



**Kauno technologijos universitetas**  
Informatikos fakultetas

## **Objektinis programavimas 2 (P175B123)**

Laboratorinių darbų ataskaita

---

**Nedaas Liaudanskis IFF-1/9**

Studentas

**Pareigų Sutrumpinimas Vardas Pavardė**

Dėstytojas / Dėstytoja

---

## TURINYS

<b>1. Rekursija (L1).....</b>	<b>4</b>
1.1. Darbo užduotis .....	4
1.2. Grafinės vartotojo sąsajos schema .....	5
1.3. Sąsajoje panaudotų komponentų keičiamos savybės .....	5
1.4. Klasių diagrama.....	5
1.5. Programos vartotojo vadovas .....	6
1.6. Programos tekstas.....	6
1.7. Pradiniai duomenys ir rezultatai .....	14
1.8. Dėstytojo pastabos.....	15
<b>2. Dinaminis atminties valdymas (L2).....</b>	<b>17</b>
2.1. Darbo užduotis .....	17
2.2. Grafinės vartotojo sąsajos schema .....	17
2.3. Sąsajoje panaudotų komponentų keičiamos savybės .....	18
2.4. Klasių diagrama.....	19
2.5. Programos vartotojo vadovas .....	19
2.6. Programos tekstas.....	20
2.7. Pradiniai duomenys ir rezultatai .....	42
2.8. Dėstytojo pastabos.....	45
<b>3. Bendrinės klasės ir testavimas (L3).....</b>	<b>46</b>
3.1. Darbo užduotis .....	46
3.2. Grafinės vartotojo sąsajos schema .....	46
3.3. Sąsajoje panaudotų komponentų keičiamos savybės .....	47
3.4. Klasių diagrama.....	48
3.5. Programos vartotojo vadovas .....	48
3.6. Programos tekstas.....	49
3.7. Pradiniai duomenys ir rezultatai .....	70

3.8.	Dėstytojo pastabos.....	74
<b>4.</b>	<b>Polimorfizmas ir išimčių valdymas (L4).....</b>	<b>75</b>
4.1.	Darbo užduotis .....	75
4.2.	Grafinės vartotojo sąsajos schema .....	75
4.3.	Sąsajoje panaudotų komponentų keičiamos savybės .....	76
4.4.	Klasių diagrama.....	77
4.5.	Programos vartotojo vadovas .....	77
4.6.	Programos tekstas.....	78
4.7.	Pradiniai duomenys ir rezultatai.....	106
4.8.	Dėstytojo pastabos.....	113
<b>5.</b>	<b>Deklaratyvusis programavimas (L5).....</b>	<b>114</b>
5.1.	Darbo užduotis .....	114
5.2.	Grafinės vartotojo sąsajos schema .....	114
5.3.	Sąsajoje panaudotų komponentų keičiamos savybės .....	114
5.4.	Klasių diagrama.....	115
5.5.	Programos vartotojo vadovas .....	115
5.6.	Programos tekstas.....	116
5.7.	Pradiniai duomenys ir rezultatai.....	132
5.8.	Dėstytojo pastabos.....	139

# 1. Rekursija (L1)

## 1.1. Darbo užduotis

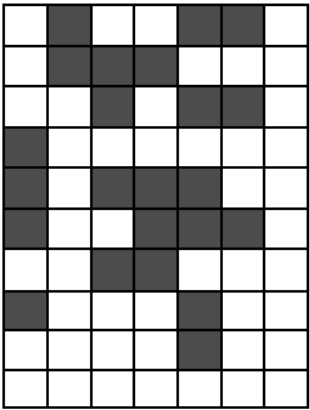
### LD\_14.Salos.

Robotas iš didelio aukščio nufotografavo stačiakampį Žemės paviršiaus plotą. Gautas vaizdas buvo sudalintas kvadratiniais langeliais  $n \times m$  ( $1 \leq n, m \leq 100$ ). Langelis tamsus, jeigu jis atitinka sausumos dalelę ir šviesus, jeigu atitinka vandens plotelį.

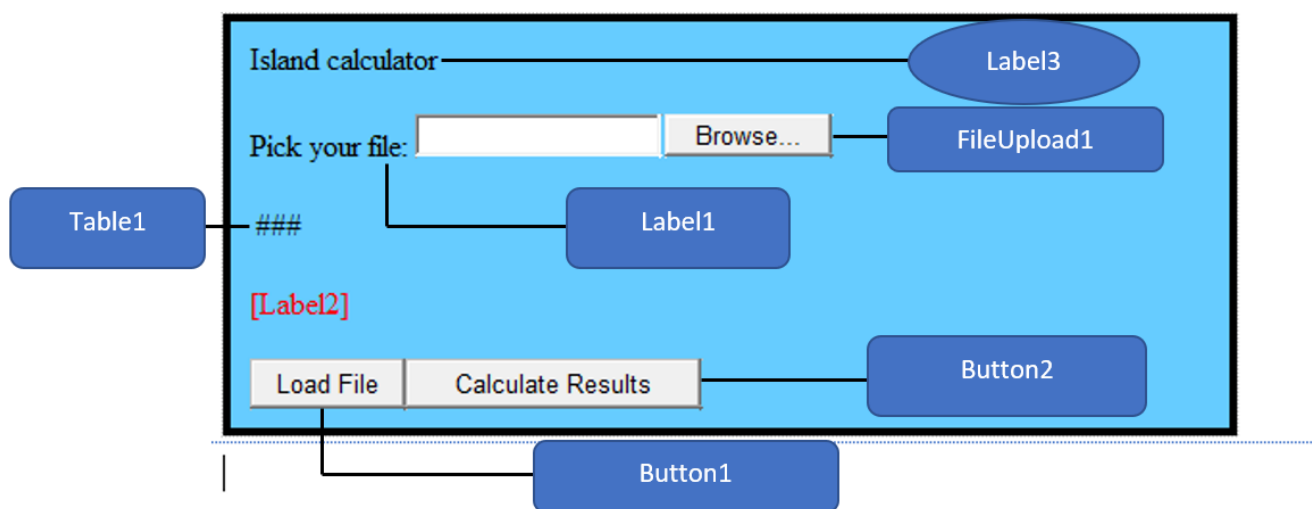
Parašykite programą, kuri suskaičiuotų, kiek yra salų ir kokio dydžio didžiausia sala. Sala – tai gretimų tamsių langelių visuma, iš visų pusių apribota šviesiais. Sala taip pat laikoma tamsių langelių visuma, esanti ant nufotografuoto ploto ribos (ne iš visų pusių supa vandenį). Programa turi analizuoti  $k$  ( $1 \leq k \leq 10$ ) tokių žemės plotų nuotraukų.

**Duomenys** surašyti tekstiname faile U3.txt, kur pirmoje eilutėje yra  $k$ . Toliau – kiekvieno stačiakampio duomenys: pirmoje stačiakampio aprašo eilutėje yra  $n$  ir  $m$  reikšmės, o kitose eilutėse – stačiakampio langelių reikšmės eilutėmis. Tamsų langelį atitinka simbolis '1', o šviesų '0'.

**Rezultatai.** Spausdinkite kiekvienoje eilutėje po du skaičius: kiekvieno paviršiaus ploto salų skaičių ir didžiausios salos langelių skaičių.

Fotografijos pavyzdys	U3.txt	Rez.
Fotografija 10 x 7 prieš ją koduojant skaitmenimis (faile trečia koduotė) 	3	2 2
	2 6	3 8
	110100	4 10
	000000	
	9 4	
	1000	
	1000	
	0110	
	1100	
	1100	
	0000	
	1010	
	1000	
	0000	
	10 7	
	0100110	
	0111000	
	0010110	
	1000000	
	1011100	
	1001110	
	0011000	
	1000100	
	0000100	
	0000000	

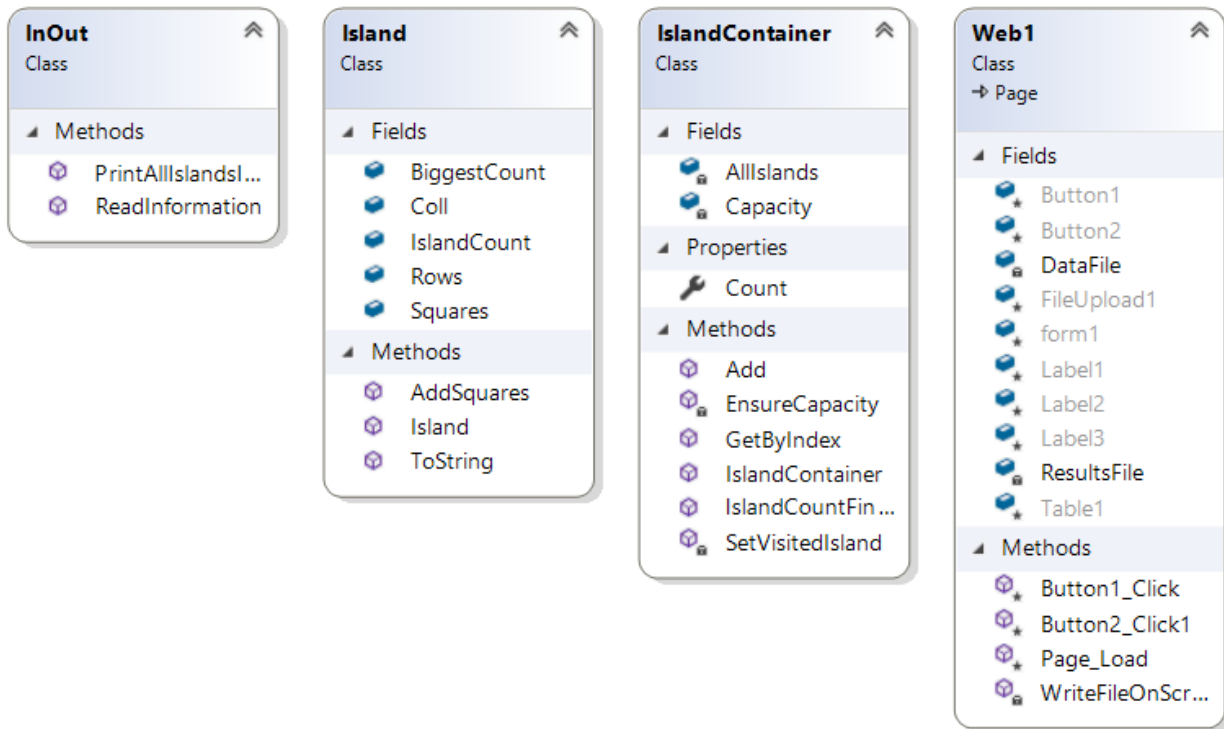
## 1.2. Grafinės vartotojo sąsajos schema



## 1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
FileUpload1		
Label1	Text	Pick your file:
Label2	Text	File you chose was the wrong type or you didn't chose a file, try again !!!
Label2	Text	Calculations couldn't be made. The information inside of the file was the wrong format !!!
Label3	Text	Island calculator
Button1	Text	Load File
Button1	OnClick	Pavaizduoti pradinis duomenis ir įrašyti į duomenų „U3.txt“ failą
Button2	Text	Calculate Results
Button2	OnClick	Apskaičiuoti rezultatus, juos įrašyti ir juos pavaizduoti ekrane
Button2	Visible	False
Table1	GridLines	

## 1.4. Klasių diagrama



## 1.5. Programos vartotojo vadovas

Paleidžiama programa. Paspaudus mygtuką „Choose File“, pasirenkame „txt“ tipo failą, kurio pirmoje eilutėje yra vienas skaičius ( $1 \leq n \leq 10$ ), nurodantis, kiek yra žemės plotų nuotraukų. Tolesnėse eilutėse tiek, kartų, kiek daryta nuotraukų, turi būti užrašytas darytos nuotraukos plotis ir ilgis ( $1 \leq n, m \leq 100$ ) ir žemės nuotrauka, nurodyto pločio ir ilgio, sudaryta iš vienetų – žemės plotelių, nulių – vandens plotelių. Pvz :

```
1
2 6
110100
000000
```

Pasirinkus failą spaudžiame mygtuką „Load File“, kad galėtume pamatyti pasirinktą failą ekrane. Po sėkmingo „Load File“ mygtuko paspaudimo atsiras dar vienas mygtukas „Calculate results“, jį paspaudus ekrane matysite rezultatus, rodančius kiek, kiekvienoje nuotraukoje yra salų ir kokio dydžio yra didžiausia sala. Rezultatai, taip pat bus išsaugoti dokumente „Rezultatai.txt“

## 1.6. Programos tekstas

Island.cs:

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



```

using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebApplication1
{
    /// <summary>
    /// Container containin all photo data
    /// </summary>
    class IslandContainer
    {
        /// <summary>
        /// Array of Island Class that has all the information about one island photo
        /// </summary>
        private Island[] AllIslands;
        /// <summary>
        /// Capacity of the Island array
        /// </summary>
        private int Capacity;
        /// <summary>
        /// Amount of Island class objects inside the array
        /// </summary>
        public int Count { get; private set; }

        /// <summary>
        /// Container Constructor
        /// </summary>
        /// <param name="Size"> How big we want the container to be </param>
        public IslandContainer(int Size = 16)
        {
            this.Capacity = Size;
            this.AllIslands = new Island[Size];
        }

        /// <summary>
        /// Sees if there is enough space to add a new Island class object
        /// to the container, adds space if there isn't
        /// </summary>
        /// <param name="min"> Minimum amount of space needed </param>
        private void EnsureCapacity(int min)
        {
            if (min > this.Capacity)
            {
                Island[] temp = new Island[min];
                for (int i = 0; i < this.Count; i++)
                {
                    temp[i] = this.AllIslands[i];
                }

                this.Capacity = min;
                this.AllIslands = temp;
            }
        }

        /// <summary>
        /// Adds a Island class object to the array
        /// </summary>
        /// <param name="island"> The Island class object we want to add </param>
        public void Add(Island island)
        {
            if(this.Count == Capacity)
            {
                EnsureCapacity(this.Capacity * 2);
            }
            this.AllIslands[this.Count++] = island;
        }
    }
}

```



```

    }

    /// <summary>
    /// Gets a Island class object from the
    /// array that has the same index as the one given
    /// </summary>
    /// <param name="index"> The index of the Island class object
    /// we want to return </param>
    /// <returns> Island class object with the same index as the one given
</returns>
    public Island GetByIndex(int index)
    {
        return this.AllIslands[index];
    }

    /// <summary>
    /// Calculates the island count in one Island class object and find the
    /// size of the biggest island in the Island class object
    /// </summary>
    public void IslandCountFinder()
    {
        for (int i = 0; i < Count; i++)
        {
            int Biggest = 0;
            for (int x = 0; x < AllIslands[i].Rows; x++)
            {
                for (int y = 0; y < AllIslands[i].Coll; y++)
                {
                    if(AllIslands[i].Squares[x, y] == 1)
                    {
                        AllIslands[i].IslandCount++;
                        SetVisitedIsland(x, y, i);
                        if(AllIslands[i].BiggestCount > Biggest)
                        {
                            Biggest = AllIslands[i].BiggestCount;
                        }
                        AllIslands[i].BiggestCount = 0;
                    }
                }
            }
            AllIslands[i].BiggestCount = Biggest;
        }
    }

    /// <summary>
    /// Find all the 1 that connect to each other and marks them as 0 "Found"
    /// </summary>
    /// <param name="x"> Rows number </param>
    /// <param name="y"> Columns number </param>
    /// <param name="i"> What Island class object in the
    /// array we are using </param>
    private void SetVisitedIsland(int x, int y, int i)
    {
        if(x < 0 || x >= AllIslands[i].Rows || y < 0 ||
            y >= AllIslands[i].Coll || AllIslands[i].Squares[x, y] == 0)
        {
            return;
        }
    }

```

```

        AllIslands[i].BiggestCount++;
        AllIslands[i].Squares[x, y] = 0;
        SetVisitedIsland(x + 1, y, i);
        SetVisitedIsland(x - 1, y, i);
        SetVisitedIsland(x + 1, y + 1, i);
        SetVisitedIsland(x + 1, y - 1, i);
        SetVisitedIsland(x , y + 1, i);
        SetVisitedIsland(x , y - 1, i);
        SetVisitedIsland(x - 1, y - 1, i);
        SetVisitedIsland(x - 1, y + 1, i);
    }

}

```

```

        linehold++;
        for (int z = 0; z < Coll; z++)
        {
            island.AddSquares(m, z,
                              int.Parse(Lines[linehold][z].ToString()));
        }
        Container.Add(island);
        linehold++;
    }
    else
    {
        return null;
    }
}

}
else
{
    return null;
}
return Container;
}
catch
{
    return null;
}

}

/// <summary>
/// Prints Rezults
/// </summary>
/// <param name="Container"> Container containing all Island
/// Class objects </param>
/// <param name="RezultFile"> The name of the file where we want to write
/// the results </param>
public static void PrintAllIslandsIntoTXT(IslandContainer Container,
                                           string RezultFile)
{
    for (int i = 0; i < Container.Count; i++)
    {
        Island island = Container.GetByIndex(i);
        File.AppendAllText(RezultFile, island.ToString());
    }
}

}

}

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

```

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Web1.aspx.cs"
Inherits="WebApplication1.Web1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server" style="background-color: #66CCFF; border-style:
solid; border-width: medium; padding: 10px; margin: 10px; width: 500px">
        <div>
            <asp:Label ID="Label3" runat="server" Text="Island calculator"></asp:Label>
            <br />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Pick your file: "></asp:Label>
            <asp:FileUpload ID="FileUpload1" runat="server" />
        </div>
        <p>
            <asp:Table ID="Table1" runat="server" Height="24px">
            </asp:Table>
        </p>
        <p>
            <asp:Label ID="Label2" runat="server" ForeColor="Red"
Visible="False"></asp:Label>
        </p>
        <p>
            <asp:Button ID="Button1" runat="server" Height="26px"
OnClick="Button1_Click" Text="Load File" />
            <asp:Button ID="Button2" runat="server" OnClick="Button2_Click1"
Text="Calculate Results" Visible="False" />
        </p>

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

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

Web1.aspx.cs:

```

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

namespace WebApplication1
{
    public partial class Web1 : System.Web.UI.Page
    {
        private string DataFile, ResultsFile;
    }
}

```

```

protected void Page_Load(object sender, EventArgs e)
{
    DataFile = Server.MapPath("App_Data/U3.txt");
    ResultsFile = Server.MapPath("App_Data/Resultatai.txt");
    Label3.Font.Size = 30;
}

protected void Button1_Click(object sender, EventArgs e)
{
    if(FileUpload1.HasFile && Path.GetExtension(FileUpload1.FileName) ==
".txt")
    {
        if (File.Exists(ResultsFile) && FileUpload1.FileName !=
"Resultatai.txt")
        {
            File.Delete(ResultsFile);
        }
        FileUpload1.SaveAs(DataFile);
        Label2.Visible = false;
        WriteFileOnScreen(DataFile);
        Button2.Visible = true;
    }
    else
    {
        Label2.Text = "File you chose was the wrong type or you didn't chose a
file, try again !!!";
        Label2.Visible = true;
    }
}

protected void Button2_Click1(object sender, EventArgs e)
{
    if(InOut.ReadInformation(DataFile) == null)
    {
        Label2.Text = "Calculations couldn't be made. The information inside of
the file was the wrong format !!!";
        Label2.Visible = true;
    }
    else
    {
        IslandContainer Islands = InOut.ReadInformation(DataFile);
        Islands.IslandCountFinder();
        InOut.PrintAllIslandsIntoTXT(Islands, ResultsFile);
        WriteFileOnScreen(ResultsFile);
        Button2.Visible = false;
    }
}

/// <summary>
/// Writes information from a file into a Table
/// </summary>
/// <param name="FileName"> File from where we will write the information on
the Table1 </param>
private void WriteFileOnScreen(string FileName)
{
    string[] Lines = File.ReadAllLines(FileName);

    foreach(string line in Lines)
    {

```

```

        var langas = new TableCell();
        var eilute = new TableRow();
        langas.Text = line;
        eilute.Cells.Add(langas);
        Table1.Rows.Add(eilute);
    }

}

}

```

## 1.7. Pradiniai duomenys ir rezultatai

### Testas Nr. 1:

**Tikslas:** Standartinis atvejis, kai apskaičiuojami keli atsakymai.

U3.txt

```

2
1 9
101100110
5 5
11111
10000
00110
00100
10000

```

Rezultatai.txt

```

3 2
3 6

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

### Testas Nr. 2:

**Tikslas:** Parodyti kas bus kai nuotraukų yra daugiau negu duomenų.

U3.txt

```

99
1 9
101100110
5 5
11111
10000
00110
00100
10000

```

Rezultatai.txt (Failas, nebuvo sukurtas)

# Island calculator

Pick your file:  No file chosen

Calculations couldn't be made. The information inside of the file was the wrong format !!!

////////////////////  
-----  
////////////////////

### Testas Nr. 3:

**Tikslas:** Parodyti kas bus kai nuotraukos dydis viršija normas.

U3.txt

