KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Objektinis programavimas II (P175B123)

Darbų aplankas

Atliko:

IFF- 6/8 gr. studentas

Tadas Laurinaitis

2017 m. birželio 13 d.

Priėmė:

Dėstytoja Dalia Čalnerytė

KAUNAS 2017

TURINYS

1. Rekursija (L1)

1.1. Darbo užduotis

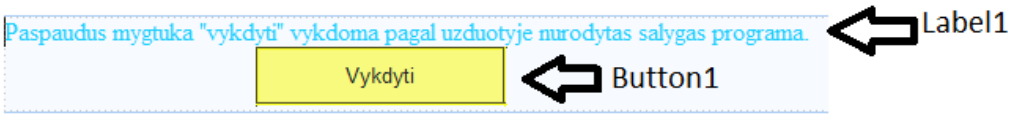
LD_24.Susitikimas.
Grupė draugų nusprendė susitikti mieste, o po to kartu eiti valgyti picos. Bet tuomet jie susiginčijo, kur geriausia susitikti, ir kurioje picerijoje valgyti. Galiausiai draugai nusprendė, jog patogiausia išsirinkti tokią susitikimo vietą ir piceriją, kad jų nueitų kelių iki susitikimo vietos, po to iki picerijos ir atgal į pradinius taškus suma būtų mažiausia. Parašykite programą, kuri rastų patogiausią susitikimo vietą ir piceriją.
Duomenys. Pirmoje *„U3.txt“* eilutėje duoti du skaičiai *n* ir *m* — miesto žemėlapio aukštis ir plotis ($2 \le n, m \le 10$). Toliau pateiktas miesto žemėlapis — *n* eilučių, kiekvienoje jų – *m* simbolių. Galimi šie simboliai:

`.` — langelis **pereinamas**.
`X` — langelis **nepereinamas** (pastatas, tvora...)
`P` — picerija. Langelis **nepereinamas**, į piceriją galima įeiti iš gretimo langelio ir išeiti į bet kurį gretimą langelį. Įeiti galima tik į tą piceriją, kurioje bus valgoma pica.
`S` — susitikimo vieta. Langelis **pereinamas**.
`D` — vieno iš draugų pradinė buvimo vieta. Langelis **pereinamas**.

Judėti galima tik aukštyn, žemyn, į kairę arba į dešinę (ne įstrižai).
Rezultatai. Ekrane atskirose eilutėse atspausdinkite pradines draugų koordinates (koordinačių pradžia – apatinis kairysis langelis, numeruojama nuo 1, pirmiausiai nurodoma x koordinatė, po to - y). Atskirose eilutėse atspausdinkite: „Susitikimo vieta“ ir susitikimo vietos koordinates; „Picerija“ ir picerijos koordinates; „Nueita“ ir draugų nueitų kelių sumą. Jei susitikimo vietos ar picerijos, kurias visi draugai galėtų pasiekti, nėra, atspausdinkite žodį „Neįmanoma“.

U3.txt	Ekranas
5 5 P.X.P ..XS. S.X.. ..X.D D.X..	1 1 5 2 Neįmanoma
5 5 D...P XX... D.S.P XX... D.S..	1 1 1 3 1 5 Susitikimo vieta 3 3 Picerija 5 3 Nueita 32

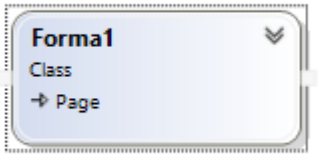
1.2. Grafinės vartotojo sąsajos schema



1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button1	BackgroundColour; BorderWidth;	Yellow; 2px;
Label1	BackgroundColour;TextColour	Purple; Light purple.

1.4. Klasių diagrama



1.5. Programos vartotojo vadovas

Programa apskaičiuoja artimiausią draugų susitikimo vietą ir artimiausią piceriją kurioje jie galetu pavalgyti.

1) Įkelti duomenų failą į pagrindinį programos aplanką.
2) Paleisti programą.
3) Paspausti mygtuką „Vykdyti“.
4) Peržiūrėti rezultatus, atsidarius failą „Rezultatai.txt“ esantį pagrindiniame programos aplanke.

