

Kauno technologijos universitetas

Informatikos fakultetas

Objektinis programavimas 2 (P175B123)

Laboratorinių darbų ataskaita

Martynas Kuliešius IFF-1/9

Studentas

Doc. Dr. Sajavičius Svajūnas

Kaunas 2024

TURINYS

1. Rekursija (L1) 4

1.1. Darbo užduotis 4

1.2. Grafinės vartotojo sąsajos schema 4

1.3. Sąsajoje panaudotų komponentų keičiamos savybės 5

1.4. Klasių diagrama 5

1.5. Programos vartotojo vadovas 6

1.6. Programos tekstas 6

1.7. Pradiniai duomenys ir rezultatai 11

1.8. Dėstytojo pastabos 13

2. Dinaminis atminties valdymas (L2) 14

2.1. Darbo užduotis 14

2.2. Grafinės vartotojo sąsajos schema 15

2.3. Sąsajoje panaudotų komponentų keičiamos savybės 15

2.4. Klasių diagrama 16

2.5. Programos vartotojo vadovas 17

2.6. Programos tekstas 17

2.7. Pradiniai duomenys ir rezultatai 37

2.8. Dėstytojo pastabos 46

3. Bendrinės klasės ir testavimas (L3) 47

3.1. Darbo užduotis 47

3.2. Grafinės vartotojo sąsajos schema 47

3.3. Sąsajoje panaudotų komponentų keičiamos savybės 47

3.4. Klasių diagrama 48

3.5. Programos vartotojo vadovas 48

3.6. Programos tekstas 49

3.7. Pradiniai duomenys ir rezultatai 83

3.8. Dėstytojo pastabos 94

4. Polimorfizmas ir išimčių valdymas (L4) 95

4.1. Darbo užduotis 95

4.2. Grafinės vartotojo sąsajos schema 96

4.3. Sąsajoje panaudotų komponentų keičiamos savybės 96

4.4. Klasių diagrama 97

4.5. Programos vartotojo vadovas 97

4.6. Programos tekstas 98

4.7. Pradiniai duomenys ir rezultatai 118

4.8. Dėstytojo pastabos 127

5. Deklaratyvusis programavimas (L5) 128

5.1. Darbo užduotis 128

5.2. Grafinės vartotojo sąsajos schema 128

5.3. Sąsajoje panaudotų komponentų keičiamos savybės 128

5.4. Klasių diagrama 129

5.5. Programos vartotojo vadovas 130

5.6. Programos tekstas 132

5.7. Pradiniai duomenys ir rezultatai 147

5.8. Dėstytojo pastabos 160

# Rekursija (L1)

## Darbo užduotis

**LD\_10. Kurmiai**. Pavasarį sode apsigyveno kurmiai. Kiekvienas kurmis išsirausė sau atskirą urvą. Suskaičiuokite, kiek kurmių apsigyveno sode ir koks yra kiekvieno kurmio išrausto urvo dydis. Duomenys. Faile U3.txt yra pateikta sodo plokštuminė kurmių urvų schema – atvaizduota dvimačiu simbolių masyvu. Pirmoje failo eilutėje yra užrašytas schemos dydis: eilučių skaičius n (5 ≤ n ≤ 500) ir stulpelių skaičius m (5≤ m ≤ 500). Tolesnėse n eilučių yra užrašyta po m simbolių: 'z' (žemė) arba 'u' (urvas). Vienas simbolis atitinka 5 cm2 plotą. Du urvo simboliai ('u') priklauso tam pačiam urvui, jeigu jie yra greta toje pačioje eilutėje arba greta tame pačiame stulpelyje. Rezultatai. Atskirose eilutėse spausdinkite sode apsigyvenusių kurmių skaičių ir kurmių urvų dydžius (cm2 ) surikiuotus mažėjimo tvarka.

A white background with black text

Description automatically generated

## Grafinės vartotojo sąsajos schema

A screenshot of a computer

Description automatically generated

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Label1 | Text | LD\_10 |
| Label2 | Text | Nuskaityti Duomenys: |
| Label3 | Text | Gauti Rezultatai: |
| Label4 | Text | Martynas Kuliešius IFF-1/9 |
| TextBox1 | TextMode | Multiline |
| TextBox2 | TextMode | Multiline |
| Button1 | Text | Atlikti užduotį |

## Klasių diagrama

A diagram of a software development process

Description automatically generated with medium confidence

## Programos vartotojo vadovas

Į failą „U3.txt” surašoma reikalinga informacija: eilučių bei stulpelių skaičius vienoje eilutėje, ir sekančiose eilutėse supildomas kiemo daržo urvų žemėlapis. Paleidžiama programa, užkraunamas duomenų failas automatiškai ir atvaizduojamas tekstiniame lauke. Paspaudžiamas mygtukas „Atlikti užduotį“ ir išvedami apskaičiuoti rezultatai į antrąjį tekstinį lauką bei į failą „Result.txt“ : kurmių kiekis bei kiekvieno kurmio urvo dydis.

## Programos tekstas

**Moles.cs failas:**

namespace LD1\_10\_MKuliesius.AppClasses

{

    public class Moles

    {

        //Kurmio urvo dydis simboliais

        public int CaveSize { get; set; }

        public Moles(int caveSize)

        {

            this.CaveSize = caveSize;

        }

        public Moles(){}

        /// <summary>

        /// Overridinamas ToString metodas tam, kad teisingai išvestų rezultatus su apskaičiuota kūbinių centimetrų reikšme

        /// </summary>

        /// <returns></returns>

        public override string ToString()

        {

            return (CaveSize\*5).ToString();

        }

    }

}

**InOutUtils.cs failas:**

using System;

using System.Collections.Generic;

using System.IO;

namespace LD1\_10\_MKuliesius.AppClasses

{

    public class InOutUtils

    {

        public static string[] ReadData(string path)

        {

            string[] lines = File.ReadAllLines(path);

            return lines;

        }

        public static void PrintToFile(List<Moles> Moles, string path)

        {

            string baseDirectory = AppDomain.CurrentDomain.BaseDirectory;

            string outputPath=Path.Combine(baseDirectory,path);

            string directory =  Path.GetDirectoryName(outputPath);

            if (!Directory.Exists(directory))

            {

                Directory.CreateDirectory(directory);

            }

            using (StreamWriter writer = new StreamWriter(outputPath))

            {

                writer.WriteLine(Moles.Count);

                foreach (Moles m in Moles)

                {

                    writer.WriteLine(m.ToString());

                }

            }

        }

    }

}

**TaskUtils.cs failas:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace LD1\_10\_MKuliesius.AppClasses

{

    public class TaskUtils

    {

        /// <summary>

        /// Iškviečiama funkcija darbo atlikimui

        /// </summary>

        /// <param name="lines"></param>

        /// <param name="moleList"></param>

        public static void Work(string [] lines, List<Moles> moleList)

        {

            string[] dimensions = lines[0].Split(' ');

            int n = int.Parse(dimensions[0]);

            int m = int.Parse(dimensions[1]);

            char[,] map = new char[n, m];

            for (int i = 0; i < n; i++)

            {

                string row = lines[i + 1];

                for(int j = 0; j < m; j++)

                {

                    map[i, j] = row[j];

                }

            }

            int[] sizes=CountMoleHoles(map, n, m);

            for (int i = 0; i < sizes.Count(); i++)

            {

                Moles moleHole = new Moles();

                moleHole.CaveSize= sizes[i];

                moleList.Add(moleHole);

            }

        }

        /// <summary>

        /// Apskaičiuojamas kurmių skylių kiekis

        /// </summary>

        /// <returns></returns>

        static int[] CountMoleHoles(char[,] map, int n, int m)

        {

            bool[,] visited = new bool[n, m]; // Masyvas, skirtas žymėti lankytas vietas

            int[] sizes = new int[0]; // Masyvas, kuriame saugosime urvų dydžius

            for (int i = 0; i < n; i++)

            {

                for (int j = 0; j < m; j++)

                {

                    if (map[i, j] == 'u' && !visited[i, j])

                    { // Jei esame urve ir dar nebuvo lankyta ši vieta

                        int size = CountMoleHoleSize(map, visited, i, j, n, m); // Skaičiuojame urvo dydį

                        Array.Resize(ref sizes, sizes.Length + 1); // Išplečiame masyvą

                        sizes[sizes.Length - 1] = size; // Pridedame naują urvo dydį į masyvą

                    }

                }

            }

            Array.Sort(sizes); // Surikiuojame dydžius mažėjimo tvarka

            Array.Reverse(sizes); // Apverčiame masyvą, kad būtų mažėjimo tvarka

            return sizes;

        }

        /// <summary>

        /// Apskaičiuojamas Kurmio urvo skylės dydis.

