Kauno technologijos universitetas

Informatikos fakultetas

Objektinis programavimas 2 (P175B123)

Laboratorinių darbų ataskaita

<Lukas Kuzmickas> <IFF 6>

Studentas

Lekt. Mindaugas Jančiukas

Dėstytojas

Kaunas 2022

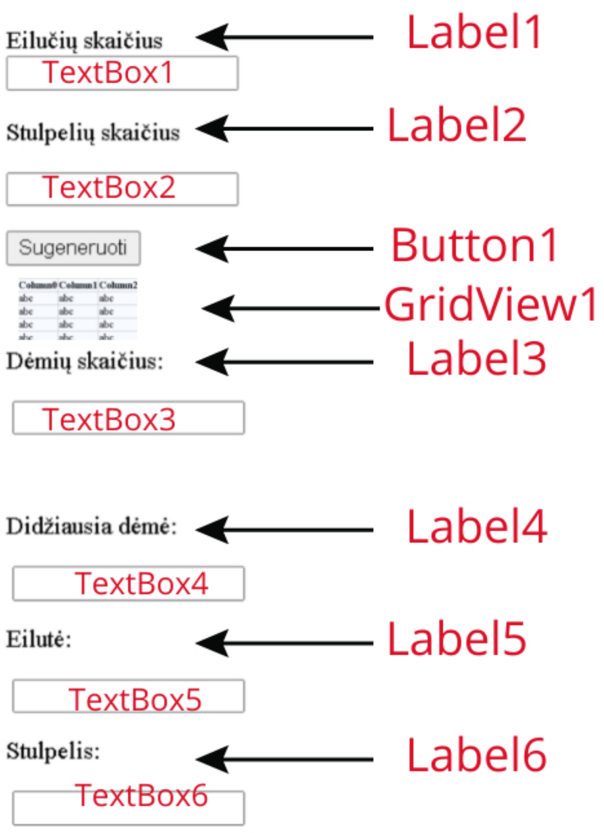
TURINYS

1. Rekursija (L1)

1.1. Darbo užduotis

LD_19. Dėmė. Turime popieriaus lapą langeliais. Langelių eilutės ir stulpeliai sunumeruoti, pradedant vienetu. Numeracijos pradžia yra kairysis viršutinis langelis. Vaikas padažė teptuką dažuose ir krestelėjo virš lapo. Suskaičiuokite, kiek dėmių lape ir iš kiek langelių sudaryta didžiausia dėmė. Langeliai arba tušti, arba dažyti. Pusiau dažytų nėra. Tai pačiai dėmei priklauso visi dažyti langeliai, kurių bent viena koordinatė skiriasi vienetu nuo gretimo dažyto langelio. Reikia parašyti dėmių generatorių, kuris NxM lape nudažytų atsitiktinai trečdalį langelių, kai 1 N 20 ir 1 M 70. Čia N – eilučių skaičius, o M –stulpelių skaičius. N ir M reikšmės įvedamos klaviatūra. Rezultatuose parodykite dėmėmis išmargintą lapą. Tuščius langelius žymėkite tašku, dažytus langelius žymėkite žvaigždute * arba spalva. Apačioje parašykite, kiek yra dėmių, kiek langelių sudaro didžiausią dėmę ir didžiausios dėmės vieno bet kurio langelio koordinates: eilutės numerį ir stulpelio numerį.

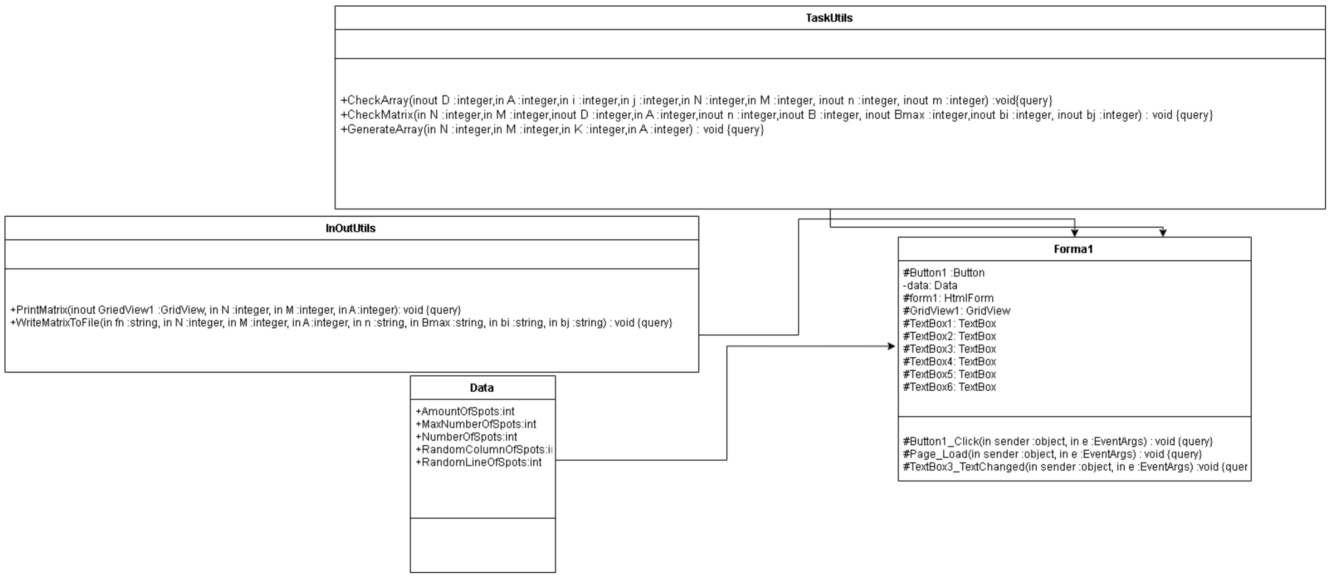
1.2. Grafinės vartotojo sąsajos schema



1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button1	Text	„Sugeneruoti“
TextBox1	Niekas nepakeista	
TextBox2	Niekas nepakeista	
TextBox3	Niekas nepakeista	
TextBox4	Niekas nepakeista	
TextBox5	Niekas nepakeista	
TextBox6	Niekas nepakeista	
Label1	Text	„Eilučių skaičius“
Label2	Text	„Stulpelių skaičius“
Label3	Text	„Dėmių skaičius:“
Label4	Text	„Didžiausia dėmė:“
Label5	Text	„Eilutė:“
Label6	Text	„Stulpelis:“
GridView1	Niekas nepakeista	

1.4. Klasių diagrama


TaskUtils
+CheckArray(inout D :integer,in A :integer,in i :integer,in j :integer,in N :integer,in M :integer, inout n :integer, inout m :integer) :void{query}
+CheckMatrix(in N :integer,in M :integer,inout D :integer,in A :integer,inout n :integer,inout B :integer, inout Bmax :integer,inout bi :integer, inout bj :integer) : void {query}
+GenerateArray(in N :integer,in M :integer,in K :integer,in A :integer) : void {query}
InOutUtils
+PrintMatrix(inout GriedView1 :GridView, in N :integer, in M :integer, in A :integer): void {query}
+WriteMatrixToFile(in fn :string, in N :integer, in M :integer, in A :integer, in n :string, in Bmax :string, in bi :string, in bj :string) : void {query}
Data
+AmountOfSpots:int
+MaxNumberOfSpots:int
+NumberOfSpots:int
+RandomColumnOfSpots:i
+RandomLineOfSpots:int
Forma1
#Button1 :Button
-data: Data
#form1: HtmlForm
#GridView1: GridView
#TextBox1: TextBox
#TextBox2: TextBox
#TextBox3: TextBox
#TextBox4: TextBox
#TextBox5: TextBox
#TextBox6: TextBox
#Button1_Click(in sender :object, in e :EventArgs) : void {query}
#Page_Load(in sender :object, in e :EventArgs) : void {query}
#TextBox3_TextChanged(in sender :object, in e :EventArgs) :void {quer

1.5. Programos vartotojo vadovas

Įjungus programą, matomi du tekstai duomenų įvedimui – lapo eilučių ir stulpelių skaičius. Įvedus šiuos duomenis ir nuspaudus mygtuką ,,Sugeneruoti‘‘, sugeneruojama atsitiktinis dvimatis masyvas su pasirinktais matmenimis, šis dvimatis masyvas sudarytas iš taškų ir dėmių (žvaigždučių).

Apačioje rodomas bendras dėmių skaičius, didžiausios dėmės bendrų dėmių suma, didžiausios dėmės atsitiktinis eilutės ir stulpelio numeris.

1.6. Programos tekstas

Data.cs

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

namespace L1
{
    /// <summary>
    /// Data class for important two-dimensional array data
    /// </summary>
    public class Data
    {
        public int NumberOfSpots;
        public int AmountOfSpots = 0;
        public int MaxNumberOfSpots = 0;
        public int RandomLineOfSpots = 0;
        public int RandomColumnOfSpots = 0;
    }
}
```

InOut.cs

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

namespace L1
{
    /// <summary>
    /// Class for reading and printing information
    /// </summary>
    public static class InOut
    {
        /// <summary>
        /// Method for printing a random matrix to GridView1 element
        /// </summary>
        /// <param name="GridView1">GridView element or a table</param>
        /// <param name="N">The number of lines</param>
        /// <param name="M">The number of columns</param>
        /// <param name="A">Random generated two-dimensional array</param>
        public static void PrintMatrix(ref
global::System.Web.UI.WebControls.GridView GridView1, int N, int M, int[,] A)
        {
            DataTable dt = new DataTable();
            DataRow dr = null;
            string s = "";
            string t00 = "Eil. nr.";
            dt.Columns.Add(new DataColumn(t00, typeof(string)));
            for (int j = 1; j <= M; j++)
            {
                dt.Columns.Add(new DataColumn(j.ToString(), typeof(string)));
            }
            for (int i = 1; i <= N; i++)
            {
                dr = dt.NewRow();
                dr[t00] = i;
                for (int j = 1; j <= M; j++)
                {
                    if (A[i - 1, j - 1] == 1) s = "*"; else s = ".";
                    dr[j.ToString()] = s;
                }
                dt.Rows.Add(dr);
            }
            GridView1.DataSource = dt;
            GridView1.DataBind();
        }


        /// <summary>
        /// Method for printing data to a result file
        /// </summary>
        /// <param name="fn">Result file</param>
        /// <param name="N">Line number</param>
        /// <param name="M">Column number</param>
        /// <param name="A">Random generated matrix</param>
        /// <param name="n">Number of spots</param>
        /// <param name="Bmax">Spot size</param>
        /// <param name="bi">Random line of max spot size</param>
        /// <param name="bj">Random column of max spot size</param>
        public static void WriteMatrixToFile(string fn, int N, int M, int[,] A,
string n, string Bmax, string bi, string bj)
        {
            using (StreamWriter sw = File.CreateText(fn))
            {
                sw.WriteLine("Pradiniai duomenys");
                sw.WriteLine("Eilutės skaičius {0}", N);
                sw.WriteLine("Stulpelių skaičius {0}", M);
                sw.WriteLine("");
                sw.WriteLine("Rezultatai:");
                sw.WriteLine("Dėmių skaičius {0}", n);
                sw.WriteLine("Didžiausia dėmė {0}", Bmax);
                sw.WriteLine("Didžiausios dėmės eilutė {0}", bi);
                sw.WriteLine("Didžiausios dėmės stulpelis {0}", bj);
                sw.WriteLine("");
                sw.WriteLine("Sugeneruota matrica:");
                string s = "";
                sw.Write("\t");
                for (int j = 0; j < M; j++)
                    sw.Write("{0}\t", j + 1);
                sw.WriteLine();
                for (int i = 0; i < N; i++)
                {
                    sw.Write("{0}\t", i + 1);
                    for (int j = 0; j < M; j++)
                    {
                        if (A[i, j] == 1) s = "*"; else s = ".";
                        sw.Write("{0}\t", s);
                    }
                    sw.WriteLine("\n");
                }

            }
        }
    }
```

```
}
```

TaskUtils.cs

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

namespace L1
{
    /// <summary>
    /// Class for other individual data operations
    /// </summary>
    public static class TaskUtils
    {
        /// <summary>
        /// Method for finding spots in a two-dimensional array
        /// </summary>
        /// <param name="D">Two-dimensional array for comparison</param>
        /// <param name="A">Random generated two-dimensional array</param>
        /// <param name="i">Comparing lines</param>
        /// <param name="j">Comparing columns</param>
        /// <param name="N">Number of lines</param>
        /// <param name="M">Number of columns</param>
        /// <param name="n">Number of spots</param>
        /// <param name="m">Number of spot size</param>
        public static void CheckArray(ref int[,] D, int[,] A, int i, int j, int N,
int M, ref int n, ref int m)
        {
            if ((D[i, j] == 0) && (A[i, j] == 1))
            {
                D[i, j] = n;
            }
            for (int ki = -1; ki <= 1; ki++)
                for (int kj = -1; kj <= 1; kj++)
                    if ((i + ki >= 0) && (i + ki < N) && (j + kj >= 0) && (j + kj
< M) && ((ki != 0) || (kj != 0)))
                    {
                        if ((D[i + ki, j + kj] == 0) && (A[i + ki, j + kj] == 1))
                        {
                            CheckArray(ref D, A, i + ki, j + kj, N, M, ref n, ref
m);
                        }
                    }
            m++;
        }
        /// <summary>
        /// Method for generating a random two-dimensional array
        /// </summary>
        /// <param name="N">The number of lines in a matrix</param>
        /// <param name="M">The number of columns in a matrix</param>
        /// <param name="K">The number of spots</param>
        /// <param name="A">Random generated matrix</param>
        public static void GenerateArray(int N, int M, int K, int[,] A)
        {
            Random rand = new Random();
            int r;
            int ri;
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < M; j++)
                {
                    r = rand.Next(N * M);
                    if (r < K) ri = 1; else ri = 0;
                    A[i, j] = ri;
                }
            }
        }
        /// <summary>
        /// Method for scanning a two-dimensional array for information
        /// </summary>
        /// <param name="N">The number of lines</param>
        /// <param name="M">The number of columns</param>
        /// <param name="D">Two-dimensional array with data</param>
        /// <param name="A">Random generated two-dimensional</param>
        /// <param name="n">Number of spots</param>
        /// <param name="B">Two-dimensional for comparison</param>
        /// <param name="Bmax">Spot size</param>
        /// <param name="bi">Random line of the largest stain</param>
        /// <param name="bj">Random column of the largest stain</param>
        public static void CheckMatrix(int N, int M, ref int[,] D, int[,] A, ref
int n, ref int[] B, ref int Bmax, ref int bi, ref int bj)
        {
            for (int i = 0; i < N; i++)
                for (int j = 0; j < M; j++)
                {
                    if ((D[i, j] == 0) && (A[i, j] == 1))
                    {
                        n++;
                        CheckArray(ref D, A, i, j, N, M, ref n, ref B[n - 1]);
                        if (B[n - 1] > Bmax)
                        {
                            bi = i; bj = j;
                            Bmax = B[n - 1];
                        }
                    }
                }
        }
    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="L1.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br />
            <br />
            Eilučių skaičius</div>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        Stulpelių skaičius<p>
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        </p>
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="Sugeneruoti" />
        <br />
        <br />
        <asp:GridView ID="GridView1" runat="server" Height="99px" style="margin-
right: 0px" Width="175px">
        </asp:GridView>
        <br />
        <p>
            Dėmių skaičius:</p>
        <p>
            &nbsp;<asp:TextBox ID="TextBox3" runat="server"
OnTextChanged="TextBox3_TextChanged"></asp:TextBox>
        </p>
        <p>
            &nbsp;</p>
        <p>
            Didžiausia dėmė:&nbsp;</p>
        <p>
            &nbsp;<asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
        </p>
        <p>
            Eilutė:</p>
        <p>
            &nbsp;<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
        </p>
        <p>
            Stulpelis:</p>
        <p>
            &nbsp;<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
        </p>
    </form>
</body>
</html>
```

Forma1.aspx.cs

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

namespace L1
{
    public partial class Forma1 : System.Web.UI.Page
    {
        Data data = new Data();
        string fn = "D:/rez.txt";
        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            int N = int.Parse(TextBox1.Text);
            int M = int.Parse(TextBox2.Text);
            int K = data.NumberOfSpots;
            int n = data.AmountOfSpots;
            int Bmax = data.MaxNumberOfSpots;
            int bi = data.RandomLineOfSpots;
            int bj = data.NumberOfSpots;
            int[,] A = new int[N, M];
            int[,] D = new int[N, M];
            K = (N * M) / 3;
            int[] B = new int[K];
            TaskUtils.GenerateArray(N, M, K, A);
            TaskUtils.CheckMatrix(N, M, ref D, A, ref n, ref B, ref Bmax, ref bi, 
ref bj);
            InOut.PrintMatrix(ref GridView1, N, M, A);
            TextBox3.Text = n.ToString();
            TextBox4.Text = Bmax.ToString();
            TextBox5.Text = (bi + 1).ToString();
            TextBox6.Text = (bj + 1).ToString();
            InOut.WriteMatrixToFile(fn, N, M, A, TextBox3.Text, TextBox4.Text, 
TextBox5.Text, TextBox6.Text);
        }

        protected void TextBox3_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

1.7. Pradiniai duomenys ir rezultatai

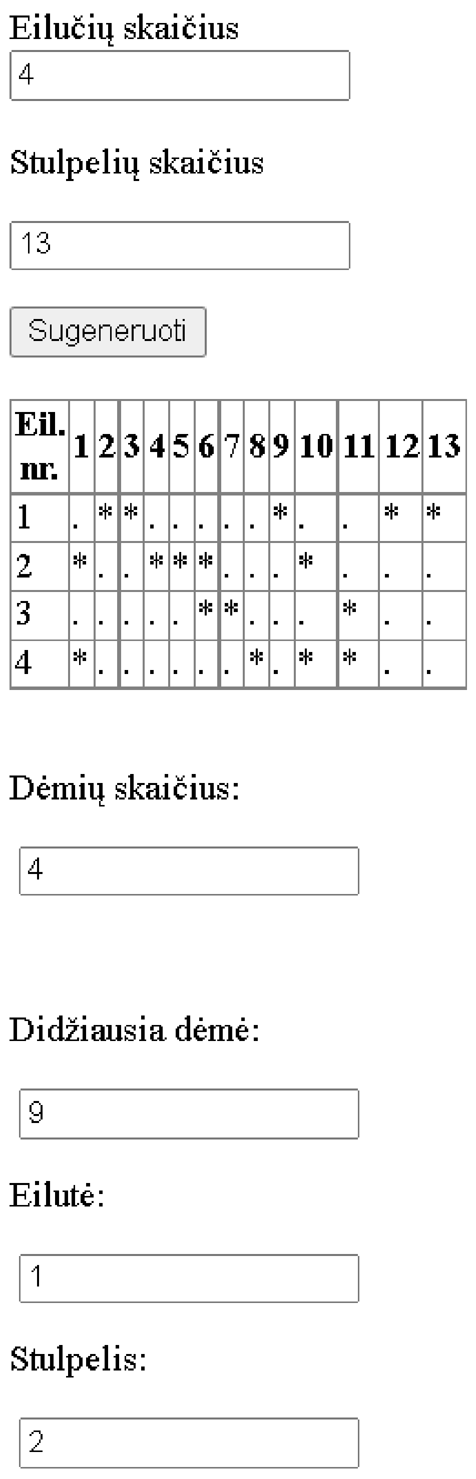
1) Pirmasis bandymas

   Pradiniai duomenys :
   - Eilučių skaičius – 4
   - Stulpelių skaičius - 13

Rezultatai:

rez.txt

```
Pradiniai duomenys
Eilutės skaičius 4
Stulpelių skaičius 13

Rezultatai:
Dėmių skaičius 6
Didžiausia dėmė 5
Didžiausios dėmės eilutė 1
Didžiausios dėmės stulpelis 4

Sugeneruota matrica:
      1     2     3     4     5     6     7     8     9     10    11    12    13

1     .     .     .     *     *     .     .     .     .     .     .     .     *


2     .     .     .     *     .     *     *     .     .     .     .     .     *


3     .     .     .     .     .     .     .     .     .     *     .     .     .


4     .     .     .     *     .     .     .     *     .     .     *     .     *
```



Eilučių skaičius

4

Stulpelių skaičius

13

Eil. nr.	1	2	3	4	5	6	7	8	9	10	11	12	13
1	.	*	*	.	.	.	.	.	*	.	.	*	*
2	*	.	.	*	*	*	.	.	.	*	.	.	.
3	.	.	.	.	.	*	*	.	.	.	*	.	.
4	*	.	.	.	.	.	.	*	.	*	*	.	.

Dėmių skaičius:

4

Didžiausia dėmė:

9

Eilutė:

1

Stulpelis:

2

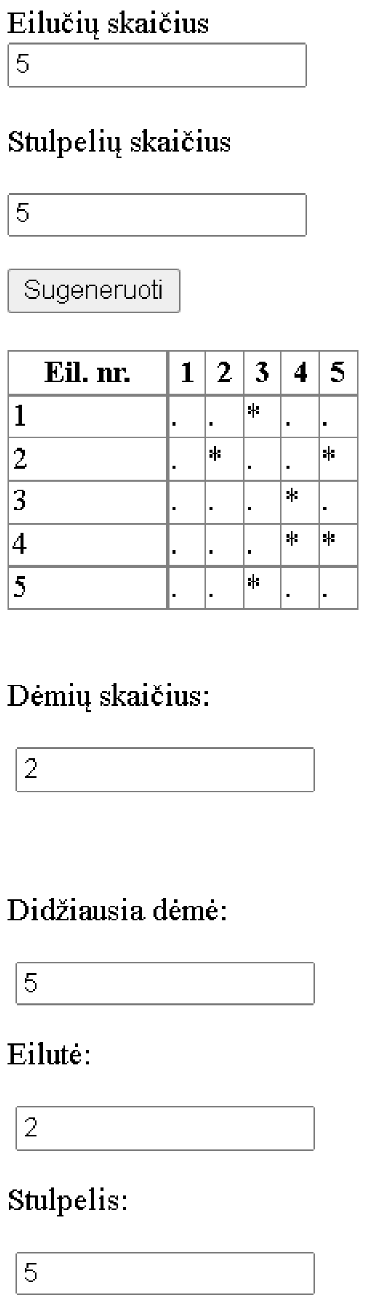
2) Antrasis bandymas

Pradiniai duomenys :

- Eilučių skaičius – 5
- Stulpelių skaičius - 5

Rezultatai :

rez.txt
```
Pradiniai duomenys
Eilutės skaičius 5
Stulpelių skaičius 5

Rezultatai:
Dėmių skaičius 2
Didžiausia dėmė 8
Didžiausios dėmės eilutė 2
Didžiausios dėmės stulpelis 1

Sugeneruota matrica:
      1     2     3     4     5
1     .     .     *     .     .

2     *     .     .     .     .

3     .     *     *     *     .

4     .     .     *     *     .

5     .     .     .     *     *
```



Eilučių skaičius

5

Stulpelių skaičius

5

Sugeneruoti

Eil. nr.	1	2	3	4	5
1	.	.	*	.	.
2	.	*	.	.	*
3	.	.	.	*	.
4	.	.	.	*	*
5	.	.	*	.	.

Dėmių skaičius:

2

Didžiausia dėmė:

5

Eilutė:

2

Stulpelis:

5

1.8. Dėstytojo pastabos

Klasiu diagrama (query tipas)

Rez.txt failas

Komentarai po metodų

2. Dinaminis atminties valdymas (L2)

2.1. Darbo užduotis

LD_19. Prekės. Prekybos tinklo parduotuvių rinkodaros skyriai atlieka prekių paklausos analizę. Sudarykite perkamiausių prekių sąrašą (prekės pavadinimas, prekės galiojimo laikas dienomis, prekės vieneto kaina). Sąrašas turi būti surikiuotas pagal prekių pavadinimus abėcėlės ir prekių vieneto kainas didėjimo tvarka. Sudarykite atskirą prekių, kurių galiojimo laikas baigiasi ne anksčiau kaip po 30 dienų nuo esamos datos, sąrašą (prekės pavadinimas, prekės galiojimo laiko pabaiga (metai, mėnuo, diena), prekės vieneto kaina). Sąrašas turi būti surikiuotas pagal prekių pavadinimus abėcėlės ir prekių vieneto kainas didėjimo tvarka. Suraskite, kurios parduotuvės rinkodaros skyrius turi didžiausią prekių asortimentą.

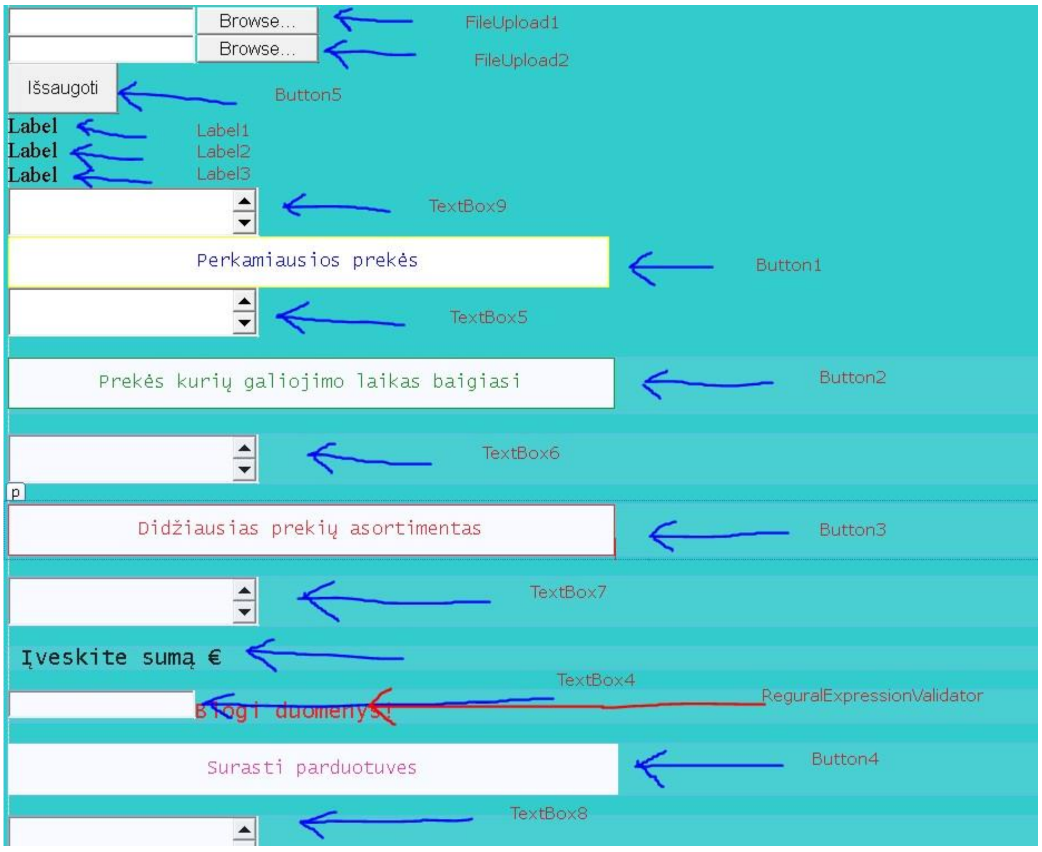
Duomenys:

• tekstiniame faile U19a.txt yra informacija apie parduotuves: parduotuvės pavadinimas, prekės pavadinimas, prekės gavimo laikas (metai, mėnuo, diena), parduotų prekių vienetų skaičius, prekių vienetų likutis parduotuvėje;

• tekstiniame faile U19b.txt yra informacija apie prekes: prekės pavadinimas, prekės galiojimo laikas dienomis, prekės vieneto kaina.

Sudarykite parduotuvių, kurios turi prekių už pinigų sumą, ne didesnę už nurodytą (įvedama klaviatūra), sąrašą (parduotuvės pavadinimas, prekių vienetų likutis parduotuvėje, pinigų suma).

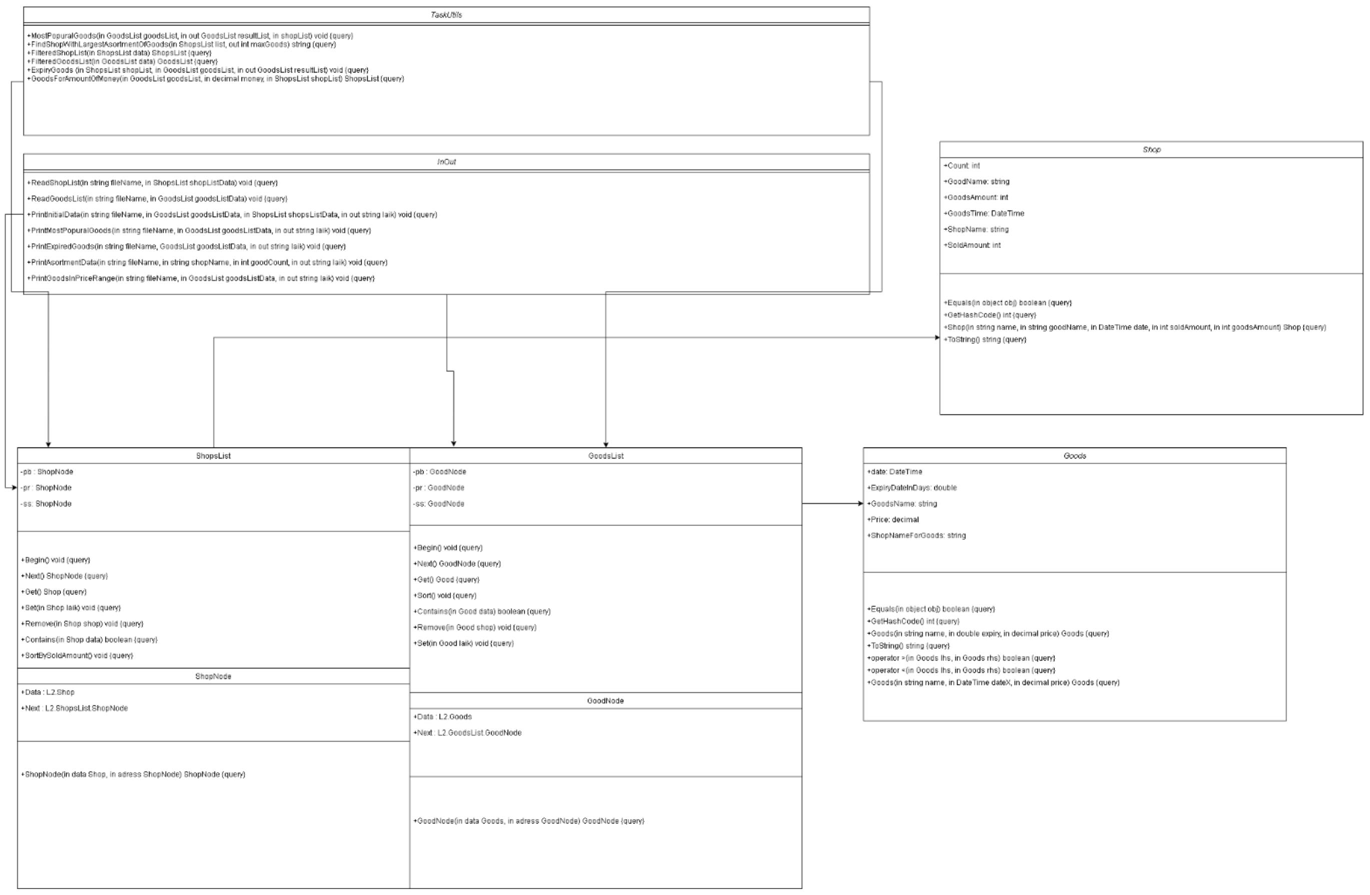
2.2. Grafinės vartotojo sąsajos schema



2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	**Savybė**	**Reikšmė**
FileUpload1	Niekas nepakeista	
FileUpload2	Niekas nepakeista	
Button5	Text	„Išsaugoti“
Label1	Visible	false
Label2	Visible	false
Button1	Text	„Perkamiausios prekės“
TextBox5	TextMode, Visible	MultiLine, false
Button2	Text	„Prekės kurių galiojimo laikas baigiasi“
TextBox6	TextMode, Visible	MultiLine, false
Button3	Text	Didžiausias prekių asortimentas
TextBox7	TextMode, Visible	MultiLine, false
Label3	Text	„Įveskite sumą €“
TextBox4	Niekas nepakeista	
ReguralExpressionValidator	ErrorMessage, ControlToValidate, ValidationGroup, ForeColor	„Blogi duomenys!“,TextBox4,^-?(([1-9]\d*)|0)(.0*[1-9](0*[1-9])*)?$, Red
Button4	Text	„Surasti parduotuves“
TextBox8	TextMode, Visible	MultiLine, false
Label3	Visible	False
TextBox9	Visible, TextMode	False, MultiLine

2.4. Klasių diagrama


TaskUtils
+MostPopuralGoods(in GoodsList goodsList, in out GoodsList resultList, in shopList) void {query}
+FindShopWithLargestAsortmentOfGoods(in ShopsList list, out int maxGoods) string {query}
+FilteredShopList(in ShopsList data) ShopsList {query}
+FilteredGoodsList(in GoodsList data) GoodsList {query}
+ExpiryGoods (in ShopsList shopList, in GoodsList goodsList, in out GoodsList resultList) void {query}
+GoodsForAmountOfMoney(in GoodsList goodsList, in decimal money, in ShopsList shopList) ShopsList {query}
InOut
+ReadShopList(in string fileName, in ShopsList shopListData) void {query}
+ReadGoodsList(in string fileName, in GoodsList goodsListData) void {query}
+PrintInitialData(in string fileName, in GoodsList goodsListData, in ShopsList shopsListData, in out string laik) void {query}
+PrintMostPopuralGoods(in string fileName, in GoodsList goodsListData, in out string laik) void {query}
+PrintExpiredGoods(in string fileName, GoodsList goodsListData, in out string laik) void {query}
+PrintAsortmentData(in string fileName, in string shopName, in int goodCount, in out string laik) void {query}
+PrintGoodsInPriceRange(in string fileName, in GoodsList goodsListData, in out string laik) void {query}
Shop
+Count: int
+GoodName: string
+GoodsAmount: int
+GoodsTime: DateTime
+ShopName: string
+SoldAmount: int
+Equals(in object obj) boolean {query}
+GetHashCode() int {query}
+Shop(in string name, in string goodName, in DateTime date, in int soldAmount, in int goodsAmount) Shop {query}
+ToString() string {query}
ShopsList
-pb : ShopNode
-pr : ShopNode
-ss: ShopNode
+Begin() void {query}
+Next() ShopNode {query}
+Get() Shop {query}
+Set(in Shop laik) void {query}
+Remove(in Shop shop) void {query}
+Contains(in Shop data) boolean {query}
+SortBySoldAmount() void {query}
ShopNode
+Data : L2.Shop
+Next : L2.ShopsList.ShopNode
+ShopNode(in data Shop, in adress ShopNode) ShopNode {query}
GoodsList
-pb : GoodNode
-pr : GoodNode
-ss: GoodNode
+Begin() void {query}
+Next() GoodNode {query}
+Get() Good {query}
+Sort() void {query}
+Contains(in Good data) boolean {query}
+Remove(in Good shop) void {query}
+Set(in Good laik) void {query}
GoodNode
+Data : L2.Goods
+Next : L2.GoodsList.GoodNode
+GoodNode(in data Goods, in adress GoodNode) GoodNode {query}
Goods
+date: DateTime
+ExpiryDateInDays: double
+GoodsName: string
+Price: decimal
+ShopNameForGoods: string
+Equals(in object obj) boolean {query}
+GetHashCode() int {query}
+Goods(in string name, in double expiry, in decimal price) Goods {query}
+ToString() string {query}
+operator >(in Goods lhs, in Goods rhs) boolean {query}
+operator <(in Goods lhs, in Goods rhs) boolean {query}
+Goods(in string name, in DateTime dateX, in decimal price) Goods {query}

2.5. Programos vartotojo vadovas

Pradžioje pasirenkame du pradinių duomenų failus su parduotuvės ir prekių informacija. Pasirinke šiuos failus spaudžiame mygtuką ,,Išsaugoti‘‘ ir taip išsaugome šiuos duomenų failus į sesiją. Visus kitus rezultatus gauname paspaudus mygtukus ,,Perkamiausios prekės‘‘ (Perkamiausių prekių rezultatai) , ,,Prekių, kurių galiojimo laikas baigiasi‘‘ (Prekių, kurių galiojimo laikas baigiasi rezultatai), ,,Didžiausias prekių asortimentas‘‘ (Didžiausio asortimento rezultatai) ir prieš spaudžiant ,,Surasti parduotuves‘‘ mygtuką privalome įvesti pinigų sumą į textbox elementą ir tik tada spausti mygtuką, taip išvedami (Prekės už pinigų sumą rezultatai).

2.6. Programos tekstas

Shop.cs

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

namespace L2
{
    /// <summary>
    /// Data class for shop information
    /// </summary>
    public class Shop
    {
        public string ShopName { get; set; }
        public string GoodName { get; set; }
        public DateTime GoodsTime { get; set; }
        public int SoldAmount { get; set; }
        public int GoodsAmount { get; set; }
        public int Count;

        /// <summary>
        /// Shop constructor for initial data
        /// </summary>
        /// <param name="name">Shop name</param>
        /// <param name="goodName">Goods name</param>
        /// <param name="goodTime">Date when good were received</param>
        /// <param name="soldAmount">Sold amount of goods</param>
        /// <param name="goodsAmount">Remaining amount of goods</param>
        public Shop (string name, string goodName, DateTime goodTime, int 
soldAmount, int goodsAmount)
        {
            this.ShopName = name;
            this.GoodName = goodName;
            this.GoodsTime = goodTime;
            this.SoldAmount = soldAmount;
            this.GoodsAmount = goodsAmount;
        }

        public override bool Equals(object obj)
        {
            return ShopName == ((Shop)obj).ShopName && GoodName == 
((Shop)obj).GoodName && GoodsTime == ((Shop)obj).GoodsTime && SoldAmount == 
((Shop)obj).SoldAmount && GoodsAmount == ((Shop)obj).GoodsAmount;
        }

        public override int GetHashCode()
        {
            return ShopName.GetHashCode() ^ GoodName.GetHashCode() ^ 
GoodsTime.GetHashCode() ^ SoldAmount.GetHashCode() ^ GoodsAmount.GetHashCode();
        }

        public override string ToString()
        {
            return string.Format("|{0,-25}|{1,-
25}|{2,25:yyyy/MM/dd}|{3,40}|{4,40}|", ShopName, GoodName, GoodsTime, SoldAmount, 
GoodsAmount);
        }
    }
}
```

Goods.cs

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

namespace L2
{
    /// <summary>
    /// Data class for Goods information
    /// </summary>
    public class Goods
    {
        public string GoodsName { get; set; }
        public double ExpiryDateinDays { get; set; }
        public decimal Price { get; set; }
        public DateTime date { get; set; }
        public string ShopNameForGoods { get; set; }

        /// <summary>
        /// Constructor for Goods initial data
        /// </summary>
        /// <param name="name">Goods name/param>
        /// <param name="expiry">Expiry date in days</param>
        /// <param name="price">Goods price</param>
        public Goods (string name, double expiry, decimal price)
        {
            this.GoodsName = name;
            this.ExpiryDateinDays = expiry;
            this.Price = price;
        }
        /// <summary>
        /// Constructor for Goods data
        /// </summary>
        /// <param name="name">Goods name</param>
        /// <param name="dateX">Date</param>
        /// <param name="price">Price</param>
        public Goods(string name, DateTime dateX, decimal price)
        {
            this.GoodsName = name;
            this.date = dateX;
            this.Price = price;
        }

        static public bool operator >(Goods lhs, Goods rhs)
        {
            return lhs.GoodsName.CompareTo(rhs.GoodsName) > 0 ||
lhs.GoodsName.CompareTo(rhs.GoodsName) == 0 && lhs.Price.CompareTo(rhs.Price) > 0;
        }


        static public bool operator <(Goods lhs, Goods rhs)
        {
            return lhs.GoodsName.CompareTo(rhs.GoodsName) < 0 ||
lhs.GoodsName.CompareTo(rhs.GoodsName) == 0 && lhs.Price.CompareTo(rhs.Price) < 0;
        }

        public override bool Equals(object obj)
        {
            return GoodsName == ((Goods)obj).GoodsName && ExpiryDateinDays ==
((Goods)obj).ExpiryDateinDays && Price == ((Goods)obj).Price;
        }

        public override int GetHashCode()
        {
            return GoodsName.GetHashCode() ^ ExpiryDateinDays.GetHashCode() ^
Price.GetHashCode();
        }

        public override string ToString()
        {
            return string.Format("|{0,-25}|{1,40}|{2,25}|", GoodsName,
ExpiryDateinDays, Price);
        }
    }
}
```

ShopsList.cs

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

namespace L2
{
    /// <summary>
    /// Linked list class for Shop data
    /// </summary>
    public sealed class ShopsList
    {

        public sealed class ShopNode
        {

            public Shop Data { get; set; }
            public ShopNode Next { get; set; }

            public ShopNode()
            {

            }

            public ShopNode(Shop data, ShopNode adress)
            {
                Data = data;
                Next = adress;
            }
        }

        private ShopNode pr;
        private ShopNode pb;
        private ShopNode ss;

        public ShopsList()
        {
            this.pr = null;
            this.pb = null;
            ss = pr;
        }



        /// <summary>
        /// Method for starting object in cycles
        /// </summary>
        public void Begin()
        {
            ss = pr;
        }

        /// <summary>
        /// Method for checking if the next object exists
        /// </summary>
        /// <returns>if next object exists</returns>
        public bool Is()
        {
```

```
    return (ss != null);
}

/// <summary>
/// Method for getting the next object from a cycle
/// </summary>
/// <returns>next object</returns>
public ShopNode Next()
{
    ss = ss.Next;
    return ss;
}


/// <summary>
/// Method for getting an object
/// </summary>
/// <returns>object data</returns>
public Shop Get()
{
    return (ss.Data);
}



/// <summary>
/// Method for putting data to a linked list
/// </summary>
/// <param name="laik">object data</param>
public void Set(Shop laik)
{
    if (pr == null)
    {
        pr = new ShopNode(laik, null);
        pb = pr;
    }
    else
    {
        pb.Next = new ShopNode(laik, null);
        pb = pb.Next;
    }
}


/// <summary>
/// Method for removing an object from a linked list
/// </summary>
/// <param name="shop"></param>
public void Remove(Shop shop)
{
    for (ShopNode s = pr; s.Next != null; s = s.Next)
    {
        if (s.Next.Data == shop)
        {
            // kam tau cia to temp, jei su juo nieko nedarai?
            ShopNode temp = s.Next;
            s.Next = s.Next.Next;
            temp = null;
        }
    }
}

/// <summary>
/// Method for checking if a linked list contains an object
/// </summary>
```

```
        /// <param name="data">object data</param>
        /// <returns></returns>
        public bool Contains(Shop data)
        {
            for (ShopNode s = pr; s.Next != null; s = s.Next)
            {
                if (s.Data == data)
                {
                    return true;
                }
            }
            return false;


        }

        /// <summary>
        /// Method for sorting goods by sold amount
        /// </summary>
        public void SortBySoldAmount()
        {
            if (pr == null) { return; }
            bool keista = true;
            while (keista)
            {
                keista = false;
                ShopNode pra = pr;
                while (pra.Next != null)
                {
                    if (pra.Data.SoldAmount < pra.Next.Data.SoldAmount)
                    {
                        Shop St = pra.Data;
                        pra.Data = pra.Next.Data;
                        pra.Next.Data = St;
                        keista = true;

                    }
                    pra = pra.Next;
                }
            }
        }
    }
}
```

GoodsList.cs

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

namespace L2
{
    /// <summary>
    /// Linked list class for Goods data
    /// </summary>
    public sealed class GoodsList
    {
        public sealed class GoodNode
        {
            public Goods Data { get; set; }
            public GoodNode Next { get; set; }

            public GoodNode()
            {
            }

            public GoodNode(Goods data, GoodNode adress)
            {
                Data = data;
                Next = adress;
            }

        }
        private GoodNode pr;
        private GoodNode pb;
        private GoodNode ss;

        public GoodsList()
        {
            this.pr = null;
            this.pb = null;
            ss = pr;
        }
        /// <summary>
        /// Method for start object in cycles
        /// </summary>
        public void Begin()
        {
            ss = pr;
        }

        /// <summary>
        /// Method for checking if an object exists in cycles
        /// </summary>
        /// <returns>if next object exists</returns>
        public bool Is()
        {
            return (ss != null);
        }


        /// <summary>
        /// Method for next object in cycles
        /// </summary>
        /// <returns>next object</returns>
        public GoodNode Next()
        {
            ss = ss.Next;
            return ss;
        }


        /// <summary>
        /// Method for getting the object
        /// </summary>
        /// <returns></returns>
        public Goods Get()
        {
            return (ss.Data);
        }



        /// <summary>
        /// Method for adding objects to a linked list
        /// </summary>
        /// <param name="laik">Goods data</param>
        public void Set(Goods laik)
```

```
{
    if (pr == null)
    {
        pr = new GoodNode(laik, null);
        pb = pr;
    }
    else
    {
        pb.Next = new GoodNode(laik, null);
        pb = pb.Next;
    }
}
/// <summary>
/// Bubble method for sorting goods by price and name
/// </summary>
public void Sort()
{

    if (pr == null) { return; }
    bool flag = true;
    while (flag)
    {
        flag = false;
        GoodNode pra = pr;
        while (pra.Next != null)
        {
            if (pra.Data > pra.Next.Data)
            {
                Goods St = pra.Data;
                pra.Data = pra.Next.Data;
                pra.Next.Data = St;
                flag = true;

            }
            pra = pra.Next;
        }
    }
}

/// <summary>
/// Method for removing selected objects
/// </summary>
/// <param name="goods">object data</param>
public void Remove(Goods goods)
{
    for (GoodNode s = pr; s.Next != null; s = s.Next)
    {
        if (s.Next.Data == goods)
        {
            GoodNode temp = s.Next;
            s.Next = s.Next.Next;
            temp = null;
        }
    }
}



/// <summary>
/// Method for checking if a linked list contains this object
/// </summary>
/// <param name="data">object data</param>
/// <returns></returns>
public bool Contains(Goods data)
{
```

```
            for (GoodNode s = pr; s.Next != null; s = s.Next)
            {
                if (s.Data == data)
                {
                    return true;
                }

