            Amount = amount;
            Price = price;
        }

        /// <summary>
        /// Gets List of goods' names
        /// </summary>
        /// <returns>names</returns>
        public List<string> GetNameList()
        {
            return GoodName;
        }

        /// <summary>
        /// Gets List of goods' prices
        /// </summary>
        /// <returns>prices</returns>
        public List<decimal> GetPriceList()
        {
            return Price;
        }

        /// <summary>
        /// Gets goods' amount
        /// </summary>
        /// <returns>amounts</returns>
        public List<int> GetAmountList()
        {
            return Amount;
        }

        /// <summary>
        /// ToString() override
        /// </summary>
        /// <returns>string</returns>
        public override string ToString()
        {
            return NameOfWarehouse;
        }
    }
}
```

Order.cs

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

namespace L5.App_Codee
{
    /// <summary>
    /// Class for Order object
    /// </summary>
    public class Order
    {
        /// <summary>
```

```
        /// Properties
        /// </summary>
        public string NameOfGood { get; set; }
        public int Amount { get; set; }
        public decimal PricePerUnit { get; set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="nameOfGood">Name of good</param>
        /// <param name="amount">amount</param>
        /// <param name="pricePerUnit">price per one unit</param>
        public Order(string nameOfGood, int amount, decimal pricePerUnit)
        {
            NameOfGood = nameOfGood;
            Amount = amount;
            PricePerUnit = pricePerUnit;
        }

        /// <summary>
        /// ToString() override
        /// </summary>
        /// <returns>string</returns>
        public override string ToString()
        {
            return string.Format("| {0,15} | {1, -6} | {2, -8} |", NameOfGood,
                Amount,PricePerUnit);
        }
    }
}
```

TaskUtils.cs

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

namespace L5.App_Codee
{
    public class TaskUtils
    {
        /// <summary>
        /// Gets all the goods that are ordered
        /// </summary>
        /// <param name="order">order</param>
        /// <param name="warehouses">warehouses</param>
        /// <returns>Goods</returns>
        public static List<Good> GetSelectedGoods(Order order,
            List<Warehouse> warehouses)
        {
            List<Good> goods = new List<Good>();
            warehouses.ForEach(x => { for (int i = 0;
                i < x.GetNameList().Count(); i++)
                { if (x.GetNameList()[i].Equals(order.NameOfGood))
                    { goods.Add(new Good(order.NameOfGood,
                        x.GetAmountList()[i], x.GetPriceList()[i]));
                    } } });
            return goods;
        }

        /// <summary>
        /// Sorts goods
        /// </summary>
        /// <param name="goods">goods</param>
        /// <returns>sorted goods</returns>
        public static List<Good> Sort(List<Good> goods)
```

```
{
    goods = goods.OrderByDescending(p => p.PricePerUnit)
        .ThenByDescending(n=> n.Name).ToList();
    return goods;
}

/// <summary>
/// Completes order
/// </summary>
/// <param name="goods">goods</param>
/// <param name="order">order</param>
/// <param name="orderedamount">ordered amount</param>
/// <returns>completed order</returns>
public static List<Good> CompleteOrder(ref List<Good> goods,
    Order order, out int orderedamount)
{
    orderedamount = 0;
    List<Good> ordered = new List<Good>();

    for (int i = 0; i < goods.Count; i++)
    {
        if (goods[i].Amount > 0)
        {
            if (goods[i].Amount < order.Amount
                - orderedamount && goods[i].PricePerUnit
                <order.PricePerUnit)
            {
                ordered.Add(new Good(order.NameOfGood,
                    goods[i].Amount, goods[i].PricePerUnit));
                orderedamount += goods[i].Amount;
                goods.RemoveAt(i);
                i--;
                goods = Sort(goods);
            }
            else if (goods[i].PricePerUnit < order.
                PricePerUnit)
            {
                ordered.Add(new Good(order.NameOfGood,
                    order.Amount - orderedamount,
                    goods[i].PricePerUnit));
                goods[i].Amount -= order.Amount - orderedamount;
                if (goods[i].Amount == 0)
                {
                    goods.RemoveAt(i);
                    i--;
                }

                orderedamount = order.Amount;
                return ordered;
            }
        }
    }
    return ordered;

