KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Objektinis programavimas II (P175B123)

Darbų aplankas

Atliko:

IFF- 6/8 gr. studentas

Tadas Laurinaitis

2017 m. gegužės 29 d.

Priėmė:

Doc. Aštrys Kirvaitis

KAUNAS 2017

TURINYS

1. Rekursija (L1)

1.1. Darbo užduotis

1.2. Grafinės vartotojo sąsajos schema

1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

1.4. Klasių diagrama

1.5. Programos vartotojo vadovas

1.6. Programos tekstas

1.7. Pradiniai duomenys ir rezultatai

1.8. Dėstytojo pastabos

2. Dinaminis atminties valdymas (L2)

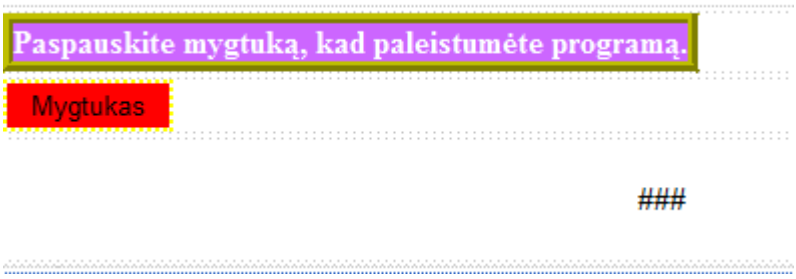
2.1. Darbo užduotis

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).

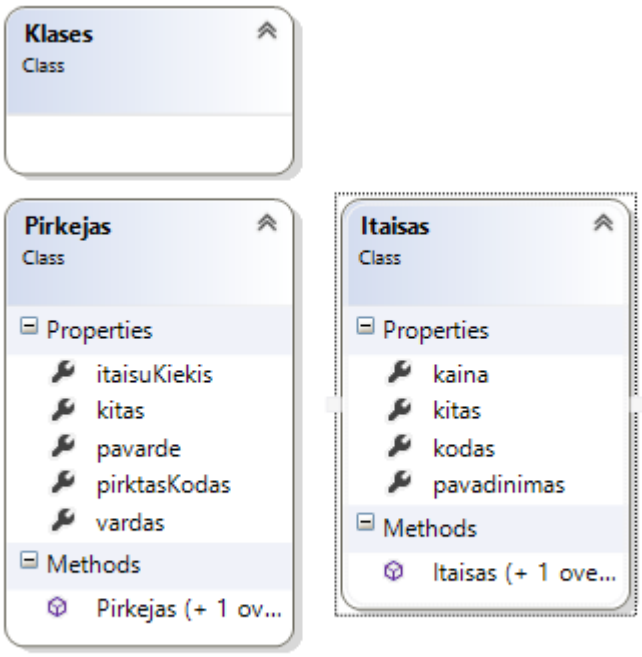
2.2. Grafinės vartotojo sąsajos schema



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

Įkėlus failą į programos aplanką ir paspaudus mygtuką, programa suranda visas reikiamas reikšmes.