            }
            return false;
        }
    }
}
```

InOut.cs

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

namespace L2
{
    /// <summary>
    /// Class for inputing and outputing data
    /// </summary>
    public static class InOut
    {
        /// <summary>
        /// Method for reading Shop file data
        /// </summary>
        /// <param name="fileName">Shop file with data</param>
        /// <param name="shopListData">List to put shop file data</param>
        public static void ReadShopList(string fileName, ShopsList shopListData)
        {
            string eil = null;
            using (var file = new StreamReader(fileName))
            {
                while ((eil = file.ReadLine()) != null)
                {
                    string[] values = eil.Split(';');
                    string shopName = values[0];
                    string goodsName = values[1];
                    DateTime goodsTime = DateTime.Parse(values[2]);
                    int soldAmount = int.Parse(values[3]);
                    int goodsAmount = int.Parse(values[4]);
                    Shop C = new Shop(shopName, goodsName, goodsTime, soldAmount,
goodsAmount);
                    shopListData.Set(C);
                }
            }
        }


        /// <summary>
        /// Method for reading Goods file data
        /// </summary>
        /// <param name="fileName">Goods file with data</param>
        /// <param name="goodsListData">List to put goods file data</param>
       public static void ReadGoodsList(string fileName, GoodsList goodsListData)
        {
            string eil = null;
            using (var file = new StreamReader(fileName))
            {
                while ((eil = file.ReadLine()) != null)
                {
                    string[] values = eil.Split(';');
                    string goodName = values[0];
                    double expiry = double.Parse(values[1]);
                    decimal price = decimal.Parse(values[2]);
                    Goods C = new Goods(goodName, expiry, price);
                    goodsListData.Set(C);
                }
            }
        }

        /// <summary>
        /// Method for printing initial data
        /// </summary>
        /// <param name="fileName">resultFile</param>
        /// <param name="goodsListData">GoodsList with data</param>
        /// <param name="shopsListData">ShopsList with data</param>
        /// <param name="laik">string with data</param>
        public static void PrintInitialData(string fileName, GoodsList
goodsListData, ShopsList shopsListData, ref string laik)
        {
            using (StreamWriter write = File.CreateText(fileName))
            {
                laik = "Parduotuvės Sąrašas:\n";
                write.WriteLine("Pradiniai duomenys");
                write.WriteLine("");
                laik += new string('-', 170) + "\n";
                laik += string.Format("|{0,-25}|{1,-25}|{2,-25}|{3,-40}|{4,-40}|",
"Parduotuvės pavadinimas", "Prekės pavadinimas", "Prekės gavimo laikas", "Parduotų
vienetų skaičius", "Prekių vienetų likutis parduotuvėje") + "\n";
                laik += new string('-', 170) + "\n";
                write.WriteLine("Parduotuvės Sąrašas: ");
                write.WriteLine("-----------------------------------------------
------------------------------------------------------------------------------------
------------------------------");
                write.WriteLine(string.Format("|{0,-25}|{1,-25}|{2,-25}|{3,-
40}|{4,-40}|", "Parduotuvės pavadinimas", "Prekės pavadinimas", "Prekės gavimo
laikas", "Parduotų vienetų skaičius", "Prekių vienetų likutis parduotuvėje"));
                write.WriteLine("-----------------------------------------------
------------------------------------------------------------------------------------
------------------------------");
                for (shopsListData.Begin(); shopsListData.Is();
shopsListData.Next())
                {
                    laik += shopsListData.Get().ToString() + "\n";
                    laik += new string('-', 170) + "\n";
                    write.WriteLine(shopsListData.Get().ToString());
                    write.WriteLine("-------------------------------------------
--------------------------------------------------------------------------------
----------------------------------");
                }
                laik += "\n";
                laik += "Prekių sąrašas:\n";
                laik += new string('-', 94) + "\n";
                laik += string.Format("|{0,-25}|{1,-40}|{2,-25}|", "Prekės
pavadinimas", "Prekės galiojimo laikas dienomis", "Prekės vieneto kaina") + "\n";
                write.WriteLine("Prekių Sąrašas: ");
                write.WriteLine("-----------------------------------------------
--------------------------------------------");
                write.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-25}|" ,"Prekės
pavadinimas", "Prekės galiojimo laikas dienomis", "Prekės vieneto kaina"));
                write.WriteLine("-----------------------------------------------
--------------------------------------------");
                for (goodsListData.Begin(); goodsListData.Is();
goodsListData.Next())
                {


                    laik += goodsListData.Get().ToString() + "\n";
                    laik += new string('-', 94) + "\n";
                    write.WriteLine(goodsListData.Get().ToString());
                    write.WriteLine("-------------------------------------------
------------------------------------------------");
                }
                write.WriteLine("");
                laik += "\n";
            }
        }

        /// <summary>
        /// Method for printing MostPopuralGoods data to a file and a string
        /// </summary>
        /// <param name="fileName">Result file</param>
        /// <param name="goodsListData">GoodsList linked list with most popural
goods data</param>
        /// <param name="laik">string to write most popural goods data</param>
        public static void PrintMostPopuralGoods(string fileName, GoodsList
goodsListData, ref string laik)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                string dashes = new string('-', 94);
                laik = "\n";
                laik += "Populiariausių prekių sąrašas:\n";
                sw.WriteLine("Populiariausių prekių sąrašas");
                sw.WriteLine("");
                laik += new string('-', 94) + "\n";
                laik += string.Format("|{0,-25}|{1,-40}|{2,-25}|", "Prekės
pavadinimas", "Prekės galiojimo laikas dienomis", "Prekės vieneto kaina") + "\n";
                laik += new string('-', 94) + "\n";
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-25}|", "Prekės
pavadinimas", "Prekės galiojimo laikas dienomis", "Prekės vieneto kaina"));
                sw.WriteLine(dashes);
                for (goodsListData.Begin(); goodsListData.Is();
goodsListData.Next())
                {

                    laik += string.Format("|{0,-25}|{1,40}|{2,25}|",
goodsListData.Get().GoodsName, goodsListData.Get().ExpiryDateinDays,
goodsListData.Get().Price) + "\n";
                    laik += new string('-', 94) + "\n";

                    sw.WriteLine(string.Format("|{0,-25}|{1,40}|{2,25}|",
goodsListData.Get().GoodsName, goodsListData.Get().ExpiryDateinDays,
goodsListData.Get().Price));
                    sw.WriteLine(dashes);
                }
                sw.WriteLine("");

            }
        }

        /// <summary>
        /// Method for printing expired goods data
        /// </summary>
        /// <param name="fileName">Result file</param>
        /// <param name="goodsListData">GoodsList with expired goods data</param>
        /// <param name="laik">string to write expired goods data</param>
        public static void PrintExpiredGoods(string fileName, GoodsList
goodsListData, ref string laik)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                string dashes = new string('-', 109);
                laik = "\n";
                laik = "Šalia galiojimo laiko pabaigos sąrašas:\n";
                sw.WriteLine("Šalia galiojimo laiko pabaigos sąrašas");
                sw.WriteLine("");
                laik += new string('-', 109) + "\n";
                laik += string.Format("|{0,-25}|{1,-40}|{2,-40}|", "Prekės
pavadinimas", "Prekės galiojimo laiko pabaiga", "Prekės vieneto kaina") + "\n";
                laik += new string('-', 109) + "\n";
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-40}|", "Prekės
pavadinimas", "Prekės galiojimo laiko pabaiga", "Prekės vieneto kaina"));
                sw.WriteLine(dashes);
                for (goodsListData.Begin(); goodsListData.Is();
goodsListData.Next())
                {

                    laik += string.Format("|{0,-25}|{1,40:yyyy/MM/dd}|{2,40}|",
goodsListData.Get().GoodsName, goodsListData.Get().date,
goodsListData.Get().Price) + "\n";
                    laik += new string('-', 109) + "\n";

                    sw.WriteLine(string.Format("|{0,-
25}|{1,40:yyyy/MM/dd}|{2,40}|", goodsListData.Get().GoodsName,
goodsListData.Get().date, goodsListData.Get().Price));
                    sw.WriteLine(dashes);
                }
                sw.WriteLine("");

            }
        }



        /// <summary>
        /// Method for printing asortment data
        /// </summary>
        /// <param name="fileName">Result file</param>
        /// <param name="shopName">The name of the shop</param>
        /// <param name="goodCount">The count of asortment</param>
        /// <param name="laik">string to write results</param>
        public static void PrintMostAsortmentData(string fileName, string
shopName,int goodCount, ref string laik)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                string dashes = new string('-', 53);
                laik = "\n";
                laik += "Didžiausio asortimento sąrašas:\n";
                sw.WriteLine("Didžiausio asortimento sąrašas");
                sw.WriteLine("");
                laik += new string('-', 53) + "\n";
                laik += string.Format("|{0,-25}|{1,-25}|", "Parduotuvės
pavadinimas", "Prekių kiekis") + "\n";
                laik += new string('-', 53) + "\n";
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-25}|{1,-25}|", "Parduotuvės
pavadinimas", "Prekių kiekis"));
                sw.WriteLine(dashes);
                laik += string.Format("|{0,-25}|{1,25}|", shopName, goodCount) +
"\n";
                laik += new string('-', 53) + "\n";
                sw.WriteLine(string.Format("|{0,-25}|{1,25}|", shopName,
goodCount));
                sw.WriteLine(dashes);
                sw.WriteLine("");

            }

        }

        /// <summary>
        /// Method for printing linked list with goods within particular price
range
        /// </summary>
        /// <param name="fileName">Result file</param>
        /// <param name="goodsListData">GoodsList linked list within particular
data in a given price range</param>
        /// <param name="laik">string to write linked list data within particular
price range</param>
        public static void PrintGoodsInPriceRange(string fileName, GoodsList
goodsListData, ref string laik)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                string dashes = new string('-', 94);
                laik = "\n";
                laik = "Prekės už pinigų sumą \n";
                sw.WriteLine("Prekės už pinigų sumą");
                sw.WriteLine("");
                laik += new string('-', 94) + "\n";
                laik += string.Format("|{0,-25}|{1,-40}|{2,-25}|", "Parduotuvės 
pavadinimas", "Prekių vienetų likutis parduotuvėje", "Pinigų suma") + "\n";
                laik += new string('-', 94) + "\n";
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-25}|", 
"Parduotuvės pavadinimas", "Prekių vienetų likutis parduotuvėje", "Pinigų suma"));
                sw.WriteLine(dashes);
                for (goodsListData.Begin(); goodsListData.Is(); 
goodsListData.Next())
                {

                    laik += string.Format("|{0,-25}|{1,40}|{2,25}|", 
goodsListData.Get().GoodsName, goodsListData.Get().ExpiryDateinDays, 
goodsListData.Get().Price) + "\n";
                    laik += new string('-', 94) + "\n";

                    sw.WriteLine(string.Format("|{0,-25}|{1,40}|{2,25}|", 
goodsListData.Get().GoodsName, goodsListData.Get().ExpiryDateinDays, 
goodsListData.Get().Price));
                    sw.WriteLine(dashes);
                }
                sw.WriteLine("");

            }
        }




    }
}
```

TaskUtils.cs

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

namespace L2
{
    /// <summary>
    /// Class for operations with data
    /// </summary>
    public static class TaskUtils
    {
        /// <summary>
        /// Method for finding most popural Goods
        /// </summary>
        /// <param name="goodsList">GoodsList with goods data</param>
        /// <param name="resultList">empty GoodsList to write data</param>
        /// <param name="shopList">ShopsList with shops data</param>
        public static void MostPopuralGoods(GoodsList goodsList, ref GoodsList 
resultList, ShopsList shopList)
        {
            shopList.SortBySoldAmount();
            ShopsList temp = new ShopsList(); //pard p, prekes p, data galiojimo, 
parduotos prekes, likutis
            temp = shopList;
            for (temp.Begin(); temp.Is(); temp.Next())
            {
                for (goodsList.Begin(); goodsList.Is(); goodsList.Next())
                {
                    if (goodsList.Get().GoodsName == temp.Get().GoodName)
                    {
                        Goods C = new Goods(goodsList.Get().GoodsName, 
goodsList.Get().ExpiryDateinDays, goodsList.Get().Price);
                        resultList.Set(C);
                    }
                }
            }
        }


        /// <summary>
        /// Method for finding shop with most asortment
        /// </summary>
        /// <param name="list">Shop list data</param>
        /// <param name="maxGoods">Count of different asortment</param>
        /// <returns>The shop name with most asortment and the count of 
asortment</returns>
        public static string FindShopWithLargestAsortmentOfGoods(ShopsList list, 
out int maxGoods)
        {
            Dictionary<string, List<string>> allGoodsInShop = new 
Dictionary<string, List<string>>();
            for(list.Begin(); list.Is(); list.Next())
            {
                Shop shop = list.Get();
                if(!allGoodsInShop.ContainsKey(shop.ShopName))
                {
                    allGoodsInShop.Add(shop.ShopName, new List<string>());
                }
                if (!allGoodsInShop[shop.ShopName].Contains(shop.GoodName))
                {
                    allGoodsInShop[shop.ShopName].Add(shop.GoodName);
                }

            }
            maxGoods = -1;
            string maxShop = "";
            foreach (KeyValuePair<string, List<string>> keyValuePair in 
allGoodsInShop)
            {
                if (maxGoods < 0 || keyValuePair.Value.Count() > maxGoods)
                {
                    maxGoods = keyValuePair.Value.Count();
                    maxShop = keyValuePair.Key;
                }
            }

            return maxShop;
        }
        /// <summary>
        /// Method for filtering the same data from a shops linked list
        /// </summary>
        /// <param name="data">Linked list with data</param>
        /// <returns>Filtered linked list</returns>
        public static ShopsList FilteredShopList(ShopsList data)
        {
            ShopsList temp = new ShopsList();
            temp = data;
            for (temp.Begin(); temp.Is(); temp.Next())
            {
                Shop temp2 = (Shop)temp.Get();
                if (data.Contains(temp2))
                {
                    temp.Remove(temp2);
                }

            }
            return temp;

        }

        /// <summary>
        /// Method for filtering the same data from a goods linked list
        /// </summary>
        /// <param name="data">Linked list with data</param>
        /// <returns>Filtered linked list</returns>
        public static GoodsList FilteredGoodsList(GoodsList data)
        {
            GoodsList temp = new GoodsList();
            temp = data;
            for (temp.Begin(); temp.Is(); temp.Next())
            {
                Goods temp2 = (Goods)temp.Get();
                if (data.Contains(temp2))
                {
                    temp.Remove(temp2);
                }

            }
            return temp;
        }

        /// <summary>
        /// Method for finding a list with expiry goods
        /// </summary>
        /// <param name="shopList">ShopsList with shops data</param>
        /// <param name="goodsList">GoodsList with goods data</param>
        /// <param name="resultList">Empty GoodsList to write results</param>
        public static void ExpiryGoods(ShopsList shopList, GoodsList goodsList,
ref GoodsList resultList)
        {
            for (shopList.Begin(); shopList.Is(); shopList.Next())
            {
                for (goodsList.Begin(); goodsList.Is(); goodsList.Next())
                {
                    if (goodsList.Get().ExpiryDateinDays <= 30 &&
shopList.Get().GoodName == goodsList.Get().GoodsName)
                    {
                        Goods C = new Goods(goodsList.Get().GoodsName,
shopList.Get().GoodsTime.AddDays(goodsList.Get().ExpiryDateinDays),
goodsList.Get().Price);
                        resultList.Set(C);
                    }
                }
            }
        }


        /// <summary>
        /// Method for finding goods for a particular amount of money
        /// </summary>
        /// <param name="goodsList">GoodsList with goods data</param>
        /// <param name="money">Amount of money</param>
        /// <param name="shopsList">ShopsList with shops data</param>
        /// <returns>GoodsList with data</returns>
        public static GoodsList GoodsForAmountOfMoney(GoodsList goodsList, decimal
money, ShopsList shopsList)
        {
            GoodsList list = new GoodsList();
            for (goodsList.Begin(); goodsList.Is(); goodsList.Next())
            {
                for (shopsList.Begin(); shopsList.Is(); shopsList.Next())
                {
                    if (goodsList.Get().Price <= money)
                    {
                        if (goodsList.Get().GoodsName == shopsList.Get().GoodName)
                        {
                            Goods C = new Goods(shopsList.Get().ShopName,
shopsList.Get().GoodsAmount, goodsList.Get().Price);
                            list.Set(C);
                        }
                    }
                }
            }
            return list;
        }


    }



}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="L2.Forma1" MaintainScrollPositionOnPostBack="true" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>

    <link href="StyleSheet1.css" rel="stylesheet"/>
</head>

<body>

    <form id="form1" runat="server" enctype="multipart/form-data">
            <img class ="image1"
src="https://ychef.files.bbci.co.uk/976x549/p087r1np.jpg"  alt="Sample Photo"
/><br />
        <br />
        <asp:FileUpload ID="FileUpload1" runat="server" />
        <br />
        <asp:FileUpload ID="FileUpload2" runat="server" />
        <br />
        <br />
        <asp:Button ID="Button5" runat="server" OnClick="Button5_Click"
Text="Išsaugoti" Height="46px" />
        <br />
        <br />
        <asp:Label ID="Label1" runat="server" Text="Label"
Visible="False"></asp:Label>
        <br />
        <asp:Label ID="Label2" runat="server" Text="Label"
Visible="False"></asp:Label>
        <br />

        <br />

        <asp:Button CssClass ="ButtonClass1"  type ="button"  ID="Button1"
runat="server" Text="Perkamiausios prekės" OnClick="Button1_Click" Height="46px"
Width="540px" />
        <br />
        <asp:TextBox ID="TextBox5" runat="server" TextMode="MultiLine"
Visible="False"></asp:TextBox>
        <br />
        <p>
            <asp:Button  CssClass ="ButtonClass2" type ="button" ID="Button2"
runat="server" Text="Prekės kurių galiojimo laikas baigiasi"
OnClick="Button2_Click" Height="46px" Width="545px" />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        </p>
        <p>
            <asp:TextBox ID="TextBox6" runat="server" TextMode="MultiLine"
Visible="False"></asp:TextBox>
        </p>
        <p>
            <asp:Button  CssClass ="ButtonClass3"  type ="button"   ID="Button3"
runat="server" Text="Didžiausias prekių asortimentas" OnClick="Button3_Click"
Height="46px" Width="545px" />
        </p>
        <p>
            <asp:TextBox ID="TextBox7" runat="server" TextMode="MultiLine"
Visible="False"></asp:TextBox>
        </p>
        <p>
            &nbsp;Įveskite sumą €
        </p>
        <p>
            <asp:TextBox ID="TextBox4" runat="server"
OnTextChanged="TextBox4_TextChanged"></asp:TextBox>
            <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ControlToValidate="TextBox4" ErrorMessage="Blogi duomenys!"
ForeColor="Red" ValidationExpression="^-?(([1-9]\d*)|0)(.0*[1-9](0*[1-
9])*)?$"></asp:RegularExpressionValidator>
        </p>
        <p>
            <asp:Button  CssClass ="ButtonClass4"  type ="button"  ID="Button4"
runat="server" Text="Surasti parduotuves" OnClick="Button4_Click" Height="46px"
Width="548px" />
        </p>
        <p>
```

```
            <asp:TextBox ID="TextBox8" runat="server"
OnTextChanged="TextBox4_TextChanged" TextMode="MultiLine"
Visible="False"></asp:TextBox>
        </p>




    </form>
</body>
</html>
```

Forma1.aspx.cs

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

namespace L2
{
    public partial class Forma1 : System.Web.UI.Page
    {



        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            string laik = string.Empty;
            string prad = string.Empty;
            TextBox5.Visible = true;
            GoodsList Goods = new GoodsList();
            ShopsList Shop = new ShopsList();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            InOut.ReadShopList((string)Session["firstFile"], Shop);
            InOut.ReadGoodsList((string)Session["secondFile"], Goods);
            InOut.PrintInitialData(resultFile, Goods, Shop, ref prad);
            Label3.Text = "Pradiniai duomenys";
            Label3.Visible = true;
            TextBox9.Visible = true;
            TextBox9.Text += prad;
            GoodsList MostPopuralGoodsList = new GoodsList();
            TaskUtils.MostPopuralGoods(Goods, ref MostPopuralGoodsList, Shop);
            MostPopuralGoodsList.Sort();
            InOut.PrintMostPopuralGoods(resultFile, MostPopuralGoodsList, ref
laik);
            TextBox5.Text += laik;
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            string laik = string.Empty;
            TextBox6.Visible = true;
            GoodsList Goods = new GoodsList();
            ShopsList Shop = new ShopsList();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            InOut.ReadShopList((string)Session["firstFile"], Shop);
            InOut.ReadGoodsList((string)Session["secondFile"], Goods);
            GoodsList expiredList = new GoodsList();
            TaskUtils.ExpiryGoods(Shop, Goods, ref expiredList);
            expiredList.Sort();
            InOut.PrintExpiredGoods(resultFile, expiredList, ref laik);
            TextBox6.Text += laik;
        }

        protected void Button3_Click(object sender, EventArgs e)
        {
            string laik = string.Empty;
            TextBox7.Visible = true;
            GoodsList Goods = new GoodsList();
            ShopsList Shops = new ShopsList();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            InOut.ReadShopList((string)Session["firstFile"], Shops);
            InOut.ReadGoodsList((string)Session["secondFile"], Goods);
            string largerstAssortmentShopName =
TaskUtils.FindShopWithLargestAsortmentOfGoods(Shops, out int countOfGoods);
            InOut.PrintMostAsortmentData(resultFile, largerstAssortmentShopName,
countOfGoods, ref laik);
            TextBox7.Text += laik;
        }

        protected void Button4_Click(object sender, EventArgs e)
        {
            decimal moneyAmount = decimal.Parse(TextBox4.Text.ToString());
            GoodsList Goods = new GoodsList();
            ShopsList Shop = new ShopsList();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            string laik = string.Empty;
            TextBox8.Visible = true;
            InOut.ReadShopList((string)Session["firstFile"], Shop);
            InOut.ReadGoodsList((string)Session["secondFile"], Goods);
            GoodsList list = TaskUtils.GoodsForAmountOfMoney(Goods, moneyAmount,
Shop);
            InOut.PrintGoodsInPriceRange(resultFile, list, ref laik);
            TextBox8.Text += laik;

        }

        protected void Button5_Click(object sender, EventArgs e)
        {
            if (FileUpload1.HasFile && FileUpload2.HasFile)
            {
                string firstFileFormat = Path.GetExtension(FileUpload1.FileName);
                string secondFileFormat = Path.GetExtension(FileUpload2.FileName);

                if (firstFileFormat.ToLower() != ".txt" &&
secondFileFormat.ToLower() != ".txt")
                {
                    Label1.Visible = true;
                    Label1.Text = "Blogas failų formatas!";
                    Label1.ForeColor = System.Drawing.Color.Red;
                }
                else
                {
                    Label1.Visible = false;
                    Label2.Visible = true;
                    string firstFileData = Server.MapPath("/App_Data1/" +
FileUpload1.FileName);
                    string secondFileData = Server.MapPath("/App_Data1/" +
FileUpload2.FileName);
```

```
                        Label2.Text = "Failai sėkmingai pasirinkti!";
                        Label2.ForeColor = System.Drawing.Color.Green;
                        Session["firstFile"] = firstFileData;
                        Session["secondFile"] = secondFileData;
                    }
                }
                else
                {
                    Label1.Visible = true;
                    Label1.Text = "Nepasirinkti failai!";
                    Label1.ForeColor = System.Drawing.Color.Red;
                }
            }

            protected void TextBox4_TextChanged(object sender, EventArgs e)
            {

            }
        }
    }
```

StyleSheet1.css

```
body {
    background: #33CCCC;
    text-align: left;
    animation: color 5s infinite linear;
    padding: 2em;
}
p {
    font-family: "Lucida Console", "Brush Script MT", monospace;
}


.image1 {
    border: 1px red solid;
    object-position:right;
    max-width:150px;
}
.ButtonClass1 {
    color: blue;
    border: 1px solid yellow;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}

.ButtonClass2 {
    color: green;
    border: 1px solid green;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}

.ButtonClass3 {
    color: red;
    border: 1px solid red;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}

.ButtonClass4 {
    color: deeppink;
    border: 1px solid white;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}
@keyframes color {
    0% {
        background: #33CCCC;
    }

    20% {
        background: #33CC36;
    }

    40% {
        background: #B8CC33;
    }

    60% {
        background: #FCCA00;
    }

    80% {
        background: #33CC36;
    }

    100% {
        background: #33CCCC;
    }
}
```

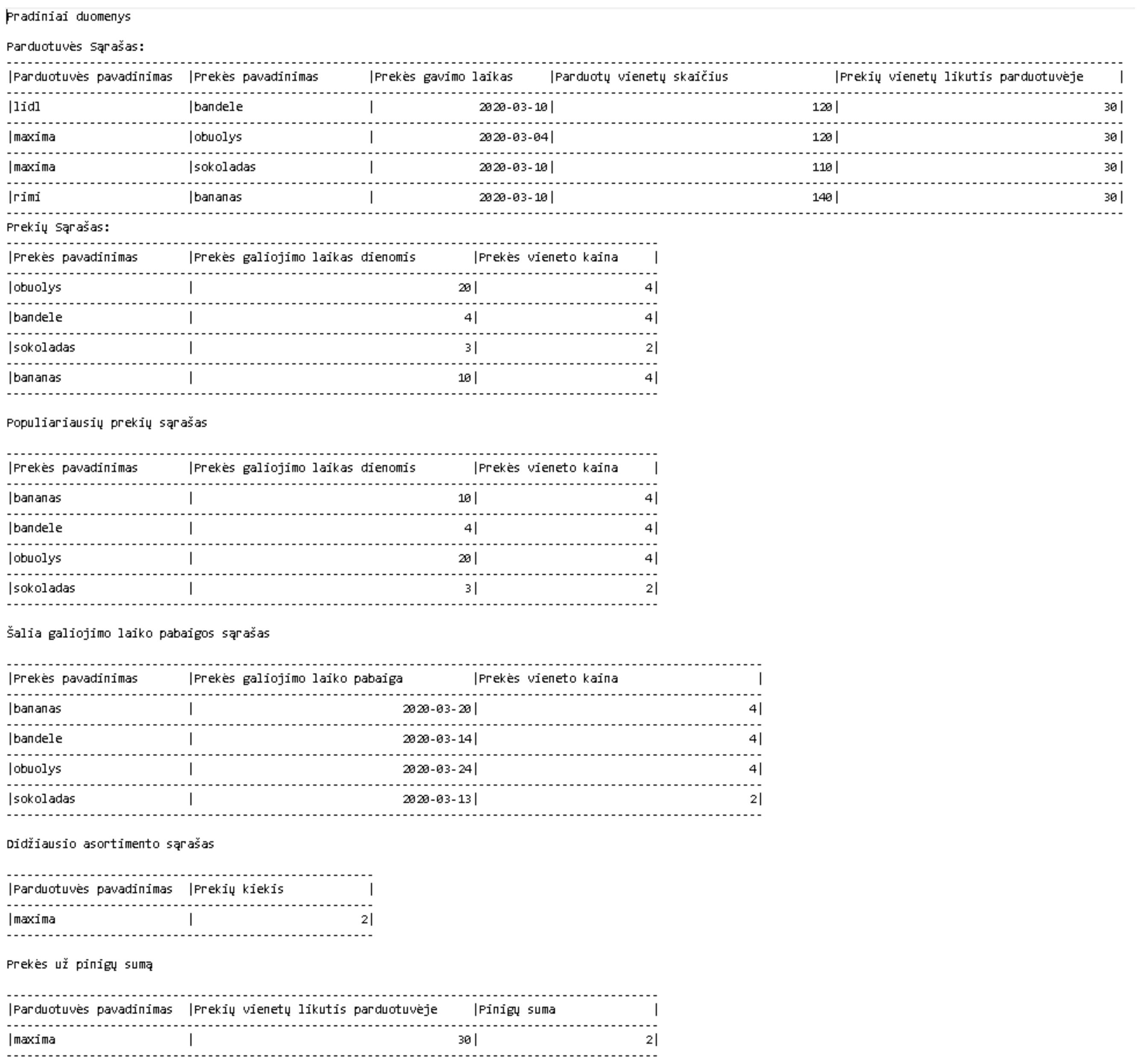
2.7. Pradiniai duomenys ir rezultatai

Pirmieji pradiniai duomenys

```
U19a.txt (Parduotuvės pradiniai duomenys)
lidl;bandele;2020/03/10;120;30
maxima;obuolys;2020/03/4;120;30
maxima;sokoladas;2020/03/10;110;30
rimi;bananas;2020/03/10;140;30

U19b.txt (Prekių pradiniai duomenys)
obuolys;20;4
bandele;4;4
sokoladas;3;2
bananas;10;4

Pinigų suma : 3
```

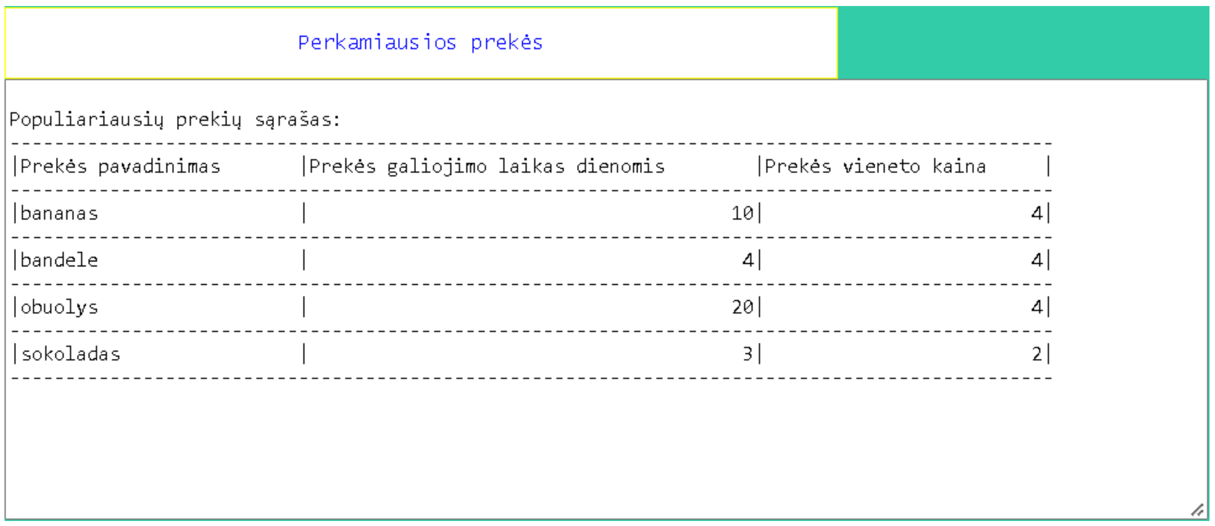
Rezultatai

```
Rezultatai.txt (Rezultatai)
```



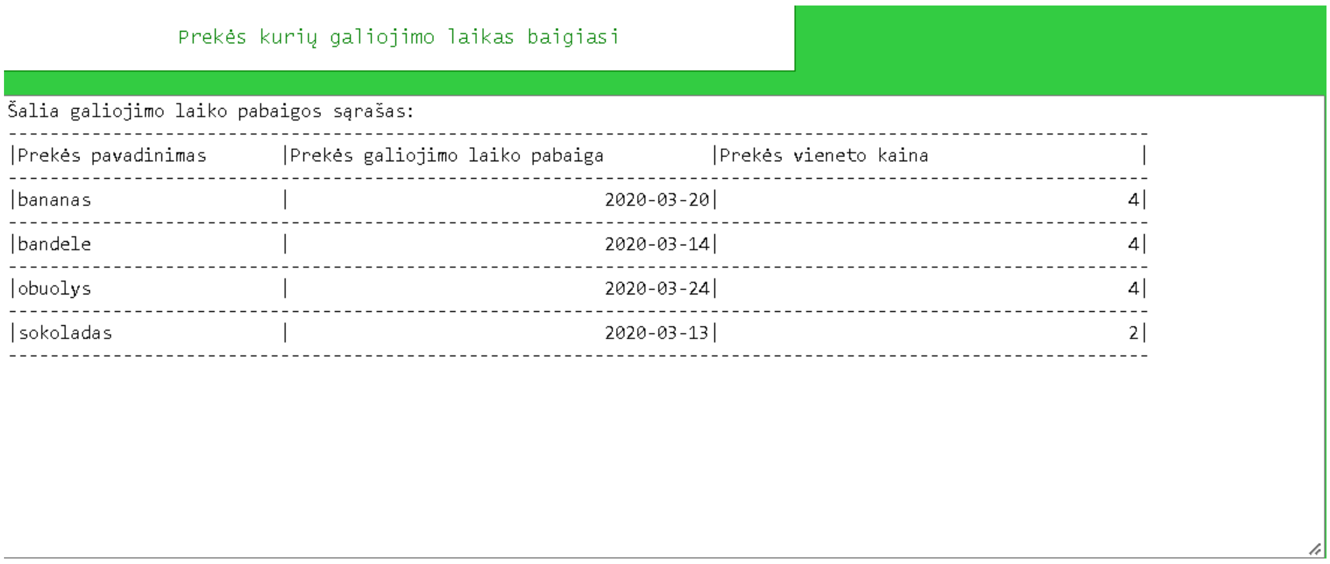
```
Pradiniai duomenys
Parduotuvės Sarašas:
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas  |Prekės pavadinimas        |Prekės gavimo laikas      |Parduotų vienetų skaičius                |Prekių vienetų likutis parduotuvėje     |
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |bandele                   |                2020-03-10|                                      120|                                      30|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                   |obuolys                   |                2020-03-04|                                      120|                                      30|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                   |sokoladas                 |                2020-03-10|                                      110|                                      30|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
|rimi                     |bananas                   |                2020-03-10|                                      140|                                      30|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
Prekių Sarašas:
-------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laikas dienomis       |Prekės vieneto kaina      |
-------------------------------------------------------------------------------------------
|obuolys                  |                                     20|                         4|
-------------------------------------------------------------------------------------------
|bandele                  |                                      4|                         4|
-------------------------------------------------------------------------------------------
|sokoladas                |                                      3|                         2|
-------------------------------------------------------------------------------------------
|bananas                  |                                     10|                         4|
-------------------------------------------------------------------------------------------

Populiariausių prekių sarašas

-------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laikas dienomis       |Prekės vieneto kaina      |
-------------------------------------------------------------------------------------------
|bananas                  |                                     10|                         4|
-------------------------------------------------------------------------------------------
|bandele                  |                                      4|                         4|
-------------------------------------------------------------------------------------------
|obuolys                  |                                     20|                         4|
-------------------------------------------------------------------------------------------
|sokoladas                |                                      3|                         2|
-------------------------------------------------------------------------------------------

Šalia galiojimo laiko pabaigos sarašas

---------------------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laiko pabaiga         |Prekės vieneto kaina                    |
---------------------------------------------------------------------------------------------------------
|bananas                  |                             2020-03-20|                                       4|
---------------------------------------------------------------------------------------------------------
|bandele                  |                             2020-03-14|                                       4|
---------------------------------------------------------------------------------------------------------
|obuolys                  |                             2020-03-24|                                       4|
---------------------------------------------------------------------------------------------------------
|sokoladas                |                             2020-03-13|                                       2|
---------------------------------------------------------------------------------------------------------

Didžiausio asortimento sarašas

---------------------------------------------------
|Parduotuvės pavadinimas  |Prekių kiekis           |
---------------------------------------------------
|maxima                   |                       2|
---------------------------------------------------

Prekės už pinigų sumą

-------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas  |Prekių vienetų likutis parduotuvėje    |Pinigų suma               |
-------------------------------------------------------------------------------------------
|maxima                   |                                     30|                         2|
-------------------------------------------------------------------------------------------
```

1 pav. Rezultatai.txt



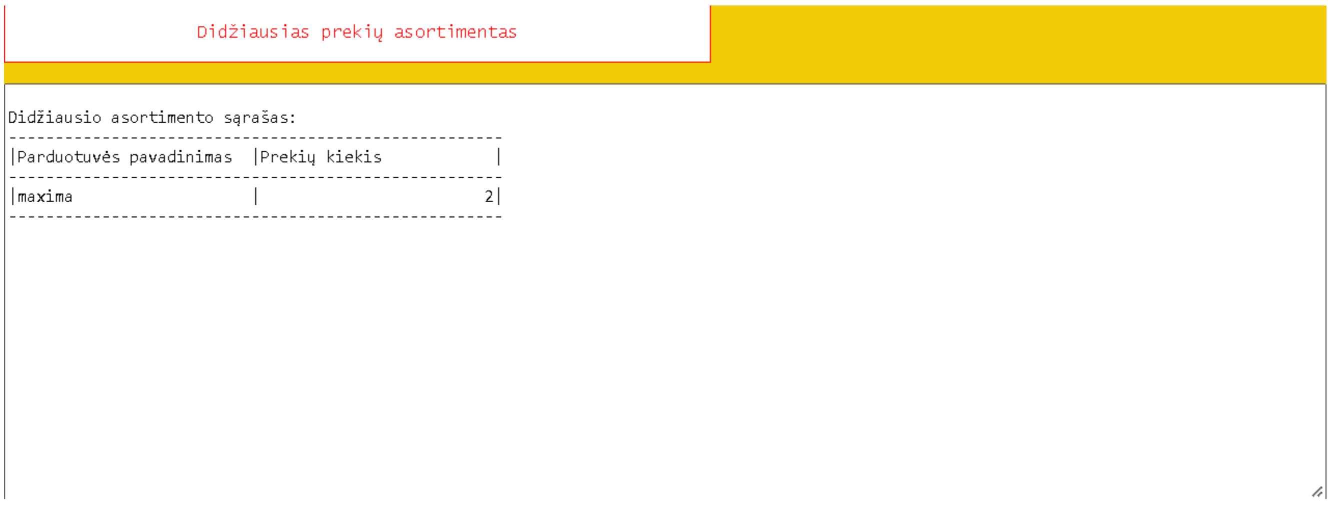
```
Populiariausių prekių sarašas:
-------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laikas dienomis       |Prekės vieneto kaina      |
-------------------------------------------------------------------------------------------
|bananas                  |                                     10|                         4|
-------------------------------------------------------------------------------------------
|bandele                  |                                      4|                         4|
-------------------------------------------------------------------------------------------
|obuolys                  |                                     20|                         4|
-------------------------------------------------------------------------------------------
|sokoladas                |                                      3|                         2|
-------------------------------------------------------------------------------------------
```

2 pav. Perkamiausios prekės rezultatai



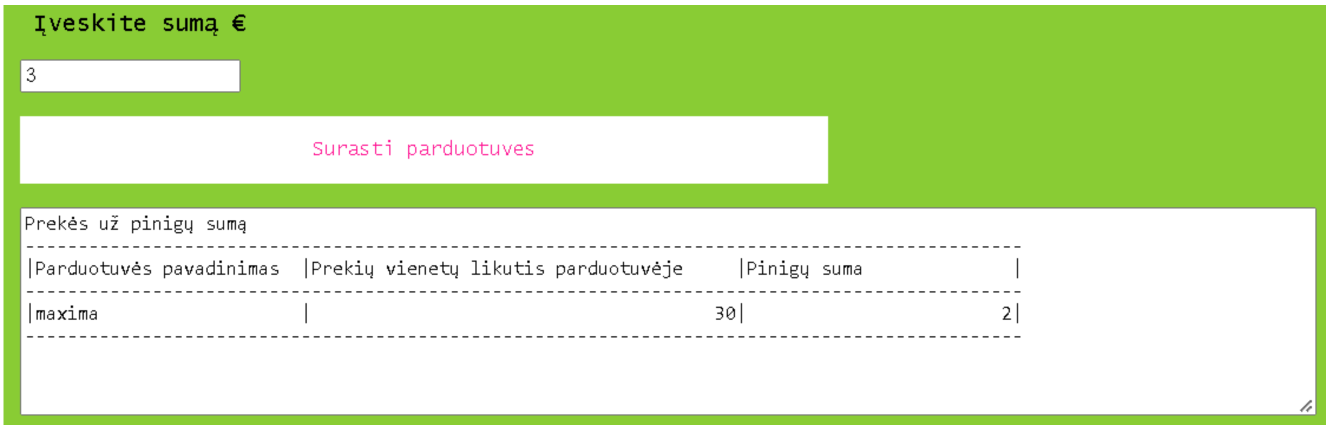
```
Šalia galiojimo laiko pabaigos sąrašas:
------------------------------------------------------------------------------------------------------------------------------
|Prekės pavadinimas          |Prekės galiojimo laiko pabaiga              |Prekės vieneto kaina                                  |
------------------------------------------------------------------------------------------------------------------------------
|bananas                     |                                  2020-03-20|                                                     4|
------------------------------------------------------------------------------------------------------------------------------
|bandele                     |                                  2020-03-14|                                                     4|
------------------------------------------------------------------------------------------------------------------------------
|obuolys                     |                                  2020-03-24|                                                     4|
------------------------------------------------------------------------------------------------------------------------------
|sokoladas                   |                                  2020-03-13|                                                     2|
------------------------------------------------------------------------------------------------------------------------------
```

3 pav. Galiojimo laiko rezultatai



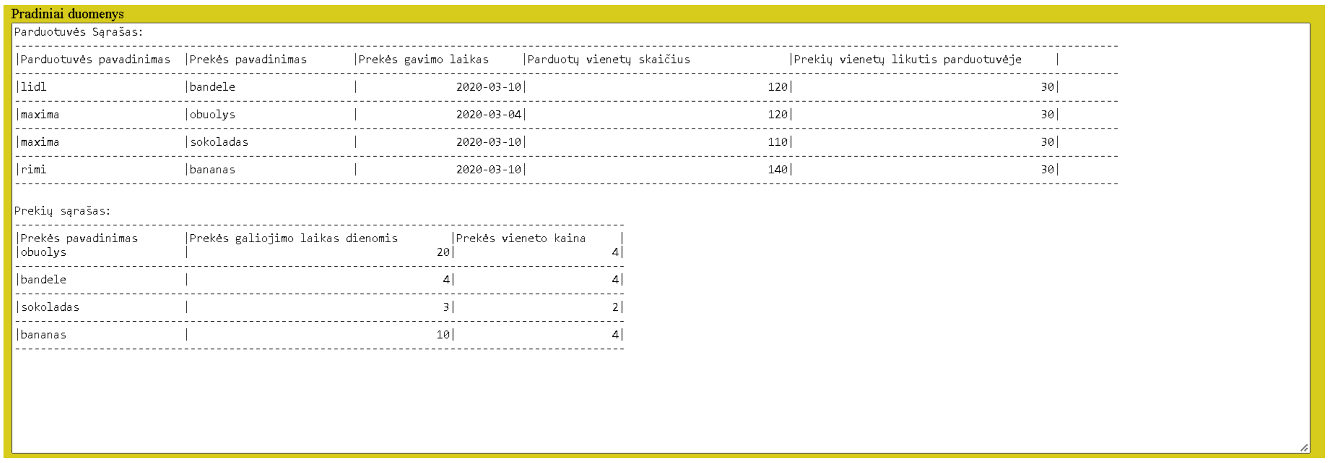
```
Didžiausio asortimento sąrašas:
------------------------------------------------------
|Parduotuvės pavadinimas   |Prekių kiekis           |
------------------------------------------------------
|maxima                    |                       2|
------------------------------------------------------
```

4 pav. Asortimentas



```
Prekės už pinigų sumą
--------------------------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas    |Prekių vienetų likutis parduotuvėje     |Pinigų suma                            |
--------------------------------------------------------------------------------------------------------------
|maxima                     |                                      30|                                      2|
--------------------------------------------------------------------------------------------------------------
```

5 pav. Prekės už pinigų sumą



Pradiniai duomenys

```
Parduotuvės Sąrašas:
------------------------------------------------------------------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas  |Prekės pavadinimas       |Prekės gavimo laikas     |Parduotų vienetų skaičius               |Prekių vienetų likutis parduotuvėje     |
------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |bandele                  |               2020-03-10|                                     120|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                   |obuolys                  |               2020-03-04|                                     120|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                   |sokoladas                |               2020-03-10|                                     110|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------
|rimi                     |bananas                  |               2020-03-10|                                     140|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------

Prekių sąrašas:
-------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laikas dienomis        |Prekės vieneto kaina    |
|obuolys                  |                                      20|                       4|
-------------------------------------------------------------------------------------------
|bandele                  |                                       4|                       4|
-------------------------------------------------------------------------------------------
|sokoladas                |                                       3|                       2|
-------------------------------------------------------------------------------------------
|bananas                  |                                      10|                       4|
-------------------------------------------------------------------------------------------
```

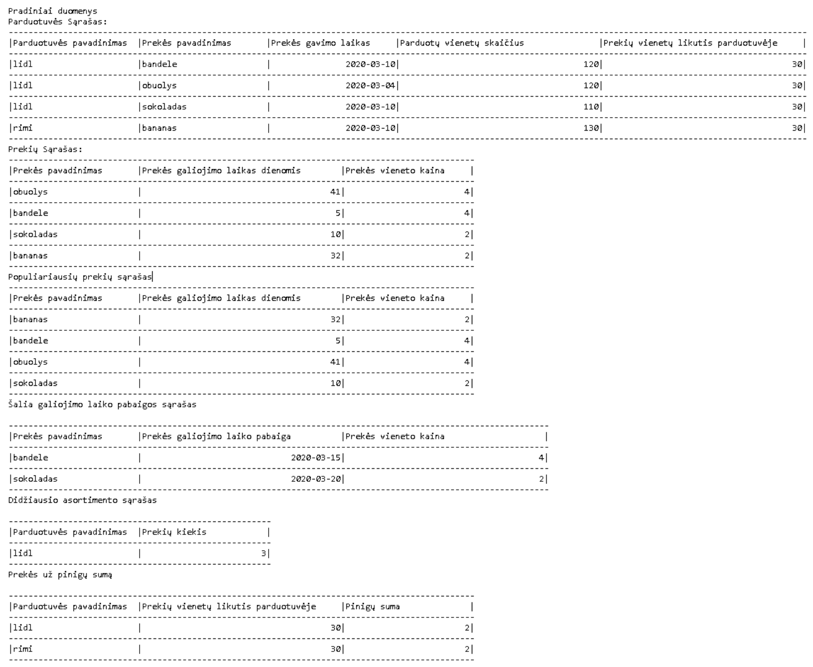
Pradiniai duomenys ekrane (pirmojo)

Antrieji pradiniai duomenys

```
U19a.txt (Parduotuvės pradiniai duomenys)
lidl;bandele;2020/03/10;120;30
lidl;obuolys;2020/03/4;120;30
lidl;sokoladas;2020/03/10;110;30
rimi;bananas;2020/03/10;130;30


U19b.txt (Prekių pradiniai duomenys)
obuolys;41;4
bandele;5;4
sokoladas;10;2
bananas;32;2

Pinigu suma : 3
```

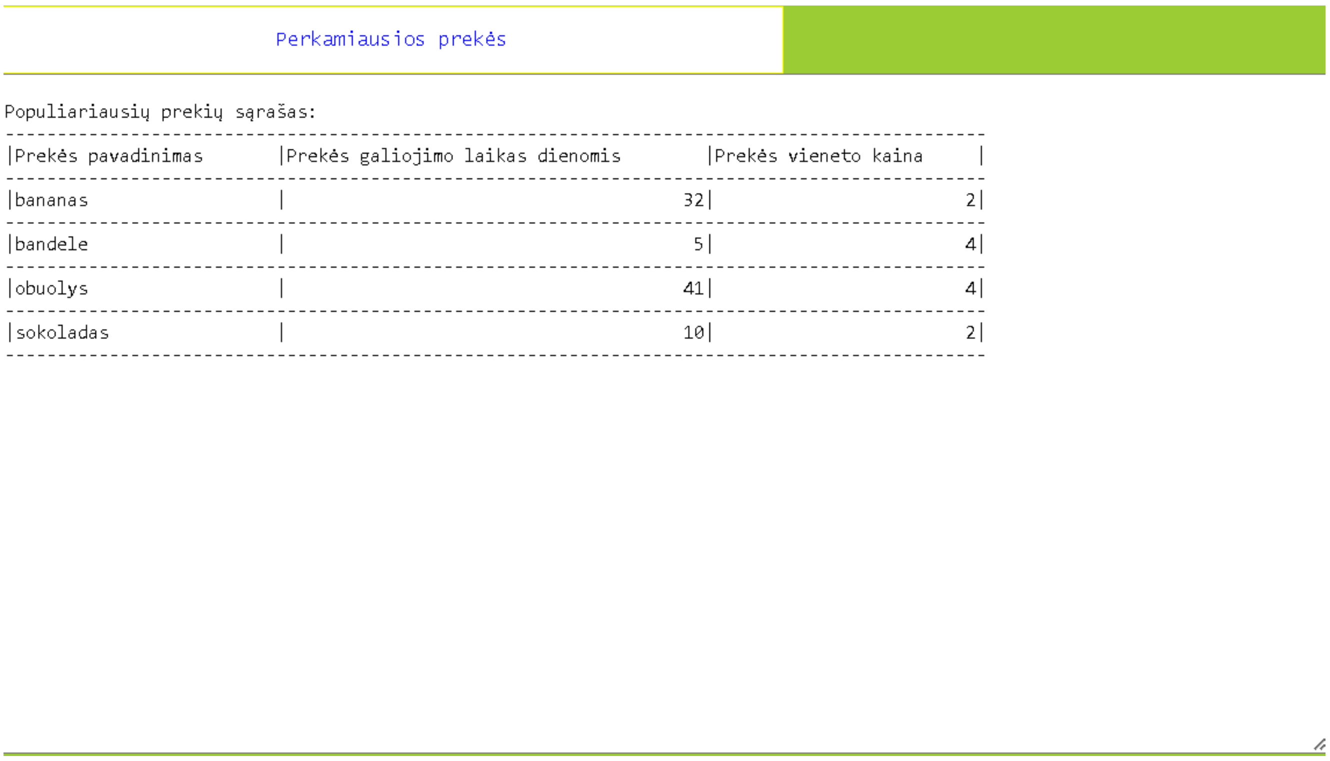
Rezultatai

Rezultatai.txt (Rezultatai)



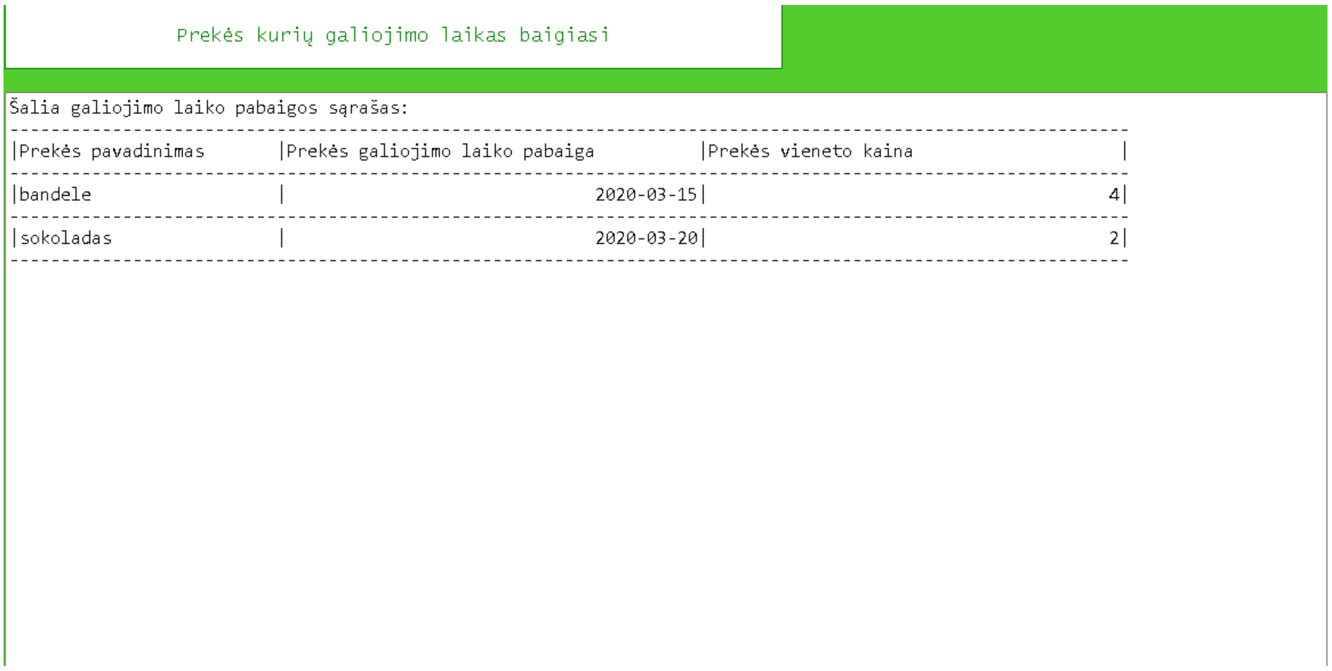
```
Pradiniai duomenys
Parduotuvės Sąrašas:
------------------------------------------------------------------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas  |Prekės pavadinimas       |Prekės gavimo laikas     |Parduotų vienetų skaičius               |Prekių vienetų likutis parduotuvėje     |
------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |bandele                  |               2020-03-10|                                     120|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |obuolys                  |               2020-03-04|                                     120|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |sokoladas                |               2020-03-10|                                     110|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------
|rimi                     |bananas                  |               2020-03-10|                                     130|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------
Prekių Sąrašas:
-------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laikas dienomis        |Prekės vieneto kaina    |
-------------------------------------------------------------------------------------------
|obuolys                  |                                      41|                       4|
-------------------------------------------------------------------------------------------
|bandele                  |                                       5|                       4|
-------------------------------------------------------------------------------------------
|sokoladas                |                                      10|                       2|
-------------------------------------------------------------------------------------------
|bananas                  |                                      32|                       2|
-------------------------------------------------------------------------------------------
Populiariausių prekių sąrašas
-------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laikas dienomis        |Prekės vieneto kaina    |
-------------------------------------------------------------------------------------------
|bananas                  |                                      32|                       2|
-------------------------------------------------------------------------------------------
|bandele                  |                                       5|                       4|
-------------------------------------------------------------------------------------------
|obuolys                  |                                      41|                       4|
-------------------------------------------------------------------------------------------
|sokoladas                |                                      10|                       2|
-------------------------------------------------------------------------------------------
Šalia galiojimo laiko pabaigos sąrašas

-----------------------------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laiko pabaiga          |Prekės vieneto kaina                    |
-----------------------------------------------------------------------------------------------------------
|bandele                  |                              2020-03-15|                                       4|
-----------------------------------------------------------------------------------------------------------
|sokoladas                |                              2020-03-20|                                       2|
-----------------------------------------------------------------------------------------------------------
Didžiausio asortimento sąrašas

----------------------------------------------------
|Parduotuvės pavadinimas  |Prekių kiekis            |
----------------------------------------------------
|lidl                     |                        3|
----------------------------------------------------
Prekės už pinigų sumą

-------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas  |Prekių vienetų likutis parduotuvėje     |Pinigų suma             |
-------------------------------------------------------------------------------------------
|lidl                     |                                      30|                       2|
-------------------------------------------------------------------------------------------
|rimi                     |                                      30|                       2|
-------------------------------------------------------------------------------------------
```