}

/// <summary>
/// Gets price of the order
/// </summary>
/// <param name="order">order</param>
/// <returns>price of order</returns>
public static decimal GetOrderPrice(List<Good> order)
{
    return order.Sum(x => x.PricePerUnit * x.Amount);
}

/// <summary>
```

```
        /// Trims the order to specified price
        /// </summary>
        /// <param name="goods">goods</param>
        /// <param name="maxprice">maxprice</param>
        /// <param name="ordered">ordered</param>
        /// <returns>trimmed order</returns>
        public static List<Good> TrimOrder(ref List<Good> goods,
            decimal maxprice, List<Good> ordered)
        {
            while (GetOrderPrice(ordered) > maxprice)
            {
                ordered = Sort(ordered);
                if (ordered[0].Amount > 0)
                {
                    ordered[0].Amount -= 1;
                    if (goods.Contains(ordered[0]))
                    {
                        foreach (Good good in goods)
                        {
                            if (good.Equals(ordered[0]))
                            {
                                good.Amount = 1;
                                break;
                            }
                        }
                    }
                    else
                        goods.Add(new Good(ordered[0].Name, 1,
                            ordered[0].PricePerUnit));
                }
                else
                {
                    ordered.Remove(ordered[0]);
                }
            }
            return ordered;
        }

        /// <summary>
        /// Adds explanations
        /// </summary>
        /// <param name="goods">goods</param>
        /// <param name="orderedamount">ordered amount</param>
        /// <param name="order">order</param>
        /// <param name="maxsum">maximal sum allowed</param>
        /// <param name="ordersum">order sum</param>
        /// <returns>Explanation</returns>
        public static List<string> AddExplanation(List<Good> goods,
            int orderedamount, Order order, decimal maxsum,
            decimal ordersum)
        {
            List<string> Data = new List<string>();
            goods.ForEach(x => { if (x.Amount == 0)
                { Data.Add("Nebuvo sandeliuose"); }
                else if (x.PricePerUnit > order.PricePerUnit)
                { Data.Add("Per brangu"); }
                else if (order.Amount == orderedamount)
                { Data.Add("Netilpo į užsakymą"); }
                else { Data.Add("Kita klaida"); }
            });
            return Data;
        }
    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs" Inherits="L5.Forma1" %>

<!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="Label3" runat="server"></asp:Label>
            <br />
            <asp:Panel ID="Panel1" runat="server"></asp:Panel>
            <br />
            <asp:Button ID="Button3" runat="server" OnClick="Button3_Click" Text="Nuskaityti
duomenis" />
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Orginalūs duomenys:"></asp:Label>
            <asp:Table ID="Table1" runat="server" GridLines="Both">
            </asp:Table>
            <br />
            <asp:Label ID="Label2" runat="server" Text="Užsakymo suma:"></asp:Label>
            <br />
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Skaičiuoti" />
            <br />
            <br />
            <asp:Label ID="Label4" runat="server" Text="Rezultatai:"></asp:Label>
            <br />
            <br />
            <asp:Table ID="Table2" runat="server" GridLines="Both">
            </asp:Table>
        </div>
    </form>
</body>
</html>
```

Forma1.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Text.RegularExpressions;
namespace L5
{
    public partial class Forma1 : System.Web.UI.Page
    {
        const string CRZ = "Original_Data.txt";
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            try
```

```
        {
            if (Session["warehouses"] != null && Session["order"] != null)
            {
                List<App_Codee.Warehouse> warehouses =
                    (List<App_Codee.Warehouse>)Session["warehouses"];
                App_Codee.Order order =
                    (App_Codee.Order)Session["order"];
                decimal maxprice;
                if (!Regex.Match(TextBox1.Text, @"\D+").Success)
                {
                    maxprice = Convert.ToDecimal(TextBox1.Text);
                }
                else
                    throw new NotImplementedException();
                List<App_Codee.Good> goods = App_Codee.TaskUtils.
                    GetSelectedGoods(order, warehouses);
                goods = App_Codee.TaskUtils.Sort(goods);
                int orderedamount;
                List<App_Codee.Good> processed = App_Codee.TaskUtils.
                    CompleteOrder(ref goods, order, out orderedamount);

                if (App_Codee.TaskUtils.GetOrderPrice
                    (processed) > maxprice)
                {
                    processed = App_Codee.TaskUtils.TrimOrder
                        (ref goods, maxprice, processed);
                }
                processed = App_Codee.TaskUtils.Sort(processed);
                List<string> explanation = App_Codee.TaskUtils.
                    AddExplanation(goods, orderedamount, order, maxprice,
                    App_Codee.TaskUtils.GetOrderPrice(processed));
                App_Codee.InOut.OutputOriginalData
                (Table1, warehouses, order);
                App_Codee.InOut.OutputRezults
                    (Table2, explanation, goods);
            }
        }
        catch (Exception err)
        {
            OutputError(err);
        }
    }

    protected void Button3_Click(object sender, EventArgs e)
    {
        try
        {
            List<Exception> errs = new List<Exception>();

            string[] warehousefiles = Directory.GetFiles
                (Server.MapPath("App_Dataa/Warehouses"));
            string orderfile = Directory.GetFiles
                (Server.MapPath("App_Dataa/Order"))[0];
            List<App_Codee.Warehouse> warehouses = App_Codee.InOut.
                ReadWarehouses(warehousefiles, errs);

            App_Codee.Order order = App_Codee.InOut.ReadOrder(orderfile);

            foreach (Exception err in errs)
            {
                OutputError(err);
            }

            App_Codee.InOut.OutputOriginalData
                (Table1, warehouses, order);
            App_Codee.InOut.OutputOriginalData(Server.MapPath(CRZ),
                warehouses, order);
```

```
                Session["warehouses"] = warehouses;
                Session["order"] = order;
            }
            catch (Exception err)
            {
                OutputError(err);
            }
        }

        protected void OutputError(Exception ex)
        {
            Label label = new Label();
            label.Text = ex.Message;
            Panel1.Controls.Add(label);
        }
    }
}
```