        /// </summary>

        /// <returns></returns>

        static int CountMoleHoleSize(char[,] map, bool[,] visited, int i, int j, int n, int m)

        {

            if (i < 0 || i >= n || j < 0 || j >= m || map[i, j] != 'u' || visited[i, j])

            {

                return 0; // Jei esame už masyvo ribų, arba ne urve, arba jau lankėme šią vietą, grąžiname 0

            }

            visited[i, j] = true; // Žymime šią vietą kaip lankytą

            // Rekursyviai ieškome kaimyninių vietų

            return 1 + CountMoleHoleSize(map, visited, i - 1, j, n, m) + // viršus

                       CountMoleHoleSize(map, visited, i + 1, j, n, m) + // apačia

                       CountMoleHoleSize(map, visited, i, j - 1, n, m) + // kairė

                       CountMoleHoleSize(map, visited, i, j + 1, n, m); // dešinė

        }

    }

}

**Forma1.aspx failas:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma.aspx.cs" Inherits="LD1\_10\_MKuliesius.Forma" %>

<!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" Text="LD\_10"></asp:Label>

            <br/>

            <br/>

            <asp:Label ID="Label2" runat="server" Text="Nuskaityti Duomenys:"></asp:Label>

            <br/>

            <asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine" Height="125px"></asp:TextBox>

            <br/>

            <asp:Button ID="Button1" runat="server" Text="Atlikti užduotį" OnClick="Button1\_Click" />

            <br/>

            <br/>

            <asp:Label ID="Label3" runat="server" Text="Gauti rezultatai:"></asp:Label>

            <br/>

            <asp:TextBox ID="TextBox2" runat="server" TextMode="MultiLine" Height="125px"></asp:TextBox>

            <br/>

            <br/>

            <br/>

            <br/>

            <asp:Label ID="Label4" runat="server" Text="Martynas Kuliešius IFF-1/9"></asp:Label>

            <br/>

        </div>

    </form>

</body>

</html>

**Forma1.aspx.cs failas:**

using LD1\_10\_MKuliesius.AppClasses;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace LD1\_10\_MKuliesius

{

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

    {

        /// <summary>

        /// Duomenų ir išvedimo failai

        /// </summary>

        const string CFd = "U3.txt";

        const string CFr = "Result.txt";

        string dataPath;

        /// <summary>

        /// Metodas vykdantis nurodaytas užduotis užkraunant puslapį.

        /// </summary>

        protected void Page\_Load(object sender, EventArgs e)

        {

            //Nuskaito pradinių duomenų failą ir išveda į teksto dėžutę

            dataPath = Server.MapPath(CFd).ToString();

            var data = File.ReadAllText(Server.MapPath(CFd));

            TextBox1.Text = data.ToString();

        }

        protected void Button1\_Click(object sender, EventArgs e)

        {

            List<Moles> moles = new List<Moles>();

            string[] lines = InOutUtils.ReadData(dataPath);

            TaskUtils.Work(lines, moles);

            TextBox2.Text= moles.Count.ToString();

            foreach(Moles mole in moles)

            {

                TextBox2.Text = TextBox2.Text + "\n" + mole.ToString();

            }

            InOutUtils.PrintToFile(moles, CFr);

        }

    }

}

## Pradiniai duomenys ir rezultatai

**Pirmas testas:**

Pirmuoju testu tikrinu pagal duotus pradinius duomenis:

6 15

zzzzzzzzzuzuzzz

uuzuuzuuzuuuzzz

zuzuzzuzuuzuzuz

zuzuzzuzzuzuuuz

zuuuuuuuzuzzzzz

zzzzzuuuzzzzzzz

Viską padarius instrukcijose ir paleidus programą, gaunamas štai toks rezultatas:

2

110

70

Rezultatai gavosi tokie, kokių ir tikėjausi pagal užduotį.

**Antras testas:**

Antruoju testu tikrinu pagal naują žemėlapį, kuris yra 8\*20 dydžio:

8 20

zzzzzzzzzzzzzzzzzzzz

zzuzuzzuzuzzuzuzzuzu

zuzuzzuzuuzuzuzuzuzz

zzzzzzzzzzzzzzzzzzzz

zzzzuzuzuzzzzzzzzzzz

uzzzuzzzzzzzzuzzzzzz

uzzuzuzzuzuzzuzzuzuz

zzzzzzzzzzzzzzzzzzzz

Po programos darbo, gaunu tokius rezultatus:

25

15

10

10

10

10

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

5

**Trečias testas:**

Šiuo testu bandau programos veikimą su 5\*10 žemėlapiu:

5 10

zzzuzzzzzz

zuuzuzzzzz

uzzzzzzzzz

uzzuzzzzzz

uzzuzzzzzz

Gaunu tokius rezultatus:

5

15

10

10

5

5

## Dėstytojo pastabos

LD1 ataskaita: P1 (-0.3), P2 (-0.3). Įv.: 0.4

Gautas įvertinimas: 7

# Dinaminis atminties valdymas (L2)

## Darbo užduotis

**LD\_10. Gamykla**. Gamykloje kiekvieną dieną registruojama informacija apie darbininkų pagamintas detales. Darbininkas gali gaminti per dieną skirtingo tipo detales. Suraskite daugiausiai uždirbusio darbininko pavardę, suskaičiuokite, kiek dienų jis dirbo, kiek iš viso detalių pagamino ir už kokią sumą. Sudarykite tik vieno pavadinimo detales gaminusių darbininkų sąrašą, pagamintų detalių skaičių ir sumą. Surikiuokite šį sąrašą pagal pavardes ir vardus. Duomenys: • Tekstiniame faile U10a.txt surašyta: data (metai, mėnuo, diena), darbininko pavardė ir vardas, detalės kodas, pagamintų vienetų skaičius. • Tekstiniame faile U10b.txt surašyta: detalės kodas, detalės pavadinimas, įkainis. Iš duomenų rinkinio faile U10a.txt sudarykite naują duomenų rinkinį pagal nurodytą požymį (pagamintų vienetų skaičius > S, įkainis < K, įvedami klaviatūra). Sąrašas turi būti surikiuotas pagal pavardes ir vardus abėcėlės tvarka.

## Grafinės vartotojo sąsajos schema

A screenshot of a computer

Description automatically generated

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Label1 | Text | LD2 10 |
| Label2 | Text | Darbuotojai |
| Label3 | Text | Detalės |
| Button1 | Text | Skaityti |
| Button2 | Text | Atrinkti |
| Button3 | Text | Atrinkti |
| Label10 | Text | Įveskite kiekį |
| Label11 | Text | Įveskite įkainį |

## Klasių diagrama

A screenshot of a computer program

Description automatically generated

## Programos vartotojo vadovas

Pasileidus programą atveriamas vaizdas, kuriame matome visus mygtukus su tekstiniais laukais. Paspaudžiame ant pirmo failo pasirinkimo ir pasirenkame darbuotojų informacijos failą. Toliau paspaudžiame ant antrojo failo pasirinkimo ir pasirenkame detalių informacijos failą. Tuomet spaudžiame Skaityti. Tuomet programa nuskaito failų informaciją ir atvaizduoja ją ekrane. Taigi atvaizduoja pradinius failus, suranda geriausią darbuotoją ir jį atvaizduoja ir taip pat atrenka ir surikiuoja unikalius darbuotojus su jų pilnom uždirbtom sumom. Tuomet galime pasirinkti rūšiavimą pagal detalę, ir tada lauke, kuriame rodė unikalius darbuotojus atvaizduoja atfiltruotus bei surikiuotus rezultatus. Žemiau esančiuose tekstiniuose laukuose prašoma užpildyti atrankos kriterijus, pagal kuriuos į failą „U10a.txt“ išves atrinktus bei surikiuotus pagal pavardę ir vardą darbuotojus.