6 pav. Rezultatai.txt



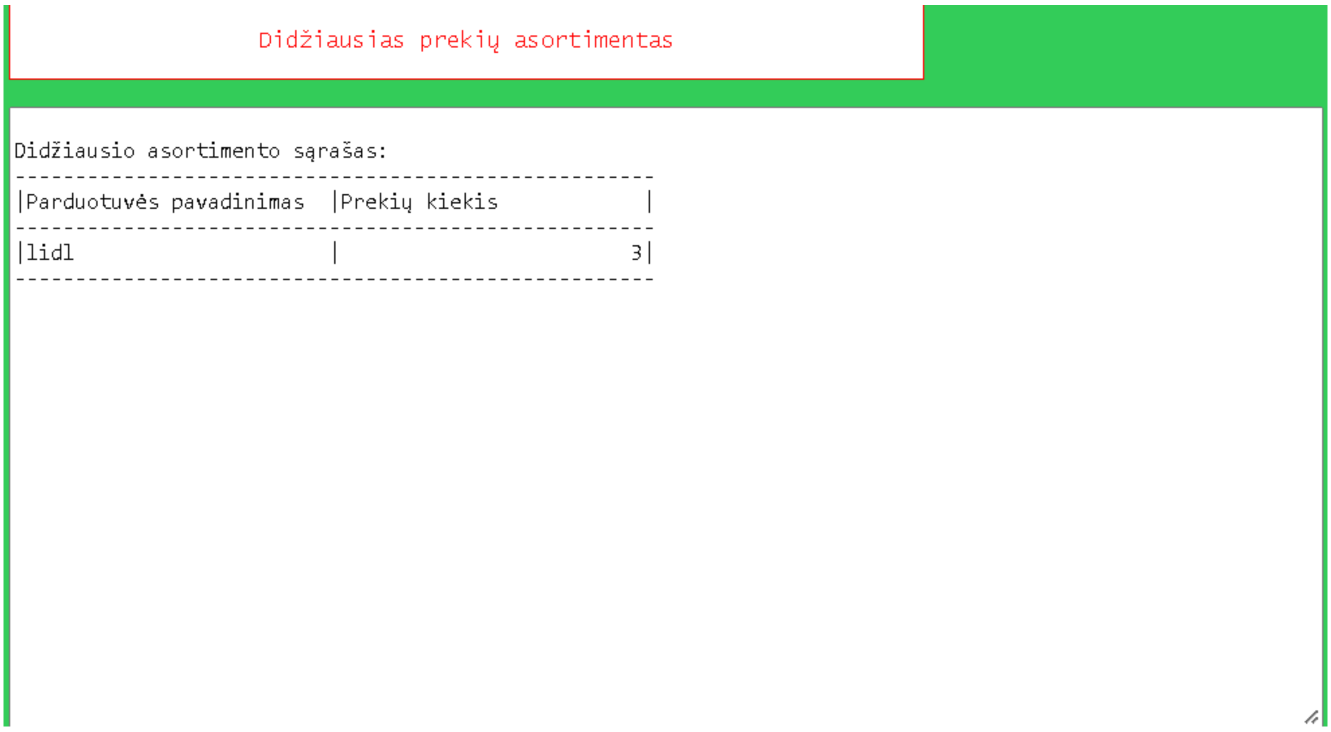
```
Perkamiausios prekės

Populiariausių prekių sąrašas:
----------------------------------------------------------------------------------------------------
|Prekės pavadinimas          |Prekės galiojimo laikas dienomis           |Prekės vieneto kaina         |
----------------------------------------------------------------------------------------------------
|bananas                     |                                         32|                            2|
----------------------------------------------------------------------------------------------------
|bandele                     |                                          5|                            4|
----------------------------------------------------------------------------------------------------
|obuolys                     |                                         41|                            4|
----------------------------------------------------------------------------------------------------
|sokoladas                   |                                         10|                            2|
----------------------------------------------------------------------------------------------------
```

7 pav. Perkamiausios prekės



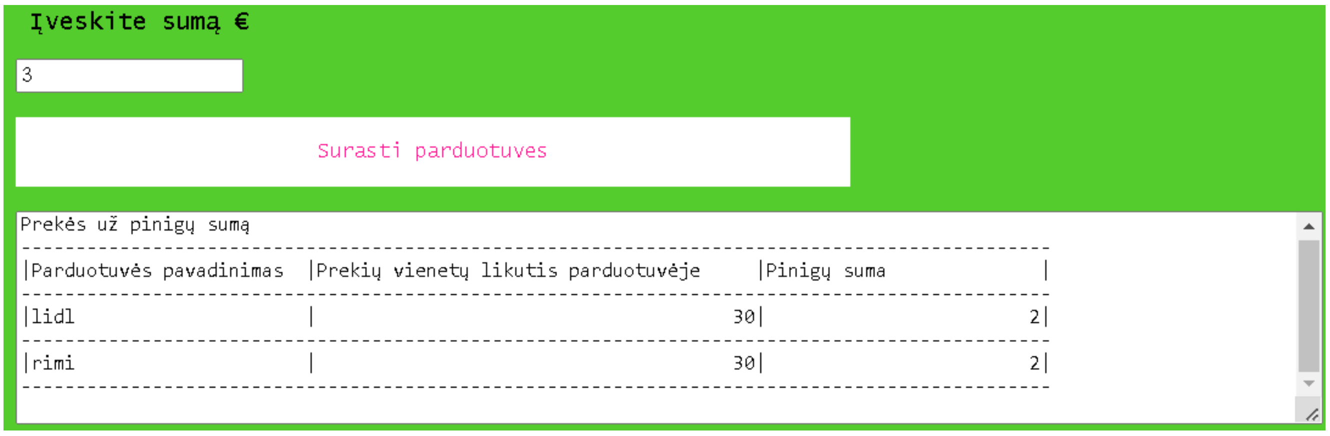
```
Prekės kurių galiojimo laikas baigiasi

Šalia galiojimo laiko pabaigos sąrašas:
--------------------------------------------------------------------------------------------------------------------
|Prekės pavadinimas          |Prekės galiojimo laiko pabaiga          |Prekės vieneto kaina                          |
--------------------------------------------------------------------------------------------------------------------
|bandele                     |                              2020-03-15|                                             4|
--------------------------------------------------------------------------------------------------------------------
|sokoladas                   |                              2020-03-20|                                             2|
--------------------------------------------------------------------------------------------------------------------
```

8 pav. Galiojimo laikas



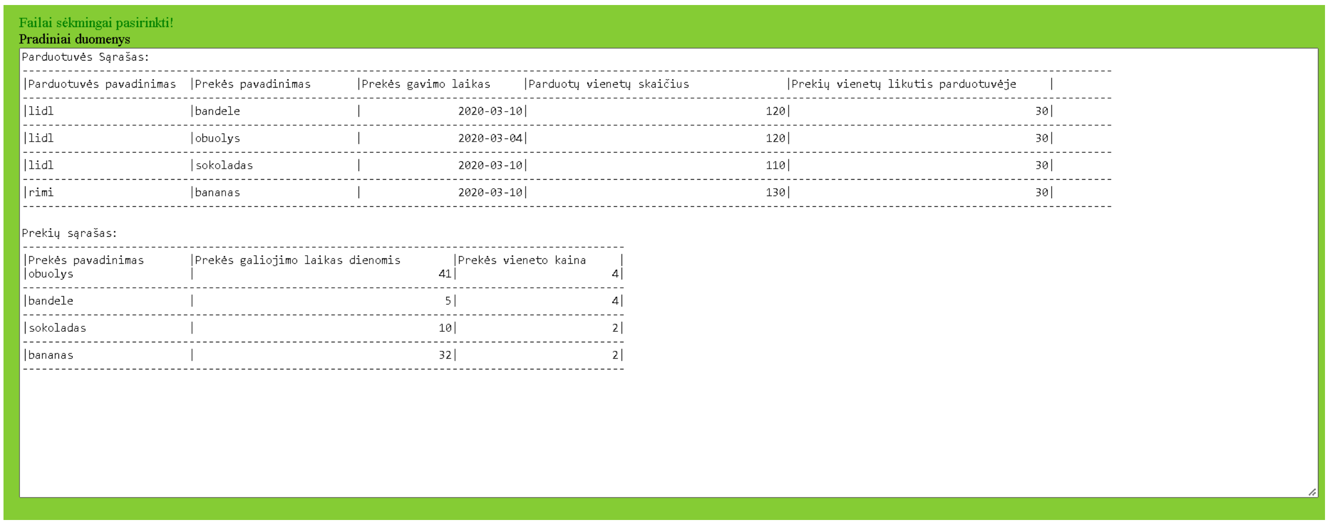
```
Didžiausio asortimento sąrašas:
------------------------------------------------------
|Parduotuvės pavadinimas  |Prekių kiekis             |
------------------------------------------------------
|lidl                     |                         3|
------------------------------------------------------
```

9 pav. Asortimentas



```
Prekės už pinigų sumą
----------------------------------------------------------------------------------------
|Parduotuvės pavadinimas  |Prekių vienetų likutis parduotuvėje    |Pinigų suma          |
----------------------------------------------------------------------------------------
|lidl                     |                                     30|                    2|
----------------------------------------------------------------------------------------
|rimi                     |                                     30|                    2|
----------------------------------------------------------------------------------------
```

10 pav. Prekės už pinigų sumą



```
Parduotuvės Sąrašas:
------------------------------------------------------------------------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas  |Prekės pavadinimas       |Prekės gavimo laikas     |Parduotų vienetų skaičius           |Prekių vienetų likutis parduotuvėje     |
------------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |bandele                  |               2020-03-10|                                 120|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |obuolys                  |               2020-03-04|                                 120|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                     |sokoladas                |               2020-03-10|                                 110|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------------
|rimi                     |bananas                  |               2020-03-10|                                 130|                                      30|
------------------------------------------------------------------------------------------------------------------------------------------------------------

Prekių sąrašas:
----------------------------------------------------------------------------------------
|Prekės pavadinimas       |Prekės galiojimo laikas dienomis      |Prekės vieneto kaina |
|obuolys                  |                                    41|                    4|
----------------------------------------------------------------------------------------
|bandele                  |                                     5|                    4|
----------------------------------------------------------------------------------------
|sokoladas                |                                    10|                    2|
----------------------------------------------------------------------------------------
|bananas                  |                                    32|                    2|
----------------------------------------------------------------------------------------
```

Pradiniai duomenys ekrane (antrasis)

2.8. Dėstytojo pastabos

Pradiniai duomenys spausdinami ekrane

3. Bendrinės klasės ir testavimas (L3)

3.1. Darbo užduotis

LD_19. Prekės. Prekybos tinklo parduotuvių rinkodaros skyriai atlieka prekių paklausos analizę. Sudarykite perkamiausių prekių sąrašą (prekės pavadinimas, prekės galiojimo laikas dienomis, prekės vieneto kaina). Sąrašas turi būti surikiuotas pagal prekių pavadinimus abėcėlės ir prekių vieneto kainas didėjimo tvarka. Sudarykite atskirą prekių, kurių galiojimo laikas baigiasi ne anksčiau kaip po 30 dienų nuo esamos datos, sąrašą (prekės pavadinimas, prekės galiojimo laiko pabaiga (metai, mėnuo, diena), prekės vieneto kaina). Sąrašas turi būti surikiuotas pagal prekių pavadinimus abėcėlės ir prekių vieneto kainas didėjimo tvarka. Suraskite, kurios parduotuvės rinkodaros skyrius turi didžiausią prekių asortimentą.

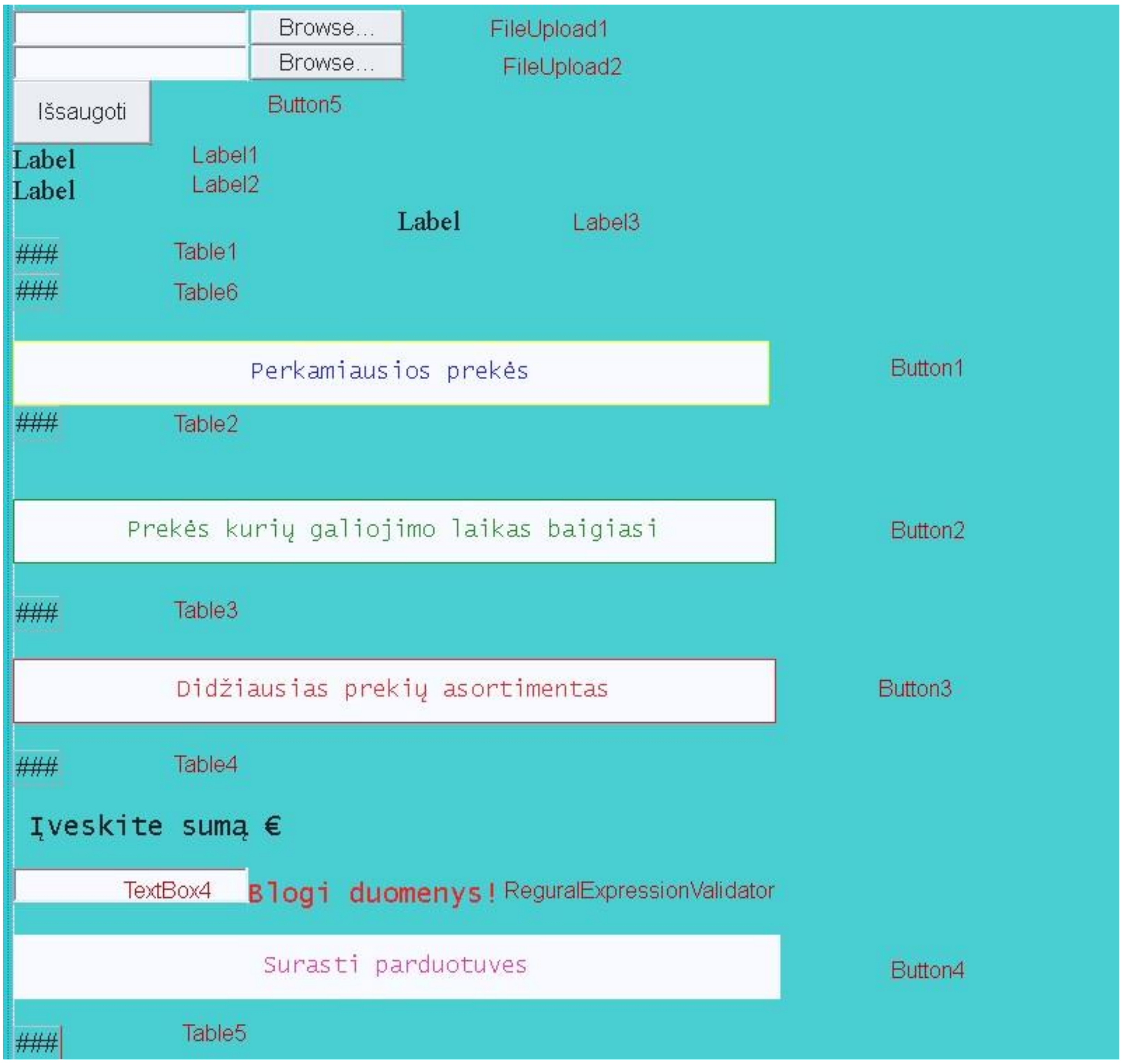
Duomenys:

• tekstiniame faile U19a.txt yra informacija apie parduotuves: parduotuvės pavadinimas, prekės pavadinimas, prekės gavimo laikas (metai, mėnuo, diena), parduotų prekių vienetų skaičius, prekių vienetų likutis parduotuvėje;

• tekstiniame faile U19b.txt yra informacija apie prekes: prekės pavadinimas, prekės galiojimo laikas dienomis, prekės vieneto kaina.

Sudarykite parduotuvių, kurios turi prekių už pinigų sumą, ne didesnę už nurodytą (įvedama klaviatūra), sąrašą (parduotuvės pavadinimas, prekių vienetų likutis parduotuvėje, pinigų suma).

3.2. Grafinės vartotojo sąsajos schema

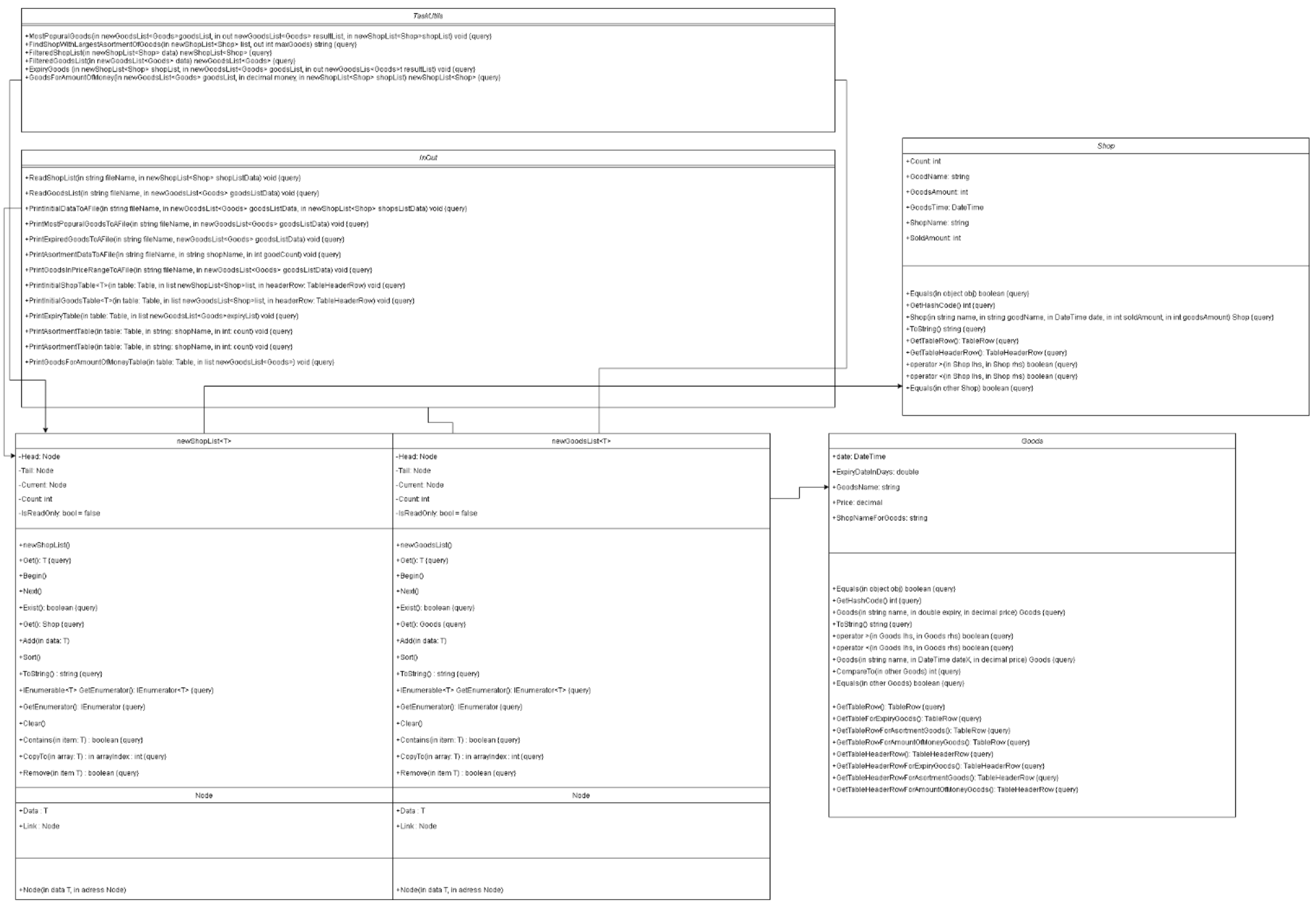


3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
FileUpload1	Niekas nepakeista	
FileUpload2	Niekas nepakeista	
Button5	Text	„Išsaugoti‘‘
Label1	Visible	false
Label2	Visible	false
Button1	Text	„Perkamiausios prekės‘‘
Button2	Text	„Prekės kurių galiojimo laikas baigiasi‘‘
Button3	Text	Didžiausias prekių asortimentas

Label3	Text	,,Įveskite sumą €‘‘
TextBox4	Niekas nepakeista	
ReguralExpressionValidator	ErrorMessage, ControlToValidate, ValidationGroup, ForeColor	,,Blogi duomenys!‘‘,TextBox4,^-?(([1-9]\d*)|0)(.0*[1-9](0*[1-9])*)?$, Red
Button4	Text	,,Surasti parduotuves‘‘
Table1	Visible. BorderStyle, GridLines, ForeColor	False, Solid, Both, black
Table2	Visible. BorderStyle, GridLines, ForeColor	False, Solid, Both, black
Table3	Visible. BorderStyle, GridLines, ForeColor	False, Solid, Both, black
Table4	Visible. BorderStyle, GridLines, ForeColor	False, Solid, Both, black
Table5	Visible. BorderStyle, GridLines, ForeColor	False, Solid, Both, black
Table6	Visible. BorderStyle, GridLines, ForeColor	False, Solid, Both, black

3.4. Klasių diagrama


TaskUtils
+MostPopuralGoods(in newGoodsList<Goods>goodsList, in out newGoodsList<Goods> resultList, in newShopList<Shop>shopList) void {query}
+FindShopWithLargestAsortmentOfGoods(in newShopList<Shop> list, out int maxGoods) string {query}
+FilteredShopList(in newShopList<Shop> data) newShopList<Shop> {query}
+FilteredGoodsList(in newGoodsList<Goods> data) newGoodsList<Goods> {query}
+ExpiryGoods (in newShopList<Shop> shopList, in newGoodsList<Goods> goodsList, in out newGoodsLis<Goods>t resultList) void {query}
+GoodsForAmountOfMoney(in newGoodsList<Goods> goodsList, in decimal money, in newShopList<Shop> shopList) newShopList<Shop> {query}
InOut
+ReadShopList(in string fileName, in newShopList<Shop> shopListData) void {query}
+ReadGoodsList(in string fileName, in newGoodsList<Goods> goodsListData) void {query}
+PrintInitialDataToAFile(in string fileName, in newGoodsList<Goods> goodsListData, in newShopList<Shop> shopsListData) void {query}
+PrintMostPopuralGoodsToAFile(in string fileName, in newGoodsList<Goods> goodsListData) void {query}
+PrintExpiredGoodsToAFile(in string fileName, newGoodsList<Goods> goodsListData) void {query}
+PrintAsortmentDataToAFile(in string fileName, in string shopName, in int goodCount) void {query}
+PrintGoodsInPriceRangeToAFile(in string fileName, in newGoodsList<Goods> goodsListData) void {query}
+PrintInitialShopTable<T>(in table: Table, in list newShopList<Shop>list, in headerRow: TableHeaderRow) void {query}
+PrintInitialGoodsTable<T>(in table: Table, in list newGoodsList<Shop>list, in headerRow: TableHeaderRow) void {query}
+PrintExpiryTable(in table: Table, in list newGoodsList<Goods>expiryList) void {query}
+PrintAsortmentTable(in table: Table, in string: shopName, in int: count) void {query}
+PrintAsortmentTable(in table: Table, in string: shopName, in int: count) void {query}
+PrintGoodsForAmountOfMoneyTable(in table: Table, in list newGoodsList<Goods>) void {query}
Shop
+Count: int
+GoodName: string
+GoodsAmount: int
+GoodsTime: DateTime
+ShopName: string
+SoldAmount: int
+Equals(in object obj) boolean {query}
+GetHashCode() int {query}
+Shop(in string name, in string goodName, in DateTime date, in int soldAmount, in int goodsAmount) Shop {query}
+ToString() string {query}
+GetTableRow(): TableRow {query}
+GetTableHeaderRow(): TableHeaderRow {query}
+operator >(in Shop lhs, in Shop rhs) boolean {query}
+operator <(in Shop lhs, in Shop rhs) boolean {query}
+Equals(in other Shop) boolean {query}
newShopList<T>
-Head: Node
-Tail: Node
-Current: Node
-Count: int
-IsReadOnly: bool = false
+newShopList()
+Get(): T {query}
+Begin()
+Next()
+Exist(): boolean {query}
+Get(): Shop {query}
+Add(in data: T)
+Sort()
+ToString() : string {query}
+IEnumerable<T> GetEnumerator(): IEnumerator<T> {query}
+GetEnumerator(): IEnumerator {query}
+Clear()
+Contains(in item: T) : boolean {query}
+CopyTo(in array: T) : in arrayIndex : int {query}
+Remove(in item T) : boolean {query}
Node
+Data : T
+Link : Node
+Node(in data T, in adress Node)
newGoodsList<T>
-Head: Node
-Tail: Node
-Current: Node
-Count: int
-IsReadOnly: bool = false
+newGoodsList()
+Get(): T {query}
+Begin()
+Next()
+Exist(): boolean {query}
+Get(): Goods {query}
+Add(in data: T)
+Sort()
+ToString() : string {query}
+IEnumerable<T> GetEnumerator(): IEnumerator<T> {query}
+GetEnumerator(): IEnumerator {query}
+Clear()
+Contains(in item: T) : boolean {query}
+CopyTo(in array: T) : in arrayIndex : int {query}
+Remove(in item T) : boolean {query}
Node
+Data : T
+Link : Node
+Node(in data T, in adress Node)
Goods
+date: DateTime
+ExpiryDateInDays: double
+GoodsName: string
+Price: decimal
+ShopNameForGoods: string
+Equals(in object obj) boolean {query}
+GetHashCode() int {query}
+Goods(in string name, in double expiry, in decimal price) Goods {query}
+ToString() string {query}
+operator >(in Goods lhs, in Goods rhs) boolean {query}
+operator <(in Goods lhs, in Goods rhs) boolean {query}
+Goods(in string name, in DateTime dateX, in decimal price) Goods {query}
+CompareTo(in other Goods) int {query}
+Equals(in other Goods) boolean {query}
+GetTableRow(): TableRow {query}
+GetTableForExpiryGoods(): TableRow {query}
+GetTableRowForAsortmentGoods(): TableRow {query}
+GetTableRowForAmountOfMoneyGoods(): TableRow {query}
+GetTableHeaderRow(): TableHeaderRow {query}
+GetTableHeaderRowForExpiryGoods(): TableHeaderRow {query}
+GetTableHeaderRowForAsortmentGoods(): TableHeaderRow {query}
+GetTableHeaderRowForAmountOfMoneyGoods(): TableHeaderRow {query}

3.5. Programos vartotojo vadovas

Pradžioje pasirenkame du pradinių duomenų failus su parduotuvės ir prekių informacija. Pasirinke šiuos failus spaudžiame mygtuką „Išsaugoti‘‘ ir taip išsaugome šiuos duomenų failus į sesiją. Visus kitus rezultatus gauname paspaudus mygtukus „Perkamiausios prekės‘‘ (Perkamiausių prekių rezultatai) , „Prekių, kurių galiojimo laikas baigiasi‘‘ (Prekių, kurių galiojimo laikas baigiasi rezultatai), „Didžiausias prekių asortimentas‘‘ (Didžiausio asortimento rezultatai) ir prieš spaudžiant „Surasti parduotuves‘‘ mygtuką privalome įvesti pinigų sumą į textbox elementą ir tik tada spausti mygtuką, taip išvedami (Prekės už pinigų sumą rezultatai). Viskas spausdinama lentelėmis į ekraną. (Table elementus).

3.6. Programos tekstas

Shop.cs

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

namespace L2
{
    /// <summary>
    /// Data class for shop information
    /// </summary>
    [Serializable]
    public class Shop : IComparable<Shop>, IEquatable<Shop>, ITableHeaders
    {
        public string ShopName { get; set; }
        public string GoodName { get; set; }
        public DateTime GoodsTime { get; set; }
        public int SoldAmount { get; set; }
        public int GoodsAmount { get; set; }
        public int Count;

        /// <summary>
        /// Shop constructor for initial data
        /// </summary>
        /// <param name="name">Shop name</param>
        /// <param name="goodName">Goods name</param>
        /// <param name="goodTime">Date when good were received</param>
        /// <param name="soldAmount">Sold amount of goods</param>
        /// <param name="goodsAmount">Remaining amount of goods</param>
        public Shop (string name, string goodName, DateTime goodTime, int
soldAmount, int goodsAmount)
        {
            this.ShopName = name;
            this.GoodName = goodName;
            this.GoodsTime = goodTime;
            this.SoldAmount = soldAmount;
            this.GoodsAmount = goodsAmount;
        }

        /// <summary>
        /// Equals method for finding the same objects
        /// </summary>
        /// <param name="obj">other object</param>
        /// <returns>same objects</returns>
        public override bool Equals(object obj)
        {
            if ((obj is null) || (Goods)obj is null)
                return false;
            return Equals((Goods)obj);

        }


        /// <summary>
        /// Method for generating a tablerow with initial shop data
        /// </summary>
        /// <returns>table row with initial shop data</returns>
        public TableRow GetTableRow()
        {
            TableRow row = new TableRow();
            TableCell shopCell = new TableCell { Text = ShopName.ToString() };
            TableCell goodCell = new TableCell { Text = GoodName.ToString() };
            TableCell dateCell= new TableCell
            {
                Text = string.Format("{0,25:yyyy/MM/dd}", GoodsTime),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell soldCell = new TableCell
            {
                Text = SoldAmount.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell amountCell = new TableCell
            {
                Text = GoodsAmount.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };

            row.Cells.Add(shopCell);
            row.Cells.Add(goodCell);
            row.Cells.Add(dateCell);
            row.Cells.Add(soldCell);
            row.Cells.Add(amountCell);
            return row;
        }

        /// <summary>
        /// Method for generating a table header row for initial shop data
        /// </summary>
        /// <returns>header row for initial shop data</returns>
        public static TableHeaderRow GetInitialTableHeader()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell shopCell
            = new TableHeaderCell { Text = "Parduotuvės pavadinimas" };
            TableHeaderCell goodCell
            = new TableHeaderCell { Text = "Prekės pavadinimas" };
            TableHeaderCell dateCell
            = new TableHeaderCell
            { Text = "Prekių gavimo data" };
            TableHeaderCell soldCell
            = new TableHeaderCell
            { Text = "Parduotų prekių kiekis" };
            TableHeaderCell amountCell
            = new TableHeaderCell
            { Text = "Prekių likutis" };
            row.Cells.Add(shopCell);
```

```
            row.Cells.Add(goodCell);
            row.Cells.Add(dateCell);
            row.Cells.Add(soldCell);
            row.Cells.Add(amountCell);
            return row;
        }

        public override int GetHashCode()
        {
            return ShopName.GetHashCode() ^ GoodName.GetHashCode() ^
GoodsTime.GetHashCode() ^ SoldAmount.GetHashCode() ^ GoodsAmount.GetHashCode();
        }

        /// <summary>
        /// Compare to method to compare goods soldamount
        /// </summary>
        /// <param name="other"></param>
        /// <returns></returns>
        public int CompareTo(Shop other)
        {
            if (SoldAmount.CompareTo(other.SoldAmount) == 0)
                return 1;
            return SoldAmount.CompareTo(other.SoldAmount);
        }

        /// <summary>
        /// Equals method for finding the same objects
        /// </summary>
        /// <param name="other">other object</param>
        /// <returns>the same objects</returns>
        public bool Equals(Shop other)
        {
            if (other is null)
                return false;
            return ShopName == other.ShopName &&
            GoodName == other.GoodName &&
            GoodsTime == other.GoodsTime &&
            SoldAmount == other.SoldAmount && GoodsAmount == other.GoodsAmount;
        }


        public override string ToString()
        {
            return string.Format("|{0,-25}|{1,-
25}|{2,25:yyyy/MM/dd}|{3,40}|{4,40}|", ShopName, GoodName, GoodsTime, SoldAmount,
GoodsAmount);
        }
    }
}
```

ShopTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Linq;
using L2;
using FluentAssertions;



namespace UnitTestProject1
{
    [TestClass]
    public class ShopTest
    {
        [TestMethod]
        public void Shop_paramsCtorTest()
        {
            Shop shop = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 20, 50);
            shop.ShopName.Should().Be("Maxima");
            shop.GoodName.Should().Be("Obuolys");
            shop.GoodsTime.Should().Be(DateTime.Parse("2020/03/20"));
            shop.SoldAmount.Should().Be(20);
            shop.GoodsAmount.Should().Be(50);
        }



        [TestMethod()]
        public void TwoStringEqualLength()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            Shop shop2 = new Shop("Maxima", "Bananas",
DateTime.Parse("2020/03/20"), 500, 252525);
            Assert.AreEqual(shop1.ToString().Length, shop2.ToString().Length);

        }

        [TestMethod()]
        public void StringContainsShopName()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            string[] parts = shop1.ToString().Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("Maxima");

        }

        [TestMethod()]
        public void StringContainsGoodsName()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            string[] parts = shop1.ToString().Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("Obuolys");


        }

        [TestMethod()]
        public void StringContainsDate()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/02/20"), 100, 50);
            string[] parts = shop1.ToString().Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("2020","02","20");
        }

        [TestMethod()]
        public void StringContainsSoldAmount()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            string[] parts = shop1.ToString().Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("100");
        }

        [TestMethod()]
        public void StringContainsGoodsAmount()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            string[] parts = shop1.ToString().Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("50");
        }

        [TestMethod()]
        public void Equals_NameParameterIsTheSame_True()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            Shop shop2 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            shop1.Equals(shop2).Should().BeTrue();
        }

        [TestMethod()]
        public void Equals_NameParameterIsNotTheSame_False()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
25, 10);
            shop1.Equals(shop2).Should().BeFalse();
        }

        [TestMethod()]
        public void CompareTo_SoldAmount_Shop1MoreThanShop2()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
25, 10);
            shop1.CompareTo(shop2).Should().BeGreaterThan(0);

        }

        [TestMethod()]
        public void CompareTo_SoldAmount_Shop2MoreThanShop1()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
115, 10);
            shop1.CompareTo(shop2).Should().BeLessThan(0);
        }

        [TestMethod()]
        public void CompareTo_SoldAmount_Same()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            Shop shop2 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 100, 50);
            shop1.CompareTo(shop2).Should().Be(0);
        }

        [TestMethod()]
        public void OperatorLT_SoldAmount_True()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 115, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
100, 10);
            Assert.IsTrue(shop1 > shop2);
        }

        [TestMethod()]
        public void OperatorLT_SoldAmount_False()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 115, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
116, 10);
            Assert.IsFalse(shop1 > shop2);
        }

        [TestMethod()]
        public void OperatorHT_SoldAmount_True()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 115, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
116, 10);
            Assert.IsTrue(shop1 < shop2);
        }

        [TestMethod()]
        public void OperatorHT_SoldAmount_False()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 120, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
116, 10);
            Assert.IsFalse(shop1 < shop2);
        }

        [TestMethod()]
        public void OpEqual_SoldAmount_False()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 120, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
116, 10);
            Assert.IsFalse(shop1 == shop2);
        }

        [TestMethod()]
        public void OpNotEqual_SoldAmount_True()
        {
            Shop shop1 = new Shop("Maxima", "Obuolys",
DateTime.Parse("2020/03/20"), 120, 50);
            Shop shop2 = new Shop("Iki", "Bananas", DateTime.Parse("2020/03/20"),
116, 10);
            Assert.IsTrue(shop1 != shop2);
        }


    }
}
```

Goods.cs

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

namespace L2
{
    /// <summary>
    /// Data class for Goods information
    /// </summary>
    [Serializable]
    public class Goods : IComparable<Goods>, IEquatable<Goods>, ITableHeaders
    {
        public string GoodsName { get; set; }
        public double ExpiryDateinDays { get; set; }
        public decimal Price { get; set; }
        public DateTime date { get; set; }
        public string ShopNameForGoods { get; set; }
        public int Count { get; set; }
        public string shopName;

        /// <summary>
        /// Constructor for Goods initial data
        /// </summary>
        /// <param name="name">Goods name/param>
        /// <param name="expiry">Expiry date in days</param>
        /// <param name="price">Goods price</param>
        public Goods (string name, double expiry, decimal price)
        {
            this.GoodsName = name;
            this.ExpiryDateinDays = expiry;
            this.Price = price;
        }
        /// <summary>
        /// Constructor for Goods data
        /// </summary>
        /// <param name="name">Goods name</param>
        /// <param name="dateX">Date</param>
        /// <param name="price">Price</param>
        public Goods(string name, DateTime dateX, decimal price)
        {
            this.GoodsName = name;
            this.date = dateX;
            this.Price = price;
        }
        /// <summary>
        /// Empty constructor for data
        /// </summary>
        public Goods()
        {


        }

        public static bool operator >(Goods lhs, Goods rhs)
        {
            return lhs.GoodsName.CompareTo(rhs.GoodsName) > 0 ||
lhs.GoodsName.CompareTo(rhs.GoodsName) == 0 && lhs.Price.CompareTo(rhs.Price) > 0;
        }


        public static bool operator <(Goods lhs, Goods rhs)
        {
            return lhs.GoodsName.CompareTo(rhs.GoodsName) < 0 ||
lhs.GoodsName.CompareTo(rhs.GoodsName) == 0 && lhs.Price.CompareTo(rhs.Price) < 0;
        }

        public static bool operator ==(Goods lhs, Goods rhs)
        {
            return lhs.Equals(rhs);
        }

        public static bool operator != (Goods lhs, Goods rhs)
        {
            return !lhs.Equals(rhs);
        }


        /// <summary>
        /// Method for generating initial tablerow data
        /// </summary>
        /// <returns>a table row with initial goods data</returns>
        public TableRow GetTableRow()
        {
            TableRow row = new TableRow();
            TableCell goodNameCell = new TableCell { Text = GoodsName.ToString()
};
            TableCell expiryDateCell = new TableCell
            {
                Text = ExpiryDateinDays.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell priceCell = new TableCell
            {
                Text = Price.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
```

```
    row.Cells.Add(goodNameCell);
    row.Cells.Add(expiryDateCell);
    row.Cells.Add(priceCell);

    return row;
}

/// <summary>
/// Method for generating table with expiry goods data
/// </summary>
/// <returns>a table row with expiry goods data</returns>
public TableRow GetTableRowForExpiryGoods()
{
    TableRow row = new TableRow();
    TableCell goodCell = new TableCell { Text = GoodsName.ToString() };
    TableCell dateCell = new TableCell
    {
        Text = string.Format("{0,25:yyyy/MM/dd}", date),
        HorizontalAlign = HorizontalAlign.Right
    };
    TableCell priceCell = new TableCell
    {
        Text = Price.ToString(),
        HorizontalAlign = HorizontalAlign.Right
    };
    row.Cells.Add(goodCell);
    row.Cells.Add(dateCell);
    row.Cells.Add(priceCell);
    return row;
}

/// <summary>
/// Method for generating a table row with asortment goods data
/// </summary>
/// <param name="shopName">the name of the shop</param>
/// <param name="count">the asortment goods amount</param>
/// <returns>a tablerow with asortment goods data</returns>
public TableRow GetTableRowForAsortmentGoods(string shopName, int count)
{
    TableRow row = new TableRow();
    TableCell shopNameCell = new TableCell { Text = shopName.ToString() };
    TableCell asortmentCountCell = new TableCell
    {
        Text = count.ToString(),
        HorizontalAlign = HorizontalAlign.Right
    };
    row.Cells.Add(shopNameCell);
    row.Cells.Add(asortmentCountCell);
    return row;
}
/// <summary>
/// Method for generating a table row with goods for amount of money data
/// </summary>
/// <returns>a table </returns>
public TableRow GetTableRowForAmountOfMoneyGoods()
{
    TableRow row = new TableRow();
    TableCell shopCell = new TableCell { Text = GoodsName.ToString() };
    TableCell amountCell = new TableCell
    {
        Text = ExpiryDateinDays.ToString(),
        HorizontalAlign = HorizontalAlign.Right
    };
    TableCell priceCell = new TableCell
    {
        Text = Price.ToString(),
        HorizontalAlign = HorizontalAlign.Right
    };
    row.Cells.Add(shopCell);
    row.Cells.Add(amountCell);
    row.Cells.Add(priceCell);
    return row;
}
/// <summary>
/// Method for generating initial tableheader row data
/// </summary>
/// <returns>initial table row data</returns>
public static TableHeaderRow GetInitialTableHeaderRow()
{
    TableHeaderRow row = new TableHeaderRow();
    TableHeaderCell goodName
    = new TableHeaderCell { Text = "Prekės pavadinimas" };
    TableHeaderCell expiryDateInDays
    = new TableHeaderCell
    { Text = "Prekės galiojimo laikas dienomis" };
    TableHeaderCell price
    = new TableHeaderCell
    { Text = "Prekės vieneto kaina" };
    row.Cells.Add(goodName);
    row.Cells.Add(expiryDateInDays);
    row.Cells.Add(price);
    return row;
}

/// <summary>
/// Method for generating expiry goods tableheader row
/// </summary>
/// <returns>expiry goods tableheader row</returns>
public static TableHeaderRow GetTableHeaderRowForExpiryGoods()
{
    TableHeaderRow row = new TableHeaderRow();
    TableHeaderCell goodName
    = new TableHeaderCell { Text = "Prekės pavadinimas" };
    TableHeaderCell expiryDate
    = new TableHeaderCell
    { Text = "Prekės galiojamo laiko pabaiga" };
    TableHeaderCell price
    = new TableHeaderCell
    { Text = "Prekės vieneto kaina" };
    row.Cells.Add(goodName);
    row.Cells.Add(expiryDate);
    row.Cells.Add(price);
    return row;
```

```
}



/// <summary>
/// Method for generating a tableheaderrow for asortment goods data
/// </summary>
/// <returns>table header row for asorment goods</returns>
public static TableHeaderRow GetTableHeaderRowForAsortmentGoods()
{
    TableHeaderRow row = new TableHeaderRow();
    TableHeaderCell shopName
    = new TableHeaderCell { Text = "Parduotuvės pavadinimas" };
    TableHeaderCell asortmentCount
    = new TableHeaderCell
    { Text = "Asortimento kiekis" };
    row.Cells.Add(shopName);
    row.Cells.Add(asortmentCount);
    return row;
}

/// <summary>
/// Method for generating a table header row for goods for amount of money
/// </summary>
/// <returns>table header for amount of money goods</returns>
public static TableHeaderRow GetTableHeaderRowForAmountOfMoneyGoods()
{
    TableHeaderRow row = new TableHeaderRow();
    TableHeaderCell goodName
    = new TableHeaderCell { Text = "Parduotuvės pavadinimas" };
    TableHeaderCell amountOfGoods
    = new TableHeaderCell
    { Text = "Prekių vienetų likutis parduotuvėje"};
    TableHeaderCell price
    = new TableHeaderCell
    { Text = "Pinigų suma"};
    row.Cells.Add(goodName);
    row.Cells.Add(amountOfGoods);
    row.Cells.Add(price);
    return row;
}



/// <summary>
/// Compare to method for sorting goods by name and price
/// </summary>
/// <param name="obj">object</param>
/// <returns>compared goods</returns>
public int CompareTo(Goods obj)
{
    if (this.GoodsName == obj.GoodsName)
        {
        return Price.CompareTo(obj.Price);
    }
    if (this.GoodsName.CompareTo(obj.GoodsName) < 0)
        return -1;
```

```
            else if (this < obj)
                return 1;
            else
                return 0;
        }
        /// <summary>
        /// Equals method to find same objects
        /// </summary>
        /// <param name="other">other object</param>
        /// <returns>equal objects</returns>
        public bool Equals(Goods other)
        {
            if (other is null)
                return false;
            return GoodsName == other.GoodsName && ExpiryDateinDays ==
other.ExpiryDateinDays && Price == other.Price;
        }

        /// <summary>
        /// Equals method for checking if the object isnt null
        /// </summary>
        /// <param name="obj">other object</param>
        /// <returns>equal objects</returns>
        public override bool Equals(object obj)
        {
            if ((obj == null) || (Goods)obj == null)
                return false;
            return Equals((Goods)obj);
        }

        public override int GetHashCode()
        {
            return GoodsName.GetHashCode() ^ ExpiryDateinDays.GetHashCode() ^
Price.GetHashCode();
        }

        public override string ToString()
        {
            return string.Format("|{0,-25}|{1,40}|{2,25}|", GoodsName,
ExpiryDateinDays, Price);
        }

    }
}
```

GoodsTest.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using L2;
using FluentAssertions;
using System.Linq;

namespace UnitTestProject1
{
    [TestClass]
    public class GoodsTest
    {
        [TestMethod]
        public void Goods_paramsCtorTest()
        {
            Goods goods = new Goods("Sausainis", 20, 13);
            goods.GoodsName.Should().Be("Sausainis");
            goods.ExpiryDateinDays.Should().Be(20);
            goods.Price.Should().Be(13);
        }

        [TestMethod]
        public void Goods_paramsCtorTestSecond()
        {
            Goods goods = new Goods("Sausainis", DateTime.Parse("2020-03-20"),
13);

            goods.GoodsName.Should().Be("Sausainis");
            goods.date.Should().Be(DateTime.Parse("2020-03-20"));
            goods.Price.Should().Be(13);
        }

        [TestMethod]
        public void Goods_paramsEmptyConstructor()
        {
            Goods goods = new Goods();
            goods.Should().Be(goods);
        }



        [TestMethod()]
        public void TwoStringEqualLength()
        {
            Goods goods = new Goods("Preke", 0, 0);
            Goods secondGoods = new Goods("Kremas", 100, 50);
            Assert.AreEqual(goods.ToString().Length,
secondGoods.ToString().Length);

        }

        [TestMethod()]
        public void StringContainsGoodName()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Goods goods = new Goods("Bananas", 0, 0);
            string [] parts = goods.ToString().Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("Bananas");

        }

        [TestMethod()]
        public void StringContainsExpiryDateInDays()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Goods goods = new Goods("", 5, 0);
            string[] parts = goods.ToString().Split(delimiters,
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("5");


        }

        [TestMethod()]
        public void StringContainsPrice()
        {
            char[] delimiters = { '|', '-', ' ', ',' };
            Goods goods = new Goods("", 0, 10);
            string[] parts = goods.ToString().Split(delimiters, 
StringSplitOptions.RemoveEmptyEntries);
            parts.Should().Contain("10");
        }

        [TestMethod()]
        public void Equals_NameParameterIsTheSame_True()
        {
            Goods goods1 = new Goods("Preke", 0, 0);
            Goods goods2 = new Goods("Preke", 0, 0);
            goods1.Equals(goods2).Should().BeTrue();
        }

        [TestMethod()]
        public void Equals_NameParameterIsNotTheSame_False()
        {

            Goods goods1 = new Goods("Preke", 2, 5);
            Goods goods2 = new Goods("Bananas", 4, 7);
            goods1.Equals(goods2).Should().BeFalse();
        }

        [TestMethod()]
        public void CompareTo_PriceDifference_Goods1MoreThanGoods2()
        {
            Goods goods1 = new Goods("Bananas", 10, 15);
            Goods goods2 = new Goods("Bananas", 10, 13);
            goods1.CompareTo(goods2).Should().BeGreaterThan(0);

        }

        [TestMethod()]
        public void CompareTo_PriceDifference_Goods2MoreThanGoods2()
        {
            Goods goods1 = new Goods("Bananas", 10, 15);
            Goods goods2 = new Goods("Bananas", 10, 16);
            goods1.CompareTo(goods2).Should().BeLessThan(0);
        }

        [TestMethod()]
        public void CompareTo_PriceDifference_PriceIsTheSameTeamNameSame()
        {
            Goods goods1 = new Goods("Bananas", 10, 15);
            Goods goods2 = new Goods("Bananas", 10, 15);
            goods1.CompareTo(goods2).Should().Be(0);
        }

        [TestMethod()]
        public void CompareTo_GoodsName_PriceIsTheSameGoods2()
        {
            Goods goods1 = new Goods("Apelsinas", 0, 0);
            Goods goods2 = new Goods("Bananas", 0, 0);
            goods1.CompareTo(goods2).Should().BeLessThan(0);
        }

        [TestMethod()]
        public void CompareTo_GoodsName_PriceIsTheSameGoods1()
        {
            Goods goods1 = new Goods("Apelsinas", 0, 0);
            Goods goods2 = new Goods("Bananas", 0, 0);
            goods1.CompareTo(goods2).Should().BeLessThan(0);
        }

        [TestMethod()]
```

```
public void CompareTo_GoodsName_PriceIsTheSameForAllGoods()
{
    Goods goods1 = new Goods("Bananas", 10, 15);
    Goods goods2 = new Goods("Bananas", 10, 15);
    goods1.CompareTo(goods2).Should().Be(0);
}

[TestMethod()]
public void OperatorLT_GoodsNameDifferent_True()
{
    Goods goods1 = new Goods("Apelsinas", 10, 15);
    Goods goods2 = new Goods("Bananas", 10, 15);
    Assert.IsTrue(goods1 < goods2);
}
[TestMethod()]
public void OperatorHT_GoodsNameDifferent_False()
{
    Goods goods1 = new Goods("Bananas", 10, 15);
    Goods goods2 = new Goods("Apelsinas", 10, 15);
    Assert.IsTrue(goods1 > goods2);
}

[TestMethod()]
public void OperatorHT_GoodsNameDifferent_False1()
{
    Goods goods1 = new Goods("Bananas", 10, 15);
    Goods goods2 = new Goods("Apelsinas", 10, 15);
    Assert.IsFalse(goods1 < goods2);
}

[TestMethod()]
public void OpEqual_GoodsNameIsDifferent_False()
{
    Goods goods1 = new Goods("Bananas", 10, 15);
    Goods goods2 = new Goods("Apelsinas", 10, 15);
    Assert.IsFalse(goods1 == goods2);
}

[TestMethod()]
public void OpNotEqual_GoodsNameIsDifferent_True()
{
    Goods goods1 = new Goods("Bananas", 10, 15);
    Goods goods2 = new Goods("Apelsinas", 10, 15);
    Assert.IsTrue(goods1 != goods2);
}

[TestMethod()]
public void OperatorLT_PriceDifference_True()
{
    Goods goods1 = new Goods("Apelsinas", 10, 16);
    Goods goods2 = new Goods("Apelsinas", 10, 15);
    Assert.IsTrue(goods1 > goods2);
}

[TestMethod()]
public void OperatorLT_PriceDifference_False()
{
    Goods goods1 = new Goods("Apelsinas", 10, 15);
    Goods goods2 = new Goods("Apelsinas", 10, 16);
    Assert.IsFalse(goods1 > goods2);
}

[TestMethod()]
public void OperatorHT_PriceDifference_True()
{
```

```
            Goods goods1 = new Goods("Apelsinas", 10, 15);
            Goods goods2 = new Goods("Apelsinas", 10, 16);
            Assert.IsTrue(goods1 < goods2);
        }

        [TestMethod()]
        public void OperatorHT_PriceDifference_False()
        {
            Goods goods1 = new Goods("Apelsinas", 10, 16);
            Goods goods2 = new Goods("Apelsinas", 10, 15);
            Assert.IsFalse(goods1 < goods2);
        }

        [TestMethod()]
        public void OpEqual_PriceDifference_False()
        {
            Goods goods1 = new Goods("Apelsinas", 10, 16);
            Goods goods2 = new Goods("Bananas", 9, 15);
            Assert.IsFalse(goods1 == goods2);
        }

        [TestMethod()]
        public void OpNotEqual_PriceDifference_True()
        {
            Goods goods1 = new Goods("Apelsinas", 10, 15);
            Goods goods2 = new Goods("Bananas", 9, 15);
            Assert.IsTrue(goods1 != goods2);
        }

    }
}
```

newShopList.cs

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

namespace L2
{
    public sealed class newShopList<T> : IEnumerable<T>, ICollection<T>
 where T : IComparable<T>, ITableHeaders
    {
        [Serializable]
        private sealed class Node
        {
            public T Data;
            public Node Link;
            public Node(T data, Node address)
            {
                Data = data;
                Link = address;
            }
        }
        private Node Head;
        private Node Tail;
        private Node Current;
        public int Count
        {
            get
            {
                int c = 0;
                foreach (T t in this)
                {
                    c++;
                }
                return c;
            }
            private set { }
        }
        public bool IsReadOnly => false;
        /// <summary>
        /// Constructor with empty parameters.
        /// </summary>
        public newShopList()
        {
            Head = null;
            Tail = null;
            Current = null;
        }
        /// <summary>
        /// Returns the node data of the list
        /// </summary>
        /// <returns>node data</returns>
        public T Get()
        {
            return Current.Data;
        }
        /// <summary>
        /// Sets the head of the node
        /// </summary>
        public void Begin()
        {
            Current = Head;
        }
        /// <summary>
        /// Sets the next node
        /// </summary>
        public void Next()
        {
            if (Current != null)
                Current = Current.Link;
            else
                Current = Head.Link;
        }
        /// <summary>
        /// Checks if node exists
        /// </summary>
        public bool Exist()
        {
            return Current != null;
        }
        /// <summary>
        /// Inserts a new data node to the end of the list
        /// </summary>
        /// <param name="data">
        /// object of the list with data
        /// </param>
        public void Add(T data)
        {
            Node node = new Node(data, null);
            if (Head != null)
            {
                Tail.Link = node;
                Tail = node;
            }
            else
            {
                Head = node;
                Tail = node;
            }
        }
        /// <summary>
        /// Sort the linked list with selection sort.
        /// </summary>
        public void Sort()
        {
        for (Node i = Head; i != null; i = i.Link)
            {
                Node min = i;
                for (Node j = i.Link; j != null; j = j.Link)
                    if (j.Data.CompareTo(min.Data) < 0)
                        min = j;
                (min.Data, i.Data) = (i.Data, min.Data);
            }
        }
        /// <summary>
        /// Link list items convert to string
        /// </summary>
        public override string ToString()
        {
            string returnString = "";
            foreach (T current in this)
            {
                returnString += current.ToString() + '\n';
            }
            return returnString;
        }
        /// <summary>
        /// Method for foreach loop
        /// </summary>
        IEnumerator<T> IEnumerable<T>.GetEnumerator()
        {
            for (
            Node current = Head;
            current != null;
            current = current.Link)
                yield return current.Data;
        }
        /// <summary>
        /// Method for foreach loop
        /// </summary>
        public IEnumerator GetEnumerator()
        {
            for (
            Node current = Head;
            current != null;
            current = current.Link)
                yield return current.Data;
        }
        /// <summary>
        /// Clears the linked list
        /// </summary>
        public void Clear()
        {
            Head = null;
            Tail = null;
        }
        /// <summary>
        /// Searches the LinkList and if the specified
        /// item is already in the list,
        /// returns true, otherwise false.
        /// </summary>
        /// <param name="item">Item to search for.</param>
        /// <returns>Returns a boolean value.</returns>
        public bool Contains(T item)
        {
        for (Begin(); Exist(); Next())
            {
                if (Current.Data.Equals(item))
                    return true;
            }
            return false;
        }
        /// <summary>
        /// Copies the entire LinkList<T> to a compatible
        /// one-dimensional Array, starting at the
        /// specified index of the target array.
        /// </summary>
        /// <param name="array">
        /// The one-dimensional Array that is the destination
        /// of the elements copied from LinkList<T>.
        /// The Array must have zero-based indexing.</param>
        /// <param name="arrayIndex">
        /// The zero-based index in array at which copying begins.
        /// </param>
        public void CopyTo(T[] array, int arrayIndex)
        {
            if (array is null)
                throw new ArgumentNullException("array is null.");
            else if (arrayIndex < 0)
                throw new ArgumentOutOfRangeException
                ("arrayIndex is less than zero.");
            else if (Count > array.Length - arrayIndex)
                throw new ArgumentException
                ("The number of elements in the " +
                "source LinkList<T> is greater than " +
               "the available space from index to " +
               "the end of the destination array.");
            for (Begin(); Exist(); Next())
                array[arrayIndex++] = Current.Data;
        }
        /// <summary>
        /// Removes the first occurrence of the
        /// specified value from the LinkList<T>.
        /// </summary>
        /// <param name="item">
        /// The value to remove from the LinkList<T>.
        /// </param>
        /// <returns>
        /// true if the element containing value
        /// is successfully removed;
        /// otherwise, false. This method also
        /// returns false if value was not
        /// found in the original LinkList<T>.
        /// </returns>
        public bool Remove(T item)
        {
            for (Node c = Head; c != null; c = c.Link)
                if (c.Data.Equals(item))
                {
                    c = c.Link;
                    return true;
                }
            return false;
```