1.6. Programos tekstas

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="Lab1Weeb.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" ForeColor="#00CCFF" Text="Paspaudus mygtuka
&amp;quot;vykdyti&amp;quot; vykdoma pagal uzduotyje nurodytas salygas programa."></asp:Label>
            <br />
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbs
p;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;n
bsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;
            <asp:Button ID="Button1" runat="server" BackColor="#FFFF66" BorderColor="Yellow"
BorderStyle="Groove" BorderWidth="2px" Height="37px" OnClick="Button1_Click" Text="Vykdyti"
Width="159px" />
        </div>
    </form>
</body>
</html>
```

Forma1.aspx.cs

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

namespace Lab1Weeb
{
    public partial class Forma1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            Main();
        }
```

```
        const int length = 5;

        void Main()
        {
            string file = @"D:\U3.txt";
            int x, y, xSus, ySus, xPic, yPic, totalLength = 0;
            int[] xDraugu, yDraugu, xSusitikimo, ySusitikimo, xPicerijos, yPicerijos = new
int[length];
            char[,] map = readMap(file, out x, out y, out xDraugu, out yDraugu, out
xSusitikimo, out  ySusitikimo, out xPicerijos, out yPicerijos);
            visuKeliones(map, xDraugu, yDraugu, xSusitikimo, ySusitikimo, xPicerijos,
yPicerijos, out xSus, out ySus, out xPic, out yPic, out totalLength);
            spausdinimas(map, xDraugu, yDraugu, xSus, ySus, xPic, yPic);
        }
        /// <summary>
        /// Vidu draugu keliones
        /// </summary>
        /// <param name="map"></param>
        /// <param name="xDraugu"></param>
        /// <param name="yDraugu"></param>
        /// <param name="xSusitikimo"></param>
        /// <param name="ySusitikimo"></param>
        /// <param name="xPicerijos"></param>
        /// <param name="yPicerijos"></param>
        /// <param name="xSus"></param>
        /// <param name="ySus"></param>
        /// <param name="xPic"></param>
        /// <param name="yPic"></param>
        /// <param name="totalLength"></param>
        void visuKeliones(char[,] map, int[] xDraugu, int[] yDraugu, int[] xSusitikimo, int[]
ySusitikimo, int[] xPicerijos, int[] yPicerijos, out int xSus, out int ySus, out int xPic, out
int yPic, out int totalLength)
        {
            int shortestPath = 999;
            int currentPath;
            totalLength = 0;
            xSus = 0;
            ySus = 0;
            xPic = 0;
            yPic = 0;
            bool changed = false;
            for (int i = 0; i < xSusitikimo.Length; i++)
            {
                currentPath = 0;
                for (int j = 0; j < xDraugu.Length; j++)
                {
                    if (draugoKelione(xDraugu[j], yDraugu[j], xSusitikimo[i], ySusitikimo[i],
out currentPath, map) == true)
                    {
                        currentPath += currentPath;
                    }
                    else
                        break;
                }
                if (currentPath < shortestPath && currentPath != 0)
                {
                    shortestPath = currentPath;
                    xSus = xSusitikimo[i];
                    ySus = ySusitikimo[i];
                    changed = true;
                }
            }
            totalLength = totalLength + shortestPath;
            int pizzaPath = 999;
            for (int i = 0; i < xPicerijos.Length; i++)
            {
                currentPath = 0;
                for (int j = 0; j < xDraugu.Length; j++)
```

```
                {
                    if (draugoKelione(xSus, ySus, xPicerijos[i], yPicerijos[i], out
currentPath, map) == true && changed == true)
                    {
                        currentPath += currentPath;
                    }
                }
                if (currentPath < pizzaPath && currentPath != 0)
                {
                    pizzaPath = currentPath;
                    xPic = xPicerijos[i];
                    yPic = yPicerijos[i];
                }

            }
            totalLength += pizzaPath;
        }
        /// <summary>
        /// Zemelapio nuskaitymo metodas
        /// </summary>
        /// <param name="file"></param>
        /// <param name="x"></param>
        /// <param name="y"></param>
        /// <param name="xDraugu"></param>
        /// <param name="yDraugu"></param>
        /// <param name="xSusitikimo"></param>
        /// <param name="ySusitikimo"></param>
        /// <param name="xPicerijos"></param>
        /// <param name="yPicerijos"></param>
        /// <returns></returns>
        char[,] readMap(string file, out int x, out int y, out int[] xDraugu, out int[]
yDraugu, out int[] xSusitikimo, out int[] ySusitikimo, out int[] xPicerijos, out int[]
yPicerijos)
        {
            using (StreamReader reader = new StreamReader(@file))
            {
                x = 0;
                y = 0;

                int count = 0;
                int count2 = 0;
                int count3 = 0;
                string line = reader.ReadLine();
                string[] koordinates = line.Split(' ');
                x = int.Parse(koordinates[0]);
                y = int.Parse(koordinates[1]);
                xDraugu = new int[x];
                yDraugu = new int[y];
                xSusitikimo = new int[x];
                ySusitikimo = new int[y];
                xPicerijos = new int[x];
                yPicerijos = new int[y];
                char[,] map = new char[x, y];
                for (int i = 0; i < y; i++)
                {
                    line = reader.ReadLine();
                    char[] ch = line.ToCharArray();
                    for (int j = 0; j < ch.Length; j++)
                    {
                        map[i, j] = ch[j];
                        if (ch[j] == 'D')
                        {
                            xDraugu[count] = j;
                            yDraugu[count] = i;
                            count++;
                        }
                        if (ch[j] == 'S')
                        {
```

```
                            xSusitikimo[count2] = j;
                            ySusitikimo[count2] = i;
                            count2++;
                        }
                        if (ch[j] == 'P')
                        {
                            xPicerijos[count3] = j;
                            yPicerijos[count3] = i;
                        }
                    }
                }
                return map;
            }
        }
        /// <summary>
        /// Vieno draugo kelione iki tikslo
        /// </summary>
        /// <param name="currentX"></param>
        /// <param name="currentY"></param>
        /// <param name="goalX"></param>
        /// <param name="goalY"></param>
        /// <param name="length"></param>
        /// <param name="map"></param>
        /// <returns></returns>
        bool draugoKelione(int currentX, int currentY, int goalX, int goalY, out int length,
char[,] map)
        {
            length = 0;

            //pasiekia susitikimo vieta-----------------------------------------------------
-------------------------------------------------------------------
            if (currentX == goalX && currentY == goalY)
            {
                return true;
            }

            //kai vienoje x linijoje kaip ir susitikimo vieta-------------------------------
-------------------------------------------------------------------
            else if (currentX == goalX)
            {
                if (currentY != goalY)
                {
                    if (currentY < goalY) //kai zemiau negu susitikimo vieta
                    {
                        if (map[currentY + 1, currentX] != 'X' && map[currentY + 1, currentX]
!= 'P' && map[currentY + 1, currentX] != 0) //eina i virsu
                        {
                            length++;
                            draugoKelione(currentX, currentY + 1, goalX, goalY, out length,
map);
                        }
                        else if (map[currentY + 1, currentX] == 'X' || map[currentY + 1,
currentX] == 'P' || map[currentY + 1, currentX] == 0)
                        {
                            if (map[currentY, currentX + 1] != 'X' && map[currentY, currentX +
1] != 'P' && map[currentY, currentX + 1] != 0)
                            {
                                length++;
                                draugoKelione(currentX + 1, currentY, goalX, goalY, out
length, map); //paeina i desine
                            }
                            else if (map[currentY, currentX + 1] == 'X' || map[currentY,
currentX + 1] == 'P' || map[currentY, currentX + 1] == 0)
                            {
                                if (map[currentY, currentX - 1] != 'X' && map[currentY,
currentX - 1] != 'P' && map[currentY, currentX - 1] != 0)
                                {
                                    length++;
```

```
                                draugoKelione(currentX - 1, currentY, goalX, goalY, out
length, map); //paeina i kaire
                            }
                            else if (map[currentY, currentX - 1] == 'X' && map[currentY,
currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                            {
                                if (map[currentY - 1, currentX] != 'X' && map[currentY,
currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                                {
                                    length++;
                                    draugoKelione(currentX, currentY - 1, goalX, goalY,
out length, map); //eina i apacia
                                }
                                else
                                    return false; //akligatvis
                            }
                        }
                    }
                    if (currentY > goalY) //kai auksciau negu susitikimo vieta
                    {
                        if (map[currentY - 1, currentX] != 'X' && map[currentY - 1, currentX]
!= 'P' && map[currentY - 1, currentX] != 0)
                        {
                            length++;
                            draugoKelione(currentX, currentY - 1, goalX, goalY, out length,
map); //eina i apacia
                        }
                        else if (map[currentY - 1, currentX] == 'X' || map[currentY - 1,
currentX] == 'P' || map[currentY - 1, currentX] == 0)
                        {
                            if (map[currentY, currentX + 1] != 'X' && map[currentY, currentX +
1] != 'P' && map[currentY, currentX + 1] != 0)
                            {
                                length++;
                                draugoKelione(currentX + 1, currentY, goalX, goalY, out
length, map); //paeina i desine
                            }
                            else if (map[currentY, currentX + 1] == 'X' || map[currentY,
currentX + 1] == 'P' || map[currentY, currentX + 1] == 0)
                            {
                                if (map[currentY, currentX - 1] != 'X' && map[currentY,
currentX - 1] != 'P' && map[currentY, currentX - 1] != 0)
                                {
                                    length++;
                                    draugoKelione(currentX - 1, currentY, goalX, goalY, out
length, map); //paeina i kaire
                                }
                                else if (map[currentY, currentX - 1] == 'X' && map[currentY,
currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                                {
                                    if (map[currentY + 1, currentX] != 'X' && map[currentY +
1, currentX] == 'P' && map[currentY + 1, currentX] == 0)
                                    {
                                        length++;
                                        draugoKelione(currentX, currentY - 1, goalX, goalY,
out length, map); //eina i virsu
                                    }
                                    else
                                        return false; //akligatvis
                                }
                            }
                        }
                    }
                }
            }
```

```
            //kai vienoje y linijoje kaip ir susitikimo vieta--------------------------------
------------------------------------------------------------------------
            else if (currentY == goalY)
            {
                if (currentX != goalX) //nebutinas
                {
                    if (currentX < goalX) //kai arciau negu susitikimo vieta
                    {
                        if (map[currentY, currentX + 1] != 'X' && map[currentY, currentX + 1]
!= 'P' && map[currentY, currentX + 1] != 0) //eina i virsu
                        {
                            length++;
                            draugoKelione(currentX + 1, currentY, goalX, goalY, out length,
map); //eina i desine
                        }
                        else if (map[currentY, currentX + 1] == 'X' && map[currentY, currentX
+ 1] == 'P' && map[currentY, currentX + 1] == 0)
                        {
                            if (map[currentY - 1, currentX] != 'X' && map[currentY - 1,
currentX] != 'P' && map[currentY - 1, currentX] != 0)
                            {
                                length++;
                                draugoKelione(currentX - 1, currentY, goalX, goalY, out
length, map); //paeina i apacia
                            }
                            else if (map[currentY - 1, currentX] == 'X' || map[currentY - 1,
currentX] == 'P' || map[currentY - 1, currentX] == 0)
                            {
                                if (map[currentY + 1, currentX] != 'X' && map[currentY + 1,
currentX] != 'P' && map[currentY + 1, currentX] != 0)
                                {
                                    length++;
                                    draugoKelione(currentX, currentY + 1, goalX, goalY, out
length, map); //paeina i virsu
                                }
                                else if (map[currentY + 1, currentX] == 'X' && map[currentY +
1, currentX] == 'P' && map[currentY + 1, currentX] == 0)
                                {
                                    if (map[currentY, currentX - 1] != 'X' && map[currentY,
currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                                    {
                                        length++;
                                        draugoKelione(currentX, currentY - 1, goalX, goalY,
out length, map); //eina i kaire
                                    }
                                    else
                                        return false; //akligatvis
                                }
                            }
                        }
                    }
                    if (currentX > goalX) //kai toliau negu susitikimo vieta
                    {
                        if (map[currentY, currentX - 1] != 'X' && map[currentY, currentX - 1]
!= 'P' && map[currentY, currentX] != 0)
                        {
                            length++;
                            draugoKelione(currentX - 1, currentY, goalX, goalY, out length,
map); //eina i kaire
                        }
                        else if (map[currentY, currentX - 1] == 'X' || map[currentY, currentX
- 1] == 'P' || map[currentY, currentX - 1] == 0)
                        {
                            if (map[currentY + 1, currentX] != 'X' && map[currentY + 1,
currentX] != 'P' && map[currentY + 1, currentX] != 0)
                            {
                                length++;
```

```
                                draugoKelione(currentX, currentY + 1, goalX, goalY, out
length, map); //paeina i virsu
                            }
                            else if (map[currentY + 1, currentX] == 'X' || map[currentY + 1,
currentX] == 'P' || map[currentY + 1, currentX] == 0)
                            {
                                if (map[currentY - 1, currentX] != 'X' && map[currentY - 1,
currentX] != 'P' && map[currentY - 1, currentX] != 0)
                                {
                                    length++;
                                    draugoKelione(currentX, currentY - 1, goalX, goalY, out
length, map); //eina i apacia
                                }
                                else if (map[currentY - 1, currentX] == 'X' && map[currentY -
1, currentX] == 'P' && map[currentY - 1, currentX] == 0)
                                {
                                    if (map[currentY, currentX + 1] != 'X' && map[currentY,
currentX + 1] == 'P' && map[currentY, currentX + 1] == 0)
                                    {
                                        length++;
                                        draugoKelione(currentX + 1, currentY, goalX, goalY,
out length, map); //eina i virsu
                                    }
                                    else
                                        return false; //akligatvis
                                }
                            }
                        }
                    }
                }
            }

            //kai nei y nei x nelygus susitikimo vietai-----------------------------------
-------------------------------------------------------------------
            else if (currentX != goalX && currentY != goalY)
            {
                if (Math.Abs(currentX - goalX) >= Math.Abs(currentY - goalY))
                {
                    if ((currentX - goalX) > 0)
                    {
                        if (map[currentY, currentX - 1] != 'X' && map[currentY, currentX - 1]
!= 'P' && map[currentY, currentX] != 0)
                        {
                            length++;
                            draugoKelione(currentX - 1, currentY, goalX, goalY, out length,
map); //eina i kaire
                        }
                        else if (map[currentY, currentX - 1] == 'X' || map[currentY, currentX
- 1] == 'P' || map[currentY, currentX - 1] == 0)
                        {
                            if (map[currentY + 1, currentX] != 'X' && map[currentY + 1,
currentX] != 'P' && map[currentY + 1, currentX] != 0)
                            {
                                length++;
                                draugoKelione(currentX, currentY + 1, goalX, goalY, out
length, map); //paeina i virsu
                            }
                            else if (map[currentY + 1, currentX] == 'X' || map[currentY + 1,
currentX] == 'P' || map[currentY + 1, currentX] == 0)
                            {
                                if (map[currentY - 1, currentX] != 'X' && map[currentY - 1,
currentX] != 'P' && map[currentY - 1, currentX] != 0)
                                {
                                    length++;
                                    draugoKelione(currentX, currentY - 1, goalX, goalY, out
length, map); //eina i apacia
                                }
```

```
                                else if (map[currentY - 1, currentX] == 'X' && map[currentY -
1, currentX] == 'P' && map[currentY - 1, currentX] == 0)
                                {
                                    if (map[currentY, currentX + 1] != 'X' && map[currentY,
currentX + 1] == 'P' && map[currentY, currentX + 1] == 0)
                                    {
                                        length++;
                                        draugoKelione(currentX + 1, currentY, goalX, goalY,
out length, map); //eina i virsu
                                    }
                                    else
                                        return false; //akligatvis
                                }
                            }
                        }
                    }
                    else
                    {
                        if (map[currentY, currentX + 1] != 'X' && map[currentY, currentX + 1]
!= 'P' && map[currentY, currentX + 1] != 0) //eina i virsu
                        {
                            length++;
                            draugoKelione(currentX + 1, currentY, goalX, goalY, out length,
map); //eina i desine
                        }
                        else if (map[currentY, currentX + 1] == 'X' && map[currentY, currentX
+ 1] == 'P' && map[currentY, currentX + 1] == 0)
                        {
                            if (map[currentY - 1, currentX] != 'X' && map[currentY - 1,
currentX] != 'P' && map[currentY - 1, currentX] != 0)
                            {
                                length++;
                                draugoKelione(currentX - 1, currentY, goalX, goalY, out
length, map); //paeina i apacia
                            }
                            else if (map[currentY - 1, currentX] == 'X' || map[currentY - 1,
currentX] == 'P' || map[currentY - 1, currentX] == 0)
                            {
                                if (map[currentY + 1, currentX] != 'X' && map[currentY + 1,
currentX] != 'P' && map[currentY + 1, currentX] != 0)
                                {
                                    length++;
                                    draugoKelione(currentX, currentY + 1, goalX, goalY, out
length, map); //paeina i virsu
                                }
                                else if (map[currentY + 1, currentX] == 'X' && map[currentY +
1, currentX] == 'P' && map[currentY + 1, currentX] == 0)
                                {
                                    if (map[currentY, currentX - 1] != 'X' && map[currentY,
currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                                    {
                                        length++;
                                        draugoKelione(currentX, currentY - 1, goalX, goalY,
out length, map); //eina i kaire
                                    }
                                    else
                                        return false; //akligatvis
                                }
                            }
                        }
                    }
                }
                else
                {
                    if ((currentY - goalY) > 0)
                    {
                        {
```

```
                            if (map[currentY - 1, currentX] != 'X' && map[currentY - 1,
currentX] != 'P' && map[currentY - 1, currentX] != 0)
                            {
                                length++;
                                draugoKelione(currentX, currentY - 1, goalX, goalY, out
length, map); //eina i apacia
                            }
                            else if (map[currentY - 1, currentX] == 'X' || map[currentY - 1,
currentX] == 'P' || map[currentY - 1, currentX] == 0)
                            {
                                if (map[currentY, currentX + 1] != 'X' && map[currentY,
currentX + 1] != 'P' && map[currentY, currentX + 1] != 0)
                                {
                                    length++;
                                    draugoKelione(currentX + 1, currentY, goalX, goalY, out
length, map); //paeina i desine
                                }
                                else if (map[currentY, currentX + 1] == 'X' || map[currentY,
currentX + 1] == 'P' || map[currentY, currentX + 1] == 0)
                                {
                                    if (map[currentY, currentX - 1] != 'X' && map[currentY,
currentX - 1] != 'P' && map[currentY, currentX - 1] != 0)
                                    {
                                        length++;
                                        draugoKelione(currentX - 1, currentY, goalX, goalY,
out length, map); //paeina i kaire
                                    }
                                    else if (map[currentY, currentX - 1] == 'X' &&
map[currentY, currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                                    {
                                        if (map[currentY + 1, currentX] != 'X' && map[currentY
+ 1, currentX] == 'P' && map[currentY + 1, currentX] == 0)
                                        {
                                            length++;
                                            draugoKelione(currentX, currentY - 1, goalX,
goalY, out length, map); //eina i virsu
                                        }
                                        else
                                            return false; //akligatvis
                                    }
                                }
                            }
                        }
                        else
                        {
                            if (map[currentY + 1, currentX] != 'X' && map[currentY + 1, currentX]
!= 'P' && map[currentY + 1, currentX] != 0) //eina i virsu
                            {
                                length++;
                                draugoKelione(currentX, currentY + 1, goalX, goalY, out length,
map);
                            }
                            else if (map[currentY + 1, currentX] == 'X' || map[currentY + 1,
currentX] == 'P' || map[currentY + 1, currentX] == 0)
                            {
                                if (map[currentY, currentX + 1] != 'X' && map[currentY, currentX +
1] != 'P' && map[currentY, currentX + 1] != 0)
                                {
                                    length++;
                                    draugoKelione(currentX + 1, currentY, goalX, goalY, out
length, map); //paeina i desine
                                }
                                else if (map[currentY, currentX + 1] == 'X' || map[currentY,
currentX + 1] == 'P' || map[currentY, currentX + 1] == 0)
                                {
                                    if (map[currentY, currentX - 1] != 'X' && map[currentY,
currentX - 1] != 'P' && map[currentY, currentX - 1] != 0)
```

```
                                {
                                    length++;
                                    draugoKelione(currentX - 1, currentY, goalX, goalY, out
length, map); //paeina i kaire
                                }
                                else if (map[currentY, currentX - 1] == 'X' && map[currentY,
currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                                {
                                    if (map[currentY - 1, currentX] != 'X' && map[currentY,
currentX - 1] == 'P' && map[currentY, currentX - 1] == 0)
                                    {
                                        length++;
                                        draugoKelione(currentX, currentY - 1, goalX, goalY,
out length, map); //eina i apacia
                                    }
                                    else
                                        return false; //akligatvis
                                }
                            }
                        }
                    }
                }
            }
            return false;
        }
        /// <summary>
        /// Rezultatu spausdinimo i faila metodas
        /// </summary>
        /// <param name="map"></param>
        /// <param name="xDraugu"></param>
        /// <param name="yDraugu"></param>
        /// <param name="xSus"></param>
        /// <param name="ySus"></param>
        /// <param name="xPic"></param>
        /// <param name="yPic"></param>
        void spausdinimas(char[,] map, int[] xDraugu, int[] yDraugu, int xSus, int ySus, int
xPic, int yPic)
        {
            using (StreamWriter writer = new StreamWriter(@"..\..\Rezultatai.txt"))
            {
                writer.WriteLine("Rezultatai: ");
                writer.WriteLine("");
                writer.WriteLine("Draugu pozicijos (x,y): ");
                for (int i = 0; i < xDraugu.Length; i++)
                {
                    writer.WriteLine("Draugo nr. {0} pozicija - ({1},{2})", i + 1, xDraugu[i],
yDraugu[i]);
                }
                writer.WriteLine("");
                writer.WriteLine("Artimiausia susitikimo vieta: ({0},{1})", xSus, ySus);
                writer.WriteLine("Pasirinkta picerija: ({0},{1})", xPic, yPic);
                writer.WriteLine("");
                writer.WriteLine("Duomenys: ");
                //for(int i = 0; i < map.GetUpperBound(0); i++)
                //{
                //    for (int j = 0; j < map.GetUpperBound(0); i++)
                //    {
                //        Console.Write("{0}", map[i, j]);
                //    }
                //}
            }
        }
    }
```

1.7. Pradiniai duomenys ir rezultatai

Duomenys:

U3.txt

Pirmoje eilutėje nurodomas „žemėlapio" eilučių ir stulpelių skaičius".
Po to nurodomi įvairūs objektai, aprašyti užduotyje.

```
5 5
D...P
XX...
D.S.P
XX...
D.S..
```

Rezultatai:

Rezultatai.txt

Rezultatai:

```
Draugu pozicijos (x,y):
Draugo nr. 1 pozicija - (0,0)
Draugo nr. 1 pozicija - (0,2)
Draugo nr. 1 pozicija - (0,4)

Artimiausia susitikimo vieta: (2,2)
Pasirinkta picerija: (4,2)

Duomenys:
```

1.8. Dėstytojo pastabos

Nebuvo komentarų, neveikia dėl assemblerio klaidos. Parašyta dviem kalbom.