```

2
1100000 12
101100110
5 5
11111
10000
00110
00100
10000

```

Rezultatai.txt (Failas, nebuvo sukurtas)

# Island calculator

Pick your file:  No file chosen

Calculations couldn't be made. The information inside of the file was the wrong format !!!

## 1.8. Dėstytojo pastabos





## 2. Dinaminis atminties valdymas (L2)

### 2.1. Darbo užduotis

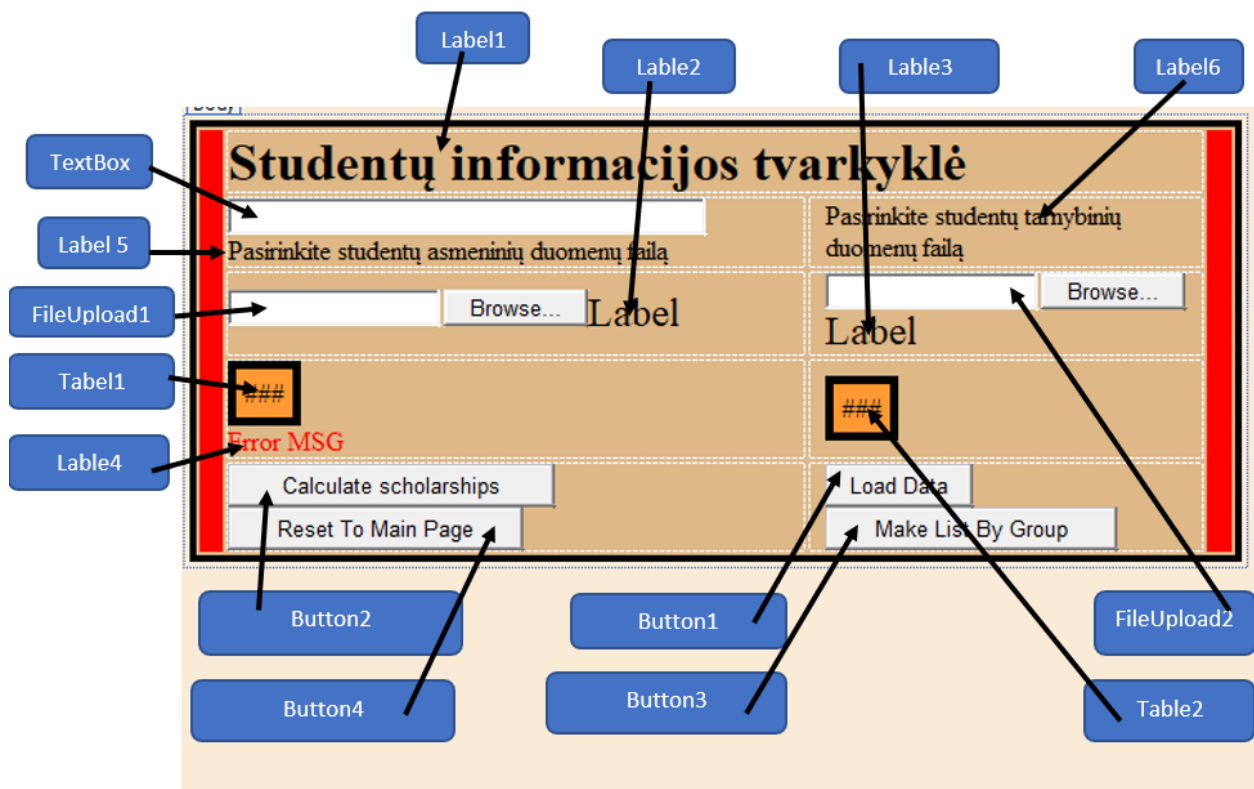
**LD\_14. Stipendijos.** Studentų stipendijoms yra skiriamas nurodyto dydžio fondas. Studentui skiriama stipendija, jei jo pažymių vidurkis viršija nurodytą dydį ir jis neturi skolų (visi pažymiai >4). Studentui skiriama 10% didesnė stipendija, jei jo visi pažymiai didesni už 8. Toks studentas vadinamas pirmūnu. Paskirstykite studentams stipendijas pagal duotą fondą. Fondą reikia maksimaliai išnaudoti, bet negalima viršyti fondo dydžio.

Duomenys:

- Tekstiniame faile U14a.txt duoti studentų asmeniniai duomenys: studento eilės numeris, pavardė, vardas, telefono numeris.
- Tekstiniame faile U14b.txt pateikta studentų tarnybinė informacija. Pirmojoje failo eilutėje nurodytas stipendijų fondo dydis ir pažymių vidurkis stipendijai gauti. Tolimesnėse eilutėse tokia informacija: studento eilės numeris, kuris sutampa su eilės numeriu U14a.txt faile, grupė, pažymių kiekis, pažymiai.

Spausdinamas sąrašas (studento pavardė, vardas, pažymiai, stipendijos dydis) turi būti surikiuotas pagal stipendijų dydį, pavardes ir vardus. Iš sąrašo pašalinkite studentus, kurie negauna stipendijos. Sudarykite ir atspausdinkite nurodytos grupės (įvedama klaviatūra) pirmūnų sąrašą (studento pavardė, vardas, pažymiai).

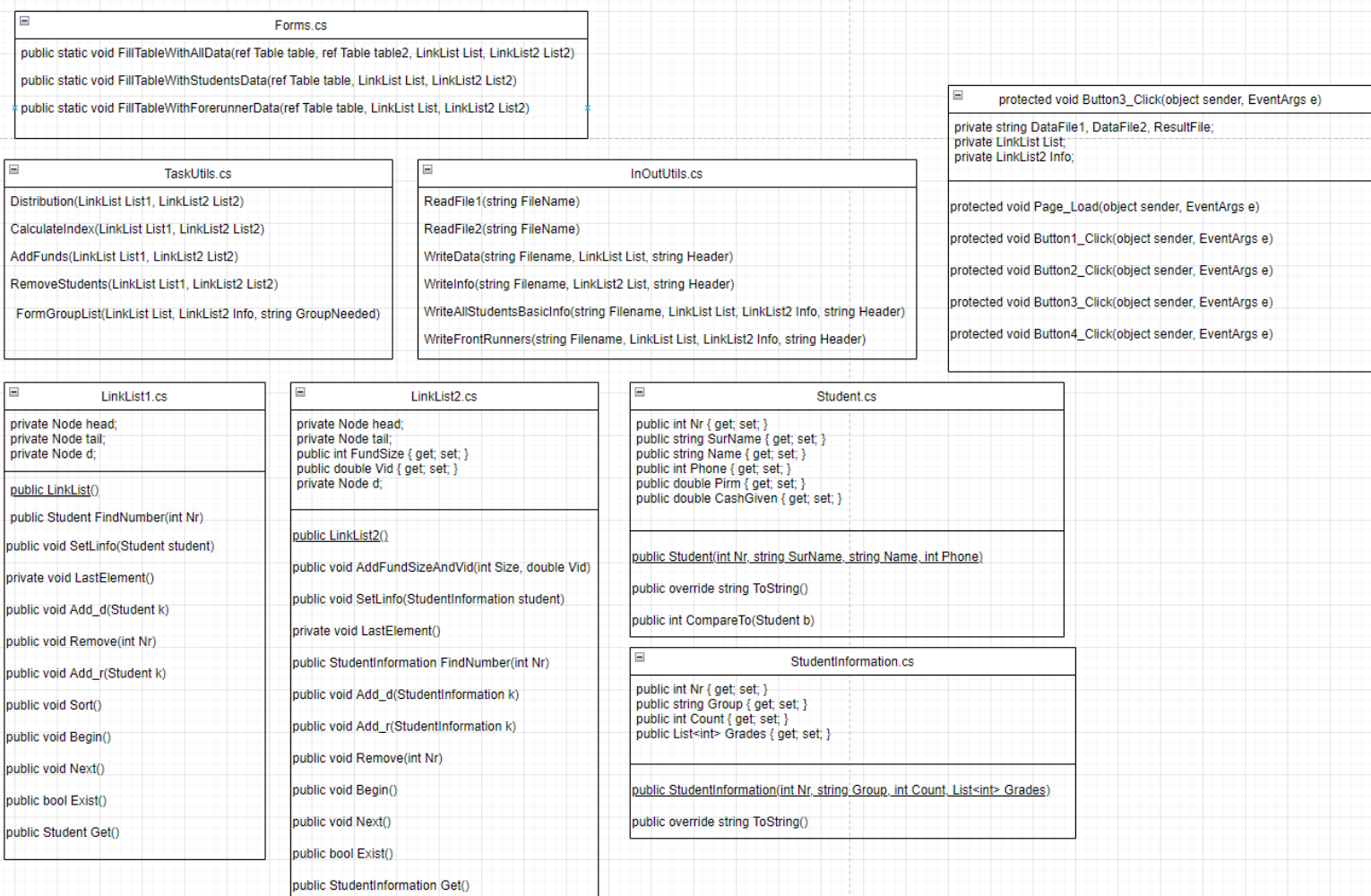
### 2.2. Grafinės vartotojo sąsajos schema



### 2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
FileUploader1	Get file	
FileUploader2	Get file	
Lable1	Text	"Studentų informacijos tvarkyklė"
Lable2	Text	"Studentų asmeniniai duomenys" // "Studentų sąrašas" // "Grupės " + TextBox1.Text + " pirmūnai"
Lable3	Text	"Studentų tarnybinė informacija" // "Sąrašas be studentų, kurie neturi stipendijos"
Lable4	Text	"* Failai/Failas, kurių pasirinkote turėjo neteisingą informacijos išdėstymą" // "* Failai/Failas, kurių pasirinkote buvo neteisingo formato !!! Bandykite dar kartą." // "* Parašėte neteisingą grupės formatą !!! (Grupės sudarytos iš vienos raidės)"
Lable5	Text	"Pasirinkite studentų asmeninių duomenų failą"
Lable6	Text	"Pasirinkite studentų tarnybinių duomenų failą"
Button1	Text	"Load Data"
Button1	OnClick	Pavaizduoja pradinis duomenys Label1 ir Label2 lentelėse.
Button2	Text	"Calculate schoarships"
Button2	OnClick	Prideda pirmūnų stipendijas, parodo Label1 visus studentus su stipendijomis ir be jų, o Label2 parodoma visus studentus su stipendijomis.
Button2	Visibility	False
Button3	Text	"Make List By Group"
Button3	OnClick	Po to, kai įrašytas norimos grupės pavadinimas, į Label1 pavaizduojama toje grupėje esančių studentų informacija.
Button3	Visibility	False
Button4	Text	"Reset To Main Page"
Button4	OnClick	Grįžta į pradinį puslapį
Button4	Visibility	False
Table1	GridLines	
Table2	GridLines	
TextBox1	Get string	Gauna studentų grupę

## 2.4. Klasių diagrama



## 2.5. Programos vartotojo vadovas

Paleidžiama programa. Kairėje esančiame failo pasirinkimo lauke pasirinkite failą, kuris turėtų asmeninę studentų informaciją.

**Failo eilutės struktūra:** Studento numeris sąrašas, pavardė, vardas, telefono numeris.

Dešinėje esančiame failo pasirinkimo lauke pasirinkite failą, kuris turėtų tarnybinę studentų informaciją.

**Failo struktūra:**

Pirmoje eilutėje: stipendijų fondas, pažymių vidurkis norit gauti stipendiją.

**Kitų eilučių struktūra:**

Studento numeris sąrašas, grupė, pažymių skaičius, pažymiai.

Pasirinkus failus galite spausti mygtuką „Load Data“, kad pamatytumėte failuose esančius duomenis. Po mygtuko paspaudimo atsiras sekantis mygtukas „Calculate scholarships“ šio mygtuko paspaudimu, bus sudarytos dvi lentelės.

**Kairėje** bus rodomi visi studentai su savo duomenimis ir jiems skirtomis stipendijomis.

**Dešinėje** bus rodomi tik studentai su savo duomenimis, kurie gavo stipendiją.

Viršutinėje dalyje atsiradusiame rašymo lauke galima įrašyti norimos pirmūnų grupės pavadinimas (Grupės sudarytos iš vienos didžiosios raidės). Įrašius norimą grupę ir paspaudus mygtuką „Make List By Group“, bus rodomi tik toje grupėje esantys pirmūnai.

Paspaudus paskutinį mygtuką „Reset To Main Page“, grįšite į pradinį puslapį.

## 2.6. Programos tekstas

Student.cs:

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

namespace WebApplication3
{
    /// <summary>
    /// A class containing one student's basic information
    /// </summary>
    public class Student
    {
        public int Nr { get; set; }
        public string SurName { get; set; }
        public string Name { get; set; }
        public int Phone { get; set; }
        public double Pirm { get; set; }
        public double CashGiven { get; set; }

        /// <summary>
        /// Basic student's information constructor
        /// </summary>
        /// <param name="Nr"> His number in line </param>
        /// <param name="SurName"> His surname </param>
        /// <param name="Name"> His name </param>
        /// <param name="Phone"> His phonenumber </param>
        public Student(int Nr, string SurName, string Name, int Phone)
        {
            this.Nr = Nr;
            this.SurName = SurName;
            this.Name = Name;
            this.Phone = Phone;
            this.Pirm = 0;
            this.CashGiven = 0;
        }

        /// <summary>
        /// ToString() override
        /// </summary>
        /// <returns> Returnrs a table row of (Students, number, surname,
```





```

private Node head;
private Node tail;
private Node d;

/// <summary>
/// Linklist constructor
/// </summary>
public LinkList()
{
    this.head = null;
    this.tail = null;
}

/// <summary>
/// Find the student with the same number as the given one
/// </summary>
/// <param name="Nr"> Number of the student we want </param>
/// <returns> Basic student's information </returns>
public Student FindNumber(int Nr)
{
    for (Begin(); Exist(); Next())
    {
        if (d.Data.Nr == Nr)
        {
            return d.Data;
        }
    }
    return null;
}

public void SetLinInfo(Student student)
{
    head = new Node(student, head);
}

private void LastElement()
{
    Node curr = head;
    while (curr.Link != null)
    {
        curr = curr.Link;
    }
    tail = curr;
}

/// <summary>
/// Adds a student class object using a backwards linklist formation
/// </summary>
/// <param name="k"> Student we want to add </param>
public void Add_d(Student k)
{
    Node d = new Node();
    d.Data = k;
    d.Link = null;
    if(this.head == null)
    {
        this.head = d;
        this.tail = d;
    }
    else
    {
        this.tail.Link = d;
        this.tail = d;
    }
}

```

```

/// <summary>
/// Adds a student class object using a direct linklist formation
/// </summary>
/// <param name="k"> Student we want to add </param>
public void Add_r(Student k)
{
    Node d = new Node();
    d.Data = k;
    d.Link = this.head;
    this.head = d;
}

/// <summary>
/// Removes a student object that fits the given number
/// </summary>
/// <param name="Nr"> Number of the student object we want to remove </param>
public void Remove(int Nr)
{
    Node d, s;
    d = s = head;
    if (s.Link != null || d == null)
    {
        while (s.Link != null)
        {
            s = d.Link;
            if (d.Data.Nr == Nr)
            {
                d.Data = s.Data;
                d.Link = s.Link;
                s = null;

                break;
            }
            if (s.Data.Nr == Nr && s.Link == null)
            {
                s = null;
                d.Link = s;

                break;
            }
            d = s;
        }
    }
    else
    {
        if (d != null)
        {
            d = null;
        }
    }
}

/// <summary>
/// Sorts linklist first of all by scholarship size, then by surname and then by
name
/// </summary>
public void Sort()
{
    if (head == null)
    {
        return;
    }
    bool done = true;
    while (done)
    {

```



```

        done = false;
        var headn = head;
        while (headn != null)
        {
            if (headn.Link != null && headn.Data.CompareTo(headn.Link.Data) > 0)
            {
                Student St = headn.Data;
                headn.Data = headn.Link.Data;
                headn.Link.Data = St;
                done = true;
            }
            headn = headn.Link;
        }
    }

    /// <summary>
    /// Sets d as the begining of the linklist
    /// </summary>
    public void Begin()
    {
        d = head;
    }
    /// <summary>
    /// Set d as the next node in the linklist
    /// </summary>
    public void Next()
    {
        d = d.Link;
    }
    /// <summary>
    /// Sees if d is not the last node of the linklist
    /// </summary>
    /// <returns></returns>
    public bool Exist()
    {
        return d != null;
    }

    /// <summary>
    /// Brings d value out of the linklist
    /// </summary>
    /// <returns> Student that d is linked to </returns>
    public Student Get()
    {
        return d.Data;
    }
}
}

```

//  
-----  
//

## Linklist2.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebApplication3
{
    /// <summary>
    /// A linklist class of official student information

```

```

/// </summary>
public sealed class LinkList2
{
    /// <summary>
    /// A node class, that will be used in the creation of a linklist
    /// where the official student information will be held as
    /// </summary>
    private sealed class Node
    {
        public StudentInformation Data { get; set; }
        public Node Link { get; set; }
        public Node()
        {

        }

        /// <summary>
        /// Node's constructor
        /// </summary>
        /// <param name="value"> StudentInformation class object </param>
        /// <param name="address"> line number </param>
        public Node(StudentInformation value, Node address)
        {
            this.Data = value;
            this.Link = address;
        }
    }

    private Node head;
    private Node tail;
    public int FundSize { get; set; }
    public double Vid { get; set; }
    private Node d;

    /// <summary>
    /// LinkList constructor
    /// </summary>
    public LinkList2()
    {
        this.head = null;
        this.tail = null;
        this.d = null;
        this.FundSize = 0;
        this.Vid = 0;
    }

    /// <summary>
    /// Adds given cash amount for all scholarships, and the grade
    /// the student needs to beat in order to get the scholarship
    /// </summary>
    /// <param name="Size"> given cash amount for all scholarships </param>
    /// <param name="Vid"> the grade
    /// the student needs to beat in order to get the scholarship </param>
    public void AddFundSizeAndVid(int Size, double Vid)
    {
        this.FundSize = Size;
        this.Vid = Vid;
    }

    public void SetLinInfo(StudentInformation student)
    {
        head = new Node(student, head);
    }

    private void LastElement()
    {
        Node curr = head;
        while (curr.Link != null)

```

```

        {
            curr = curr.Link;
        }
        tail = curr;
    }

    /// <summary>
    /// Find the student's object with the same number as the given one
    /// </summary>
    /// <param name="Nr"> Number of the studentinformation's object we want to get
</param>
    /// <returns> Official student's information (studentinformation class object)
</returns>
    public StudentInformation FindNumber(int Nr)
    {
        for(Begin(); Exist(); Next())
        {
            if(d.Data.Nr == Nr)
            {
                return d.Data;
            }
        }
        return null;
    }

    /// <summary>
    /// Adds a studentinformation class object using a backwards linklist formation
    /// </summary>
    /// <param name="k"> Student class object we want to add </param>
    public void Add_d(StudentInformation k)
    {
        Node d = new Node();
        d.Data = k;
        d.Link = null;
        if (this.head == null)
        {
            this.head = d;
            this.tail = d;
        }
        else
        {
            this.tail.Link = d;
            this.tail = d;
        }
    }

    /// <summary>
    /// Adds a studentinformation class object using a direct linklist formation
    /// </summary>
    /// <param name="k"> student information class object we want to add </param>
    public void Add_r(StudentInformation k)
    {
        Node d = new Node();
        d.Data = k;
        d.Link = this.head;
        this.head = d;
    }

    /// <summary>
    /// Removes a studentinformation class object that fits the given number
    /// </summary>
    /// <param name="Nr"> Number of the studentinformation object we want to remove
</param>
    public void Remove(int Nr)
    {
        Node d, s;
        d = s = head;
        if (s.Link != null || d == null)
        {

```





```

        double SumIndex = 0;
        for (List1.Begin(); List1.Exist(); List1.Next())
        {
            Student student = List1.Get();
            SumIndex += student.Pirm;
        }
        return SumIndex;
    }

    /// <summary>
    /// Adds schoralship amount to each student
    /// </summary>
    /// <param name="List1"> linklist with all student class objects </param>
    /// <param name="List2"> linklist with all studentinformation class objects
</param>
    public static void AddFunds(LinkList List1, LinkList2 List2)
    {
        double SumIndex = TaskUtils.CalculateIndex(List1, List2);
        double IndexWorth = List2.FundSize / SumIndex;
        for (List1.Begin(); List1.Exist(); List1.Next())
        {
            Student student = List1.Get();
            student.CashGiven = student.Pirm * IndexWorth;
        }
    }

    /// <summary>
    /// Removes students that have no scholarship from both linklists
    /// </summary>
    /// <param name="List1"> linklist with all student class objects </param>
    /// <param name="List2"> linklist with all studentinformation class objects
</param>
    public static void RemoveStudents(LinkList List1, LinkList2 List2)
    {
        bool reset = true;
        while(reset)
        {
            reset = false;
            for (List1.Begin(); List1.Exist(); List1.Next())
            {
                Student student = List1.Get();
                if (student.CashGiven == 0)
                {
                    List2.Remove(student.Nr);
                    List1.Remove(student.Nr);
                    reset = true;
                    break;
                }
            }
        }
    }

    }

    /// <summary>
    ///
    /// </summary>
    /// <param name="List1"> linklist with all student class objects </param>
    /// <param name="List2"> linklist with all studentinformation class objects
</param>
    /// <param name="GroupNeeded"> linklist of student class objects
    /// that are all from the same group </param>
    /// <returns> linklist of student class objects that
    /// are all from the same group </returns>
    public static LinkList FormGroupList(LinkList List, LinkList2 Info, string
GroupNeeded)
    {

```

```

        LinkedList GroupedList = new LinkedList();
        for (Info.Begin(); Info.Exist(); Info.Next())
        {
            StudentInformation info = Info.Get();
            if(info.Group == GroupNeeded)
            {
                Student student = List.FindNumber(info.Nr);
                GroupedList.Add_d(student);
            }
        }
        return GroupedList;
    }
}

//
//-----
//

```

## Form.cs:

```

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

namespace WebApplication3
{
    public partial class Form1 : System.Web.UI.Page
    {
        /// <summary>
        /// Creates 2 tables of two different linklists. First
        /// one contains basic student information.
        /// The second one contains official student information.
        /// </summary>
        /// <param name="table"> Table element where the basic
        /// information will be written in </param>
        /// <param name="table2"> Table element where the official
        /// information will be written in </param>
        /// <param name="List"> Basic student information linklist </param>
        /// <param name="List2"> Official student information linklist </param>
        public static void FillTableWithAllData(ref Table table,
            ref Table table2, LinkedList List, LinkedList List2)
        {
            var firstRow = new TableRow();
            var firstRow2 = new TableRow();

            firstRow.Cells.Add(new TableCell()
            {
                Text = "Nr",
                CssClass = "bold-text"
            });
            firstRow.Cells.Add(new TableCell()
            {
                Text = "Pavardé",
                CssClass = "bold-text"
            });
            firstRow.Cells.Add(new TableCell()
            {
                Text = "Vardas",
                CssClass = "bold-text"
            });

```

```

));
firstRow.Cells.Add(new TableCell()
{
    Text = "Telefono numeris",
    CssClass = "bold-text"
});

table.Rows.Add(firstRow);

for (List.Begin(); List.Exist(); List.Next())
{
    var row = new TableRow();

    Student Student = List.Get();
    row.Cells.Add(new TableCell()
    {
        Text = Student.Nr.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = Student.SurName.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = Student.Name.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = Student.Phone.ToString()
    }
    );

    table.Rows.Add(row);
}

firstRow2.Cells.Add(new TableCell()
{
    Text = "Nr",
    CssClass = "bold-text"
});
firstRow2.Cells.Add(new TableCell()
{
    Text = "Grupė",
    CssClass = "bold-text"
});
firstRow2.Cells.Add(new TableCell()
{
    Text = "Pažymių kiekis",
    CssClass = "bold-text"
});
firstRow2.Cells.Add(new TableCell()
{
    Text = "Pažymiai",
    CssClass = "bold-text"
});

table2.Rows.Add(firstRow2);

```



```

for (List2.Begin(); List2.Exist(); List2.Next())
{
    var row = new TableRow();

    StudentInformation Info = List2.Get();
    row.Cells.Add(new TableCell()
    {
        Text = Info.Nr.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Info.Group.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Info.Grades.Count().ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Info.ToString()
    });
    table2.Rows.Add(row);
}
}
/// <summary>
/// Table creator, that makes a table of mixed student's information
/// </summary>
/// <param name="table"> Table element where the
/// information will be written in </param>
/// <param name="List"> Basic information linklist </param>
/// <param name="List2"> Official information linklist </param>
public static void FillTableWithStudentsData(ref Table table,
    LinkList List, LinkList2 List2)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Pavardė",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Vardas",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Pažymiai",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Stipendijos dydis",
        CssClass = "bold-text"
    });
}
}

```

```

table.Rows.Add(firstRow);

for (List.Begin(); List.Exist(); List.Next())
{
    var row = new TableRow();

    Student Student = List.Get();
    StudentInformation Info = List2.FindNumber(Student.Nr);

    row.Cells.Add(new TableCell()
    {
        Text = Student.SurName.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Student.Name.ToString()
    });

    row.Cells.Add(new TableCell()
    {
        Text = Info.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Student.CashGiven.ToString()
    });
    table.Rows.Add(row);
}

}

/// <summary>
/// Table creator, that makes a table of mixed student's information
/// </summary>
/// <param name="table"> Table element where the
/// information will be written in </param>
/// <param name="List"> Basic information linklist </param>
/// <param name="List2"> Official information linklist </param>
public static void FillTableWithForerunnerData(ref Table table,
    LinkList List, LinkList2 List2)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Pavardė",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Vardas",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Pažymiai",

```



```

public static LinkList ReadFile1(string FileName)
{
    try
    {
        string[] Lines = File.ReadAllLines(FileName, Encoding.ASCII);
        LinkList List = new LinkList();
        for (int i = 0; i < Lines.Count(); i++)
        {
            string[] Words = Lines[i].Split(' ');
            int Nr = int.Parse(Words[0]);
            string SurName = Words[1];
            string Name = Words[2];
            int Phone = int.Parse(Words[3]);
            List.Add_d(new Student(Nr, SurName, Name, Phone));
        }
        return List;
    }
    catch
    {
        return null;
    }
}

/// <summary>
/// Reads official student information
/// </summary>
/// <param name="FileName"> The name of the file, where
/// the information is written in </param>
/// <returns> A LinkList, that contains student information </returns>
public static LinkList2 ReadFile2(string FileName)
{
    try
    {
        string[] Lines = File.ReadAllLines(FileName, Encoding.ASCII);
        LinkList2 List = new LinkList2();
        List.AddFundSizeAndVid(int.Parse(Lines[0].Split(' ')[0]),
            int.Parse(Lines[0].Split(' ')[1]));
        for (int i = 1; i < Lines.Count(); i++)
        {
            List<int> Grades = new List<int>();
            string[] Words = Lines[i].Split(' ');
            int Nr = int.Parse(Words[0]);
            string Group = Words[1];
            int Count = int.Parse(Words[2]);
            for (int l = 3; l < Count + 3; l++)
            {
                Grades.Add(int.Parse(Words[l]));
            }
            List.Add_d(new StudentInformation(Nr, Group, Count, Grades));
        }
        return List;
    }
    catch
    {
        return null;
    }
}

/// <summary>
/// Creates a table of basic student information (Number, Surname, Name, Phone)
/// </summary>
/// <param name="Filename"> File where the tabel will be created </param>
/// <param name="List"> Linked list of basic information
/// which the table will be made of </param>
/// <param name="Header"> A string above the table </param>
public static void WriteData(string Filename, LinkList List, string Header)
{
    File.AppendAllText(Filename, String.Format(Header + "\n"));
}

```

```

File.AppendAllText(Filename, String.Format(new string('-', 59) + "\n"));
File.AppendAllText(Filename, String.Format("| {0, -8} |" +
    " {1, -12} | {2, -14} | {3, -12} |\n", "Numeris",
    "Paverdė", "Vardas", "Telefonas"));
File.AppendAllText(Filename, String.Format(new string('-', 59) + "\n"));
for(List.Begin(); List.Exist(); List.Next())
{
    File.AppendAllText(Filename, String.Format(List.Get().ToString()));
}
File.AppendAllText(Filename, String.Format(new string('-', 59) + "\n" + "\n"));
}

/// <summary>
/// Creates a table of official student information (Number,
/// Group, How many grades you have, All the grades)
/// </summary>
/// <param name="Filename"> File where the tabel will be created </param>
/// <param name="List"> Linked list of official information
/// which the table will be made of </param>
/// <param name="Header"> A string above the table </param>
public static void WriteInfo(string Filename, LinkList2 List, string Header)
{
    File.AppendAllText(Filename, String.Format(Header + "\n"));
    File.AppendAllText(Filename, String.Format(new string('-', 67) + "\n"));
    File.AppendAllText(Filename, String.Format("| {0, -8} |" +
        " {1, -12} | {2, -14} | {3, -20} |\n", "Numeris",
        "Grupė", "Pažymių kiekis", "Pažymiai"));
    File.AppendAllText(Filename, String.Format(new string('-', 67) + "\n"));
    for (List.Begin(); List.Exist(); List.Next())
    {
        StudentInformation Info = List.Get();
        File.AppendAllText(Filename, String.Format("| {0, -8} " +
            " | {1, -12} | {2, -14} | {3, -20} |\n", Info.Nr,
            Info.Group, Info.Count, Info.ToString()));
    }
    File.AppendAllText(Filename, String.Format(new string('-', 67) + "\n" + "\n"));
}

/// <summary>
/// Creates a table of students (Number, Surname, Name, Grades, And their
scholarship)
/// </summary>
/// <param name="Filename"> File where the table will be created in </param>
/// <param name="List"> Basic student information containing linklist </param>
/// <param name="Info"> Official student information containing linklist </param>
/// <param name="Header"> A string above the table </param>
public static void WriteAllStudentsBasicInfo(string Filename,
LinkList List, LinkList2 Info, string Header)
{
    File.AppendAllText(Filename, String.Format(Header + "\n"));
    File.AppendAllText(Filename, String.Format(new string('-', 81) + "\n"));
    File.AppendAllText(Filename, String.Format("| {0, -8} | {1, -12} " +
        " | {2, -14} | {3, -20} | {4, -12} |\n", "Numeris",
        "Paverdė", "Vardas", "Pažymiai", "Stipendija"));
    File.AppendAllText(Filename, String.Format(new string('-', 81) + "\n"));
    for (List.Begin(); List.Exist(); List.Next())
    {
        Student student = List.Get();
        File.AppendAllText(Filename, String.Format("| {0, -8} |" +
            " {1, -12} | {2, -14} | {3, -20} | {4, -12} |\n",
            student.Nr, student.SurName, student.Name,
            Info.FindNumber(student.Nr).ToString(), student.CashGiven));
    }
    File.AppendAllText(Filename, String.Format(new string('-', 81) + "\n" + "\n"));
}

```

```

    /// <summary>
    /// Creates a table of students who are fruntrunners in their group (Surname, Name,
    Grades)
    /// </summary>
    /// <param name="Filename"> File where the table will be created in </param>
    /// <param name="List"> Basic student information containing linklist </param>
    /// <param name="Info"> Official student information containing linklist </param>
    /// <param name="Header"> A string above the table </param>
    public static void WriteFrontRunners(string Filename, LinkList List,
        LinkList2 Info, string Header)
    {
        {
            File.AppendAllText(Filename, String.Format(Header + "\n"));
            File.AppendAllText(Filename, String.Format(new string('-', 56) + "\n"));
            File.AppendAllText(Filename, String.Format("| {0, -12} |" +
                " {1, -14} | {2, -20} |\n", "Paverdè", "Vardas", "Pažymiai"));
            File.AppendAllText(Filename, String.Format(new string('-', 56) + "\n"));
            for (List.Begin(); List.Exist(); List.Next())
            {
                Student student = List.Get();
                File.AppendAllText(Filename, String.Format("| {0, -12} |" +
                    " {1, -14} | {2, -20} |\n", student.SurName, student.Name,
                    Info.FindNumber(student.Nr).ToString()));
            }
            File.AppendAllText(Filename, String.Format(new string('-', 56) + "\n" +
                "\n"));
        }
    }
}

////////////////////////////////////
-----
////////////////////////////////////

```

## Web.aspx.cs:

```

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

namespace WebApplication3
{
    public partial class Web : System.Web.UI.Page
    {
        private string DataFile1, DataFile2, ResultFile;
        private LinkList List;
        private LinkList2 Info;

        protected void Page_Load(object sender, EventArgs e)
        {
            DataFile1 = Server.MapPath("App_Data/U14a.txt");
            DataFile2 = Server.MapPath("App_Data/U14b.txt");
            ResultFile = Server.MapPath("App_Data/Results.txt");
            Label4.Visible = false;
        }
    }
}

```

```

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (FileUpload2.HasFile && FileUpload1.HasFile &&
            Path.GetExtension(FileUpload1.FileName) == ".txt" &&
            Path.GetExtension(FileUpload2.FileName) == ".txt")
        {

            if (File.Exists(Server.MapPath("App_Data/Results.txt")))
            {
                File.Delete(Server.MapPath("App_Data/Results.txt"));
                ResultFile = Server.MapPath("App_Data/Results.txt");
            }

            FileUpload1.SaveAs(DataFile1);
            FileUpload2.SaveAs(DataFile2);

            if (new FileInfo(DataFile1).Length == 0 &&
                new FileInfo(DataFile2).Length == 0 &&
                InOutUtils.ReadFile1(DataFile1) == null &&
                InOutUtils.ReadFile2(DataFile2) == null
                )
            {

                Label4.Text = "* Failai/Failas, kurį pasirinkote" +
                    " turi būti neteisinga informacijos išdėstymą";
                Label4.Visible = true;
            }
            else
            {

                List = InOutUtils.ReadFile1(DataFile1);
                Info = InOutUtils.ReadFile2(DataFile2);

                Table1.Visible = true;
                Table2.Visible = true;

                Form1.FillTableWithAllData(ref Table1, ref Table2, List, Info);

                InOutUtils.WriteData(ResultFile, List, "Pradiniai studentų
duomenys:");
                InOutUtils.WriteInfo(ResultFile, Info, "Pradiniai studentų
duomenys:");

                Button2.Visible = true;
                Button1.Visible = false;
                FileUpload1.Visible = false;
                FileUpload2.Visible = false;
                Table1.Visible = true;
                Table2.Visible = true;
                Label2.Text = "Studentų asmeniniai duomenys";
                Label3.Text = "Studentų tarnybinė informacija";
                Label2.Visible = true;
                Label3.Visible = true;
                Label5.Visible = false;
                Label6.Visible = false;
            }
        }
    }
    else

```