```
        }
    }

}
```

ShopListTest.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using L2;
using FluentAssertions;
using System.Linq;

namespace UnitTestProject1
{
    [TestClass]
    public class ShopListTest
    {
        [TestMethod()]
        public void Get_EmptyLinkList_NullReferemceExceptiom()
        {
            var linkList = new newShopList<Shop>();
            linkList.Begin();
            Action action = () => linkList.Get();
            action.Should().Throw<NullReferenceException>();
        }
        [TestMethod()]
        public void Get_NotEmptyLinkList_Team()
        {
            var linkList = new newShopList<Shop>();
            Shop TestTe = new Shop("", "", DateTime.Parse("2020/03/20"), 0, 0);
            linkList.Add(TestTe);
            linkList.Begin();

        }
        [TestMethod()]
        public void Add_NullObjectParameter_FirstElementShouldBeNull1()
        {
            var linkList = new newShopList<Shop>();
            linkList.Add(null);
            linkList.Begin();
            linkList.Get().Should().BeNull();
        }
        [TestMethod()]
        public void Add_TeamObjectParameter_FirstElementShouldBeThatTeam1()
        {
            var linkList = new newShopList<Shop>();
            Shop shop = new Shop("", "", DateTime.Parse("2020/03/20"), 0, 0);
            linkList.Add(shop);
            linkList.Begin();
            linkList.Get().Should().Be(shop);
        }
        [TestMethod()]
        public void Sort_5GoodsWithDifferentSoldAmount1()
        {
            var sortedLinkList = new newShopList<Shop>();
            var expectedSortedLinkList = new newShopList<Shop>();
            Shop shop1 = new Shop("", "", DateTime.Parse("2020/03/20"), 10, 0);
            Shop shop2 = new Shop("", "", DateTime.Parse("2020/03/20"), 15, 0);
            Shop shop3 = new Shop("", "", DateTime.Parse("2020/03/20"), 9, 0);
            Shop shop4 = new Shop("", "", DateTime.Parse("2020/03/20"), 25, 0);
            Shop shop5 = new Shop("", "", DateTime.Parse("2020/03/20"), 20, 0);
            sortedLinkList.Add(shop1);
            sortedLinkList.Add(shop2);
            sortedLinkList.Add(shop3);
            sortedLinkList.Add(shop4);
            sortedLinkList.Add(shop5);
            sortedLinkList.Sort();
            expectedSortedLinkList.Add(shop3);
            expectedSortedLinkList.Add(shop1);
            expectedSortedLinkList.Add(shop2);
            expectedSortedLinkList.Add(shop5);
            expectedSortedLinkList.Add(shop4);
            Assert.AreEqual(sortedLinkList.ToString(),
            expectedSortedLinkList.ToString());
        }


        [TestMethod()]
        public void Contains_EmptyList_False()
        {
            Shop shop1 = new Shop("", "", DateTime.Parse("2020/03/20"), 10, 0);
            var list = new newShopList<Shop>();
            Assert.IsFalse(list.Contains(shop1));
        }
        [TestMethod()]
        public void Contains_TwoItemsInList_False()
        {
            Shop shop11 = new Shop("maxima", "obuolys",
DateTime.Parse("2020/03/20"), 10, 0);
            Shop shop22 = new Shop("rimi", "apelsinas",
DateTime.Parse("2020/03/20"), 5, 4);
            Shop shop33 = new Shop("iki", "bananas", DateTime.Parse("2020/03/20"),
20, 1);
            var list = new newShopList<Shop>();
            Assert.IsFalse(list.Contains(shop11));
            Assert.IsFalse(list.Contains(shop22));
        }


        [TestMethod()]
        public void Exist_EmptyLinkList_False1()
        {
            newShopList<Shop> list = new newShopList<Shop>();
            list.Begin();
            list.Exist().Should().BeFalse();
        }

        [TestMethod()]
        public void Exist_1ElementLinkList_True1()
        {
            newShopList<Shop> list = new newShopList<Shop>();
            Shop goods = new Shop("", "", DateTime.Parse("2020/03/20"), 0, 0);
            list.Add(goods);
            list.Begin();
            list.Exist().Should().BeTrue();
        }

        [TestMethod()]
        public void Next_EmptyLinkList_NullReferenceException1()
        {
            newShopList<Shop> list = new newShopList<Shop>();
            list.Begin();

        }
        [TestMethod()]
        public void Next_1ElementLinkList_NullReferenceException1()
        {
            newShopList<Shop> list = new newShopList<Shop>();
            Shop goods = new Shop("", "", DateTime.Parse("2020/03/20"), 0, 0);
            list.Add(goods);
            list.Begin();
            list.Next();
            Action action = () => list.Get();
            action.Should().Throw<NullReferenceException>();
        }
        [TestMethod()]
        public void Next_2ElementLinkList_True1()
        {
            newShopList<Shop> list = new newShopList<Shop>();
            Shop goods1 = new Shop("a", "", DateTime.Parse("2020/03/20"), 0, 0);
            Shop goods2 = new Shop("b", "", DateTime.Parse("2020/03/20"), 0, 0);
            list.Add(goods1);
            list.Add(goods2);
            list.Begin();
            list.Next();
            list.Get().ToString().Should().Be(goods2.ToString());
        }

        [TestMethod()]
        public void
Begin_EmptyLinkList_GetShouldThrowNullReferenceExceptionThrown1()
        {
            newShopList<Shop> list = new newShopList<Shop>();
            list.Begin();
            Action action = () => list.Get();
            action.Should().Throw<NullReferenceException>();
        }
        [TestMethod()]
        public void Begin_1ElementLinkList_GetShouldBeFirstElement1()
        {
            newShopList<Shop> list = new newShopList<Shop>();
            Shop goods1 = new Shop("a", "", DateTime.Parse("2020/03/20"), 0, 0);
            list.Add(goods1);
            list.Begin();
            list.Get().ToString().Should().Be(goods1.ToString());
        }

    }
}
```

newGoodsList.cs

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

namespace L2
{
    public sealed class newGoodsList<T> : IEnumerable<T>, ICollection<T>
 where T : IComparable<T>, ITableHeaders
    {
        [Serializable]
        private sealed class Node
        {
            public T Data;
            public Node Link;
            public Node(T data, Node address)
            {
                Data = data;
                Link = address;
            }
        }
        private Node Head;
        private Node Tail;
        private Node Current;
        public int Count
        {
            get
            {
                int c = 0;
                foreach (T t in this)
                {
                    c++;
                }
                return c;
            }
            private set { }
        }
        public bool IsReadOnly => false;
        /// <summary>
        /// Constructor with empty parameters.
        /// </summary>
        public newGoodsList()
        {
            Head = null;
            Tail = null;
            Current = null;
        }
        /// <summary>
        /// Returns the selected node data
        /// </summary>
        /// <returns></returns>
        public T Get()
        {
            return Current.Data;
        }
        /// <summary>
        /// Sets the beginning of the linked list
        /// </summary>
        public void Begin()
        {
            Current = Head;
        }
        /// <summary>
        /// Sets the next node of the linked list
        /// </summary>
        public void Next()
        {
            if (Current != null)
                Current = Current.Link;
            else
                Current = Head.Link;
        }
        /// <summary>
        /// Checks if node exists
        /// </summary>
        public bool Exist()
        {
            return Current != null;
        }
        /// <summary>
        /// Inserts a new data object to the end of the linked list.
        /// </summary>
        /// <param name="data">
        /// Any object of the same type as this list.
        /// </param>
        public void Add(T data)
        {
            Node node = new Node(data, null);
            if (Head != null)
            {
                Tail.Link = node;
                Tail = node;
            }
            else
            {
                Head = node;
                Tail = node;
            }
        }
        /// <summary>
        /// Sort the linked list with selection sort.
        /// </summary>
        public void Sort()
        {
            for (Node i = Head; i != null; i = i.Link)
            {
                Node min = i;
                for (Node j = i.Link; j != null; j = j.Link)
                {
                    if (j.Data.CompareTo(min.Data) < 0)
                        min = j;
                }

                (min.Data, i.Data) = (i.Data, min.Data);
            }
        }
        /// <summary>
        /// Returns the LinkList items in a string.
        /// </summary>
        public override string ToString()
        {
            string returnString = "";
            foreach (T current in this)
            {
                returnString += current.ToString() + '\n';
            }
            return returnString;
        }
        /// <summary>
        /// Method for foreach loop
        /// </summary>
        IEnumerator<T> IEnumerable<T>.GetEnumerator()
        {
            for (
            Node current = Head;
            current != null;
            current = current.Link)
                yield return current.Data;
        }
        /// <summary>
        /// Method for foreach loop
        /// </summary>
        public IEnumerator GetEnumerator()
        {
            for (
            Node current = Head;
            current != null;
            current = current.Link)
                yield return current.Data;
        }
        /// <summary>
        /// Clears the LinkList
        /// </summary>
        public void Clear()
        {
            Head = null;
            Tail = null;
        }
        /// <summary>
        /// Searches the LinkList and if the specified
        /// item is already in the list,
        /// returns true, otherwise false.
        /// </summary>
        /// <param name="item">Item to search for.</param>
        /// <returns>Returns a boolean value.</returns>
        public bool Contains(T item)
        {
            for (Begin(); Exist(); Next())
            {
                if (Current.Data.Equals(item))
                    return true;
            }
            return false;
        }
        /// <summary>
        /// Copies the entire LinkList<T> to a compatible
        /// one-dimensional Array, starting at the
        /// specified index of the target array.
        /// </summary>
        /// <param name="array">
        /// The one-dimensional Array that is the destination
        /// of the elements copied from LinkList<T>.
        /// The Array must have zero-based indexing.</param>
        /// <param name="arrayIndex">
        /// The zero-based index in array at which copying begins.
        /// </param>
        public void CopyTo(T[] array, int arrayIndex)
        {
            if (array is null)
                throw new ArgumentNullException("array is null.");
            else if (arrayIndex < 0)
                throw new ArgumentOutOfRangeException
                ("arrayIndex is less than zero.");
            else if (Count > array.Length - arrayIndex)
                throw new ArgumentException
                ("The number of elements in the " +
                "source LinkList<T> is greater than " +
               "the available space from index to " +
               "the end of the destination array.");
            for (Begin(); Exist(); Next())
                array[arrayIndex++] = Current.Data;
        }
        /// <summary>
        /// Removes the first occurrence of the
        /// specified value from the LinkList<T>.
        /// </summary>
        /// <param name="item">
        /// The value to remove from the LinkList<T>.
        /// </param>
        /// <returns>
```

```
        /// true if the element containing value
        /// is successfully removed;
        /// otherwise, false. This method also
        /// returns false if value was not
        /// found in the original LinkList<T>.
        /// </returns>
        public bool Remove(T item)
        {
            for (Node c = Head; c != null; c = c.Link)
                if (c.Data.Equals(item))
                {
                    c = c.Link;
                    return true;
                }
            return false;
        }
    }

}
```

GoodsListTest.cs

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using L2;
using FluentAssertions;
using System.Linq;



namespace UnitTestProject1
{
    [TestClass]
    public class GoodsListTest
    {
        [TestMethod()]
        public void Get_EmptyLinkList_NullReferemceExceptiom()
        {
            var linkList = new newGoodsList<Goods>();
            linkList.Begin();
            Action action = () => linkList.Get();
            action.Should().Throw<NullReferenceException>();
        }
        [TestMethod()]
        public void Get_NotEmptyLinkList_Team()
        {
            var linkList = new newGoodsList<Goods>();
            Goods testTe = new Goods("b", 0, 0);
            linkList.Add(testTe);
            linkList.Begin();

        }
        [TestMethod()]
        public void Add_NullObjectParameter_FirstElementShouldBeNull()
        {
            var linkList = new newGoodsList<Goods>();
            linkList.Add(null);
            linkList.Begin();
            linkList.Get().Should().BeNull();
        }

        [TestMethod()]
        public void Add_TeamObjectParameter_FirstElementShouldBeThatTeam()
        {
            var linkList = new newGoodsList<Goods>();
            Goods goods = new Goods("a", 0, 0);
            linkList.Add(goods);
            linkList.Begin();
            linkList.Get().Should().Be(goods);
        }
        [TestMethod()]
        public void Sort_5GoodsWithDifferentGoodsName()
        {
            var sortedLinkList = new newGoodsList<Goods>();
            var expectedSortedLinkList = new newGoodsList<Goods>();
            Goods goods1 = new Goods("obuolys", 0, 0);
            Goods goods2 = new Goods("figa", 0, 0);
            Goods goods3 = new Goods("citrina", 0, 0);
            Goods goods4 = new Goods("bananas", 0, 0);
            Goods goods5 = new Goods("apelsinas", 0, 0);
            sortedLinkList.Add(goods1);
            sortedLinkList.Add(goods2);
            sortedLinkList.Add(goods3);
            sortedLinkList.Add(goods4);
            sortedLinkList.Add(goods5);
            sortedLinkList.Sort();
            expectedSortedLinkList.Add(goods5);
            expectedSortedLinkList.Add(goods4);
            expectedSortedLinkList.Add(goods3);
            expectedSortedLinkList.Add(goods2);
            expectedSortedLinkList.Add(goods1);
            Assert.AreEqual(sortedLinkList.ToString(),
            expectedSortedLinkList.ToString());
        }
        [TestMethod()]
        public void Sort_5GoodsButSameGoodsName()
        {

            newGoodsList<Goods> sortedLinkList = new newGoodsList<Goods>();
            newGoodsList<Goods> expectedSort = new newGoodsList<Goods>();
            Goods goods1 = new Goods("", 0, 52);
            Goods goods2 = new Goods("", 0, 15);
            Goods goods3 = new Goods("", 0, 14);
            Goods goods4 = new Goods("", 0, 13);
            Goods goods5 = new Goods("", 0, 12);
            sortedLinkList.Add(goods2);
            sortedLinkList.Add(goods1);
            sortedLinkList.Add(goods4);
            sortedLinkList.Add(goods3);
            sortedLinkList.Add(goods5);
            sortedLinkList.Sort();
            expectedSort.Add(goods5);
            expectedSort.Add(goods4);
            expectedSort.Add(goods3);
            expectedSort.Add(goods2);
            expectedSort.Add(goods1);
            Assert.AreEqual(expectedSort.ToString(), sortedLinkList.ToString());
        }

        [TestMethod()]
        public void Contains_EmptyList_False()
        {
            Goods goods = new Goods("", 0, 0);
            var list = new newGoodsList<Goods>();
            Assert.IsFalse(list.Contains(goods));
        }
        [TestMethod()]
        public void Contains_TwoItemsInList_False()
        {
            Goods goods1 = new Goods("a", 0, 0);
            Goods goods2 = new Goods("b", 0, 0);
            Goods goods3 = new Goods("c", 0, 0);
            var list = new newGoodsList<Goods>();
            list.Add(goods2);
            list.Add(goods3);
            Assert.IsFalse(list.Contains(goods1));
        }
        [TestMethod()]
        public void Contains_ItemsInList_True()
        {
            Goods goods1 = new Goods("a", 0, 0);
            Goods goods2 = new Goods("b", 0, 0);
            Goods goods3 = new Goods("c", 0, 0);
            var list = new newGoodsList<Goods>();
            list.Add(goods1);
            list.Add(goods2);
            list.Add(goods3);
            Assert.IsTrue(list.Contains(goods1));
            Assert.IsTrue(list.Contains(goods2));
            Assert.IsTrue(list.Contains(goods3));
        }


        [TestMethod()]
        public void Exist_EmptyLinkList_False()
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            list.Begin();
            list.Exist().Should().BeFalse();
        }

        [TestMethod()]
        public void Exist_1ElementLinkList_True()
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            Goods goods = new Goods("", 0, 0);
            list.Add(goods);
            list.Begin();
            list.Exist().Should().BeTrue();
        }

        [TestMethod()]
        public void Next_EmptyLinkList_NullReferenceException()
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            list.Begin();

        }
        [TestMethod()]
        public void Next_1ElementLinkList_NullReferenceException()
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            Goods goods = new Goods("a", 0, 0);
            list.Add(goods);
            list.Begin();
            list.Next();
            Action action = () => list.Get();
            action.Should().Throw<NullReferenceException>();
        }
        [TestMethod()]
        public void Next_2ElementLinkList_True()
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            Goods goods1 = new Goods("a", 0, 0);
            Goods goods2 = new Goods("b", 0, 0);
```

```
            list.Add(goods1);
            list.Add(goods2);

            list.Begin();
            list.Next();
            list.Get().ToString().Should().Be(goods2.ToString());
        }

        [TestMethod()]
        public void
Begin_EmptyLinkList_GetShouldThrowNullReferenceExceptionThrown()
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            list.Begin();
            Action action = () => list.Get();
            action.Should().Throw<NullReferenceException>();
        }
        [TestMethod()]
        public void Begin_1ElementLinkList_GetShouldBeFirstElement()
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            Goods goods1 = new Goods("a", 0, 0);
            list.Add(goods1);
            list.Begin();
            list.Get().ToString().Should().Be(goods1.ToString());
        }



    }
}
```

TaskUtils.cs

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

namespace L2
{
    /// <summary>
    /// Class for operations with data
    /// </summary>
    public static class TaskUtils
    {
        /// <summary>
        /// Method for finding most popural Goods
        /// </summary>
        /// <param name="goodsList">GoodsList with goods data</param>
        /// <param name="resultList">empty GoodsList to write data</param>
        /// <param name="shopList">ShopsList with shops data</param>
        public static void MostPopuralGoods(newGoodsList<Goods> goodsList, ref
newGoodsList<Goods> resultList, newShopList<Shop> shopList)
        {
            newShopList<Shop> temp = new newShopList<Shop>();
            temp = shopList;
            for (temp.Begin(); temp.Exist(); temp.Next())
            {
                for (goodsList.Begin(); goodsList.Exist(); goodsList.Next())
                {
                    if (goodsList.Get().GoodsName == temp.Get().GoodName)
                    {
                        Goods C = new Goods(goodsList.Get().GoodsName, 
goodsList.Get().ExpiryDateinDays, goodsList.Get().Price);
                        resultList.Add(C);
                    }
                }
            }
        }


        /// <summary>
        /// Method for finding shop with most asortment
        /// </summary>
        /// <param name="list">Shop list data</param>
        /// <param name="maxGoods">Count of different asortment</param>
        /// <returns>The shop name with most asortment and the count of 
asortment</returns>
        public static string FindShopWithLargestAsortmentOfGoods(newShopList<Shop> 
list, out int maxGoods)
        {
            Dictionary<string, List<string>> allGoodsInShop = new 
Dictionary<string, List<string>>();
            for(list.Begin(); list.Exist(); list.Next())
            {
                Shop shop = list.Get();
                if(!allGoodsInShop.ContainsKey(shop.ShopName))
                {
                    allGoodsInShop.Add(shop.ShopName, new List<string>());
                }
                if (!allGoodsInShop[shop.ShopName].Contains(shop.GoodName))
                {
                    allGoodsInShop[shop.ShopName].Add(shop.GoodName);
                }

            }
            maxGoods = -1;
            string maxShop = "";
            foreach (KeyValuePair<string, List<string>> keyValuePair in 
allGoodsInShop)
            {
                if (maxGoods < 0 || keyValuePair.Value.Count() > maxGoods)
                {
                    maxGoods = keyValuePair.Value.Count();
                    maxShop = keyValuePair.Key;
                }
            }

            return maxShop;
        }




        /// <summary>
        /// Method for finding a list with expiry goods
        /// </summary>
        /// <param name="shopList">ShopsList with shops data</param>
        /// <param name="goodsList">GoodsList with goods data</param>
        /// <param name="resultList">Empty GoodsList to write results</param>
        public static void ExpiryGoods(newShopList<Shop> shopList, 
newGoodsList<Goods> goodsList, ref newGoodsList<Goods> resultList)
        {
            for (shopList.Begin(); shopList.Exist(); shopList.Next())
            {
                for (goodsList.Begin(); goodsList.Exist(); goodsList.Next())
                {
                    if (goodsList.Get().ExpiryDateinDays <= 30 &&
shopList.Get().GoodName == goodsList.Get().GoodsName)
                    {
                        Goods C = new Goods(goodsList.Get().GoodsName,
shopList.Get().GoodsTime.AddDays(goodsList.Get().ExpiryDateinDays),
goodsList.Get().Price);
                        resultList.Add(C);
                    }
                }
            }
        }


        /// <summary>
        /// Method for finding goods for a particular amount of money
        /// </summary>
        /// <param name="goodsList">GoodsList with goods data</param>
        /// <param name="money">Amount of money</param>
        /// <param name="shopsList">ShopsList with shops data</param>
        /// <returns>GoodsList with data</returns>
        public static newGoodsList<Goods>
GoodsForAmountOfMoney(newGoodsList<Goods> goodsList, decimal money,
newShopList<Shop> shopsList)
        {
            newGoodsList<Goods> list = new newGoodsList<Goods>();
            for (goodsList.Begin(); goodsList.Exist(); goodsList.Next())
            {
                for (shopsList.Begin(); shopsList.Exist(); shopsList.Next())
                {
                    if (goodsList.Get().Price <= money)
                    {
                        if (goodsList.Get().GoodsName == shopsList.Get().GoodName)
                        {
                            Goods C = new Goods(shopsList.Get().ShopName,
shopsList.Get().GoodsAmount, goodsList.Get().Price);
                            list.Add(C);
                        }
                    }
                }
            }
            return list;
        }


    }



}
```

InOut.cs

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;
using System.Web;
using System.Web.UI.WebControls;

namespace L2
{
    /// <summary>
    /// Class for inputing and outputing data
    /// </summary>
    public static class InOut
    {
        /// <summary>
        /// Method for reading Shop file data
        /// </summary>
        /// <param name="fileName">Shop file with data</param>
        /// <param name="shopListData">List to put shop file data</param>
        public static void ReadShopList(string fileName, newShopList<Shop>
shopListData)
        {
            string eil = null;
            using (var file = new StreamReader(fileName))
            {
                while ((eil = file.ReadLine()) != null)
                {
                    string[] values = eil.Split(';');
                    string shopName = values[0];
                    string goodsName = values[1];
                    DateTime goodsTime = DateTime.Parse(values[2]);
                    int soldAmount = int.Parse(values[3]);
                    int goodsAmount = int.Parse(values[4]);
                    Shop C = new Shop(shopName, goodsName, goodsTime, soldAmount,
goodsAmount);
                    shopListData.Add(C);
                }
            }
        }


        /// <summary>
        /// Method for reading Goods file data
        /// </summary>
        /// <param name="fileName">Goods file with data</param>
        /// <param name="goodsListData">List to put goods file data</param>
       public static void ReadGoodsList(string fileName, newGoodsList<Goods>
goodsListData)
        {
            string eil = null;
            using (var file = new StreamReader(fileName))
            {
                while ((eil = file.ReadLine()) != null)
                {
                    string[] values = eil.Split(';');
                    string goodName = values[0];
                    double expiry = double.Parse(values[1]);
                    decimal price = decimal.Parse(values[2]);
                    Goods C = new Goods(goodName, expiry, price);
                    goodsListData.Add(C);
                }
            }
        }


        /// <summary>
        /// Method for printing initial shop table
        /// </summary>
        /// <typeparam name="T">class type</typeparam>
        /// <param name="table">table</param>
        /// <param name="list">shop list data</param>
        /// <param name="headerRow">the head row of the table</param>
        public static void PrintInitialShopTable<T>(Table table, newShopList<T>
list, TableHeaderRow headerRow)
        where T : IComparable<T>, ITableHeaders
        {
            table.Rows.Add(headerRow);
            foreach (T data in list)
            {
                table.Rows.Add(data.GetTableRow());
            }

        }

        /// <summary>
        /// Method for printing initial good data
        /// </summary>
        /// <typeparam name="T">class type</typeparam>
        /// <param name="table">table</param>
        /// <param name="list">goods list data</param>
        /// <param name="headerRow">the head row the table</param>
        public static void PrintInitialGoodsTable<T>(Table table, newGoodsList<T>
list, TableHeaderRow headerRow)
        where T : IComparable<T>, ITableHeaders
        {
            table.Rows.Add(headerRow);
            foreach (T data in list)
            {
                table.Rows.Add(data.GetTableRow());
            }

        }

        /// <summary>
        /// Method for printing expiry goods table
        /// </summary>
        /// <param name="table">table</param>
        /// <param name="expiryData">expiry goods data</param>
        public static void PrintExpiryTable(Table table,
newGoodsList<Goods>expiryData)
        {
            TableHeaderRow headerRow = Goods.GetTableHeaderRowForExpiryGoods();
            table.Rows.Add(headerRow);
            foreach(Goods data in expiryData)
            {
                table.Rows.Add(data.GetTableRowForExpiryGoods());
            }
        }
        /// <summary>
        /// Method for printing asortment data table
        /// </summary>
        /// <param name="table">table</param>
        /// <param name="shopName">the name of the shop</param>
        /// <param name="count">the asortment count</param>
        public static void PrintAsortmentTable(Table table, string shopName, int
count)
        {
            Goods data = new Goods();
            TableHeaderRow headerRow = Goods.GetTableHeaderRowForAsortmentGoods();
            table.Rows.Add(headerRow);
            table.Rows.Add(data.GetTableRowForAsortmentGoods(shopName, count));
        }

        /// <summary>
        /// Method for printing goods for amount of money table
        /// </summary>
        /// <param name="table">table</param>
        /// <param name="goodsForAmountOfMoney">goods for amount of money
table</param>
        public static void PrintGoodsForAmountOfMoneyTable(Table table,
newGoodsList<Goods> goodsForAmountOfMoney)
        {
            TableHeaderRow headerRow =
Goods.GetTableHeaderRowForAmountOfMoneyGoods();
            table.Rows.Add(headerRow);

            foreach (Goods data in goodsForAmountOfMoney)
            {
                table.Rows.Add(data.GetTableRowForAmountOfMoneyGoods());
            }
        }

        /// <summary>
        /// Method for saving information when page is refreshed
        /// </summary>
        /// <typeparam name="T">type class</typeparam>
        /// <param name="CFb">the file to save to</param>
        /// <param name="list">goods list data</param>
        public static void SaveGoodsList<T>(string CFb, newGoodsList<T> list)
             where T : IComparable<T>, ITableHeaders
        {
            BinaryFormatter formatter = new BinaryFormatter();
            FileStream file
            = new FileStream(
            CFb, FileMode.OpenOrCreate, FileAccess.Write
           );
            formatter.Serialize(file, list);
            file.Close();
        }

        /// <summary>
        /// Method for saving information when page is refreshed
        /// </summary>
        /// <typeparam name="T">type class</typeparam>
        /// <param name="CFb">the file to save to</param>
        /// <param name="list">shop list data</param>
        public static void SaveShopList<T>(string CFb, newShopList<T> list)
               where T : IComparable<T>, ITableHeaders
        {
            BinaryFormatter formatter = new BinaryFormatter();
            FileStream file
            = new FileStream(
            CFb, FileMode.OpenOrCreate, FileAccess.Write
           );
            formatter.Serialize(file, list);
            file.Close();
        }
        /// <summary>
        /// Method for loading page information from a file
        /// </summary>
        /// <typeparam name="T">type class</typeparam>
        /// <param name="CFb">File with information</param>
        /// <param name="list">good list information</param>
        public static void LoadGoodsList<T>(string CFb, out newGoodsList<T> list)
            where T : IComparable<T>, ITableHeaders
        {
            BinaryFormatter formatter = new BinaryFormatter();
            FileStream file
            = new FileStream(CFb, FileMode.Open, FileAccess.Read);
            list = (newGoodsList<T>)formatter.Deserialize(file);
            file.Close();
        }

        /// <summary>
        /// Method for loading page information from a file
        /// </summary>
        /// <typeparam name="T">type class</typeparam>
        /// <param name="CFb">File with information</param>
        /// <param name="list">shop list data</param>
        public static void LoadShopList<T>(string CFb, out newShopList<T> list)
            where T : IComparable<T>, ITableHeaders
        {
            BinaryFormatter formatter = new BinaryFormatter();
            FileStream file
            = new FileStream(CFb, FileMode.Open, FileAccess.Read);
            list = (newShopList<T>)formatter.Deserialize(file);
            file.Close();
        }

        /// <summary>
        /// Method for printing initial data
        /// </summary>
        /// <param name="fileName">resultFile</param>
        /// <param name="goodsListData">GoodsList with data</param>
        /// <param name="shopsListData">ShopsList with data</param>
        public static void PrintInitialDataToAFile(string fileName,
newGoodsList<Goods> goodsListData, newShopList<Shop> shopsListData)
        {
            using (StreamWriter write = File.CreateText(fileName))
            {
                write.WriteLine("Pradiniai duomenys");
                write.WriteLine("");
                write.WriteLine("Parduotuvės Sąrašas: ");
                write.WriteLine("-------------------------------------------------
----------------------------------------------------------------------------------
------------------------------");
                write.WriteLine(string.Format("|{0,-25}|{1,-25}|{2,-25}|{3,-
40}|{4,-40}|", "Parduotuvės pavadinimas", "Prekės pavadinimas", "Prekės gavimo
laikas", "Parduotų vienetų skaičius", "Prekių vienetų likutis parduotuvėje"));
                write.WriteLine("-------------------------------------------------
----------------------------------------------------------------------------------
------------------------------");
                for (shopsListData.Begin(); shopsListData.Exist();
shopsListData.Next())
                {
                    write.WriteLine(shopsListData.Get().ToString());
                    write.WriteLine("---------------------------------------------
----------------------------------------------------------------------------------
----------------------------------");
                }
                write.WriteLine("Prekių Sąrašas: ");
                write.WriteLine("-------------------------------------------------
---------------------------------------------");
                write.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-25}|", "Prekės
pavadinimas", "Prekės galiojimo laikas dienomis", "Prekės vieneto kaina"));
                write.WriteLine("-------------------------------------------------
---------------------------------------------");
                for (goodsListData.Begin(); goodsListData.Exist();
goodsListData.Next())
                {

                    write.WriteLine(goodsListData.Get().ToString());
                    write.WriteLine("---------------------------------------------
-------------------------------------------------");
                }
                write.WriteLine("");
            }
        }



        /// <summary>
        /// Method printing the most popural goods to a file
        /// </summary>
        /// <param name="fileName">file to print data to</param>
        /// <param name="mostPopural">most popural goods list</param>
        /// <param name="header">the header of the list</param>
        public static void PrintMostPopuralGoodsToAFile(string fileName,
newGoodsList<Goods>mostPopural, string header)
        {
                using (StreamWriter write = new StreamWriter(fileName, true,
Encoding.UTF8))
                {
                write.WriteLine(header);
                int mostPopuralGoodsTableLength = 94;
                write.WriteLine(new string('-', mostPopuralGoodsTableLength));
                write.WriteLine("|{0,-25}|{1,-40}|{2,-25}|", "Prekės pavadinimas",
"Galiojimo laikas", "Kaina");
                write.WriteLine(new string('-', mostPopuralGoodsTableLength));
                foreach(Goods goods in mostPopural)
                {
                    write.WriteLine(goods.ToString());
                }
                write.WriteLine(new string('-', mostPopuralGoodsTableLength));
            }
        }


        /// <summary>
        /// Method for printing expiry goods to a file
        /// </summary>
        /// <param name="fileName">File to write goods data to</param>
        /// <param name="expiry">goods list with data</param>
        /// <param name="header">the header of the list</param>
        public static void PrintExpiryGoodsToAFile(string fileName,
newGoodsList<Goods>expiry, string header)
        {
            using (StreamWriter write = new StreamWriter(fileName, true,
Encoding.UTF8))
            {
                write.WriteLine(header);
                int expiryGoodsTableLength = 109;
                write.WriteLine(new string('-', expiryGoodsTableLength));
                write.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-40}|", "Prekės
pavadinimas", "Prekės galiojimo laiko pabaiga", "Prekės vieneto kaina"));
                write.WriteLine(new string('-', expiryGoodsTableLength));
                foreach (Goods goods in expiry)
                {
                    write.WriteLine(string.Format("|{0,-25}|{1,-
40:yyyy/MM/dd}|{2,-40}|", goods.GoodsName, goods.date, goods.Price));
                }
                write.WriteLine(new string('-', expiryGoodsTableLength));
            }

        }

        /// <summary>
        /// Method for printing asortment data
        /// </summary>
        /// <param name="fileName">Result file</param>
        /// <param name="shopName">The name of the shop</param>
        /// <param name="goodCount">The count of asortment</param>
        public static void PrintMostAsortmentDataToAFile(string fileName, string
shopName, int goodCount)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                string dashes = new string('-', 53);
                sw.WriteLine("Didžiausio asortimento sąrašas");
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-25}|{1,-25}|", "Parduotuvės
pavadinimas", "Prekių kiekis"));
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-25}|{1,25}|", shopName,
goodCount));
                sw.WriteLine(dashes);
                sw.WriteLine("");
            }

        }
        /// <summary>
        /// Method for printing goods for amount of money to a file
        /// </summary>
        /// <param name="fileName">the file to write data to</param>
        /// <param name="goodsListData">goods list data with goods for amount of
money</param>
        /// <param name="header">the header of the list</param>
        public static void PrintGoodsForAmountToAFile(string fileName,
newGoodsList<Goods> goodsListData, string header)
        {
            using (StreamWriter write = new StreamWriter(fileName, true,
Encoding.UTF8))
            {
                write.WriteLine(header);
                int goodsforAmountTableLength = 94;
                write.WriteLine(new string('-', goodsforAmountTableLength));
                write.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-25}|",
"Parduotuvės pavadinimas", "Prekių vienetų likutis parduotuvėje", "Pinigų suma"));
                write.WriteLine(new string('-', goodsforAmountTableLength));
                foreach (Goods goods in goodsListData)
                {
                    write.WriteLine(string.Format("|{0,-25}|{1,-40}|{2,-25}|",
goods.GoodsName, goods.ExpiryDateinDays, goods.Price));
                }
                write.WriteLine(new string('-', goodsforAmountTableLength));
            }

        }




        /// <summary>
        /// Method for removing same files
        /// </summary>
        /// <param name="CFr">file</param>
        public static void RemoveFile(string CFr)
        {
            if (File.Exists(CFr))
                File.Delete(CFr);
        }


    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="L2.Forma1" MaintainScrollPositionOnPostBack="true" %>

<!DOCTYPE html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
   <link href="App_Codes1/StyleSheet1.css" rel="stylesheet" />
</head>

<body>

    <form id="form1" runat="server" enctype="multipart/form-data">
        <br />
        <asp:FileUpload ID="FileUpload1" runat="server" />
        <br />
        <asp:FileUpload ID="FileUpload2" runat="server" />
        <br />
        <asp:Button ID="Button5" runat="server" OnClick="Button5_Click"
Text="Išsaugoti" Height="46px" />
        <br />
        <asp:Label ID="Label1" runat="server" Text="Label"
Visible="False"></asp:Label>
        <br />
        <asp:Label ID="Label2" runat="server" Text="Label"
Visible="False"></asp:Label>
        <br />


&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

        <asp:Label ID="Label3" runat="server" Text="Label" Visible="False"
ForeColor="Black"></asp:Label>
            <asp:Table ID="Table1" runat="server" Visible="False"
BorderStyle="Solid" GridLines="Both">
        </asp:Table>
            <asp:Table ID="Table6" runat="server" Visible="False"
BorderStyle="Solid" GridLines="Both">
        </asp:Table>

        <br />

        <asp:Button CssClass ="ButtonClass1"  type ="button"  ID="Button1"
runat="server" Text="Perkamiausios prekės" OnClick="Button1_Click" Height="46px"
Width="540px" />
        <br />
        <asp:Table ID="Table2" runat="server" Visible="False" BorderStyle="Solid"
GridLines="Both">
        </asp:Table>
        <br />
        <p>
            <asp:Button  CssClass ="ButtonClass2" type ="button" ID="Button2"
runat="server" Text="Prekės kurių galiojimo laikas baigiasi"
OnClick="Button2_Click" Height="46px" Width="545px" />

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbs
p;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        </p>
        <asp:Table ID="Table3" runat="server" BorderStyle="Solid"
GridLines="Both">
        </asp:Table>
        <p>
```

```
            <asp:Button  CssClass ="ButtonClass3"  type ="button"   ID="Button3"
runat="server" Text="Didžiausias prekių asortimentas" OnClick="Button3_Click"
Height="46px" Width="545px" />
        </p>
        <asp:Table ID="Table4" runat="server" Visible="False" BorderStyle="Solid"
GridLines="Both">
        </asp:Table>
        <p>
            &nbsp;Įveskite sumą €
        </p>
        <p>
            <asp:TextBox ID="TextBox4" runat="server"
OnTextChanged="TextBox4_TextChanged"></asp:TextBox>
            <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ControlToValidate="TextBox4" ErrorMessage="Blogi duomenys!"
ForeColor="Red" ValidationExpression="^-?(([1-9]\d*)|0)(.0*[1-9](0*[1-
9])*)?$"></asp:RegularExpressionValidator>
        </p>
        <p>
            <asp:Button  CssClass ="ButtonClass4"  type ="button"  ID="Button4"
runat="server" Text="Surasti parduotuves" OnClick="Button4_Click" Height="46px"
Width="548px" />
        </p>
        <asp:Table ID="Table5" runat="server" Visible="False" BorderStyle="Solid"
GridLines="Both">
        </asp:Table>




    </form>
</body>
</html>
```

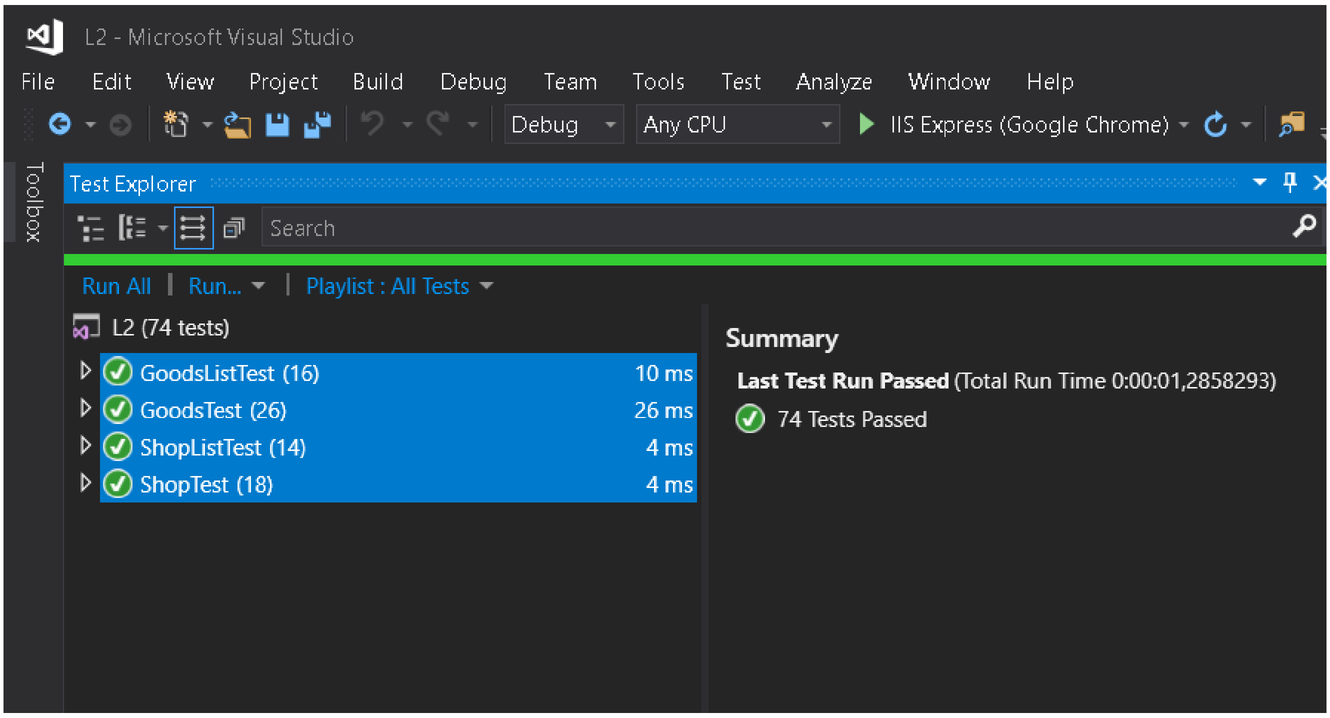
Forma1.aspx.cs

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

namespace L2
{


    public interface ITableHeaders
    {
        TableRow GetTableRow();
    }






    public partial class Forma1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {


        }







        protected void Button1_Click(object sender, EventArgs e)
        {
            newGoodsList<Goods> GoodsList = new newGoodsList<Goods>();
            newShopList<Shop> ShopsList = new newShopList<Shop>();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            InOut.ReadShopList((string)Session["firstFile"], ShopsList);
            InOut.ReadGoodsList((string)Session["secondFile"], GoodsList);
            InOut.PrintInitialDataToAFile(resultFile, GoodsList, ShopsList);
            newGoodsList<Goods> MostPopuralGoodsList = new newGoodsList<Goods>();
            TaskUtils.MostPopuralGoods(GoodsList, ref MostPopuralGoodsList, 
ShopsList);
            MostPopuralGoodsList.Sort();
            Session["PopuralGoods"] = MostPopuralGoodsList;
            InOut.PrintMostPopuralGoodsToAFile(resultFile, MostPopuralGoodsList, 
"Populiariausių prekių sąrašas");
            Label3.Text = "Pradiniai duomenys";
            Label3.Visible = true;
            Table2.Visible = true;
            Table1.Visible = true;
            Table6.Visible = true;
            InOut.PrintInitialShopTable<Shop>(Table1, ShopsList, 
Shop.GetInitialTableHeader());
            Session["ShopInitialData"] = Table1;
            InOut.PrintInitialGoodsTable<Goods>(Table6, GoodsList, 
Goods.GetInitialTableHeaderRow());
            Session["GoodsInitialData"] = Table6;
            InOut.PrintInitialGoodsTable<Goods>(Table2, MostPopuralGoodsList, 
Goods.GetInitialTableHeaderRow());
            Session["MostPopuralGoodsTable"] = Table2;

        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            newGoodsList<Goods> GoodsList = new newGoodsList<Goods>();
            newShopList<Shop> ShopsList = new newShopList<Shop>();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            InOut.ReadShopList((string)Session["firstFile"], ShopsList);
            InOut.ReadGoodsList((string)Session["secondFile"], GoodsList);
            newGoodsList<Goods> expiredList = new newGoodsList<Goods>();
            TaskUtils.ExpiryGoods(ShopsList, GoodsList, ref expiredList);
            expiredList.Sort();
            InOut.PrintExpiryGoodsToAFile(resultFile, expiredList, "Galiojimo 
laiko sąrašas");
            Table3.Visible = true;
            InOut.PrintExpiryTable(Table3, expiredList);
            Session["ExpiryTable"] = Table3;
        }
        protected void Button3_Click(object sender, EventArgs e)
        {
            newGoodsList<Goods> Goods = new newGoodsList<Goods>();
            newShopList<Shop> Shop = new newShopList<Shop>();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            InOut.ReadShopList((string)Session["firstFile"], Shop);
            InOut.ReadGoodsList((string)Session["secondFile"], Goods);

            string largerstAssortmentShopName =
TaskUtils.FindShopWithLargestAsortmentOfGoods(Shop, out int countOfGoods);
            InOut.PrintMostAsortmentDataToAFile(resultFile,
largerstAssortmentShopName, countOfGoods);
            Table4.Visible = true;
            InOut.PrintAsortmentTable(Table4, largerstAssortmentShopName,
countOfGoods);
            Session["AsortmentTable"] = Table4;

        }

        protected void Button4_Click(object sender, EventArgs e)
        {
            decimal moneyAmount = decimal.Parse(TextBox4.Text.ToString());
            newGoodsList<Goods> Goods = new newGoodsList<Goods>();
            newShopList<Shop> Shop = new newShopList<Shop>();
            string resultFile = Server.MapPath("App_Data1/Rezultatai.txt");
            InOut.ReadShopList((string)Session["firstFile"], Shop);
            InOut.ReadGoodsList((string)Session["secondFile"], Goods);
            newGoodsList<Goods> list = TaskUtils.GoodsForAmountOfMoney(Goods,
moneyAmount, Shop);
            InOut.PrintGoodsForAmountToAFile(resultFile, list, "Prekės už pinigų
sumą");
            Table5.Visible = true;
            InOut.PrintGoodsForAmountOfMoneyTable(Table5, list);
            Session["AsortmentTable"] = Table5;
        }

        protected void Button5_Click(object sender, EventArgs e)
        {
            if (FileUpload1.HasFile && FileUpload2.HasFile)
            {
                string firstFileFormat = Path.GetExtension(FileUpload1.FileName);
                string secondFileFormat = Path.GetExtension(FileUpload2.FileName);

                if (firstFileFormat.ToLower() != ".txt" &&
secondFileFormat.ToLower() != ".txt")
                {
                    Label1.Visible = true;
                    Label1.Text = "Blogas failų formatas!";
                    Label1.ForeColor = System.Drawing.Color.Red;
                }
                else
                {
                    Label1.Visible = false;
                    Label2.Visible = true;
                    string firstFileData = Server.MapPath("/App_Data1/" +
FileUpload1.FileName);
                    string secondFileData = Server.MapPath("/App_Data1/" +
FileUpload2.FileName);
                    Label2.Text = "Failai sėkmingai pasirinkti!";
                    Label2.ForeColor = System.Drawing.Color.Green;
                    Session["firstFile"] = firstFileData;
                    Session["secondFile"] = secondFileData;
                }
            }
            else
            {
                Label1.Visible = true;
                Label1.Text = "Nepasirinkti failai!";
                Label1.ForeColor = System.Drawing.Color.Red;
            }
        }


        protected void TextBox4_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

StyleSheet1.css

```
body {
    background: #33CCCC;
    text-align: left;
    animation: color 5s infinite linear;
    padding: 2em;
}
p {
    font-family: "Lucida Console", "Brush Script MT", monospace;
}


.image1 {
    border: 1px red solid;
    object-position:right;
    max-width:150px;
}
.ButtonClass1 {
    color: blue;
    border: 1px solid yellow;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}

.ButtonClass2 {
    color: green;
    border: 1px solid green;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}

.ButtonClass3 {
    color: red;
    border: 1px solid red;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}

.ButtonClass4 {
    color: deeppink;
    border: 1px solid white;
    background-color: white;
    font-family: "Lucida Console", "Brush Script MT", monospace;
}
@keyframes color {
    0% {
        background: #33CCCC;
    }

    20% {
        background: #33CC36;
    }

    40% {
        background: #B8CC33;
    }

    60% {
        background: #FCCA00;
    }

    80% {
        background: #33CC36;
    }

    100% {
        background: #33CCCC;
    }
}
```