2. Dinaminis atminties valdymas (L2)

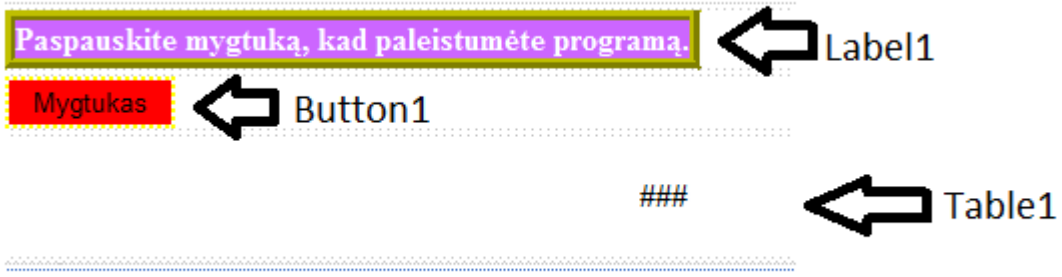
2.1. Darbo užduotis

LD_24. Detalės. Internetinėje parduotuvėje pirkėjai užsisakinėja robotų gamybai reikalingus įtaisus. Suraskite populiariausią įtaisą, kiek tokių įtaisų parduota ir už kokią sumą. Sudarykite tik vienos rūšies įtaisus pirkusių pirkėjų sąrašą, nupirktų įtaisų skaičių ir už juos sumokėtų pinigų sumą. Duomenys:

- tekstiniame faile U24a.txt yra informacija apie parduotuvėje parduodamus įtaisus: įtaiso kodas, įtaiso pavadinimas, įtaiso kaina;
- tekstiniame faile U24b.txt yra informacija apie pirkėjus: pirkėjo pavardė, vardas, pirkto įtaiso kodas, pirktų įtaisų kiekis.

Į kitą rinkinį atrinkite įtaisus, kurių parduota ne mažiau kaip n vienetų ir kurių vieneto kaina ne didesnė kaip k litų (n ir k įvedami klaviatūra).

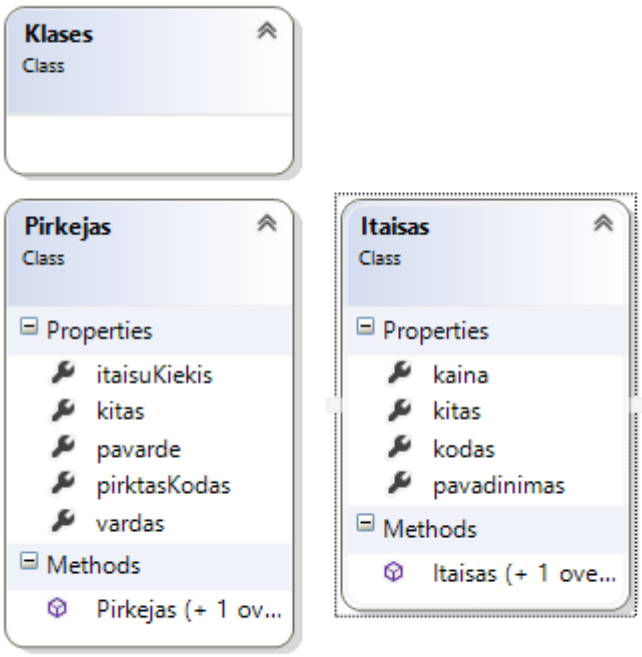
2.2. Grafinės vartotojo sąsajos schema



2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label1	BackgroundColor, Font, TextColor	Purple, Bold, White,
Button1	BackgroundColor, Bordercolor	Red, Yellow.
Table1		

2.4. Klasių diagrama



2.5. Programos vartotojo vadovas

1) Įkelti reikiamus, užduotyje nurodytus duomenų failus.
2) Vykdyti programą.
3) Paspausti mygtuką „mygtukas“.
4) 4 patikrinti rezultatus.

Internetinėje parduotuvėje pirkėjai užsisakinėja robotų gamybai reikalingus įtaisus

- tekstiniame faile U24a.txt yra informacija apie parduotuvėje parduodamus įtaisus: įtaiso kodas, įtaiso pavadinimas, įtaiso kaina;
- tekstiniame faile U24b.txt yra informacija apie pirkėjus: pirkėjo pavardė, vardas, pirkto įtaiso kodas, pirktų įtaisų kiekis.

Į kitą rinkinį atrinkite įtaisus, kurių parduota ne mažiau kaip n vienetų ir kurių vieneto kaina ne didesnė kaip k litų (n ir k įvedami klaviatūra).

2.6. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
/// <summary>
/// Tadas Laurinaitis IFF-6/8
/// </summary>
namespace Lab2web
{
    public partial class Forma1 : System.Web.UI.Page
```

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

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            Main();
        }

        const int maxItaisuSk = 5;
        const string failas1 = @"..\..\U24a.txt";
        const string failas2 = @"..\..\U24b.txt";
        const string file = @"D:\Rezultatai.txt";

        static void Main()
        {

            Itaisas visiItaisai = skaitytiItaisuDuomenis(failas1);
            Pirkejas visiPirkejai = skaitytiPirkejuDuomenis(failas2);
            int popItaisuKiek, vienosRusiesKiekis = 0;
            double suma1, suma2 = 0;
            string popItaisoPav = populiariausiasItaisas(visiItaisai, visiPirkejai, out
popItaisuKiek, out suma1);
            string[] vienosRusiesPirkejai = tikVienosRusies("ranka", visiItaisai,
visiPirkejai, out vienosRusiesKiekis, out suma2);
            Itaisas kitasRinkinys = KitasRinkinys(1, 1000, visiItaisai, visiPirkejai);
            Spausdinimas(file, popItaisoPav, popItaisuKiek, suma1, suma2,
vienosRusiesPirkejai, vienosRusiesKiekis, kitasRinkinys);
        }
        /// <summary>
        /// perskaito Itaisu duomenis
        /// </summary>
        /// <param name="failas"></param>
        /// <returns></returns>
        static Itaisas skaitytiItaisuDuomenis(string failas)
        {
            Itaisas visiItaisai = new Itaisas();

            using (StreamReader reader = new StreamReader(@failas))
            {
                string line = reader.ReadLine();
                if (line != null)
                {
                    string[] values = line.Split(';');
                    visiItaisai.kodas = values[0];
                    visiItaisai.pavadinimas = values[1];
                    visiItaisai.kaina = int.Parse(values[2]);
                }
                else if (line == null)
                {
                    return null;
                }
                while (null != (line = reader.ReadLine()))
                {
                    string[] values = line.Split(';');
                    Itaisas itaisas = new Itaisas();
                    itaisas.kodas = values[0];
                    itaisas.pavadinimas = values[1];
                    itaisas.kaina = int.Parse(values[2]);

                    itaisas.kitas = visiItaisai;
                    visiItaisai = itaisas;
                }
            }
            return visiItaisai;
        }
```

```
        /// <summary>
        /// perskaito pirkeju duomenis
        /// </summary>
        /// <param name="failas"></param>
        /// <returns></returns>
        static Pirkejas skaitytiPirkejuDuomenis(string failas)
        {
            Pirkejas visiPirkejai = new Pirkejas();

            using (StreamReader reader = new StreamReader(@failas))
            {
                string line = reader.ReadLine();
                if (line != null)
                {
                    string[] values = line.Split(';');
                    visiPirkejai.pavarde = values[0];
                    visiPirkejai.vardas = values[1];
                    visiPirkejai.pirktasKodas = values[2];
                    visiPirkejai.itaisuKiekis = int.Parse(values[3]);
                }
                else if (line == null)
                {
                    return null;
                }
                while (null != (line = reader.ReadLine()))
                {
                    string[] values = line.Split(';');
                    Pirkejas pirkejas = new Pirkejas();
                    pirkejas.pavarde = values[0];
                    pirkejas.vardas = values[1];
                    pirkejas.pirktasKodas = values[2];
                    pirkejas.itaisuKiekis = int.Parse(values[3]);

                    pirkejas.kitas = visiPirkejai;
                    visiPirkejai = pirkejas;
                }
            }
            return visiPirkejai;
        }
        /// <summary>
        /// suranda populiariausio itaiso pavadinima, taip pat grazina ju skaiciu ir kainu
suma
        /// </summary>
        /// <param name="visiItaisai"></param>
        /// <param name="visiPirkejai"></param>
        /// <param name="ind2"></param>
        /// <param name="suma"></param>
        /// <returns></returns>
        static string populiariausiasItaisas(Itaisas visiItaisai, Pirkejas visiPirkejai, out
int ind2, out double suma)
        {
            int ind1 = 0;
            ind2 = 0;
            suma = 0;
            string popItaisPav = "";
            Pirkejas visiPirkejai2 = new Pirkejas();
            Itaisas visiItaisai2 = new Itaisas();
            visiItaisai2 = visiItaisai;
            while (visiItaisai != null)
            {
                visiPirkejai2 = visiPirkejai;
                ind1 = 0;
                while (visiPirkejai2 != null)
                {
                    if (visiItaisai.kodas == visiPirkejai2.pirktasKodas)
                    {
                        ind1 += visiPirkejai2.itaisuKiekis;
                    }
```

```
                visiPirkejai2 = visiPirkejai2.kitas;
            }
            if (ind1 >= ind2)
            {
                ind2 = ind1;
                popItaisPav = visiItaisai.pavadinimas;
            }
            visiItaisai = visiItaisai.kitas;
        }

        while (visiItaisai2 != null)
        {
            if (popItaisPav == visiItaisai2.pavadinimas)
            {
                suma = ind2 * visiItaisai2.kaina;
            }
            visiItaisai2 = visiItaisai2.kitas;
        }
        return popItaisPav;
    }
    /// <summary>
    /// suranda tik vienos rusies itaiso pirkejus, ju kieki, pinigu suma
    /// </summary>
    /// <param name="itaisoPavadinimas"></param>
    /// <param name="visiItaisai"></param>
    /// <param name="visiPirkejai"></param>
    /// <param name="kiekis"></param>
    /// <param name="suma"></param>
    /// <returns></returns>
    static string[] tikVienosRusies(string itaisoPavadinimas, Itaisas visiItaisai,
Pirkejas visiPirkejai, out int kiekis, out double suma)
    {
        string kodas = "";
        double kaina = 0;
        suma = 0;
        kiekis = 0;
        int count = 0;
        string[] vardai = new string[maxItaisuSk];
        while (visiItaisai != null)
        {
            if (itaisoPavadinimas == visiItaisai.pavadinimas)
            {
                kodas = visiItaisai.kodas;
                kaina = visiItaisai.kaina;
            }
            visiItaisai = visiItaisai.kitas;
        }
        if (kodas == "" && kaina == 0)
        {
            Console.WriteLine("Itaiso tokiu pavadinimu nera.");
        }
        while (visiPirkejai != null)
        {
            if (visiPirkejai.pirktasKodas == kodas)
            {
                vardai[count] = visiPirkejai.vardas + " " + visiPirkejai.pavarde;
                kiekis += visiPirkejai.itaisuKiekis;
                suma += visiPirkejai.itaisuKiekis * kaina;
                count++;
            }
            visiPirkejai = visiPirkejai.kitas;
        }
        return vardai;
    }
    /// <summary>
    /// atrenka itaisus pagal uzduoties nurodymus
    /// </summary>
    /// <param name="n"></param>
```

```
        /// <param name="k"></param>
        /// <param name="visiItaisai"></param>
        /// <param name="visiPirkejai"></param>
        /// <returns></returns>
        static Itaisas KitasRinkinys(int n, double k, Itaisas visiItaisai, Pirkejas
visiPirkejai)
        {
            int sk = 0;
            Itaisas naujasRinkinys = new Itaisas();
            Pirkejas visiPirkejai2 = new Pirkejas();
            while (visiItaisai != null)
            {
                if (visiItaisai.kaina <= k)
                {
                    sk = 0;
                    visiPirkejai2 = visiPirkejai;
                    while (visiPirkejai2 != null)
                    {
                        if (visiItaisai.kodas == visiPirkejai2.pirktasKodas)
                        {
                            sk = sk + visiPirkejai2.itaisuKiekis;
                        }
                        visiPirkejai2 = visiPirkejai2.kitas;
                    }
                    if (sk >= n)
                    {
                        Itaisas naujesnis = new Itaisas();
                        naujesnis.kodas = visiItaisai.kodas;
                        naujesnis.pavadinimas = visiItaisai.pavadinimas;
                        naujesnis.kaina = visiItaisai.kaina;

                        naujesnis.kitas = naujasRinkinys;
                        naujasRinkinys = naujesnis;

                    }
                }
                visiItaisai = visiItaisai.kitas;
            }
            return naujasRinkinys;
        }
        /// <summary>
        /// spausdina rezultatus faile
        /// </summary>
        /// <param name="file"></param>
        /// <param name="popItPav"></param>
        /// <param name="popItKiek"></param>
        /// <param name="suma1"></param>
        /// <param name="suma2"></param>
        /// <param name="tikVienosRusies"></param>
        /// <param name="vienosRusiesKiekis"></param>
        /// <param name="kitasRinkinys"></param>
        static void Spausdinimas(string file, string popItPav, int popItKiek, double suma1,
double suma2, string[] tikVienosRusies, int vienosRusiesKiekis, Itaisas kitasRinkinys)
        {
            using (StreamWriter writer = new StreamWriter(@file))
            {
                writer.WriteLine("Populiariausio prietaiso pavadinimas, jo pardavimo skaičius
ir kaina: pavadinimas - {0}, skaicius - {1}, kaina - {2}euro.", popItPav, popItKiek, suma1);
                writer.WriteLine("Vienos rusies pirkeju sarasas, nupirktu itaisu skaicius ir
uz juos sumoketu pinigu suma: ");
                for(int i = 0; i < tikVienosRusies.Length-1; i++)
                {
                    writer.WriteLine("|{0, 15}|", tikVienosRusies[i]);
                }
                writer.WriteLine("Itaisu skaicius: {0} Sumoketa suma: {1}euro.",
vienosRusiesKiekis, suma2);
                while (kitasRinkinys != null)
                {
```

```
                    if (kitasRinkinys.pavadinimas != "")
                    {
                        writer.Write("|{0, 4}", kitasRinkinys.pavadinimas);
                        kitasRinkinys = kitasRinkinys.kitas;
                    }
                }
            }
        }
    }
}
```