```

    {
        Label4.Visible = true;
        Label4.Text = "* Failai/Failas, kuri pasirinkote buvo" +
            " neteisingo formato !!! Bandykite dar kartą.";
    }
}

```

```
protected void Button2_Click(object sender, EventArgs e)
```

```

{
    List = InOutUtils.ReadFile1(DataFile1);
    Info = InOutUtils.ReadFile2(DataFile2);

    TaskUtils.Distribution(List, Info);
    TaskUtils.AddFunds(List, Info);
    List.Sort();

    Form1.FillTableWithStudentsData(ref Table1, List, Info);
    InOutUtils.WriteAllStudentsBasicInfo(ResultFile, List, Info,
        "Studentai po stendijos skaičiavimo:");

    TaskUtils.RemoveStudents(List, Info);
    Form1.FillTableWithStudentsData(ref Table2, List, Info);
    InOutUtils.WriteAllStudentsBasicInfo(ResultFile, List, Info,
        "Studentai tik su stipendijomis:");

    Table1.Visible = true;
    Table2.Visible = true;
    Label2.Text = "Studentų sąrašas";
    Label3.Text = "Sąrašas be studentų, kurie neturi stipendijos";
    TextBox1.Visible = true;
    Button2.Visible = false;
    Button3.Visible = true;
}

```

```
protected void Button3_Click(object sender, EventArgs e)
```

```

{
    Label2.Visible = false;
    Label3.Visible = false;
    if (TextBox1.Text.Count() == 1)
    {
        Table2.Visible = true;
        Label2.Visible = true;

        List = InOutUtils.ReadFile1(DataFile1);
        Info = InOutUtils.ReadFile2(DataFile2);

        //How to fix this so List and Info would save
        TaskUtils.Distribution(List, Info);
        TaskUtils.AddFunds(List, Info);
        List.Sort();
        TaskUtils.RemoveStudents(List, Info);

        LinkList ListByGroup = TaskUtils.FormGroupList(List, Info,
TextBox1.Text);
        Form1.FillTableWithForerunnerData(ref Table1, ListByGroup, Info);
        Label2.Text = "Grupės " + TextBox1.Text + " pirmūnai";
        InOutUtils.WriteFrontRunners(ResultFile,
            ListByGroup, Info, "Group's " + TextBox1.Text + " pirmūnai: ");

        TextBox1.Text = "";
        Button4.Visible = true;
    }
}

```



```

    }
    else
    {
        Label4.Text = "* Parašėte neteisingą grupės formatą" +
            " !!! (Grupės sudarytos iš vienos raidės)";
        Label4.Visible = true;
    }
}

protected void Button4_Click(object sender, EventArgs e)
{
    FileUpload1.Visible = true;
    FileUpload2.Visible = true;
    Button1.Visible = true;
    Label2.Visible = false;
    TextBox1.Visible = false;
    Table1.Visible = false;
    Button3.Visible = false;
    Button4.Visible = false;
    Label5.Visible = true;
    Label6.Visible = true;
}
}
}

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

PageStyle.css:

```

.auto-style1 {
    width: 100%;
    padding: 10px;
    background-color: burlywood;
    border-style: solid;
    border-color: black;
}

body {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: antiquewhite;
}

.auto-style2 {
    height: 29px;
}

.auto-style3 {
    height: 23px;
}

.Tableless {
    background-color: #FF9933;
    border-color: Black;
    border-style: Solid;
    border-width: 5px;
}

```

```
.auto-style5 {
    height: 22px;
    justify-content: center;
    align-items: center;
}
```

## 2.7. Pradiniai duomenys ir rezultatai

### Testas Nr. 1:

**Tikslas:** Standartinis atvejis, kai apskaičiuojami keli atsakymai.

#### U14a.txt

```
12 Redis Jokubas 86989745
79 Regis Petris 86784588
78 Delgis Romis 89777845
41 Kukutis Jonas 86784558
79 Remis Dodukas 88888888
120 Dalgis Katinas 121321333
1 Redis Astolinis 877874455
```

#### U14b.txt

```
800 6
41 B 4 9 8 9 9
12 C 1 2
79 B 4 9 9 8 9
78 B 4 5 7 5 9
1 Z 3 5 5 3
79 C 5 9 9 9 9 8
120 C 4 4 10 9 9
```

#### Results.txt

Pradiniai studentų duomenys:

Numeris	Pavardė	Vardas	Telefonas
12	Redis	Jokubas	86989745
79	Regis	Petris	86784588
78	Delgis	Romis	89777845
41	Kukutis	Jonas	86784558
79	Remis	Dodukas	88888888
120	Dalgis	Katinas	121321333
1	Redis	Astolinis	877874455

Pradiniai studentų duomenys:

Numeris	Grupė	Pažymių kiekis	Pažymiai
41	B	4	9, 8, 9, 9
12	C	1	2
79	B	4	9, 9, 8, 9
78	B	4	5, 7, 5, 9
1	Z	3	5, 5, 3

79	C	5	9, 9, 9, 9, 8	
120	C	4	4, 10, 9, 9	

-----

Studentai po stiendijos skaičiavimo:

-----

-				
Numeris	Paverdė	Vardas	Pažymiai	Stipendija
-----				
-				
41	Kukutis	Jonas	9, 8, 9, 9	
204,651162790698				
79	Regis	Petris	9, 9, 8, 9	
204,651162790698				
79	Remis	Dodukas	9, 9, 8, 9	
204,651162790698				
78	Delgis	Romis	5, 7, 5, 9	
186,046511627907				
120	Dalgis	Katinas	4, 10, 9, 9	0
1	Redis	Astolinis	5, 5, 3	0
12	Redis	Jokubas	2	0
-----				
-				

Studentai tik su stipendijomis:

-----

-				
Numeris	Paverdė	Vardas	Pažymiai	Stipendija
-----				
-				
41	Kukutis	Jonas	9, 8, 9, 9	
204,651162790698				
79	Regis	Petris	9, 9, 8, 9	
204,651162790698				
79	Remis	Dodukas	9, 9, 8, 9	
204,651162790698				
78	Delgis	Romis	5, 7, 5, 9	
186,046511627907				
-----				
-				

Group's B pirmūnai:

-----

Paverdė	Vardas	Pažymiai	
-----			
Kukutis	Jonas	9, 8, 9, 9	
Regis	Petris	9, 9, 8, 9	
Delgis	Romis	5, 7, 5, 9	
-----			

Group's C pirmūnai:

-----

Paverdė	Vardas	Pažymiai	
-----			
Regis	Petris	9, 9, 8, 9	
-----			
Group's Z pirmūnai:			
-----			
Paverdė	Vardas	Pažymiai	
-----			
-----			

User interface

## Studentų informacijos tvarkyklė

Pasirinkite studentų asmeninių duomenų failą
Pasirinkite studentų tarnybinių duomenų failą

Choose File No file chosen

Choose File No file chosen

Load Data

## Studentų informacijos tvarkyklė

Studentų asmeniniai duomenys

Nr	Pavardė	Vardas	Telefono numeris
12	Redis	Jokubas	86989745
79	Regis	Petris	86784588
78	Delgis	Romis	89777845
41	Kukutis	Jonas	86784558
79	Remis	Dodukas	88888888
120	Dalgis	Katinas	121321333
1	Redis	Astolinis	877874455

Calculate scholarships

Studentų tarnybinė informacija

Nr	Grupė	Pažymių kiekis	Pažymiai
41	B	4	9, 8, 9, 9
12	C	1	2
79	B	4	9, 9, 8, 9
78	B	4	5, 7, 5, 9
1	Z	3	5, 5, 3
79	C	5	9, 9, 9, 9, 8
120	C	4	4, 10, 9, 9

## Studentų informacijos tvarkyklė

### Studentų sąrašas

Pavardė	Vardas	Pažymiai	Stipendijos dydis
Kukutis	Jonas	9, 8, 9, 9	204,651162790698
Regis	Petris	9, 9, 8, 9	204,651162790698
Remis	Dodukas	9, 9, 8, 9	204,651162790698
Delgis	Romis	5, 7, 5, 9	186,046511627907
Dalgis	Katinas	4, 10, 9, 9	0
Redis	Astolinis	5, 5, 3	0
Redis	Jokubas	2	0

### Sąrašas be studentų, kurie neturi stipendijos

Pavardė	Vardas	Pažymiai	Stipendijos dydis
Kukutis	Jonas	9, 8, 9, 9	204,651162790698
Regis	Petris	9, 9, 8, 9	204,651162790698
Remis	Dodukas	9, 9, 8, 9	204,651162790698
Delgis	Romis	5, 7, 5, 9	186,046511627907

Make List By Group

### Testas Nr. 2:

**Tikslas:** Parodyti programos veikimą, kai failuose nėra duomenų.

U14a.txt

U14b.txt

Results.txt (Rezultatų failas nebuvo sukurtas)

## Studentų informacijos tvarkyklė

Pasirinkite studentų asmeninių duomenų failą

No file chosen

\* Failai/Failas, kurį pasirinkote buvo neteisingo formato !!! Bandykite dar kartą.

Pasirinkite studentų tarnybinių duomenų failą

No file chosen

## 2.8. Dėstytojo pastabos

### 3. Bendrinės klasės ir testavimas (L3)

#### 3.1. Darbo užduotis

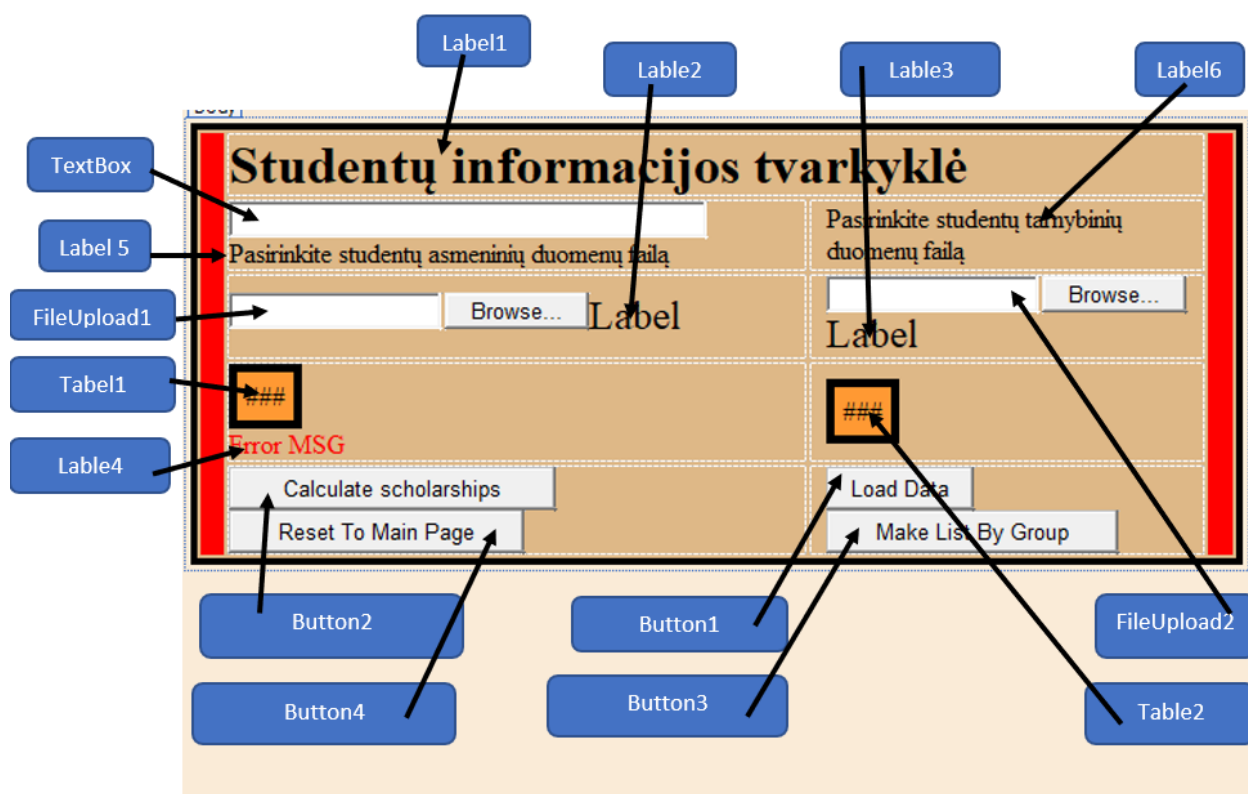
**LD\_14. Stipendijos.** Studentų stipendijoms yra skiriamas nurodyto dydžio fondas. Studentui skiriama stipendija, jei jo pažymių vidurkis viršija nurodytą dydį ir jis neturi skolų (visi pažymiai >4). Studentui skiriama 10% didesnė stipendija, jei jo visi pažymiai didesni už 8. Toks studentas vadinamas pirmūnu. Paskirstykite studentams stipendijas pagal duotą fondą. Fondą reikia maksimaliai išnaudoti, bet negalima viršyti fondo dydžio.

Duomenys:

- Tekstiniame faile U14a.txt duoti studentų asmeniniai duomenys: studento eilės numeris, pavardė vardas, telefono numeris.
- Tekstiniame faile U14b.txt pateikta studentų tarnybinė informacija. Pirmojoje failo eilutėje nurodytas stipendijų fondo dydis ir pažymių vidurkis stipendijai gauti. Tolimesnėse eilutėse tokia informacija: studento eilės numeris, kuris sutampa su eilės numeriu U14a.txt faile, grupė, pažymių kiekis, pažymiai.

Spausdinamas sąrašas (studento pavardė, vardas, pažymiai, stipendijos dydis) turi būti surikiuotas pagal stipendijų dydį, pavardes ir vardus. Iš sąrašo pašalinkite studentus, kurie negauna stipendijos. Sudarykite ir atspausdinkite nurodytos grupės (įvedama klaviatūra) pirmūnų sąrašą (studento pavardė, vardas, pažymiai).

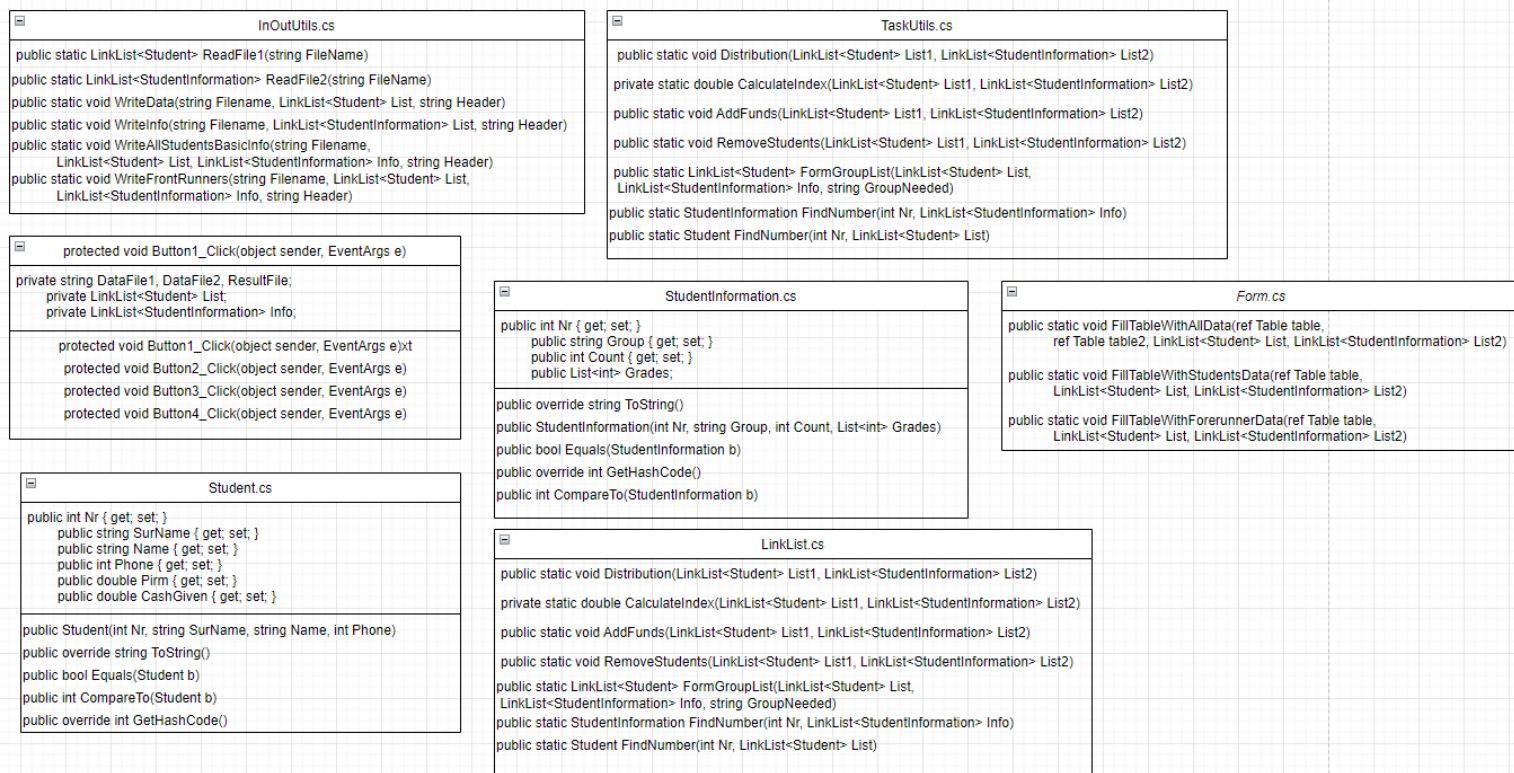
#### 3.2. Grafinės vartotojo sąsajos schema



### 3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
FileUploader1	Get file	
FileUploader2	Get file	
Lable1	Text	"Studentų informacijos tvarkyklė"
Lable2	Text	"Studentų asmeniniai duomenys" // "Studentų sąrašas" // "Grupės " + TextBox1.Text + " pirmūnai"
Lable3	Text	"Studentų tarnybinė informacija" // "Sąrašas be studentų, kurie neturi stipendijos"
Lable4	Text	"* Failai/Failas, kuri pasirinkote turėjo neteisingą informacijos išdėstymą" // "* Failai/Failas, kuri pasirinkote buvo neteisingo formato !!! Bandykite dar kartą." // "* Parašėte neteisingą grupės formatą !!! (Grupės sudarytos iš vienos raidės)"
Lable5	Text	"Pasirinkite studentų asmeninių duomenų failą"
Lable6	Text	"Pasirinkite studentų tarnybinių duomenų failą"
Button1	Text	"Load Data"
Button1	OnClick	Pavaizduoja pradinis duomenys Label1 ir Label2 lentelėse.
Button2	Text	"Calculate schoarships"
Button2	OnClick	Prideda pirmūnų stipendijas, parodo Label1 visus studentus su stipendijomis ir be jų, o Label2 parodoma visus studentus su stipendijomis.
Button2	Visibility	False
Button3	Text	"Make List By Group"
Button3	OnClick	Po to, kai įrašytas norimos grupės pavadinimas, į Label1 pavaizduojama toje grupėje esančių studentų informacija.
Button3	Visibility	False
Button4	Text	"Reset To Main Page"
Button4	OnClick	Grįžta į pradinį puslapį
Button4	Visibility	False
Table1	GridLines	
Table2	GridLines	
TextBox1	Get string	Gauna studentų grupę

### 3.4. Klasių diagrama



### 3.5. Programos vartotojo vadovas

Paleidžiama programa. Kairėje esančiame failo pasirinkimo lauke pasirinkite failą, kuris turėtų asmeninę studentų informaciją.

**Failo eilutės struktūra:** Studento numeris sąrašas, pavardė, vardas, telefono numeris.

Dešinėje esančiame failo pasirinkimo lauke pasirinkite failą, kuris turėtų tarnybinę studentų informaciją.

**Failo struktūra:**

Pirmoje eilutėje: stipendijų fondas, pažymių vidurkis norit gauti stipendiją.

**Kitų eilučių struktūra:**

Studento numeris sąrašas, grupė, pažymių skaičius, pažymiai.

Pasirinkus failus galite spausti mygtuką „Load Data“, kad pamatytumėte failuose esančius duomenis. Po mygtuko paspaudimo atsiras sekantis mygtukas „Calculate scholarships“ šio mygtuko paspaudimu, bus sudarytos dvi lentelės.

**Kairėje** bus rodomi visi studentai su savo duomenimis ir jiems skirtomis stipendijomis.

**Dešinėje** bus rodomi tik studentai su savo duomenimis, kurie gavo stipendiją.



Viršutinėje dalyje atsiradusiame rašymo lauke galima įrašyti norimos pirmūnų grupės pavadinimas (Grupės sudarytos iš vienos didžiosios raidės). Įrašius norimą grupę ir paspaudus mygtuką „Make List By Group“, bus rodomi tik toje grupėje esantys pirmūnai.

Paspaudus paskutinį mygtuką „Reset To Main Page“, grįšite į pradinį puslapį.

### 3.6. Programos tekstas

Student.cs:

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

namespace WebApplication3
{
    /// <summary>
    /// A class containing one student's basic information
    /// </summary>
    public class Student : IComparable<Student>, IEquatable<Student>
    {
        public int Nr { get; set; }
        public string SurName { get; set; }
        public string Name { get; set; }
        public int Phone { get; set; }
        public double Pirm { get; set; }
        public double CashGiven { get; set; }

        /// <summary>
        /// Basic student's information constructor
        /// </summary>
        /// <param name="Nr"> His number in line </param>
        /// <param name="SurName"> His surname </param>
        /// <param name="Name"> His name </param>
        /// <param name="Phone"> His phonenumber </param>
        public Student(int Nr, string SurName, string Name, int Phone)
        {
            this.Nr = Nr;
            this.SurName = SurName;
            this.Name = Name;
            this.Phone = Phone;
            this.Pirm = 0;
            this.CashGiven = 0;
        }

        public bool Equals(Student b)
        {
            if (this.Nr.Equals(b.Nr))
            {
                return true;
            }
        }
    }
}
```



```

namespace WebApplication3
{
    /// <summary>
    /// Class that containing one student's official information
    /// </summary>
    public class StudentInformation : IEquatable<StudentInformation>,
    IComparable<StudentInformation>
    {

        public int Nr { get; set; }
        public string Group { get; set; }
        public int Count { get; set; }
        public List<int> Grades;

        /// <summary>
        /// Official student information constructor
        /// </summary>
        /// <param name="Nr"> Line number </param>
        /// <param name="Group"> Group </param>
        /// <param name="Count"> Grade count </param>
        /// <param name="Grades"> Grade List </param>
        public StudentInformation(int Nr, string Group, int Count, List<int> Grades)
        {
            this.Nr = Nr;
            this.Group = Group;
            this.Count = Count;
            this.Grades = Grades;
        }

        /// <summary>
        /// ToString() override
        /// </summary>
        /// <returns> A string with all the student grades in </returns>
        public override string ToString()
        {
            string line = "";
            for (int i = 0; i < Grades.Count; i++)
            {
                if(i == 0)
                {
                    line = line + Grades[i].ToString();
                }
                else
                {
                    line = line + ", " + Grades[i].ToString();
                }
            }
            return line;
        }

        /// <summary>
        /// Sees if one students information is equal to the other students
        /// </summary>
        /// <param name="b"> Studetns information that we are comparing with </param>
        /// <returns> True if the students number maches, false if it doesn't </returns>
        public bool Equals(StudentInformation b)
        {
            if (this.Nr.Equals(b.Nr))
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}

```



```

        this.Data = value;
        this.Link = address;
    }
}

private Node<type> head;
private Node<type> tail;
private Node<type> d;
public int FundSize { get; set; }
public double Vid { get; set; }

public void AddFundSizeAndVid(int Size, double Vid)
{
    this.FundSize = Size;
    this.Vid = Vid;
}

/// <summary>
/// Linklist constructor
/// </summary>
public LinkList()
{
    this.head = null;
    this.tail = null;
    this.d = null;
}

public void SetLininfo(type student)
{
    head = new Node<type>(student, head);
}

private void LastElement()
{
    Node<type> curr = head;
    while (curr.Link != null)
    {
        curr = curr.Link;
    }
    tail = curr;
}

/// <summary>
/// Adds a class object using a backwards linklist formation
/// </summary>
/// <param name="k"> Object we want to add </param>
public void Add_d(type k)
{
    Node<type> d = new Node<type>();
    d.Data = k;
    d.Link = null;
    if(this.head == null)
    {
        this.head = d;
        this.tail = d;
    }
    else
    {
        this.tail.Link = d;
        this.tail = d;
    }
}

```

```

    }
}

/// <summary>
/// Adds a class object using a direct linklist formation
/// </summary>
/// <param name="k"> Object we want to add </param>
public void Add_r(type k)
{
    Node<type> d = new Node<type>();
    d.Data = k;
    d.Link = this.head;
    this.head = d;
}

/// <summary>
/// Removes an object that fits the given number
/// </summary>
/// <param name="Nr"> Number of the object we want to remove </param>
public void Remove(type Nr)
{
    Node<type> d, s;
    d = s = head;
    if (s.Link != null || d == null)
    {
        while (s.Link != null)
        {
            s = d.Link;
            if (d.Data.Equals(Nr))
            {
                d.Data = s.Data;
                d.Link = s.Link;
                s = null;

                break;
            }
            if (s.Data.Equals(Nr) && s.Link == null)
            {
                s = null;
                d.Link = s;

                break;
            }
            d = s;
        }
    }
    else
    {
        if (d != null)
        {
            d = null;
        }
    }
}

/// <summary>
/// Sorts linklist first of all by scholarship size, then by surname and then by
name
/// </summary>
public void Sort()

```

```

{
    if (head == null)
    {
        return;
    }
    bool done = true;
    while (done)
    {
        done = false;
        var headn = head;
        while (headn != null)
        {
            if (headn.Link != null && headn.Data.CompareTo(headn.Link.Data) > 0)
            {
                type St = headn.Data;
                headn.Data = headn.Link.Data;
                headn.Link.Data = St;
                done = true;
            }
            headn = headn.Link;
        }
    }
}

/// <summary>
/// Sets d as the begining of the linklist
/// </summary>
public void Begin()
{
    d = head;
}

/// <summary>
/// Set d as the next node in the linklist
/// </summary>
public void Next()
{
    d = d.Link;
}

/// <summary>
/// Sees if d is not the last node of the linklist
/// </summary>
/// <returns></returns>
public bool Exist()
{
    return d != null;
}

/// <summary>
/// Brings d value out of the linklist
/// </summary>
/// <returns> Object that d is linked to </returns>
public type Get()
{
    return d.Data;
}
}
}

```

//





```

    /// <param name="List1"> linklist with all student class objects </param>
    /// <param name="List2"> linklist with all studentinformation class objects
</param>
    /// <returns></returns>
    private static double CalculateIndex(LinkList<Student> List1,
LinkList<StudentInformation> List2)
    {
        double SumIndex = 0;
        for (List1.Begin(); List1.Exist(); List1.Next())
        {
            Student student = List1.Get();
            SumIndex += student.Pirm;

        }
        return SumIndex;
    }

    /// <summary>
    /// Adds schoralship amount to each student
    /// </summary>
    /// <param name="List1"> linklist with all student class objects </param>
    /// <param name="List2"> linklist with all studentinformation class objects
</param>
    public static void AddFunds(LinkList<Student> List1,
LinkList<StudentInformation> List2)
    {
        double SumIndex = TaskUtils.CalculateIndex(List1, List2);
        double IndexWorth = List2.FundSize / SumIndex;
        for (List1.Begin(); List1.Exist(); List1.Next())
        {
            Student student = List1.Get();
            student.CashGiven = student.Pirm * IndexWorth;
        }
    }