3.7. Pradiniai duomenys ir rezultatai


L2 - Microsoft Visual Studio

Test Explorer

L2 (74 tests)

GoodsListTest (16)	10 ms
GoodsTest (26)	26 ms
ShopListTest (14)	4 ms
ShopTest (18)	4 ms

Summary

Last Test Run Passed (Total Run Time 0:00:01,2858293)

74 Tests Passed

Pirmieji duomenys (ne visos prekės už pinigų sumą, visų galiojimo laikas baigiasi)

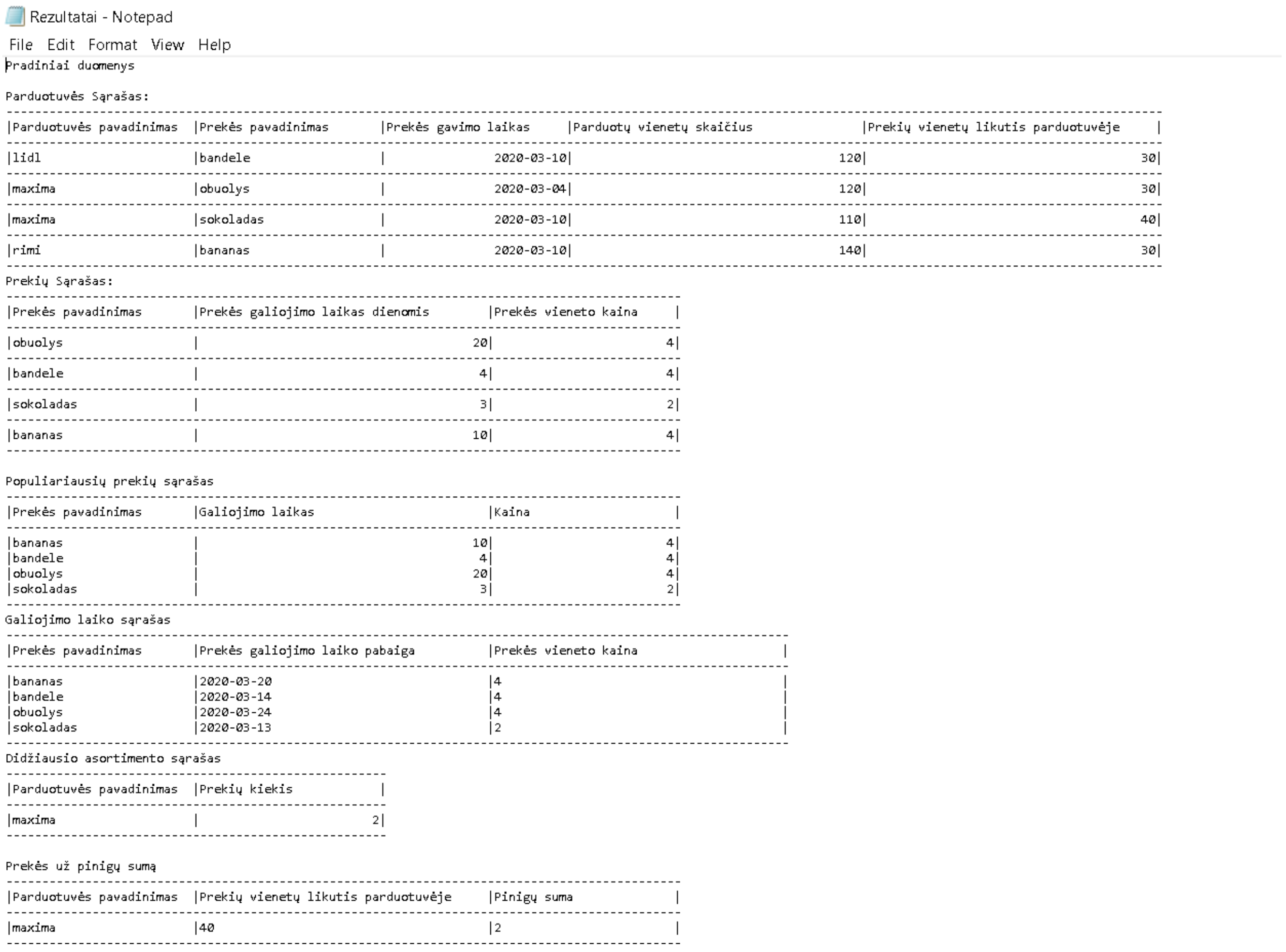
U19a.txt

```
lidl;bandele;2020/03/10;120;30

maxima;obuolys;2020/03/4;120;30

maxima;sokoladas;2020/03/10;110;40

rimi;bananas;2020/03/10;140;30
```

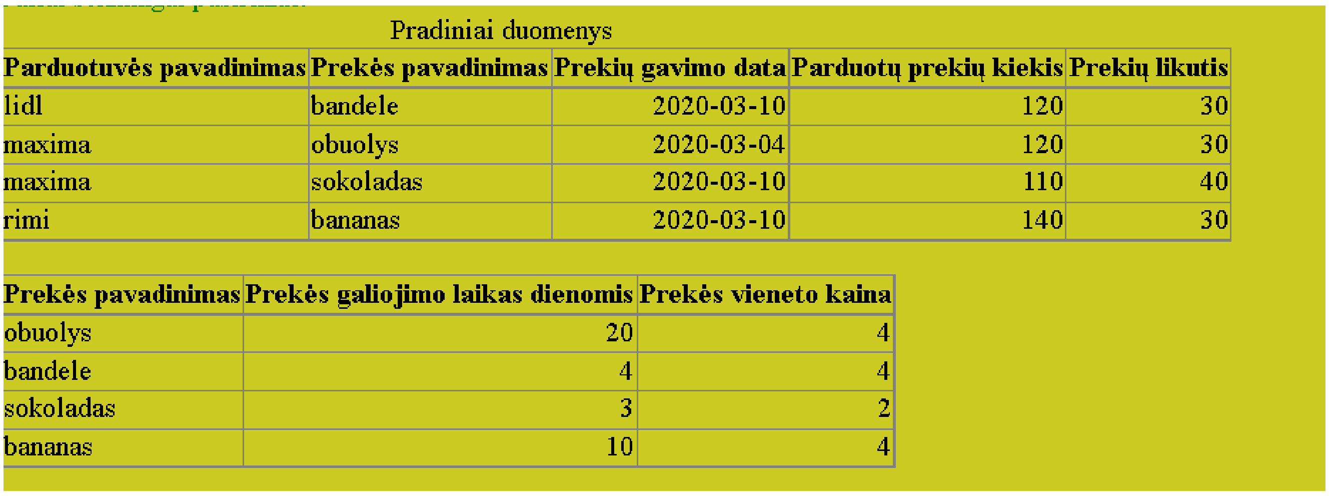
U19b.txt

```
obuolys;20;4

bandele;4;4

sokoladas;3;2

bananas;10;4


Pasirinkta pinigų suma: 2 (Price).
```

Rezultatai.txt



```
Pradiniai duomenys

Parduotuvės Sąrašas:
------------------------------------------------------------------------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas |Prekės pavadinimas      |Prekės gavimo laikas    |Parduotų vienetų skaičius              |Prekių vienetų likutis parduotuvėje    |
------------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                    |bandele                 |              2020-03-10|                                    120|                                     30|
------------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                  |obuolys                 |              2020-03-04|                                    120|                                     30|
------------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                  |sokoladas               |              2020-03-10|                                    110|                                     40|
------------------------------------------------------------------------------------------------------------------------------------------------------------
|rimi                    |bananas                 |              2020-03-10|                                    140|                                     30|
------------------------------------------------------------------------------------------------------------------------------------------------------------
Prekių Sąrašas:
-------------------------------------------------------------------------------------------
|Prekės pavadinimas      |Prekės galiojimo laikas dienomis       |Prekės vieneto kaina    |
-------------------------------------------------------------------------------------------
|obuolys                 |                                     20|                       4|
-------------------------------------------------------------------------------------------
|bandele                 |                                      4|                       4|
-------------------------------------------------------------------------------------------
|sokoladas               |                                      3|                       2|
-------------------------------------------------------------------------------------------
|bananas                 |                                     10|                       4|
-------------------------------------------------------------------------------------------

Populiariausių prekių sąrašas
-------------------------------------------------------------------------------------------
|Prekės pavadinimas      |Galiojimo laikas                       |Kaina                   |
-------------------------------------------------------------------------------------------
|bananas                 |                                     10|                       4|
|bandele                 |                                      4|                       4|
|obuolys                 |                                     20|                       4|
|sokoladas               |                                      3|                       2|
-------------------------------------------------------------------------------------------
Galiojimo laiko sąrašas
----------------------------------------------------------------------------------------------------------
|Prekės pavadinimas      |Prekės galiojimo laiko pabaiga         |Prekės vieneto kaina                   |
----------------------------------------------------------------------------------------------------------
|bananas                 |2020-03-20                             |4                                      |
|bandele                 |2020-03-14                             |4                                      |
|obuolys                 |2020-03-24                             |4                                      |
|sokoladas               |2020-03-13                             |2                                      |
----------------------------------------------------------------------------------------------------------
Didžiausio asortimento sąrašas
---------------------------------------------------
|Parduotuvės pavadinimas |Prekių kiekis           |
---------------------------------------------------
|maxima                  |                       2|
---------------------------------------------------

Prekės už pinigų sumą
-------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas |Prekių vienetų likutis parduotuvėje    |Pinigų suma             |
-------------------------------------------------------------------------------------------
|maxima                  |40                                     |2                       |
-------------------------------------------------------------------------------------------
```

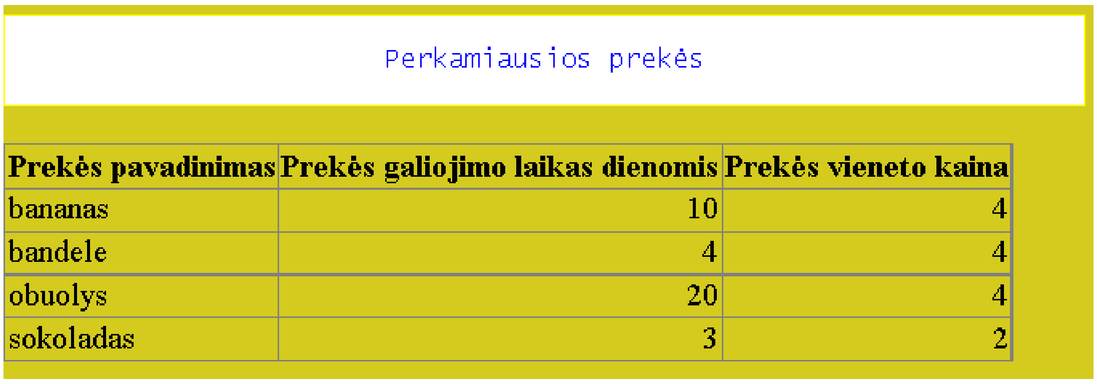


Pradiniai duomenys

Parduotuvės pavadinimas	Prekės pavadinimas	Prekių gavimo data	Parduotų prekių kiekis	Prekių likutis
lidl	bandele	2020-03-10	120	30
maxima	obuolys	2020-03-04	120	30
maxima	sokoladas	2020-03-10	110	40
rimi	bananas	2020-03-10	140	30

Prekės pavadinimas	Prekės galiojimo laikas dienomis	Prekės vieneto kaina
obuolys	20	4
bandele	4	4
sokoladas	3	2
bananas	10	4

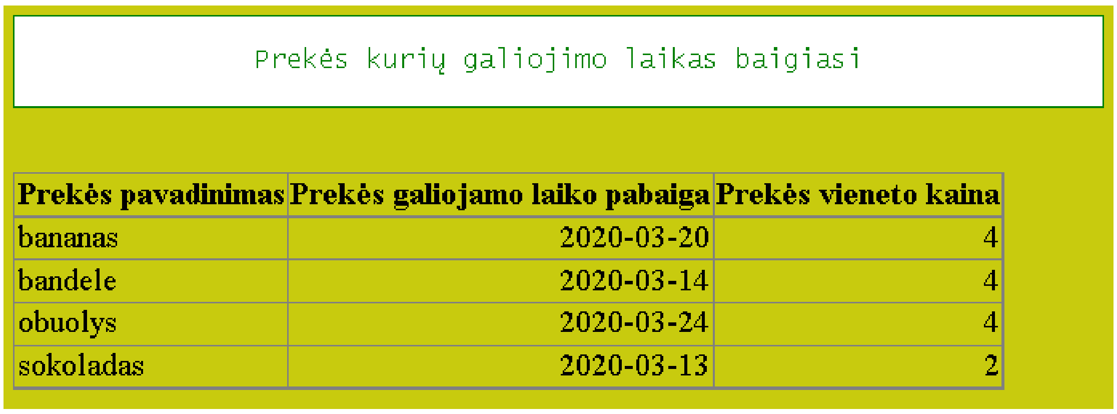
Pradiniai duomenys ekrane



Perkamiausios prekės

Prekės pavadinimas	Prekės galiojimo laikas dienomis	Prekės vieneto kaina
bananas	10	4
bandele	4	4
obuolys	20	4
sokoladas	3	2

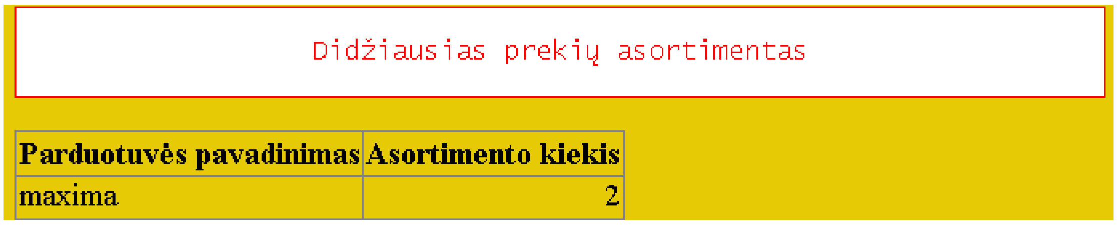
Perkamiausios prekės ekrane



Prekės kurių galiojimo laikas baigiasi

Prekės pavadinimas	Prekės galiojamo laiko pabaiga	Prekės vieneto kaina
bananas	2020-03-20	4
bandele	2020-03-14	4
obuolys	2020-03-24	4
sokoladas	2020-03-13	2

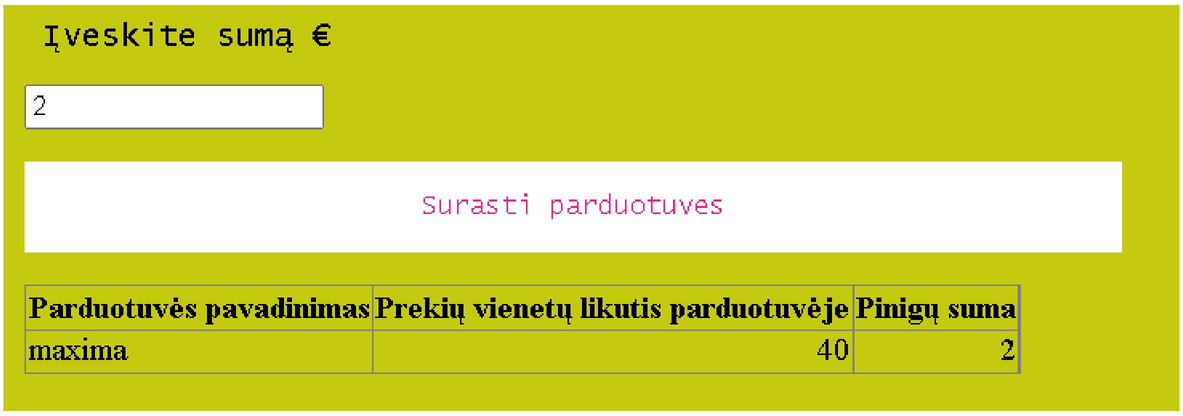
Prekės, kurių galiojimo laikas baigiasi



Didžiausias prekių asortimentas

Parduotuvės pavadinimas	Asortimento kiekis
maxima	2

Prekių asortimentas didžiausias


Įveskite sumą €

2

Surasti parduotuves

Parduotuvės pavadinimas	Prekių vienetų likutis parduotuvėje	Pinigų suma
maxima	40	2

Prekės už pinigus (Pasirinkta suma - 2)

Antrieji duomenys (Visos prekės už pinigų sumą, galiojimo laikas)

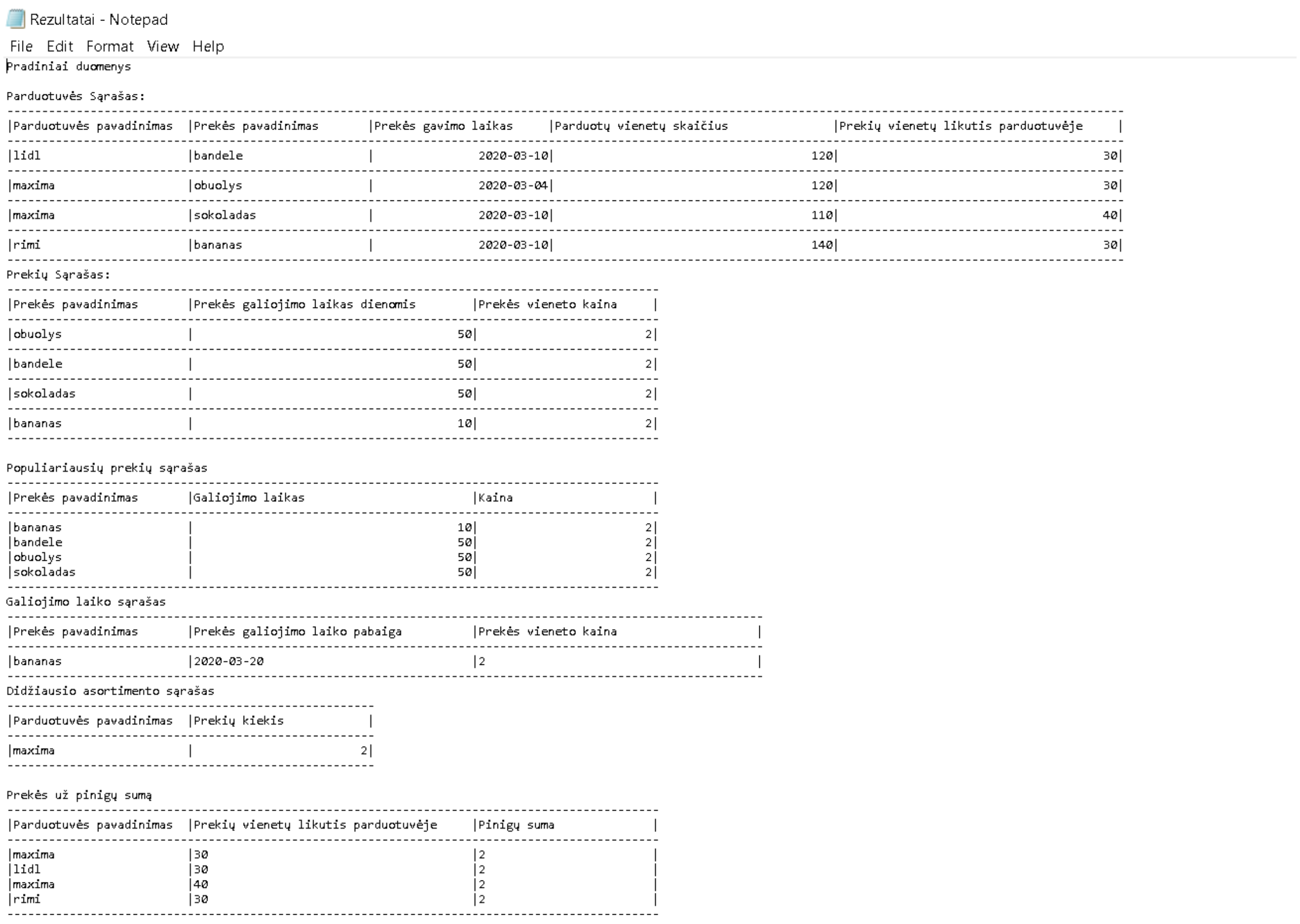
U19a.txt

```
lidl;bandele;2020/03/10;120;30

maxima;obuolys;2020/03/4;120;30

maxima;sokoladas;2020/03/10;110;40

rimi;bananas;2020/03/10;140;30
```

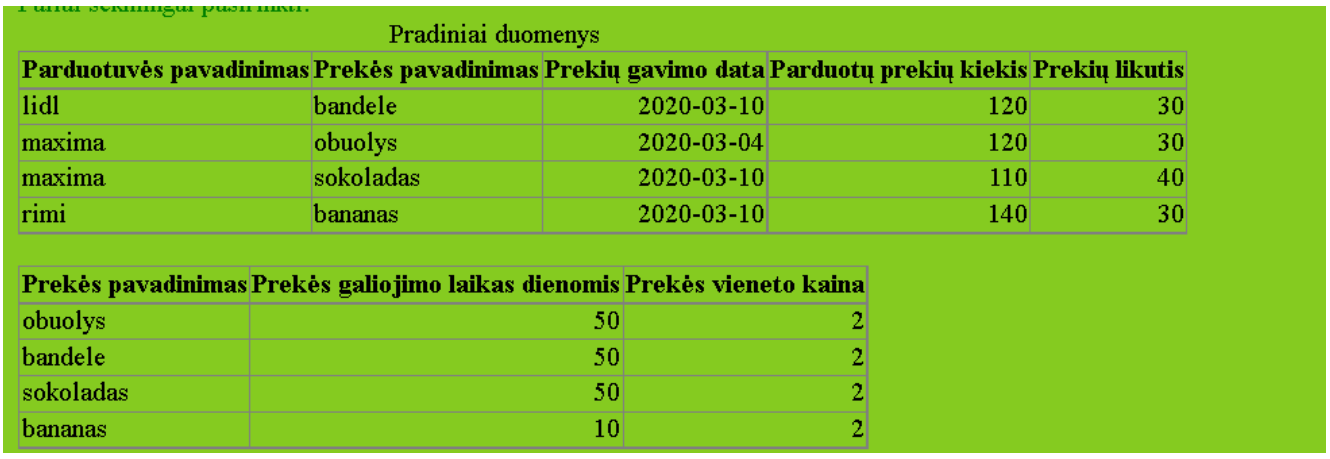
U19b.txt

```
obuolys;50;2

bandele;50;2

sokoladas;50;2

bananas;10;2


Pasirinkta pinigų suma: 2 (Price).
```

Rezultatai.txt



```
Pradiniai duomenys

Parduotuvės Sąrašas:
-------------------------------------------------------------------------------------------------------------------------------------------------------
|Parduotuvės pavadinimas |Prekės pavadinimas       |Prekės gavimo laikas    |Parduotų vienetų skaičius            |Prekių vienetų likutis parduotuvėje  |
-------------------------------------------------------------------------------------------------------------------------------------------------------
|lidl                    |bandele                  |              2020-03-10|                                  120|                                   30|
-------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                  |obuolys                  |              2020-03-04|                                  120|                                   30|
-------------------------------------------------------------------------------------------------------------------------------------------------------
|maxima                  |sokoladas                |              2020-03-10|                                  110|                                   40|
-------------------------------------------------------------------------------------------------------------------------------------------------------
|rimi                    |bananas                  |              2020-03-10|                                  140|                                   30|
-------------------------------------------------------------------------------------------------------------------------------------------------------
Prekių Sąrašas:
-----------------------------------------------------------------------------------------
|Prekės pavadinimas      |Prekės galiojimo laikas dienomis     |Prekės vieneto kaina    |
-----------------------------------------------------------------------------------------
|obuolys                 |                                   50|                       2|
-----------------------------------------------------------------------------------------
|bandele                 |                                   50|                       2|
-----------------------------------------------------------------------------------------
|sokoladas               |                                   50|                       2|
-----------------------------------------------------------------------------------------
|bananas                 |                                   10|                       2|
-----------------------------------------------------------------------------------------

Populiariausių prekių sąrašas
-----------------------------------------------------------------------------------------
|Prekės pavadinimas      |Galiojimo laikas                     |Kaina                   |
-----------------------------------------------------------------------------------------
|bananas                 |                                   10|                       2|
|bandele                 |                                   50|                       2|
|obuolys                 |                                   50|                       2|
|sokoladas               |                                   50|                       2|
-----------------------------------------------------------------------------------------
Galiojimo laiko sąrašas
-------------------------------------------------------------------------------------------------------
|Prekės pavadinimas      |Prekės galiojimo laiko pabaiga       |Prekės vieneto kaina                  |
-------------------------------------------------------------------------------------------------------
|bananas                 |2020-03-20                           |2                                     |
-------------------------------------------------------------------------------------------------------
Didžiausio asortimento sąrašas
---------------------------------------------------
|Parduotuvės pavadinimas |Prekių kiekis           |
---------------------------------------------------
|maxima                  |                       2|
---------------------------------------------------

Prekės už pinigų sumą
-----------------------------------------------------------------------------------------
|Parduotuvės pavadinimas |Prekių vienetų likutis parduotuvėje  |Pinigų suma             |
-----------------------------------------------------------------------------------------
|maxima                  |30                                   |2                       |
|lidl                    |30                                   |2                       |
|maxima                  |40                                   |2                       |
|rimi                    |30                                   |2                       |
-----------------------------------------------------------------------------------------
```

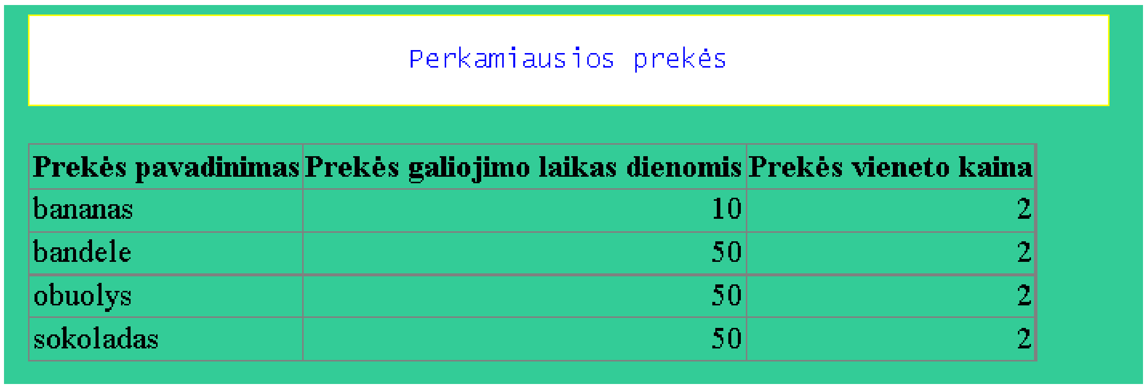


Pradiniai duomenys

Parduotuvės pavadinimas	Prekės pavadinimas	Prekių gavimo data	Parduotų prekių kiekis	Prekių likutis
lidl	bandele	2020-03-10	120	30
maxima	obuolys	2020-03-04	120	30
maxima	sokoladas	2020-03-10	110	40
rimi	bananas	2020-03-10	140	30

Prekės pavadinimas	Prekės galiojimo laikas dienomis	Prekės vieneto kaina
obuolys	50	2
bandele	50	2
sokoladas	50	2
bananas	10	2

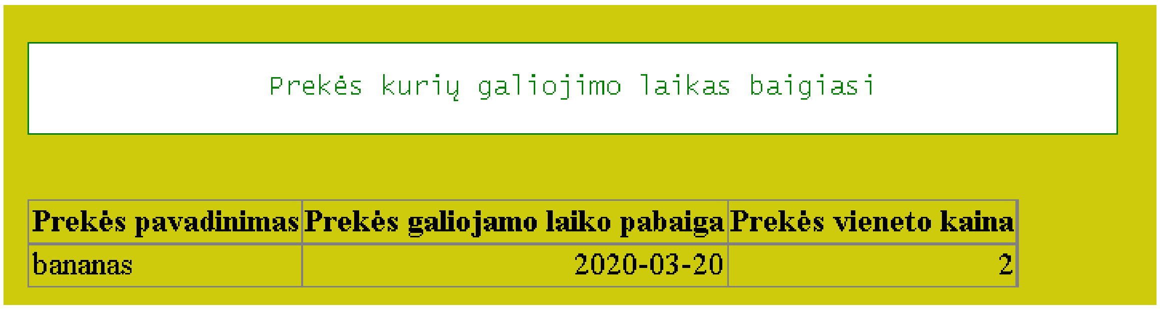
Pradiniai duomenys



Perkamiausios prekės

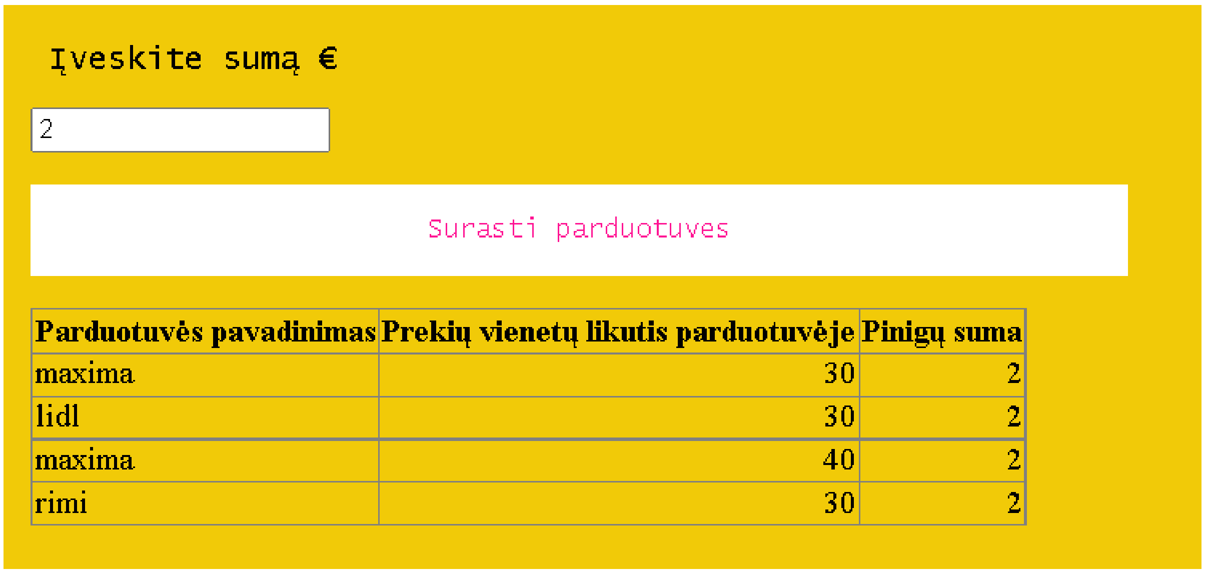
Prekės pavadinimas	Prekės galiojimo laikas dienomis	Prekės vieneto kaina
bananas	10	2
bandele	50	2
obuolys	50	2
sokoladas	50	2

Perkamiausios prekės


Prekės kurių galiojimo laikas baigiasi

Prekės pavadinimas	Prekės galiojamo laiko pabaiga	Prekės vieneto kaina
bananas	2020-03-20	2

Prekės, kurių galiojimo laikas baigiasi


Įveskite sumą €

2

Surasti parduotuves

Parduotuvės pavadinimas	Prekių vienetų likutis parduotuvėje	Pinigų suma
maxima	30	2
lidl	30	2
maxima	40	2
rimi	30	2

Prekės už pinigų sumą (2 – Price)

3.8. Dėstytojo pastabos

4. Polimorfizmas ir išimčių valdymas (L4)

4.1. Darbo užduotis

U4_19. WCG turnyras. Turite keleto turnyro ratų duomenis. Pirmoje eilutėje – rato numeris, antroje – data. Toliau pateikta informacija apie to rato rezultatus. Turnyre varžosi kelių skirtingų žaidimų („League of Legends“ ir „Counter Strike“) žaidėjai. Sukurkite abstrakčią klasę „Player“ (savybės - vardas, pavardė, komanda), kurią paveldės klasės “LoLPlayer” (savybės - pozicija, čempionas, nužudymai(K), mirtys(D), dalyvavimai nužudymuose(A)) ir “CSPlayer” (savybės - nužudymai(K), mirtys(D), mėgstamiausias ginklas).

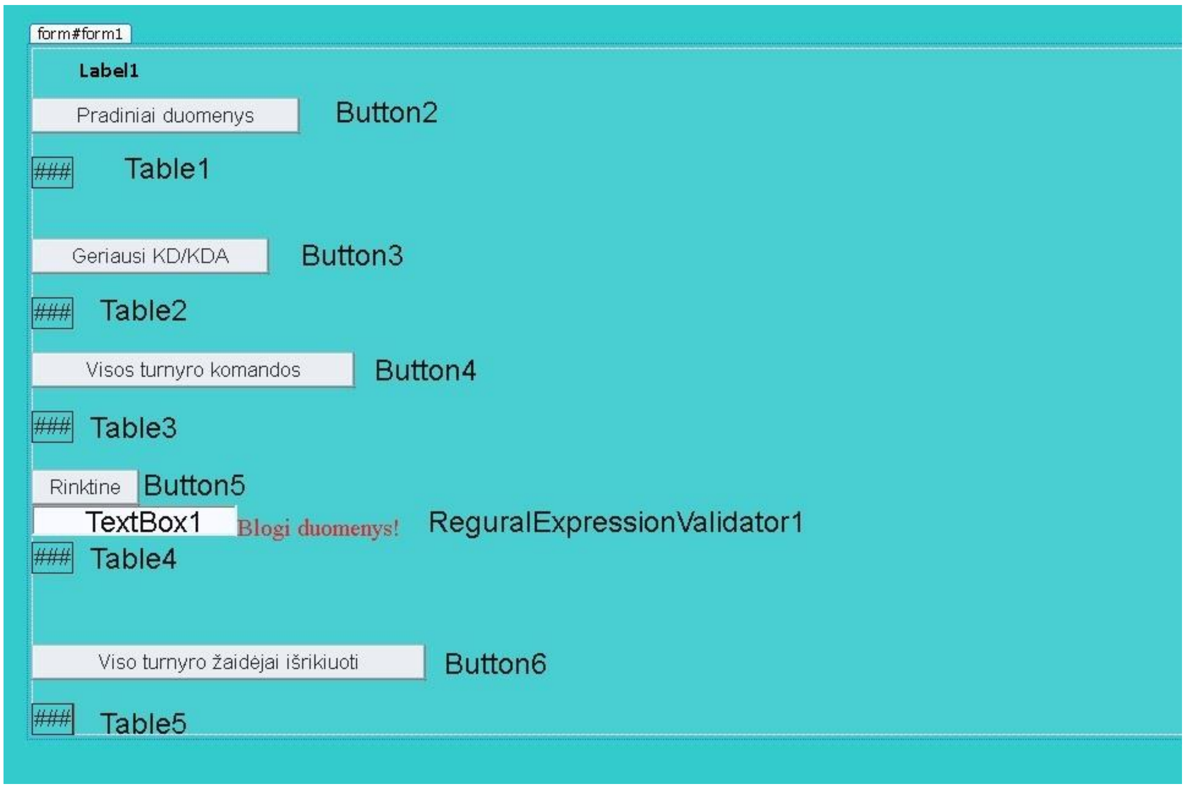
• Raskite LoL ir CS žaidėją, pademonstravusį geriausią bendrą (per visus tris ratus) asmeninį rezultatą. LoL žaidėjų palyginimui naudokite KDA santykį (nužudymai + dalyvavimai nužudymuose)/mirtys t. y. (K+A)/D, o CS žaidėjams palyginti - KD santykį (K/D). Ekrane atspausdinkite jų vardus, pavardes ir komandos pavadinimą.

• Sudarykite visų turnyre dalyvaujančių komandų sąrašą, ir atspausdinkite faile „Komandos.csv“.

• Sudarykite bendrą žaidėjų rinktinę. Į šią rinktinę patenka LoL žaidėjai, jei jų nužudymų kiekis yra didesnis už nurodytą. Į šią rinktinę patenka CS žaidėjai, jei jų mirčių kiekis yra mažesnis už nurodytą. Rezultatus įrašykite į failą „Rinktine.csv“.

• Sudarykite visų turnyro dalyvių sąrašą, LoL žaidėjus rikiuokite pagal KDA santykį, o CS žaidėjus – pagal KD santykį. Rezultatus įrašykite į failą „Visi.csv“.

4.2. Grafinės vartotojo sąsajos schema

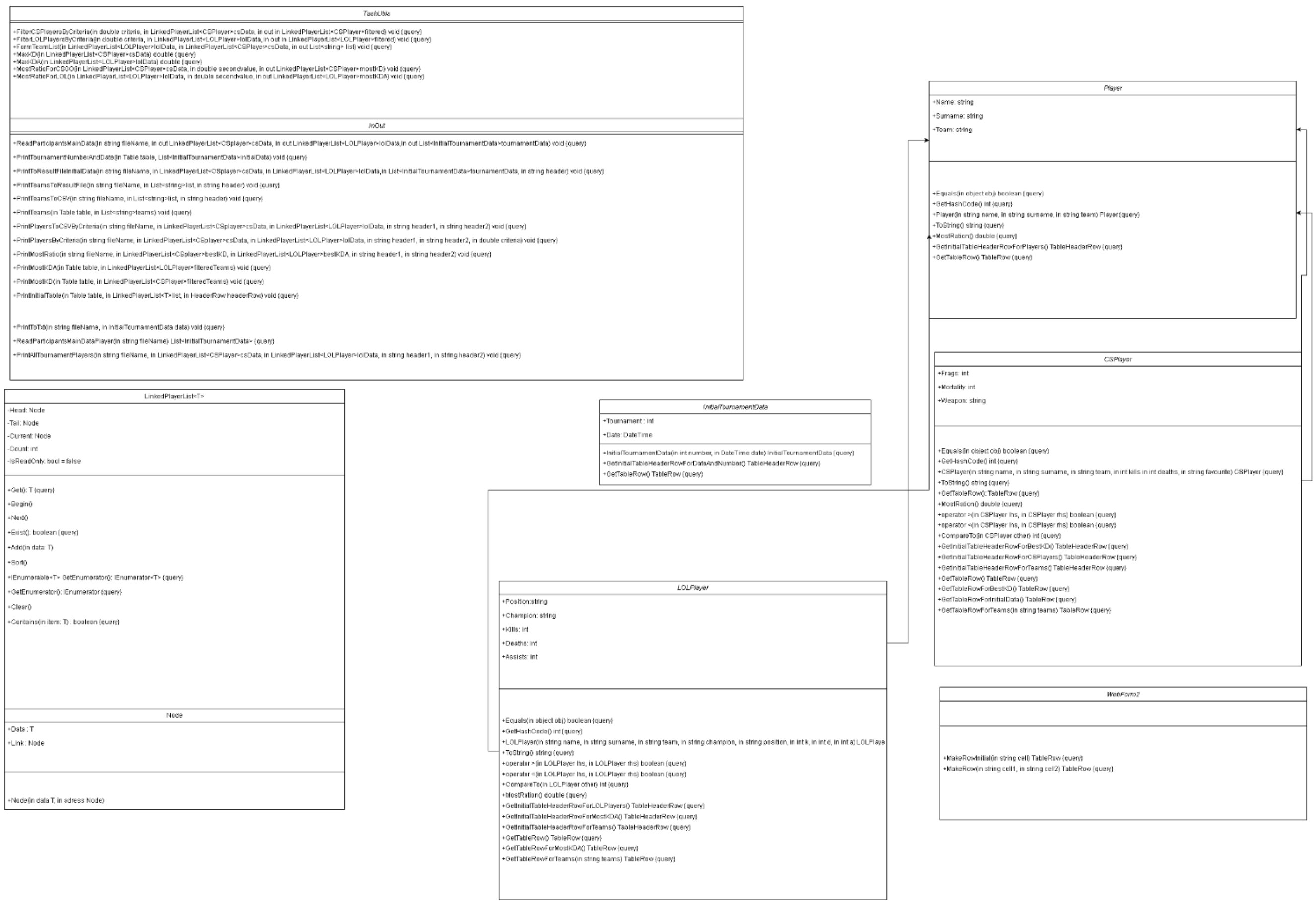


4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Table1	BorderStyle, GridLines, ForeColor	Solid, Both, black
Table2	BorderStyle, GridLines, ForeColor	Solid, Both, black
Table3	BorderStyle, GridLines, ForeColor	Solid, Both, black
Table4	BorderStyle, GridLines, ForeColor	Solid, Both, black
Table5	BorderStyle, GridLines, ForeColor	Solid, Both, black
Button2	Text	„Pradiniai duomenys“,
Button3	Text	„Geriausi KD/KDA“,
Button4	Text	„Visos turnyro komandos“,
Button5	Text	„Rinktine“,
Button6	Text	„Viso turnyro žaidėjai išrikiuoti“,
RegularExpressionValidator	ErrorMessage, ControlToValidate,	„Blogi duomenys!‘‘,TextBox4,^-?(([1-9]\d*)|0)(.0*[1-9](0*[1-9])*)?$, Red

	ValidationGroup, ForeColor	
TextBox1	Niekas nepakeista	
Label1	Visible	False

4.4. Klasių diagrama



4.5. Programos vartotojo vadovas

Paleidus programą turnyrų failų direktorijoje turime tam tikrus failus, kurie pavadinime turi „turnyras‘‘ ir tam tikrus duomenis. Norėdami matyti visus pradinius duomenis, spaudžiame mygtuką „Pradiniai duomenys‘‘. Norėdami surasti abiejų žaidimų žaidėjų geriausius KD/KDA rodiklius, spaudžiame mygtuką „Geriausi KD/KDA‘‘. Surasti visoms komandoms spaudžiame mygtuką „Visos turnyro komandos‘‘. Sudaryti rinktinei į TextBox elementą įvedame tam tikrą rodiklį ir spaudžiame mygtuką „Rinktine‘‘. Visiems turnyro žaidėjams išrušiuoti spaudžiame mygtuką „Viso turnyro žaidėjai išrikiuoti‘‘. Visus rezultatus galime pažiurėti Rezultatai.txt faile arba kituose duomenų .csv failuose.

4.6. Programos tekstas

Player.cs

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

namespace L4s.App_Code1
{
    /// <summary>
    /// Data class for Player information
    /// </summary>
    public abstract class Player : IComparable<Player>, IEquatable<Player>,
Forma1.ITableHeaders
    {
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Team { get; set; }
        public string Position { get; set; }
        /// <summary>
        /// Constructor for Player data
        /// </summary>
        /// <param name="name">Name</param>
        /// <param name="surname">Surname</param>
        /// <param name="team">Team</param>
        public Player(string name, string surname, string team)
        {

            this.Name = name;
            this.Surname = surname;
            this.Team = team;
        }

        /// <summary>
        /// Equals method for comparing objects
        /// </summary>
        /// <param name="obj">other object</param>
        /// <returns>If objects are the same</returns>
        public override bool Equals(object obj)
        {
            return Name == ((Player)obj).Name && Surname == ((Player)obj).Surname
&& Team == ((Player)obj).Team;
        }

        public override int GetHashCode()
        {
            return Name.GetHashCode() ^ Surname.GetHashCode() ^
Team.GetHashCode();
        }
        /// <summary>
        /// Abstract method for KD or KDA ratio
        /// </summary>
        /// <returns></returns>
        public abstract double MostRation();

        public override string ToString()
        {
            return String.Format("| {0, -20} | {1, -20} | {2, -10} |", Name, 
Surname, Team);
        }

        /// <summary>
        /// CompareTo method for comparing by KD or KDA
        /// </summary>
        /// <param name="other">other player data</param>
        /// <returns>compared by KDA or KD</returns>
        public int CompareTo(Player other)
        {
            if (MostRation().CompareTo(other.MostRation()) > 0)
                return 1;
            else if (MostRation().CompareTo(other.MostRation()) == 0)
                return 0;
            else
                return -1;
        }
        /// <summary>
        /// Equals method for finding equal objects
        /// </summary>
        /// <param name="other">other object</param>
        /// <returns>Equal objects</returns>
        public bool Equals(Player other)
        {
            if (other is null)
                return false;
            return Name == other.Name && Surname == other.Surname && Team == 
other.Team;

        }


        public static TableHeaderRow GetInitialTableHeaderRowForAllPlayers()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell name = new TableHeaderCell { Text = "Vardas" };
            TableHeaderCell surname = new TableHeaderCell { Text = "Pavardė" };
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            TableHeaderCell kills = new TableHeaderCell { Text = "CS 
Nužudymai/Pozicija" };
            TableHeaderCell deaths = new TableHeaderCell { Text = "CS 
Mirtys/Čempionas" };
            TableHeaderCell gun = new TableHeaderCell { Text = "Ginklas/LOL 
nužudymai" };
            TableHeaderCell loldeath = new TableHeaderCell { Text = "LOL mirtys" 
};
            TableHeaderCell lolassist = new TableHeaderCell { Text = "LOL 
assistai" };
            row.Cells.Add(name);
            row.Cells.Add(surname);
            row.Cells.Add(team);
            row.Cells.Add(kills);
            row.Cells.Add(deaths);
            row.Cells.Add(gun);
            row.Cells.Add(loldeath);
            row.Cells.Add(lolassist);
            return row;
        }
        /// <summary>
        /// Abstract method for printing tables
        /// </summary>
        /// <returns>TableRow</returns>
        public abstract TableRow GetTableRow();
    }
```

```
}
```

CSPlayer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Web.UI.WebControls;
using L4s;

namespace L4s.App_Code1
{
    /// <summary>
    /// CS player data class
    /// </summary>
    public class CSPlayer : Player, IComparable<CSPlayer>, IEquatable<CSPlayer>,
Formal.ITableHeaders
    {
        public int Frags { get; private set; }
        public int Mortality { get; private set; }
        public string Weapon { get; private set; }
        /// <summary>
        /// Data class constructor
        /// </summary>
        /// <param name="name">Name</param>
        /// <param name="surname">Surname</param>
        /// <param name="team">Team</param>
        /// <param name="kills">Kills</param>
        /// <param name="deaths">Deaths</param>
        /// <param name="favorite">Favourite gun</param>
        public CSPlayer(string name, string surname, string team, int kills, int
deaths, string favorite) : base(name, surname, team)
        {
            this.Frags = kills;
            this.Mortality = deaths;
            this.Weapon = favorite;
        }


        public override string ToString()
        {
            return String.Format(base.ToString() + " {0, -12} | {1, 25} | {2, 10}
| {3, 10} | {4, 10} | {5, 31} |", "", "", Frags, Mortality, "", Weapon);
        }
        /// <summary>
        /// TableRow for initial CS player data
        /// </summary>
        /// <returns>TableRow</returns>
        public TableRow GetTableRowForInitialData()
        {
            TableRow row = new TableRow();

            TableCell contenderName = new TableCell { Text = Name.ToString() };
            TableCell contenderSurname = new TableCell { Text = Surname.ToString()
};
            TableCell contenderTeam = new TableCell { Text = Team.ToString() };
            TableCell contenderKills = new TableCell
            {
                Text = Frags.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell contenderDeaths = new TableCell
            {
                Text = Mortality.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell contenderGun = new TableCell { Text = Weapon.ToString() };
            row.Cells.Add(contenderName);
            row.Cells.Add(contenderSurname);
            row.Cells.Add(contenderTeam);
            row.Cells.Add(contenderKills);
            row.Cells.Add(contenderDeaths);
            row.Cells.Add(contenderGun);
            return row;
        }

        /// <summary>
        /// TableRow for CSPlayer data
        /// </summary>
        /// <returns>TableRow</returns>
        public override TableRow GetTableRow()
        {
            TableRow row = new TableRow();

            TableCell contenderName = new TableCell { Text = Name.ToString() };
            TableCell contenderSurname = new TableCell { Text = Surname.ToString()
};
            TableCell contenderTeam = new TableCell { Text = Team.ToString() };
            TableCell contenderKills = new TableCell
            {
                Text = Frags.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell contenderDeaths = new TableCell
            {
                Text = Mortality.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell contenderGun = new TableCell { Text = Weapon.ToString() };
            row.Cells.Add(contenderName);
            row.Cells.Add(contenderSurname);
            row.Cells.Add(contenderTeam);
            row.Cells.Add(contenderKills);
            row.Cells.Add(contenderDeaths);
            row.Cells.Add(contenderGun);
            return row;
        }

        /// <summary>
        /// TableRow with for CS players with best KD
        /// </summary>
        /// <returns>TableRow</returns>
        public TableRow GetTableRowForBestKD()
        {
            TableRow row = new TableRow();
            TableCell contenderName = new TableCell { Text = Name.ToString() };
            TableCell contenderSurname = new TableCell { Text = Surname.ToString()
};
            TableCell contenderTeam = new TableCell { Text = Team.ToString() };
            row.Cells.Add(contenderName);
            row.Cells.Add(contenderSurname);
            row.Cells.Add(contenderTeam);
            return row;
        }
        /// <summary>
        /// TableRow for teams
        /// </summary>
        /// <param name="teams">teams</param>
        /// <returns>TableRow</returns>
        public TableRow GetTableRowForTeams(string teams)
        {
            TableRow row = new TableRow();
            TableCell contenderTeam = new TableCell { Text = teams.ToString() };
            row.Cells.Add(contenderTeam);
            return row;
        }
        /// <summary>
        /// TableHeaderRow for teams
        /// </summary>
        /// <returns>TableHeaderRow</returns>
        public static TableHeaderRow GetInitialTableHeaderRowForTeams()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            row.Cells.Add(team);
            return row;
        }


        /// <summary>
        /// TableHeaderRow for initial CS player data
        /// </summary>
        /// <returns>TableHeaderRow</returns>
        public static TableHeaderRow GetInitialTableHeaderRowForCSPlayers()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell name = new TableHeaderCell { Text = "Vardas" };
            TableHeaderCell surname = new TableHeaderCell { Text = "Pavardė" };
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            TableHeaderCell kills = new TableHeaderCell { Text = "Nužudymai" };
            TableHeaderCell deaths = new TableHeaderCell { Text = "Mirtys" };
            TableHeaderCell gun = new TableHeaderCell { Text = "Mėgstamiausias 
ginklas" };
            row.Cells.Add(name);
            row.Cells.Add(surname);
            row.Cells.Add(team);
            row.Cells.Add(kills);
            row.Cells.Add(deaths);
            row.Cells.Add(gun);
            return row;
        }

        /// <summary>
        /// TableHeaderRow for best KD
        /// </summary>
        /// <returns>TableHeaderRow</returns>
        public static TableHeaderRow GetInitialTableHeaderRowForBestKD()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell name = new TableHeaderCell { Text = "Vardas" };
            TableHeaderCell surname = new TableHeaderCell { Text = "Pavardė" };
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            row.Cells.Add(name);
            row.Cells.Add(surname);
            row.Cells.Add(team);
            return row;
        }


        public static bool operator <(CSPlayer lhs, CSPlayer rhs)
        {
            return lhs.MostRation() < rhs.MostRation();
        }

        public static bool operator >(CSPlayer lhs, CSPlayer rhs)
        {
            return lhs.MostRation() > rhs.MostRation();
        }


        /// <summary>
        /// Abstract method for finding KD ration
        /// </summary>
        /// <returns>KD ratio</returns>
        public override double MostRation()
        {
            return Frags * 1.0 / Mortality * 1.0;
        }

        /// <summary>
        /// CompareTo method for comparing by KD ratio
        /// </summary>
        /// <param name="other">other KD ratio</param>
        /// <returns>compared by KD ratio</returns>
        public int CompareTo(CSPlayer other)
        {
            if (MostRation().CompareTo(other.MostRation()) > 0)
                return 1;
            else if (MostRation().CompareTo(other.MostRation()) == 0)
                return 0;
            else
                return -1;
        }

        /// <summary>
        /// Equals method for CS players data
        /// </summary>
        /// <param name="other">Other cs players data</param>
        /// <returns>Equal CS player data</returns>
        public bool Equals(CSPlayer other)
        {
            if (other is null)
                return false;
            return Frags == other.Frags && Mortality == other.Mortality && Weapon
== other.Weapon;
        }
    }
}
```