## Programos tekstas

**Node.cs failas:**

namespace LD2\_10\_MKuliesius.AppCode

{

    public class Node<T>

    {

        #region private variables

        //Aprašomi privatūs Node kintamieji

        private T data;

        private Node<T> next;

        #endregion

        #region Contructors

        /// <summary>

        /// Node konstruktorius

        /// </summary>

        /// <param name="data"> Node duomenys</param>

        /// <param name="next"> Sekantis Node</param>

        public Node(T data, Node<T> next)

        {

            this.data = data;

            this.next = next;

        }

        #endregion

        #region Properties

        /// <summary>

        /// Data property

        /// </summary>

        public T Data

        {

            get { return this.data; }

            set { this.data = value; }

        }

        /// <summary>

        /// Sekančio Node Property

        /// </summary>

        public Node<T> Next

        {

            get { return this.next; }

            set { this.next = value; }

        }

        #endregion

        #region Methods

        /// <summary>

        /// Gražina Node informaciją string forma

        /// </summary>

        public override string ToString()

        {

            return data.ToString();

        }

        #endregion

    }

}

**LinkedList.cs failas:**

using System;

using System.Collections;

using System.Collections.Generic;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class LinkedList<T> : IEnumerable<T> where T : IComparable<T>

    {

        private Node<T> Head;

        private int count;

        public IEnumerator<T> GetEnumerator()

        {

            for (Node<T> current = Head; current != null; current = current.Next)

            {

                yield return current.Data;

            }

        }

        IEnumerator IEnumerable.GetEnumerator()

        {

            return GetEnumerator();

        }

        public LinkedList()

        {

            Head = null;

            count = 0;

        }

        public bool Empty => count == 0;

        public int Count => count;

        public void Add(T obj)

        {

            AddToEnd(obj);

        }

        private void AddToEnd(T obj)

        {

            Node<T> newNode = new Node<T>(obj, null);

            if (Empty)

            {

                Head = newNode;

            }

            else

            {

                GetLastNode().Next = newNode;

            }

            count++;

        }

        /// <summary>

        /// Sorts the list ascending

        /// </summary>

        public void Sort()

        {

            // If the list is empty or contains only one element, it's already sorted

            if (Empty || Head.Next == null)

            {

                return;

            }

            bool swapped;

            do

            {

                swapped = false;

                Node<T> current = Head;

                Node<T> previous = null;

                while (current.Next != null)

                {

                    if (current.Data.CompareTo(current.Next.Data) > 0)

                    {

                        // Swap data of current node and next node

                        T temp = current.Data;

                        current.Data = current.Next.Data;

                        current.Next.Data = temp;

                        swapped = true;

                    }

                    previous = current;

                    current = current.Next;

                }

            } while (swapped);

        }

        /// <summary>

        /// Returns the indexed information.

        /// </summary>

        /// <param name="index"></param>

        /// <returns></returns>

        /// <exception cref="ArgumentOutOfRangeException"></exception>

        public T Get(int index)

        {

            if (index < 0 || index >= count)

            {

                throw new ArgumentOutOfRangeException(nameof(index), "Index is out of range.");

            }

            Node<T> current = Head;

            for (int i = 0; i < index; i++)

            {

                current = current.Next;

            }

            return current.Data;

        }

        /// <summary>

        /// Gets the first element that matches the specified condition.

        /// </summary>

        /// <param name="predicate">The condition to match.</param>

        /// <returns>The first element that matches the condition, or null if no such element is found.</returns>

        public T Get(Func<T, bool> predicate)

        {

            for (Node<T> current = Head; current != null; current = current.Next)

            {

                if (predicate(current.Data))

                {

                    return current.Data;

                }

            }

            return default; // Return default value if no element matches the condition

        }

        private Node<T> GetLastNode()

        {

            Node<T> current = Head;

            while (current.Next != null)

            {

                current = current.Next;

            }

            return current;

        }

    }

}

**Worker.cs failas:**

using System;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class Worker : IComparable<Worker>

    {

        public string Name { get; set; }

        public string Date { get; set; }

        public string DetailCode { get; set; }

        public int DetailProduced { get; set; }

        public int TotalDaysWorked { get; set; }

        public int TotalDetailsProduced { get; set; }

        public decimal TotalEarnings { get; set; }

        public Worker(string name, string date, string detailCode, int detailProduced)

        {

            Name = name;

            Date = date;

            DetailCode = detailCode;

            DetailProduced = detailProduced;

            TotalDaysWorked = 0;

            TotalDetailsProduced = 0;

            TotalEarnings = 0;

        }

        public Worker() { }

        public int CompareTo(Worker other)

        {

            return this.Name.CompareTo(other.Name);

        }

        public int CompareTo(decimal sum)

        {

            return this.TotalEarnings.CompareTo(sum);

        }

        // Override ToString method to provide meaningful string representation

        public override string ToString()

        {

            return $"| {Date, 10} | {Name, 20} | {DetailCode, 5} | {DetailProduced, 4} | {TotalDaysWorked,4} | {TotalDetailsProduced,5} | {TotalEarnings,4} |";

        }

    }

}

**Detail.cs failas:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class Detail : IComparable<Detail>

    {

        public string Code { get; private set; }

        public string Name { get; set; }

        public decimal Price { get; set; }

        public Detail(string code, string name, decimal price)

        {

            Code = code;

            Name = name;

            Price = price;

        }

        public int CompareTo(Detail other)

        {

            return Price.CompareTo( other.Price);

        }

        // Override ToString method to provide meaningful string representation

        public override string ToString()

        {

            return $"{Name, 10} | {Code, 12} | {Price, 8}";

        }

    }

}

**InOut.cs failas:**

using System.Collections.Generic;

using System.IO;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class InOut

    {

        /// <summary>

        /// Reads workers information

        /// </summary>

        /// <param name="fileName"> data file</param>

        /// <returns></returns>

        public static LinkedList<Worker> ReadWorkerFile(string fileName)

        {

            LinkedList<Worker> workers = new LinkedList<Worker>();

            string[] lines = File.ReadAllLines(fileName);

            foreach (string line in lines)

            {

                string[] parts = line.Split(';');

                if (parts.Length == 4) // Assuming each line has 4 parts

                {

                    string dateStr = parts[0].Trim();

                    string workerName = parts[1].Trim();

                    string detailCode = parts[2].Trim();

                    int quantity = int.Parse(parts[3].Trim());

                    Worker worker = new Worker(workerName, dateStr, detailCode, quantity);

                    // Add the worker to the linked list

                    workers.Add(worker);

                }

            }

            return workers;

        }

        /// <summary>

        /// Reads details info

        /// </summary>

        /// <param name="fileName"> Data file</param>

        /// <returns></returns>

        public static LinkedList<Detail> ReadDetailFile(string fileName)

        {

            LinkedList<Detail> details = new LinkedList<Detail>();

            string[] lines = File.ReadAllLines(fileName);

            foreach (string line in lines)

            {

                string[] parts = line.Split(';');

                if(parts.Length == 3)

                {

                    string code = parts[0].Trim();

                    string name = parts[1].Trim();

                    decimal price = decimal.Parse(parts[2].Trim());

                    Detail det = new Detail(code, name, price);

                    details.Add(det);

                }

            }

            return details;

        }

        /// <summary>

        /// Writes initial Worker detail

        /// </summary>

        /// <param name="fileName"></param>

        /// <param name="header"></param>

        /// <param name="list"></param>

        public static void WriteInitialWorkerData(string fileName, string header, LinkedList<Worker> list)

        {

            string dashes = new string('-', 89);

            List<string> lines = new List<string>

            {

                header,

                dashes,

                string.Format($"| {"Data", -10} | {"Vardas", -20} | {"Det. Kodas", 5} | {"Pagamino", 4} | {"Viso dirbo", 4} | {"Viso pagamino", 4} | {"Viso uzdirbo", 4} |"),

                dashes

            };

            foreach (Worker worker in list)

            {

                lines.Add(worker.ToString() + "\n");

            }

            lines.Add(dashes);

            lines.Add("\n");

            File.AppendAllLines(fileName, lines);

        }

        /// <summary>

        /// Writes initial data of detail

        /// </summary>

        /// <param name="fileName"></param>

        /// <param name="header"></param>

        /// <param name="list"></param>

        public static void WriteInitialDetailData(string fileName, string header, LinkedList<Detail> list)

        {

            string dashes = new string('-', 60);

            List<string> lines = new List<string>

            {

                header,

                dashes,

                string.Format($"| {"Pavadinimas",-15} | {"Det. Kodas", 12} | {"Kaina", 8} |"),

                dashes

            };

            foreach (Detail detail in list)

            {

                lines.Add(detail.ToString());

            }

            lines.Add(dashes);

            lines.Add("\n");

            File.AppendAllLines(fileName, lines);

        }

        /// <summary>

        /// Copies the existing result file.