    /// <summary>
    /// Removes students that have no scholarship from both linklists
    /// </summary>
    /// <param name="List1"> linklist with all student class objects </param>
    /// <param name="List2"> linklist with all studentinformation class objects
</param>
    public static void RemoveStudents(LinkList<Student> List1,
LinkList<StudentInformation> List2)
    {
        bool reset = true;
        while(reset)
        {
            reset = false;
            for (List1.Begin(); List1.Exist(); List1.Next())
            {
                Student student = List1.Get();
                if (student.CashGiven == 0)
                {
                    List2.Remove(TaskUtils.FindNumber(student.Nr, List2));
                    List1.Remove(student);
                    reset = true;
                    break;
                }
            }
        }
    }
}

```

```

    /// <summary>
    ///
    /// </summary>
    /// <param name="List1"> linklist with all student class objects </param>
    /// <param name="List2"> linklist with all studentinformation class objects
</param>
    /// <param name="GroupNeeded"> name of the group we want to form a list of
    /// that are all from the same group </param>
    /// <returns> linklist of student class objects that
    /// are all from the same group </returns>
    public static LinkList<Student> FormGroupList(LinkList<Student> List,
LinkList<StudentInformation> Info, string GroupNeeded)
    {
        LinkList<Student> GroupedList = new LinkList<Student>();
        for (Info.Begin(); Info.Exist(); Info.Next())
        {
            StudentInformation info = Info.Get();
            if(info.Group == GroupNeeded)
            {
                Student student = TaskUtils.FindNumber(info.Nr, List);
                GroupedList.Add_d(student);
            }
        }
        return GroupedList;
    }

    public static StudentInformation FindNumber(int Nr, LinkList<StudentInformation>
Info)
    {
        for(Info.Begin(); Info.Exist(); Info.Next())
        {
            if (Nr == Info.Get().Nr)
            {
                return Info.Get();
            }
        }

        return Info.Get();
    }

    public static Student FindNumber(int Nr, LinkList<Student> List)
    {
        for (List.Begin(); List.Exist(); List.Next())
        {
            if (Nr == List.Get().Nr)
            {
                return List.Get();
            }
        }

        return List.Get();
    }
}
}

```

Form.cs:

59

```

for (List.Begin(); List.Exist(); List.Next())
{
    var row = new TableRow();

    Student Student = List.Get();
    row.Cells.Add(new TableCell()
    {
        Text = Student.Nr.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Student.SurName.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Student.Name.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Student.Phone.ToString()
    });

    table.Rows.Add(row);
}

firstRow2.Cells.Add(new TableCell()
{
    Text = "Nr",
    CssClass = "bold-text"
});
firstRow2.Cells.Add(new TableCell()
{
    Text = "Grupė",
    CssClass = "bold-text"
});
firstRow2.Cells.Add(new TableCell()
{
    Text = "Pažymių kiekis",
    CssClass = "bold-text"
});
firstRow2.Cells.Add(new TableCell()
{
    Text = "Pažymiai",
    CssClass = "bold-text"
});

table2.Rows.Add(firstRow2);

for (List2.Begin(); List2.Exist(); List2.Next())
{
    var row = new TableRow();

    StudentInformation Info = List2.Get();
    row.Cells.Add(new TableCell()
    {

```

```

        Text = Info.Nr.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = Info.Group.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = Info.Grades.Count().ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = Info.ToString()
    }
    );

    table2.Rows.Add(row);
}
}

/// <summary>
/// Table creator, that makes a table of mixed student's information
/// </summary>
/// <param name="table"> Table element where the
/// information will be written in </param>
/// <param name="List"> Basic information linklist </param>
/// <param name="List2"> Official information linklist </param>
public static void FillTableWithStudentsData(ref Table table,
    LinkList<Student> List, LinkList<StudentInformation> List2)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Pavardė",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Vardas",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Pažymiai",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Stipendijos dydis",
        CssClass = "bold-text"
    });

    table.Rows.Add(firstRow);
}

```

```

for (List.Begin(); List.Exist(); List.Next())
{
    var row = new TableRow();

    Student Student = List.Get();
    StudentInformation Info = TaskUtils.FindNumber(Student.Nr, List2);

    row.Cells.Add(new TableCell()
    {
        Text = Student.SurName.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Student.Name.ToString()
    });

    row.Cells.Add(new TableCell()
    {
        Text = Info.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = Student.CashGiven.ToString()
    });
    table.Rows.Add(row);
}

}

/// <summary>
/// Table creator, that makes a table of mixed student's information
/// </summary>
/// <param name="table"> Table element where the
/// information will be written in </param>
/// <param name="List"> Basic information linklist </param>
/// <param name="List2"> Official information linklist </param>
public static void FillTableWithForerunnerData(ref Table table,
    LinkList<Student> List, LinkList<StudentInformation> List2)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Pavardè",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Vardas",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()

```



```

/// <summary>
/// Reads basic student information
/// </summary>
/// <param name="FileName"> The name of the file, where
/// the information is written in </param>
/// <returns> A LinkedList, that contains basic student information </returns>
public static LinkedList<Student> ReadFile1(string FileName)
{
    try
    {
        string[] Lines = File.ReadAllLines(FileName, Encoding.ASCII);
        LinkedList<Student> List = new LinkedList<Student>();
        for (int i = 0; i < Lines.Count(); i++)
        {
            string[] Words = Lines[i].Split(' ');
            int Nr = int.Parse(Words[0]);
            string SurName = Words[1];
            string Name = Words[2];
            int Phone = int.Parse(Words[3]);
            List.Add_d(new Student(Nr, SurName, Name, Phone));
        }
        return List;
    }
    catch
    {
        return null;
    }
}

/// <summary>
/// Reads official student information
/// </summary>
/// <param name="FileName"> The name of the file, where
/// the information is written in </param>
/// <returns> A LinkedList, that contains student information </returns>
public static LinkedList<StudentInformation> ReadFile2(string FileName)
{
    try
    {
        string[] Lines = File.ReadAllLines(FileName, Encoding.ASCII);
        LinkedList<StudentInformation> List = new LinkedList<StudentInformation>();
        List.AddFundSizeAndVid(int.Parse(Lines[0].Split(' ')[0]),
            int.Parse(Lines[0].Split(' ')[1]));
        for (int i = 1; i < Lines.Count(); i++)
        {
            List<int> Grades = new List<int>();
            string[] Words = Lines[i].Split(' ');
            int Nr = int.Parse(Words[0]);
            string Group = Words[1];
            int Count = int.Parse(Words[2]);
            for (int l = 3; l < Count + 3; l++)
            {
                Grades.Add(int.Parse(Words[l]));
            }
            List.Add_d(new StudentInformation(Nr, Group, Count, Grades));
        }
        return List;
    }
    catch
    {
        return null;
    }
}

```



```

/// <summary>
/// Creates a table of basic student information (Number, Surname, Name, Phone)
/// </summary>
/// <param name="Filename"> File where the tabel will be created </param>
/// <param name="List"> Linked list of basic information
/// which the table will be made of </param>
/// <param name="Header"> A string above the table </param>
public static void WriteData(string Filename, LinkList<Student> List, string
Header)
{
    File.AppendAllText(Filename, String.Format(Header + "\n"));
    File.AppendAllText(Filename, String.Format(new string('-', 59) + "\n"));
    File.AppendAllText(Filename, String.Format("| {0, -8} |" +
        " {1, -12} | {2, -14} | {3, -12} |\n", "Numeris",
        "Pavardė", "Vardas", "Telefonas"));
    File.AppendAllText(Filename, String.Format(new string('-', 59) + "\n"));
    for(List.Begin(); List.Exist(); List.Next())
    {
        File.AppendAllText(Filename, String.Format(List.Get().ToString()));
    }
    File.AppendAllText(Filename, String.Format(new string('-', 59) + "\n" +
"\n"));
}

/// <summary>
/// Creates a table of official student information (Number,
/// Group, How many grades you have, All the grades)
/// </summary>
/// <param name="Filename"> File where the tabel will be created </param>
/// <param name="List"> Linked list of official information
/// which the table will be made of </param>
/// <param name="Header"> A string above the table </param>
public static void WriteInfo(string Filename, LinkList<StudentInformation>
List, string Header)
{
    File.AppendAllText(Filename, String.Format(Header + "\n"));
    File.AppendAllText(Filename, String.Format(new string('-', 67) + "\n"));
    File.AppendAllText(Filename, String.Format("| {0, -8} |" +
        " {1, -12} | {2, -14} | {3, -20} |\n", "Numeris",
        "Grupė", "Pažymių kiekis", "Pažymiai"));
    File.AppendAllText(Filename, String.Format(new string('-', 67) + "\n"));
    for (List.Begin(); List.Exist(); List.Next())
    {
        StudentInformation Info = List.Get();
        File.AppendAllText(Filename, String.Format("| {0, -8} " +
            " | {1, -12} | {2, -14} | {3, -20} |\n", Info.Nr,
            Info.Group, Info.Count, Info.ToString()));
    }
    File.AppendAllText(Filename, String.Format(new string('-', 67) + "\n" +
"\n"));
}

/// <summary>
/// Creates a table of students (Number, Surname, Name, Grades, And their
scholarship)
/// </summary>
/// <param name="Filename"> File where the table will be created in </param>
/// <param name="List"> Basic student information containing linklist </param>
/// <param name="Info"> Official student information containing linklist
</param>
/// <param name="Header"> A string above the table </param>
public static void WriteAllStudentsBasicInfo(string Filename,
LinkList<Student> List, LinkList<StudentInformation> Info, string Header)
{

```

```

File.AppendAllText(Filename, String.Format(Header + "\n"));
File.AppendAllText(Filename, String.Format(new string('-', 81) + "\n"));
File.AppendAllText(Filename, String.Format("| {0, -8} | {1, -12}" +
    " | {2, -14} | {3, -20} | {4, -12} |\n", "Numeris",
    "Paverdė", "Vardas", "Pažymiai", "Stipendija"));
File.AppendAllText(Filename, String.Format(new string('-', 81) + "\n"));
for (List.Begin(); List.Exist(); List.Next())
{
    Student student = List.Get();
    File.AppendAllText(Filename, String.Format("| {0, -8} |" +
        " {1, -12} | {2, -14} | {3, -20} | {4, -12} |\n",
        student.Nr, student.SurName, student.Name,
        TaskUtils.FindNumber(student.Nr, Info).ToString(),
student.CashGiven));
}
File.AppendAllText(Filename, String.Format(new string('-', 81) + "\n" +
"\n"));
}

/// <summary>
/// Creates a table of students who are fruntrunners in their group (Surname,
Name, Grades)
/// </summary>
/// <param name="Filename"> File where the table will be created in </param>
/// <param name="List"> Basic student information containing linklist </param>
/// <param name="Info"> Official student information containing linklist
</param>
/// <param name="Header"> A string above the table </param>
public static void WriteFrontRunners(string Filename, LinkList<Student> List,
LinkList<StudentInformation> Info, string Header)
{
    {
        File.AppendAllText(Filename, String.Format(Header + "\n"));
        File.AppendAllText(Filename, String.Format(new string('-', 56) +
"\n"));

        File.AppendAllText(Filename, String.Format("| {0, -12} |" +
            " {1, -14} | {2, -20} |\n", "Paverdė", "Vardas", "Pažymiai"));
        File.AppendAllText(Filename, String.Format(new string('-', 56) +
"\n"));

        for (List.Begin(); List.Exist(); List.Next())
        {
            Student student = List.Get();
            File.AppendAllText(Filename, String.Format("| {0, -12} |" +
                " {1, -14} | {2, -20} |\n", student.SurName, student.Name,
                TaskUtils.FindNumber(student.Nr, Info).ToString()));
        }
        File.AppendAllText(Filename, String.Format(new string('-', 56) + "\n" +
"\n"));
    }
}

}

}

////////////////////
//
////////////////////

```

Web.aspx.cs:

```
using System;
```

```

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

namespace WebApplication3
{

    public partial class Web : System.Web.UI.Page
    {

        private string DataFile1, DataFile2, ResultFile;
        private LinkList<Student> List;
        private LinkList<StudentInformation> Info;

        protected void Page_Load(object sender, EventArgs e)
        {

            DataFile1 = Server.MapPath("App_Data/U14a.txt");
            DataFile2 = Server.MapPath("App_Data/U14b.txt");
            ResultFile = Server.MapPath("App_Data/Results.txt");
            Label4.Visible = false;

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            if (FileUpload2.HasFile && FileUpload1.HasFile &&
                Path.GetExtension(FileUpload1.FileName) == ".txt" &&
                Path.GetExtension(FileUpload2.FileName) == ".txt")
            {

                if (File.Exists(Server.MapPath("App_Data/Results.txt")))
                {
                    File.Delete(Server.MapPath("App_Data/Results.txt"));
                    ResultFile = Server.MapPath("App_Data/Results.txt");
                }

                FileUpload1.SaveAs(DataFile1);
                FileUpload2.SaveAs(DataFile2);

                if (new FileInfo(DataFile1).Length == 0 &&
                    new FileInfo(DataFile2).Length == 0 &&
                    InOutUtils.ReadFile1(DataFile1) == null &&
                    InOutUtils.ReadFile2(DataFile2) == null
                )
                {
                    Label4.Text = "* Failai/Failas, kurį pasirinkote" +
                        " turėjo neteisingą informacijos išdėstymą";
                    Label4.Visible = true;
                }
                else
                {

                    List = InOutUtils.ReadFile1(DataFile1);
                    Info = InOutUtils.ReadFile2(DataFile2);
                }
            }
        }
    }
}

```

```

        Table1.Visible = true;
        Table2.Visible = true;

        Form1.FillTableWithAllData(ref Table1, ref Table2, List, Info);

        InOutUtils.WriteData(ResultFile, List, "Pradiniai studentų
duomenys:");
        InOutUtils.WriteInfo(ResultFile, Info, "Pradiniai studentų
duomenys:");

        Button2.Visible = true;
        Button1.Visible = false;
        FileUpload1.Visible = false;
        FileUpload2.Visible = false;
        Table1.Visible = true;
        Table2.Visible = true;
        Label2.Text = "Studentų asmeniniai duomenys";
        Label3.Text = "Studentų tarnybinė informacija";
        Label2.Visible = true;
        Label3.Visible = true;
        Label5.Visible = false;
        Label6.Visible = false;
    }

}
else
{
    Label4.Visible = true;
    Label4.Text = "* Failai/Failas, kurį pasirinkote buvo" +
        " neteisingo formato !!! Bandykite dar kartą.";
}
}

```

```

protected void Button2_Click(object sender, EventArgs e)
{
    List = InOutUtils.ReadFile1(DataFile1);
    Info = InOutUtils.ReadFile2(DataFile2);

    TaskUtils.Distribution(List, Info);
    TaskUtils.AddFunds(List, Info);
    List.Sort();

    Form1.FillTableWithStudentsData(ref Table1, List, Info);
    InOutUtils.WriteAllStudentsBasicInfo(ResultFile, List, Info,
        "Studentai po stienijos skaičiavimo:");

    TaskUtils.RemoveStudents(List, Info);
    Form1.FillTableWithStudentsData(ref Table2, List, Info);
    InOutUtils.WriteAllStudentsBasicInfo(ResultFile, List, Info,
        "Studentai tik su stipendijomis:");

    Table1.Visible = true;
    Table2.Visible = true;
    Label2.Text = "Studentų sąrašas";
    Label3.Text = "Sąrašas be studentų, kurie neturi stipendijos";
    TextBox1.Visible = true;
    Button2.Visible = false;
    Button3.Visible = true;
}

protected void Button3_Click(object sender, EventArgs e)
{
    Label2.Visible = false;
}

```

```

Label3.Visible = false;
if (TextBox1.Text.Count() == 1)
{
    Table2.Visible = true;
    Label2.Visible = true;

    List = InOutUtils.ReadFile1(DataFile1);
    Info = InOutUtils.ReadFile2(DataFile2);

    //How to fix this so List and Info would save
    TaskUtils.Distribution(List, Info);
    TaskUtils.AddFunds(List, Info);
    List.Sort();
    TaskUtils.RemoveStudents(List, Info);

    LinkList<Student> ListByGroup = TaskUtils.FormGroupList(List, Info,
TextBox1.Text);
    Form1.FillTableWithForerunnerData(ref Table1, ListByGroup, Info);
    Label2.Text = "Grupės " + TextBox1.Text + " pirmūnai";
    InOutUtils.WriteFrontRunners(ResultFile,
        ListByGroup, Info, "Group's " + TextBox1.Text + " pirmūnai: ");

    TextBox1.Text = "";
    Button4.Visible = true;
}
else
{
    Label4.Text = "* Parašėte neteisingą grupės formata" +
        " !!! (Grupės sudarytos iš vienos raidės)";
    Label4.Visible = true;
}
}

protected void Button4_Click(object sender, EventArgs e)
{
    FileUpload1.Visible = true;
    FileUpload2.Visible = true;
    Button1.Visible = true;
    Label2.Visible = false;
    TextBox1.Visible = false;
    Table1.Visible = false;
    Button3.Visible = false;
    Button4.Visible = false;
    Label5.Visible = true;
    Label6.Visible = true;
}
}
}

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

PageStyle.css:

```

.auto-style1 {
    width: 100%;
    padding: 10px;
    background-color: burlywood;
    border-style: solid;
    border-color: black;
}

```

```

}

body {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: antiquewhite;
}

.auto-style2 {
    height: 29px;
}

.auto-style3 {
    height: 23px;
}

.Tableless {
    background-color: #FF9933;
    border-color: Black;
    border-style: Solid;
    border-width: 5px;
}

.auto-style5 {
    height: 22px;
    justify-content: center;
    align-items: center;
}

```

### 3.7. Pradiniai duomenys ir rezultatai

#### Testas Nr. 1:

**Tikslas:** Standartinis atvejis, kai apskaičiuojami keli atsakymai.

#### U14a.txt

```

12 Redis Jokubas 86989745
79 Regis Petris 86784588
78 Delgis Romis 89777845
41 Kukutis Jonas 86784558
79 Remis Dodukas 88888888
120 Dalgis Katinas 121321333
1 Redis Astolinis 877874455

```

#### U14b.txt

```

800 6
41 B 4 9 8 9 9
12 C 1 2
79 B 4 9 9 8 9
78 B 4 5 7 5 9
1 Z 3 5 5 3
79 C 5 9 9 9 9 8

```

## Results.txt

Pradiniai studentų duomenys:

Numeris	Pavardė	Vardas	Telefonas
12	Redis	Jokubas	86989745
79	Regis	Petris	86784588
78	Delgis	Romis	89777845
41	Kukutis	Jonas	86784558
79	Remis	Dodukas	88888888
120	Dalgis	Katinas	121321333
1	Redis	Astolinis	877874455

Pradiniai studentų duomenys:

Numeris	Grupė	Pažymių kiekis	Pažymiai
41	B	4	9, 8, 9, 9
12	C	1	2
79	B	4	9, 9, 8, 9
78	B	4	5, 7, 5, 9
1	Z	3	5, 5, 3
79	C	5	9, 9, 9, 9, 8
120	C	4	4, 10, 9, 9

Studentai po stiendijos skaičiavimo:

Numeris	Pavardė	Vardas	Pažymiai	Stipendija
41	Kukutis	Jonas	9, 8, 9, 9	
204,651162790698				
79	Regis	Petris	9, 9, 8, 9	
204,651162790698				
79	Remis	Dodukas	9, 9, 8, 9	
204,651162790698				
78	Delgis	Romis	5, 7, 5, 9	
186,046511627907				
120	Dalgis	Katinas	4, 10, 9, 9	0
1	Redis	Astolinis	5, 5, 3	0
12	Redis	Jokubas	2	0

Studentai tik su stipendijomis:

-

Numeris	Paverdė	Vardas	Pažymiai	Stipendija
41	Kukutis	Jonas	9, 8, 9, 9	
204,651162790698	79	Regis	9, 9, 8, 9	
204,651162790698	79	Remis	9, 9, 8, 9	
204,651162790698	78	Delgis	5, 7, 5, 9	
186,046511627907				

Group's B pirmūnai:

Paverdė	Vardas	Pažymiai
Kukutis	Jonas	9, 8, 9, 9
Regis	Petris	9, 9, 8, 9
Delgis	Romis	5, 7, 5, 9

Group's C pirmūnai:

Paverdė	Vardas	Pažymiai
Regis	Petris	9, 9, 8, 9

Group's Z pirmūnai:

Paverdė	Vardas	Pažymiai
---------	--------	----------

## User interface

# Studentų informacijos tvarkyklė

Pasirinkite studentų asmeninių duomenų failą
Pasirinkite studentų tarnybinių duomenų failą

Choose File No file chosen
Choose File No file chosen

Load Data



## Studentų informacijos tvarkyklė

Studentų asmeniniai duomenys

Nr	Pavardė	Vardas	Telefono numeris
12	Redis	Jokubas	86989745
79	Regis	Petris	86784588
78	Delgis	Romis	89777845
41	Kukutis	Jonas	86784558
79	Remis	Dodukas	88888888
120	Dalgis	Katinas	121321333
1	Redis	Astolinis	877874455

Studentų tarnybinė informacija

Nr	Grupė	Pažymių kiekis	Pažymiai
41	B	4	9, 8, 9, 9
12	C	1	2
79	B	4	9, 9, 8, 9
78	B	4	5, 7, 5, 9
1	Z	3	5, 5, 3
79	C	5	9, 9, 9, 9, 8
120	C	4	4, 10, 9, 9

Calculate scholarships

## Studentų informacijos tvarkyklė

Studentų sąrašas

Pavardė	Vardas	Pažymiai	Stipendijos dydis
Kukutis	Jonas	9, 8, 9, 9	204,651162790698
Regis	Petris	9, 9, 8, 9	204,651162790698
Remis	Dodukas	9, 9, 8, 9	204,651162790698
Delgis	Romis	5, 7, 5, 9	186,046511627907
Dalgis	Katinas	4, 10, 9, 9	0
Redis	Astolinis	5, 5, 3	0
Redis	Jokubas	2	0

Sąrašas be studentų, kurie neturi stipendijos

Pavardė	Vardas	Pažymiai	Stipendijos dydis
Kukutis	Jonas	9, 8, 9, 9	204,651162790698
Regis	Petris	9, 9, 8, 9	204,651162790698
Remis	Dodukas	9, 9, 8, 9	204,651162790698
Delgis	Romis	5, 7, 5, 9	186,046511627907

Make List By Group

Testas Nr. 2:

**Tikslas:** Parodyti programos veikimą, kai failuose nėra duomenų.

U14a.txt

U14b.txt

Results.txt (Rezultatų failas nebuvo sukurtas)

## Studentų informacijos tvarkyklė

Pasirinkite studentų asmeninių duomenų failą

Choose File

No file chosen

Pasirinkite studentų tarnybinių duomenų failą

Choose File

No file chosen

\* Failai/Failas, kurį pasirinkote buvo neteisingo formato !!! Bandykite dar kartą.

Load Data

### 3.8. Dėstytojo pastabos



Vacius Jusas - Kt, 28 bal. 2022, 10:48

Dėstytojo pastabos tuščios praėjusiame darbe.



Vacius Jusas - Kt, 28 bal. 2022, 10:49

public List Grades; - niekada



Vacius Jusas - Kt, 28 bal. 2022, 10:50

/// Adds a class object using a backwards linklist formation

/// /// Object we want to add public void Add\_d(type k

Netiesa komentaruose.



Vacius Jusas - Kt, 28 bal. 2022, 10:51

Foreach nepanaudotas.



Vacius Jusas - Kt, 28 bal. 2022, 10:52

Vienetų testų nėra.

## 4. Polimorfizmas ir išimčių valdymas (L4)

### 4.1. Darbo užduotis

**U4\_14. Automobilių parkas.** Įmonė UAB „Strėlė“ plečiasi ir atidarė naujus filialus. Pirmoje eilutėje yra miestas, antroje – adresas, trečioje – telefonas. Įmonės automobilių parką sudaro lengvieji automobiliai, krovininiai automobiliai ir autobusai. Sukurkite abstrakčiąją klasę „Transport“ (savybės – valstybinis numeris, gamintojas, modelis, pagaminimo metai ir mėnuo, atliktos techninės apžiūros data, kuras, vidutinės kuro sąnaudos (100 km)), kurią paveldės klasės „Car“ (savybė – odometro rodmenys), „Truck“ (savybė – priekabos talpa) ir „Bus“ (savybė – sėdimų vietų skaičius).

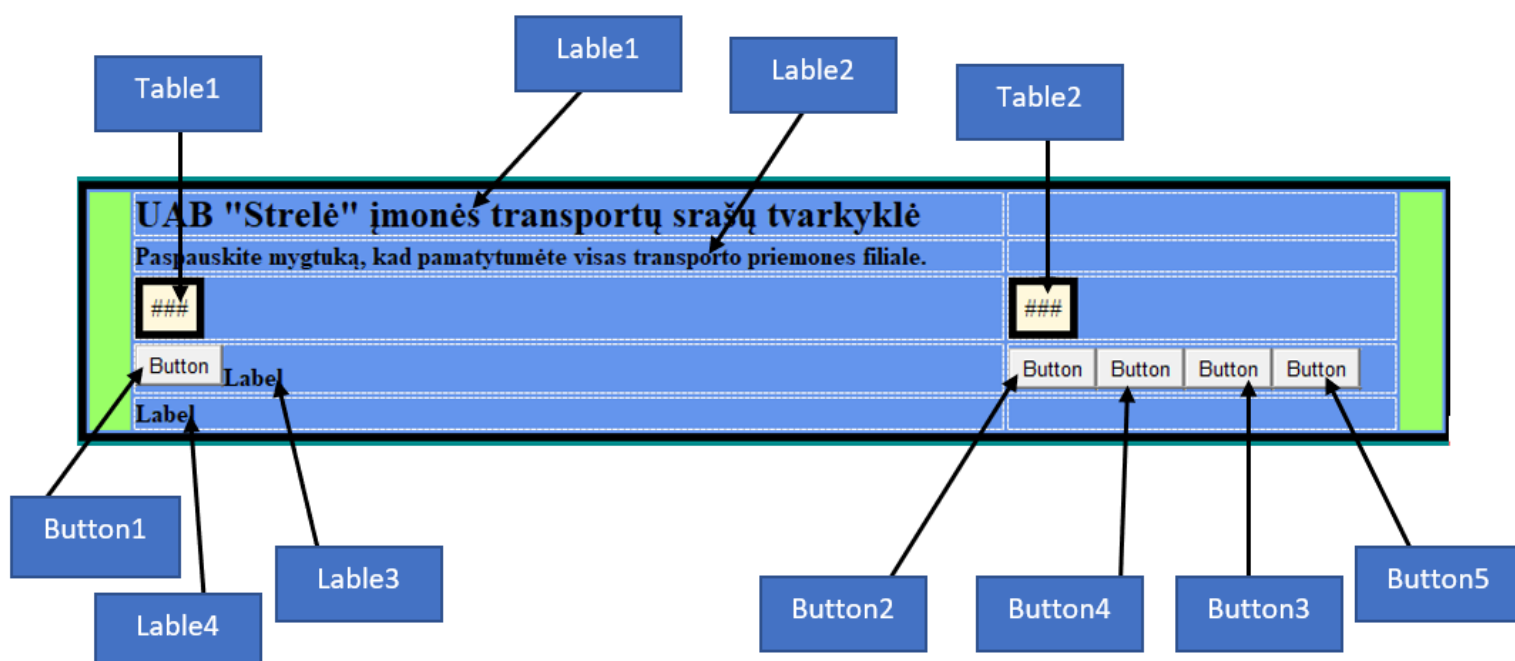
- Raskite geriausią transporto priemonę kiekvienoje grupėje, atspausdinkite jos gamintoją, modelį, valstybinius numerius ir amžių. Lengvasis geriausias – nuvažiuota mažiausiai kilometrų, krovininis geriausias – didžiausia priekabos talpa, autobusas geriausias – daugiausia sėdimų vietų.