2.7. Pradiniai duomenys ir rezultatai

```
Pradinių duomenų failai:


U24a.txt

Kodas;pavadinimas;vieneto kaina;

24fa;ranka;15;
25fb;koja;14;
26fc;sirdis;75;
27fd;petis;18;
28fe;nosis;142;
29ff;akis;155;

U24b.txt

Pavarde;Vardas;irenginio kodas;pirktas kiekis;

Zemaitis;Kazimieras;24fa;16;
Petraitis;Kazimieras;24fa;2;
Keturakis;Kazimieras;24fa;2;
Jonaitis;Kazimieras;25fb;4;
Zaliasis;Kazimieras;26fc;8;
Mentaitis;Kazimieras;27fd;1;
Brigaitis;Kazimieras;29ff;12;
Ausraitis;Kazimieras;28fe;7;
Kinderis;Kazimieras;25fb;55;
Karaitis;Kazimieras;29ff;13;
Zvaigzdaitis;Kazimieras;25fb;5;
Selmaitis;Kazimieras;24fa;9;
Vienaitis;Kazimieras;29ff;3;

Rezultatai:

Rezultatai.txt

Populiariausio prietaiso pavadinimas, jo pardavimo skaičius ir kaina: pavadinimas
- koja, skaicius - 64, kaina - 896euro.
Vienos rusies pirkeju sarasas, nupirktu itaisu skaicius ir uz juos sumoketu pinigu
suma:
|Kazimieras Selmaitis|
|Kazimieras Keturakis|
|Kazimieras Petraitis|
|Kazimieras Zemaitis |
Itaisu skaicius: 29 Sumoketa suma: 435euro.
|ranka|koja|sirdis|petis|nosis|akis|
```

2.8. Dėstytojo pastabos

Rezultatu faile truksta elementu, skurdi grafine sasaja, nesukurtus atskiras aplankas klasiu duomenims saugoti.

3. Bendrinės klasės ir sąsajos (L3)

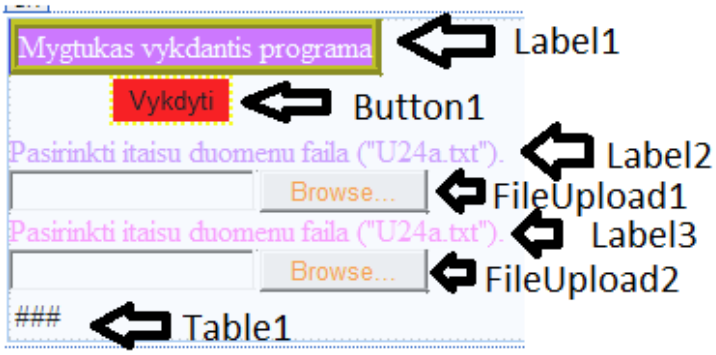
3.1. Darbo užduotis

LD_24. Detalės. Internetinėje parduotuvėje pirkėjai užsisakinėja robotų gamybai reikalingus įtaisus. Suraskite populiariausią įtaisą, kiek tokių įtaisų parduota ir už kokią sumą. Sudarykite tik vienos rūšies įtaisus pirkusių pirkėjų sąrašą, nupirktų įtaisų skaičių ir už juos sumokėtų pinigų sumą. Duomenys:

- tekstiniame faile U24a.txt yra informacija apie parduotuvėje parduodamus įtaisus: įtaiso kodas, įtaiso pavadinimas, įtaiso kaina;
- tekstiniame faile U24b.txt yra informacija apie pirkėjus: pirkėjo pavardė, vardas, pirkto įtaiso kodas, pirktų įtaisų kiekis.

Į kitą rinkinį atrinkite įtaisus, kurių parduota ne mažiau kaip n vienetų ir kurių vieneto kaina ne didesnė kaip k litų (n ir k įvedami klaviatūra).

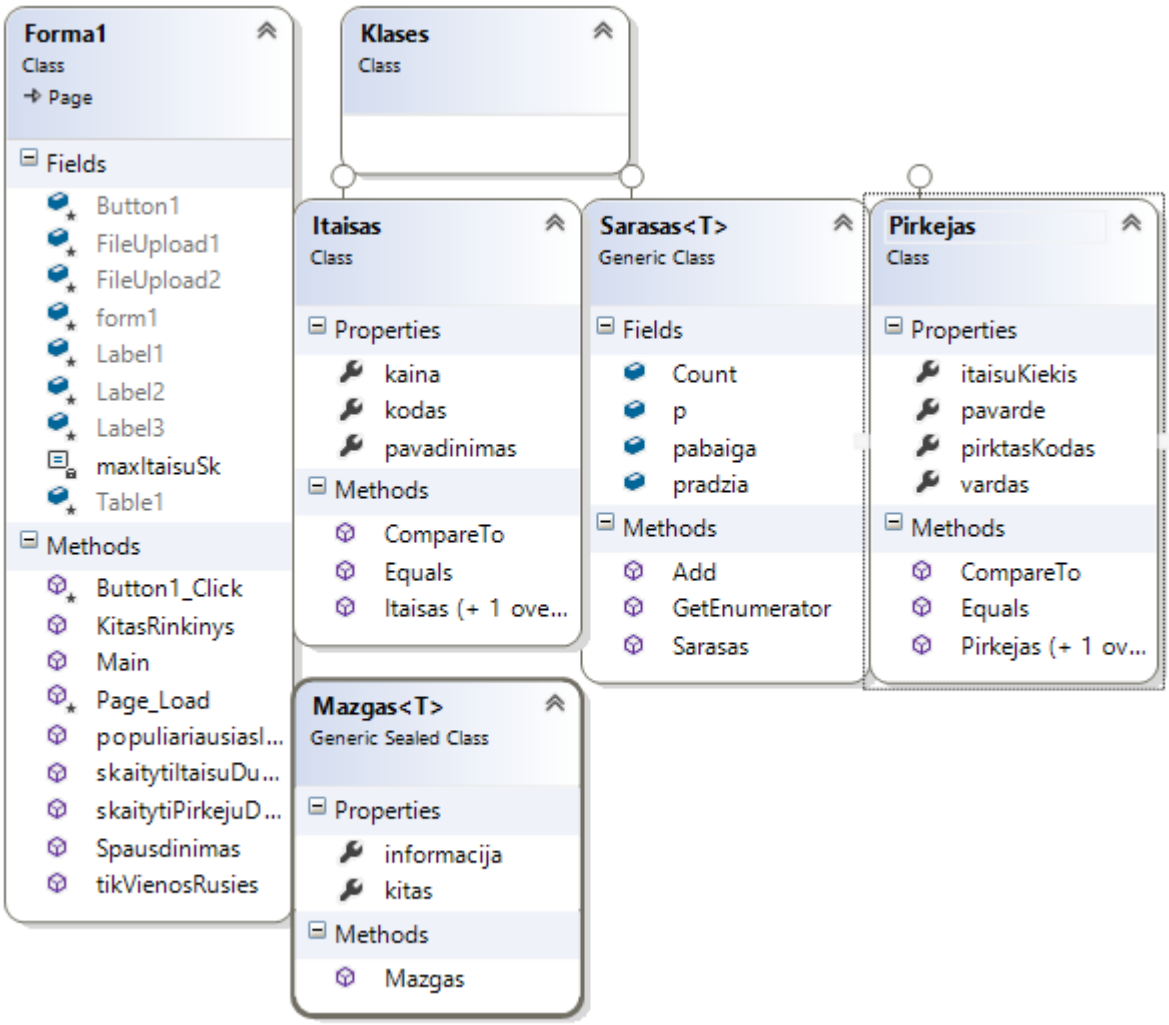
3.2. Grafinės vartotojo sąsajos schema



3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label1	BackgroundColour;BorderWidth;BorderColour;	Purple;3px;orange;
Label2	TextColour;	purple
Label3	TextColour;	purple
Button1	BackgroundColour;BorderWidth;BorderColour;	Red;2px;yellow;
FileUpload1	TextColour;	Orange;
FileUpload2	TextColour;	Orange;
Table1		

3.4. Klasių diagrama



3.5. Programos vartotojo vadovas

5) Įkelti reikiamus, užduotyje nurodytus duomenų failus.
6) Vykdyti programą.
7) Paspausti mygtuką „mygtukas“.
8) 4 patikrinti rezultatus.

Internetinėje parduotuvėje pirkėjai užsisakinėja robotų gamybai reikalingus įtaisus

- tekstiniame faile U24a.txt yra informacija apie parduotuvėje parduodamus įtaisus: įtaiso kodas, įtaiso pavadinimas, įtaiso kaina;
- tekstiniame faile U24b.txt yra informacija apie pirkėjus: pirkėjo pavardė, vardas, pirkto įtaiso kodas, pirktų įtaisų kiekis.

Į kitą rinkinį atrinkite įtaisus, kurių parduota ne mažiau kaip n vienetų ir kurių vieneto kaina ne didesnė kaip k litų (n ir k įvedami klaviatūra).

3.6. Programos tekstas

Klases.cs

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

namespace Lab2web
{
    public class Klases
    {
    }

    public class Sarasas<T> : IEnumerable where T : IComparable<T>
    {
        public Mazgas<T> pradzia;
        public Mazgas<T> pabaiga;
        public Mazgas<T> p;
        public int Count;

        public Sarasas()
        {
            Count = 0;
            pradzia = null;
            pabaiga = null;
            p = null;
        }

        public void Add(T item)
        {
            Count++;
            if (pradzia == null)
            {
                pradzia = pabaiga = new Mazgas<T>(item, pradzia);
                p = pabaiga;
            }
            else
            {
                Mazgas<T> laikinas = new Mazgas<T>(item, pradzia);
                pabaiga.kitas = laikinas;
                pabaiga = pabaiga.kitas;
                p = pabaiga;
            }
        }
        public IEnumerator GetEnumerator()
        {
            Mazgas<T> prad = pradzia;
            for (int i = 0; i < Count; i++)
            {
                yield return prad.informacija;
                prad = prad.kitas;
            }
        }
    }

    public class Mazgas<T>
    {
        public T informacija { get; set; }
        public Mazgas<T> kitas { get; set; }

        public Mazgas(T info, Mazgas<T> kt)
```

```
    {
        informacija = info;
        kitas = kt;
    }
}

public class Pirkejas : IComparable<Pirkejas>, IEquatable<Pirkejas>
{
    public string pavarde { get; set; }
    public string vardas { get; set; }
    public string pirktasKodas { get; set; }
    public int itaisuKiekis { get; set; }

    public Pirkejas()
    {

    }

    public Pirkejas(string pv, string vd, string pK, int itK)
    {
        pavarde = pv;
        vardas = vd;
        pirktasKodas = pK;
        itaisuKiekis = itK;
    }

    public int CompareTo(Pirkejas kitas)
    {
        return this.vardas.CompareTo(kitas.vardas);
    }

    public bool Equals(Pirkejas kitas)
    {
        return this.vardas == kitas.vardas;
    }
}

public class Itaisas : IComparable<Itaisas>, IEquatable<Itaisas>
{
    public string kodas { get; set; }
    public string pavadinimas { get; set; }
    public double kaina { get; set; }

    public Itaisas()
    {

    }

    public Itaisas(string kd, string pav, double kn)
    {
        kodas = kd;
        pavadinimas = pav;
        kaina = kn;
    }

    public int CompareTo(Itaisas kitas)
    {
        return this.kodas.CompareTo(kitas.kodas);
    }

    public bool Equals(Itaisas kitas)
    {
        return this.kodas == kitas.kodas;
```