        /// </summary>

        /// <param name="fileName">File name</param>

        /// <returns>An array of lines</returns>

        public static string[] SaveCurrentResultFile(string fileName)

        {

            string[] lines = File.ReadAllLines(fileName);

            return lines;

        }

        public static List<string> WriteInitialWorkerDataList( string header, LinkedList<Worker> list)

        {

            string dashes = new string('-', 89);

            List<string> lines = new List<string>

            {

                header+"\n",

                dashes + "\n",

                string.Format($"| {"Data", -10} | {"Vardas", -20} | {"Det. Kodas", 5} | {"Pagamino", 4} | {"Viso dirbo", 4} | {"Viso pagamino", 4} | {"Viso uzdirbo", 4} |")+"\n",

                dashes+"\n"

            };

            foreach (Worker worker in list)

            {

                lines.Add(worker.ToString()+"\n");

            }

            lines.Add(dashes+"\n");

            lines.Add("\n");

            return lines;

        }

        public static List<string> WriteInitialDetailDataList( string header, LinkedList<Detail> list)

        {

            string dashes = new string('-', 60);

            List<string> lines = new List<string>

            {

                header+ "\n",

                dashes+ "\n",

                string.Format($"| {"Pavadinimas",-15} | {"Det. Kodas", 12} | {"Kaina", 8} |")+ "\n",

                dashes+ "\n"

            };

            foreach (Detail detail in list)

            {

                lines.Add(detail.ToString() + "\n");

            }

            lines.Add(dashes + "\n");

            lines.Add("\n");

            return lines;

        }

    }

}

**TaskUtils failas:**

using System;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class TaskUtils

    {

        public static decimal CalculateEarningsForPart(LinkedList<Worker> workerList, LinkedList<Detail> detailList, string desiredWorker, string desiredPart)

        {

            decimal earnings = 0;

            foreach(Detail detail in detailList)

            {

                if(detail.Code == desiredPart)

                {

                    foreach(Worker worker in workerList)

                    {

                        if(worker.Name == desiredWorker)

                        {

                            earnings += worker.DetailProduced \* detail.Price;

                        }

                        // worker.TotalEarnings += earnings;

                    }

                }

            }

            return earnings;

        }

        /// <summary>

        /// Makes sepparate list with unique workers

        /// </summary>

        /// <param name="workerList"></param>

        /// <returns></returns>

        public static LinkedList<Worker> MakeSepparateWorkerList(LinkedList<Worker> workerList, LinkedList<Detail> detailList)

        {

            LinkedList<Worker> result = new LinkedList<Worker>();

            foreach (Worker worker in workerList)

            {

                if (!IsInList(result, worker))

                {

                    result.Add(worker);

                }

            }

            return result;

        }

        /// <summary>

        /// Updates workers information in list

        /// </summary>

        /// <param name="Uniqueworkers"></param>

        /// <param name="workers"></param>

        /// <param name="detailList"></param>

        public static void UpdateWorkers(LinkedList<Worker> Uniqueworkers,LinkedList<Worker>workers, LinkedList<Detail> detailList)

        {

            foreach (Worker worker in Uniqueworkers)

            {

                worker.TotalEarnings = TotalEarnings(workers, worker, detailList);

                //decimal total = TotalEarnings(workers, worker, detailList);

                worker.TotalDaysWorked = TotalWorkDays(workers, worker);

                worker.TotalDetailsProduced = TotalDetails(workers, worker);

            }

        }

        /// <summary>

        /// Checks if worker is in list

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <returns></returns>

        public static bool IsInList(LinkedList<Worker> workerList, Worker worker)

        {

            if (workerList.Empty)

            {

                return false;

            }

            else

            {

                foreach (Worker worker1 in workerList)

                {

                    if (worker.Name.Equals(worker1.Name))

                    {

                        return true;

                    }

                }

            }

            return false;

        }

        /// <summary>

        /// Calculates the total ammount of earnings for the worker

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <param name="detailList"></param>

        /// <returns></returns>

        public static decimal TotalEarnings(LinkedList<Worker> workerList, Worker worker, LinkedList<Detail> detailList)

        {

            decimal total = 0;

            foreach (Worker worker1 in workerList)

            {

                if (worker1.Name.Equals(worker.Name))

                {

                    foreach (Detail detail in detailList)

                    {

                        if (detail.Code.Equals(worker.DetailCode))

                        {

                            worker1.TotalEarnings += worker1.DetailProduced \* detail.Price;

                            total += worker1.DetailProduced \* detail.Price;

                        }

                    }

                }

            }

            return total;

        }

        /// <summary>

        /// Calculates how many details worker has made in total

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <returns></returns>

        public static int TotalDetails(LinkedList<Worker> workerList, Worker worker)

        {

            int total = 0;

            foreach (Worker worker1 in workerList)

            {

                if (worker1.Name.Equals(worker.Name))

                {

                    total+=worker1.DetailProduced;

                    worker1.TotalDetailsProduced += worker1.DetailProduced;

                }

            }

            return total;

        }

        /// <summary>

        /// Calculates how many work days worker worked in total

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <returns></returns>

        public static int TotalWorkDays(LinkedList<Worker> workerList, Worker worker)

        {

            int total = 0;

            string date = "";

            foreach (Worker worker1 in workerList)

            {

                if (worker1.Name.Equals(worker.Name) && !worker1.Date.Equals(date))

                {

                    total++;

                    date = worker1.Date;

                }

            }

            return total;

        }

        /// <summary>

        /// finds best worker from unique workers list

        /// </summary>

        /// <param name="workerList"></param>

        /// <returns></returns>

        public static Worker BestWorker(LinkedList<Worker> workerList)

        {

            Worker worker = new Worker();

            decimal max = 0;

            foreach (Worker worker1 in workerList)

            {

                if (worker1.TotalEarnings.CompareTo(max) >0)

                {

                    max = worker1.TotalEarnings;

                    worker = worker1;

                }

            }

            return worker;

        }

        /// <summary>

        /// Makes a list of workers that worked on a certain part

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="partName"></param>

        /// <param name="detailList"></param>

        /// <returns></returns>

        public static LinkedList<Worker> MakeWorkerListByPartName(LinkedList<Worker> workerList, string partName, LinkedList<Detail> detailList)

        {

            LinkedList<Worker> result = new LinkedList<Worker>();

            foreach(Detail detail in detailList)

            {

                if (detail.Name.Equals(partName))

                {

                    string partCode = detail.Code;

                    foreach (Worker worker1 in workerList)

                    {

                        if(worker1.DetailCode.Equals(partCode))

                        {

                            result.Add(worker1);

                        }

                    }

                }

            }

            return result;

        }

        /// <summary>

        /// calculates total parts made for list.

        /// </summary>

        /// <param name="workerList"></param>

        /// <returns></returns>

        public static int CalculateTotalParts(LinkedList<Worker> workerList)

        {

            int total = 0;

            foreach (Worker worker in workerList)

            {

                total += worker.DetailProduced;

            }

            return total;

        }

        /// <summary>

        /// returns total ammount of money earned

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="detailList"></param>

        /// <returns></returns>

        public static decimal CalculateTotalEarned(LinkedList<Worker> workerList, LinkedList<Detail> detailList)

        {

            decimal total = 0;

            foreach (Worker worker in workerList)

            {

                foreach (Detail detail in detailList)

                {

                    if(detail.Code.Equals(worker.DetailCode))

                    {

                        total += detail.Price \* worker.DetailProduced;

                    }

                }

            }

            return total;

        }

        /// <summary>

        /// Makes list of workers that fit the user defined preferences

        /// </summary>

        /// <param name="workers"></param>

        /// <param name="details"></param>

        /// <param name="pref1"></param>

        /// <param name="pref2"></param>

        /// <returns></returns>

        public static LinkedList<Worker> MakeListByPreferences(LinkedList<Worker> workers, LinkedList<Detail> details, string pref1, string pref2)

        {

            LinkedList<Worker> result = new LinkedList<Worker>();

            foreach (Worker worker in workers)

            {

                foreach(Detail detail in details)

                {

                    if(detail.Code == worker.DetailCode)