- Sudarykite visų benzinu varomų lengvųjų automobilių sąrašą, ekrane atspausdinkite jų valstybinį numerį, gamintoją, modelį, bei pagaminimo metus.

- Sudarykite kiekvieno filialo autobusų sąrašą, surikiuokite pagal pagaminimo metus ir gamintoją.

- Nustatykite kiekvienai transporto priemonei artimiausios techninės apžiūros datą, jei lengvajam automobiliui techninė apžiūra galioja 2 metus, kroviniam automobiliui – metus, o autobusui – pusę metų. Į failą „Apžiūra.txt“ įrašykite tų transporto priemonių duomenis (transporto priemonių gamintojus, modelius, valstybinius numerius, techninės apžiūros galiojimo pabaigą), kurioms iki techninės apžiūros galiojimo pabaigos liko mažiau kaip 2 mėnesiai

### 4.2. Grafinės vartotojo sąsajos schema



#### 4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Lable1	Text	„UAB "Strelė" įmonės transportų srašų tvarkyklė“
Lable2	Text	„Paspauskite mygtuką, kad pamatytumėte visas transporto priemones filiale.“ + "Geriausios transporto priemonės:" + "Visos mašinos, kurios naudoja benziną:" + "Visų filialų autobusų sąrašai:" + "Transporto priemonės, kurioms liko 2 mėn. arba mažiau iki techninio:" + "Visos transporto priemonės:"
Lable3	Text	"Paspaudę mygtuką dar kartą, pamatysite geriausiais transporto priemones." + "Paspaudę mygtuką dar kartą, pamatysite visas mašinas, kurios naudoja benziną" + "Paspaudę mygtuką dar kartą, pamatysite visų filialų autobusų sąrašus" + "Paspaudę mygtuką dar kartą, pamatysite transporto priemonių sąrašą" + "Paspaudę mygtuką grįšite į pradžią"
Lable4	Text	"kurioms iki techninio liko mažiau nei 2 mėn"
Button1	Text	„Button“
Button1	OnClick	Į lentelę sudeda visuose failuose esančias transporto priemones.
Button2	Text	„Button“
Button2	OnClick	Paspaudę mygtuką, pamatysite geriausiais transporto priemones
Button3	Text	„Button“
Button3	OnClick	Paspaudę mygtuką, pamatysite visas mašinas, kurions naudoja benziną
Button4	Text	„Button“
Button4	OnClick	Paspaudę mygtuką pamatysite visų filialų autobusų sąrašus.
Button5	Text	„Button“
Button5	OnClick	Paspaude mygtuką pamatysite, Transporto priemones, kurioms liko 2 mėn. arba mažiau iki techninio
Table1	GridLines	
Table2	GridLines	

## 4.4. Klasių diagrama

```
TaskOut.cs
public static List<VRegister> ReadFile(string Path, string Rez_File)
public static void PrintDataVRegister(string FileName, List<VRegister> Container, string Header)
public static void PrintBestTransport(string FileName, TransportContainer Container, string Header)
public static void PrintCarsWithGas(string FileName, TransportContainer Container, string Header)
public static void PrintReviewDates(string FileName, TransportContainer Container, string Header)
```

```
Truck.cs
public class Truck : Transport
{
    public double TrailerSize { get; set; }
}

public Truck(string Nr, string Brand, string Model,
    DateTime Data, DateTime Review, string Gas, double AverageGas,
    double TrailerSize) : base(Nr, Brand, Model, Data, Review, Gas, AverageGas)
{
}

public override string ToString()
{
    return base.ToString() + "TrailerSize: " + TrailerSize;
}

public override bool Equals(Transport other)
{
    return base.Equals(other) && TrailerSize == other.TrailerSize;
}

public override int CompareTo(Transport other)
{
    return base.CompareTo(other);
}

public override void AddReviewDate()
{
    base.AddReviewDate();
}

public static bool operator >(Truck a, Truck b)
{
    return a.TrailerSize > b.TrailerSize;
}

public static bool operator <(Truck a, Truck b)
{
    return a.TrailerSize < b.TrailerSize;
}


```

```
Bus.cs
class Bus : Transport
{
    public int SeatNumber { get; set; }
}

public Bus(string Nr, string Brand, string Model,
    DateTime Data, DateTime Review, string Gas, double AverageGas,
    int SeatNumber) : base(Nr, Brand, Model, Data, Review, Gas, AverageGas)
{
}

public override string ToString()
{
    return base.ToString() + "SeatNumber: " + SeatNumber;
}

public override bool Equals(Transport other)
{
    return base.Equals(other) && SeatNumber == other.SeatNumber;
}

public override int CompareTo(Transport other)
{
    return base.CompareTo(other);
}

public static bool operator >(Bus a, Bus b)
{
    return a.SeatNumber > b.SeatNumber;
}

public static bool operator <(Bus a, Bus b)
{
    return a.SeatNumber < b.SeatNumber;
}

public override void AddReviewDate()
{
    base.AddReviewDate();
}


```

```
Transport.cs
public abstract class Transport : IEquatable<Transport>, IComparable<Transport>
{
    public string Nr { get; set; }
    public string Brand { get; set; }
    public string Model { get; set; }
    public DateTime Data { get; set; }
    public DateTime Review { get; set; }
    public string Gas { get; set; }
    public double AverageGasUsage { get; set; }
    public DateTime Validity { get; set; }
}

public Transport(string Nr, string Brand, string Model,
    DateTime Data, DateTime Review, string Gas, double AverageGas)
{
    Nr = Nr; Brand = Brand; Model = Model; Data = Data; Review = Review; Gas = Gas; AverageGas = AverageGas;
}

public override string ToString()
{
    return "Nr: " + Nr + " Brand: " + Brand + " Model: " + Model + " Data: " + Data + " Review: " + Review + " Gas: " + Gas + " AverageGas: " + AverageGas;
}

public virtual int CompareTo(Transport other)
{
    return Nr.CompareTo(other.Nr);
}

public virtual bool Equals(Transport other)
{
    return Nr == other.Nr;
}


```

```
Car.cs
class Car : Transport
{
    public double Odometre { get; set; }
}

public Car(string Nr, string Brand, string Model,
    DateTime Data, DateTime Review, string Gas, double AverageGas,
    double Odometre) : base(Nr, Brand, Model, Data, Review, Gas, AverageGas)
{
}

public override string ToString()
{
    return base.ToString() + "Odometre: " + Odometre;
}

public override bool Equals(Transport other)
{
    return base.Equals(other) && Odometre == other.Odometre;
}

public override int CompareTo(Transport other)
{
    return base.CompareTo(other);
}

public static bool operator >(Car a, Car b)
{
    return a.Odometre > b.Odometre;
}

public static bool operator <(Car a, Car b)
{
    return a.Odometre < b.Odometre;
}

public override void AddReviewDate()
{
    base.AddReviewDate();
}


```

```
TaskUtils.cs / static class TaskUtils
{
    public static TransportContainer CarsWithGas(List<VRegister> ContainerReg)
    {
        return new TransportContainer(CarsWithGas(ContainerReg));
    }

    public static TransportContainer AddReviewTime(List<VRegister> Container)
    {
        return new TransportContainer(AddReviewTime(Container));
    }

    public static Transport BestBus(List<VRegister> Container)
    {
        return new Transport(BestBus(Container));
    }

    private static Transport BusFinder(List<VRegister> Container)
    {
        return new Transport(BusFinder(Container));
    }

    public static Transport CarFinder(List<VRegister> Container)
    {
        return new Transport(CarFinder(Container));
    }

    public static Transport BestCar(List Container)
    {
        return new Transport(BestCar(Container));
    }

    private static Transport TruckFinder(List<VRegister> Container)
    {
        return new Transport(TruckFinder(Container));
    }

    public static Transport BestTruck(List<VRegister> Container)
    {
        return new Transport(BestTruck(Container));
    }
}


```

```
VRegister.cs
private TransportContainer Container;
public string City { get; set; }
public string Address { get; set; }
public string Phone { get; set; }

public VRegister(TransportContainer Container, string City, string Address, string Phone)
{
    Container = Container; City = City; Address = Address; Phone = Phone;
}

public Transport Get(int index)
{
    return Container.Get(index);
}

public void Add(Transport vroom)
{
    Container.Add(vroom);
}

public int Count()
{
    return Container.Count;
}


```

```
TransportContainer.cs
private Transport[] Alltransports;
private int Capacity;

public int Count { get; private set; }

public TransportContainer(int Capacity = 16)
{
    Capacity = Capacity;
}

private void EnsureCapacity(int min)
{
    if (min > Capacity)
    {
        Capacity = min;
    }
}

public void Add(Transport transport)
{
    EnsureCapacity(Count + 1);
    Alltransports[Count] = transport;
    Count++;
}

public Transport Get(int index)
{
    return Alltransports[index];
}

public void Sort()
{
    Alltransports.Sort();
}


```

## 4.5. Programos vartotojo vadovas

### Failo struktūra:

Vienoje eilutėje einantys dydžiai, skiriami kableliais.

Faile aprašomas filialas, su jame esančiomis transporto priemonėmis (Priemonės bus galima suskirstyti į 3 grupes, Transporto priemonės, kurių Valstybiniai numeriai prasideda raide „A“ yra sunkvežimiai, prasidedantys raide „B“ yra lengvosios mašinos, prasidedantys „C“ yra autobusai.)

- Pirmoje eilutėje yra miestas
- Antroje – adresas
- Trečioje – telefonas

### Tolimesnių eilučių struktūra :

- Valstybinis numeris
- Gamintojas
- Modelis
- Pagaminimo metai ir mėnuo (atskiriami „/“)
- Atliktos techninės apžiūros data (atskirama „/“)
- Kuras
- Vidutinės kuro sąnaudos (100 km)

### Programos veikimas:

1. Paleidžiama programa. Kairiame kampe paspaudus mygtuką pavadinimu „Button“ ant ekrano atsiras iš visų filialų nuskaitytų transporto priemonių duomenys.
2. Spaudžiant naujai atsiradusį mygtuką „Button“ Ektrane bus parodomos geriausios transporto priemonės savo grupėje. Lengvasis geriausias – nuvažiuota mažiausiai kilometrų, krovininis geriausias – didžiausia priekabos talpa, autobusas geriausias – daugiausia sėdimų vietų.
3. Spaudžiant mygtuką dar kartą, bus sudarytas ir atspausdintas sąrašas su visomis lengvosiomis mašinomis, kurios naudoja benzina, kaip kurą.
4. Spaudžiant dar kartą, bus sudarytas kiekvieno filialo autobusų sąrašas ir atspausdintas į ekraną.
5. Paspaudus dar kartą, bus sudarytas sąrašas automobilių, kuriems greičiau nei per 2 mėnesius reikės atlikti technikinę peržiūrą.
6. Paspaudus mygtuką dar kartą, programa pradės skaičiavimus iš naujo.

**Visi atsakymai ir sąrašai bus atspausdinti į „Results.txt“ failą, taip pat sąrašas automobilių, kuriems greičiau nei per 2 mėnesius reikės atlikti technikinę peržiūrą, bus atspausdintas į failą „Apžiūra.txt“.**

## 4.6. Programos tekstas

Transport.cs:

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

namespace WebApplication4
{
    /// <summary>
    /// A parent class of any transpot, containing base information about a transport
    /// </summary>
    public abstract class Transport : IEquatable<Transport>, IComparable<Transport>
    {
        public string Nr { get; set; }
        public string Brand { get; set; }
        public string Model { get; set; }
        public DateTime Data { get; set; }
        public DateTime Review_Data { get; set; }
        public string Gas { get; set; }
        public double AverageGasUsage { get; set; }
    }
}
```

```

public DateTime Validity { get; set; }

/// <summary>
/// Basic Transport class constructor
/// </summary>
/// <param name="Nr"> Transport's Number plate </param>
/// <param name="Brand"> Transport's brand </param>
/// <param name="Model"> Model of our transport </param>
/// <param name="Data"> Purchase date </param>
/// <param name="Review"> Last review appointment date </param>
/// <param name="Gas"> What type of fuel the transport is using </param>
/// <param name="AverageGas"> Average fuel usage </param>
public Transport(string Nr, string Brand, string Model,
    DateTime Data, DateTime Review, string Gas, double AverageGas)
{
    this.Nr = Nr;
    this.Brand = Brand;
    this.Model = Model;
    this.Data = Data;
    this.Review_Data = Review;
    this.Gas = Gas;
    this.AverageGasUsage = AverageGas;
}

/// <summary>
/// Base transport's object ToString() override
/// </summary>
/// <returns> Returns a table row of (Number plate, Transport's brand, Model of
our transport,
/// Purchase date , Last review appointment date, What type of fuel the transport
is using,
/// Average fuel usage </returns>
public override string ToString()
{
    return String.Format("| {0, -8} | {1, -12} | {2, -12} | {3, -18} | {4, -18} |
{5, -10} | {6, -25} |", Nr,
        Brand, Model, Data.ToShortDateString(), Review_Data.ToShortDateString(),
Gas, AverageGasUsage);
}

/// <summary>
/// Compares 2 transport object to see if they are the same
/// </summary>
/// <param name="other"> Another transport object </param>
/// <returns> Return 0 if the objects are the same </returns>
public virtual int CompareTo(Transport other)
{
    return this.CompareTo(other);
}

/// <summary>
/// Compares 2 transport object to see if they are the same
/// </summary>
/// <param name="other"> Another transport object </param>
/// <returns> Returns true if the objects are the same, false if they are not
</returns>
public virtual bool Equals(Transport other)
{
    return this.Equals(other);
}

public abstract void AddReviewDate();
}
}

```

```
/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////
```

## Truck.cs:

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

namespace WebApplication4
{
    /// <summary>
    /// Transport's child class called Truck
    /// </summary>
    class Truck : Transport
    {
        public double TrailerSize { get; set; }

        /// <summary>
        /// Basic Truck class constructor
        /// </summary>
        /// <param name="Nr"> Transport's Number plate </param>
        /// <param name="Brand"> Transport's brand </param>
        /// <param name="Model"> Model of our transport </param>
        /// <param name="Data"> Purchase date </param>
        /// <param name="Review"> Last review appointment date </param>
        /// <param name="Gas"> What type of fuel the transport is using </param>
        /// <param name="AverageGas"> Average fuel usage </param>
        /// <param name="TrailerSize"> Trailers size </param>
        public Truck(string Nr, string Brand, string Model,
            DateTime Data, DateTime Review, string Gas, double AverageGas,
            double TrailerSize) : base(Nr, Brand, Model, Data, Review, Gas, AverageGas)
        {
            this.TrailerSize = TrailerSize;
        }

        /// <summary>
        /// ToString() override
        /// </summary>
        /// <returns> Returns a table row of base class ToString method +
        /// another parameter called trailer size as a string</returns>
        public override string ToString()
        {
            return base.ToString() + String.Format(" {0, -18} |", TrailerSize);
        }

        /// <summary>
        /// Compares 2 transport object to see if they are the same
        /// </summary>
        /// <param name="other"> Another transport object </param>
        /// <returns> Returns true if the objects are the same, false if they are not
        </returns>
        public override bool Equals(Transport other)
        {
            return this.Equals(other);
        }

        /// <summary>
        /// Compares 2 transport object to see if they are the same
        /// </summary>
        /// <param name="other"> Another transport object </param>
        /// <returns> Return 0 if the objects are the same </returns>
    }
}
```



```

public override int CompareTo(Transport other)
{
    return base.CompareTo(other);
}

/// <summary>
/// Adds the date of the next review appointment
/// </summary>
public override void AddReviewDate()
{
    Validity = Review_Data.AddYears(1);
}

/// <summary>
/// Compares who's trailers size is bigger
/// </summary>
/// <param name="a"> Truck object a </param>
/// <param name="b"> Truck object b </param>
/// <returns> true if object a has a bigger trailers size then object b </returns>
public static bool operator >(Truck a, Truck b)
{
    return a.TrailerSize > b.TrailerSize;
}

/// <summary>
/// Compares who's trailers size is bigger
/// </summary>
/// <param name="a"> Truck object a </param>
/// <param name="b"> Truck object b </param>
/// <returns> true if object b has a bigger trailers size then object a </returns>
public static bool operator <(Truck a, Truck b)
{
    return a.TrailerSize < b.TrailerSize;
}
}
}

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

### Bus.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebApplication4
{
    /// <summary>
    /// Transport's child class called Bus
    /// </summary>
    class Bus : Transport
    {
        public int SeatNumber { get; set; }

        /// <summary>
        /// Basic Bus class constructor
        /// </summary>
        /// <param name="Nr"> Transport's Number plate </param>

```

```

/// <param name="Brand"> Transport's brand </param>
/// <param name="Model"> Model of our transport </param>
/// <param name="Data"> Purchase date </param>
/// <param name="Review"> Last review appointment date </param>
/// <param name="Gas"> What type of fuel the transport is using </param>
/// <param name="AverageGas"> Average fuel usage </param>
/// <param name="SeatNumber"> Transports seat count </param>
public Bus(string Nr, string Brand, string Model,
    DateTime Data, DateTime Review, string Gas, double AverageGas,
    int SeatNumber) : base(Nr, Brand, Model, Data, Review, Gas, AverageGas)
{
    this.SeatNumber = SeatNumber;
}

/// <summary>
/// ToString() override
/// </summary>
/// <returns> Returns a table row of base class ToString method +
/// another parameter called SeatNumber as a string </returns>
public override string ToString()
{
    return base.ToString() + String.Format(" {0, -18} |", SeatNumber);
}

/// <summary>
/// Compares 2 transport object to see if they are the same
/// </summary>
/// <param name="other"> Another transport object </param>
/// <returns> Returns true if the objects are the same, false if they are not
</returns>
public override bool Equals(Transport other)
{
    return this.Equals(other);
}

/// <summary>
/// Compares 2 transport object first by their purchase date then by their brand
name
/// </summary>
/// <param name="other"> Another transport object </param>
/// <returns> Returns 0 if the objects are the
/// same, 1 if the first object's
/// date is later then the second's
/// object and return -1 if if the second
/// object's date is later then the first's </returns>
public override int CompareTo(Transport other)
{
    int r = this.Data.CompareTo(other.Data);
    if (r == 0)
    {
        r = this.Brand.CompareTo(other.Brand);
    }
    return r;
}

/// <summary>
/// Compares who has more seats
/// </summary>
/// <param name="a"> Bus object a </param>
/// <param name="b"> Bus object b </param>
/// <returns> true if object a has more seats then object b </returns>
public static bool operator >(Bus a, Bus b)

```

```

{
    return a.SeatNumber > b.SeatNumber;
}

/// <summary>
/// Compares who has more seats
/// </summary>
/// <param name="a"> Bus object a </param>
/// <param name="b"> Bus object b </param>
/// <returns> true if object b has more seats then object a </returns>
public static bool operator <(Bus a, Bus b)
{
    return a.SeatNumber < b.SeatNumber;
}

/// <summary>
/// Adds the date of the next review appointment
/// </summary>
public override void AddReviewDate()
{
    Validity = Review_Data.AddMonths(6);
}
}
}

```

```

/////////////////////////////////////////////////////////////////
-----
/////////////////////////////////////////////////////////////////
Car.cs:

```

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebApplication4
{
    /// <summary>
    /// Transport's child class called Car
    /// </summary>
    class Car : Transport
    {

        public double Odometre { get; set; }

        /// <summary>
        /// Basic Car class constructor
        /// </summary>
        /// <param name="Nr"> Transport's Number plate </param>
        /// <param name="Brand"> Transport's brand </param>
        /// <param name="Model"> Model of our transport </param>
        /// <param name="Data"> Purchase date </param>
        /// <param name="Review"> Last review appointment date </param>
        /// <param name="Gas"> What type of fuel the transport is using </param>
        /// <param name="AverageGas"> Average fuel usage </param>
        /// <param name="Odometre"> How many km has the car drove </param>
        public Car(string Nr, string Brand, string Model,
            DateTime Data, DateTime Review, string Gas, double AverageGas,
            double Odometre) : base(Nr, Brand, Model, Data, Review, Gas, AverageGas)
        {
            this.Odometre = Odometre;
        }
    }
}

```

```

    /// <summary>
    /// ToString() override
    /// </summary>
    /// <returns> Returns a table row of base class ToString method +
    /// another parameter called Odometre as a string </returns>
    public override string ToString()
    {
        return base.ToString() + String.Format(" {0, -18} |", Odometre);
    }

    /// <summary>
    /// Compares 2 transport object to see if they are the same
    /// </summary>
    /// <param name="other"> Another trasport object </param>
    /// <returns> Returns true if the objects are the same, false if they are not
</returns>
    public override bool Equals(Transport other)
    {
        return this.Equals(other);
    }

    /// <summary>
    /// Compares 2 transport object to see if they are the same
    /// </summary>
    /// <param name="other"> Another trasport object </param>
    /// <returns> Returns true if the objects are the same, false if they are not
</returns>
    public override int CompareTo(Transport other)
    {
        return base.CompareTo(other);
    }

    /// <summary>
    /// Adds the date of the next review appointment
    /// </summary>
    public override void AddReviewDate()
    {
        Validity = Review_Data.AddYears(2);
    }

    /// <summary>
    /// Compares who drove less in total
    /// </summary>
    /// <param name="a"> Bus object a </param>
    /// <param name="b"> Bus object b </param>
    /// <returns> true if object b has drove less then object a </returns>
    public static bool operator >(Car a, Car b)
    {
        return a.Odometre > b.Odometre;
    }

    /// <summary>
    /// Compares who drove less in total
    /// </summary>
    /// <param name="a"> Bus object a </param>
    /// <param name="b"> Bus object b </param>
    /// <returns> true if object a has drove less then object b </returns>
    public static bool operator <(Car a, Car b)
    {
        return a.Odometre < b.Odometre;
    }
}

```

//////////////////////////////////////  
 \_\_\_\_\_  
 //////////////////////////////////////

85



```

/// <summary>
/// TrnasportContainer object register
/// </summary>
class VRegister
{

    private TransportContainer Container;
    public string City { get; set; }
    public string Address { get; set; }
    public string Phone { get; set; }

    /// <summary>
    /// Basic VRegister constructor
    /// </summary>
    /// <param name="Container"> The TranspoerContainer we want to add </param>
    /// <param name="City"> Cities name </param>
    /// <param name="Address"> Filial Address </param>
    /// <param name="Phone"> Filial's phone number </param>
    public VRegister(TransportContainer Container, string City, string Address, string
Phone)
    {
        this.Container = Container;
        this.City = City;
        this.Address = Address;
        this.Phone = Phone;
    }

    /// <summary>
    /// Gets a Transport from TransportContainer with the same index as the given one
    /// </summary>
    /// <param name="index"> Index of the trasnport we want to get </param>
    /// <returns> A trasport class object </returns>
    public Transport Get(int index)
    {
        return Container.Get(index);
    }

    /// <summary>
    /// Adds a transport class object to the TrasportContainer
    /// </summary>
    /// <param name="vroom"> Transport class object we want to add </param>
    public void Add(Transport vroom)
    {
        this.Container.Add(vroom);
    }

    /// <summary>
    /// Counts how many objects are in this register
    /// </summary>
    /// <returns> How many objects are in this register </returns>
    public int Count()
    {
        return this.Container.Count;
    }

}

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
-----
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

**InOut.cs:**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;

namespace WebApplication4
{
    static class InOut
    {
        /// <summary>
        /// Reads files from directory
        /// </summary>
        /// <param name="Path"> Direcroty path </param>
        /// <param name="Rez_File"> Rezult file's oath </param>
        /// <returns> List<VRegister> object, that has a list of register containing
information
        /// about every filial and every tranport </returns>
        public static List<VRegister> ReadFile(string Path, string Rez_File)
        {
            string filename = "";
            try
            {
                string Punc = ",/";
                List<VRegister> AllProperties = new List<VRegister>();
                string[] Files = Directory.GetFiles(Path);
                foreach (string file in Files)
                {
                    filename = file;
                    TransportContainer Transports = new TransportContainer();
                    string[] Lines = File.ReadAllLines(file, Encoding.ASCII);
                    string City = Lines[0];
                    string Address = Lines[1];
                    string Phone = Lines[2];
                    for (int i = 3; i < Lines.Count(); i++)
                    {
                        string[] Words = Lines[i].Split(Punc.ToCharArray(),
                            StringSplitOptions.RemoveEmptyEntries);
                        string Nr = Words[0];
                        string Brand = Words[1];
                        string Model = Words[2];
                        DateTime Data = new DateTime(Convert.ToInt32(Words[3]),
                            Convert.ToInt32(Words[4]), 1);
                        DateTime Review_Data = new DateTime(Convert.ToInt32(Words[5]),
                            Convert.ToInt32(Words[6]), Convert.ToInt32(Words[7]));
                        string Gas = Words[8];
                        double AverageGas = Double.Parse(Words[9]);
                        int LastSpot = int.Parse(Words[10]);
                        switch (Nr[0])
                        {
                            case ('A'):
                                Transports.Add(new Truck(Nr, Brand, Model, Data,
                                    Review_Data, Gas, AverageGas, LastSpot));

                                break;
                            case ('B'):

                                Transports.Add(new Car(Nr, Brand, Model, Data,
                                    Review_Data, Gas, AverageGas, LastSpot));

                                break;
                            case ('C'):

```



```

        Transports.Add(new Bus(Nr, Brand, Model, Data,
                                Review_Data, Gas, AverageGas, LastSpot));

        break;
    }

    }
    VRegister Register = new VRegister(Transports, City, Address, Phone);
    AllProperties.Add(Register);
}
return AllProperties;
}
catch (FormatException)
{
    File.Delete(filename);
    File.AppendAllText(Rez_File, String.Format("The program ran into a" +
        " problem while reading {0}. In order for the program to work" +
        ", the file was deleted\n", filename));
    return InOut.ReadFile(Path, Rez_File);
}
catch (IndexOutOfRangeException)
{
    File.Delete(filename);
    File.AppendAllText(Rez_File, String.Format("The program ran into a" +
        " problem while reading {0}. In order for the program to work," +
        " the file was deleted\n", filename));
    return InOut.ReadFile(Path, Rez_File);
}

}

/// <summary>
/// Prints base data from A VRegister list
/// </summary>
/// <param name="FileName"> File name where the data will be printed at </param>
/// <param name="Container"> The VRegister list </param>
/// <param name="Header"> Header used at the top of the table </param>
public static void PrintDataVRegister(string FileName, List<VRegister> Container,
string Header)
{
    File.AppendAllText(FileName, String.Format("{0}\n", Header));
    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Register = Container[i];
        File.AppendAllText(FileName, String.Format("{0}, {1}, {2} \n", Register.City
            , Register.Address, Register.Phone));
        File.AppendAllText(FileName, String.Format("| {0, -8} | {1, -12} | {2, -12}
| " +
            " {3, -18} | {4, -18} | {5, -10} | {6, -25} | {7, -18} |\n",
            "Nr", "Gamintojas", "Modelis", "Pagaminimo data", "Technikinio data",
            "Kuras", "Vidutinės kuro sanaudos", "Papildomi duomenys"));
        for (int m = 0; m < Register.Count(); m++)
        {
            File.AppendAllText(FileName, Register.Get(m).ToString() + "\n");
        }
    }
    File.AppendAllText(FileName, "\n");
}
}

```