```
        }
    }



}
```

Forma1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="Lab2web.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        <asp:Label ID="Label1" runat="server" BackColor="#CC66FF"
BorderColor="Yellow" BorderStyle="Ridge" BorderWidth="5px" ForeColor="White"
Text="Mygtukas vykdantis programa"></asp:Label>
        <br />
             <asp
:Button ID="Button1" runat="server" BackColor="Red" BorderColor="Yellow"
BorderStyle="Dotted" OnClick="Button1_Click" Text="Vykdyti" />

                  
        <br />
        <asp:Label ID="Label3" runat="server" ForeColor="#CC99FF" Text="Pasirinkti
itaisu duomenu faila (&quot;U24a.txt&quot;)."></asp:Label>
        <br />
        <asp:FileUpload ID="FileUpload1" runat="server" ForeColor="#FF9933" />
        <br />
        <asp:Label ID="Label2" runat="server" ForeColor="#FF99FF" Text="Pasirinkti
itaisu duomenu faila (&quot;U24a.txt&quot;)."></asp:Label>
        <br />
        <asp:FileUpload ID="FileUpload2" runat="server" ForeColor="#FF9933" />
        <br />
        <asp:Table ID="Table1" runat="server">
        </asp:Table>

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

Forma1.aspx.cs

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

namespace Lab2web
{
    public partial class Forma1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
```

```
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            if (FileUpload1.HasFile && FileUpload2.HasFile)
            {
                Main();
            }
            else
            {
                Label1.Text = " Pasirinkite duomenų failus";
                Label1.ForeColor = System.Drawing.Color.Red;
                Label1.Visible = true;
            }
        }

        public void Main()
        {
            //skaitymas
            Sarasas<Itaisas> visiItaisai = skaitytiItaisuDuomenis();
            Sarasas<Pirkejas> visiPirkejai = skaitytiPirkejuDuomenis();

            //populiariausi itaisai
            int popItaisuKiek, vienosRusiesKiekis = 0;
            double suma1, suma2 = 0;
            string popItaisoPav = populiariausiasItaisas(visiItaisai, visiPirkejai, out
popItaisuKiek, out suma1);

            //vienos rusies itaisu pirkejai
            string[] vienosRusiesPirkejai = tikVienosRusies("ranka", visiItaisai,
visiPirkejai, out vienosRusiesKiekis, out suma2);

            //atrinkti i kita rinkini itaiasai
            Sarasas<Itaisas> atrinkti = KitasRinkinys(50, 16, visiItaisai, visiPirkejai);

            //spausdinimas
            Spausdinimas(popItaisoPav, popItaisuKiek, suma1, suma2, vienosRusiesPirkejai,
vienosRusiesKiekis, atrinkti, visiItaisai, visiPirkejai);
        }
        /// <summary>
        /// Itaisu duomenu skaitymo metodas
        /// </summary>
        /// <returns></returns>
        public Sarasas<Itaisas> skaitytiItaisuDuomenis()
        {
            Sarasas<Itaisas> visiItaisai = new Sarasas<Itaisas>();

            using (StreamReader reader = new StreamReader(FileUpload1.FileContent))
            {
                string line = reader.ReadLine();
                if (line != null)
                {
                    string[] values = line.Split(';');
                    string kodas = values[0];
                    string pavadinimas = values[1];
                    int kaina = int.Parse(values[2]);
                    Itaisas temp = new Itaisas(kodas, pavadinimas, kaina);
                    visiItaisai.Add(temp);
                }
                else if (line == null)
                {
                    return null;
                }
                while (null != (line = reader.ReadLine()))
                {
                    string[] values = line.Split(';');
```

```
                    string kodas = values[0];
                    string pavadinimas = values[1];
                    int kaina = int.Parse(values[2]);
                    Itaisas temp = new Itaisas(kodas, pavadinimas, kaina);
                    visiItaisai.Add(temp);
                }
            }
            return visiItaisai;
        }
        /// <summary>
        /// metodas skaitantis pirkeju duomenis
        /// </summary>
        /// <returns></returns>
        public Sarasas<Pirkejas> skaitytiPirkejuDuomenis()
        {
            Sarasas<Pirkejas> visiPirkejai = new Sarasas<Pirkejas>();

            using (StreamReader reader = new StreamReader(FileUpload2.FileContent))
            {
                string line = reader.ReadLine();
                if (line != null)
                {
                    string[] values = line.Split(';');
                    string pavarde = values[0];
                    string vardas = values[1];
                    string kodas = values[2];
                    int itaisuKiekis = int.Parse(values[3]);
                    Pirkejas temp = new Pirkejas(pavarde, vardas, kodas, itaisuKiekis);
                    visiPirkejai.Add(temp);
                }
                else if (line == null)
                {
                    return null;
                }
                while (null != (line = reader.ReadLine()))
                {
                    string[] values = line.Split(';');
                    string pavarde = values[0];
                    string vardas = values[1];
                    string kodas = values[2];
                    int itaisuKiekis = int.Parse(values[3]);
                    Pirkejas temp = new Pirkejas(pavarde, vardas, kodas, itaisuKiekis);
                    visiPirkejai.Add(temp);
                }
            }
            return visiPirkejai;
        }
        /// <summary>
        /// metodas randantis populiariausia itaisa
        /// </summary>
        /// <param name="visiItaisai"></param>
        /// <param name="visiPirkejai"></param>
        /// <param name="ind2"></param>
        /// <param name="suma"></param>
        /// <returns></returns>
        public string populiariausiasItaisas(Sarasas<Itaisas> visiItaisai, Sarasas<Pirkejas>
visiPirkejai, out int ind2, out double suma)
        {
            int ind1 = 0;
            ind2 = 0;
            suma = 0;
            string popItaisPav = "";
            Sarasas<Itaisas> visiItaisai2 = visiItaisai;
            for (int i = 0; i < visiItaisai.Count; i++)
            {
                ind1 = 0;
                for (int j = 0; j < visiPirkejai.Count; j++)
                {
```

```
                    if (visiItaisai.pradzia.informacija.kodas ==
visiPirkejai.pradzia.informacija.pirktasKodas)
                    {
                        ind1 += visiPirkejai.pradzia.informacija.itaisuKiekis;
                    }
                    visiPirkejai.pradzia = visiPirkejai.pradzia.kitas;
                }
                if (ind1 >= ind2)
                {
                    ind2 = ind1;
                    popItaisPav = visiItaisai.pradzia.informacija.pavadinimas;
                }
                visiItaisai.pradzia = visiItaisai.pradzia.kitas;
            }
            for (int i = 0; i < visiItaisai2.Count; i++)
            {
                if (popItaisPav == visiItaisai2.pradzia.informacija.pavadinimas)
                {
                    suma = ind2 * visiItaisai2.pradzia.informacija.kaina;
                }
                visiItaisai2.pradzia = visiItaisai2.pradzia.kitas;
            }
            return popItaisPav;
        }
        /// <summary>
        /// metodas randantis tik vienos rusies
        /// </summary>
        /// <param name="itaisoPavadinimas"></param>
        /// <param name="visiItaisai"></param>
        /// <param name="visiPirkejai"></param>
        /// <param name="kiekis"></param>
        /// <param name="suma"></param>
        /// <returns></returns>
        public string[] tikVienosRusies(string itaisoPavadinimas, Sarasas<Itaisas>
visiItaisai, Sarasas<Pirkejas> visiPirkejai, out int kiekis, out double suma)
        {
            string kodas = "";
            double kaina = 0;
            suma = 0;
            kiekis = 0;
            int count = 0;
            int c = 0;
            for (int i = 0; i < visiItaisai.Count; i++)
            {
                if (itaisoPavadinimas == visiItaisai.pradzia.informacija.pavadinimas)
                {
                    kodas = visiItaisai.pradzia.informacija.kodas;
                    kaina = visiItaisai.pradzia.informacija.kaina;
                }
                visiItaisai.pradzia = visiItaisai.pradzia.kitas;
            }
            if (kodas == "" && kaina == 0)
            {
                Console.WriteLine("Itaiso tokiu pavadinimu nera.");
            }
            for (int i = 0; i < visiPirkejai.Count; i++)
            {
                if (visiPirkejai.pradzia.informacija.pirktasKodas == kodas)
                {
                    c++;
                }
                visiPirkejai.pradzia = visiPirkejai.pradzia.kitas;
            }
            string[] vardai = new string[c];
            for (int i = 0; i < visiPirkejai.Count; i++)
            {
                if (visiPirkejai.pradzia.informacija.pirktasKodas == kodas)
                {
```

```
                    vardai[count] = visiPirkejai.pradzia.informacija.vardas + " " +
visiPirkejai.pradzia.informacija.pavarde;
                    kiekis += visiPirkejai.pradzia.informacija.itaisuKiekis;
                    suma += visiPirkejai.pradzia.informacija.itaisuKiekis * kaina;
                    count++;
                }
                visiPirkejai.pradzia = visiPirkejai.pradzia.kitas;
            }
            return vardai;
        }
        /// <summary>
        /// metodas atrenkantis pagal uzduoti i kita rinkini
        /// </summary>
        /// <param name="n"></param>
        /// <param name="k"></param>
        /// <param name="visiItaisai"></param>
        /// <param name="visiPirkejai"></param>
        /// <returns></returns>
        public Sarasas<Itaisas> KitasRinkinys(int n, double k, Sarasas<Itaisas> visiItaisai,
Sarasas<Pirkejas> visiPirkejai)
        {
            int sk = 0;
            Sarasas<Itaisas> naujasRinkinys = new Sarasas<Itaisas>();
            Sarasas<Pirkejas> visiPirkejai2 = new Sarasas<Pirkejas>();
            for (int i = 0; i < visiItaisai.Count; i++)
            {
                if (visiItaisai.pradzia.informacija.kaina <= k)
                {
                    sk = 0;
                    visiPirkejai2 = visiPirkejai;
                    for (int j = 0; j < visiPirkejai2.Count; j++)
                    {
                        if (visiItaisai.pradzia.informacija.kodas ==
visiPirkejai2.pradzia.informacija.pirktasKodas)
                        {
                            sk = sk + visiPirkejai2.pradzia.informacija.itaisuKiekis;
                        }
                        visiPirkejai2.pradzia = visiPirkejai2.pradzia.kitas;
                    }
                    if (sk >= n)
                    {
                        Itaisas naujesnis = new Itaisas();
                        naujesnis.kodas = visiItaisai.pradzia.informacija.kodas;
                        naujesnis.pavadinimas = visiItaisai.pradzia.informacija.pavadinimas;
                        naujesnis.kaina = visiItaisai.pradzia.informacija.kaina;

                        naujasRinkinys.Add(naujesnis);
                    }
                }
                visiItaisai.pradzia = visiItaisai.pradzia.kitas;
            }
            return naujasRinkinys;
        }
        /// <summary>
        /// metodas spausdinantis rezultatus ir duomenis lentelemis i faila.
        /// </summary>
        /// <param name="popItPav"></param>
        /// <param name="popItKiek"></param>
        /// <param name="suma1"></param>
        /// <param name="suma2"></param>
        /// <param name="tikVienosRusies"></param>
        /// <param name="vienosRusiesKiekis"></param>
        /// <param name="kitasRinkinys"></param>
        /// <param name="itDuom"></param>
        /// <param name="pirkDuom"></param>
        public void Spausdinimas(string popItPav, int popItKiek, double suma1, double suma2,
string[] tikVienosRusies, int vienosRusiesKiekis, Sarasas<Itaisas> kitasRinkinys,
Sarasas<Itaisas> itDuom, Sarasas<Pirkejas> pirkDuom)
```

```
        {
            using (StreamWriter writer = new
StreamWriter(System.Web.HttpContext.Current.Server.MapPath("App_Data/Rezultatai.txt")))
            {
                writer.WriteLine("
Rezultatai: ");
                writer.WriteLine("-------------------------------------------------------------
-------------------------------------------------------------------------------
");
                writer.WriteLine("Populiariausio prietaiso pavadinimas, jo pardavimo skaičius
ir kaina: pavadinimas - {0}, skaicius - {1}, kaina - {2}euro.", popItPav, popItKiek, suma1);
                writer.WriteLine("Vienos rusies pirkeju sarasas, nupirktu itaisu skaicius ir
uz juos sumoketu pinigu suma: ");
                writer.WriteLine("");
                writer.WriteLine("     Vardas,   Pavarde");
                writer.WriteLine("-----------------------");
                for (int i = 0; i < tikVienosRusies.Length - 1; i++)
                {
                    if(tikVienosRusies[i] != "")
                    {
                        writer.WriteLine("{0}. {1, -12}", i+1, tikVienosRusies[i]);
                    }
                }
                writer.WriteLine(" ");
                writer.WriteLine("Itaisu skaicius: {0} Sumoketa suma: {1}euro.",
vienosRusiesKiekis, suma2);
                writer.WriteLine("-----------------------------------------");
                writer.WriteLine(" ");
                writer.WriteLine("I kita rinkini atrinkti itaisai: ");
                writer.WriteLine(" ");
                writer.WriteLine("Nr.  Pav.  Kodas  Kaina");
                writer.WriteLine("-----------------------");
                for (int i = 0; i < kitasRinkinys.Count; i++)
                {
                    if (kitasRinkinys.pradzia.informacija.pavadinimas != "")
                    {
                        writer.Write("{0, -2}. {1, -6}, {2, -5}, {3, -5}", i + 1,
kitasRinkinys.pradzia.informacija.pavadinimas, kitasRinkinys.pradzia.informacija.kodas,
kitasRinkinys.pradzia.informacija.kaina);
                        kitasRinkinys.pradzia = kitasRinkinys.pradzia.kitas;
                    }
                }
                writer.WriteLine("");
                writer.WriteLine("
Pradiniai duomenys: ");
                writer.WriteLine("-------------------------------------------------------------
-------------------------------------------------------------------------------
");
                writer.WriteLine("U24a.txt: ");
                writer.WriteLine("");
                for (int i = 0; i < itDuom.Count; i++)
                {
                    writer.WriteLine("|{0, -6}| {1, -6} | {2, -6} |",
itDuom.pradzia.informacija.kodas, itDuom.pradzia.informacija.pavadinimas,
itDuom.pradzia.informacija.kaina);
                    itDuom.pradzia = itDuom.pradzia.kitas;
                }
                writer.WriteLine("-------------------------------------------------------------
-------------------------------------------------------------------------------
");
                writer.WriteLine("U24b.txt: ");
                writer.WriteLine("");
                for (int i = 0; i < pirkDuom.Count; i++)
                {
                    writer.WriteLine("|{0, -12}| {1, -12} | {2, -6} | {2, -2} |",
pirkDuom.pradzia.informacija.vardas, pirkDuom.pradzia.informacija.pavarde,
pirkDuom.pradzia.informacija.pirktasKodas, pirkDuom.pradzia.informacija.itaisuKiekis);
                    pirkDuom.pradzia = pirkDuom.pradzia.kitas;
```

```
            }
        }
    }
}
```