LOLPlayer.cs

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

namespace L4s.App_Code1
{
    /// <summary>
    /// LOLPlayer data class
    /// </summary>
    public class LOLPlayer : Player , IComparable<LOLPlayer>,
IEquatable<LOLPlayer>, Forma1.ITableHeaders
    {
        public string Champion { get; private set; }
        public int Kills { get; private set; }
        public int Deaths { get; private set; }
        public int Assists { get; private set; }
        /// <summary>
        /// Data class constructor
        /// </summary>
        /// <param name="name">Name</param>
        /// <param name="surname">Surname</param>
        /// <param name="team">Team</param>
        /// <param name="position">Position</param>
        /// <param name="champion">Champion</param>
        /// <param name="k">Kills</param>
        /// <param name="d">Deaths</param>
        /// <param name="a">Assists</param>
        public LOLPlayer(string name, string surname, string team, string
position, string champion, int k, int d, int a) : base(name, surname, team)
        {
            this.Position = position;
            this.Champion = champion;
            this.Kills = k;
            this.Deaths = d;
            this.Assists = a;
        }
        public override string ToString()
        {
            return String.Format(base.ToString() + " {0, -12} | {1, 25} | {2, 10}
| {3, 10} | {4, 10} | {5, 31} |", Position, Champion, Kills, Deaths, Assists, "");
        }

        public static bool operator <(LOLPlayer lhs, LOLPlayer rhs)
        {
            return lhs.MostRation() < rhs.MostRation();
        }

        public static bool operator >(LOLPlayer lhs, LOLPlayer rhs)
        {
            return lhs.MostRation() > rhs.MostRation();
        }

        /// <summary>
        /// TableRow for teams
        /// </summary>
        /// <param name="list">teams</param>
        /// <returns>TableRow</returns>
        public TableRow GetTableRowForTeams(string list)
        {
            TableRow row = new TableRow();
            TableCell contenderTeam = new TableCell { Text = list.ToString() };
            row.Cells.Add(contenderTeam);
            return row;
        }

        /// <summary>
        /// TableHeaderRow for teams
        /// </summary>
        /// <returns>TableHeaderRow</returns>
        public static TableHeaderRow GetInitialTableHeaderRowForTeams()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            row.Cells.Add(team);
            return row;
        }

        /// <summary>
        /// TableRow for LOLPlayer data
        /// </summary>
        /// <returns>TableRow</returns>
        public override TableRow GetTableRow()
        {
            TableRow row = new TableRow();
            TableCell contenderName = new TableCell { Text = Name.ToString() };
            TableCell contenderSurname = new TableCell { Text = Surname.ToString()
};
            TableCell contenderTeam = new TableCell { Text = Team.ToString() };
            TableCell contenderPosition = new TableCell { Text =
Position.ToString() };
            TableCell contenderChampion = new TableCell { Text =
Champion.ToString() };
            TableCell contenderKills = new TableCell
            {
                Text = Kills.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell contenderDeaths = new TableCell
            {
                Text = Deaths.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };
            TableCell contenderAssists = new TableCell
            {
                Text = Assists.ToString(),
                HorizontalAlign = HorizontalAlign.Right
            };

            row.Cells.Add(contenderName);
            row.Cells.Add(contenderSurname);
            row.Cells.Add(contenderTeam);
            row.Cells.Add(contenderPosition);
            row.Cells.Add(contenderChampion);
            row.Cells.Add(contenderKills);
            row.Cells.Add(contenderDeaths);
            row.Cells.Add(contenderAssists);
            return row;
        }

        /// <summary>
        /// TableRow for Most KDA ratio
        /// </summary>
        /// <returns>TableRow</returns>
        public TableRow GetTableRowForMostKDA()
        {
            TableRow row = new TableRow();
            TableCell contenderName = new TableCell { Text = Name.ToString() };
            TableCell contenderSurname = new TableCell { Text = Surname.ToString()
};
            TableCell contenderTeam = new TableCell { Text = Team.ToString() };
            row.Cells.Add(contenderName);
            row.Cells.Add(contenderSurname);
            row.Cells.Add(contenderTeam);
            return row;
        }

        /// <summary>
        /// TableHeaderRow for Initial LOL player data
        /// </summary>
        /// <returns>TableHeaderRow</returns>
        public static TableHeaderRow GetInitialTableHeaderRowForLOLPlayers()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell name = new TableHeaderCell { Text = "Vardas" };
            TableHeaderCell surname = new TableHeaderCell { Text = "Pavardė" };
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            TableHeaderCell position = new TableHeaderCell { Text = "Pozicija" };
            TableHeaderCell champion = new TableHeaderCell { Text = "Čempionas" };
            TableHeaderCell kills = new TableHeaderCell { Text = "Nužudymai" };
            TableHeaderCell deaths = new TableHeaderCell { Text = "Mirtys" };
            TableHeaderCell assists = new TableHeaderCell { Text = "Dalyvavimai
nužudymuose" };
            row.Cells.Add(name);
            row.Cells.Add(surname);
            row.Cells.Add(team);
            row.Cells.Add(position);
            row.Cells.Add(champion);
            row.Cells.Add(kills);
            row.Cells.Add(deaths);
            row.Cells.Add(assists);
            return row;
        }

        /// <summary>
        /// TableHeaderRow for most KDA
        /// </summary>
        /// <returns>TableHeaderRow</returns>
        public static TableHeaderRow GetInitialTableHeaderRowForMostKDA()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell name = new TableHeaderCell { Text = "Vardas" };
            TableHeaderCell surname = new TableHeaderCell { Text = "Pavardė" };
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            row.Cells.Add(name);
            row.Cells.Add(surname);
            row.Cells.Add(team);
            return row;
        }


        /// <summary>
        /// Abstract method for finding KDA ratio
        /// </summary>
        /// <returns>KDA ratio</returns>
        public override double MostRation()
        {
            return (Kills + Assists) * 1.0 / Deaths * 1.0;
        }

        /// <summary>
        /// CompareTo method for KDA ratio
        /// </summary>
        /// <param name="other">other KDA ratio</param>
        /// <returns>Compared KDA ratio</returns>
        public int CompareTo(LOLPlayer other)
        {
            if (MostRation().CompareTo(other.MostRation()) > 0)
                return 1;
            else if (MostRation().CompareTo(other.MostRation()) == 0)
                return 0;
            else
                return -1;
        }
        /// <summary>
        /// Equals method for LOLPlayers data
        /// </summary>
        /// <param name="other">Other LOL player data</param>
        /// <returns>Equal LOL player data</returns>
        public bool Equals(LOLPlayer other)
        {
            if (other is null)
                return false;
            return Position == other.Position && Champion == other.Champion &&
Kills == other.Kills &&  Deaths== other.Deaths && Assists == other.Assists;
        }
    }
}
```

WebForm2.cs

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

namespace L4s.App_Code1
{
    public partial class WebForm2 : System.Web.UI.Page
    {
        public static TableRow MakeRowInitial(string cell1)
        {
            TableRow row = new TableRow();
            TableCell Cell1 = new TableCell();
            Cell1.Text = cell1;
            row.Cells.Add(Cell1);
            return row;
        }

        public static TableRow MakeRow(string cell1, string cell2)
        {
            TableRow row = new TableRow();
            TableCell Cell1 = new TableCell();
            Cell1.Text = cell1;
            row.Cells.Add(Cell1);
            TableCell Cell2 = new TableCell();
            Cell2.Text = cell2;
            row.Cells.Add(Cell2);
            return row;
        }
    }
}
```

InitialTournamentData.cs

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

```
namespace L4s.App_Code1
{
    /// <summary>
    /// Class for initial tournament data
    /// </summary>
    public class InitialTournamentData : Forma1.ITableHeaders
    {
        public int Tournament { get; set; }
        public DateTime Date { get; set; }
        public List<Player> newList = new List<Player>();

        /// <summary>
        /// InitialTournamentData constructor
        /// </summary>
        /// <param name="tournament">Tournament number</param>
        /// <param name="date">Tournament date</param>
        public InitialTournamentData(int tournament, DateTime date,
List<Player>list)
        {
            this.Tournament = tournament;
            this.Date = date;
            this.newList = list;
        }

        public InitialTournamentData(int tournament, DateTime date)
        {
            this.Tournament = tournament;
            this.Date = date;
        }


        /// <summary>
        /// Method for getting TableRow for InitialTournamentData
        /// </summary>
        /// <returns>TableRow of initialtournament data</returns>
        public TableRow GetTableRow()
        {
            string mainData = "";
            for (int i = 0; i < newList.Count(); i++)
            {
                mainData = newList[i].ToString();
            }
            TableRow row = new TableRow();
            TableCell tournamentNumber = new TableCell
            {
                Text = Tournament.ToString(),
                HorizontalAlign
             = HorizontalAlign.Right
            };

            TableCell tournamentDate = new TableCell
            {
                Text = string.Format("{0,25:yyyy/MM/dd}", Date),
                HorizontalAlign
             = HorizontalAlign.Right
            };

            TableCell tournament = new TableCell
            {


                Text = string.Format("|{0}|", mainData),
                HorizontalAlign = HorizontalAlign.Right
            };
            row.Cells.Add(tournamentNumber);
            row.Cells.Add(tournamentDate);
            row.Cells.Add(tournament);
            return row;
        }

        /// <summary>
        /// Method for getting initial table header row for InitialTournamentData
        /// </summary>
        /// <returns>TableHeaderRow</returns>
        public static TableHeaderRow GetInitialTableHeaderRowForDateAndNumber()
        {
            TableHeaderRow row = new TableHeaderRow();
            TableHeaderCell number = new TableHeaderCell { Text = "Numeris" };
            TableHeaderCell date = new TableHeaderCell { Text = "Data" };
            TableHeaderCell name = new TableHeaderCell { Text = "Vardas" };
            TableHeaderCell surname = new TableHeaderCell { Text = "Pavarde" };
            TableHeaderCell team = new TableHeaderCell { Text = "Komanda" };
            TableHeaderCell position = new TableHeaderCell { Text = "Pozicija" };
            TableHeaderCell champion = new TableHeaderCell { Text = "Čempionas" };
            TableHeaderCell kills = new TableHeaderCell { Text = "Nužudymai" };
            TableHeaderCell deaths = new TableHeaderCell { Text = "Mirtys" };
            TableHeaderCell assists = new TableHeaderCell { Text = "Pagalba" };
            TableHeaderCell  gun = new TableHeaderCell { Text = "Ginklai" };
            row.Cells.Add(number);
            row.Cells.Add(date);
            row.Cells.Add(name);
            row.Cells.Add(surname);
            row.Cells.Add(team);
            row.Cells.Add(position);
            row.Cells.Add(champion);
            row.Cells.Add(kills);
            row.Cells.Add(deaths);
            row.Cells.Add(assists);
            row.Cells.Add(gun);
            return row;
        }

    }
}
```

LinkedPlayerList.cs

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

namespace L4s.App_Code1
{
    /// <summary>
    /// LinkedPlayerList class
    /// </summary>
    /// <typeparam name="T">Type of LinkedPlayerList data</typeparam>
    public sealed class LinkedPlayerList<T> : IEnumerable<T>
 where T : IComparable<T>, Forma1.ITableHeaders
    {
        [Serializable]
        private sealed class Node
        {
            public T Data;
            public Node Link;
            public Node(T data, Node address)
            {
```

```
            Data = data;
            Link = address;
        }
    }
    private Node Head;
    private Node Tail;
    private Node Current;
    public int Count
    {
        get
        {
            int c = 0;
            foreach (T t in this)
            {
                c++;
            }
            return c;
        }
        private set { }
    }
    public bool IsReadOnly => false;
    /// <summary>
    /// Constructor with empty parameters.
    /// </summary>
    public LinkedPlayerList()
    {
        Head = null;
        Tail = null;
        Current = null;
    }
    /// <summary>
    /// Returns the selected node data
    /// </summary>
    /// <returns></returns>
    public T Get()
    {
        return Current.Data;
    }
    /// <summary>
    /// Sets the beginning of the linked list
    /// </summary>
    public void Begin()
    {
        Current = Head;
    }
    /// <summary>
    /// Sets the next node of the linked list
    /// </summary>
    public void Next()
    {
        if (Current != null)
            Current = Current.Link;
        else
            Current = Head.Link;
    }
    /// <summary>
    /// Checks if node exists
    /// </summary>
    public bool Exist()
    {
        return Current != null;
    }
```

```
        /// <summary>
        /// Inserts a new data object to the end of the linked list.
        /// </summary>
        /// <param name="data">
        /// Any object of the same type as this list.
        /// </param>
        public void Add(T data)
        {
            try
            {
                Node node = new Node(data, null);
                if (Head != null)
                {
                    Tail.Link = node;
                    Tail = node;
                }
                else
                {
                    Head = node;
                    Tail = node;
                }
            }


            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }
        }
        /// <summary>
        /// Sort the linked list with selection sort.
        /// </summary>
        public void Sort()
        {
            try
            {
                for (Node i = Head; i != null; i = i.Link)
                {
                    Node max = i;
                    for (Node j = i.Link; j != null; j = j.Link)
                    {
                        if (j.Data.CompareTo(max.Data) > 0)
                            max = j;
                    }

                    (max.Data, i.Data) = (i.Data, max.Data);
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }
        }

        /// <summary>
        /// Method for foreach loop
        /// </summary>
        IEnumerator<T> IEnumerable<T>.GetEnumerator()
        {
            for (
            Node current = Head;
            current != null;
            current = current.Link)
```

```
                yield return current.Data;
        }
        /// <summary>
        /// Method for foreach loop
        /// </summary>
        public IEnumerator GetEnumerator()
        {
            for (
            Node current = Head;
            current != null;
            current = current.Link)
                yield return current.Data;
        }
        /// <summary>
        /// Clears the LinkList
        /// </summary>
        public void Clear()
        {
            Head = null;
            Tail = null;
        }
        /// <summary>
        /// Searches the LinkList and if the specified
        /// item is already in the list,
        /// returns true, otherwise false.
        /// </summary>
        /// <param name="item">Item to search for.</param>
        /// <returns>Returns a boolean value.</returns>
        public bool Contains(T item)
        {
            for (Begin(); Exist(); Next())
            {
                if (Current.Data.Equals(item))
                    return true;
            }
            return false;
        }
    }

}
```

InOut.cs

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

namespace L4s.App_Code1
{
    /// <summary>
    /// Class for inputing and outputting data
    /// </summary>
    public class InOut
    {

    /// <summary>
    /// Method for reading files from a directory and putting all data to data
structures
    /// </summary>
    /// <param name="fileName">The directory with files</param>
    /// <param name="csplayerData">CSPlayer data</param>
    /// <param name="lolplayerData">LOLPlayer data</param>
    /// <param name="initialData">Initial tournament data</param>
    public static void ReadParticipantsMainData(string fileName,
LinkedPlayerList<CSPlayer> csplayerData, LinkedPlayerList<LOLPlayer>
lolplayerData, List<InitialTournamentData>initialData)
        {
            try
            {
                    foreach (string txtName in Directory.GetFiles(fileName,
"turnyras*.txt"))
                    {
                        string[] Lines = File.ReadAllLines(txtName);
                        int tournament = int.Parse(Lines[0]);
                        DateTime date = DateTime.Parse(Lines[1]);
                        InitialTournamentData data = new
InitialTournamentData(tournament, date);
                        initialData.Add(data);
                        foreach (string line in Lines.Skip(2))
                        {
                            string[] values = line.Split(';');
                            bool LeagueOfLegends = values.Length == 8 ? true :
false;
                            string name = values[0];
                            string surname = values[1];
                            string team = values[2];
                            if (LeagueOfLegends)
                            {
                                string position = values[3];
                                string champion = values[4];
                                int k = Convert.ToInt32(values[5]);
                                int d = Convert.ToInt32(values[6]);
                                int a = Convert.ToInt32(values[7]);
                                LOLPlayer player = new LOLPlayer(name, surname,
team, position, champion, k, d, a);
                                lolplayerData.Add(player);
                            }
                            else
                            {
                                int kills = Convert.ToInt32(values[3]);
                                int deaths = Convert.ToInt32(values[4]);
                                string favorite = values[5];
                                CSPlayer player = new CSPlayer(name, surname,
team, kills, deaths, favorite);
                                csplayerData.Add(player);
                            }
                        }

                    }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }
        }
/// <summary>
        /// Method for reading allplayers data from a given directory
        /// </summary>
        /// <param name="fileName">Given directory</param>
        /// <returns>List with player data</returns>
        public static List<InitialTournamentData>
ReadParticipantsMainDataPlayer(string fileName)
        {
            try
            {
                List<InitialTournamentData> initial = new
List<InitialTournamentData>();
                foreach (string txtName in Directory.GetFiles(fileName,
"turnyras*.txt"))
                {

                    List<Player> newList = new List<Player>();
                    string[] Lines = File.ReadAllLines(txtName);
                    int tournament = int.Parse(Lines[0]);
                    DateTime date = DateTime.Parse(Lines[1]);

                    foreach (string line in Lines.Skip(2))
                    {
                        string[] values = line.Split(';');
                        bool LeagueOfLegends = values.Length == 8 ? true : false;
                        string name = values[0];
                        string surname = values[1];
                        string team = values[2];
                        if (LeagueOfLegends)
                        {
                            string position = values[3];
                            string champion = values[4];
                            int k = Convert.ToInt32(values[5]);
                            int d = Convert.ToInt32(values[6]);
                            int a = Convert.ToInt32(values[7]);
                            LOLPlayer player = new LOLPlayer(name, surname, team,
position, champion, k, d, a);
                            newList.Add(player);
                        }
                        else
                        {
                            int kills = Convert.ToInt32(values[3]);
                            int deaths = Convert.ToInt32(values[4]);
                            string favorite = values[5];
                            CSPlayer player = new CSPlayer(name, surname, team,
kills, deaths, favorite);
                            newList.Add(player);
                        }
                    }
                    InitialTournamentData data = new
InitialTournamentData(tournament, date, newList);
                    initial.Add(data);


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


/// <summary>
        /// Method for printing table elements to a txt file
        /// </summary>
        /// <param name="fileName"></param>
        /// <param name="data"></param>
        public static void PrintToTxt(string fileName, InitialTournamentData data)
        {
            using (StreamWriter write = File.AppendText(fileName))
            {
                write.WriteLine(data.Tournament);
                write.WriteLine(string.Format("{0:yyyy-MM-dd}", data.Date));
                for (int i = 0; i < data.newList.Count(); i++)
                {
                    write.WriteLine(new string('-', 176));
                    write.WriteLine(data.newList[i].ToString());

                }
                write.WriteLine(new string('-', 176));
                write.WriteLine();
            }
        }
       /// <summary>
       /// Method for printing data table
       /// </summary>
       /// <typeparam name="T">Type data table</typeparam>
       /// <param name="table">Table element</param>
       /// <param name="list">The Linked list data</param>
       /// <param name="headerRow">The headerrow of the table</param>
       public static void PrintInitialTable<T>(Table table, LinkedPlayerList<T>
list, TableHeaderRow headerRow)
       where T : IComparable<T>, Forma1.ITableHeaders
        {
            try
            {
                table.Rows.Add(headerRow);
                foreach (T data in list)
                {
                    table.Rows.Add(data.GetTableRow());
                }
            }
            catch(Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }
        }

        /// <summary>
        /// Method for printing InitialTournament data to a file
        /// </summary>
        /// <param name="fileName">The result file</param>
        /// <param name="initialCSDataList">CSPlayer data</param>
        /// <param name="initialLOLDataList">LOLPlayer data</param>
        /// <param name="tournamentData">Initial tournament data</param>
        /// <param name="header">the additional information</param>
        public static void PrintToResultFileInitialData(string fileName,
LinkedPlayerList<CSPlayer> initialCSDataList, LinkedPlayerList<LOLPlayer>
initialLOLDataList, List<InitialTournamentData> tournamentData, string header)
        {
            try
            {
                using (StreamWriter write = File.CreateText(fileName))
                {
                    string dashescs = new string('-', 127);
                    string dasheslol = new string('-', 169);
                    write.WriteLine(header);
                    write.WriteLine();
                    foreach (var data in tournamentData)
                    {
                        write.WriteLine(data.Tournament);
                        write.WriteLine(String.Format("{0:yyyy-MM-dd}", 
data.Date));
                    }
                    write.WriteLine("");
                    write.WriteLine(dashescs);
                    write.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-
20}|{3,20}|{4,20}|{5,-20}|", "Vardas", "Pavardė", "Komanda", "Nužudymai", 
"Mirtys", "Ginklas"));
                    for (initialCSDataList.Begin(); initialCSDataList.Exist(); 
initialCSDataList.Next())
                    {
                        var dd = initialCSDataList.Get();
                        write.WriteLine(dashescs);
                        write.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-
20}|{3,20}|{4,20}|{5,-20}|", dd.Name, dd.Surname, dd.Team, dd.Frags, dd.Mortality, 
dd.Weapon));

                    }
                    write.WriteLine(dashescs);
                    write.WriteLine("");
                    write.WriteLine(dasheslol);
                    write.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|{3,-
20}|{4,-20}|{5,20}|{6,20}|{7,20}|", "Vardas", "Pavardė", "Komanda", "Pozicija", 
"Čempionas", "Nužudymai", "Mirtys", "Assistai"));

                    for (initialLOLDataList.Begin(); initialLOLDataList.Exist(); 
initialLOLDataList.Next())
                    {
                        var cc = initialLOLDataList.Get();
                        write.WriteLine(dasheslol);
                        write.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-
20}|{3,-20}|{4,-20}|{5,20}|{6,20}|{7,20}|", cc.Name, cc.Surname, cc.Team, 
cc.Position, cc.Champion, cc.Kills, cc.Deaths, cc.Assists));

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

        }



        /// <summary>
        /// Method for printing the most KD ratio data to a file
        /// </summary>
        /// <param name="fileName">The result file</param>
        /// <param name="bestKD">Best CS Player data</param>
        /// <param name="bestKDA">Best LOL Player data</param>
        /// <param name="header1">additional information about CS players</param>
        /// <param name="header2">additional information about LOL players</param>
        public static void PrintMostRatio(string fileName, 
LinkedPlayerList<CSPlayer>bestKD, LinkedPlayerList<LOLPlayer>bestKDA, string 
header1, string header2)
        {
            try
            {
                using (StreamWriter sw = File.AppendText(fileName))
                {
                    string dashes = new string('-', 64);
                    sw.WriteLine(header1);
                    sw.WriteLine(dashes);
                    sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|",
"Vardas", "Pavardė", "Komanda"));
                    sw.WriteLine(dashes);
                    for (bestKD.Begin(); bestKD.Exist(); bestKD.Next())
                    {
                        var dd = bestKD.Get();
                        sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|",
dd.Name, dd.Surname, dd.Team));
                    }
                    sw.WriteLine(dashes);
                    sw.WriteLine();

                    sw.WriteLine(header2);
                    sw.WriteLine(dashes);
                    sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|",
"Vardas", "Pavardė", "Komanda"));
                    sw.WriteLine(dashes);
                    for (bestKDA.Begin(); bestKDA.Exist(); bestKDA.Next())
                    {
                        var ff = bestKDA.Get();
                        sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|",
ff.Name, ff.Surname, ff.Team));
                    }
                    sw.WriteLine(dashes);
                    sw.WriteLine();
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }
        /// <summary>
        /// Method for printing players data by selected criteria to file
        /// </summary>
        /// <param name="fileName">file</param>
        /// <param name="csData">CS Player data</param>
        /// <param name="lolData">LOL Player data</param>
        /// <param name="header1">CS player additional information</param>
        /// <param name="header2">LOL player additional information</param>
        /// <param name="criteria">criteria number</param>
        public static void PrintPlayersByCriteria(string fileName,
LinkedPlayerList<CSPlayer> csData, LinkedPlayerList<LOLPlayer> lolData, string
header1, string header2, double criteria)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                string dashes = new string('-', 127);
                string dasheslol = new string('-', 169);
                sw.WriteLine(string.Format("Pasirinktas rodiklis {0}", criteria));
                sw.WriteLine();
                sw.WriteLine(header1);
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-
20}|{3,20}|{4,20}|{5,-20}|", "Vardas", "Pavardė", "Komanda", "Nužudymai",
"Mirtys", "Ginklas"));
                sw.WriteLine(dashes);
                for (csData.Begin(); csData.Exist(); csData.Next())
                {
                    var dd = csData.Get();
                    sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-
20}|{3,20}|{4,20}|{5,-20}|", dd.Name, dd.Surname, dd.Team, dd.Frags, dd.Mortality,
dd.Weapon));
                }
                sw.WriteLine(dashes);
                sw.WriteLine();
                sw.WriteLine(header2);
                sw.WriteLine(dasheslol);
                sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|{3,-20}|{4,-
20}|{5,20}|{6,20}|{7,20}|", "Vardas", "Pavardė", "Komanda", "Pozicija",
"Čempionas", "Nužudymai", "Mirtys", "Assistai"));
                sw.WriteLine(dasheslol);
                for (lolData.Begin(); lolData.Exist(); lolData.Next())
                {
                    var cc = lolData.Get();
                    sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|{3,-
20}|{4,-20}|{5,20}|{6,20}|{7,20}|", cc.Name, cc.Surname, cc.Team, cc.Position,
cc.Champion, cc.Kills, cc.Deaths, cc.Assists));
                }
                sw.WriteLine(dasheslol);
                sw.WriteLine();
            }
        }

        /// <summary>
        /// Method for printing players data by selected criteria to CSV file
        /// </summary>
        /// <param name="fileName">CSV file</param>
        /// <param name="csData">CS Player data</param>
        /// <param name="lolData">LOL Player data</param>
        /// <param name="header1">additional information about CS players</param>
        /// <param name="header2">additional information about LOL players</param>
        public static void PrintPlayersToCSVByCriteria(string fileName,
LinkedPlayerList<CSPlayer> csData, LinkedPlayerList<LOLPlayer> lolData, string
header1, string header2)
        {
            try
            {
                using (StreamWriter write = File.CreateText(fileName))
                {
                    write.WriteLine(header1);
                    for (csData.Begin(); csData.Exist(); csData.Next())
                    {
                        var dd = csData.Get();
                        write.WriteLine("{0};{1};{2};{3};{4};{5}", dd.Name,
dd.Surname, dd.Team, dd.Frags, dd.Mortality, dd.Weapon);
                    }
                    write.WriteLine(header2);
                    for (lolData.Begin(); lolData.Exist(); lolData.Next())
                    {
                        var cc = lolData.Get();
                        write.WriteLine("{0};{1};{2};{3};{4};{5};{6};{7}",
cc.Name, cc.Surname, cc.Team, cc.Position, cc.Champion, cc.Kills, cc.Deaths,
cc.Assists);
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }
        }

        /// <summary>
        /// Method for printing all tournament teams to CSV file
        /// </summary>
        /// <param name="fileName">CSV file</param>
        /// <param name="list">List with teams</param>
        /// <param name="header">additional information</param>
        public static void PrintTeamsToCSV(string fileName, List<string> list,
string header)
        {
            try
            {
                using (StreamWriter sw = File.CreateText(fileName))
                {
                    sw.WriteLine(header);
                    foreach (string data in list)
                    {
                        sw.WriteLine(data);
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }


        }
        /// <summary>
        /// Method for printing teams to a file
        /// </summary>
        /// <param name="fileName">File</param>
        /// <param name="list">List with teams</param>
        /// <param name="header">additional information</param>
        public static void PrintTeamsToResultFile(string fileName,
List<string>list, string header)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                sw.WriteLine(header);
                foreach(string data in list)
                {
                    sw.WriteLine(data);

                }
                sw.WriteLine();
            }
        }
        /// <summary>
        /// Method for printing all players data sorted
        /// </summary>
        /// <param name="fileName">File</param>
        /// <param name="csData">CS player data</param>
        /// <param name="lolData">LOL player data</param>
        /// <param name="header1">additional information</param>
        /// <param name="header2">additional information</param>
        public static void PrintAllTournamentPlayers(string fileName,
LinkedPlayerList<CSPlayer> csData, LinkedPlayerList<LOLPlayer> lolData, string
header1, string header2)
        {
            using (StreamWriter sw = File.AppendText(fileName))
            {
                string dashes = new string('-', 127);
                string dasheslol = new string('-', 169);
                sw.WriteLine(header1);
                sw.WriteLine(dashes);
                sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-
20}|{3,20}|{4,20}|{5,-20}|", "Vardas", "Pavardė", "Komanda", "Nužudymai",
"Mirtys", "Ginklas"));
                sw.WriteLine(dashes);
                for (csData.Begin(); csData.Exist(); csData.Next())
                {
                    var dd = csData.Get();
                    sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-
20}|{3,20}|{4,20}|{5,-20}|", dd.Name, dd.Surname, dd.Team, dd.Frags, dd.Mortality,
dd.Weapon));
                }
                sw.WriteLine(dashes);
                sw.WriteLine();
                sw.WriteLine(header2);
                sw.WriteLine(dasheslol);
                sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|{3,-20}|{4,-
20}|{5,20}|{6,20}|{7,20}|", "Vardas", "Pavardė", "Komanda", "Pozicija",
"Čempionas", "Nužudymai", "Mirtys", "Assistai"));
                sw.WriteLine(dasheslol);
                for (lolData.Begin(); lolData.Exist(); lolData.Next())
                {
                    var cc = lolData.Get();
                    sw.WriteLine(string.Format("|{0,-20}|{1,-20}|{2,-20}|{3,-
20}|{4,-20}|{5,20}|{6,20}|{7,20}|", cc.Name, cc.Surname, cc.Team, cc.Position,
cc.Champion, cc.Kills, cc.Deaths, cc.Assists));
                }
                sw.WriteLine(dasheslol);
                sw.WriteLine();
            }
        }

        /// <summary>
        /// Method for printing a table element with tournament number and date
        /// </summary>
        /// <param name="table">Table element</param>
        /// <param name="initialData">Initial data</param>
        public static void PrintTournamentNumberAndDate(Table table,
List<InitialTournamentData>initialData)
        {
            TableHeaderRow headerRow =
InitialTournamentData.GetInitialTableHeaderRowForDateAndNumber();
            table.Rows.Add(headerRow);
            foreach (InitialTournamentData data in initialData)
            {
                table.Rows.Add(data.GetTableRow());
            }
        }

        /// <summary>
        /// Method for printing Teams to a Table element
        /// </summary>
        /// <param name="table">Table element</param>
        /// <param name="list">List with players</param>
        public static void PrintTeams(Table table, List<string> list)
        {
            CSPlayer csPlayer = new CSPlayer("", "", "", 0, 0, "");
            TableHeaderRow headerRow =
CSPlayer.GetInitialTableHeaderRowForTeams();
            table.Rows.Add(headerRow);
            foreach(string data in list)
            {
                table.Rows.Add(csPlayer.GetTableRowForTeams(data));
            }
        }
        /// <summary>
        /// Method for printing most KD to a Table element
        /// </summary>
        /// <param name="table">Table element</param>
        /// <param name="filteredTeams">CS player filtered data</param>
        public static void PrintMostKD(Table table, LinkedPlayerList<CSPlayer>
filteredTeams)
        {
            TableHeaderRow headerRow =
CSPlayer.GetInitialTableHeaderRowForBestKD();
            table.Rows.Add(headerRow);
            foreach (CSPlayer data in filteredTeams)
            {
                table.Rows.Add(data.GetTableRowForBestKD());
            }
        }

        /// <summary>
        /// Method for printing most KDA to a Table element
        /// </summary>
        /// <param name="table">Table element</param>
        /// <param name="filteredTeams">LOL Player filtered data</param>
        public static void PrintMostKDA(Table table, LinkedPlayerList<LOLPlayer>
filteredTeams)
        {
            TableHeaderRow headerRow =
LOLPlayer.GetInitialTableHeaderRowForMostKDA();
            table.Rows.Add(headerRow);
            foreach (LOLPlayer data in filteredTeams)
            {
                table.Rows.Add(data.GetTableRowForMostKDA());
            }
        }
```

TaskUtils.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using L4s;

namespace L4s.App_Code1
{
    /// <summary>
    /// Class for operations with data
    /// </summary>
    public class TaskUtils
    {
        /// <summary>
        /// Method for filtering CSPlayers by criteria number
        /// </summary>
        /// <param name="criteriaNumber">The criteria number</param>
        /// <param name="csPlayerAllData">All CSPlayer data</param>
        /// <param name="filteredData">Filtered CSPlayer data</param>
        public static void FilterCSPlayersByCriteria(double criteriaNumber,
LinkedPlayerList<CSPlayer>csPlayerAllData, LinkedPlayerList<CSPlayer>filteredData)
        {
            try
            {
                for (csPlayerAllData.Begin(); csPlayerAllData.Exist();
csPlayerAllData.Next())
                {
                    var dd = csPlayerAllData.Get();
                    if (criteriaNumber > dd.Mortality)
                    {
                        filteredData.Add(dd);
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }
        }
        /// <summary>
        /// Method for filtering LOLPlayers by criteria number
        /// </summary>
        /// <param name="criteriaNumber">Criteria number</param>
        /// <param name="lolPlayerAllData">All of LOLPlayer data</param>
        /// <param name="filteredData">Filtered LOLPlayer data</param>
        public static void FilterLOLPlayersByCriteria(double criteriaNumber,
LinkedPlayerList<LOLPlayer> lolPlayerAllData, LinkedPlayerList<LOLPlayer>
filteredData)
        {
            try
            {
                for (lolPlayerAllData.Begin(); lolPlayerAllData.Exist();
lolPlayerAllData.Next())
                {
                    var dd = lolPlayerAllData.Get();
                    if (criteriaNumber < dd.Kills)
                    {
                        filteredData.Add(dd);
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }

        }

        /// <summary>
        /// Method for forming a list with all the teams who participated in the
tournament
        /// </summary>
        /// <param name="csplayerData">CSPlayer data</param>
        /// <param name="list">The list with all teams who participated</param>
        /// <param name="lolplayerData">LOLPlayer data</param>
        public static void FormTeamList (LinkedPlayerList<CSPlayer> csplayerData,
List<string> list, LinkedPlayerList<LOLPlayer>lolplayerData)
        {
            try
            {
                foreach (CSPlayer player in csplayerData)
                {
                    foreach (LOLPlayer player1 in lolplayerData)
                    {
                        if (!list.Contains(player.Team))
                        {
                            list.Add(player.Team);
                        }
                        else if (!list.Contains(player1.Team))
                        {
                            list.Add(player1.Team);
                        }
                    }

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

        }

        /// <summary>
        /// Method for finding the Maximum KD ratio for CSPlayers
        /// </summary>
        /// <param name="allCSData">CSPlayers data</param>
        /// <returns>the max KD ratio</returns>
        public static double MaxKD(LinkedPlayerList<CSPlayer>allCSData)
        {
            double max = 0;
            try
            {
                for (allCSData.Begin(); allCSData.Exist(); allCSData.Next())
                {

                    if (allCSData.Get().MostRation() >= max)
                    {
                        max = allCSData.Get().MostRation();

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

        }
        /// <summary>
        /// Method for finding the most KDA ratio for LOLPlayers
        /// </summary>
        /// <param name="allLOLData">LOLPlayer data</param>
        /// <returns>The maximum KDA ratio</returns>
        public static double MaxKDA(LinkedPlayerList<LOLPlayer> allLOLData)
        {
            double max = 0;
            try
            {
                for (allLOLData.Begin(); allLOLData.Exist(); allLOLData.Next())
                {

                    if (allLOLData.Get().MostRation() >= max)
                    {
                        max = allLOLData.Get().MostRation();
                    }
                }
                return max;
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }

        }

        /// <summary>
        /// Method for finding the maximum ratio player data for CSPlayers
        /// </summary>
        /// <param name="secondvalue">The maximum KD ratio</param>
        /// <param name="allCSData">CSPlayers data</param>
        /// <param name="mostKD">CSPlayer with most KD</param>
        public static void MostRatioForCSGO(double secondvalue,
LinkedPlayerList<CSPlayer> allCSData, LinkedPlayerList<CSPlayer> mostKD)
        {
            try
            {
                for (allCSData.Begin(); allCSData.Exist(); allCSData.Next())
                {

                    if (allCSData.Get().MostRation() >= secondvalue)
                    {
                        secondvalue = allCSData.Get().MostRation();
                        if (!mostKD.Contains(allCSData.Get()))
                        {
                            mostKD.Add(allCSData.Get());
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }

        }

        /// <summary>
        /// Method for finding the most KDA players information
        /// </summary>
        /// <param name="secondvalue">The maximum KDA ratio</param>
        /// <param name="allLOLData">All LOLPlayers data</param>
        /// <param name="mostKDA">LOLPlayers data with most KDA</param>
        public static void MostRatioForLOL(double secondvalue,
LinkedPlayerList<LOLPlayer> allLOLData, LinkedPlayerList<LOLPlayer> mostKDA)
        {
            try
            {
                for (allLOLData.Begin(); allLOLData.Exist(); allLOLData.Next())
                {

                    if (allLOLData.Get().MostRation() >= secondvalue)
                    {
                        secondvalue = allLOLData.Get().MostRation();
                        if (!mostKDA.Contains(allLOLData.Get()))
                        {
                            mostKDA.Add(allLOLData.Get());
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                throw new Exception(string.Format("Method {0}, Message {1}, Source
{2}", ex.TargetSite, ex.Message, ex.Source));
            }

        }
    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="L4s.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="App_Code1/StyleSheet1.css" rel="stylesheet" />
</head>
<body>
    <form id="form1" runat="server" enctype="multipart/form-data">
        <br />
        <asp:Label ID="Label1" runat="server" Text="Label"
Visible="False"></asp:Label>
        <p>
            <asp:Button ID="Button2" runat="server" Text="Pradiniai duomenys"
OnClick="Button2_Click" />
        </p>
        <asp:Table ID="Table1" runat="server" BorderColor="Black"
BorderStyle="Solid" GridLines="Both">
        </asp:Table>
        <br />
        <p>
            <asp:Button ID="Button3" runat="server" Text="Geriausi KD/KDA"
OnClick="Button3_Click" />
        </p>
        <asp:Table ID="Table2" runat="server" BorderColor="Black"
BorderStyle="Solid" GridLines="Both">
        </asp:Table>
        <p>
            <asp:Button ID="Button4" runat="server" Text="Visos turnyro komandos"
OnClick="Button4_Click" />
        </p>
        <asp:Table ID="Table3" runat="server" BorderColor="Black"
BorderStyle="Solid" GridLines="Both">
        </asp:Table>
        <br />
            <asp:Button ID="Button5" runat="server" Text="Rinktine"
OnClick="Button5_Click" />
        <br />
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ControlToValidate="TextBox1" ErrorMessage="Blogi duomenys!"
ForeColor="Red" ValidationExpression="^-?(([1-9]\d*)|0)(.0*[1-9](0*[1-9])*)?$"
Display="Dynamic"></asp:RegularExpressionValidator>
        <asp:Table ID="Table4" runat="server" BorderColor="Black"
BorderStyle="Solid" GridLines="Both">
        </asp:Table>
        <p>
            &nbsp;</p>
        <p>
```

```
            <asp:Button ID="Button6" runat="server" Text="Viso turnyro žaidėjai 
išrikiuoti" OnClick="Button6_Click" />
        </p>
        <asp:Table ID="Table5" runat="server" BorderColor="Black" 
BorderStyle="Solid" GridLines="Both">
        </asp:Table>
    </form>
</body>
</html>
```

Forma1.aspx.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using L4s.App_Code1;
using System.ComponentModel;
using L4s;
using System.Drawing;

namespace L4s
{
    public partial class Forma1 : System.Web.UI.Page
    {
        public interface ITableHeaders
        {
            TableRow GetTableRow();
        }
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
            Label1.Visible = false;
            LinkedPlayerList<CSPlayer> allCSPlayerData = new 
LinkedPlayerList<CSPlayer>();
            LinkedPlayerList<LOLPlayer> allLOLPlayerData = new 
LinkedPlayerList<LOLPlayer>();
            string resultFile = Server.MapPath("App_Results1/Rezultatai.txt");
            List<InitialTournamentData> initialPlayerData = new 
List<InitialTournamentData>();
            if(File.Exists(resultFile))
            {
                File.Delete(resultFile);
            }
            File.AppendAllText(resultFile, "Pradiniai duomenys \n");
            try
            {
                foreach (InitialTournamentData player in 
InOut.ReadParticipantsMainDataPlayer(@"C:\File"))
                {
                    InOut.PrintToTxt(resultFile, player);
                    Table1.Rows.Add(WebForm2.MakeRowInitial(string.Format("{0} 
turnyro ratas", player.Tournament.ToString())));
                    Table1.Rows.Add(WebForm2.MakeRow(player.Tournament.ToString(), 
string.Format("{0:yyyy-MM-dd}", player.Date)));

Table1.Rows.Add(Player.GetInitialTableHeaderRowForAllPlayers());
                    foreach (Player data in player.newList)
                    {
                        if (data is CSPlayer)
                        {
                            Table1.Rows.Add(data.GetTableRow());
                        }
                        else if (data is LOLPlayer)
                        {
                            Table1.Rows.Add(data.GetTableRow());
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                Label1.Visible = true;
                Label1.Text = ex.Message;
                Label1.ForeColor = Color.Red;
            }
        }



        protected void Button3_Click(object sender, EventArgs e)
        {
            List<InitialTournamentData> initialPlayerData = new
List<InitialTournamentData>();
            LinkedPlayerList<CSPlayer> allCSPlayerData = new
LinkedPlayerList<CSPlayer>();
            LinkedPlayerList<LOLPlayer> allLOLPlayerData = new
LinkedPlayerList<LOLPlayer>();
            InOut.ReadParticipantsMainData(@"C:\File", allCSPlayerData,
allLOLPlayerData, initialPlayerData);
            double mostKD = TaskUtils.MaxKD(allCSPlayerData);
            double mostKDA = TaskUtils.MaxKDA(allLOLPlayerData);
            LinkedPlayerList<CSPlayer> mostKDPlayers = new
LinkedPlayerList<CSPlayer>();
            LinkedPlayerList<LOLPlayer> mostKDAPlayers = new
LinkedPlayerList<LOLPlayer>();
            TaskUtils.MostRatioForCSGO(mostKD, allCSPlayerData, mostKDPlayers);
            TaskUtils.MostRatioForLOL(mostKDA, allLOLPlayerData, mostKDAPlayers);
            InOut.PrintMostKD(Table2, mostKDPlayers);
            InOut.PrintMostKDA(Table2, mostKDAPlayers);
            string resultFile = Server.MapPath("App_Results1/Rezultatai.txt");
            InOut.PrintMostRatio(resultFile, mostKDPlayers, mostKDAPlayers,
"Geriausias KD", "Geriausias KDA");
        }

        protected void Button4_Click(object sender, EventArgs e)
        {
            List<InitialTournamentData> initialPlayerData = new
List<InitialTournamentData>();
            LinkedPlayerList<CSPlayer> allCSPlayerData = new
LinkedPlayerList<CSPlayer>();
            LinkedPlayerList<LOLPlayer> allLOLPlayerData = new
LinkedPlayerList<LOLPlayer>();
            InOut.ReadParticipantsMainData(@"C:\File", allCSPlayerData,
allLOLPlayerData, initialPlayerData);
            List<string> allTeams = new List<string>();
            TaskUtils.FormTeamList(allCSPlayerData, allTeams, allLOLPlayerData);
            InOut.PrintTeams(Table3, allTeams);
            string resultFile = Server.MapPath("App_Results1/Rezultatai.txt");
            string allPlayersSortedCSV =
Server.MapPath("App_Results1/Komandos.csv");
            InOut.PrintTeamsToCSV(allPlayersSortedCSV, allTeams, "All teams");
            InOut.PrintTeamsToResultFile(resultFile, allTeams, "Visos komandos");
        }
        protected void Button5_Click(object sender, EventArgs e)
        {
            List<InitialTournamentData> initialPlayerData = new
List<InitialTournamentData>();
            double comparisonNumber = double.Parse(TextBox1.Text.ToString());
            LinkedPlayerList<CSPlayer> allCSPlayerData = new
LinkedPlayerList<CSPlayer>();
            LinkedPlayerList<LOLPlayer> allLOLPlayerData = new
LinkedPlayerList<LOLPlayer>();
            InOut.ReadParticipantsMainData(@"C:\File", allCSPlayerData,
allLOLPlayerData, initialPlayerData);
            LinkedPlayerList<CSPlayer> CSfilteredData = new
LinkedPlayerList<CSPlayer>();
            LinkedPlayerList<LOLPlayer> LOLfilteredData = new
LinkedPlayerList<LOLPlayer>();
            TaskUtils.FilterCSPlayersByCriteria(comparisonNumber, allCSPlayerData,
CSfilteredData);
            TaskUtils.FilterLOLPlayersByCriteria(comparisonNumber,
allLOLPlayerData, LOLfilteredData);
            InOut.PrintInitialTable<LOLPlayer>(Table4, LOLfilteredData,
LOLPlayer.GetInitialTableHeaderRowForLOLPlayers());
            InOut.PrintInitialTable<CSPlayer>(Table4, CSfilteredData,
CSPlayer.GetInitialTableHeaderRowForCSPlayers());
            string resultFile = Server.MapPath("App_Results1/Rezultatai.txt");
            InOut.PrintPlayersByCriteria(resultFile, CSfilteredData,
LOLfilteredData, "CS žaidėjai", "LOL žaidėjai", comparisonNumber);
            string comparisonFile = Server.MapPath("App_Results1/Rinktine.csv");
            InOut.PrintPlayersToCSVByCriteria(comparisonFile, CSfilteredData,
LOLfilteredData, "CS players", "LOL players");
        }

        protected void Button6_Click(object sender, EventArgs e)
        {
            List<InitialTournamentData> initialPlayerData = new
List<InitialTournamentData>();
            LinkedPlayerList<CSPlayer> allCSPlayerData = new
LinkedPlayerList<CSPlayer>();
            LinkedPlayerList<LOLPlayer> allLOLPlayerData = new
LinkedPlayerList<LOLPlayer>();
            InOut.ReadParticipantsMainData(@"C:\File", allCSPlayerData,
allLOLPlayerData, initialPlayerData);
            allLOLPlayerData.Sort();
            allCSPlayerData.Sort();
            InOut.PrintInitialTable<LOLPlayer>(Table5, allLOLPlayerData,
LOLPlayer.GetInitialTableHeaderRowForLOLPlayers());
            InOut.PrintInitialTable<CSPlayer>(Table5, allCSPlayerData,
CSPlayer.GetInitialTableHeaderRowForCSPlayers());
            string resultFile = Server.MapPath("App_Results1/Rezultatai.txt");
            InOut.PrintAllTournamentPlayers(resultFile, allCSPlayerData,
allLOLPlayerData, "Visi CS žaidėjai išrušiuoti pagal KD", "Visi LOL žaidėjai
išrušiuoti pagal KDA");
            string allPlayersSortedCSV = Server.MapPath("App_Results1/Visi.csv");
            InOut.PrintPlayersToCSVByCriteria(allPlayersSortedCSV,
allCSPlayerData, allLOLPlayerData, "CS players sorted", "LOL players sorted");
        }
    }
}
```

StyleSheet1.css

```
body {
    background: #33CCCC;
    text-align: left;
    animation: color 5s infinite linear;
    padding: 2em;
}
p {
    font-family: "Lucida Console", "Brush Script MT", monospace;
}

@keyframes color {
    0% {
        background: #33CCCC;
    }

    20% {
        background: #33CC36;
    }

    40% {
        background: #B8CC33;
    }

    60% {
        background: #FCCA00;
    }

    80% {
        background: #33CC36;
    }

    100% {
        background: #33CCCC;
    }
}
```

4.7. Pradiniai duomenys ir rezultatai

Pirmieji pradiniai duomenys (2 failai) :

turnyras1.txt

```
1

2021-12-05

Lukass;Lukauskass;Komanda4;0;0;ak47

Mikass;Petrauskass;Komanda1;0;0;zeus

Petrass;Petrauskass;Komanda2;0;0;m4a1

Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5

Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2

Mikasss;Petrauskasss;Komanda1;Bot;Ashe;0;0;0
```

turnyras2.txt

```
2
2021-12-06
Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5
Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2
Mikasss;Petrauskasss;Komanda3;Bot;Ashe;0;0;0
Lukass;Lukauskass;Komanda1;0;0;ak47
Mikass;Petrauskass;Komanda1;0;0;zeus
Petrass;Petrauskass;Komanda1;0;0;m4a1
```

Rezultatai:

Komandos.csv

```
All teams
Komanda4
Komanda1
Komanda3
Komanda2
```

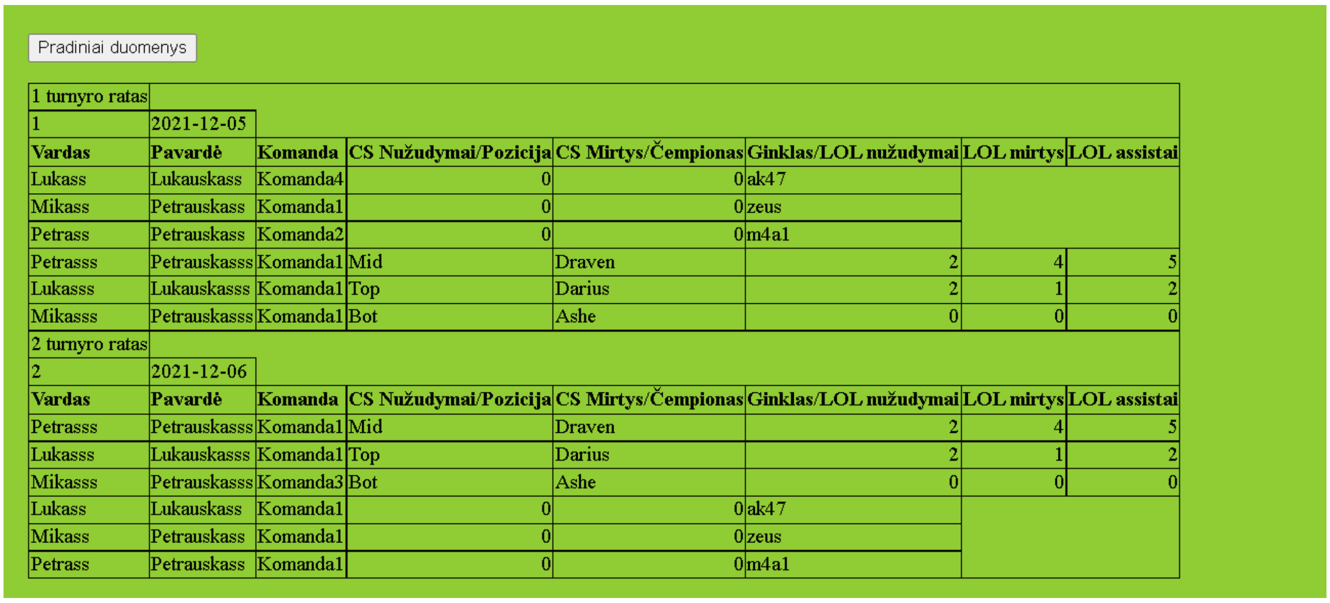
Visi.csv

```
CS players sorted
Lukass;Lukauskass;Komanda4;0;0;ak47
Mikass;Petrauskass;Komanda1;0;0;zeus
Petrass;Petrauskass;Komanda2;0;0;m4a1
Lukass;Lukauskass;Komanda1;0;0;ak47
Mikass;Petrauskass;Komanda1;0;0;zeus
Petrass;Petrauskass;Komanda1;0;0;m4a1
LOL players sorted
Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2
Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2
Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5
```

```
Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5

Mikasss;Petrauskasss;Komanda1;Bot;Ashe;0;0;0

Mikasss;Petrauskasss;Komanda3;Bot;Ashe;0;0;0
```

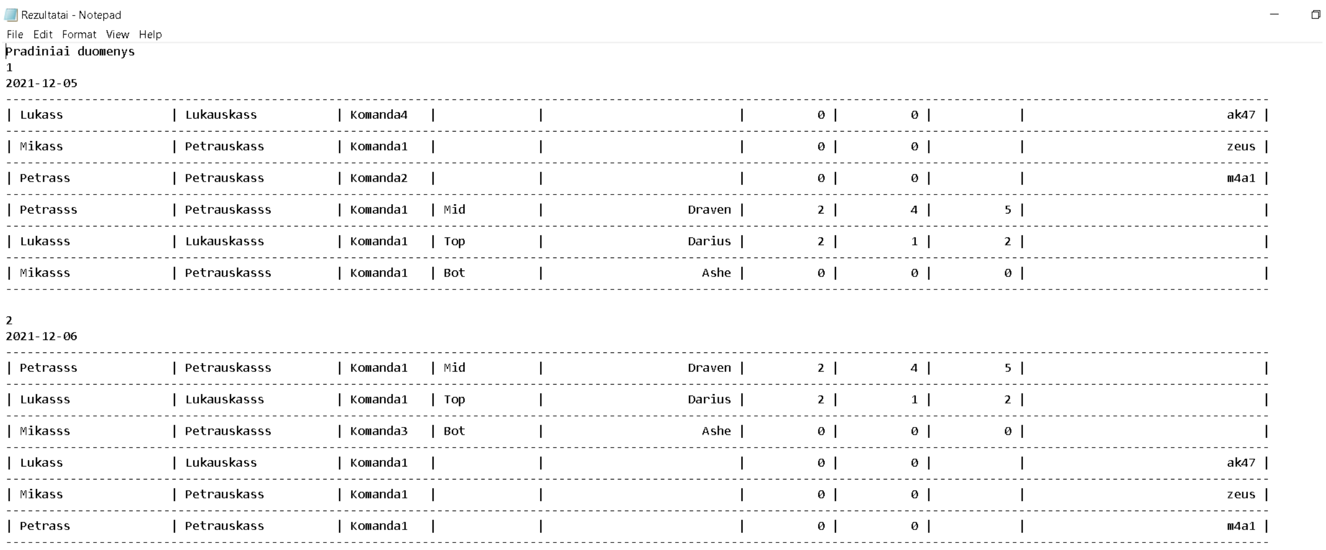
Rinktine.csv

```
CS players

Lukass;Lukauskass;Komanda4;0;0;ak47

Mikass;Petrauskass;Komanda1;0;0;zeus

Petrass;Petrauskass;Komanda2;0;0;m4a1

Lukass;Lukauskass;Komanda1;0;0;ak47

Mikass;Petrauskass;Komanda1;0;0;zeus

Petrass;Petrauskass;Komanda1;0;0;m4a1

LOL players
```



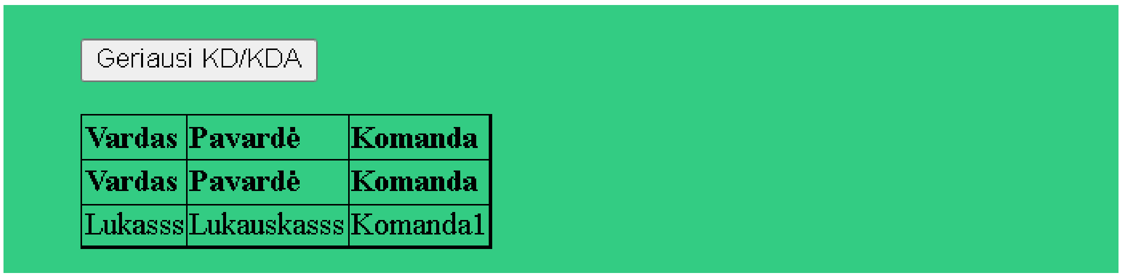
1 turnyro ratas							
1	2021-12-05						
Vardas	**Pavardė**	**Komanda**	**CS Nužudymai/Pozicija**	**CS Mirtys/Čempionas**	**Ginklas/LOL nužudymai**	**LOL mirtys**	**LOL assistai**
Lukass	Lukauskass	Komanda4	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda2	0	0	m4a1		
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Mikasss	Petrauskasss	Komanda1	Bot	Ashe	0	0	0
2 turnyro ratas							
2	2021-12-06						
Vardas	**Pavardė**	**Komanda**	**CS Nužudymai/Pozicija**	**CS Mirtys/Čempionas**	**Ginklas/LOL nužudymai**	**LOL mirtys**	**LOL assistai**
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Mikasss	Petrauskasss	Komanda3	Bot	Ashe	0	0	0
Lukass	Lukauskass	Komanda1	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda1	0	0	m4a1		

Pav. Pradiniai duomenys ekrane



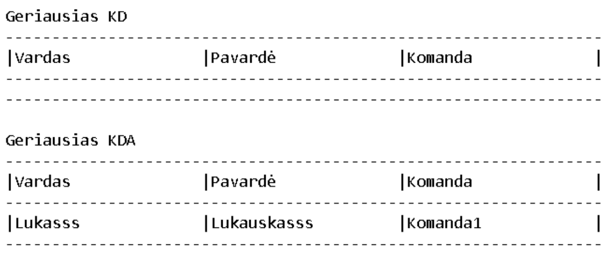
```
Pradiniai duomenys
1
2021-12-05
---------------------------------------------------------------------------------------------------------------------------------------
| Lukass            | Lukauskass        | Komanda4 |          |                     |         0 |         0 |         |                 ak47 |
---------------------------------------------------------------------------------------------------------------------------------------
| Mikass            | Petrauskass       | Komanda1 |          |                     |         0 |         0 |         |                 zeus |
---------------------------------------------------------------------------------------------------------------------------------------
| Petrass           | Petrauskass       | Komanda2 |          |                     |         0 |         0 |         |                 m4a1 |
---------------------------------------------------------------------------------------------------------------------------------------
| Petrasss          | Petrauskasss      | Komanda1 | Mid      |              Draven |         2 |         4 |       5 |                      |
---------------------------------------------------------------------------------------------------------------------------------------
| Lukasss           | Lukauskasss       | Komanda1 | Top      |              Darius |         2 |         1 |       2 |                      |
---------------------------------------------------------------------------------------------------------------------------------------
| Mikasss           | Petrauskasss      | Komanda1 | Bot      |                Ashe |         0 |         0 |       0 |                      |
---------------------------------------------------------------------------------------------------------------------------------------

2
2021-12-06
---------------------------------------------------------------------------------------------------------------------------------------
| Petrasss          | Petrauskasss      | Komanda1 | Mid      |              Draven |         2 |         4 |       5 |                      |
---------------------------------------------------------------------------------------------------------------------------------------
| Lukasss           | Lukauskasss       | Komanda1 | Top      |              Darius |         2 |         1 |       2 |                      |
---------------------------------------------------------------------------------------------------------------------------------------
| Mikasss           | Petrauskasss      | Komanda3 | Bot      |                Ashe |         0 |         0 |       0 |                      |
---------------------------------------------------------------------------------------------------------------------------------------
| Lukass            | Lukauskass        | Komanda1 |          |                     |         0 |         0 |         |                 ak47 |
---------------------------------------------------------------------------------------------------------------------------------------
| Mikass            | Petrauskass       | Komanda1 |          |                     |         0 |         0 |         |                 zeus |
---------------------------------------------------------------------------------------------------------------------------------------
| Petrass           | Petrauskass       | Komanda1 |          |                     |         0 |         0 |         |                 m4a1 |
---------------------------------------------------------------------------------------------------------------------------------------
```

Pav. Rezultatai.txt failas



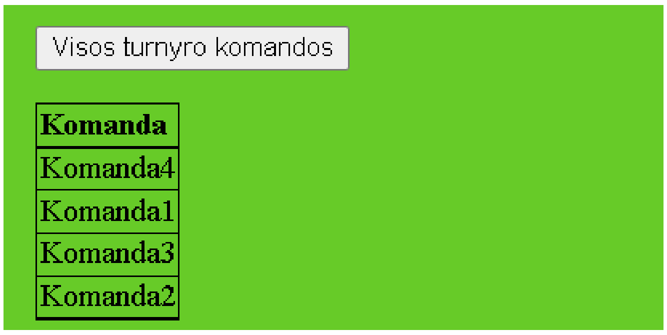
Vardas	Pavardė	Komanda
Vardas	**Pavardė**	**Komanda**
Lukasss	Lukauskasss	Komanda1

Pav. Geriausias KD ir KDA ekrane



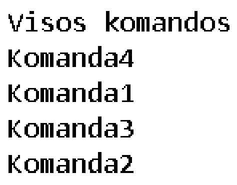
```
Geriausias KD
-------------------------------------------------------------------
|Vardas                |Pavardė               |Komanda               |
-------------------------------------------------------------------
-------------------------------------------------------------------

Geriausias KDA
-------------------------------------------------------------------
|Vardas                |Pavardė               |Komanda               |
-------------------------------------------------------------------
|Lukasss               |Lukauskasss           |Komanda1              |
-------------------------------------------------------------------
```

Pav. Geriausias KD ir KDA faile



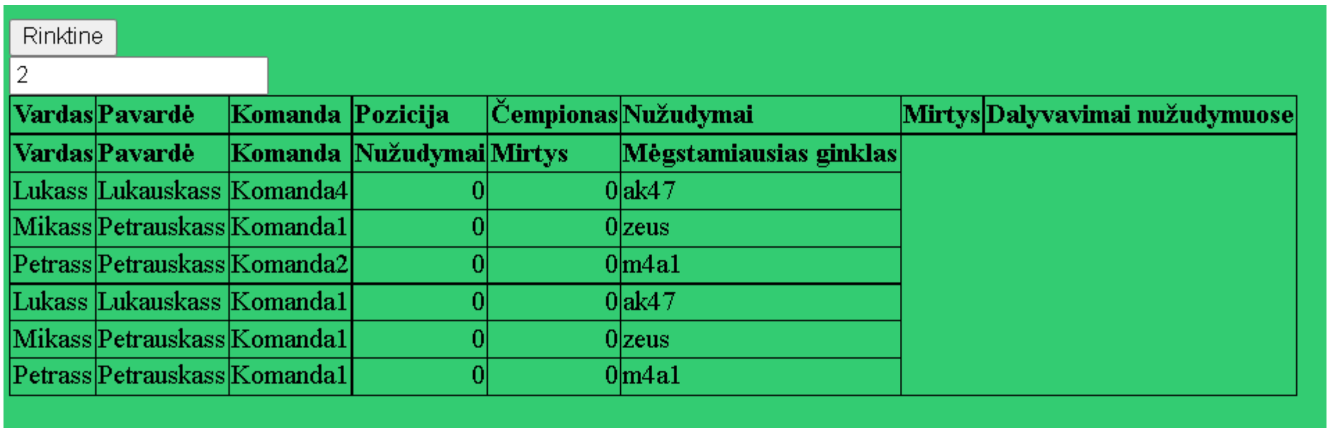
Komanda
Komanda4
Komanda1
Komanda3
Komanda2

Pav. Visos turnyro komandos ekrane



```
Visos komandos
Komanda4
Komanda1
Komanda3
Komanda2
```

Pav. Visos turnyro komandos faile

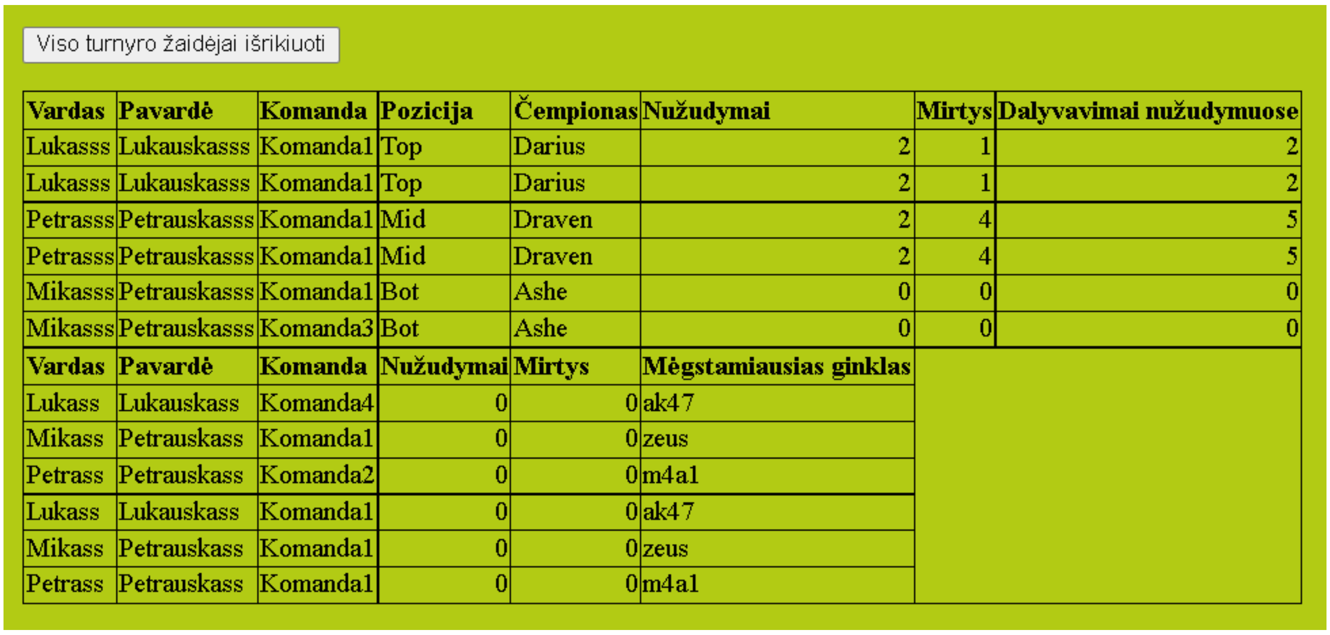


Vardas	Pavardė	Komanda	Pozicija	Čempionas	Nužudymai	Mirtys	Dalyvavimai nužudymuose
Vardas	**Pavardė**	**Komanda**	**Nužudymai**	**Mirtys**	**Mėgstamiausias ginklas**		
Lukass	Lukauskass	Komanda4	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda2	0	0	m4a1		
Lukass	Lukauskass	Komanda1	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda1	0	0	m4a1		

Pav. Rinktine ekrane

```
Pasirinktas rodiklis 2

CS žaidėjai
------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |          Nužudymai|              Mirtys|Ginklas             |
------------------------------------------------------------------------------------------------------------------------
|Lukass              |Lukauskass          |Komanda4            |                  0|                   0|ak47                |
|Mikass              |Petrauskass         |Komanda1            |                  0|                   0|zeus                |
|Petrass             |Petrauskass         |Komanda2            |                  0|                   0|m4a1                |
|Lukass              |Lukauskass          |Komanda1            |                  0|                   0|ak47                |
|Mikass              |Petrauskass         |Komanda1            |                  0|                   0|zeus                |
|Petrass             |Petrauskass         |Komanda1            |                  0|                   0|m4a1                |
------------------------------------------------------------------------------------------------------------------------

LOL žaidėjai
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |Pozicija            |Čempionas           |          Nužudymai|              Mirtys|            Assistai|
----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Pav. Rinktine faile



Vardas	Pavardė	Komanda	Pozicija	Čempionas	Nužudymai	Mirtys	Dalyvavimai nužudymuose
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Mikasss	Petrauskasss	Komanda1	Bot	Ashe	0	0	0
Mikasss	Petrauskasss	Komanda3	Bot	Ashe	0	0	0
Vardas	**Pavardė**	**Komanda**	**Nužudymai**	**Mirtys**	**Mėgstamiausias ginklas**		
Lukass	Lukauskass	Komanda4	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda2	0	0	m4a1		
Lukass	Lukauskass	Komanda1	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda1	0	0	m4a1		

Pav. Visi turnyrų žaidėjai išrikiuoti pagal KD ir KDA ekrane

```
Visi CS žaidėjai išrušiuoti pagal KD
------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |          Nužudymai|              Mirtys|Ginklas             |
------------------------------------------------------------------------------------------------------------------------
|Lukass              |Lukauskass          |Komanda4            |                  0|                   0|ak47                |
|Mikass              |Petrauskass         |Komanda1            |                  0|                   0|zeus                |
|Petrass             |Petrauskass         |Komanda2            |                  0|                   0|m4a1                |
|Lukass              |Lukauskass          |Komanda1            |                  0|                   0|ak47                |
|Mikass              |Petrauskass         |Komanda1            |                  0|                   0|zeus                |
|Petrass             |Petrauskass         |Komanda1            |                  0|                   0|m4a1                |
------------------------------------------------------------------------------------------------------------------------

Visi LOL žaidėjai išrušiuoti pagal KDA
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |Pozicija            |Čempionas           |          Nužudymai|              Mirtys|            Assistai|
----------------------------------------------------------------------------------------------------------------------------------------------------------------
|Lukasss             |Lukauskasss         |Komanda1            |Top                 |Darius              |                  2|                   1|                   2|
|Lukasss             |Lukauskasss         |Komanda1            |Top                 |Darius              |                  2|                   1|                   2|
|Petrasss            |Petrauskasss        |Komanda1            |Mid                 |Draven              |                  2|                   4|                   5|
|Petrasss            |Petrauskasss        |Komanda1            |Mid                 |Draven              |                  2|                   4|                   5|
|Mikasss             |Petrauskasss        |Komanda1            |Bot                 |Ashe                |                  0|                   0|                   0|
|Mikasss             |Petrauskasss        |Komanda3            |Bot                 |Ashe                |                  0|                   0|                   0|
----------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Pav. Visi turnyrų žaidėjai išrikiuoti pagal KD ir KDA faile

Antrieji pradiniai duomenys (3 failai) :

turnyras1.txt

```
1
2021-12-05
Lukass;Lukauskass;Komanda4;0;0;ak47
Mikass;Petrauskass;Komanda1;0;0;zeus
Petrass;Petrauskass;Komanda2;0;0;m4a1
Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5
Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2
Mikasss;Petrauskasss;Komanda1;Bot;Ashe;0;0;0
```

turnyras2.txt

```
2
2021-12-06
Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5
Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2
Mikasss;Petrauskasss;Komanda3;Bot;Ashe;0;0;0
Lukass;Lukauskass;Komanda1;0;0;ak47
Mikass;Petrauskass;Komanda1;0;0;zeus
Petrass;Petrauskass;Komanda1;0;0;m4a1
```

turnyras3.txt

```
3
2021-12-07
Petrassss;Petrauskassss;Komanda1;Mid;Draven;0;0;0
Lukassss;Lukauskassss;Komanda1;Top;Darius;0;0;0
Mikassss;Petrauskassss;Komanda1;Bot;Ashe;0;0;0
Lukasssss;Lukauskasssss;Komanda5;4;2;ak47
Mikasssss;Petrauskasssss;Komanda1;0;0;zeus
Petrasssss;Petrauskasssss;Komanda1;0;0;m4a1
```

Rezultatai:

Komandos.csv

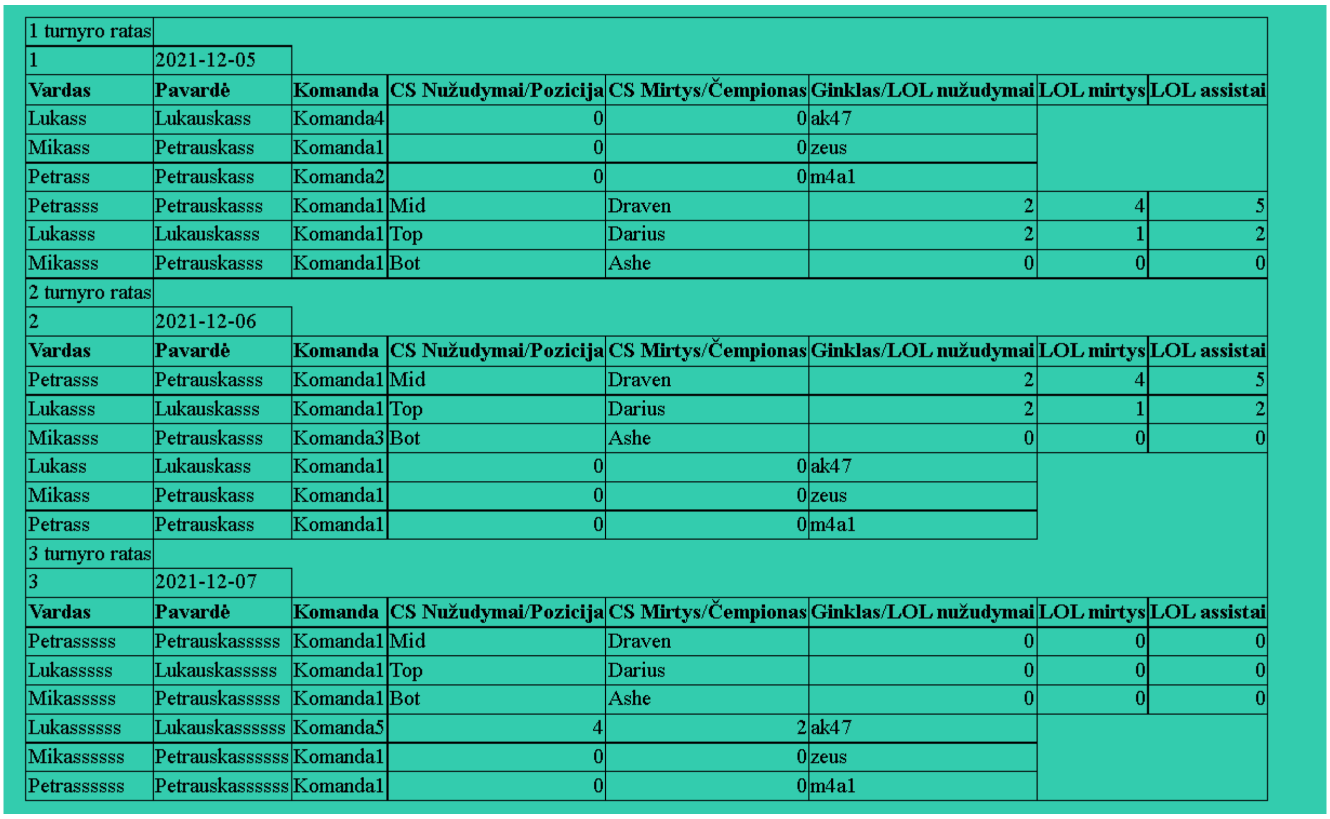
```
All teams

Komanda4

Komanda1

Komanda3

Komanda2

Komanda5
```

Visi.csv

```
CS players sorted

Lukassssss;Lukauskassssss;Komanda5;4;2;ak47

Mikass;Petrauskass;Komanda1;0;0;zeus

Petrass;Petrauskass;Komanda2;0;0;m4a1

Lukass;Lukauskass;Komanda1;0;0;ak47

Mikass;Petrauskass;Komanda1;0;0;zeus

Petrass;Petrauskass;Komanda1;0;0;m4a1

Lukass;Lukauskass;Komanda4;0;0;ak47

Mikassssss;Petrauskassssss;Komanda1;0;0;zeus

Petrassssss;Petrauskassssss;Komanda1;0;0;m4a1

LOL players sorted

Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2

Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2

Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5

Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5

Mikasss;Petrauskasss;Komanda1;Bot;Ashe;0;0;0

Mikasss;Petrauskasss;Komanda3;Bot;Ashe;0;0;0

Petrasssss;Petrauskasssss;Komanda1;Mid;Draven;0;0;0

Lukasssss;Lukauskasssss;Komanda1;Top;Darius;0;0;0

Mikasssss;Petrauskasssss;Komanda1;Bot;Ashe;0;0;0
```

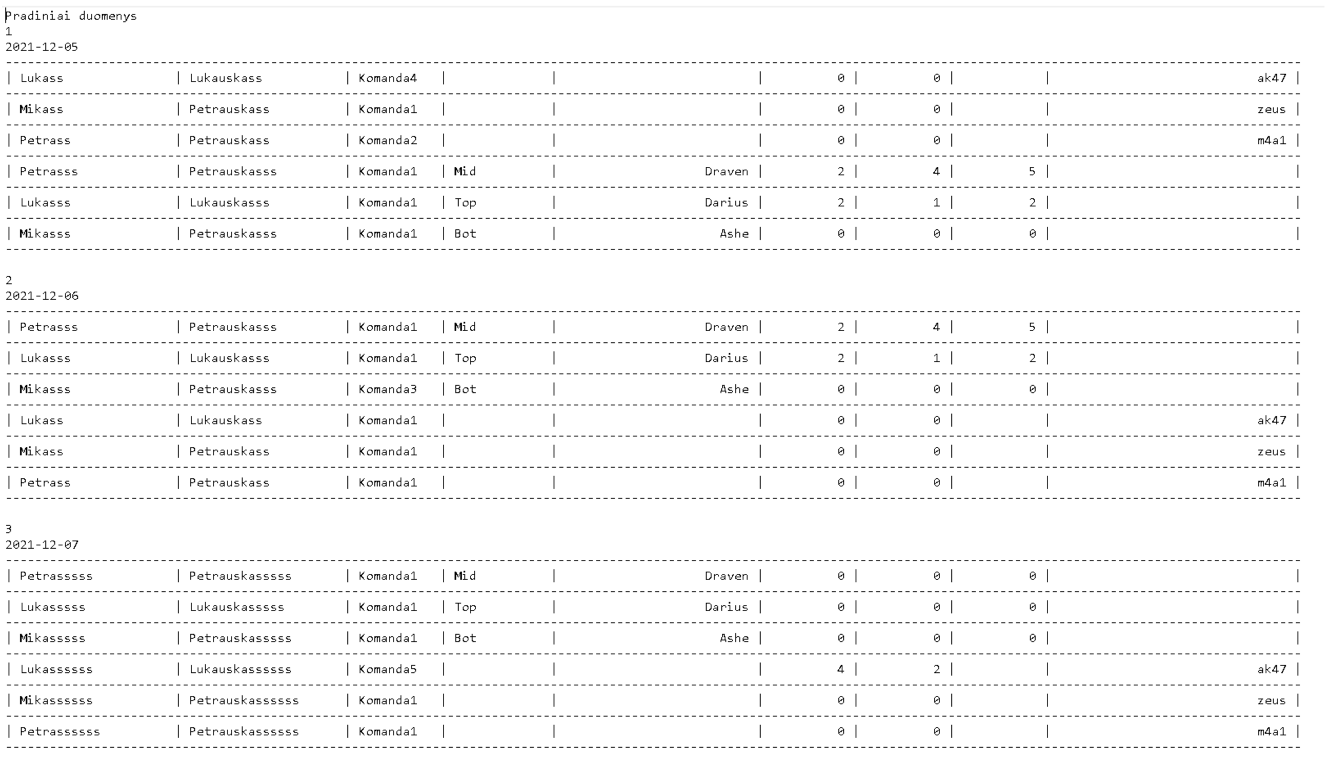
Rinktine.csv

```
CS players

Lukass;Lukauskass;Komanda4;0;0;ak47

Mikass;Petrauskass;Komanda1;0;0;zeus

Petrass;Petrauskass;Komanda2;0;0;m4a1

Lukass;Lukauskass;Komanda1;0;0;ak47

Mikass;Petrauskass;Komanda1;0;0;zeus

Petrass;Petrauskass;Komanda1;0;0;m4a1

Mikassssss;Petrauskassssss;Komanda1;0;0;zeus

Petrassssss;Petrauskassssss;Komanda1;0;0;m4a1

LOL players

Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5

Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2

Petrasss;Petrauskasss;Komanda1;Mid;Draven;2;4;5

Lukasss;Lukauskasss;Komanda1;Top;Darius;2;1;2
```



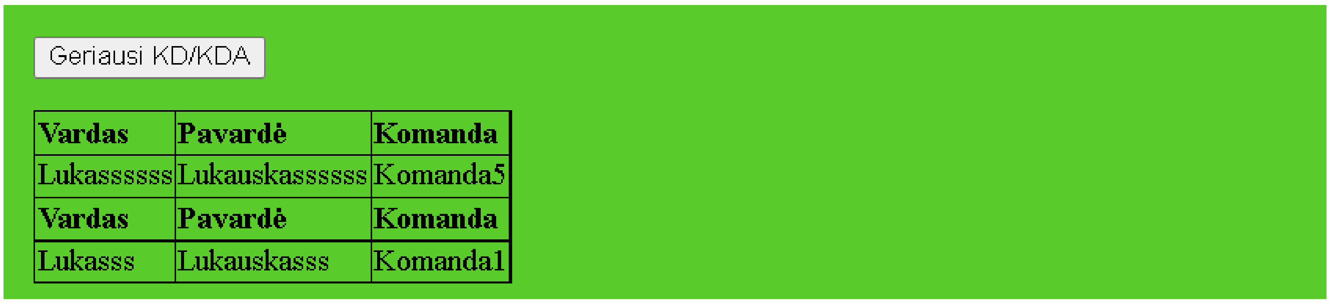
1 turnyro ratas							
1	2021-12-05						
Vardas	**Pavardė**	**Komanda**	**CS Nužudymai/Pozicija**	**CS Mirtys/Čempionas**	**Ginklas/LOL nužudymai**	**LOL mirtys**	**LOL assistai**
Lukass	Lukauskass	Komanda4	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda2	0	0	m4a1		
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Mikasss	Petrauskasss	Komanda1	Bot	Ashe	0	0	0
2 turnyro ratas							
2	2021-12-06						
Vardas	**Pavardė**	**Komanda**	**CS Nužudymai/Pozicija**	**CS Mirtys/Čempionas**	**Ginklas/LOL nužudymai**	**LOL mirtys**	**LOL assistai**
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Mikasss	Petrauskasss	Komanda3	Bot	Ashe	0	0	0
Lukass	Lukauskass	Komanda1	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda1	0	0	m4a1		
3 turnyro ratas							
3	2021-12-07						
Vardas	**Pavardė**	**Komanda**	**CS Nužudymai/Pozicija**	**CS Mirtys/Čempionas**	**Ginklas/LOL nužudymai**	**LOL mirtys**	**LOL assistai**
Petrasssss	Petrauskasssss	Komanda1	Mid	Draven	0	0	0
Lukasssss	Lukauskasssss	Komanda1	Top	Darius	0	0	0
Mikasssss	Petrauskasssss	Komanda1	Bot	Ashe	0	0	0
Lukassssss	Lukauskassssss	Komanda5	4	2	ak47		
Mikassssss	Petrauskassssss	Komanda1	0	0	zeus		
Petrassssss	Petrauskassssss	Komanda1	0	0	m4a1		

Pav. Pradiniai duomenys ekrane



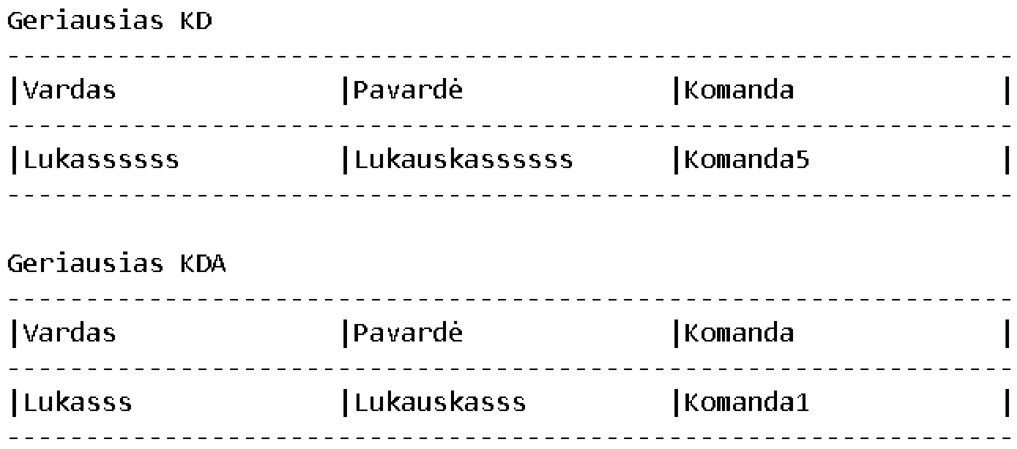
```
Pradiniai duomenys
1
2021-12-05
-------------------------------------------------------------------------------------------------------------------------------------------
| Lukass                 | Lukauskass             | Komanda4    |                 |                           |            0 |            0 |            |                      ak47 |
-------------------------------------------------------------------------------------------------------------------------------------------
| Mikass                 | Petrauskass            | Komanda1    |                 |                           |            0 |            0 |            |                      zeus |
-------------------------------------------------------------------------------------------------------------------------------------------
| Petrass                | Petrauskass            | Komanda2    |                 |                           |            0 |            0 |            |                      m4a1 |
-------------------------------------------------------------------------------------------------------------------------------------------
| Petrasss               | Petrauskasss           | Komanda1    | Mid             |                    Draven |            2 |            4 |          5 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Lukasss                | Lukauskasss            | Komanda1    | Top             |                    Darius |            2 |            1 |          2 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Mikasss                | Petrauskasss           | Komanda1    | Bot             |                      Ashe |            0 |            0 |          0 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------

2
2021-12-06
-------------------------------------------------------------------------------------------------------------------------------------------
| Petrasss               | Petrauskasss           | Komanda1    | Mid             |                    Draven |            2 |            4 |          5 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Lukasss                | Lukauskasss            | Komanda1    | Top             |                    Darius |            2 |            1 |          2 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Mikasss                | Petrauskasss           | Komanda3    | Bot             |                      Ashe |            0 |            0 |          0 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Lukass                 | Lukauskass             | Komanda1    |                 |                           |            0 |            0 |            |                      ak47 |
-------------------------------------------------------------------------------------------------------------------------------------------
| Mikass                 | Petrauskass            | Komanda1    |                 |                           |            0 |            0 |            |                      zeus |
-------------------------------------------------------------------------------------------------------------------------------------------
| Petrass                | Petrauskass            | Komanda1    |                 |                           |            0 |            0 |            |                      m4a1 |
-------------------------------------------------------------------------------------------------------------------------------------------

3
2021-12-07
-------------------------------------------------------------------------------------------------------------------------------------------
| Petrasssss             | Petrauskasssss         | Komanda1    | Mid             |                    Draven |            0 |            0 |          0 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Lukassss               | Lukauskassss           | Komanda1    | Top             |                    Darius |            0 |            0 |          0 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Mikassss               | Petrauskasssss         | Komanda1    | Bot             |                      Ashe |            0 |            0 |          0 |                           |
-------------------------------------------------------------------------------------------------------------------------------------------
| Lukasssss              | Lukauskasssss          | Komanda5    |                 |                           |            4 |            2 |            |                      ak47 |
-------------------------------------------------------------------------------------------------------------------------------------------
| Mikasssss              | Petrauskassssss        | Komanda1    |                 |                           |            0 |            0 |            |                      zeus |
-------------------------------------------------------------------------------------------------------------------------------------------
| Petrassssss            | Petrauskassssss        | Komanda1    |                 |                           |            0 |            0 |            |                      m4a1 |
-------------------------------------------------------------------------------------------------------------------------------------------
```

Pav. Pradiniai duomenys faile



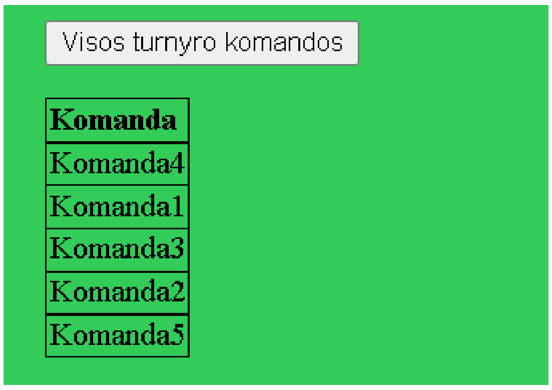
Vardas	Pavardė	Komanda
Lukasssss	Lukauskasssss	Komanda5
Vardas	**Pavardė**	**Komanda**
Lukasss	Lukauskasss	Komanda1

Pav. Geriausi KD ir KDA ekrane



```
Geriausias KD
----------------------------------------------------------------
|Vardas                |Pavardė               |Komanda              |
----------------------------------------------------------------
|Lukasssss             |Lukauskasssss         |Komanda5             |
----------------------------------------------------------------

Geriausias KDA
----------------------------------------------------------------
|Vardas                |Pavardė               |Komanda              |
----------------------------------------------------------------
|Lukasss               |Lukauskasss           |Komanda1             |
----------------------------------------------------------------
```

Pav. Geriausi KD ir KDA faile

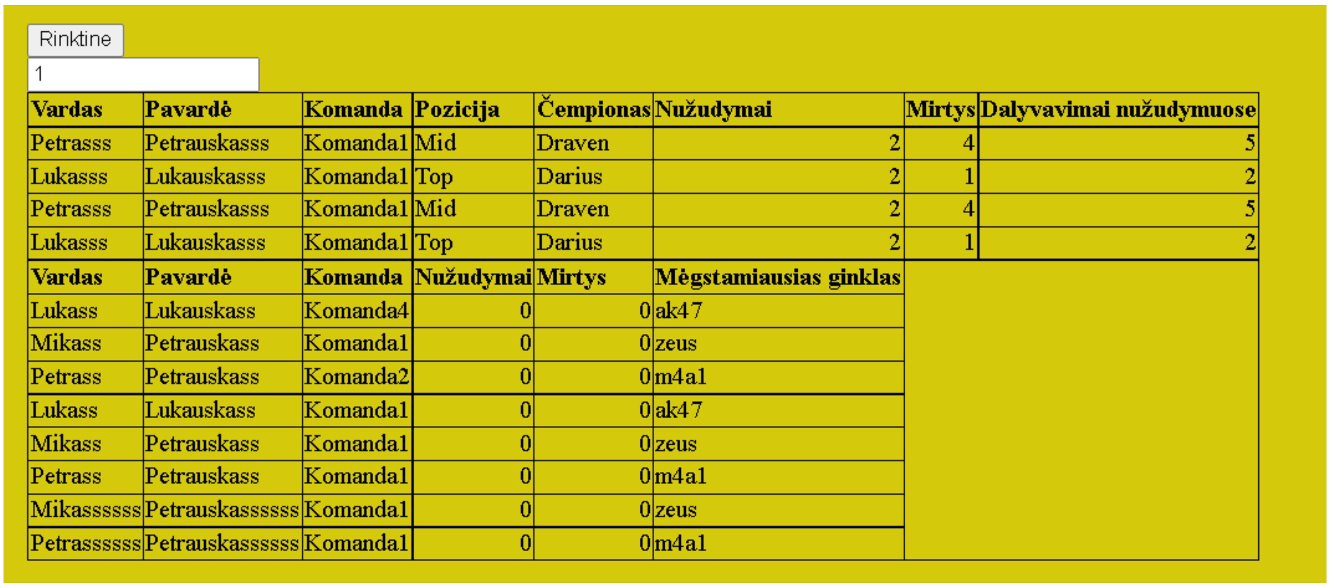


Komanda
Komanda4
Komanda1
Komanda3
Komanda2
Komanda5

Pav. Visos komandos ekrane

```
Visos komandos
Komanda4
Komanda1
Komanda3
Komanda2
Komanda5
```

Pav. Visos turnyro komandos faile

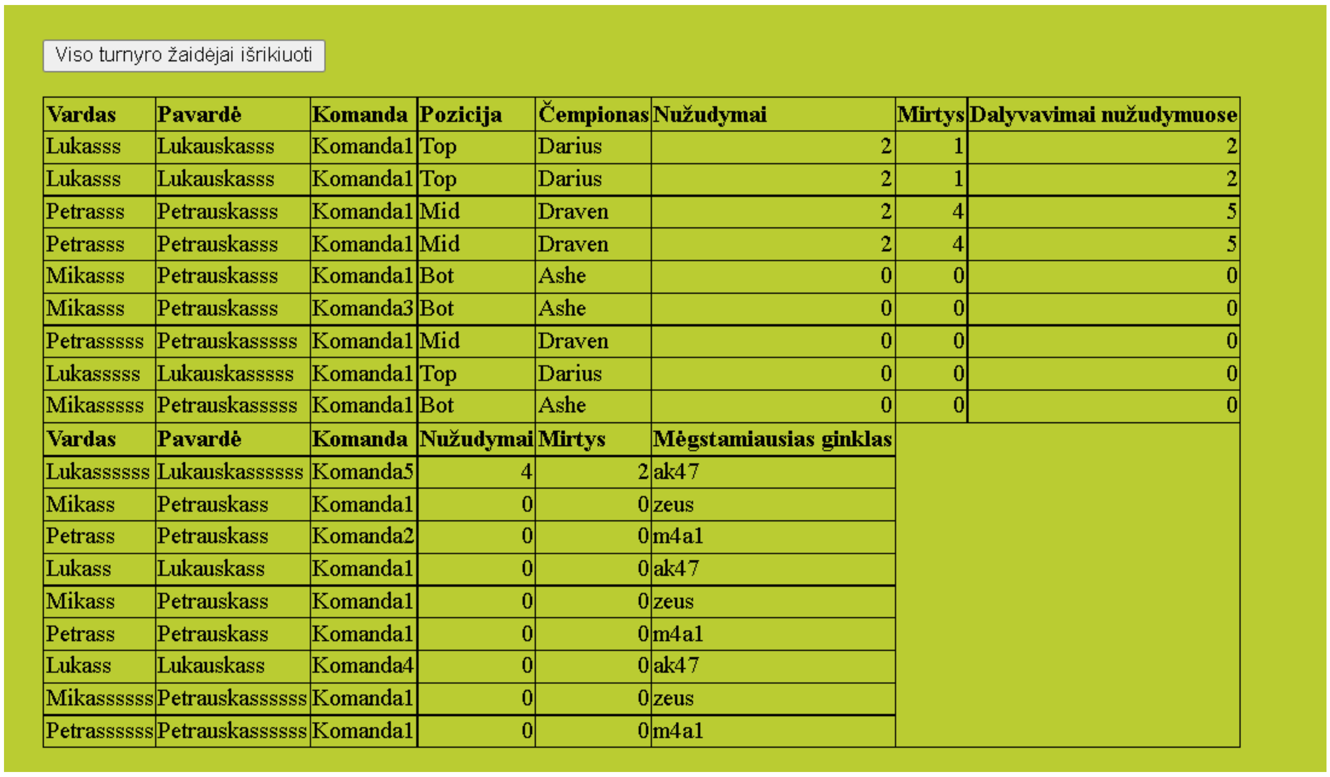


Vardas	Pavardė	Komanda	Pozicija	Čempionas	Nužudymai	Mirtys	Dalyvavimai nužudymuose
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Vardas	**Pavardė**	**Komanda**	**Nužudymai**	**Mirtys**	**Mėgstamiausias ginklas**		
Lukass	Lukauskass	Komanda4	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda2	0	0	m4a1		
Lukass	Lukauskass	Komanda1	0	0	ak47		
Mikass	Petrauskass	Komanda1	0	0	zeus		
Petrass	Petrauskass	Komanda1	0	0	m4a1		
Mikasssssss	Petrauskasssssss	Komanda1	0	0	zeus		
Petrasssssss	Petrauskasssssss	Komanda1	0	0	m4a1		

Pav. Rinktine ekrane

```
Pasirinktas rodiklis 1

CS žaidėjai
-------------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |           Nužudymai|              Mirtys|Ginklas             |
-------------------------------------------------------------------------------------------------------------------------------
|Lukass              |Lukauskass          |Komanda4            |                   0|                   0|ak47                |
|Mikass              |Petrauskass         |Komanda1            |                   0|                   0|zeus                |
|Petrass             |Petrauskass         |Komanda2            |                   0|                   0|m4a1                |
|Lukass              |Lukauskass          |Komanda1            |                   0|                   0|ak47                |
|Mikass              |Petrauskass         |Komanda1            |                   0|                   0|zeus                |
|Petrass             |Petrauskass         |Komanda1            |                   0|                   0|m4a1                |
|Mikassssss          |Petrauskassssss     |Komanda1            |                   0|                   0|zeus                |
|Petrassssss         |Petrauskassssss     |Komanda1            |                   0|                   0|m4a1                |
-------------------------------------------------------------------------------------------------------------------------------

LOL žaidėjai
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |Pozicija            |Čempionas           |           Nužudymai|              Mirtys|            Assistai|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
|Petrasss            |Petrauskasss        |Komanda1            |Mid                 |Draven              |                   2|                   4|                   5|
|Lukasss             |Lukauskasss         |Komanda1            |Top                 |Darius              |                   2|                   1|                   2|
|Petrasss            |Petrauskasss        |Komanda1            |Mid                 |Draven              |                   2|                   4|                   5|
|Lukasss             |Lukauskasss         |Komanda1            |Top                 |Darius              |                   2|                   1|                   2|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

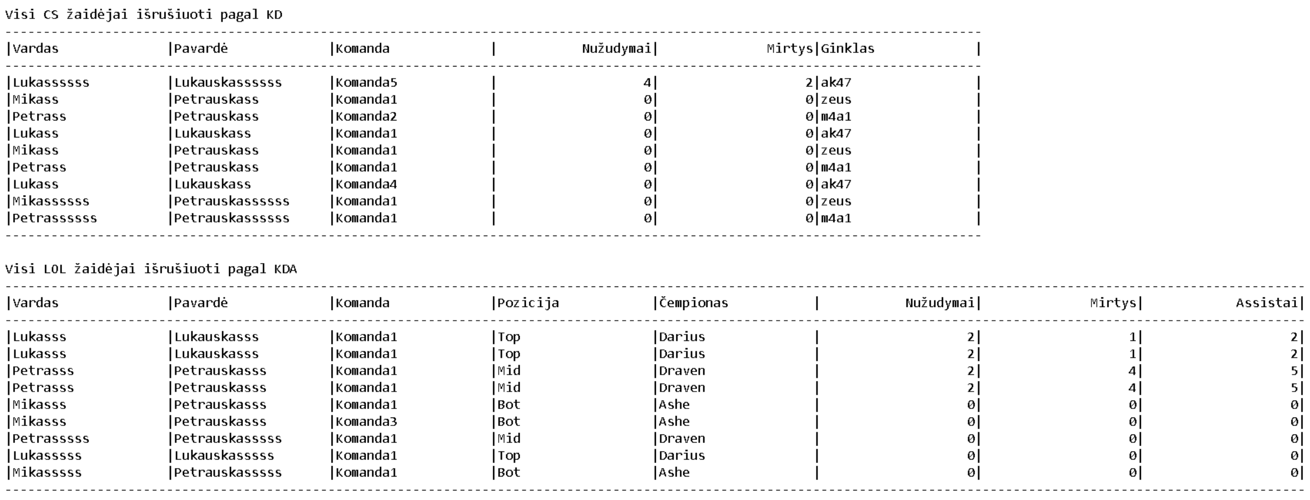
Pav. Rinktine faile



Vardas	Pavardė	Komanda	Pozicija	Čempionas	Nužudymai	Mirtys	Dalyvavimai nužudymuose
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Lukasss	Lukauskasss	Komanda1	Top	Darius	2	1	2
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Petrasss	Petrauskasss	Komanda1	Mid	Draven	2	4	5
Mikasss	Petrauskasss	Komanda1	Bot	Ashe	0	0	0
Mikasss	Petrauskasss	Komanda3	Bot	Ashe	0	0	0
Petrasssss	Petrauskasssss	Komanda1	Mid	Draven	0	0	0
Lukasssss	Lukauskasssss	Komanda1	Top	Darius	0	0	0
Mikasssss	Petrauskasssss	Komanda1	Bot	Ashe	0	0	0

Vardas	Pavardė	Komanda	Nužudymai	Mirtys	Mėgstamiausias ginklas
Lukassssss	Lukauskassssss	Komanda5	4	2	ak47
Mikass	Petrauskass	Komanda1	0	0	zeus
Petrass	Petrauskass	Komanda2	0	0	m4a1
Lukass	Lukauskass	Komanda1	0	0	ak47
Mikass	Petrauskass	Komanda1	0	0	zeus
Petrass	Petrauskass	Komanda1	0	0	m4a1
Lukass	Lukauskass	Komanda4	0	0	ak47
Mikassssss	Petrauskassssss	Komanda1	0	0	zeus
Petrassssss	Petrauskassssss	Komanda1	0	0	m4a1

Pav. Išrikiuoti žaidėjai pagal KD ir KDA ekrane



```
Visi CS žaidėjai išrušiuoti pagal KD
--------------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |                  Nužudymai|               Mirtys|Ginklas              |
--------------------------------------------------------------------------------------------------------------------------------
|Lukassssss          |Lukauskassssss      |Komanda5            |                          4|                    2|ak47                 |
|Mikass              |Petrauskass         |Komanda1            |                          0|                    0|zeus                 |
|Petrass             |Petrauskass         |Komanda2            |                          0|                    0|m4a1                 |
|Lukass              |Lukauskass          |Komanda1            |                          0|                    0|ak47                 |
|Mikass              |Petrauskass         |Komanda1            |                          0|                    0|zeus                 |
|Petrass             |Petrauskass         |Komanda1            |                          0|                    0|m4a1                 |
|Lukass              |Lukauskass          |Komanda4            |                          0|                    0|ak47                 |
|Mikassssss          |Petrauskassssss     |Komanda1            |                          0|                    0|zeus                 |
|Petrassssss         |Petrauskassssss     |Komanda1            |                          0|                    0|m4a1                 |
--------------------------------------------------------------------------------------------------------------------------------

Visi LOL žaidėjai išrušiuoti pagal KDA
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|Vardas              |Pavardė             |Komanda             |Pozicija            |Čempionas           |                  Nužudymai|               Mirtys|             Assistai|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
|Lukasss             |Lukauskasss         |Komanda1            |Top                 |Darius              |                          2|                    1|                    2|
|Lukasss             |Lukauskasss         |Komanda1            |Top                 |Darius              |                          2|                    1|                    2|
|Petrasss            |Petrauskasss        |Komanda1            |Mid                 |Draven              |                          2|                    4|                    5|
|Petrasss            |Petrauskasss        |Komanda1            |Mid                 |Draven              |                          2|                    4|                    5|
|Mikasss             |Petrauskasss        |Komanda1            |Bot                 |Ashe                |                          0|                    0|                    0|
|Mikasss             |Petrauskasss        |Komanda3            |Bot                 |Ashe                |                          0|                    0|                    0|
|Petrasssss          |Petrauskasssss      |Komanda1            |Mid                 |Draven              |                          0|                    0|                    0|
|Lukasssss           |Lukauskasssss       |Komanda1            |Top                 |Darius              |                          0|                    0|                    0|
|Mikasssss           |Petrauskasssss      |Komanda1            |Bot                 |Ashe                |                          0|                    0|                    0|
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

Pav. Išrikiuoti žaidėjai pagal KD ir KDA faile

4.8. Dėstytojo pastabos

Polimorfizmo papildytas panaudojimas.