```

/// <summary>
/// Prints TransportContainer type data into a file
/// </summary>
/// <param name="FileName"> File name where the data will be printed at </param>
/// <param name="Container"> The TransportContainer container </param>
/// <param name="Header"> Header used at the top of the table </param>
public static void PrintBestTransport(string FileName, TransportContainer Container,
string Header)
{
    File.AppendAllText(FileName, String.Format("{0}\n", Header));
    File.AppendAllText(FileName, String.Format("| {0, -12} | {1, -12} | {2, -8} |
{3, -20} |\n",
        "Gamintojas", "Modelis", "Nr", "Amžius (Metais)"));
    for (int i = 0; i < Container.Count; i++)
    {
        Transport transport = Container.Get(i);
        File.AppendAllText(FileName, String.Format("| {0, -12} | {1, -12} | {2, -8}"
+
        " | {3, -20} |\n", transport.Brand, transport.Model, transport.Nr,
        (DateTime.Today.AddYears(-transport.Data.Year).Year)));
    }
    File.AppendAllText(FileName, "\n");
}

/// <summary>
/// Prints TransportContainer type data into a file
/// </summary>
/// <param name="FileName"> File name where the data will be printed at </param>
/// <param name="Container"> The TransportContainer container </param>
/// <param name="Header"> Header used at the top of the table </param>
public static void PrintCarsWithGas(string FileName, TransportContainer Container,
string Header)
{
    File.AppendAllText(FileName, String.Format("{0}\n", Header));
    File.AppendAllText(FileName, String.Format("| {2, -8} | {0, -12} | {1, -12} |
{3, -15} |\n",
        "Gamintojas", "Modelis", "Nr", "Pagaminimo data"));
    for (int i = 0; i < Container.Count; i++)
    {
        Transport transport = Container.Get(i);
        File.AppendAllText(FileName, String.Format("| {2, -8} | {0, -12} | {1, -12}
|" +
        " {3, -15} |\n", transport.Brand, transport.Model, transport.Nr,
        transport.Data.ToShortDateString()));
    }
    File.AppendAllText(FileName, "\n");
}

/// <summary>
/// Prints TransportContainer type data into a file
/// </summary>
/// <param name="FileName"> File name where the data will be printed at </param>
/// <param name="Container"> The TransportContainer container </param>
/// <param name="Header"> Header used at the top of the table </param>
public static void PrintReviewDates(string FileName, TransportContainer Container,
string Header)
{
    File.AppendAllText(FileName, String.Format("{0}\n", Header));
    File.AppendAllText(FileName, String.Format("| {0, -12} | {1, -12} | {2, -8} |
{3, -30} |\n",
        "Gamintojas", "Modelis", "Nr", "Technikinio galiojimo pabaiga"));

```

```

        for (int i = 0; i < Container.Count; i++)
        {
            Transport transport = Container.Get(i);
            File.AppendAllText(FileName, String.Format("| {0, -12} | {1, -12} | {2, -8}
|" +
                " {3, -30} |\n", transport.Brand, transport.Model, transport.Nr,
transport.Validity.ToShortDateString() ));
        }
        File.AppendAllText(FileName, "\n");
    }
}
}
}

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
-----
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

### TaskUtils.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebApplication4
{
    /// <summary>
    /// A class containing all the methos, which calculate all the necessary calculations
    /// </summary>
    static class TaskUtils
    {
        /// <summary>
        /// Creates a TransportContainer made of the cars that use gas as fuel
        /// </summary>
        /// <param name="ContainerReg"> VRegister list </param>
        /// <returns> TransportContainer made of the cars that use gas as fuel </returns>
        public static TransportContainer CarsWithGas(List<VRegister> ContainerReg)
        {
            TransportContainer Container = new TransportContainer();

            for (int i = 0; i < ContainerReg.Count; i++)
            {
                VRegister Reg = ContainerReg[i];
                for (int m = 0; m < Reg.Count(); m++)
                {
                    Transport B = Reg.Get(m);
                    if (B.Nr[0] == 'B')
                    {
                        Container.Add(B);
                    }
                }
            }
            return Container;
        }

        /// <summary>
        /// Creates a VRegister list with all busses and their filial
        /// </summary>
        /// <param name="ContainerReg"> VRegister list </param>
        /// <returns> VRegister list with all busses and their filial </returns>
    }
}

```

```

public static List<VRegister> Busses(List<VRegister> ContainerReg)
{
    List<VRegister> Container = new List<VRegister>();

    for (int i = 0; i < ContainerReg.Count; i++)
    {
        TransportContainer Transports = new TransportContainer();

        VRegister Reg = ContainerReg[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            if (B.Nr[0] == 'C')
            {
                Transports.Add(B);
            }
        }
        Transports.Sort();
        VRegister TempReg = new VRegister(Transports, Reg.City, Reg.Address,
Reg.Phone);
        Container.Add(TempReg);
    }
    return Container;
}

/// <summary>
/// Creates a TransportContainer containing all the transports
/// that will need to go to a review appointment in less then 2 months
/// </summary>
/// <param name="Container"> VRegister list </param>
/// <returns> TransportContainer containing all the transports
/// that will need to go to a review appointment in less then 2 months </returns>
public static TransportContainer AddReviewTime(List<VRegister> Container)
{
    TransportContainer Transports = new TransportContainer();

    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Reg = Container[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            B.AddReviewDate();
            if (DateTime.Today < B.Validity && DateTime.Today >
B.Validity.AddMonths(-2))
            {
                Transports.Add(B);
            }
        }
    }
    return Transports;
}

/// <summary>
/// Finds the best Bus
/// </summary>
/// <param name="Container"> VRegister list </param>
/// <returns> Returns transport object who's child object is the bus with the most
seats </returns>
public static Transport BestBus(List<VRegister> Container)

```

```

{
    Transport Best_Bus = BusFinder(Container);
    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Reg = Container[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            if (B is Bus)
            {
                if ((Bus)B > (Bus)Best_Bus)
                {
                    Best_Bus = B;
                }
            }
        }
    }
    return Best_Bus;
}

/// <summary>
/// Finds the first bus in the VRegister list
/// </summary>
/// <param name="Container"> VRegister list </param>
/// <returns> First bus in VRegister list </returns>
private static Transport BusFinder(List<VRegister> Container)
{
    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Reg = Container[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            if (B is Bus)
            {
                return B;
            }
        }
    }
    return null;
}

/// <summary>
/// Finds the first car in the VRegister list
/// </summary>
/// <param name="Container"> VRegister list </param>
/// <returns> First car in VRegister list </returns>
private static Transport CarFinder(List<VRegister> Container)
{
    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Reg = Container[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            if (B is Car)
            {
                return B;
            }
        }
    }
    return null;
}

```

```

    }
}

}
return null;
}

/// <summary>
/// Finds the best car
/// </summary>
/// <param name="Container"> VRegister list </param>
/// <returns> Returns transport object who's child object
/// is the car with the least km driven </returns>
public static Transport BestCar(List<VRegister> Container)
{
    Transport Best_Car = CarFinder(Container);
    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Reg = Container[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            if (B is Car)
            {
                if ((Car)B < (Car)Best_Car)
                {
                    Best_Car = B;
                }
            }
        }
    }

    return Best_Car;
}

/// <summary>
/// Finds the first truck in the VRegister list
/// </summary>
/// <param name="Container"> VRegister list </param>
/// <returns> First trcuk in VRegister list </returns>
private static Transport TruckFinder(List<VRegister> Container)
{
    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Reg = Container[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            if (B is Truck)
            {
                return B;
            }
        }
    }

    return null;
}

/// <summary>

```

```

/// Finds the best truck
/// </summary>
/// <param name="Container"> VRegister list </param>
/// <returns> Returns transport object who's child object
/// is the truck with the most trailer size </returns>
public static Transport BestTruck(List<VRegister> Container)
{
    Transport Best_Truck = TruckFinder(Container);
    for (int i = 0; i < Container.Count; i++)
    {
        VRegister Reg = Container[i];
        for (int m = 0; m < Reg.Count(); m++)
        {
            Transport B = Reg.Get(m);
            if (B is Truck)
            {
                if ((Truck)B > (Truck)Best_Truck)
                {
                    Best_Truck = B;
                }
            }
        }
    }
    return Best_Truck;
}
}

```

```

/////////////////////////////////////////////////////////////////
-----
/////////////////////////////////////////////////////////////////

```

### Web.aspx.cs:

```

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

namespace WebApplication4
{
    public partial class Web : System.Web.UI.Page
    {
        private string DirectoryPath;
        private List<VRegister> Container;
        private List<VRegister> Busses;
        private TransportContainer TransportC;
        private string Rez_Path;
        private string Rez_Path2;

        protected void Page_Load(object sender, EventArgs e)
        {
            DirectoryPath = Server.MapPath("App_Data");
            Rez_Path = Server.MapPath("Rez/Results.txt");

```

```

        Rez_Path2 = Server.MapPath("Rez/Apžiūra.txt");
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        if (File.Exists(Rez_Path))
        {
            File.Delete(Rez_Path);
        }
        if (File.Exists(Rez_Path2))
        {
            File.Delete(Rez_Path2);
        }
        Container = InOut.ReadFile(DirectoryPath, Rez_Path);
        InOut.PrintDataVRegister(Rez_Path, Container, "Visos transporto priemonės:");
        FillAllData(ref Table1, Container);

        Label2.Text = "Visos transporto priemonės:";
        Label3.Text = "Paspaudę mygtuką dar kartą, pamatysite geriausiais transporto
priemonės.";
        Label3.Visible = true;
        Button2.Visible = true;
        Button1.Visible = false;
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        Container = InOut.ReadFile(DirectoryPath, Rez_Path);
        Transport Best_Car = TaskUtils.BestCar(Container);
        Transport Best_Bus = TaskUtils.BestBus(Container);
        Transport Best_Truck = TaskUtils.BestTruck(Container);
        TransportC = new TransportContainer();
        TransportC.Add(Best_Bus);
        TransportC.Add(Best_Car);
        TransportC.Add(Best_Truck);

        InOut.PrintBestTransport(Rez_Path, TransportC, "Geriausios transporto
priemonės:");
        FillTableWithBestTransportData(ref Table1, TransportC);

        Label2.Text = "Geriausios transporto priemonės:";
        Label3.Text = "Paspaudę mygtuką dar kartą, pamatysite visas mašinas, kurios
naudoja benzina";
        Button3.Visible = true;
        Button2.Visible = false;
    }

    protected void Button3_Click(object sender, EventArgs e)
    {
        Container = InOut.ReadFile(DirectoryPath, Rez_Path);
        TransportC = TaskUtils.CarsWithGas(Container);

        InOut.PrintCarsWithGas(Rez_Path, TransportC, "Visos mašinos, kurios naudoja
benzina:");
        FillTableWithGasCarsData(ref Table1, TransportC);

        Label2.Text = "Visos mašinos, kurios naudoja benzina:";
        Label3.Text = "Paspaudę mygtuką dar kartą, pamatysite visų filialų autobusų
sarašus";
        Button4.Visible = true;
        Button3.Visible = false;
    }
}

```



```

protected void Button4_Click1(object sender, EventArgs e)
{
    Container = InOut.ReadFile(DirectoryPath, Rez_Path);
    Busses = TaskUtils.Busses(Container);

    InOut.PrintDataVRegister(Rez_Path, Busses, "Visų filialų autobusų sarašai:");
    FillAllBusFilialData(ref Table1, Busses);

    Label2.Text = "Visų filialų autobusų sarašai:";
    Label3.Text = "Paspaudę mygtuką dar kartą, pamatysite transporto priemonių
sarašą"; //Here
    Label4.Visible = true;
    Label4.Text = "kurioms iki technikinio liko mažiau nei 2 mėn";
    Button5.Visible = true;
    Button4.Visible = false;
}

protected void Button5_Click(object sender, EventArgs e)
{
    Container = InOut.ReadFile(DirectoryPath, Rez_Path);
    TransportC = TaskUtils.AddReviewTime(Container);

    InOut.PrintReviewDates(Rez_Path, TransportC, "Transporto priemonės," +
        " kurioms liko 2 mėn arba mažiau iki technikinio:");
    InOut.PrintReviewDates(Rez_Path2, TransportC, "Transporto priemonės," +
        " kurioms liko 2 mėn arba mažiau iki technikinio:");
    FillTableWithReviewData(ref Table1, TransportC);

    Label4.Visible = false;
    Label2.Text = "Transporto priemonės, kurioms liko 2 mėn. arba mažiau iki
technikinio:";
    Label3.Text = "Paspaudę mygtuką grįšite į pradžią";
    Button1.Visible = true;
    Button5.Visible = false;
}

private static void FillAllData(ref Table table, List<VRegister> ContainerF)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Nr",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Gamintojas",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Modelis",
        CssClass = "bold-text"
    });
}

```

```

firstRow.Cells.Add(new TableCell()
{
    Text = "Pagaminimo data",
    CssClass = "bold-text"

});
firstRow.Cells.Add(new TableCell()
{
    Text = "Technikinio data",
    CssClass = "bold-text"

});
firstRow.Cells.Add(new TableCell()
{
    Text = "Kuras",
    CssClass = "bold-text"

});
firstRow.Cells.Add(new TableCell()
{
    Text = "Vidutinės kuro sanaudos",
    CssClass = "bold-text"

});
firstRow.Cells.Add(new TableCell()
{
    Text = "Papildomi duomenys",
    CssClass = "bold-text"

});
table.Rows.Add(firstRow);
for (int i = 0; i < ContainerF.Count(); i++)
{
    VRegister Regis = ContainerF[i];

    for (int m = 0; m < Regis.Count(); m++)
    {
        Transport T = Regis.Get(m);

        var row = new TableRow();

        row.Cells.Add(new TableCell()
        {
            Text = T.Nr.ToString()
        }
        );
        row.Cells.Add(new TableCell()
        {
            Text = T.Brand.ToString()
        }
        );
        row.Cells.Add(new TableCell()
        {
            Text = T.Model.ToString()
        }
        );
        row.Cells.Add(new TableCell()
        {
            Text = T.Data.ToShortDateString()
        }
        );
        row.Cells.Add(new TableCell()
        {

```

```

        Text = T.Review_Data.ToShortDateString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = T.Gas.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = T.AverageGasUsage.ToString()
    });
    switch (T.Nr[0])
    {

        case ('A'):

            row.Cells.Add(new TableCell()
            {
                Text = (T as Truck).TrailerSize.ToString()
            });

            break;
        case ('B'):
            row.Cells.Add(new TableCell()
            {
                Text = (T as Car).Odometre.ToString()
            });

            break;
        case ('C'):
            row.Cells.Add(new TableCell()
            {
                Text = (T as Bus).SeatNumber.ToString()
            });

            break;
    }
}

```

```

table.Rows.Add(row);

```

```

    }
}

```

```

private static void FillAllBusFilialData(ref Table table, List<VRegister>
ContainerF)
{

```

```

    for (int i = 0; i < ContainerF.Count(); i++)
    {
        var Header2 = new TableRow();

        Header2.Cells.Add(new TableCell()
        {
            Text = "Miestsas",

```

```

        CssClass = "bold-text"
    });
Header2.Cells.Add(new TableCell()
{
    Text = "Adresas",
    CssClass = "bold-text"
});
Header2.Cells.Add(new TableCell()
{
    Text = "Telefonas",
    CssClass = "bold-text"
});

var firstRow = new TableRow();

firstRow.Cells.Add(new TableCell()
{
    Text = "Nr",
    CssClass = "bold-text"
});
firstRow.Cells.Add(new TableCell()
{
    Text = "Gamintojas",
    CssClass = "bold-text"
});
firstRow.Cells.Add(new TableCell()
{
    Text = "Modelis",
    CssClass = "bold-text"
});
firstRow.Cells.Add(new TableCell()
{
    Text = "Pagaminimo data",
    CssClass = "bold-text"
});
firstRow.Cells.Add(new TableCell()
{
    Text = "Technikinio data",
    CssClass = "bold-text"
});
firstRow.Cells.Add(new TableCell()
{
    Text = "Kuras",
    CssClass = "bold-text"
});
firstRow.Cells.Add(new TableCell()
{
    Text = "Vidutinės kuro sanaudos",
    CssClass = "bold-text"
});
firstRow.Cells.Add(new TableCell()
{
    Text = "Vietų skaičius",
    CssClass = "bold-text"
});
});

```

```

VRegister Regis = ContainerF[i];

table.Rows.Add(Header2);

var Header = new TableRow();
Header.Cells.Add(new TableCell()
{
    Text = Regis.City
});
Header.Cells.Add(new TableCell()
{
    Text = Regis.Address
});
Header.Cells.Add(new TableCell()
{
    Text = Regis.Phone
});

table.Rows.Add(Header);

table.Rows.Add(firstRow);

for (int m = 0; m < Regis.Count(); m++)
{
    Transport T = Regis.Get(m);

    var row = new TableRow();

    row.Cells.Add(new TableCell()
    {
        Text = T.Nr.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Brand.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Model.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Data.ToShortDateString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Review_Data.ToShortDateString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Gas.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.AverageGasUsage.ToString()
    }
    );
    switch (T.Nr[0])
    {

        case ('A'):

```

```

        row.Cells.Add(new TableCell()
        {
            Text = (T as Truck).TrailerSize.ToString()
        });

        break;
    case ('B'):
        row.Cells.Add(new TableCell()
        {
            Text = (T as Car).Odometre.ToString()
        });

        break;
    case ('C'):
        row.Cells.Add(new TableCell()
        {
            Text = (T as Bus).SeatNumber.ToString()
        });

        break;
    }

    table.Rows.Add(row);

}
}
}

```

```

private static void FillTableWithReviewData(ref Table table, TransportContainer Container)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Gamintojas",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Modelis",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Nr",
        CssClass = "bold-text"
    });

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Technikinės apžiūros galiojimo pabaiga",
        CssClass = "bold-text"
    });
}

```

```

table.Rows.Add(firstRow);

for (int m = 0; m < Container.Count; m++)
{
    Transport T = Container.Get(m);

    var row = new TableRow();

    row.Cells.Add(new TableCell()
    {
        Text = T.Brand.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = T.Model.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = T.Nr.ToString()
    });
    row.Cells.Add(new TableCell()
    {
        Text = T.Validity.ToShortDateString()
    });

    table.Rows.Add(row);

}

}

private static void FillTableWithBestTransportData(ref Table table,
TransportContainer Container)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Gamintojas",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Modelis",
        CssClass = "bold-text"
    });
    firstRow.Cells.Add(new TableCell()
    {
        Text = "Nr",
        CssClass = "bold-text"
    });

```

```

));
firstRow.Cells.Add(new TableCell()
{
    Text = "Amžius (Metais)",
    CssClass = "bold-text"

});

table.Rows.Add(firstRow);

for (int m = 0; m < Container.Count; m++)
{
    Transport T = Container.Get(m);

    var row = new TableRow();

    row.Cells.Add(new TableCell()
    {
        Text = T.Brand.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Model.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Nr.ToString()
    }
    );

    row.Cells.Add(new TableCell()
    {
        Text = DateTime.Today.AddYears(-T.Data.Year).Year.ToString()
    }

    );

    table.Rows.Add(row);

}

```

```

}

private static void FillTableWithGasCarsData(ref Table table, TransportContainer
Container)
{
    var firstRow = new TableRow();

    firstRow.Cells.Add(new TableCell()
    {
        Text = "Nr",
        CssClass = "bold-text"

    });
    firstRow.Cells.Add(new TableCell()

```



```

{
    Text = "Gamintojas",
    CssClass = "bold-text"

});
firstRow.Cells.Add(new TableCell()
{
    Text = "Modelis",
    CssClass = "bold-text"

});
firstRow.Cells.Add(new TableCell()
{
    Text = "Pagaminimo metai",
    CssClass = "bold-text"

});

table.Rows.Add(firstRow);

for (int m = 0; m < Container.Count; m++)
{
    Transport T = Container.Get(m);

    var row = new TableRow();

    row.Cells.Add(new TableCell()
    {
        Text = T.Nr.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Brand.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Model.ToString()
    }
    );
    row.Cells.Add(new TableCell()
    {
        Text = T.Data.ToShortDateString()
    }

    );

    table.Rows.Add(row);

}

}

}

```

```

////////////////////////////////////

```

---

**Page\_Css.css:**

```
.auto-style1 {
    width: 100%;
    padding: 10px;
    background-color: cornflowerblue;
    border-style: solid;
    border-color: black;
}

body {
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: darkcyan;
}

.auto-style2 {
    height: 29px;
}

.auto-style3 {
    height: 23px;
}

table {
    background-color: cornsilk;
    border-color: Black;
    border-style: Solid;
    border-width: 5px;
}

.auto-style5 {
    height: 22px;
    justify-content: center;
    align-items: center;
}

.tb1 {
    background-color: cornsilk;
    border-color: Black;
    border-style: Solid;
    border-width: 5px;
}

.tb1side {
    background-color: darkslateblue;
}
```



---

## 4.7. Pradiniai duomenys ir rezultatai

### Testas Nr. 1:

**Tikslas:** Standartinis atvejis, kai apskaičiuojami keli atsakymai.

Directory „App\_Data“:

Name	Date modified	Type	Size
 U4	2022-05-08 18:57	Text Document	1 KB
 u7	2022-05-08 19:49	Text Document	1 KB

U4.txt:

```
Radviliskis
Birutes. 56
879846546
ATT 996,Toyota,g5,2004/10,2002/1/10,Benzinas,7,24000
BTM 666,Toyota,g7,2004/10,2002/1/10,Benzinas,8,100000
CCC 789,Pros,856,2004/1,1000/1/10,Benzinas,9,1000
CHC 788,Pros,856,2002/1,1000/1/10,Benzinas,2,1000
CHC 784,Pros,856,2003/1,1000/1/10,Benzinas,7,1000
ATS 996,Toyota,g5,2003/10,2002/1/10,Benzinas,7,210
```

u7.txt:

```
Jonava
Rodis. 12
879846546
BSS 105,Toyota,g2,1000/10,2021/06/10,Benzinas,7,24000
CSS 106,Toyota,g2,2015/10,2021/06/10,Dyzelis,7,24000
AGS 956,Toyota,g5,2021/10,2021/06/10,Benzinas,7,210000
CuS 16,Toyota,g2,2021/04,2021/12/08,Dyzelis,7,240005
BuS 16,Toyota,f12,2021/04,2020/05/28,Benzinas,7,240
AuS 16,Subaru,g88,2021/04,2021/04/08,Dyzelis,7,240
```

Results.txt:

Visos transporto priemonės:							
Radviliskis, Birutes. 56, 879846546							
Nr	Gamintojas	Modelis	Pagaminimo data	Technikinio data	Kuras	Vidutinės kuro sanaudos	Papildomi duomenys
ATT 996	Toyota	g5	2004-10-01	2002-01-10	Benzinas	7	24000
BTM 666	Toyota	g7	2004-10-01	2002-01-10	Benzinas	8	100000
CCC 789	Pros	856	2004-01-01	1000-01-10	Benzinas	9	1000
CHC 788	Pros	856	2002-01-01	1000-01-10	Benzinas	2	1000
CHC 784	Pros	856	2003-01-01	1000-01-10	Benzinas	7	1000
ATS 996	Toyota	g5	2003-10-01	2002-01-10	Benzinas	7	210
Jonava, Rodis. 12, 879846546							
Nr	Gamintojas	Modelis	Pagaminimo data	Technikinio data	Kuras	Vidutinės kuro sanaudos	Papildomi duomenys
BSS 105	Toyota	g2	1000-10-01	2021-06-10	Benzinas	7	24000
CSS 106	Toyota	g2	2015-10-01	2021-06-10	Dyzelis	7	24000
AGS 956	Toyota	g5	2021-10-01	2021-06-10	Benzinas	7	210000
CuS 16	Toyota	g2	2021-04-01	2021-12-08	Dyzelis	7	240005
BuS 16	Toyota	f12	2021-04-01	2020-05-28	Benzinas	7	240
AuS 16	Subaru	g88	2021-04-01	2021-04-08	Dyzelis	7	240
Geriausios transporto priemonės:							
Gamintojas	Modelis	Nr	Amžius (Metais)				
Toyota	g2	CuS 16	1				
Toyota	f12	BuS 16	1				
Toyota	g5	AGS 956	1				
Visos mašinos, kurios naudoja benziną:							
Nr	Gamintojas	Modelis	Pagaminimo data				
BTM 666	Toyota	g7	2004-10-01				
BSS 105	Toyota	g2	1000-10-01				
BuS 16	Toyota	f12	2021-04-01				
Visų filialų autobusų sąrašai:							
Radviliskis, Birutes. 56, 879846546							
Nr	Gamintojas	Modelis	Pagaminimo data	Technikinio data	Kuras	Vidutinės kuro sanaudos	Papildomi duomenys
CCC 789	Pros	856	2004-01-01	1000-01-10	Benzinas	9	1000
CHC 784	Pros	856	2003-01-01	1000-01-10	Benzinas	7	1000
CHC 788	Pros	856	2002-01-01	1000-01-10	Benzinas	2	1000
Jonava, Rodis. 12, 879846546							
Nr	Gamintojas	Modelis	Pagaminimo data	Technikinio data	Kuras	Vidutinės kuro sanaudos	Papildomi duomenys
CuS 16	Toyota	g2	2021-04-01	2021-12-08	Dyzelis	7	240005
CSS 106	Toyota	g2	2015-10-01	2021-06-10	Dyzelis	7	24000
Transporto priemonės, kurioms like 2 mėn arba mažiau iki technikinio:							
Gamintojas	Modelis	Nr	Technikinio galiojimo pabaiga				
Toyota	g5	AGS 956	2022-06-10				
Toyota	g2	CuS 16	2022-06-08				
Toyota	f12	BuS 16	2022-05-28				

Apžiūra.txt:

Transporto priemonės, kurioms like 2 mėn arba mažiau iki technikinio:				
Gamintojas	Modelis	Nr	Technikinio galiojimo pabaiga	
Toyota	g5	AGS 956	2022-06-10	
Toyota	g2	CuS 16	2022-06-08	
Toyota	f12	BuS 16	2022-05-28	

User interface



## UAB "Strelė" įmonės transportų srašų tvarkyklė

Visos transporto priemonės:

Nr	Gamintojas	Modelis	Pagaminimo data	Techninio data	Kuras	Vidutinės kuro sąnaudos	Papildomi duomenys
ATT 996	Toyota	g5	2004-10-01	2002-01-10	Benzinas	7	24000
BTM 666	Toyota	g7	2004-10-01	2002-01-10	Benzinas	8	100000
CCC 789	Pros	856	2004-01-01	1000-01-10	Benzinas	9	1000
CHC 788	Pros	856	2002-01-01	1000-01-10	Benzinas	2	1000
CHC 784	Pros	856	2003-01-01	1000-01-10	Benzinas	7	1000
ATS 996	Toyota	g5	2003-10-01	2002-01-10	Benzinas	7	210
BSS 105	Toyota	g2	1000-10-01	2021-06-10	Benzinas	7	24000
CSS 106	Toyota	g2	2015-10-01	2021-06-10	Dyzelis	7	24000
AGS 956	Toyota	g5	2021-10-01	2021-06-10	Benzinas	7	210000
CuS 16	Toyota	g2	2021-04-01	2021-12-08	Dyzelis	7	240005
BuS 16	Toyota	f12	2021-04-01	2020-05-28	Benzinas	7	240
AuS 16	Subaru	g88	2021-04-01	2021-04-08	Dyzelis	7	240

Paspaudę mygtuką dar kartą, pamatysite geriausias transporto priemones.

Button

## UAB "Strelė" įmonės transportų srašų tvarkyklė

Geriausios transporto priemonės:

Gamintojas	Modelis	Nr	Amžius (Metais)
Toyota	g2	CuS 16	1
Toyota	f12	BuS 16	1
Toyota	g5	AGS 956	1

Paspaudę mygtuką dar kartą, pamatysite visas mašinas, kurios naudoja benzina

Button

## UAB "Strelė" įmonės transportų srašų tvarkyklė

Visos mašinos, kurios naudoja benzina:

Nr	Gamintojas	Modelis	Pagaminimo metai
BTM 666	Toyota	g7	2004-10-01
BSS 105	Toyota	g2	1000-10-01
BuS 16	Toyota	f12	2021-04-01

Paspaudę mygtuką dar kartą, pamatysite visų filialų autobusų sąrašus

Button

## UAB "Strelė" įmonės transportų srašų tvarkyklė

Visų filialų autobusų sąrašai:

Miestas	Adresas	Telefonas					
Radviliskis	Birutes. 56	879846546					
Nr	Gamintojas	Modelis	Pagaminimo data	Technikinio data	Kuras	Vidutinės kuro sanaudos	Vietų skaičius
CCC 789	Pros	856	2004-01-01	1000-01-10	Benzinas	9	1000
CHC 784	Pros	856	2003-01-01	1000-01-10	Benzinas	7	1000
CHC 788	Pros	856	2002-01-01	1000-01-10	Benzinas	2	1000
Miestas	Adresas	Telefonas					
Jonava	Rodis. 12	879846546					
Nr	Gamintojas	Modelis	Pagaminimo data	Technikinio data	Kuras	Vidutinės kuro sanaudos	Vietų skaičius
CuS 16	Toyota	g2	2021-04-01	2021-12-08	Dyzelis	7	240005
CSS 106	Toyota	g2	2015-10-01	2021-06-10	Dyzelis	7	24000

Paspaudę mygtuką dar kartą, pamatysite transporto priemonių sąrašą  
kurios iki technikinio liko mažiau nei 2 mėn

Button

## UAB "Strelė" įmonės transportų srašų tvarkyklė

Transporto priemonės, kurioms liko 2 mėn. arba mažiau iki techninio:

Gamintojas	Modelis	Nr	Techninės apžiūros galiojimo pabaiga
Toyota	g5	AGS 956	2022-06-10
Toyota	g2	CuS 16	2022-06-08
Toyota	f12	BuS 16	2022-05-28

Button

Paspaudę mygtuką grįšite į pradžią

### Testas Nr. 2:

Tikslas: Parodyti išimčių veikimą

Directory „App\_Data“:

Name	Date modified	Type	Size
U4	2022-05-08 18:57	Text Document	1 KB
u7	2022-05-08 19:49	Text Document	1 KB

U4.txt:

```
Radviliskis  
Birutes. 56  
879846546
```

```
ATT 996,Toyota,g5,2004/10,2002/1/10,Benzinas,7,24000
BTM 666,Toyota,g7,2004/10,2002/1/10,Benzinas,8,100000
CCC 789,Pros,856,2004/1,1000/1/10,Benzinas,9,1000
CHC 788,Pros,856,2002/1,1000/1/10,Benzinas,2,1000
CHC 784,Pros,856,2003/1,1000/1/10,Benzinas,7,1000
ATS 996,Toyota,g5,2003/10,2002/1/10,Benzinas,7,210
```

u7.txt:

```
Jonava
Rodis. 12
879846546
BSS 105,Toyota,g2,1000/10,2021/06/10,Benzinas,7,24000
CSS 106,Toyota,g2,2015/10,2021/06/10,Dyzelis,7,24000
AGS 956,Toyota,g5,2021/10,2021/06/10,Benzinas,7,210000
CuS 16,Toyota,g2,2021/04,2021/12/08,Dyzelis,7,240005
BuS 16,Toyota,f12,2021/04,2020/05/28,Benzinas,7,240
AuS 16,Subaru,g88,2021/04,2021/04/08,Dyzelis,7,240
,sad.lams.c,as/.czmx.,vbns.d,knfv
```

Results.txt:

```
The program ran into a proble while reading C:\Users\Lenovo\source\repos\WebApplication4\WebApplication4\AppData\u7.txt. In order for the program to work, the file was delet
Visos transporto priemonės:
Radviliskis, Birutes. 56, 879846546
| Nr | Gamintojas | Modelis | Pagaminimo data | Technikinio data | Kuras | Vidutinės kuro sanaudos | Papildomi duomenys |
| ATT 996 | Toyota | g5 | 2004-10-01 | 2002-01-10 | Benzinas | 7 | 24000 |
| BTM 666 | Toyota | g7 | 2004-10-01 | 2002-01-10 | Benzinas | 8 | 100000 |
| CCC 789 | Pros | 856 | 2004-01-01 | 1000-01-10 | Benzinas | 9 | 1000 |
| CHC 788 | Pros | 856 | 2002-01-01 | 1000-01-10 | Benzinas | 2 | 1000 |
| CHC 784 | Pros | 856 | 2003-01-01 | 1000-01-10 | Benzinas | 7 | 1000 |
| ATS 996 | Toyota | g5 | 2003-10-01 | 2002-01-10 | Benzinas | 7 | 210 |

Geriausios transporto priemonės:
| Gamintojas | Modelis | Nr | Amžius (Metais) |
| Pros | 856 | CCC 789 | 18 |
| Toyota | g7 | BTM 666 | 18 |
| Toyota | g5 | ATT 996 | 18 |

Visos mašinos, kurios naudoja benzina:
| Nr | Gamintojas | Modelis | Pagaminimo data |
| BTM 666 | Toyota | g7 | 2004-10-01 |

Visų filialų autobusų sąrašai:
Radviliskis, Birutes. 56, 879846546
| Nr | Gamintojas | Modelis | Pagaminimo data | Technikinio data | Kuras | Vidutinės kuro sanaudos | Papildomi duomenys |
| CCC 789 | Pros | 856 | 2004-01-01 | 1000-01-10 | Benzinas | 9 | 1000 |
| CHC 784 | Pros | 856 | 2003-01-01 | 1000-01-10 | Benzinas | 7 | 1000 |
| CHC 788 | Pros | 856 | 2002-01-01 | 1000-01-10 | Benzinas | 2 | 1000 |

Transporto priemonės, kurioms like 2 mėn arba mažiau iki technikinio:
| Gamintojas | Modelis | Nr | Technikinio galiojimo pabaiga |
```

Apžiūra.txt:

```
Transporto priemonės, kurioms like 2 mėn arba mažiau iki technikinio:
| Gamintojas | Modelis | Nr | Technikinio galiojimo pabaiga |
|
```

User interface

## UAB "Strelė" įmonės transportų srašų tvarkyklė

Paspauskite mygtuką, kad pamatytumėte visas transporto priemones filiale.

Button

### UAB "Strelė" įmonės transportų srašų tvarkyklė

Visos transporto priemonės:

Nr	Gamintojas	Modelis	Pagaminimo data	Techninio data	Kuras	Vidutinės kuro sanaudos	Papildomi duomenys
ATT 996	Toyota	g5	2004-10-01	2002-01-10	Benzinas	7	24000
BTM 666	Toyota	g7	2004-10-01	2002-01-10	Benzinas	8	100000
CCC 789	Pros	856	2004-01-01	1000-01-10	Benzinas	9	1000
CHC 788	Pros	856	2002-01-01	1000-01-10	Benzinas	2	1000
CHC 784	Pros	856	2003-01-01	1000-01-10	Benzinas	7	1000
ATS 996	Toyota	g5	2003-10-01	2002-01-10	Benzinas	7	210

Paspaudę mygtuką dar kartą, pamatysite geriausias transporto priemones.

Button

### UAB "Strelė" įmonės transportų srašų tvarkyklė

Geriausios transporto priemonės:

Gamintojas	Modelis	Nr	Amžius (Metais)
Pros	856	CCC 789	18
Toyota	g7	BTM 666	18
Toyota	g5	ATT 996	18

Paspaudę mygtuką dar kartą, pamatysite visas mašinas, kurios naudoja benzina

Button

### UAB "Strelė" įmonės transportų srašų tvarkyklė

Visos mašinos, kurios naudoja benzina:

Nr	Gamintojas	Modelis	Pagaminimo metai
BTM 666	Toyota	g7	2004-10-01

Paspaudę mygtuką dar kartą, pamatysite visų filialų autobusų sąrašus

Button



## UAB "Strelė" įmonės transportų srašų tvarkyklė

Visų filialų autobusų sąrašai:

Miestas	Adresas	Telefonas					
Radviliskis	Birutes. 56	879846546					
Nr	Gamintojas	Modelis	Pagaminimo data	Techninio data	Kuras	Vidutinės kuro sanaudos	Vietų skaičius
CCC 789	Pros	856	2004-01-01	1000-01-10	Benzinas	9	1000
CHC 784	Pros	856	2003-01-01	1000-01-10	Benzinas	7	1000
CHC 788	Pros	856	2002-01-01	1000-01-10	Benzinas	2	1000

Paspaudę mygtuką dar kartą, pamatysite transporto priemonių sąrašą kurioms iki techninio liko mažiau nei 2 mėn

Button

## UAB "Strelė" įmonės transportų srašų tvarkyklė

Transporto priemonės, kurioms liko 2 mėn. arba mažiau iki techninio:

Gamintojas	Modelis	Nr	Techninės apžiūros galiojimo pabaiga
------------	---------	----	--------------------------------------

Button

Paspaudę mygtuką grįšite į pradžią

### 4.8. Dėstytojo pastabos

Pažymys: 6



Vacius Jusas - Kt, 12 geg. 2022, 11:04

1. Ne visi metodai komentuoti.



Vacius Jusas - Kt, 12 geg. 2022, 11:07

2. O kur ryšiai klasių diagramoje?



Vacius Jusas - Kt, 12 geg. 2022, 11:08

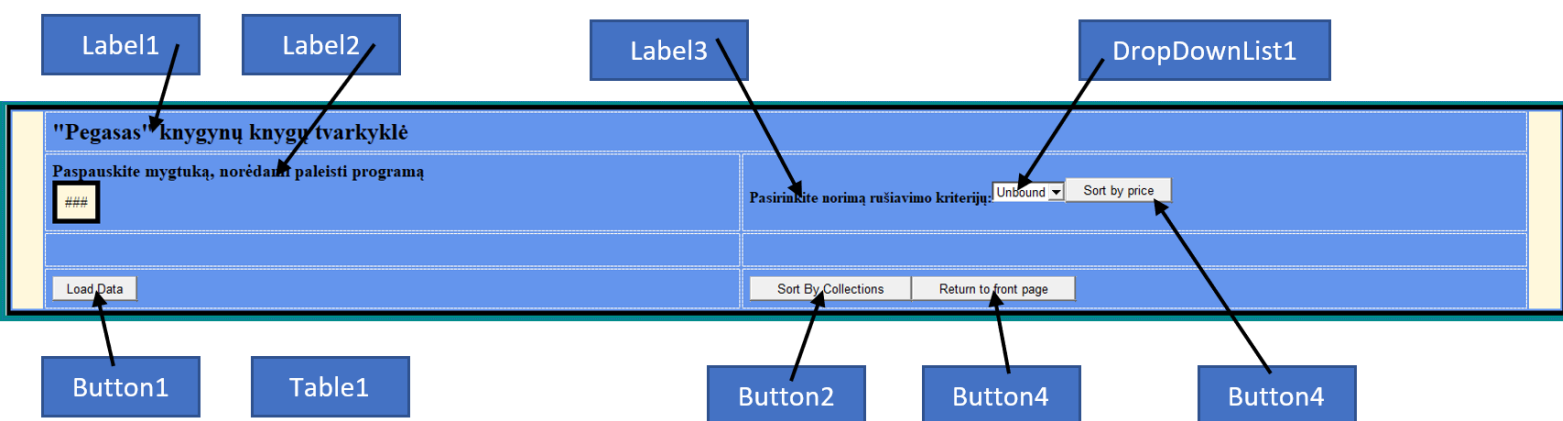
3. Dėstytojo pastabose nėra pažymių.

## 5. Deklaratyvusis programavimas (L5)

### 5.1. Darbo užduotis

**LDD\_14. Knygynai.** Knygynų tinklo „Pegasus“ knygynų knygų asortimentai surašyti atskiruose failuose: pirmoje failo eilutėje knygyno pavadinimas (**failų daug**), toliau kiekvienai knygai skiriama po vieną eilutę: knygos autorius, pavadinimas, leidimo metai, kaina, parduotų egzempliorių skaičius, likusių neparduotų egzempliorių skaičius. **Atskirame faile** įrašyti knygų atrinkimo kriterijų rinkiniai. Jų gali būti daug. Vieną kriterijų rinkinį sudaro leidimo metų intervalas (nuo-iki) ir ribinė tinkama pirkimo kaina. Vienas rinkinys užima vieną failo eilutę. Suformuoti knygų, tenkinančių atrankos kriterijus, sąrašus kiekvienam kriterijų rinkiniui atskirai. Sąrašų elementus surikiuoti pagal pavadinimą ir populiarumą. Kiekviename iš sąrašų suskaičiuoti kainų reikšmių vidurkį. Sudarykite sąrašą knygų, kurių kaina mažesnė arba didesnė (pasirinkimas per vartotojo sąsają) už sąrašo vidurkį.

### 5.2. Grafinės vartotojo sąsajos schema

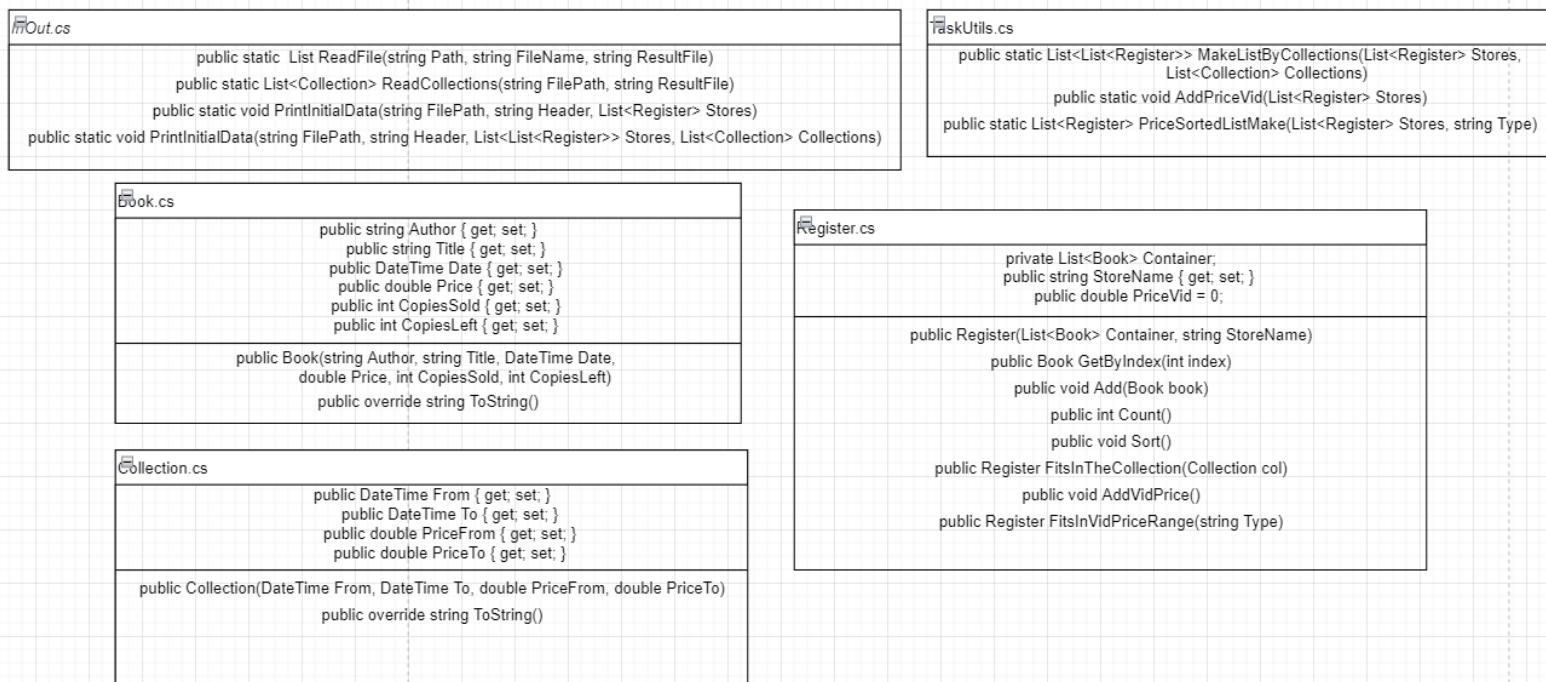


### 5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label1	Text	„Pegasus“ knygynų knygų tvarkyklė“
Label2	Text	„Paspauskite mygtuką, norėdami paleisti programą“ + "Pradiniai duomenys:" + "Duomenys pagal kriterijus iš kriterijų failo:" + "Atskirti pagal kainas, kurios yra " + DropDownList1.SelectedItem.Value.ToString() + " už vidurkį:" + "Paspauskite mygtuką, norėdami paleisti programą"
Label3	Text	„Pasirinkite norimą rūšiavimo kriterijų:“

Button1	Text	„Load Data“
Button1	OnClick	Loads initial data
Button2	Text	„Sort By Collections“
Button2	OnClick	Loads All data sorted by Collections
Button3	Text	„Sort by price“
Button3	OnClick	Sorts by price
Button4	Text	„Return to front page“
Button4	OnClick	Returns to main page
Tabel1	GridLines	
DropDownList1	List	

## 5.4. Klasių diagrama



## 5.5. Programos vartotojo vadovas

Pirmiausia, norint, jog programa galėtų daryti savo skaičiavimus, reikia sukurti keletą failų. Pirmieji failai turės visą informaciją apie kiekvieną parduotuvę „Pegasas“. Jų gali būti daug.

### Failo struktūra:

Pirmoje failo eilutėje yra parduotuvės pavadinimas.

Sekančiuose eilutėse yra, kiekvienos knygos esančios šioje parduotuvėje duomenys.

### Eilutės struktūra:

Knygos autorius, pavadinimas, leidimo metai, kaina, parduotų egzempliorių skaičius, likusių neparduotų egzempliorių skaičius. (Visi šie dydžiai skiriami kableliais, be tarpų)(Data skiriama „/“ tarp metų, mėnesių ir dienų).

Kai turime susikūrę šiuos failus, mus reikia sukurti dar vieną failą, pavadinimu: „**Collections.txt**“. (Atskiras failas, tik vienas). Šiame faile bus laikoma kriterijais, pagal kuriuos bus sudaromi rinkiniai.

#### **Failo struktūra:**

Failą sudaro eilutės, su kriterijais.

#### **Eilutės struktūra:**

Metų intervalas (nuo-iki) ir kaina (nuo-iki). Visą eilutę sudaro 4 dydžiai, kuriuos skiria kableliai(Metus, dienas ir mėnesius skiria „/“).

Kai turime sukūrę ir sukėlę failus, galime paleisti programą..

#### **Programos veikimas:**

Programa gan paprasta. Paspaudus pirmąjį mygtuką, bus atspausdinti pradiniai duomenys į ekraną. Sekantis mygtukas atspausdint sudarytus rinkinius pagal kriterijus iš kriterijų failo. Tada ekrane atsiras išlendantis pasirinkimo laukas, kurio pagalba galėsite pasirinkti ar bus spausdinamos knygos, kurių kaina viršija vidurkį, ar kurių kaina yra mažesnė už kainų vidurkį.

Paskutinis mygtukas sugrąžins į pagrindinį svetainės puslapį.

Visi skaičiavimai ir informacija bus išsaugota „**Results.txt**“ faile.

## **5.6. Programos tekstas**

### **Book.cs:**

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

namespace WebApplication5
{
    /// <summary>
    /// Book object class
    /// </summary>
    class Book
    {

        public string Author { get; set; }
        public string Title { get; set; }
        public DateTime Date { get; set; }
        public double Price { get; set; }
    }
}
```

```

public int CopiesSold { get; set; }
public int CopiesLeft { get; set; }

/// <summary>
/// Object's Book Constructor
/// </summary>
/// <param name="Author"> Author of the book </param>
/// <param name="Title"> Title of the book </param>
/// <param name="Date"> Date when the book was realised </param>
/// <param name="Price"> Price of the book </param>
/// <param name="CopiesSold"> How many copies where sold </param>
/// <param name="CopiesLeft"> How many copies are still left </param>
public Book(string Author, string Title, DateTime Date,
    double Price, int CopiesSold, int CopiesLeft)
{
    this.Author = Author;
    this.Title = Title;
    this.Date = Date;
    this.Price = Price;
    this.CopiesSold = CopiesSold;
    this.CopiesLeft = CopiesLeft;
}

/// <summary>
/// A ToString() Override method
/// </summary>
/// <returns> Returns a string used to form tables
/// in the result file </returns>
public override string ToString()
{
    return String.Format("| {0, -25} | {1, -25} |" +
        " {2, -14} | {3, -8} | {4," +
        " -10} | {5, -22} |\n", Author, Title, Date.ToShortDateString(),
        Price, CopiesSold, CopiesLeft);
}
}
}

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
-----
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

### Collection.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebApplication5
{
    /// <summary>
    /// Collection object class
    /// </summary>
    class Collection
    {
        public DateTime From { get; set; }
        public DateTime To { get; set; }
        public double PriceFrom { get; set; }
        public double PriceTo { get; set; }
    }
}

```

```

    /// <summary>
    /// Object' s Collection Constructor
    /// </summary>
    /// <param name="From"> Date from when we want
    /// to check </param>
    /// <param name="To"> Date until when we want
    /// to check </param>
    /// <param name="PriceFrom"> Price from which
    /// we want to check </param>
    /// <param name="PriceTo"> Price until whom we want to check </param>
    public Collection(DateTime From, DateTime To,
        double PriceFrom, double PriceTo)
    {
        this.From = From;
        this.To = To;
        this.PriceFrom = PriceFrom;
        this.PriceTo = PriceTo;
    }

    /// <summary>
    /// A ToString() Override method
    /// </summary>
    /// <returns> Returns a string used to form tables in
    /// the result file </returns>
    public override string ToString()
    {
        return String.Format("Data nuo: {0} || Data iki:" +
            " {1} || Kaina nuo:" +
            " {2, -10} || Kaina iki {3, -10}\n",
            From.ToShortDateString(),
            To.ToShortDateString(), PriceFrom, PriceTo);
    }
}
}

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

### Register.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace WebApplication5
{
    /// <summary>
    /// A Register that cointains all the information about one store
    /// </summary>
    class Register
    {
        private List<Book> Container;
        public string StoreName { get; set; }
        public double PriceVid = 0;

        /// <summary>
        /// Register's constructor
        /// </summary>
        /// <param name="Container"> A list containing all the

```

```

/// information about all the books </param>
/// <param name="StoreName"> The name of the store </param>
public Register(List<Book> Container, string StoreName)
{
    this.Container = Container;
    this.StoreName = StoreName;
}

/// <summary>
/// Returns a Book object with the same index as the given one
/// </summary>
/// <param name="index"> Index of the Book
/// object we want to get </param>
/// <returns> A Book object with the same
/// index as the given one </returns>
public Book GetByIndex(int index)
{
    return this.Container[index];
}

/// <summary>
/// Adds a Book object to the Register
/// </summary>
/// <param name="book"> The Book object we want to add </param>
public void Add(Book book)
{
    this.Container.Add(book);
}

/// <summary>
/// Counts for many Book objects are in the register
/// </summary>
/// <returns> A number, telling how many Book
/// object are in the register </returns>
public int Count()
{
    return this.Container.Count();
}

/// <summary>
/// Sorts all the Book objects by their title first
/// and then by the amount of Copies sold
/// </summary>
public void Sort()
{
    List<Book> Sort = Container
        .OrderBy(nn => nn.Title)
        .ThenBy(nn => nn.CopiesSold)
        .ToList<Book>();
    Container = Sort;
}

/// <summary>
/// Makes a new Register with the Books that fit
/// the requirements given by the Collection object
/// </summary>
/// <param name="col"> A Collection object </param>
/// <returns> A register with the Books that fit
/// the requirements given by the Collection object </returns>
public Register FitsInTheCollection(Collection col)
{
    List<Book> Sorted = Container.Where(nn =>
        ((nn.Date >= col.From &&
        nn.Date <= col.To) &&
        (nn.Price >= col.PriceFrom && nn.Price <= col.PriceTo)))

```

```

        .ToList<Book>());
    return new Register(Sorted, this.StoreName);
}

/// <summary>
/// Adds average values to the book prices
/// </summary>
public void AddVidPrice()
{
    double pricerange = 0;

    foreach(Book book in Container)
    {
        pricerange += book.Price;
    }
    this.PriceVid = pricerange / Count();
}

/// <summary>
/// Makes a new Register with book prices
/// being either bigger then average or lower
/// </summary>
/// <param name="Type"> A parametres that tells us if we want to get
/// Book object with prices bigger then average or lower </param>
/// <returns> a new Register with book prices
/// being either bigger then average or lower </returns>
public Register FitsInVidPriceRange(string Type)
{
    List<Book> Sorted = Container.Where(nn => (Type.ToLower() ==
        "didesnès" &&
        nn.Price >= this.PriceVid || Type.ToLower() ==
        "mažesnès" && nn.Price <= this.PriceVid))
        .ToList<Book>();
    return new Register(Sorted, this.StoreName);
}
}
}

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

### InOut.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;

namespace WebApplication5
{
    /// <summary>
    /// Class used to read files and print data inside files.
    /// </summary>
    static class InOut
    {
        /// <summary>
        /// Reads all files filled with store's and their books information
        /// </summary>
        /// <param name="Path"> Data file from where the

```



```

/// information will be read from </param>
/// <param name="FileName"> A file that we should not read,
/// since it contains different information </param>
/// <param name="ResultFile"> A result file where we
/// will print all the errors that might happen when reading the file </param>
/// <returns> Returns a Register list List<Register> filled with
/// all books and stores information </returns>
public static List<Register> ReadFile(string Path, string FileName, string ResultFile)
{
    string Punc = ",/";
    List<Register> AllStores = new List<Register>();
    string[] Files = Directory.GetFiles(Path);
    foreach (string file in Files)
    {
        if (FileName == file)
        {
            continue;
        }

        List<Book> Books = new List<Book>();
        string[] Lines = File.ReadAllLines(file, Encoding.ASCII);
        string StoreName = Lines[0];

        for (int i = 1; i < Lines.Count(); i++)
        {
            try
            {
                string[] Words = Lines[i].Split(Punc.ToCharArray(),
                    StringSplitOptions.RemoveEmptyEntries);
                string Author = Words[0];
                string Title = Words[1];
                DateTime Date = new DateTime(Convert.ToInt32(Words[2]),
                    Convert.ToInt32(Words[3]), Convert.ToInt32(Words[4]));
                double Price = Double.Parse(Words[5]);
                int CopiesSold = int.Parse(Words[6]);
                int CopiesLeft = int.Parse(Words[7]);
                Books.Add(new Book(Author, Title,
                    Date, Price, CopiesSold, CopiesLeft));
            }
            catch (IndexOutOfRangeException)
            {
                File.AppendAllText(ResultFile,
                    String.Format("Susidūrėme su problema skaitant Failo:" +
                        " {2} {0} eilutę.\nEilutė: {1}\nDėl sukeltų problemų" +
                        " teko šią eilutę panaikinti.\n\n", i, Lines[i], file));
                DeleteLine(file, Lines[i]);
                continue;
            }
            catch (FormatException)
            {
                File.AppendAllText(ResultFile,
                    String.Format("Susidūrėme su problema skaitant Failo:" +
                        " {2} {0} eilutę.\nEilutė: {1}\nDėl sukeltų problemų teko šią" +
                        " eilutę panaikinti.\n\n", i, Lines[i], file));
                DeleteLine(file, Lines[i]);
                continue;
            }
        }

        AllStores.Add(new Register(Books, StoreName));
    }
}