                    {

                        if (worker.DetailProduced > Convert.ToInt64(pref1))

                        {

                            if (detail.Price \* worker.DetailProduced < Convert.ToInt64(pref2))

                            {

                                result.Add(worker);

                            }

                        }

                    }

                }

            }

            return result;

        }

    }

}

**Forma.aspx failas:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma.aspx.cs" Inherits="LD2\_10\_MKuliesius.Forma" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <div>

            <div>

                <asp:Label ID="Label1" runat="server" CssClass="MainLabelTop" Font-Bold="True" Font-Size="XX-Large" Text="LD2 10"></asp:Label>

            </div>

        </div>

        <div>

            <asp:Label ID="Label2" runat="server" Text="Darbuotojai: "></asp:Label>

            <asp:FileUpload ID="FileUpload1" runat="server" />

            <br />

            <br />

            <asp:Label ID="Label3" runat="server" Text="Detales: "></asp:Label>

            <asp:FileUpload ID="FileUpload2" runat="server" />

            <br />

            <br />

            <asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Text="Skaityti" CausesValidation="False" BackColor="#99FF99" />

            <br />

            <asp:Label ID="Label4" runat="server"></asp:Label>

            <br />

            <br />

            <asp:Label ID="Label5" runat="server"></asp:Label>

            <br />

            <asp:TextBox ID="TextBox1" runat="server" TextMode="MultiLine" Height="200px" Width="800px"></asp:TextBox>

            <br />

            <asp:Label ID="Label6" runat="server"></asp:Label>

            <br />

            <asp:TextBox ID="TextBox2" runat="server" TextMode="MultiLine" Width="800px" Height="150px"></asp:TextBox>

            <br />

            <asp:Label ID="Label9" runat="server"></asp:Label>

            <br />

            <br />

            <asp:Label ID="Label7" runat="server"></asp:Label>

            <br />

            <asp:TextBox ID="TextBox3" runat="server" TextMode="MultiLine" Height="150px" Width="800px"></asp:TextBox>

            <br />

            <br />

            <asp:Label ID="Label8" runat="server"></asp:Label>

            <asp:TextBox ID="TextBox5" runat="server" Visible="True"></asp:TextBox>

            <br />

            <asp:Button ID="Button2" runat="server" Text="Atrinkti" Visible="True" OnClick="Button2\_Click" BackColor="#99FF99" ForeColor="Black" />

            <br />

            <br />

            <br />

            <asp:Label ID="Label10" runat="server"> Iveskite kieki</asp:Label>

            <br />

            <asp:TextBox ID="TextBox4" runat="server" Visible="True"></asp:TextBox>

            <br />

            <asp:Label ID="Label11" runat="server">Iveskite ikaini</asp:Label>

            <br />

            <asp:TextBox ID="TextBox6" runat="server" Visible="True"></asp:TextBox>

            <br />

            <asp:Button ID="Button3" runat="server" Text="Atrinkti" Visible="True" OnClick="Button3\_Click" BackColor="#99FF99" ForeColor="Black" />

            <br />

            <br />

            <br />

         </div>

        </form>

    </body>

**Forma.aspx.cs failas:**

using LD2\_10\_MKuliesius.AppCode;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace LD2\_10\_MKuliesius

{

    public partial class Forma : 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)

            {

                File.Delete(Server.MapPath("AppData/Rezultatai.txt"));

                string path1 = Server.MapPath(FileUpload1.FileName);

                string path2 = Server.MapPath(FileUpload2.FileName);

                string file1 = Server.HtmlEncode(FileUpload1.FileName);

                string extension1 = Path.GetExtension(file1);

                string file2 = Server.HtmlEncode(FileUpload2.FileName);

                string extension2 = Path.GetExtension(file2);

                // Checks if files are .txt

                if (extension1.Equals(".txt") && extension2.Equals(".txt"))

                {

                    // Saving files

                    FileUpload1.SaveAs(path1);

                    FileUpload2.SaveAs(path2);

                    // Status update

                    Label4.ForeColor = Color.Green;

                    Label4.Text = "Failai sukelti teisingai.";

                    //Read Files

                    AppCode.LinkedList<Worker> workers = InOut.ReadWorkerFile(path1);

                    AppCode.LinkedList<Detail> details = InOut.ReadDetailFile(path2);

                    Session["initial1"] = workers;

                    Session["initial2"] = details;

                    //path to result file.

                    string pathResults = Server.MapPath("AppData/Rezultatai.txt");

                    Session["Results"] = pathResults;

                    //Output initial data to file.

                    InOut.WriteInitialWorkerData(pathResults, "Pradiniai darbuotoju duomenys", workers);

                    InOut.WriteInitialDetailData(pathResults, "Pradiniai detaliu duomenys", details);

                    //Output initial data to textboxes.

                    Label5.Text = "Pradiniai darbuotoju duomenys";

                    DisplayInTextBox(InOut.WriteInitialWorkerDataList("Pradiniai darbuotoju duomenys", workers), TextBox1);

                    Label6.Text = "Pradiniai Detaliu duomenys";

                    DisplayInTextBox(InOut.WriteInitialDetailDataList("Pradiniai detaliu duomenys", details), TextBox2);

                    // Daugiausiai uzdirbusio darbuotojo paieska, darbo dienu skaicius, detaliu kiekis bei viso uzdirbta suma

                    //Pirma sugeneruojam atskira sarasa su unikaliais darbuotojais, kuriame suskaiciuosime kiekvieno darbuotojo uzdirbtus pinigus, visas detales bei darbo dienu skaiciu

                    AppCode.LinkedList<Worker> UniqueWorkers = TaskUtils.MakeSepparateWorkerList(workers, details);

                    TaskUtils.UpdateWorkers(UniqueWorkers, workers, details);

                    Worker BestWorker = TaskUtils.BestWorker(UniqueWorkers);

                    Label9.Text= "Geriausias darbuotojas: "+ BestWorker.ToString();

                    DisplayInTextBox(InOut.WriteInitialWorkerDataList("Unikalus darbuotojai", UniqueWorkers), TextBox3);

                }

            }

        }

        protected void Button2\_Click(object sender, EventArgs e)

        {

            AppCode.LinkedList<Worker> workers = (AppCode.LinkedList<Worker>)Session["Initial1"];

            AppCode.LinkedList<Detail> details = (AppCode.LinkedList<Detail>)Session["Initial2"];

            AppCode.LinkedList<Worker> partWorkers = TaskUtils.MakeWorkerListByPartName(workers, TextBox5.Text, details);

            string pathResults = (string)Session["Results"];

            partWorkers.Sort();

            DisplayInTextBox(InOut.WriteInitialWorkerDataList("Darbuotojai nustatytai detalei " + TextBox5.Text, partWorkers), TextBox3);

            TextBox3.Text = TextBox3.Text + "\n" + "Viso pagaminta detaliu:" + TaskUtils.CalculateTotalParts(partWorkers) +

                "\n" + "Is viso uzdirbta:" + TaskUtils.CalculateTotalEarned(partWorkers, details);

        }

        protected void Button3\_Click(object sender, EventArgs e)

        {

            string path1 = Server.MapPath("U10a.txt");

            AppCode.LinkedList<Worker> workers = (AppCode.LinkedList<Worker>)Session["Initial1"];

            AppCode.LinkedList<Detail> details = (AppCode.LinkedList<Detail>)Session["Initial2"];

            AppCode.LinkedList<Worker> PreferenceWorkers = TaskUtils.MakeListByPreferences(workers, details, TextBox4.Text, TextBox6.Text);

            PreferenceWorkers.Sort();

            InOut.WriteInitialWorkerData(path1, "Atrinktu pagal kriterijus darbuotoju sarasas", PreferenceWorkers);

            //textbox4 ir textbox6

        }

        protected void DisplayInTextBox(List<string> list, TextBox textBox)

        {

            StringBuilder sb = new StringBuilder();

            foreach (string item in list)

            {

                sb.Append(item);

            }

            textBox.Text = sb.ToString();

        }

    }

 }

## Pradiniai duomenys ir rezultatai

Testavimam sukūriau 3 darbuotojų duomenų failus bei 1 detalių informacijos failą.