3.7. Pradiniai duomenys ir rezultatai

```
Rezultatai.txt:
                                                                                Rezultatai:
-------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
Populiariausio prietaiso pavadinimas, jo pardavimo skaičius ir kaina: pavadinimas
- koja, skaicius - 64, kaina - 896euro.
Vienos rusies pirkeju sarasas, nupirktu itaisu skaicius ir uz juos sumoketu pinigu
suma:

     Vardas,   Pavarde
----------------------
1. Kazimieras Zemaitis
2. Kazimieras Petraitis
3. Kazimieras Keturakis

Itaisu skaicius: 29 Sumoketa suma: 435euro.
-----------------------------------------

I kita rinkini atrinkti itaisai:

Nr.  Pav.  Kodas  Kaina
------------------------
1 . koja  , 25fb , 14
                                                                        Pradiniai
duomenys:
-------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
U24a.txt:

|24fa  | ranka  | 15     |
|25fb  | koja   | 14     |
|26fc  | sirdis | 75     |
|27fd  | petis  | 18     |
|28fe  | nosis  | 142    |
|29ff  | akis   | 155    |
-------------------------------------------------------------------------------------------
-----------------------------------------------------------------------
U24b.txt:

|Kazimieras  | Zemaitis     | 24fa   | 24fa |
|Kazimieras  | Petraitis    | 24fa   | 24fa |
|Kazimieras  | Keturakis    | 24fa   | 24fa |
|Kazimieras  | Jonaitis     | 25fb   | 25fb |
|Kazimieras  | Zaliasis     | 26fc   | 26fc |
|Kazimieras  | Mentaitis    | 27fd   | 27fd |
|Kazimieras  | Brigaitis    | 29ff   | 29ff |
|Kazimieras  | Ausraitis    | 28fe   | 28fe |
|Kazimieras  | Kinderis     | 25fb   | 25fb |
|Kazimieras  | Karaitis     | 29ff   | 29ff |
```

```
|Kazimieras  | Zvaigzdaitis | 25fb   | 25fb |
|Kazimieras  | Selmaitis    | 24fa   | 24fa |
|Kazimieras  | Vienaitis    | 29ff   | 29ff |
```

3.8. Dėstytojo pastabos

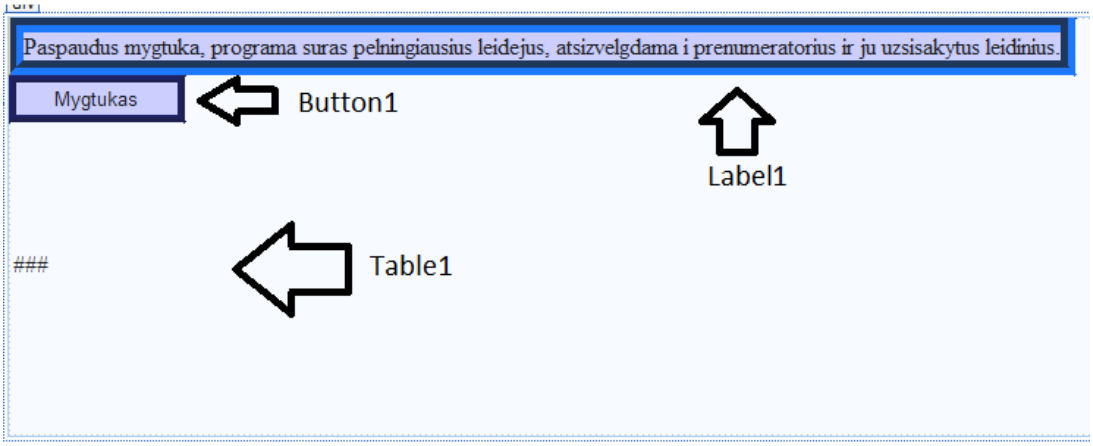
Testas – 2;
Nera komentarų, skurdus programos vartotojo vadovas, programa parašyta dviem kalbom.

4. Kolekcijos ir išimčių valdymas (L4)

4.1. Darbo užduotis

LDD_24. **Leidėjai.** Pirmoje failo eilutėje nurodyta įvedimo data, o tolesnėse eilutėse nurodyta prenumeratoriaus pavardė, adresas, laikotarpio pradžia (nurodyta sveiku skaičiumi 1..12), laikotarpio ilgis, leidinio kodas, leidinių kiekis. Kitame faile duota tokia informacija apie leidinius: kodas, pavadinimas, leidėjo pavadinimas, vieno mėnesio kaina. Suskaičiuoti kiekvienam leidėjui nurodyto mėnesio (įvedama klaviatūra) pajamas. Atspausdinkite leidėjų pajamas, surikiuotas pagal dydį ir leidėjų pavadinimus, nurodant ir leidėjų leidinius su jų atneštomis pajamomis. Leidėjų pavadinimai neturi kartotis.

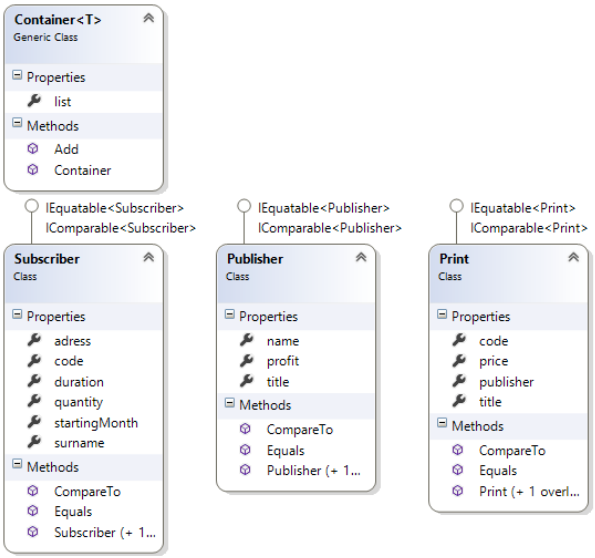
4.2. Grafinės vartotojo sąsajos schema



4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label1	BorderWidth, BorderColour,BackgroundColour	5px, Blue, Cyan
Button1	BorderWidth, BorderColour,BackgroundColour	3px, Dark Blue, Cyan
Table1		

4.4. Klasių diagrama



4.5. Programos vartotojo vadovas

Iš pradžių turi būti įkeliami failai į programos duomenų folderį (Lab4Web/App_Data/Data). Leidinių duomenys įkeliami į „prints“ folderį, prenumeratorių duomenys įkeliami į „subscribers“ folderį. Leidinių failo formatas: (Kodas;Pavadinimas;Leidejas;Kaina;), Prenumeratorių failo formatas: (Pavardė;Adresas;Pradinis mėnuo;Laikotarpis;Užsakymo kodas;Užsakymo kiekis;) (visi duomenys atskiriami kabliataškiais „ ; “). Paspaudus mygtuką, programa apskaičiuos pelningiausius leidejus, pagal leidinius ir leidiniu prenumeratorius. Rezultatai bus atspausdinti grafinėje sąsajoje lentelėje, taip pat ir faile, kurį galima rasti „Lab4Web/App_Data/Data“ direktorijoje, pavadinimu „Rezultatai.txt“.

4.6. Programos tekstas

Container.cs failas

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

namespace Lab4Web
{
    public class Container<T> where T : IEquatable<T>, IComparable<T>
    {
        public List<T> list { get; set; }

        public Container()
        {
            list = new List<T>();
        }
        /// <summary>
        /// Objekto idejimo i konteineri metodas
        /// </summary>
        /// <param name="data">Kurios nors klases objektas</param>
        public void Add(T data)
        {
```

```
            list.Add(data);
        }
    }
}
```

Print.cs failas

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

namespace Lab4Web
{
    public class Print : IEquatable<Print>, IComparable<Print>
    {
        public string code { get; set; }
        public string title { get; set; }
        public string publisher { get; set; }
        public double price { get; set; }

        public Print() { }

        public Print(string code, string title, string publisher, double price)
        {
            this.code = code;
            this.title = title;
            this.publisher = publisher;
            this.price = price;
        }
        /// <summary>
        /// IEquatable igyvendinimas
        /// </summary>
        /// <param name="other">Print klases objektas</param>
        /// <returns></returns>
        public bool Equals(Print other)
        {
            return (price == other.price);
        }
        /// <summary>
        /// Icomparable igyvendinimas
        /// </summary>
        /// <param name="other">Print klases objektas</param>
        /// <returns></returns>
        public int CompareTo(Print other)
        {
            return (price.CompareTo(other.price));
        }
    }
}
```

Publisher.cs failas

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

namespace Lab4Web
{
    public class Publisher : IEquatable<Publisher>, IComparable<Publisher>
    {
        public string name { get; set; }
        public double profit { get; set; }
        public string title { get; set; }

        public Publisher() { }
```

```
        public Publisher(string name, double profit, string title)
        {
            this.name = name;
            this.profit = profit;
            this.title = title;
        }
        /// <summary>
        /// IEquatable igyvendinimas
        /// </summary>
        /// <param name="other">Publisher klases objektas</param>
        /// <returns></returns>
        public bool Equals(Publisher other)
        {
            return (profit == other.profit);
        }
        /// <summary>
        /// Icomparable igyvendinimas
        /// </summary>
        /// <param name="other">Publisher klases objektas</param>
        /// <returns></returns>
        public int CompareTo(Publisher other)
        {
            return (profit.CompareTo(other.profit));
        }
    }
}
```

Subscriber.cs failas

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

namespace Lab4Web
{
    public class Subscriber : IEquatable<Subscriber>, IComparable<Subscriber>
    {
        public string surname { get; set; }
        public string adress { get; set; }
        public int startingMonth { get; set; }
        public int duration { get; set; }
        public string code { get; set; }
        public int quantity { get; set; }

        public Subscriber() { }

        public Subscriber(string surname, string adress, int startingMonth, int duration,
string code, int quantity)
        {
            this.surname = surname;
            this.adress = adress;
            this.startingMonth = startingMonth;
            this.duration = duration;
            this.code = code;
            this.quantity = quantity;
        }
        /// <summary>
        /// IEquatable igyvendinimas
        /// </summary>
        /// <param name="other">Subscriber klases objektas</param>
        /// <returns></returns>
        public bool Equals(Subscriber other)
        {
            return (code == other.code);
        }
```

```
        /// <summary>
        /// Icomparable igyvendinimas
        /// </summary>
        /// <param name="other">Subscriber klases objektas</param>
        /// <returns></returns>
        public int CompareTo(Subscriber other)
        {
            return (code.CompareTo(other.code));
        }
    }
}
```

Forma1.aspx.cs failas

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

namespace Lab4Web
{
    public partial class Forma1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Main();
        }
        void Main()
        {
            Container<Subscriber> subscribers = new Container<Subscriber>();
            Container<Print> allPrints = new Container<Print>();
            Container<Publisher> publishers = new Container<Publisher>();
            ReadSubscriberData(subscribers);
            ReadPrintData(allPrints);
            GetPublishers(allPrints, publishers);
            Profits(12, subscribers, allPrints, publishers);
            publishers.list = publishers.list.OrderByDescending(o => o.profit).ToList();
            PrintToFile(subscribers, allPrints, publishers);
            ShowData(publishers);



        }
        /// <summary>
        /// Perskaito prenumeratoiu duomenis
        /// </summary>
        /// <param name="subs">Prenumeratoriu konteineris</param>
        public void ReadSubscriberData(Container<Subscriber> subs)
        {
            string[] paths =
Directory.GetFiles(System.Web.HttpContext.Current.Server.MapPath(@"App_Data/Data/Subscribers")
);
            try
            {
                foreach (string path in paths)
                {
                    using (StreamReader reader = new StreamReader(path))
                    {
                        string line = "";
                        while (null != (line = reader.ReadLine()))
                        {
```