5.7. Pradiniai duomenys ir rezultatai

Testas 1 – visi duomenys yra duoti

Duomenys:
Data1.txt

```
1
Knyga;14;3,45
Telefonas;4;397,98
Mašina;1;3500
Pelytė;3;17,99
```

Data2.txt

```
2
Knyga;0;3,45
Telefonas;0;397,98
Mašina;3;3600
Pelytė;0;17,99
```

Data3.txt

```
3
Knyga;3;4,50
Telefonas;0;400
```

Data4.txt

```
4
Knyga;8;3,45
Telefonas;2;250,15
Mašina;2;5000
Pelytė;1;19,99
```

Order.txt

```
1   Mašina;3;4000;
```

Rezultatai:
Original_Data.txt

Original_Data - Notepad

File Edit Format View Help

```
Sandelio numeris:1
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |     14 |      345 |
---------------------------------------
| Telefonas        |      4 |    39798 |
---------------------------------------
| Mašina           |      1 |     3500 |
---------------------------------------
| Pelytė           |      3 |     1799 |
---------------------------------------
Sandelio numeris:2
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |      0 |      345 |
---------------------------------------
| Telefonas        |      0 |    39798 |
---------------------------------------
| Mašina           |      3 |     3600 |
---------------------------------------
| Pelytė           |      0 |     1799 |
---------------------------------------
Sandelio numeris:3
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |      3 |      450 |
---------------------------------------
| Telefonas        |      0 |      400 |
---------------------------------------
Sandelio numeris:4
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |      8 |      345 |
---------------------------------------
| Telefonas        |      2 |    25015 |
---------------------------------------
| Mašina           |      2 |     5000 |
---------------------------------------
| Pelytė           |      1 |     1999 |
---------------------------------------
Užsakymas:
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
|           Mašina | 3      | 4000     |
---------------------------------------
```

Vartotojo sąsaja:

localhost

Nuskaityti duomenis

Orginalūs duomenys:

Sandėlio numeris:		
1		
Prekės pavadinimas	Kiekis	Kaina
Knyga	14	345
Telefonas	4	39798
Mašina	1	3500
Pelytė	3	1799
Sandėlio numeris:		
2		
Prekės pavadinimas	Kiekis	Kaina
Knyga	0	345
Telefonas	0	39798
Mašina	3	3600
Pelytė	0	1799
Sandėlio numeris:		
3		
Prekės pavadinimas	Kiekis	Kaina
Knyga	3	450
Telefonas	0	400
Sandėlio numeris:		
4		
Prekės pavadinimas	Kiekis	Kaina
Knyga	8	345
Telefonas	2	25015
Mašina	2	5000
Pelytė	1	1999
Užsakymas:		
Prekės pavadinimas	Kiekis	Kaina
Mašina	3	4000

Užsakymo suma:

900000 ×

Skaičiuoti

Rezultatai:

Prekė	Kiekis	Kaina	Paaiškinimas
Mašina	2	5000	Per brangu
Mašina	1	3500	Netilpo į užsakymą

Testas 2 – duomenų failai yra tušti

Rezultatai:
Vartotojo sąsaja:

Duomenys pateikti blogu formatu arba jų nėra

Nuskaityti duomenis

Orginalūs duomenys:

Užsakymo suma:

Skaičiuoti

Rezultatai:

Testas 3 – sandėlyje nėra nei vienos likusios norimos prekės

Duomenys:

Data1.txt

```
1
Knyga;14;3,45
Telefonas;4;397,98
Mašina;0;3500
Pelytė;3;17,99
```

Data2.txt

```
2
Knyga;0;3,45
Telefonas;0;397,98
Mašina;0;3600
Pelytė;0;17,99
```

Data3.txt

```
3
Knyga;3;4,50
Telefonas;0;400
```

Data4.txt

```
4
Knyga;8;3,45
Telefonas;2;250,15
Mašina;0;5000
Pelytė;1;19,99
```

Order.txt

```
Mašina;3;4000;
```

Rezultatai:

Original_Data.txt

Original_Data - Notepad

File Edit Format View Help

```
Sandelio numeris:1
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |     14 |      345 |
---------------------------------------
| Telefonas        |      4 |    39798 |
---------------------------------------
| Mašina           |      0 |     3500 |
---------------------------------------
| Pelytė           |      3 |     1799 |
---------------------------------------
Sandelio numeris:2
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |      0 |      345 |
---------------------------------------
| Telefonas        |      0 |    39798 |
---------------------------------------
| Mašina           |      0 |     3600 |
---------------------------------------
| Pelytė           |      0 |     1799 |
---------------------------------------
Sandelio numeris:3
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |      3 |      450 |
---------------------------------------
| Telefonas        |      0 |      400 |
---------------------------------------
Sandelio numeris:4
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
| Knyga            |      8 |      345 |
---------------------------------------
| Telefonas        |      2 |    25015 |
---------------------------------------
| Mašina           |      0 |     5000 |
---------------------------------------
| Pelytė           |      1 |     1999 |
---------------------------------------
Užsakymas:
---------------------------------------
| Pavadinimas      | Kiekis | Kaina    |
---------------------------------------
|           Mašina | 3      | 5500     |
---------------------------------------
```

Vartotojo sąsaja:

Nuskaityti duomenis

Orginalūs duomenys:

Sandėlio numeris:		
1		
Prekės pavadinimas	Kiekis	Kaina
Knyga	14	345
Telefonas	4	39798
Mašina	0	3500
Pelytė	3	1799
Sandėlio numeris:		
2		
Prekės pavadinimas	Kiekis	Kaina
Knyga	0	345
Telefonas	0	39798
Mašina	0	3600
Pelytė	0	1799
Sandėlio numeris:		
3		
Prekės pavadinimas	Kiekis	Kaina
Knyga	3	450
Telefonas	0	400
Sandėlio numeris:		
4		
Prekės pavadinimas	Kiekis	Kaina
Knyga	8	345
Telefonas	2	25015
Mašina	0	5000
Pelytė	1	1999
Užsakymas:		
Prekės pavadinimas	Kiekis	Kaina
Mašina	3	5500

Užsakymo suma:

9000

Skaičiuoti

Rezultatai:

Prekė	Kiekis	Kaina	Paaiškinimas
Mašina	0	5000	Nebuvo sandeliuose
Mašina	0	3600	Nebuvo sandeliuose
Mašina	0	3500	Nebuvo sandeliuose

5.8. Dėstytojo pastabos