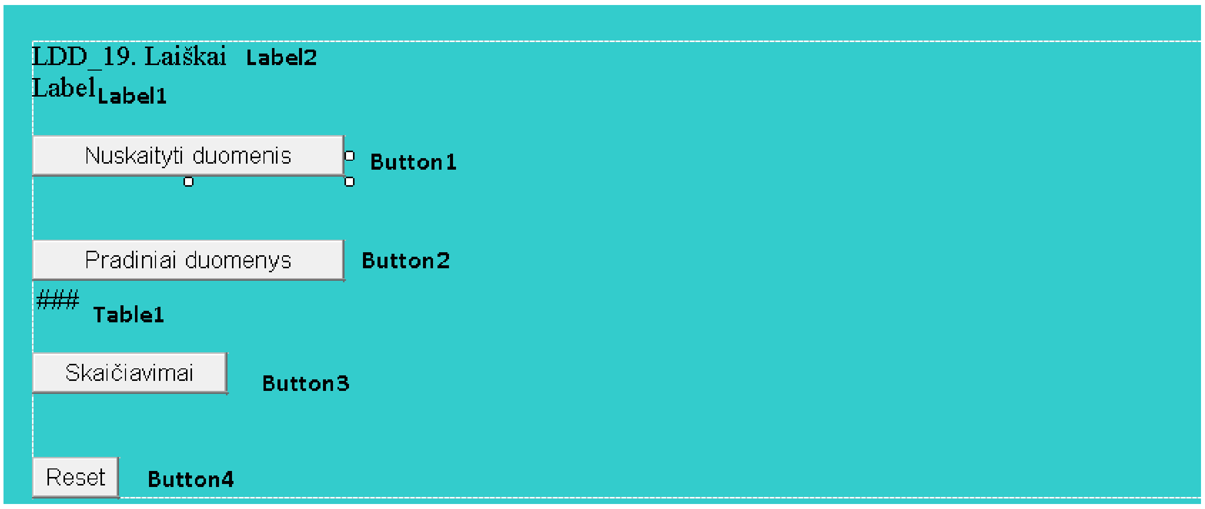
Papildytos išimtys.

5. Deklaratyvusis programavimas (L5)

5.1. Darbo užduotis

LDD_19. Laiškai. Kataloge registruojama elektroninio pašto perdavimo į kitus serverius informacija. Atskiruose failuose saugoma laiškų, kiekvieną dieną persiunčiamų į tam tikrą serverį (turi vardą), informacija. Failo pirmoje eilutėje yra dienos data ir serverio į kurį perduoti laiškai, vardas. Kitose eilutėse yra persiųstų laiškų informacija: laikas, siuntėjo adresas, gavėjo adresas, laiško dydis baitais. Taip pat viename faile yra serverių sąsajos su internetu greitis: serverio vardas ir greitis (baitais/s). Sudaryti sąrašus serverių, į kuriuos nebuvo perduota nei vieno baito, kurios nors valandos bėgyje. Turėdami perdavimo pradžios laiką, jūs galite suskaičiuoti kada perdavimas buvo baigtas. Jeigu perdavimas baigtas kitoje valandoje, tai joje taip pat buvo perduota informacija. Sąraše: serveris, diena ir valandos (surasti visas), kuriomis jis neperdavė nei vieno baito. Rūšiuoti pagal serverį ir dieną.

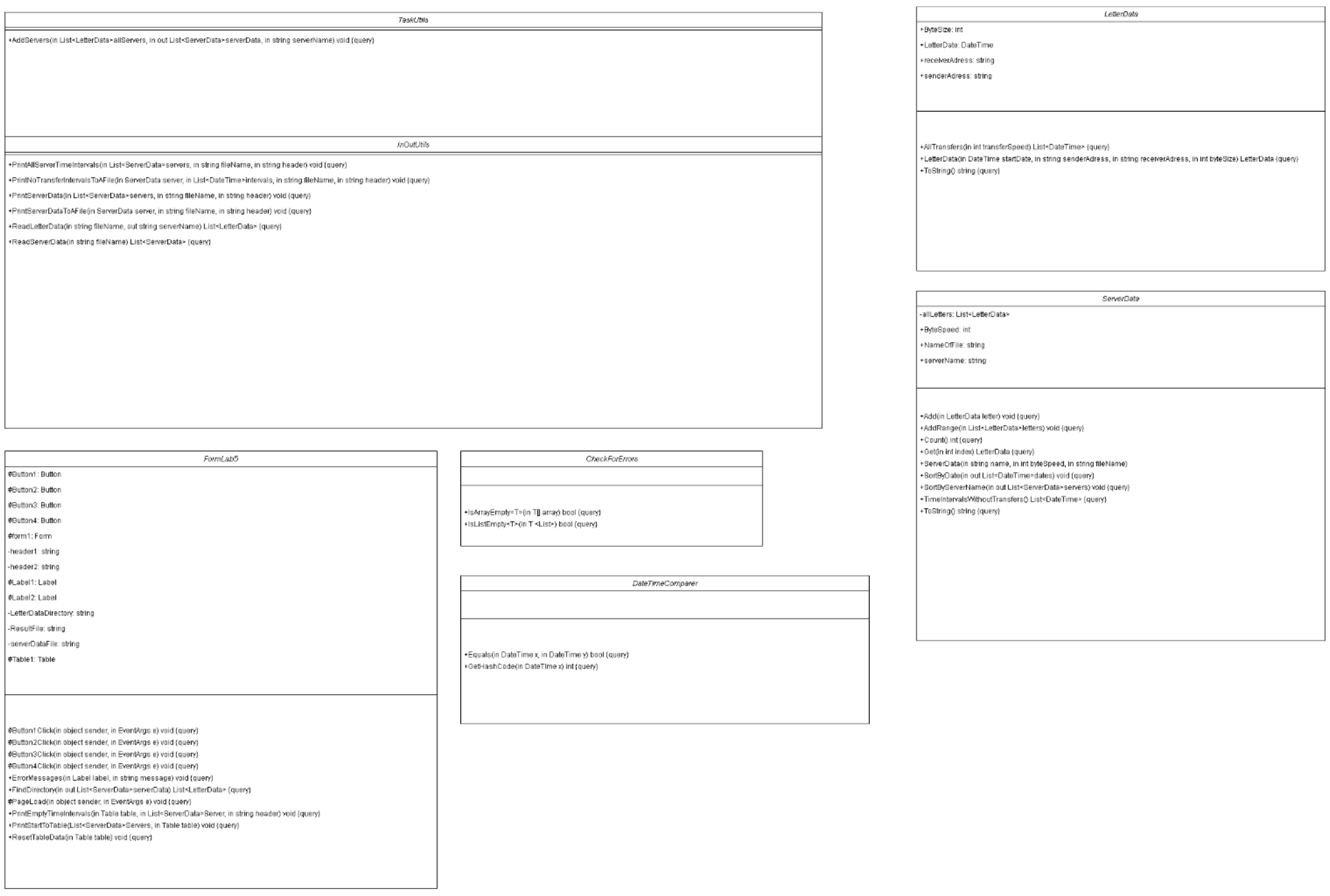
5.2. Grafinės vartotojo sąsajos schema



5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button1	Text	„Nuskaityti duomenis‘‘
Button2	Text	„Pradiniai duomenys“
Button3	Text	„Skaičiavimai“
Button4	Text	„Reset“
Label1	CssClass	„ErrorStyle‘‘
Label2	Text	„LDD_19. Laiškai‘‘
Table1		Solid, Both, black

5.4. Klasių diagrama


TaskUtils
+AddServers(in List<LetterData>allServers, in out List<ServerData>serverData, in string serverName) void {query}
InOutUtils
+PrintAllServerTimeIntervals(in List<ServerData>servers, in string fileName, in string header) void {query}
+PrintNoTransferIntervalsToAFile(in ServerData server, in List<DateTime>intervals, in string fileName, in string header) void {query}
+PrintServerData(in List<ServerData>servers, in string fileName, in string header) void {query}
+PrintServerDataToAFile(in ServerData server, in string fileName, in string header) void {query}
+ReadLetterData(in string fileName, out string serverName) List<LetterData> {query}
+ReadServerData(in string fileName) List<ServerData> {query}
LetterData
+ByteSize: int
+LetterDate: DateTime
+receiverAdress: string
+senderAdress: string
+AllTransfers(in int transferSpeed) List<DateTime> {query}
+LetterData(in DateTime startDate, in string senderAdress, in string receiverAdress, in int byteSize) LetterData {query}
+ToString() string {query}
ServerData
-allLetters: List<LetterData>
+ByteSpeed: int
+NameOfFile: string
+serverName: string
+Add(in LetterData letter) void {query}
+AddRange(in List<LetterData>letters) void {query}
+Count() int {query}
+Get(in int index) LetterData {query}
+ServerData(in string name, in int byteSpeed, in string fileName)
+SortByDate(in out List<DateTime>dates) void {query}
+SortByServerName(in out List<ServerData>servers) void {query}
+TimeIntervalsWithoutTransfers() List<DateTime> {query}
+ToString() string {query}
FormLab5
#Button1 : Button
#Button2: Button
#Button3: Button
#Button4: Button
#form1: Form
-header1: string
-header2: string
#Label1: Label
#Label2: Label
-LetterDataDirectory: string
-ResultFile: string
-serverDataFile: string
#Table1: Table
#Button1 Click(in object sender, in EventArgs e) void {query}
#Button2Click(in object sender, in EventArgs e) void {query}
#Button3Click(in object sender, in EventArgs e) void {query}
#Button4Click(in object sender, in EventArgs e) void {query}
+ErrorMessages(in Label label, in string message) void {query}
+FindDirectory(in out List<ServerData>serverData) List<LetterData> {query}
#PageLoad(in object sender, in EventArgs e) void {query}
+PrintEmptyTimeIntervals(in Table table, in List<ServerData>Server, in string header) void {query}
+PrintStartToTable(List<ServerData>Servers, in Table table) void {query}
+ResetTableData(in Table table) void {query}
CheckForErrors
+IsArrayEmpty<T>(in T[] array) bool {query}
+IsListEmpty<T>(in T <List>) bool {query}
DateTimeComparer
+Equals(in DateTime x, in DateTime y) bool {query}
+GetHashCode(in DateTime x) int {query}

5.5. Programos vartotojo vadovas

Pradėję programą, turime vieną mygtuką - „Nuskaityti duomenis‘‘. Jį paspaudus, mes įrašome mūsų pasirinktus duomenis tam tikroje App_Files1 direktorijoje ir juos išsaugome. Mums atsiranda trys nauji mygtukai - „Pradiniai duomenys’’, „Skaičiavimai‘‘ ir „Reset‘‘. „Pradiniai duomenys‘‘ mygtukas ekrane atspausdina mūsų pasirinktus pradinius duomenis, „Skaičiavimai‘‘ – valandų intervalus, o Reset mygtukas išvalo viską – ir galime vėl nuskaityti tam tikrus duomenis.

5.6. Programos tekstas

LetterData.cs

```
using System;
using System.Collections.Generic;

namespace Lab5.App_Codes1
{
    /// <summary>
    /// Class for LetterData
    /// </summary>
    public class LetterData
    {
        public DateTime LetterDate { get; private set; }
        public string senderAddress { get; private set; }
        public string receiverAddress { get; private set; }
        public int ByteSize { get; private set; }

        /// <summary>
        /// LetterData class constructor
        /// </summary>
        /// <param name="startDate">The starting date</param>
        /// <param name="senderAddress">The sender adress</param>
        /// <param name="receiverAdress">The receiver adress</param>
        /// <param name="byteSize">The size of the file</param>
        public LetterData(DateTime startDate, string senderAddress, string
receiverAdress, int byteSize)
        {
            this.LetterDate = startDate;
            this.senderAddress = senderAddress;
            this.receiverAddress = receiverAdress;
            this.ByteSize = byteSize;
        }
        /// <summary>
        /// Method for finding transfer in 1 hour intervals
        /// </summary>
        /// <param name="transferSpeed">the transfer speed of the server</param>
        /// <returns>DateTime list with objects that transfered data in 1 hour
intervals</returns>
        public List<DateTime> AllTransfers(int transferSpeed)
        {
            List<DateTime> transferDates = new List<DateTime>();
            int secondsPass = ByteSize / transferSpeed;
            DateTime endDate = new DateTime(LetterDate.Year, LetterDate.Month,
LetterDate.Day, LetterDate.Hour, LetterDate.Minute, LetterDate.Second);
            endDate = endDate.AddSeconds(secondsPass);
            DateTime temp = new DateTime(LetterDate.Year, LetterDate.Month,
LetterDate.Day, LetterDate.Hour, LetterDate.Minute, LetterDate.Second);
            while(temp < endDate)
            {
                transferDates.Add(new DateTime(temp.Year, temp.Month, temp.Day,
temp.Hour, temp.Minute, temp.Second));
                temp = temp.AddHours(1);
            }
            transferDates.Add(endDate);
            return transferDates;
        }
        public override string ToString()
        {
           return string.Format("|{0,-25}|{1,-15}|{2,-15}|{3,-20}|",
this.LetterDate.ToString("yyyy-MM-dd-HH-mm-ss"), this.senderAddress,
this.receiverAddress, this.ByteSize);
        }

    }
}
```

ServerData.cs

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

namespace Lab5.App_Codes1
{
    /// <summary>
    /// A class for server data information
    /// </summary>
    public class ServerData
    {
        public string serverName { get; private set; }
        public int ByteSpeed { get;private set; }
        public string NameOfFile { get; private set; }
        private List<LetterData> allLetters = new List<LetterData>();

        /// <summary>
        /// ServerData constructor for server data
        /// </summary>
        /// <param name="name">The name of the server</param>
        /// <param name="byteSpeed">The speed of the server in bytes</param>
        /// <param name="fileName">The name of the file</param>
        public ServerData(string name, int byteSpeed,string fileName)
        {
            this.serverName = name;
            this.ByteSpeed = byteSpeed;
            this.NameOfFile = fileName;
        }
        /// <summary>
        /// Method for getting the count of letters
        /// </summary>
        /// <returns>The count of letters</returns>
        public int Count()
        {
            return this.allLetters.Count;
        }
        /// <summary>
        ///  Method for getting the objects data with an index
        /// </summary>
        /// <param name="index">index of the element</param>
        /// <returns>data of selected object</returns>
        public LetterData Get(int index)
        {
            return this.allLetters[index];
        }
        /// <summary>
        /// Method for adding LetterData element to a List
        /// </summary>
        /// <param name="letter">object to add</param>
        public void Add(LetterData letter)
        {
            this.allLetters.Add(letter);
        }
        /// <summary>
        /// Method for adding a range of mail objects to a the end of a list
        /// </summary>
        /// <param name="letters">mail object list</param>
        public void AddRange(List<LetterData> letters)
        {
            this.allLetters.AddRange(letters);
        }
        /// <summary>
        /// Method for finding objects without any transfer times
        /// </summary>
        /// <returns>DateTime list with date intervals without any
transfers</returns>
```

```
        public List<DateTime> TimeIntervalsWithoutTransfers()
        {
            int[] allHours = new int[] { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 };
            List<DateTime> emptyTime = new List<DateTime>();
            List<DateTime> times = this.allLetters.SelectMany(x =>
x.AllTransfers(this.ByteSpeed)).ToList();
            foreach(var day in times.Distinct(new DateTimeComparer()))
            {
                List<DateTime> dayTimes = times.Where(x => x.Year == day.Year &&
x.Month == day.Month && x.Day == day.Day).ToList();
                foreach(int hour in allHours)
                {
                    if(!dayTimes.Select(x => x.Hour).Contains(hour))
                    {
                        emptyTime.Add(new DateTime(day.Year, day.Month, day.Day,
hour, 0, 0));
                    }
                }
            }
            return emptyTime.Distinct().ToList();
        }
        /// <summary>
        /// Method for sorting a ServerData List by server name
        /// </summary>
        /// <param name="servers">object to sort</param>
        /// <returns>sorted list</returns>
        public static List<ServerData> SortByServerName(List<ServerData> servers)
        {
            return servers.OrderBy(x => x.serverName).ToList();
        }
        /// <summary>
        /// Method for sorting a list by day
        /// </summary>
        /// <param name="dateTimes">DateTime list that needs to be sorted</param>
        public static void SortByDate(List<DateTime> dateTimes)
        {
            dateTimes = dateTimes.OrderBy(x => x.Day).ToList();
        }
        public override string ToString()
        {
            return string.Format("|{0,-20}|{1,15}|{2,15}|", this.serverName,
this.ByteSpeed, this.allLetters.Count);
        }

    }
}
```

FormLab5.cs

```
using Lab5.App_Codes1;
using System;
using System.Collections.Generic;
using System.IO;
using System.Web.UI.WebControls;

namespace Lab5
{
    /// <summary>
    /// FormLab5 class for tasks with certain data
    /// </summary>
    public partial class FormLab5 : System.Web.UI.Page
    {
        /// <summary>
        /// a method for updating a label depending
        /// </summary>
        /// <param name="label">the selected label to update</param>
        /// <param name="message">the wanted message to print</param>
        public static void ErrorMessages(Label label, string message)
        {
            if (message.Length != 0) label.ForeColor = System.Drawing.Color.Red;
            else label.ForeColor = System.Drawing.Color.Blue;
            label.ToolTip = message;
        }
        /// <summary>
        /// a method for removing all table data
        /// </summary>
        /// <param name="table">the selected table to remove the
information</param>
        public static void ResetTableData(Table table)
        {
            while (table.Rows.Count > 0) table.Rows.RemoveAt(0);
        }
```

```
        /// <summary>
        /// a method for finding information from a given folder
        /// </summary>
        /// <param name="serverData">a list of server objects</param>
        /// <returns>a list of LetterData objects</returns>
        public List<LetterData> FindDirectory(ref List<ServerData> serverData)
        {
            List<LetterData> AllLetters = new List<LetterData>();
            string[] Files =
Directory.GetFiles(Server.MapPath(letterDataDirectory), "*letter*.txt");
            int Count = 0;
            foreach (string file in Files)
            {
                string serverName = "";
                if (file != Server.MapPath(ResultFile) && file !=
Server.MapPath(serverDataFile))
                {
                    try
                    {
                        AllLetters = InOutUtils.ReadLetterData(file, out
serverName);
                    }
                    catch
                    {
                        Count++;
                        continue;
                    }
                    TaskUtils.AddServers(AllLetters, ref serverData, serverName);
                }
            }
            if (Count > 0)
            {
                Label1.Text = "Klaidinga duomenų failų informacija";
                Label1.Visible = true;
            }
            if (AllLetters.Count == 0)
            { throw new Exception("Blogas duomenų failų formatas"); }
            return AllLetters;
        }


        /// <summary>
        /// a method for printing starting data to a table
        /// </summary>
        /// <param name="Servers">a list of all Servers data</param>
        /// <param name="table">the selected table</param>
        public static void PrintStartToTable(List<ServerData> Servers, Table
table)
        {
            foreach (var server in Servers)
            {
                TableHeaderRow Header = new TableHeaderRow();
                TableHeaderCell headerCell = new TableHeaderCell();
                headerCell.Text = "<b>" + "Serverio failas: " + server.NameOfFile
+ "</b>";

                headerCell.ColumnSpan = 26;
                Header.Cells.Add(headerCell);
                table.Rows.Add(Header);

                TableRow row1 = new TableRow();
                TableRow row2 = new TableRow();
                TableRow row3 = new TableRow();
                TableRow row4 = new TableRow();
                TableRow row5 = new TableRow();

                TableCell ServerName1 = new TableCell();
                TableCell ServerSpeed1 = new TableCell();

                TableCell ServerName2 = new TableCell();
                TableCell ServerSpeed2 = new TableCell();

                TableCell StartDate1 = new TableCell();
                TableCell Sender1 = new TableCell();
                TableCell Receiver1 = new TableCell();
                TableCell ByteSize1 = new TableCell();

                ServerName1.Text = "<b>Serverio pavadinimas</b>";
                ServerSpeed1.Text = "<b>Serverio greitis</b>";

                ServerName2.Text = server.serverName;
                ServerSpeed2.Text = server.ByteSpeed.ToString();

                ServerName1.ColumnSpan = 26;
                ServerSpeed1.ColumnSpan = 26;

                ServerName2.ColumnSpan = 26;
                ServerSpeed2.ColumnSpan = 26;

                row1.Cells.Add(ServerName1);
                row2.Cells.Add(ServerName2);
                row3.Cells.Add(ServerSpeed1);
                row4.Cells.Add(ServerSpeed2);
                table.Rows.Add(row1);
                table.Rows.Add(row2);
                table.Rows.Add(row3);
                table.Rows.Add(row4);

                StartDate1.Text = "<b>Gavimo data</b>";
                Sender1.Text = "<b>Siuntėjo adresas</b>";
                Receiver1.Text = "<b>Gavėjo adresas</b>";
                ByteSize1.Text = "<b>Failo dydis</b>";
                row5.Cells.Add(StartDate1);
                row5.Cells.Add(Sender1);
                row5.Cells.Add(Receiver1);
                row5.Cells.Add(ByteSize1);
                table.Rows.Add(row5);

                for (int i = 0; i < server.Count(); i++)
                {
                    TableCell StartDate2 = new TableCell();
                    TableCell Sender2 = new TableCell();
                    TableCell Receiver2 = new TableCell();
                    TableCell ByteSize2 = new TableCell();
                    TableRow row6 = new TableRow();
                    StartDate2.Text = server.Get(i).LetterDate.ToString();
                    Sender2.Text = server.Get(i).senderAddress;
                    Receiver2.Text = server.Get(i).receiverAddress;
                    ByteSize2.Text = server.Get(i).ByteSize.ToString();
                    StartDate2.HorizontalAlign = HorizontalAlign.Right;
                    ByteSize2.HorizontalAlign = HorizontalAlign.Right;
                    row6.Cells.Add(StartDate2);
                    row6.Cells.Add(Sender2);
                    row6.Cells.Add(Receiver2);
                    row6.Cells.Add(ByteSize2);
                    table.Rows.Add(row6);
                }
            }
        }
        /// <summary>
        /// a method for printing calculated data of empty time slots to a table
        /// </summary>
        /// <param name="Servers">a list of all Servers data</param>
        /// <param name="table">the selected table</param>
        /// <param name="header">the header of printed information</param>
        public static void PrintEmptyTimeSlot(List<ServerData> Servers, Table
table, string header)
        {
            TableHeaderRow Header = new TableHeaderRow();
            TableHeaderCell headerCell = new TableHeaderCell();
            headerCell.Text = "<b>" + header + "</b>";
            headerCell.ColumnSpan = 26;
            Header.Cells.Add(headerCell);
            table.Rows.Add(Header);

            TableRow row = new TableRow();
            foreach (var server in Servers)
            {
                List<DateTime> emptyTimeSlots = new List<DateTime>();
                emptyTimeSlots = server.TimeIntervalsWithoutTransfers();

                TableHeaderRow Header2 = new TableHeaderRow();
                TableHeaderCell headerCell3 = new TableHeaderCell();
                headerCell3.Text = "<b>" + "Serveris: " + server.serverName +
"</b>";

                headerCell3.ColumnSpan = 26;
                Header2.Cells.Add(headerCell3);
                table.Rows.Add(Header2);

                Lab5.App_Codes1.ServerData.SortByDate(emptyTimeSlots);
                for (int i = 0; i < emptyTimeSlots.Count; i++)
                {
                    if (i == 0)
                    {
                        TableHeaderRow headerRow = new TableHeaderRow();
                        TableHeaderCell headerCell2 = new TableHeaderCell();
                        headerCell2.Text = "Diena ir valanda";
```

```
                            headerRow.Cells.Add(headerCell2);
                            table.Rows.Add(headerRow);
                        }
                        row = new TableRow();
                        TableCell cell = new TableCell();
                        cell.Text = emptyTimeSlots[i].ToString("dd-HH");
                        row.Cells.Add(cell);
                        table.Rows.Add(row);
                    }
                }
            }
        }
    }
```

CheckForErrors.cs

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

namespace Lab5.App_Codes1
{
    /// <summary>
    /// A class for checking for initial errors
    /// </summary>
    public class CheckForErrors
    {
        /// <summary>
        /// a method for checking if an array is empty
        /// </summary>
        /// <typeparam name="T">the wanted type to check</typeparam>
        /// <param name="array">the wanted array to check</param>
        /// <returns>if true returns count 0</returns>
        public static bool IsArrayEmpty<T>(T[] array)
        {
            return array.Count() == 0;
        }
        /// <summary>
        /// a method for checking if a list is empty
        /// </summary>
        /// <typeparam name="T">the wanted type of the list</typeparam>
        /// <param name="list">the wanted list to check</param>
        /// <returns>if true returns count 0</returns>
        public static bool IsListEmpty<T>(List<T> list)
        {
            return list.Count == 0;
        }

    }
}
```

DateTimeComparer.cs

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

namespace Lab5.App_Codes1
{
    /// <summary>
    /// A class for DateTime comparison
    /// </summary>
    public class DateTimeComparer : IEqualityComparer<DateTime>
    {
        public bool Equals(DateTime x, DateTime y)
        {
            return x.Year.Equals(y.Year) && x.Month.Equals(y.Month) &&
x.Day.Equals(y.Day);
        }
        public int GetHashCode(DateTime x)
        {
            if (Object.ReferenceEquals(x, null)) return 0;
            return x.Year.GetHashCode() ^ x.Month.GetHashCode() ^
x.Day.GetHashCode();
        }
    }
}
```

InOutUtils.cs

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

namespace Lab5.App_Codes1
{
    /// <summary>
    /// Class for inputting and outputting information
    /// </summary>
    public static class InOutUtils
    {
        /// <summary>
        /// Method for reading LetterData information from a specified path
        /// </summary>
        /// <param name="path">the path of the given file</param>
        /// <param name="serverName">the name of the server</param>
        /// <returns>a list of letter data objects</returns>
        public static List<LetterData> ReadLetterData(string path, out string
serverName)
        {
            List<LetterData> allMail = new List<LetterData>();
            using (StreamReader read = new StreamReader(path))
            {
                string line;
                string[] Values = read.ReadLine().Split(';');
                int[] date = Values[0].Split('-').Select(x =>
int.Parse(x)).ToArray();
                serverName = Values[1];
                while((line = read.ReadLine()) != null)
                {
                    if (line == string.Empty) continue;
                    Values = line.Split(';');
                    int[] time = Values[0].Split('-').Select(x =>
int.Parse(x)).ToArray();
                    string sender = Values[1];
                    string receiver = Values[2];
                    int byteSize = int.Parse(Values[3]);
                    allMail.Add(new LetterData(new DateTime(date[0], date[1],
date[2], time[0], time[1], time[2]), sender, receiver, byteSize));
                }
            }
            return allMail;
        }
        /// <summary>
        /// Method for reading Server data information from a specified path
        /// </summary>
        /// <param name="path">the server information path</param>
        /// <returns>a list of ServerData objects/returns>
        public static List<ServerData> ReadServerData(string path)
        {
            List<ServerData> Servers = new List<ServerData>();
            using (StreamReader read = new StreamReader(path))
            {
                string line;
                string filePath = Path.GetFileName(path);
                while ((line = read.ReadLine()) != null)
                {
                    string[] Values = line.Split(';');
                    Servers.Add(new ServerData(Values[0], int.Parse(Values[1]),
filePath));
                }
            }
            return Servers;
        }

        /// <summary>
        /// Method for reading through the list elements and printing them out
        /// </summary>
        /// <param name="servers">List of all the servers data</param>
        /// <param name="path">the destination file to print to</param>
        /// <param name="header">the header of the printed data</param>
        public static void PrintServerData(List<ServerData> servers, string path,
string header)
        {
            foreach (var server in servers)
            {
                PrintServerDataToAFile(server, path, header);
            }
        }


        /// <summary>
        /// Method for printing server data to a file
        /// </summary>
        /// <param name="server">server data</param>
        /// <param name="path">file to print to</param>
        /// <param name="header">header of file</param>
        private static void PrintServerDataToAFile(ServerData server, string path,
string header)
        {
            using (StreamWriter write = File.AppendText(path))
            {
                write.WriteLine(header);
                write.WriteLine(new string('-', 80));
                write.WriteLine("|Serveris: {0,-20}| Greitis (Baitais/s):{1,-
26}|", server.serverName, server.ByteSpeed);
                write.WriteLine(new string('-', 80));
                for (int i = 0; i < server.Count(); i++)
                {
                    write.WriteLine(server.Get(i).ToString());
                    write.WriteLine(new string('-', 80));
                }
                write.WriteLine();
            }
        }



        /// <summary>
        /// Method for reading through server data list and printing all files
        /// </summary>
        /// <param name="servers">all server data in a list</param>
        /// <param name="path">the destination file</param>
        /// <param name="header">the header</param>
        public static void PrintEmptyServerMailData(List<ServerData> servers,
string path, string header)
        {
            foreach (var server in servers)
            {
                PrintNoTransferIntervalsToAFile(server,
server.TimeIntervalsWithoutTransfers(), path, header);
            }
        }


        /// <summary>
        /// Method for printing a list of no transfer intervals to a file
        /// </summary>
        /// <param name="server">the serverdata information</param>
        /// <param name="emptyTimeSlots">the list with all time intervals</param>
        /// <param name="path">the file to write to</param>
        /// <param name="header">the header of file/param>
        private static void PrintNoTransferIntervalsToAFile(ServerData server,
List<DateTime> emptyTimeSlots, string path, string header)
        {
            using (StreamWriter write = File.AppendText(path))
            {
                write.WriteLine(header);
                write.WriteLine(new string('-', 26));
                write.WriteLine("|Serveris: {0}|", server.serverName);
                write.WriteLine(new string('-', 26));
                write.WriteLine("|Diena-Valanda|");
                write.WriteLine(new string('-', 26));
                Lab5.App_Codes1.ServerData.SortByDate(emptyTimeSlots);
                foreach (var empty in emptyTimeSlots)
                {
                    write.WriteLine(empty.ToString("|dd-HH|"));
                    write.WriteLine(new string('-', 26));
                }
                write.WriteLine();
            }
        }



    }
}
```

TaskUtils.cs

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

namespace Lab5.App_Codes1
{
    public static class TaskUtils
    {
        /// <summary>
        /// Method for adding server data information to a ServerData list if they
match
        /// </summary>
        /// <param name="AllServers">a list to add new server data</param>
        /// <param name="serverData">List of servers</param>
        /// <param name="serverName">the wanted name of the server</param>
        public static void AddServers(List<LetterData> AllServers, ref
List<ServerData> serverData, string serverName)
        {
            foreach (var server in serverData)
            {
                if (server.serverName == serverName)
                {
                    server.AddRange(AllServers);
                }
            }
        }
    }
}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="Lab5.FormLab5" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="App_Codes1/Design.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label2" runat="server" CssClass="Header" Text="LDD_19.
Laiškai"></asp:Label>
            <br />
            <asp:Label ID="Label1" runat="server" Text="Label"
CssClass="ErrorStyle"></asp:Label>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="Nuskaityti duomenis" CssClass="Button1" />
            <br />
            <br />
            <br />
        </div>
        <asp:Button ID="Button2" runat="server" OnClick="Button2_Click"
Text="Pradiniai duomenys" CssClass="Button1" />
        <asp:Table ID="Table1" runat="server" CssClass="Table" BorderColor="Black"
BorderStyle="Solid" GridLines="Both">
        </asp:Table>
        <br />
        <asp:Button ID="Button3" runat="server" CssClass="Button1"
OnClick="Button3_Click" Text="Skaičiavimai" />
        <br />
        <br />
        <br />
        <asp:Button ID="Button4" runat="server" CssClass="Button1"
OnClick="Button4_Click" Text="Reset" />
    </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 Lab5.App_Codes1;

namespace Lab5
{
    public partial class FormLab5 : System.Web.UI.Page
    {
        const string letterDataDirectory = "~/App_Files1";
        const string serverDataFile = "/App_Files1/ServerInfo.txt";
        const string ResultFile = "~/App_Files1/Rezultatai.txt";
        const string header1 = "Pradiniai duomenys";
        const string header2 = "Laiškų neišsiuntimo intervalas";

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                Session["Servers"] = new List<ServerData>();
                Button2.Visible = false;
                Button3.Visible = false;
                Button4.Visible = false;
                Label1.Visible = false;
            }
            Button2.Visible = false;
            Button3.Visible = false;
            Button4.Visible = false;
            Label1.Visible = false;
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Button4.Visible = true;
            if (File.Exists(Server.MapPath(ResultFile)))
            {
                File.Delete(Server.MapPath(ResultFile));
            }
            try
            {
                List<ServerData> serverData =
InOutUtils.ReadServerData(Server.MapPath(serverDataFile));
                List<LetterData> allLetters = FindDirectory(ref serverData);
                if (CheckForErrors.IsListEmpty(allLetters))
                {
                    Label1.Visible = true;
                    Label1.Text = "Nėra laiškų failuose";
                    throw new Exception(String.Format("Nėra laiškų failuose"));
                }
                Session["Servers"] = serverData;
                Session["Mail"] = allLetters;

                if (CheckForErrors.IsListEmpty<ServerData>((Session["Servers"] as
List<ServerData>)))
                {
                    Label1.Visible = true;
                    Label1.Text = "Nėra serverių informacijos duomenų faile";
                    throw new Exception("Nėra serverių informacijos duomenų
faile");
                }
                Button2.Visible = true;
                Button3.Visible = true;
                Button1.Visible = false;
            }
            catch(Exception ex)
            {
                Label1.Visible = true;
                File.AppendAllText(Server.MapPath(ResultFile), ex.Message +
"\n\n");
                FormLab5.ErrorMessages(Label1, ex.Message);
                return;
            }
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
            Button3.Visible = true;
            Button4.Visible = true;
            try
            {
                FormLab5.ResetTableData(Table1);
                FormLab5.PrintStartToTable((Session["Servers"] as
List<ServerData>), Table1);
                InOutUtils.PrintServerData((Session["Servers"] as
List<ServerData>), Server.MapPath(ResultFile), header1);
                InOutUtils.PrintEmptyServerMailData((Session["Servers"] as
List<ServerData>), Server.MapPath(ResultFile), header2);
                Session["Servers"] =
Lab5.App_Codes1.ServerData.SortByServerName((Session["Servers"] as
List<ServerData>));
            }
            catch(Exception ex)
            {
                Label1.Visible = true;
                File.AppendAllText(Server.MapPath(ResultFile), ex.Message +
"\n\n");
                FormLab5.ErrorMessages(Label1, ex.Message);
                return;
            }
        }
        protected void Button3_Click(object sender, EventArgs e)
        {
            Button2.Visible = true;
            Button4.Visible = true;
            if ((Session["Servers"] as List<ServerData>).Count == 0)
            {
                return;
            }
            FormLab5.ResetTableData(Table1);
            try
            {
                FormLab5.PrintEmptyTimeSlot((Session["Servers"] as
List<ServerData>), Table1, header2);
            }
            catch (Exception ex)
            {
                Label1.Visible = true;
                File.AppendAllText(Server.MapPath(ResultFile), ex.Message +
"\n\n");
                FormLab5.ErrorMessages(Label1, ex.Message);
                return;
            }
        }
        protected void Button4_Click(object sender, EventArgs e)
        {
            Session.Clear();
            Label1.Visible = false;
            Button1.Visible = true;
            Button2.Visible = false;
            Button3.Visible = false;
        }
    }
}
```

Design.css

```
body {
    text-align: left;
    background: #33CCCC;
    border-style: dashed;
    border-color: #94F602;
    animation: color 8s infinite linear;
    padding: 2em;
    overflow: auto;
}

p {
    font-family: "Lucida Console", "Brush Script MT", monospace;
}



@keyframes color {
    0% {
        background: #39CCCC;
    }

    20% {
        background: #01FF70;
    }

    40% {
        background: #DDDDDD;
    }

    60% {
        background: #FF851B;
    }

    80% {
        background: #7FDBFF;
    }

    100% {
        background: #F012BE;
    }
}
```

5.7. Pradiniai duomenys ir rezultatai

Pirmieji pradiniai duomenys:

letter1.txt

```
2022-03-20;Topocentras
21-50-8;siuntejas1;gavejas1;12312
22-11-0;siuntejas2;gavejas2;23123123
0-2-4;siuntejas3;gavejas3;123123123
14-55-7;siuntejas4;gavejas4;12121
15-59-7;siuntejas5;gavejas5;512314
```

ServerInfo.txt

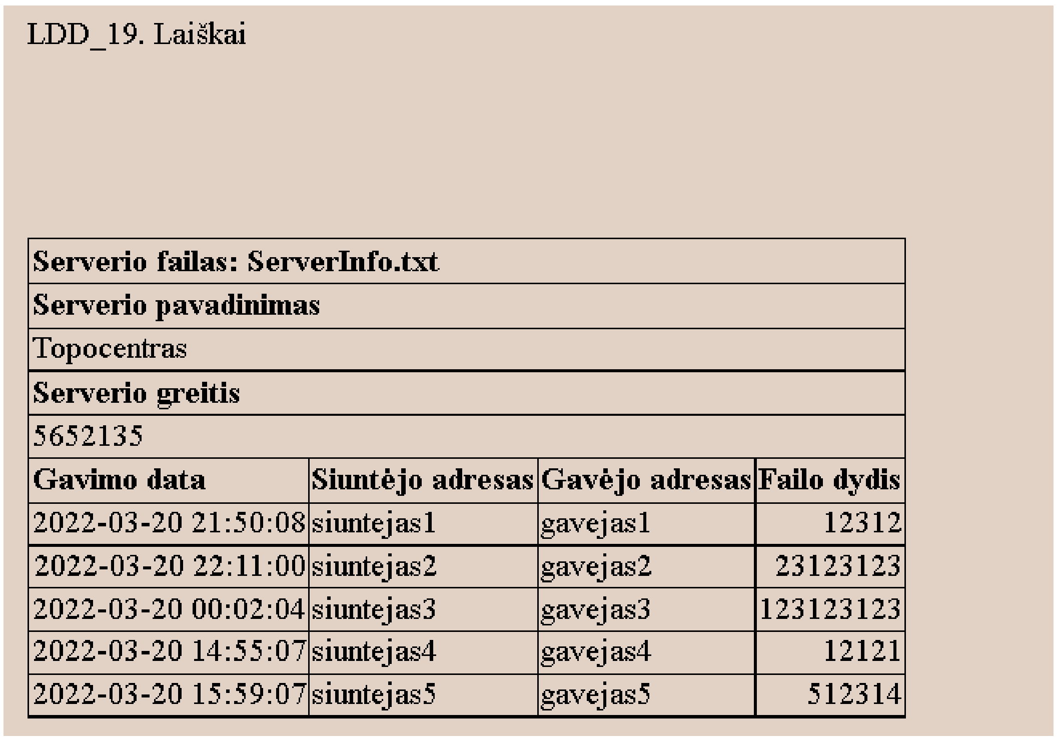
```
Topocentras;5652135
```

Rezultatai.txt

```
Pradiniai duomenys
--------------------------------------------------------------------------------
|Serveris: Topocentras         | Greitis (Baitais/s):5652135                   |
--------------------------------------------------------------------------------
|2022-03-20-21-50-08      |siuntejas1     |gavejas1       |12312               |
--------------------------------------------------------------------------------
|2022-03-20-22-11-00      |siuntejas2     |gavejas2       |23123123            |
--------------------------------------------------------------------------------
|2022-03-20-00-02-04      |siuntejas3     |gavejas3       |123123123           |
--------------------------------------------------------------------------------
|2022-03-20-14-55-07      |siuntejas4     |gavejas4       |12121               |
--------------------------------------------------------------------------------
|2022-03-20-15-59-07      |siuntejas5     |gavejas5       |512314              |
--------------------------------------------------------------------------------
```

```
Laiškų neišsiuntimo intervalas

--------------------------

|Serveris: Topocentras|

--------------------------

|Diena-Valanda|

--------------------------

|20-01|

--------------------------

|20-02|

--------------------------

|20-03|

--------------------------

|20-04|

--------------------------

|20-05|

--------------------------

|20-06|

--------------------------

|20-07|

--------------------------

|20-08|

--------------------------

|20-09|

--------------------------

|20-10|

--------------------------

|20-11|

--------------------------

|20-12|

--------------------------

|20-13|

--------------------------
```

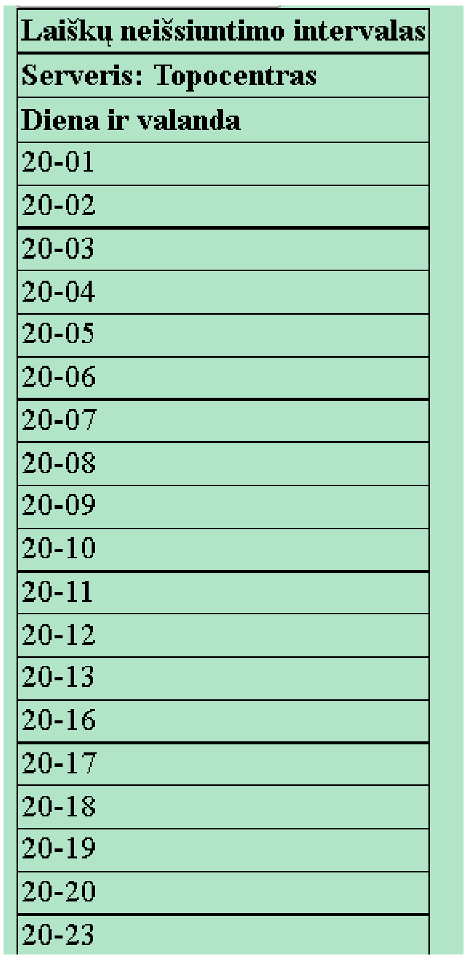
```
|20-16|
--------------------------
|20-17|
--------------------------
|20-18|
--------------------------
|20-19|
--------------------------
|20-20|
--------------------------
|20-23|
--------------------------
```



LDD_19. Laiškai

Serverio failas: ServerInfo.txt			
Serverio pavadinimas			
Topocentras			
Serverio greitis			
5652135			
Gavimo data	**Siuntėjo adresas**	**Gavėjo adresas**	**Failo dydis**
2022-03-20 21:50:08	siuntejas1	gavejas1	12312
2022-03-20 22:11:00	siuntejas2	gavejas2	23123123
2022-03-20 00:02:04	siuntejas3	gavejas3	123123123
2022-03-20 14:55:07	siuntejas4	gavejas4	12121
2022-03-20 15:59:07	siuntejas5	gavejas5	512314

Pav. Pradiniai duomenys



Laiškų neišsiuntimo intervalas
Serveris: Topocentras
Diena ir valanda
20-01
20-02
20-03
20-04
20-05
20-06
20-07
20-08
20-09
20-10
20-11
20-12
20-13
20-16
20-17
20-18
20-19
20-20
20-23

Pav. Valandų intervalas

Antrieji pradiniai duomenys:

Letter1.txt

```
2022-03-20;Topocentras

21-50-8;siuntejas1;gavejas1;12312

22-11-0;siuntejas2;gavejas2;23123123

0-2-4;siuntejas3;gavejas3;123123123

14-55-7;siuntejas4;gavejas4;12121

15-59-7;siuntejas5;gavejas5;512314
```

Letter2.txt

```
2022-04-23;Kilobaitas

22-50-8;siuntejas1;gavejas1;12
```

```
23-11-0;siuntejas2;gavejas2;233

0-2-4;siuntejas3;gavejas3;125

18-55-7;siuntejas4;gavejas4;6

12-59-7;siuntejas5;gavejas5;512
```

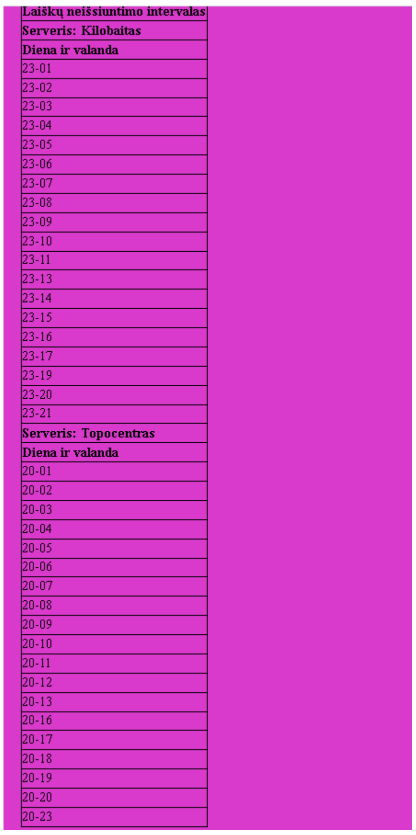
ServerInfo.txt

```
Topocentras;5652135

Kilobaitas;2323
```

Rezultatai.txt

```
Pradiniai duomenys

--------------------------------------------------------------------------------

|Serveris: Topocentras         | Greitis (Baitais/s):5652135                   |

--------------------------------------------------------------------------------

|2022-03-20-21-50-08     |siuntejas1     |gavejas1       |12312                |

--------------------------------------------------------------------------------

|2022-03-20-22-11-00     |siuntejas2     |gavejas2       |23123123             |

--------------------------------------------------------------------------------

|2022-03-20-00-02-04     |siuntejas3     |gavejas3       |123123123            |

--------------------------------------------------------------------------------

|2022-03-20-14-55-07     |siuntejas4     |gavejas4       |12121                |

--------------------------------------------------------------------------------

|2022-03-20-15-59-07     |siuntejas5     |gavejas5       |512314               |

--------------------------------------------------------------------------------


Pradiniai duomenys

--------------------------------------------------------------------------------

|Serveris: Kilobaitas          | Greitis (Baitais/s):2323                      |

--------------------------------------------------------------------------------

|2022-04-23-22-50-08     |siuntejas1     |gavejas1       |12                   |

--------------------------------------------------------------------------------

|2022-04-23-23-11-00     |siuntejas2     |gavejas2       |233                  |
--------------------------------------------------------------------------------
|2022-04-23-00-02-04      |siuntejas3     |gavejas3       |125                   |
--------------------------------------------------------------------------------
|2022-04-23-18-55-07      |siuntejas4     |gavejas4       |6                     |
--------------------------------------------------------------------------------
|2022-04-23-12-59-07      |siuntejas5     |gavejas5       |512                   |
--------------------------------------------------------------------------------


Laiškų neišsiuntimo intervalas
--------------------------
|Serveris: Topocentras|
--------------------------
|Diena-Valanda|
--------------------------
|20-01|
--------------------------
|20-02|
--------------------------
|20-03|
--------------------------
|20-04|
--------------------------
|20-05|
--------------------------
|20-06|
--------------------------
|20-07|
--------------------------
|20-08|
--------------------------
|20-09|
--------------------------
```

```
|20-10|
--------------------------
|20-11|
--------------------------
|20-12|
--------------------------
|20-13|
--------------------------
|20-16|
--------------------------
|20-17|
--------------------------
|20-18|
--------------------------
|20-19|
--------------------------
|20-20|
--------------------------
|20-23|
--------------------------

Laiškų neišsiuntimo intervalas
--------------------------
|Serveris: Kilobaitas|
--------------------------
|Diena-Valanda|
--------------------------
|23-01|
--------------------------
|23-02|
--------------------------
|23-03|
--------------------------
|23-04|
--------------------------
|23-05|
--------------------------
|23-06|
--------------------------
|23-07|
--------------------------
|23-08|
--------------------------
|23-09|
--------------------------
|23-10|
--------------------------
|23-11|
--------------------------
|23-13|
--------------------------
|23-14|
--------------------------
|23-15|
--------------------------
|23-16|
--------------------------
|23-17|
--------------------------
|23-19|
--------------------------
|23-20|
--------------------------
|23-21|
```



Laiškų neišsiuntimo intervalas
Serveris: Kilobaitas
Diena ir valanda
23-01
23-02
23-03
23-04
23-05
23-06
23-07
23-08
23-09
23-10
23-11
23-13
23-14
23-15
23-16
23-17
23-19
23-20
23-21
Serveris: Topocentras
Diena ir valanda
20-01
20-02
20-03
20-04
20-05
20-06
20-07
20-08
20-09
20-10
20-11
20-12
20-13
20-16
20-17
20-18
20-19
20-20
20-23

Pav. Valandų intervalai

5.8. Dėstytojo pastabos