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

3.2. Grafinės vartotojo sąsajos schema

3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

3.4. Klasių diagrama

3.5. Programos vartotojo vadovas

3.6. Programos tekstas

3.7. Pradiniai duomenys ir rezultatai

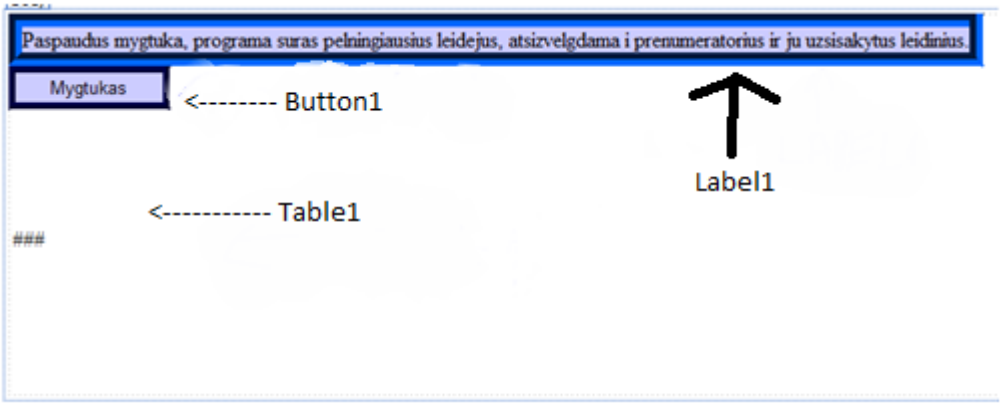
3.8. Dėstytojo pastabos

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.

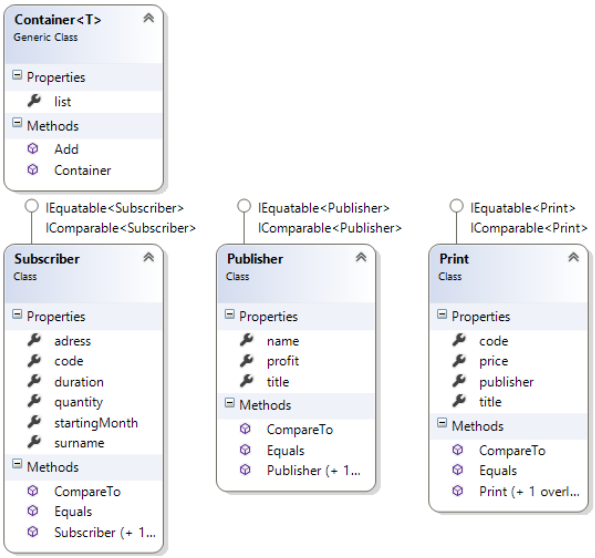
4.2. Grafinės vartotojo sąsajos schema



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

1) Iš pradžių turi būti įkeliami failai į programos duomenų folderį (Lab4Web/App_Data/Data).
2) 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 „ ; “).
3) Paspaudus mygtuką „Mygtukas“, 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 failas

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>
```

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);
            }
        }
    }
}
```

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

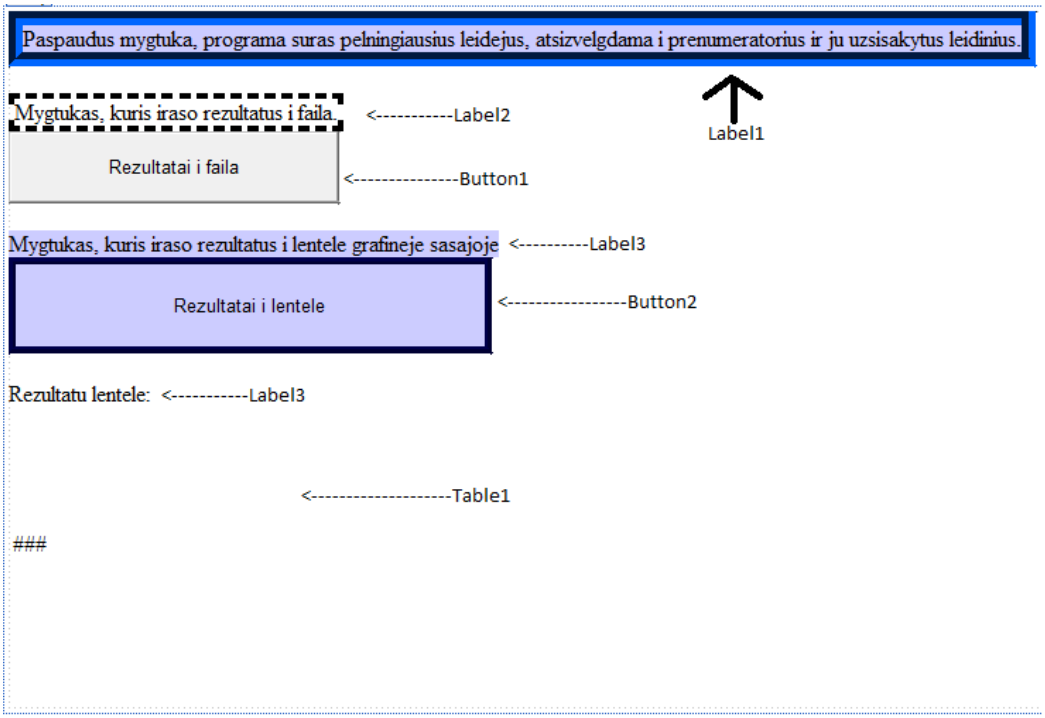
Peržiūrėti failų išdėstymo tvarką, pataisyti vartotojo vadovą, grafinės sąsajos schemą.
Testas - 2