Pirmasis darbuotojų failas:

U10a1.txt:

2024-04-01;Smith John;A123;50

2024-04-01;Smith John;B456;30

2024-04-01;Johnson Alice;A123;40

2024-04-02;Smith John;A123;60

2024-04-02;Johnson Alice;C789;20

2024-04-03;Johnson Alice;B456;25

2024-04-03;Johnson Alice;A123;35

Antrasis darbuotojų failas:

U10a2.txt

2024-04-01;Brown Bob;A123;40

2024-04-01;Brown Bob;B456;30

2024-04-01;Brown Bob;C789;20

2024-04-02;Brown Bob;A123;50

2024-04-02;Brown Bob;C789;15

2024-04-02;White Wendy;B456;35

2024-04-03;White Wendy;A123;25

2024-04-03;White Wendy;C789;30

Trečiasis darbuotojų failas:

U10a3.txt

2024-04-01;Green Gary;A123;60

2024-04-01;Green Gary;B456;20

2024-04-01;Green Gary;C789;40

2024-04-02;Green Gary;B456;25

2024-04-02;Green Gary;A123;45

2024-04-02;Green Gary;C789;30

2024-04-03;Green Gary;A123;35

2024-04-03;Green Gary;B456;30

Detalių failas:

U10b.txt

A123;DetaleA;10.50

B456;DetaleB;15.75

C789;DetaleC;20.00

**Testavimas:**

* Pirmasis testas

Pirmam testui turime du asmenis: Smith John ir Johnson Alice. John dirbo 3 dienas, o Alice dirbo 4 dienas. Kaip matoma duomenų failuose viršuje, naudojame pirmą darbuotojų testavimo failą.

Paleidus programą ir atidarius failus gauname tokį vaizdą:

A screenshot of a computer

Description automatically generated

Kaip matome, apskaičiavo kuris darbuotojas geriausiai pasirodė darbe, atvaizdavo pradinius duomenis, bei atrinko unikalius darbuotojus.

Toliau pasirenkame pagal kokią detalę norime filtruoti ir šio testo atveju pasirenkame DetaleA:

A screenshot of a computer

Description automatically generated

Kaip matome, atrinko visus iš sąrašo darbuotojus, kurie gamino šią dalį, parodo datą kada dirbo, surikiavo iš eilės darbuotojus. Suskaičiavo kiek iš viso detalių pagamino bei kiek uždirbo iš gamybos

Žemiau esančiuose tekstiniuose laukuose pasirenkame norimus kriterijus kaip atsirinkti darbuotojus:

A white background with black and white clouds

Description automatically generated

Atrenkame, kad pagamintų daugiau nei 20 detalių ir kainuotų mažiau nei 500.

Atrinkus, išveda rezultatus į failą U10a.txt:

Atrinktu pagal kriterijus darbuotoju sarasas

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Johnson Alice | A123 | 40 | 3 | 120 | 1260.00 |

| 2024-04-03 | Johnson Alice | B456 | 25 | 0 | 25 | 262.50 |

| 2024-04-03 | Johnson Alice | A123 | 35 | 0 | 35 | 367.50 |

| 2024-04-01 | Smith John | B456 | 30 | 0 | 30 | 315.00 |

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

Rezultatų failas:

Pradiniai darbuotoju duomenys

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Smith John | A123 | 50 | 0 | 0 | 0 |

| 2024-04-01 | Smith John | B456 | 30 | 0 | 0 | 0 |

| 2024-04-01 | Johnson Alice | A123 | 40 | 0 | 0 | 0 |

| 2024-04-02 | Smith John | A123 | 60 | 0 | 0 | 0 |

| 2024-04-02 | Johnson Alice | C789 | 20 | 0 | 0 | 0 |

| 2024-04-03 | Johnson Alice | B456 | 25 | 0 | 0 | 0 |

| 2024-04-03 | Johnson Alice | A123 | 35 | 0 | 0 | 0 |

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

Pradiniai detaliu duomenys

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

| Pavadinimas | Det. Kodas | Kaina |

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

DetaleA | A123 | 10.50

DetaleB | B456 | 15.75

DetaleC | C789 | 20.00

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

* Antrasis testas

Antram testui turime du asmenis: Brown Bob ir White Wendy. Bob dirbo 2 dienas, o Wendy dirbo 2 dienas. Kaip matoma duomenų failuose viršuje, naudojame antrą darbuotojų testavimo failą.

Paleidus programą ir atidarius failus gauname tokį vaizdą:

A screenshot of a computer

Description automatically generated

Kaip matome, apskaičiavo kuris darbuotojas geriausiai pasirodė darbe, atvaizdavo pradinius duomenis, bei atrinko unikalius darbuotojus.

Toliau pasirenkame pagal kokią detalę norime filtruoti ir šio testo atveju pasirenkame DetaleA:

A screenshot of a computer

Description automatically generated

Kaip matome, atrinko visus iš sąrašo darbuotojus, kurie gamino šią dalį, parodo datą kada dirbo, surikiavo iš eilės darbuotojus. Suskaičiavo kiek iš viso detalių pagamino bei kiek uždirbo iš gamybos

Žemiau esančiuose tekstiniuose laukuose pasirenkame norimus kriterijus kaip atsirinkti darbuotojus:

A white background with black and white clouds

Description automatically generated

Atrenkame, kad pagamintų daugiau nei 20 detalių ir kainuotų mažiau nei 300.

Atrinkus, išveda rezultatus į failą U10a.txt:

Atrinktu pagal kriterijus darbuotoju sarasas

Atrinktu pagal kriterijus darbuotoju sarasas

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-03 | White Wendy | A123 | 25 | 0 | 25 | 393.75 |

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

Rezultatų failas:

Pradiniai darbuotoju duomenys

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Brown Bob | A123 | 40 | 0 | 0 | 0 |

| 2024-04-01 | Brown Bob | B456 | 30 | 0 | 0 | 0 |

| 2024-04-01 | Brown Bob | C789 | 20 | 0 | 0 | 0 |

| 2024-04-02 | Brown Bob | A123 | 50 | 0 | 0 | 0 |

| 2024-04-02 | Brown Bob | C789 | 15 | 0 | 0 | 0 |

| 2024-04-02 | White Wendy | B456 | 35 | 0 | 0 | 0 |

| 2024-04-03 | White Wendy | A123 | 25 | 0 | 0 | 0 |

| 2024-04-03 | White Wendy | C789 | 30 | 0 | 0 | 0 |

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

Pradiniai detaliu duomenys

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

| Pavadinimas | Det. Kodas | Kaina |

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

DetaleA | A123 | 10.50

DetaleB | B456 | 15.75

DetaleC | C789 | 20.00

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

* Trečiasis testas

Trečiam testui turime vieną asmenį: Green Gary, kuris dirbo 3 dienas. Kaip matoma duomenų failuose viršuje, naudojame antrą darbuotojų testavimo failą.

Paleidus programą ir atidarius failus gauname tokį vaizdą:

A screenshot of a computer

Description automatically generated

Kaip matome, apskaičiavo kuris darbuotojas geriausiai pasirodė darbe, atvaizdavo pradinius duomenis, bei atrinko unikalius darbuotojus.

Toliau pasirenkame pagal kokią detalę norime filtruoti ir šio testo atveju pasirenkame DetaleA:

A close-up of a number

Description automatically generated

Kaip matome, atrinko visus iš sąrašo darbuotojus, kurie gamino šią dalį, parodo datą kada dirbo, surikiavo iš eilės darbuotojus. Suskaičiavo kiek iš viso detalių pagamino bei kiek uždirbo iš gamybos

Žemiau esančiuose tekstiniuose laukuose pasirenkame norimus kriterijus kaip atsirinkti darbuotojus:

A white background with black and white clouds

Description automatically generated

Atrenkame, kad pagamintų daugiau nei 30 detalių ir kainuotų mažiau nei 400.