```
                            string[] values = line.Split(';');
                            string surname = values[0];
                            string adress = values[1];
                            int startingMonth = int.Parse(values[2]);
                            int duration = int.Parse(values[3]);
                            string code = values[4];
                            int quantity = int.Parse(values[5]);

                            Subscriber temp = new Subscriber(surname, adress, startingMonth,
duration, code, quantity);
                            subs.Add(temp);
                        }
                    }
                }
            }
            catch
            {
                throw new Exception("There are no files to read (1). ");
            }
        }
        /// <summary>
        /// Perskaito leidiniu duomenis
        /// </summary>
        /// <param name="prints">leidiniu konteineris</param>
        public void ReadPrintData(Container<Print> prints)
        {
            string[] paths =
Directory.GetFiles(System.Web.HttpContext.Current.Server.MapPath(@"App_Data/Data/Prints"));
            try
            {
                foreach (string path in paths)
                {
                    using (StreamReader reader = new StreamReader(@path))
                    {
                        string line = "";
                        while (null != (line = reader.ReadLine()))
                        {
                            string[] values = line.Split(';');
                            string code = values[0];
                            string title = values[1];
                            string publisher = values[2];
                            double price = double.Parse(values[3]);

                            Print temp = new Print(code, title, publisher, price);
                            prints.Add(temp);
                        }
                    }
                }
            }
            catch
            {
                throw new Exception("There are no files to read (2). ");
            }
        }
        /// <summary>
        /// Suranda visus leidejus is leidiniu konteinerio bei sudeda juos i atskira
konteineri
        /// </summary>
        /// <param name="prints">leidiniu konteineris</param>
        /// <param name="pubs">leideju konteineris</param>
        public void GetPublishers(Container<Print> prints, Container<Publisher> pubs)
        {
            try
            {
                foreach (Print p in prints.list)
                {
                    Publisher pub1 = new Publisher(p.publisher, 0, p.title);
                    pubs.Add(pub1);
```

```
                }
            }
            catch(NullReferenceException ex)
            {
                throw new Exception("Exception message: " +ex.Message);
            }
        }
        /// <summary>
        /// Suskaiciuoja leideju pelnus
        /// </summary>
        /// <param name="month">Ranka ivedamas menuo</param>
        /// <param name="subs">Prenumeratoriu konteineris</param>
        /// <param name="prints">Leidiniu konteineris</param>
        /// <param name="pubs">Leideju konteineris</param>
        public void Profits(int month, Container<Subscriber> subs, Container<Print> prints,
Container<Publisher> pubs)
        {
            foreach (Publisher pub in pubs.list)
            {
                foreach (Print p in prints.list)
                {
                    foreach (Subscriber s in subs.list)
                    {
                        if (pub.name == p.publisher)
                        {
                            if (p.code == s.code)
                            {
                                if ((s.startingMonth + s.duration) / 12 >= month / 12)
                                {
                                    pub.profit += (s.quantity * p.price);
                                }
                            }
                        }
                    }
                }
            }
        }
        /// <summary>
        /// Spausdina duomenis ir rezultatus lentelemis faile
        /// </summary>
        /// <param name="subscribers">Prenumeratoriu Konteineris</param>
        /// <param name="allPrints">Leidiniu Konteineris</param>
        /// <param name="publishers">Leideju konteineris</param>
        public void PrintToFile(Container<Subscriber> subscribers, Container<Print> allPrints,
Container<Publisher> publishers)
        {
            using (StreamWriter writer = new
StreamWriter(System.Web.HttpContext.Current.Server.MapPath("App_Data/Rezultatai.txt")))
            {
                writer.WriteLine("Pradiniai duomenys: ");
                writer.WriteLine(" ");
                writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | {4, -14} | {5,
-14} | ", "Pavarde", "Adresas", "Prad. Men.", "Trukme", "Kodas", "Kiekis");
                writer.WriteLine("-----------------------------------------------------------------
-------------------------------------");
                foreach (Subscriber s in subscribers.list)
                {
                    writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | {4, -14} |
{5, -14} | ", s.surname, s.adress, s.startingMonth, s.duration, s.code, s.quantity);
                }
                writer.WriteLine(" ");
                writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | ", "Kodas",
"Pavadinimas", "Leidejas", "Kaina");
                writer.WriteLine("-----------------------------------------------------------------
---");
                foreach (Print p in allPrints.list)
                {
```

```
                    writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | ", p.code,
p.title, p.publisher, p.price);
                }
                writer.WriteLine(" ");
                writer.WriteLine("Surikiuoti rezultatai: ");
                writer.WriteLine(" ");
                writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} |", "Pelnas", "Leidejas",
"Leidinys");
                writer.WriteLine("-----------------------------------------------");
                foreach (Publisher pub in publishers.list)
                {
                    writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | ", pub.profit,
pub.name, pub.title);
                }
            }
        }
        /// <summary>
        /// Parodo rezultatus lenteles pavidalu grafineje sasajoje
        /// </summary>
        /// <param name="publishers">Leideju konteineris</param>
        public void ShowData(Container<Publisher> publishers)
        {
            TableCell profit1 = new TableCell();
            TableCell name1 = new TableCell();
            TableCell title1 = new TableCell();
            TableRow infRow = new TableRow();
            profit1.Text = "Profit";
            name1.Text = "Name";
            title1.Text = "Title";
            infRow.Cells.Add(profit1);
            infRow.Cells.Add(name1);
            infRow.Cells.Add(title1);
            Table1.Rows.Add(infRow);
            foreach (Publisher pub in publishers.list)
            {
                TableCell profit = new TableCell();
                TableCell name = new TableCell();
                TableCell title = new TableCell();
                profit.Text = "" + pub.profit;
                name.Text = "" + pub.name;
                title.Text = "" + pub.title;

                TableRow row = new TableRow();
                row.Cells.Add(profit);
                row.Cells.Add(name);
                row.Cells.Add(title);
                Table1.Rows.Add(row);
            }
        }
    }
}
```

Forma1.aspx failas

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="Lab4Web.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
```

```
        <asp:Label ID="Label1" runat="server" BackColor="#CCCCFF" BorderColor="#0066FF"
BorderStyle="Groove" BorderWidth="10px" Text="Paspaudus mygtuka, programa suras pelningiausius
leidejus, atsizvelgdama i prenumeratorius ir ju uzsisakytus leidinius."></asp:Label>
        <br />

        <asp:Button ID="Button1" runat="server" BackColor="#CCCCFF" BorderColor="#000066"
BorderStyle="Ridge" BorderWidth="5px" OnClick="Button1_Click" Text="Mygtukas" Width="120px" />
        <asp:Table ID="Table1" runat="server" Height="194px" Width="232px">
        </asp:Table>
        <br />

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

4.7. Pradiniai duomenys ir rezultatai

Pradiniai duomenys:

Pavarde	Adresas	Prad. Men.	Trukme	Kodas	Kiekis
Antanaitis	Jono-5	2	6	4A	1
Mantainas	Sid-8	5	5	4A	2
Kombainas	Zuko-13	8	4	4A	22
Miestelenas	Jono-415	12	3	4B	3
Anglaitis	Jono-75	2	2	4B	14
Giedraitis	Jonso-5	7	12	4C	1
Bliumas	Sieed-8	9	2	4C	2
Zomsinas	Zxcuko-13	8	1	4D	1
Nielsenas	Jkolno-415	12	5	4D	5
Leslis	Jonlolo-75	5	6	4D	7

Kodas	Pavadinimas	Leidejas	Kaina
4A	Ratai	Aibe	14
4B	Sodas	Alba	9
4C	Elektronika	Fortas	4
4D	Miestas	Sigma	13

Surikiuoti rezultatai:

Pelnas	Leidejas	Leidinys
308	Aibe	Ratai
65	Sigma	Miestas
27	Alba	Sodas
4	Fortas	Elektronika

4.8. Dėstytojo pastabos

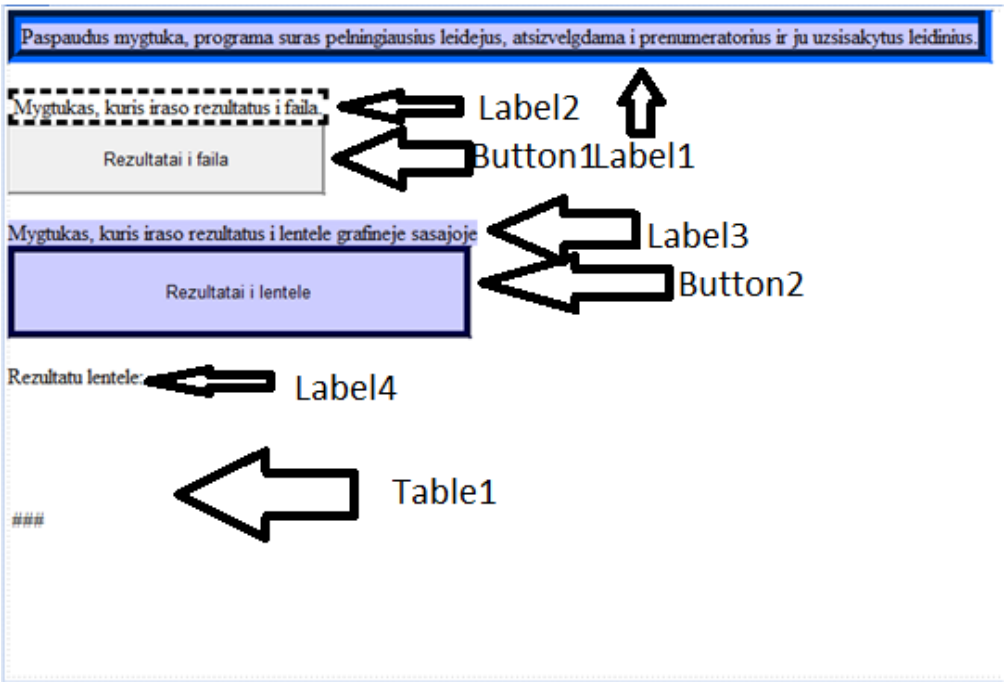
Peržiūrėti failų išdėstymo tvarką, pataisyti vartotojo vadovą, grafinės sąsajos schemą.
Testas - 2

5. Deklaratyvusis programavimas (L5)

5.1. Darbo užduotis

LDD_24. **Leidėjai.** Pirmoje failo eilutėje nurodyta įvedimo data, o tolesnėse eilutėse nurodyta prenumeratoriaus pavardė, adresas, laikotarpio pradžia (nurodyta sveiku skaičiumi 1..12), laikotarpio ilgis, leidinio kodas, leidinių kiekis. Kitame faile duota tokia informacija apie leidinius: kodas, pavadinimas, leidėjo pavadinimas, vieno mėnesio kaina. Suskaičiuoti kiekvienam leidėjui nurodyto mėnesio (įvedama klaviatūra) pajamas. Atspausdinkite leidėjų pajamas, surikiuotas pagal dydį ir leidėjų pavadinimus, nurodant ir leidėjų leidinius su jų atneštomis pajamomis. Leidėjų pavadinimai neturi kartotis.

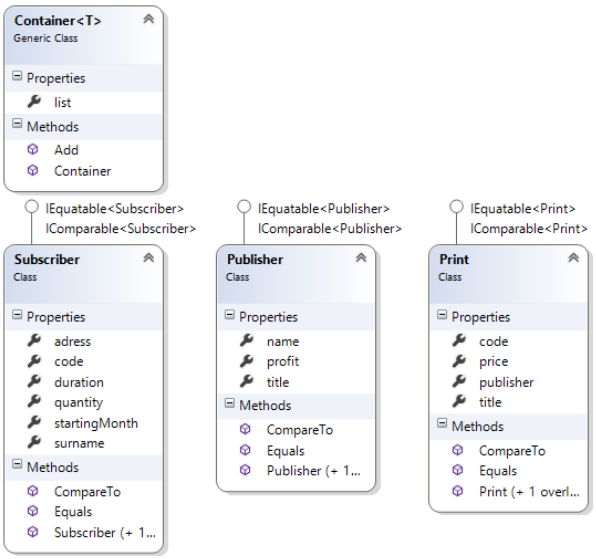
5.2. Grafinės vartotojo sąsajos schema



5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label1	BackgroundColor;bordercolor;	Purple;blue;
Label2	BorderStyle	Dotted;
Label3	BackgroundColor	
Label4		
Button1	BackgroundColour;	Blue;
Button2		
Table1		

5.4. Klasių diagrama



5.5. Programos vartotojo vadovas

Iš pradžių turi būti įkeliami failai į programos duomenų folderį (Lab4Web/App_Data/Data). Leidinių duomenys įkeliami į „prints“ folderį, prenumeratorių duomenys įkeliami į „subscribers“ folderį. Leidinių failo formatas: (Kodas;Pavadinimas;Leidejas;Kaina;), Prenumeratorių failo formatas: (Pavardė;Adresas;Pradinis mėnuo;Laikotarpis;Užsakymo kodas;Užsakymo kiekis;) (visi duomenys atskiriami kabliataškiais „ ; “). Paspaudus mygtukus, programa apskaičiuos pelningiausius leidejus, pagal leidinius ir leidiniu prenumeratorius. Paspaudus antrą mygtuką, rezultatai bus atspausdinti grafinėje sąsajoje lentelėje, o paspaudus pirmą - faile, kurį galima rasti „Lab4Web/App_Data/Data“ direktorijoje, pavadinimu „Rezultatai.txt“.

5.6. Programos tekstas

Container.cs failas

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

namespace Lab4Web
{
    public class Container<T> where T : IEquatable<T>, IComparable<T>
    {
        public List<T> list { get; set; }

        public Container()
        {
            list = new List<T>();
        }
        /// <summary>
        /// Objekto idejimo i konteineri metodas
        /// </summary>
        /// <param name="data">Kurios nors klases objektas</param>
        public void Add(T data)
        {
            list.Add(data);
        }
    }
}
```