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.

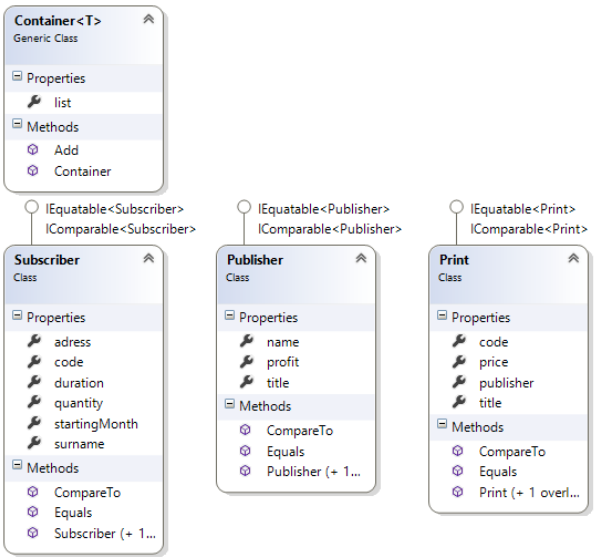
5.2. Grafinės vartotojo sąsajos schema



5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Label1		
Label2		
Label3		
Label4		
Button1		
Button2		
Table1		

5.4. Klasių diagrama



5.5. Programos vartotojo vadovas

1) Iš pradžių turi būti įkeliami failai į programos duomenų folderį (Lab4Web/App_Data/Data).
2) 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 „ ; “).
3) Paspaudus bet kurį iš mygtukų, programa apskaičiuos pelningiausius leidėjus, pagal leidinius ir leidinių prenumeratorius.
4) 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 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 />
        <br />
        <asp:Label ID="Label2" runat="server" BorderStyle="Dashed" Text="Mygtukas, kuris iraso
rezultatus i faila." ViewStateMode="Disabled"></asp:Label>
        <br />
        <asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="Rezultatai i
faila" Height="51px" Width="234px" />
        <br />
        <br />
        <asp:Label ID="Label3" runat="server" BackColor="#CCCCFF" Text="Mygtukas, kuris iraso
rezultatus i lentele grafineje sasajoje"></asp:Label>
        <br />
        <asp:Button ID="Button1" runat="server" BackColor="#CCCCFF" BorderColor="#000066"
BorderStyle="Ridge" BorderWidth="5px" OnClick="Button1_Click" Text="Rezultatai i lentele"
Width="342px" Height="68px" />
        <br />
        <br />
        <asp:Label ID="Label4" runat="server" Text="Rezultatu lentele:"></asp:Label>
        <asp:Table ID="Table1" runat="server" Height="194px" Width="232px">
        </asp:Table>
        <br />

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

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

Sutvarkyti failų tvarka, nėra aspx failo, pataisyti grafinės sasajos schemą, užpildyti reikšmių lentelę.
Testas – 1