Atrinkus, išveda rezultatus į failą U10a.txt:

Atrinktu pagal kriterijus darbuotoju sarasas

Atrinktu pagal kriterijus darbuotoju sarasas

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-03 | Green Gary | A123 | 35 | 0 | 35 | 367.50 |

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

Rezultatų failas:

Pradiniai darbuotoju duomenys

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Green Gary | A123 | 60 | 0 | 0 | 0 |

| 2024-04-01 | Green Gary | B456 | 20 | 0 | 0 | 0 |

| 2024-04-01 | Green Gary | C789 | 40 | 0 | 0 | 0 |

| 2024-04-02 | Green Gary | B456 | 25 | 0 | 0 | 0 |

| 2024-04-02 | Green Gary | A123 | 45 | 0 | 0 | 0 |

| 2024-04-02 | Green Gary | C789 | 30 | 0 | 0 | 0 |

| 2024-04-03 | Green Gary | A123 | 35 | 0 | 0 | 0 |

| 2024-04-03 | Green Gary | B456 | 30 | 0 | 0 | 0 |

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

Pradiniai detaliu duomenys

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

| Pavadinimas | Det. Kodas | Kaina |

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

DetaleA | A123 | 10.50

DetaleB | B456 | 15.75

DetaleC | C789 | 20.00

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

## Dėstytojo pastabos

Testukas: 0/3

Ataskaita: LD2 ataskaita: P4 (-0.2), P6 (-0.2), P7 (-0.2), P11 (-0.2), P13 (-0.2). Įv.: 0

Už programą: 6

# Bendrinės klasės ir testavimas (L3)

## Darbo užduotis

**LD\_10. Gamykla**. Gamykloje kiekvieną dieną registruojama informacija apie darbininkų pagamintas detales. Darbininkas gali gaminti per dieną skirtingo tipo detales. Suraskite daugiausiai uždirbusio darbininko pavardę, suskaičiuokite, kiek dienų jis dirbo, kiek iš viso detalių pagamino ir už kokią sumą. Sudarykite tik vieno pavadinimo detales gaminusių darbininkų sąrašą, pagamintų detalių skaičių ir sumą. Surikiuokite šį sąrašą pagal pavardes ir vardus. Duomenys: • Tekstiniame faile U10a.txt surašyta: data (metai, mėnuo, diena), darbininko pavardė ir vardas, detalės kodas, pagamintų vienetų skaičius. • Tekstiniame faile U10b.txt surašyta: detalės kodas, detalės pavadinimas, įkainis. Iš duomenų rinkinio faile U10a.txt sudarykite naują duomenų rinkinį pagal nurodytą požymį (pagamintų vienetų skaičius > S, įkainis < K, įvedami klaviatūra). Sąrašas turi būti surikiuotas pagal pavardes ir vardus abėcėlės tvarka.

## Grafinės vartotojo sąsajos schema

A screenshot of a computer

Description automatically generated

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Label1 | Text | LD2 10 |
| Label2 | Text | Darbuotojai |
| Label3 | Text | Detalės |
| Button1 | Text | Skaityti |
| Button2 | Text | Atrinkti |
| Button3 | Text | Atrinkti |
| Label10 | Text | Įveskite kiekį |
| Label11 | Text | Įveskite įkainį |

## Klasių diagrama

A screenshot of a computer program

Description automatically generated

## Programos vartotojo vadovas

Pasileidus programą atveriamas vaizdas, kuriame matome visus mygtukus su tekstiniais laukais. Paspaudžiame ant pirmo failo pasirinkimo ir pasirenkame darbuotojų informacijos failą. Toliau paspaudžiame ant antrojo failo pasirinkimo ir pasirenkame detalių informacijos failą. Tuomet spaudžiame Skaityti. Tuomet programa nuskaito failų informaciją ir atvaizduoja ją ekrane. Taigi atvaizduoja pradinius failus, suranda geriausią darbuotoją ir jį atvaizduoja ir taip pat atrenka ir surikiuoja unikalius darbuotojus su jų pilnom uždirbtom sumom. Tuomet galime pasirinkti rūšiavimą pagal detalę, ir tada lauke, kuriame rodė unikalius darbuotojus atvaizduoja atfiltruotus bei surikiuotus rezultatus. Žemiau esančiuose tekstiniuose laukuose prašoma užpildyti atrankos kriterijus, pagal kuriuos į failą „U10a.txt“ išves atrinktus bei surikiuotus pagal pavardę ir vardą darbuotojus.

## Programos tekstas

**Node.cs failas:**

namespace LD2\_10\_MKuliesius.AppCode

{

    public class Node<T>

    {

        #region private variables

        //Aprašomi privatūs Node kintamieji

        private T data;

        private Node<T> next;

        #endregion

        #region Contructors

        /// <summary>

        /// Node konstruktorius

        /// </summary>

        /// <param name="data"> Node duomenys</param>

        /// <param name="next"> Sekantis Node</param>

        public Node(T data, Node<T> next)

        {

            this.data = data;

            this.next = next;

        }

        #endregion

        #region Properties

        /// <summary>

        /// Data property

        /// </summary>

        public T Data

        {

            get { return this.data; }

            set { this.data = value; }

        }

        /// <summary>

        /// Sekančio Node Property

        /// </summary>

        public Node<T> Next

        {

            get { return this.next; }

            set { this.next = value; }

        }

        #endregion

        #region Methods

        /// <summary>

        /// Gražina Node informaciją string forma

        /// </summary>

        public override string ToString()

        {

            return data.ToString();

        }

        #endregion

    }

}

**LinkedList.cs failas:**

using System;

using System.Collections;

using System.Collections.Generic;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class LinkedList<T> : IEnumerable<T> where T : IComparable<T>

    {

        private Node<T> Head;

        private int count;

        public IEnumerator<T> GetEnumerator()

        {

            for (Node<T> current = Head; current != null; current = current.Next)

            {

                yield return current.Data;

            }

        }

        IEnumerator IEnumerable.GetEnumerator()

        {

            return GetEnumerator();

        }

        public LinkedList()

        {

            Head = null;

            count = 0;

        }

        public bool Empty => count == 0;

        public int Count => count;

        public void Add(T obj)

        {

            AddToEnd(obj);

        }

        private void AddToEnd(T obj)

        {

            Node<T> newNode = new Node<T>(obj, null);

            if (Empty)

            {

                Head = newNode;

            }

            else

            {

                GetLastNode().Next = newNode;

            }

            count++;

        }

        /// <summary>

        /// Sorts the list ascending

        /// </summary>

        public void Sort()

        {

            // If the list is empty or contains only one element, it's already sorted

            if (Empty || Head.Next == null)

            {

                return;

            }

            bool swapped;

            do

            {

                swapped = false;

                Node<T> current = Head;

                Node<T> previous = null;

                while (current.Next != null)

                {

                    if (current.Data.CompareTo(current.Next.Data) > 0)

                    {

                        // Swap data of current node and next node

                        T temp = current.Data;

                        current.Data = current.Next.Data;

                        current.Next.Data = temp;

                        swapped = true;

                    }

                    previous = current;

                    current = current.Next;

                }

            } while (swapped);

        }

        /// <summary>

        /// Returns the indexed information.

        /// </summary>

        /// <param name="index"></param>

        /// <returns></returns>

        /// <exception cref="ArgumentOutOfRangeException"></exception>

        public T Get(int index)

        {

            if (index < 0 || index >= count)

            {

                throw new ArgumentOutOfRangeException(nameof(index), "Index is out of range.");

            }

            Node<T> current = Head;

            for (int i = 0; i < index; i++)

            {

                current = current.Next;

            }

            return current.Data;

        }

        /// <summary>

        /// Gets the first element that matches the specified condition.

        /// </summary>

        /// <param name="predicate">The condition to match.</param>

        /// <returns>The first element that matches the condition, or null if no such element is found.</returns>

        public T Get(Func<T, bool> predicate)

        {

            for (Node<T> current = Head; current != null; current = current.Next)

            {

                if (predicate(current.Data))

                {

                    return current.Data;

                }

            }

            return default; // Return default value if no element matches the condition

        }

        private Node<T> GetLastNode()

        {

            Node<T> current = Head;

            while (current.Next != null)

            {

                current = current.Next;

            }

            return current;

        }

    }

}

**Worker.cs failas:**

using System;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class Worker : IComparable<Worker>

    {

        public string Name { get; set; }

        public string Date { get; set; }

        public string DetailCode { get; set; }

        public int DetailProduced { get; set; }

        public int TotalDaysWorked { get; set; }

        public int TotalDetailsProduced { get; set; }

        public decimal TotalEarnings { get; set; }

        public Worker(string name, string date, string detailCode, int detailProduced)

        {

            Name = name;

            Date = date;

            DetailCode = detailCode;

            DetailProduced = detailProduced;

            TotalDaysWorked = 0;

            TotalDetailsProduced = 0;

            TotalEarnings = 0;

        }

        public Worker() { }

        public int CompareTo(Worker other)

        {

            return this.Name.CompareTo(other.Name);

        }

        public int CompareTo(decimal sum)