Print.cs failas

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

namespace Lab4Web
{
    public class Print : IEquatable<Print>, IComparable<Print>
    {
        public string code { get; set; }
        public string title { get; set; }
        public string publisher { get; set; }
        public double price { get; set; }

        public Print() { }

        public Print(string code, string title, string publisher, double price)
        {
            this.code = code;
            this.title = title;
            this.publisher = publisher;
            this.price = price;
        }
        /// <summary>
        /// IEquatable igyvendinimas
        /// </summary>
        /// <param name="other">Print klases objektas</param>
        /// <returns></returns>
        public bool Equals(Print other)
        {
            return (price == other.price);
        }
        /// <summary>
        /// Icomparable igyvendinimas
        /// </summary>
        /// <param name="other">Print klases objektas</param>
        /// <returns></returns>
        public int CompareTo(Print other)
        {
            return (price.CompareTo(other.price));
        }
    }
}
```

Publisher.cs failas

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

namespace Lab4Web
{
    public class Publisher : IEquatable<Publisher>, IComparable<Publisher>
    {
        public string name { get; set; }
        public double profit { get; set; }
        public string title { get; set; }

        public Publisher() { }

        public Publisher(string name, double profit, string title)
        {
            this.name = name;
            this.profit = profit;
            this.title = title;
```

```
        }
        /// <summary>
        /// IEquatable igyvendinimas
        /// </summary>
        /// <param name="other">Publisher klases objektas</param>
        /// <returns></returns>
        public bool Equals(Publisher other)
        {
            return (profit == other.profit);
        }
        /// <summary>
        /// Icomparable igyvendinimas
        /// </summary>
        /// <param name="other">Publisher klases objektas</param>
        /// <returns></returns>
        public int CompareTo(Publisher other)
        {
            return (profit.CompareTo(other.profit));
        }
    }
}
```

Subscriber.cs failas

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

namespace Lab4Web
{
    public class Subscriber : IEquatable<Subscriber>, IComparable<Subscriber>
    {
        public string surname { get; set; }
        public string adress { get; set; }
        public int startingMonth { get; set; }
        public int duration { get; set; }
        public string code { get; set; }
        public int quantity { get; set; }

        public Subscriber() { }

        public Subscriber(string surname, string adress, int startingMonth, int duration,
string code, int quantity)
        {
            this.surname = surname;
            this.adress = adress;
            this.startingMonth = startingMonth;
            this.duration = duration;
            this.code = code;
            this.quantity = quantity;
        }
        /// <summary>
        /// IEquatable igyvendinimas
        /// </summary>
        /// <param name="other">Subscriber klases objektas</param>
        /// <returns></returns>
        public bool Equals(Subscriber other)
        {
            return (code == other.code);
        }
        /// <summary>
        /// Icomparable igyvendinimas
        /// </summary>
        /// <param name="other">Subscriber klases objektas</param>
        /// <returns></returns>
```

```
        public int CompareTo(Subscriber other)
        {
            return (code.CompareTo(other.code));
        }
    }
}
```

Forma1.aspx.cs failas

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

namespace Lab4Web
{
    public partial class Forma1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {
            Container<Subscriber> subscribers = new Container<Subscriber>();
            Container<Print> allPrints = new Container<Print>();
            Container<Publisher> publishers = new Container<Publisher>();

            ReadSubscriberData(subscribers);
            ReadPrintData(allPrints);
            GetPublishers(allPrints, publishers);
            Profits(12, subscribers, allPrints, publishers);

            publishers.list = publishers.list.OrderByDescending(o => o.profit).ThenBy(o =>
o.name).ToList();

            ShowData(publishers);
        }
        protected void Button2_Click(object sender, EventArgs e)
        {
            Container<Subscriber> subscribers = new Container<Subscriber>();
            Container<Print> allPrints = new Container<Print>();
            Container<Publisher> publishers = new Container<Publisher>();

            ReadSubscriberData(subscribers);
            ReadPrintData(allPrints);
            GetPublishers(allPrints, publishers);
            Profits(12, subscribers, allPrints, publishers);

            publishers.list = publishers.list.OrderByDescending(o => o.profit).ToList();

            PrintToFile(subscribers, allPrints, publishers);
        }
        void Main()
        {
        }
        /// <summary>
        /// Perskaito prenumeratoiu duomenis
        /// </summary>
        /// <param name="subs">Prenumeratoriu konteineris</param>
        public void ReadSubscriberData(Container<Subscriber> subs)
        {
            string[] paths =
Directory.GetFiles(System.Web.HttpContext.Current.Server.MapPath(@"App_Data/Data/Subscribers")
);
```

```
            try
            {
                foreach (string path in paths)
                {
                    using (StreamReader reader = new StreamReader(path))
                    {
                        string line = "";
                        while (null != (line = reader.ReadLine()))
                        {
                            string[] values = line.Split(';');
                            string surname = values[0];
                            string adress = values[1];
                            int startingMonth = int.Parse(values[2]);
                            int duration = int.Parse(values[3]);
                            string code = values[4];
                            int quantity = int.Parse(values[5]);

                            Subscriber temp = new Subscriber(surname, adress, startingMonth,
duration, code, quantity);
                            subs.Add(temp);
                        }
                    }
                }
            }
            catch
            {
                throw new Exception("There are no files to read (1). ");
            }
        }
        /// <summary>
        /// Perskaito leidiniu duomenis
        /// </summary>
        /// <param name="prints">leidiniu konteineris</param>
        public void ReadPrintData(Container<Print> prints)
        {
            string[] paths =
Directory.GetFiles(System.Web.HttpContext.Current.Server.MapPath(@"App_Data/Data/Prints"));
            try
            {
                foreach (string path in paths)
                {
                    using (StreamReader reader = new StreamReader(@path))
                    {
                        string line = "";
                        while (null != (line = reader.ReadLine()))
                        {
                            string[] values = line.Split(';');
                            string code = values[0];
                            string title = values[1];
                            string publisher = values[2];
                            double price = double.Parse(values[3]);

                            Print temp = new Print(code, title, publisher, price);
                            prints.Add(temp);
                        }
                    }
                }
            }
            catch
            {
                throw new Exception("There are no files to read (2). ");
            }
        }
        /// <summary>
        /// Suranda visus leidejus is leidiniu konteinerio bei sudeda juos i atskira
konteineri
        /// </summary>
        /// <param name="prints">leidiniu konteineris</param>
```

```
        /// <param name="pubs">leideju konteineris</param>
        public void GetPublishers(Container<Print> prints, Container<Publisher> pubs)
        {
            try
            {
                foreach (Print p in prints.list)
                {
                    Publisher pub1 = new Publisher(p.publisher, 0, p.title);
                    pubs.Add(pub1);
                }
            }
            catch(NullReferenceException ex)
            {
                throw new Exception("Exception message: " +ex.Message);
            }
        }
        /// <summary>
        /// Suskaiciuoja leideju pelnus
        /// </summary>
        /// <param name="month">Ranka ivedamas menuo</param>
        /// <param name="subs">Prenumeratoriu konteineris</param>
        /// <param name="prints">Leidiniu konteineris</param>
        /// <param name="pubs">Leideju konteineris</param>
        public void Profits(int month, Container<Subscriber> subs, Container<Print> prints,
Container<Publisher> pubs)
        {
            foreach (Publisher pub in pubs.list)
            {
                foreach (Print p in prints.list)
                {
                    foreach (Subscriber s in subs.list)
                    {
                        if (pub.name == p.publisher)
                        {
                            if (p.code == s.code)
                            {
                                if ((s.startingMonth + s.duration) / 12 >= month / 12)
                                {
                                    pub.profit += (s.quantity * p.price);
                                }
                            }
                        }
                    }
                }
            }
        }
        /// <summary>
        /// Spausdina duomenis ir rezultatus lentelemis faile
        /// </summary>
        /// <param name="subscribers">Prenumeratoriu Konteineris</param>
        /// <param name="allPrints">Leidiniu Konteineris</param>
        /// <param name="publishers">Leideju konteineris</param>
        public void PrintToFile(Container<Subscriber> subscribers, Container<Print> allPrints,
Container<Publisher> publishers)
        {
            using (StreamWriter writer = new
StreamWriter(System.Web.HttpContext.Current.Server.MapPath("App_Data/Rezultatai.txt")))
            {
                writer.WriteLine("Pradiniai duomenys: ");
                writer.WriteLine(" ");
                writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | {4, -14} | {5,
-14} | ", "Pavarde", "Adresas", "Prad. Men.", "Trukme", "Kodas", "Kiekis");
                writer.WriteLine("-----------------------------------------------------------------
-----------------------------------");
                foreach (Subscriber s in subscribers.list)
                {
                    writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | {4, -14} |
{5, -14} | ", s.surname, s.adress, s.startingMonth, s.duration, s.code, s.quantity);
```

```
                }
                writer.WriteLine(" ");
                writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | ", "Kodas",
"Pavadinimas", "Leidejas", "Kaina");
                writer.WriteLine("-------------------------------------------------------------
---");
                foreach (Print p in allPrints.list)
                {
                    writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | {3, -14} | ", p.code,
p.title, p.publisher, p.price);
                }
                writer.WriteLine(" ");
                writer.WriteLine("Surikiuoti rezultatai: ");
                writer.WriteLine(" ");
                writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} |", "Pelnas", "Leidejas",
"Leidinys");
                writer.WriteLine("-----------------------------------------------");
                foreach (Publisher pub in publishers.list)
                {
                    writer.WriteLine("| {0, -12} | {1, -12} | {2, -12} | ", pub.profit,
pub.name, pub.title);
                }
            }
        }
        /// <summary>
        /// Parodo rezultatus lenteles pavidalu grafineje sasajoje
        /// </summary>
        /// <param name="publishers">Leideju konteineris</param>
        public void ShowData(Container<Publisher> publishers)
        {
            TableCell profit1 = new TableCell();
            TableCell name1 = new TableCell();
            TableCell title1 = new TableCell();
            TableRow infRow = new TableRow();
            profit1.Text = "Profit";
            name1.Text = "Name";
            title1.Text = "Title";
            infRow.Cells.Add(profit1);
            infRow.Cells.Add(name1);
            infRow.Cells.Add(title1);
            Table1.Rows.Add(infRow);
            foreach (Publisher pub in publishers.list)
            {
                TableCell profit = new TableCell();
                TableCell name = new TableCell();
                TableCell title = new TableCell();
                profit.Text = "" + pub.profit;
                name.Text = "" + pub.name;
                title.Text = "" + pub.title;

                TableRow row = new TableRow();
                row.Cells.Add(profit);
                row.Cells.Add(name);
                row.Cells.Add(title);
                Table1.Rows.Add(row);
            }
        }
    }
}
```

5.7. Pradiniai duomenys ir rezultatai

Pradiniai duomenys:

```
| Pavarde      | Adresas      | Prad. Men.   | Trukme         | Kodas          |
Kiekis         |
-------------------------------------------------------------------------------
---------------
| Antanaitis   | Jono-5       | 2            | 6              | 4A             | 1
|
| Mantainas    | Sid-8        | 5            | 5              | 4A             | 2
|
| Kombainas    | Zuko-13      | 8            | 4              | 4A             |
22             |
| Miestelenas  | Jono-415     | 12           | 3              | 4B             | 3
|
| Anglaitis    | Jono-75      | 2            | 2              | 4B             |
14             |
| Giedraitis   | Jonso-5      | 7            | 12             | 4C             | 1
|
| Bliumas      | Sieed-8      | 9            | 2              | 4C             | 2
|
| Zomsinas     | Zxcuko-13    | 8            | 1              | 4D             | 1
|
| Nielsenas    | Jkolno-415   | 12           | 5              | 4D             | 5
|
| Leslis       | Jonlolo-75   | 5            | 6              | 4D             | 7
|

| Kodas        | Pavadinimas  | Leidejas     | Kaina          |
-----------------------------------------------------------------
| 4A           | Ratai        | Aibe         | 14             |
| 4B           | Sodas        | Alba         | 9              |
| 4C           | Elektronika  | Fortas       | 4              |
| 4D           | Miestas      | Sigma        | 13             |

Surikiuoti rezultatai:

| Pelnas       | Leidejas     | Leidinys     |
-----------------------------------------------
| 308          | Aibe         | Ratai        |
| 65           | Sigma        | Miestas      |
| 27           | Alba         | Sodas        |
| 4            | Fortas       | Elektronika  |
```

5.8. Dėstytojo pastabos

Keistas išimčių valdymas, netaisyklinga tvarka išdėstyti failai.