```

```

        return AllStores;
    }

    /// <summary>
    /// Reads file containing all the collections and their criteria
    /// </summary>
    /// <param name="FilePath"> File name that contains all the collections </param>
    /// <param name="ResultFile"> A result file where we will print all
    /// the errors that might happen when reading the file </param>
    /// <returns> Returns a Collection object list(List<Collection>) </returns>
    public static List<Collection> ReadCollections(string FilePath, string ResultFile)
    {
        string Punc = ",/";
        List<Collection> AllCollections = new List<Collection>();
        string[] Lines = File.ReadAllLines(FilePath, Encoding.ASCII);
        for (int i = 0; i < Lines.Count(); i++)
        {
            string line = Lines[i];
            try
            {
                string[] Words = line.Split(Punc.ToCharArray(),
                    StringSplitOptions.RemoveEmptyEntries);
                DateTime From = new DateTime(Convert.ToInt32(Words[0]),
                    Convert.ToInt32(Words[1]), Convert.ToInt32(Words[2]));
                DateTime To = new DateTime(Convert.ToInt32(Words[3]),
                    Convert.ToInt32(Words[4]), Convert.ToInt32(Words[5]));
                double PriceFrom = Double.Parse(Words[6]);
                double PriceTo = Double.Parse(Words[7]);
                AllCollections.Add(new Collection(From, To, PriceFrom, PriceTo));
            }
            catch (IndexOutOfRangeException)
            {
                File.AppendAllText(ResultFile,
                    String.Format("Susidūrėme su problema skaitant Failo:" +
                        " {2} {0} eilutę.\nEilutė: {1}\nDėl sukeltų problemų teko" +
                        " šią eilutę panaikinti.\n\n", i, Lines[i], FilePath));
                DeleteLine(FilePath, Lines[i]);
                continue;
            }
            catch (FormatException)
            {
                File.AppendAllText(ResultFile,
                    String.Format("Susidūrėme su problema skaitant" +
                        " Failo: {2} {0} eilutę.\nEilutė:" +
                        " {1}\nDėl sukeltų problemų teko šią eilutę" +
                        " panaikinti.\n\n", i, Lines[i], FilePath));
                DeleteLine(FilePath, Lines[i]);
                continue;
            }
        }
        return AllCollections;
    }

    /// <summary>
    /// Prints all data with the same type as the
    /// initial data that was read from
    /// the file (data in List<Register>)
    /// </summary>
    /// <param name="FilePath"> File where the data will
    /// be printed in </param>
    /// <param name="Header"> A header used for the

```

```

/// discription of the information printed </param>
/// <param name="Stores"> A Register object list that
/// has the information we need to print </param>
public static void PrintInitialData(string FilePath,
    string Header, List<Register> Stores)
{
    File.AppendAllText(FilePath, String.Format("{0}\n", Header));
    foreach(Register Reg in Stores)
    {
        File.AppendAllText(FilePath, String.Format("{0}:\n", Reg.StoreName));
        File.AppendAllText(FilePath, String.Format("|" +
            " {0, -25} | {1, -25} | {2, -14} |" +
            " {3, -8} | {4, -10} | {5, -22} |\n",
            "Autorius", "Knygos pavadinimas",
            "Leidimo data", "Kaina", "Parduota", "Sandelyje liko"));
        for (int i = 0; i < Reg.Count(); i++)
        {
            File.AppendAllText(FilePath,
                String.Format(Reg.GetByIndex(i).ToString()));
        }
        File.AppendAllText(FilePath, String.Format("\n"));
    }
}

/// <summary>
/// Prints A List Containing a Register object list (List<List<Register>>)
/// </summary>
/// <param name="FilePath"> File where the data
/// will be printed in </param>
/// <param name="Header"> A header used for the
/// discription of the information printed </param>
/// <param name="Stores"> A list containing a
/// register object list, that holds the data that
/// we need to print </param>
/// <param name="Collections"> List of all collections. Used to a
/// collection, so it would be easy to tell how the Registers are sorted </param>
public static void PrintInitialData(string FilePath, string Header,
    List<List<Register>> Stores, List<Collection> Collections)
{
    File.AppendAllText(FilePath, String.Format("{0}\n", Header));
    for (int i = 0; i < Stores.Count(); i++)
    {
        Collection col = Collections[i];
        File.AppendAllText(FilePath,
            String.Format("Kriterijai:" + col.ToString()));
        foreach (Register Reg in Stores[i])
        {
            File.AppendAllText(FilePath,
                String.Format("{0}:\n", Reg.StoreName));
            File.AppendAllText(FilePath,
                String.Format("| {0, -25} | {1, -25} | {2, -14} |" +
                    " {3, -8} | {4, -10} | {5, -22} |\n",
                    "Autorius", "Knygos pavadinimas",
                    "Leidimo data", "Kaina", "Parduota", "Sandelyje liko"));
            for (int m = 0; m < Reg.Count(); m++)
            {
                File.AppendAllText(FilePath,
                    String.Format(Reg.GetByIndex(m).ToString()));
            }
            File.AppendAllText(FilePath, String.Format("\n"));
        }
    }
}

```

```

}

/// <summary>
/// Deletes a wanted line inside a file
/// </summary>
/// <param name="FilePath"> File from where the
/// line has to be deleted from </param>
/// <param name="line"> String object containing
/// the line that has to be deleted </param>
private static void DeleteLine(string FilePath, string line)
{
    string[] Lines = File.ReadAllLines(FilePath, Encoding.ASCII);
    string[] NewLines = Lines.Where(nn =>
nn != line).ToArray<string>();
    File.WriteAllLines(FilePath, NewLines);
}
}
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
-----
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

### TaskUtils.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Threading.Tasks;

namespace WebApplication5
{
    static class TaskUtils
    {
        /// <summary>
        /// Makes A List of List<Register> object
        /// </summary>
        /// <param name="Stores"> A List<Register> object
        /// containing all the information
        /// about every store and their books </param>
        /// <param name="Collections"> A Collection object </param>
        /// <returns> A list of Register Lists that filled with the Books that fit
        /// the requirements given by the Collection object </returns>
        public static List<List<Register>> MakeListByCollections(List<Register> Stores,
            List<Collection> Collections)
        {
            List<List<Register>> ListOfRegisterLists = new List<List<Register>>();

            foreach(Collection Col in Collections)
            {
                List<Register> RegList = new List<Register>();
                foreach (Register reg in Stores)
                {
                    Register sorted = reg.FitsInTheCollection(Col);
                    sorted.Sort();
                    RegList.Add(sorted);
                }
                ListOfRegisterLists.Add(RegList);
            }
        }
    }
}

```

```

    }
    return ListOfRegisterLists;
}

/// <summary>
/// Adds average values to the book prices
/// </summary>
/// <param name="Stores"> To what Register List the
/// prices will be added </param>
public static void AddPriceVid(List<Register> Stores)
{
    foreach(Register reg in Stores)
    {
        reg.AddVidPrice();
    }
}

/// <summary>
/// Makes a List<Register> object filled with all the
/// stores and their books that have been sorted by price
/// </summary>
/// <param name="Stores"> A List<Register> object that
/// contains store data </param>
/// <param name="Type"> A parametres that tells us if we want to get
/// Book object with prices bigger then average or lower </param>
/// <returns> A List<Register> object filled with all the
/// stores and their books that have been sorted by price </returns>
public static List<Register> PriceSortedListMake(List<Register> Stores,
    string Type)
{
    List<Register> Reg = new List<Register>();
    foreach (Register reg in Stores)
    {
        Register sorted = reg.FitsInVidPriceRange(Type);
        sorted.Sort();
        Reg.Add(sorted);
    }
    return Reg;
}
}
}

```

```

////////////////////////////////////
-----
////////////////////////////////////

```

### Web.aspx.cs:

```

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

/// <summary>
/// Web class
/// </summary>
namespace WebApplication5
{
    public partial class Web : System.Web.UI.Page

```

```

{
    private string DirectoryPath;
    private string ResultPath;
    private string CollectionFilePath;
    private List<Register> Stores;
    private List<Register> SortedStores;
    private List<Collection> Collections;
    private List<List<Register>> StoresByCollectionData;
    private ListItem[] WordCol = { new ListItem("Didesnė kaina" +
        ", negu kainų vidurkis",
        "didesnės"), new ListItem("Mažesnė kaina, negu " +
        "kainų vidurkis", "mažesnės") };

    /// <summary>
    /// Loads page, and data files, and A droplist items
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Page_Load(object sender, EventArgs e)
    {
        DirectoryPath = Server.MapPath("App_Data");
        ResultPath = Server.MapPath("Rez/Results.txt");
        CollectionFilePath = Server.MapPath("App_Data/Collections.txt");
        if(!IsPostBack)
        {
            AddWordsToDropList(WordCol);
        }
    }

    /// <summary>
    /// Loads initial data into a table
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (File.Exists(ResultPath))
        {
            File.Delete(ResultPath);
        }

        Stores = InOut.ReadFile(DirectoryPath, CollectionFilePath, ResultPath);
        InOut.PrintInitialData(ResultPath, "Pradiniai duomenys:", Stores);
        FillTableWithInitialData(ref Table1, Stores);
        Button2.Visible = true;
        Button1.Visible = false;
        Label2.Text = "Pradiniai duomenys:";
    }

    /// <summary>
    /// Loads data sorted by collection file data
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    protected void Button2_Click(object sender, EventArgs e)
    {
        Stores = InOut.ReadFile(DirectoryPath, CollectionFilePath,
            ResultPath);
        Collections = InOut.ReadCollections(CollectionFilePath,
            ResultPath);
        StoresByCollectionData = TaskUtils.MakeListByCollections(Stores,
            Collections);
        InOut.PrintInitialData(ResultPath, "Su kriterijais:",
            StoresByCollectionData, Collections);
        FillTableWithCollectionFixedData(ref Table1,

```

```

        StoresByCollectionData, Collections);

    Button3.Visible = true;
    Button2.Visible = false;
    DropDownList1.Visible = true;
    Label2.Text = "Duomenys pagal kriterijus iš kriterijų failo:";
    Label3.Visible = true;
}

/// <summary>
/// Loads data sorted by price average
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void Button3_Click(object sender, EventArgs e)
{
    Stores = InOut.ReadFile(DirectoryPath, CollectionFilePath,
        ResultPath);
    TaskUtils.AddPriceVid(Stores);
    SortedStores = TaskUtils.PriceSortedListMake(Stores,
        DropDownList1.SelectedItem.Value.ToString());
    FillTableWithInitialData(ref Table1, SortedStores);
    InOut.PrintInitialData(ResultPath, "Atskirti pagal kainas, kurios yra " +
        DropDownList1.SelectedItem.Value.ToString() + " už vidurkį:" ,
        SortedStores);

    Button4.Visible = true;
    Button3.Visible = false;
    DropDownList1.Visible = false;
    Label2.Text = "Atskirti pagal kainas, kurios yra " +
        DropDownList1.SelectedItem.Value.ToString() + " už vidurkį:";
    Label3.Visible = false;
}

/// <summary>
/// Returns to home page
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void Button4_Click(object sender, EventArgs e)
{
    Button1.Visible = true;
    Button4.Visible = false;
    Label2.Text = "Paspauskyte mygtuką, norėdami paleisti programą";
}

/// <summary>
/// Fills table with a List<Register> objects
/// </summary>
/// <param name="table"> The table that will be filled </param>
/// <param name="Register"> A List<Register> object that
/// hold all the information about every store </param>
private static void FillTableWithInitialData(ref Table table,
    List<Register> Register)
{
    foreach (Register Reg in Register)
    {
        var HeaderRowStoreName = new TableRow();

        HeaderRowStoreName.Cells.Add(new TableCell()
        {
            Text = ("Knigyno pavadinimas: " +
                Reg.StoreName.ToString()),
            CssClass = "bold-text"
        })
    }
}

```

```

});
table.Rows.Add(HeaderRowStoreName);

var Header = new TableRow();

Header.Cells.Add(new TableCell()
{
    Text = "Autorius",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Pavadinimas",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Leidimo metai",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Kaina",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Parduota",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Sandelyje liko",
    CssClass = "bold-text"
});

table.Rows.Add(Header);

for (int i = 0; i < Reg.Count(); i++)
{
    var DataRow = new TableRow();
    Book book = Reg.GetByIndex(i);
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Author

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Title

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Date.ToShortDateString()

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Price.ToString()

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.CopiesSold.ToString()
    }

```



```

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.CopiesLeft.ToString()
    });
    table.Rows.Add(DataRow);
}

}

}

/// <summary>
/// Fills the table with List<List<Register>> object's data
/// </summary>
/// <param name="table"> The table that will be filled </param>
/// <param name="RegisterList"> List<List<Register>> object that
/// hold the data of every store sorted by every collection </param>
/// <param name="Collections"> A collection object list used to write
/// a text so that it would be easy to know how the stores are sorted </param>
private static void FillTableWithCollectionFixedData(ref Table table,
    List<List<Register>> RegisterList, List<Collection> Collections)
{
    for (int m = 0; m < RegisterList.Count; m++)
    {
        List<Register> Register = RegisterList[m];
        var CollectionHeader = new TableRow();
        Collection Col = Collections[m];

        CollectionHeader.Cells.Add(new TableCell()
        {
            Text = "Kriterijai:",
            CssClass = "bold-text"
        });
        CollectionHeader.Cells.Add(new TableCell()
        {
            Text = ("Nuo: " + Col.From.ToShortDateString()),
            CssClass = "bold-text"
        });
        CollectionHeader.Cells.Add(new TableCell()
        {
            Text = ("Iki: " + Col.To.ToShortDateString()),
            CssClass = "bold-text"
        });
        CollectionHeader.Cells.Add(new TableCell()
        {
            Text = ("Kaina nuo: " + Col.PriceFrom.ToString()),
            CssClass = "bold-text"
        });
        CollectionHeader.Cells.Add(new TableCell()
        {
            Text = ("Kaina iki: " + Col.PriceTo.ToString()),
            CssClass = "bold-text"
        });
        table.Rows.Add(CollectionHeader);

        foreach (Register Reg in Register)
        {
            var HeaderRowStoreName = new TableRow();

            HeaderRowStoreName.Cells.Add(new TableCell()

```

```

    {
        Text = ("Knigyno pavadinimas:" +
            Reg.StoreName.ToString()),
        CssClass = "bold-text"
    });
table.Rows.Add(HeaderRowStoreName);

var Header = new TableRow();

Header.Cells.Add(new TableCell()
{
    Text = "Autorius",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Pavadinimas",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Leidimo metai",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Kaina",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Parduota",
    CssClass = "bold-text"
});
Header.Cells.Add(new TableCell()
{
    Text = "Sandelyje liko",
    CssClass = "bold-text"
});

table.Rows.Add(Header);

for (int i = 0; i < Reg.Count(); i++)
{
    var DataRow = new TableRow();
    Book book = Reg.GetByIndex(i);
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Author

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Title

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Date.ToShortDateString()

    });
    DataRow.Cells.Add(new TableCell()
    {
        Text = book.Price.ToString()
    }

```



```

        background-color: cornsilk;
        border-color: Black;
        border-style: Solid;
        border-width: 5px;
    }

    .auto-style6 {
        background-color: cornsilk;
    }




```

## 5.7. Pradiniai duomenys ir rezultatai

### Testas Nr. 1:

**Tikslas:** Standartinis atvejis, kai apskaičiuojami keli atsakymai.

Directory „App\_Data“:

 Collections	2022-05-15 17:40	Text Document	1 KB
 Data	2022-05-15 17:40	Text Document	1 KB
 Data2	2022-05-15 17:40	Text Document	1 KB

Data.txt:

```

Lord is alive
Jonas,Katinai ir sunys,2002/10/01,12,157,15
Rogis,Bulve ir Rope,2004/08/01,15,26,478
Regutis,Agurkai ir tiek,2019/12/01,26,300,128

```

Data2.txt:

```

Gatve ir ko
Yakuza,Domenis,2020/02/15,7,178,19
Karzys,Durys ir laiptai,2004/08/01,45,29,158
Regutis,Agurkai ir tiek,2019/12/01,26,3,128

```

Collections.txt:

```

2002/05/10,2008/02/05,10,28
2008/05/14,2020/01/10,5,45

```

Results.txt:

```

Pradiniai duomenys:
Lord is alive:
| Autorius                | Knygos pavadinimas          | Leidimo data  | Kaina
| Parduota   | Sandelyje liko           |
| Jonas      | Katinai ir sunys        | 2002-10-01    | 12
| 157        | 15                       |
| Rogis      | Bulve ir Rope           | 2004-08-01    | 15
| 26         | 478                      |
| Regutis    | Agurkai ir tiek        | 2019-12-01    | 26
| 300        | 128                     |

```

Gatve ir ko:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Yakuza		Domenis		2020-02-15	7
178	19				
Karzys		Durys ir laiptai		2004-08-01	45
29	158				
Regutis		Agurkai ir tiek		2019-12-01	26
3	128				

Su kriterijais:

Kriterijai:Data nuo: 2002-05-10 || Data iki: 2008-02-05 || Kaina nuo: 10  
|| Kaina iki 28

Lord is alive:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Rogis		Bulve ir Rope		2004-08-01	15
26	478				
Jonas		Katinai ir sunys		2002-10-01	12
157	15				

Gatve ir ko:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				

Kriterijai:Data nuo: 2008-05-14 || Data iki: 2020-01-10 || Kaina nuo: 5  
|| Kaina iki 45

Lord is alive:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Regutis		Agurkai ir tiek		2019-12-01	26
300	128				

Gatve ir ko:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Regutis		Agurkai ir tiek		2019-12-01	26
3	128				

Atskirti pagal kainas, kurios yra mažesnės už vidurkį:

Lord is alive:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Rogis		Bulve ir Rope		2004-08-01	15
26	478				
Jonas		Katinai ir sunys		2002-10-01	12
157	15				

Gatve ir ko:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Regutis		Agurkai ir tiek		2019-12-01	26
3	128				
Yakuza		Domenis		2020-02-15	7
178	19				

## "Pegasas" knygynų knygų tvarkyklė

Paspauskite mygtuką, norėdami paleisti programą

Load Data

## "Pegasas" knygynų knygų tvarkyklė

Pradiniai duomenys:

Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Jonas	Katinai ir sunys	2002-10-01	12	157	15
Rogis	Bulve ir Rope	2004-08-01	15	26	478
Regutis	Agurkai ir tiek	2019-12-01	26	300	128
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Yakuza	Domenis	2020-02-15	7	178	19
Karzys	Durys ir laiptai	2004-08-01	45	29	158
Regutis	Agurkai ir tiek	2019-12-01	26	3	128

Sort By Collections

## "Pegasas" knygynų knygų tvarkyklė

Duomenys pagal kriterijus iš kriterijų failo:

Kriterijai:	Nuo: 2002-05-10	Iki: 2008-02-05	Kaina nuo: 10	Kaina iki: 28	
Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Rogis	Bulve ir Rope	2004-08-01	15	26	478
Jonas	Katinai ir sunys	2002-10-01	12	157	15
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Kriterijai:	Nuo: 2008-05-14	Iki: 2020-01-10	Kaina nuo: 5	Kaina iki: 45	
Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Regutis	Agurkai ir tiek	2019-12-01	26	300	128
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Regutis	Agurkai ir tiek	2019-12-01	26	3	128

Pasirinkite norimą rūšiavimo kriterijų: Didesnė kaina, negu kainų vidurkis ▼ Sort by price

## "Pegasas" knygynų knygų tvarkyklė

Atskirti pagal kainas, kurios yra mažesnės už vidurkį:




Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Rogis	Bulve ir Rope	2004-08-01	15	26	478
Jonas	Katinai ir sunys	2002-10-01	12	157	15
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Regutis	Agurkai ir tiek	2019-12-01	26	3	128
Yakuza	Domenis	2020-02-15	7	178	19

[Return to front page](#)

### Testas Nr. 2:

**Tikslas:** Parodyti kaip programa veikia su exceptions(Klaidingais duomenų failais).

Directory „App\_Data“:

 Collections	2022-05-15 17:40	Text Document	1 KB
 Data	2022-05-15 17:40	Text Document	1 KB
 Data2	2022-05-15 17:40	Text Document	1 KB

Data.txt:

```
Lord is alive
Jonas,Katinai ir sunys,2002/10/01,12,157,15
Rogis,Bulve ir Rope,2004/08/01,15,26,478
s/d,a/s.d,as/.f,s/fdn.dsm,/f.msd/f.smm/.sdm/
Regutis,Agurkai ir tiek,2019/12/01,26,300,128
```

Data2.txt:

```
Gatve ir ko
Yakuza,Domenis,2020/02/15,7,178,19
Karzys,Durys ir laiptai,2004/08/01,45,29,158
Regutis,Agurkai ir tiek,2019/12/01,26,3,128
```

Collections.txt:

```
2002/05/10,2008/02/05,10,28
2008/05/14,2020/01/10,5,45
sA?<s/ds,b,dBAS?dbd
db/Sd.abS?d.BA?S.dB
dbs/d.sb?D.sB?D.asBD
```

## Results.txt:

Susidūre su problema skaitant Failo:

C:\Users\Lenovo\source\repos\WebApplication5\WebApplication5\App\_Data\Data.txt 3 eilutę.

Eilutė: s/d,a/s.d,as/.f,s/fdn.dsm,/f.msd/f.smm/.sdm/

Dėl sukeltų problemų teko šią eilutę panaikinti.

Pradiniai duomenys:

Lord is alive:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Jonas		Katinai ir sunys		2002-10-01	12
157	15				
Rogis		Bulve ir Rope		2004-08-01	15
26	478				
Regutis		Agurkai ir tiek		2019-12-01	26
300	128				

Gatve ir ko:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Yakuza		Domenis		2020-02-15	7
178	19				
Karzys		Durys ir laiptai		2004-08-01	45
29	158				
Regutis		Agurkai ir tiek		2019-12-01	26
3	128				

Susidūre su problema skaitant Failo:

C:\Users\Lenovo\source\repos\WebApplication5\WebApplication5\App\_Data\Collection s.txt 2 eilutę.

Eilutė: sA?<s/ds,b,dBAS?dbd

Dėl sukeltų problemų teko šią eilutę panaikinti.

Susidūre su problema skaitant Failo:

C:\Users\Lenovo\source\repos\WebApplication5\WebApplication5\App\_Data\Collection s.txt 3 eilutę.

Eilutė: db/Sd.abS?d.BA?S.dB

Dėl sukeltų problemų teko šią eilutę panaikinti.

Susidūre su problema skaitant Failo:

C:\Users\Lenovo\source\repos\WebApplication5\WebApplication5\App\_Data\Collection s.txt 4 eilutę.

Eilutė: dbs/d.sb?D.sB?D.asBD

Dėl sukeltų problemų teko šią eilutę panaikinti.

Su kriterijais:

Kriterijai:Data nuo: 2002-05-10 || Data iki: 2008-02-05 || Kaina nuo: 10

|| Kaina iki 28

Lord is alive:

Autorius		Knygos pavadinimas		Leidimo data	Kaina
Parduota	Sandelyje liko				
Rogis		Bulve ir Rope		2004-08-01	15
26	478				



Jonas	Katinai ir sunys	2002-10-01	12
157	15		

Gatve ir ko:

Autorius	Knygos pavadinimas	Leidimo data	Kaina
Parduota	Sandelyje liko		

Kriterijai: Data nuo: 2008-05-14 || Data iki: 2020-01-10 || Kaina nuo: 5  
|| Kaina iki 45

Lord is alive:

Autorius	Knygos pavadinimas	Leidimo data	Kaina
Parduota	Sandelyje liko		
Regutis	Agurkai ir tiek	2019-12-01	26
300	128		

Gatve ir ko:

Autorius	Knygos pavadinimas	Leidimo data	Kaina
Parduota	Sandelyje liko		
Regutis	Agurkai ir tiek	2019-12-01	26
3	128		

Atskirti pagal kainas, kurios yra didesnės už vidurkį:

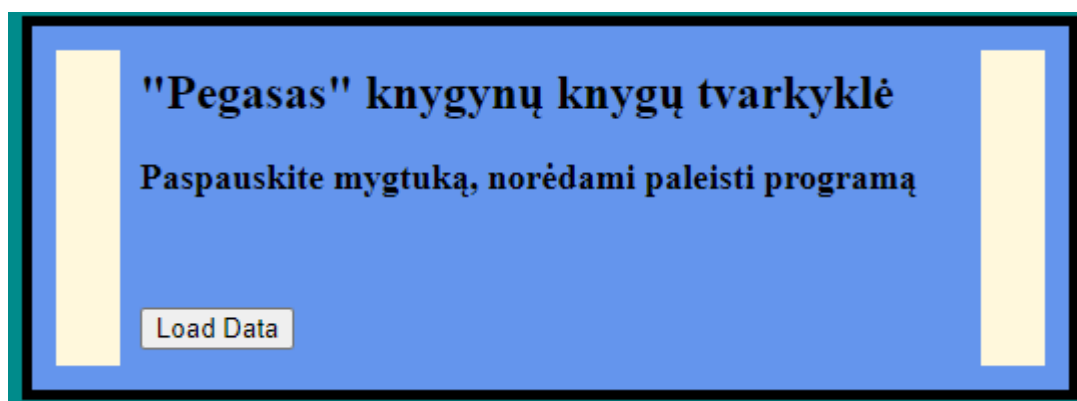
Lord is alive:

Autorius	Knygos pavadinimas	Leidimo data	Kaina
Parduota	Sandelyje liko		
Regutis	Agurkai ir tiek	2019-12-01	26
300	128		

Gatve ir ko:

Autorius	Knygos pavadinimas	Leidimo data	Kaina
Parduota	Sandelyje liko		
Regutis	Agurkai ir tiek	2019-12-01	26
3	128		
Karzys	Duryš ir laiptai	2004-08-01	45
29	158		

## User interface



## "Pegasas" knygynų knygų tvarkyklė

### Pradiniai duomenys:

Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Jonas	Katinai ir sunys	2002-10-01	12	157	15
Rogis	Bulve ir Rope	2004-08-01	15	26	478
Regutis	Agurkai ir tiek	2019-12-01	26	300	128
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Yakuza	Domenis	2020-02-15	7	178	19
Karzys	Durys ir laiptai	2004-08-01	45	29	158
Regutis	Agurkai ir tiek	2019-12-01	26	3	128

Sort By Collections

## "Pegasas" knygynų knygų tvarkyklė

### Duomenys pagal kriterijus iš kriterijų failo:

Kriterijai:	Nuo: 2002-05-10	Iki: 2008-02-05	Kaina nuo: 10	Kaina iki: 28	
Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Rogis	Bulve ir Rope	2004-08-01	15	26	478
Jonas	Katinai ir sunys	2002-10-01	12	157	15
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Knigyno pavadinimas: Lord is alive					
Kriterijai:	Nuo: 2008-05-14	Iki: 2020-01-10	Kaina nuo: 5	Kaina iki: 45	
Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Regutis	Agurkai ir tiek	2019-12-01	26	300	128
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Regutis	Agurkai ir tiek	2019-12-01	26	3	128

Pasirinkite norimą rūšiavimo kriterijų: Didesnė kaina, negu kainų vidurkis ▼ Sort by price

## "Pegasas" knygynų knygų tvarkyklė

### Atskirti pagal kainas, kurios yra mažesnės už vidurkį:

Knigyno pavadinimas: Lord is alive					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Rogis	Bulve ir Rope	2004-08-01	15	26	478
Jonas	Katinai ir sunys	2002-10-01	12	157	15
Knigyno pavadinimas: Gatve ir ko					
Autorius	Pavadinimas	Leidimo metai	Kaina	Parduota	Sandelyje liko
Regutis	Agurkai ir tiek	2019-12-01	26	3	128
Yakuza	Domenis	2020-02-15	7	178	19

Return to front page

## **5.8. Dėstytojo pastabos**