        {

            return this.TotalEarnings.CompareTo(sum);

        }

        // Override ToString method to provide meaningful string representation

        public override string ToString()

        {

            return $"| {Date, 10} | {Name, 20} | {DetailCode, 5} | {DetailProduced, 4} | {TotalDaysWorked,4} | {TotalDetailsProduced,5} | {TotalEarnings,4} |";

        }

    }

}

**Detail.cs failas:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class Detail : IComparable<Detail>

    {

        public string Code { get; private set; }

        public string Name { get; set; }

        public decimal Price { get; set; }

        public Detail(string code, string name, decimal price)

        {

            Code = code;

            Name = name;

            Price = price;

        }

        public int CompareTo(Detail other)

        {

            return Price.CompareTo( other.Price);

        }

        // Override ToString method to provide meaningful string representation

        public override string ToString()

        {

            return $"{Name, 10} | {Code, 12} | {Price, 8}";

        }

    }

}

**InOut.cs failas:**

using System.Collections.Generic;

using System.IO;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class InOut

    {

        /// <summary>

        /// Reads workers information

        /// </summary>

        /// <param name="fileName"> data file</param>

        /// <returns></returns>

        public static LinkedList<Worker> ReadWorkerFile(string fileName)

        {

            LinkedList<Worker> workers = new LinkedList<Worker>();

            string[] lines = File.ReadAllLines(fileName);

            foreach (string line in lines)

            {

                string[] parts = line.Split(';');

                if (parts.Length == 4) // Assuming each line has 4 parts

                {

                    string dateStr = parts[0].Trim();

                    string workerName = parts[1].Trim();

                    string detailCode = parts[2].Trim();

                    int quantity = int.Parse(parts[3].Trim());

                    Worker worker = new Worker(workerName, dateStr, detailCode, quantity);

                    // Add the worker to the linked list

                    workers.Add(worker);

                }

            }

            return workers;

        }

        /// <summary>

        /// Reads details info

        /// </summary>

        /// <param name="fileName"> Data file</param>

        /// <returns></returns>

        public static LinkedList<Detail> ReadDetailFile(string fileName)

        {

            LinkedList<Detail> details = new LinkedList<Detail>();

            string[] lines = File.ReadAllLines(fileName);

            foreach (string line in lines)

            {

                string[] parts = line.Split(';');

                if(parts.Length == 3)

                {

                    string code = parts[0].Trim();

                    string name = parts[1].Trim();

                    decimal price = decimal.Parse(parts[2].Trim());

                    Detail det = new Detail(code, name, price);

                    details.Add(det);

                }

            }

            return details;

        }

        /// <summary>

        /// Writes initial Worker detail

        /// </summary>

        /// <param name="fileName"></param>

        /// <param name="header"></param>

        /// <param name="list"></param>

        public static void WriteInitialWorkerData(string fileName, string header, LinkedList<Worker> list)

        {

            string dashes = new string('-', 89);

            List<string> lines = new List<string>

            {

                header,

                dashes,

                string.Format($"| {"Data", -10} | {"Vardas", -20} | {"Det. Kodas", 5} | {"Pagamino", 4} | {"Viso dirbo", 4} | {"Viso pagamino", 4} | {"Viso uzdirbo", 4} |"),

                dashes

            };

            foreach (Worker worker in list)

            {

                lines.Add(worker.ToString() + "\n");

            }

            lines.Add(dashes);

            lines.Add("\n");

            File.AppendAllLines(fileName, lines);

        }

        /// <summary>

        /// Writes initial data of detail

        /// </summary>

        /// <param name="fileName"></param>

        /// <param name="header"></param>

        /// <param name="list"></param>

        public static void WriteInitialDetailData(string fileName, string header, LinkedList<Detail> list)

        {

            string dashes = new string('-', 60);

            List<string> lines = new List<string>

            {

                header,

                dashes,

                string.Format($"| {"Pavadinimas",-15} | {"Det. Kodas", 12} | {"Kaina", 8} |"),

                dashes

            };

            foreach (Detail detail in list)

            {

                lines.Add(detail.ToString());

            }

            lines.Add(dashes);

            lines.Add("\n");

            File.AppendAllLines(fileName, lines);

        }

        /// <summary>

        /// Copies the existing result file.

        /// </summary>

        /// <param name="fileName">File name</param>

        /// <returns>An array of lines</returns>

        public static string[] SaveCurrentResultFile(string fileName)

        {

            string[] lines = File.ReadAllLines(fileName);

            return lines;

        }

        public static List<string> WriteInitialWorkerDataList( string header, LinkedList<Worker> list)

        {

            string dashes = new string('-', 89);

            List<string> lines = new List<string>

            {

                header+"\n",

                dashes + "\n",

                string.Format($"| {"Data", -10} | {"Vardas", -20} | {"Det. Kodas", 5} | {"Pagamino", 4} | {"Viso dirbo", 4} | {"Viso pagamino", 4} | {"Viso uzdirbo", 4} |")+"\n",

                dashes+"\n"

            };

            foreach (Worker worker in list)

            {

                lines.Add(worker.ToString()+"\n");

            }

            lines.Add(dashes+"\n");

            lines.Add("\n");

            return lines;

        }

        public static List<string> WriteInitialDetailDataList( string header, LinkedList<Detail> list)

        {

            string dashes = new string('-', 60);

            List<string> lines = new List<string>

            {

                header+ "\n",

                dashes+ "\n",

                string.Format($"| {"Pavadinimas",-15} | {"Det. Kodas", 12} | {"Kaina", 8} |")+ "\n",

                dashes+ "\n"

            };

            foreach (Detail detail in list)

            {

                lines.Add(detail.ToString() + "\n");

            }

            lines.Add(dashes + "\n");

            lines.Add("\n");

            return lines;

        }

    }

}

**TableUtils failas:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI.WebControls;

namespace LD3\_10\_MKuliesius

{

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

    {

        /// <summary>

        /// Makes a table with 7 cells for workers.

        /// </summary>

        /// <param name="cell1">A cell</param>

        /// <param name="cell2">A cell</param>

        /// <param name="cell3">A cell</param>

        /// <param name="cell4">A cell</param>

        /// <param name="cell5">A cell</param>

        /// <param name="cell6">A cell</param>

        /// <param name="cell7">A cell</param>

        /// <returns> A Row of cells</returns>

        public TableRow MakeRow(string cell1, string cell2, string cell3, string cell4, string cell5, string cell6, string cell7)

        {

            TableRow row = new TableRow();

            TableCell Cell1 = new TableCell();

            Cell1.Text = cell1;

            row.Cells.Add(Cell1);

            TableCell Cell2 = new TableCell();

            Cell2.Text = cell2;

            row.Cells.Add(Cell2);

            TableCell Cell3 = new TableCell();

            Cell3.Text = cell3;

            row.Cells.Add(Cell3);

            TableCell Cell4 = new TableCell();

            Cell4.Text = cell4;

            row.Cells.Add(Cell4);

            TableCell Cell5 = new TableCell();

            Cell5.Text = cell5;

            row.Cells.Add(Cell5);

            TableCell Cell6 = new TableCell();

            Cell6.Text = cell6;

            row.Cells.Add(Cell6);

            TableCell Cell7 = new TableCell();

            Cell7.Text = cell7 + " e";

            row.Cells.Add(Cell7);

            return row;

        }

        /// <summary>

        /// Makes a table with 3 cells for parts.

        /// </summary>

        /// <param name="cell1">A cell</param>

        /// <param name="cell2">A cell</param>

        /// <param name="cell3">A cell</param>

        /// <returns> A Row of cells</returns>

        public TableRow MakeRow(string cell1, string cell2, string cell3)

        {

            TableRow row = new TableRow();

            TableCell Cell1 = new TableCell();

            Cell1.Text = cell1;

            row.Cells.Add(Cell1);

            TableCell Cell2 = new TableCell();

            Cell2.Text = cell2;

            row.Cells.Add(Cell2);

            TableCell Cell3 = new TableCell();

            Cell3.Text = cell3+" e";

            row.Cells.Add(Cell3);

            return row;

        }

        /// <summary>

        /// Makes a table with 4 cells for workers.

        /// </summary>

        /// <param name="cell1">A cell</param>

        /// <param name="cell2">A cell</param>

        /// <param name="cell3">A cell</param>

        /// <param name="cell4">A cell</param>

        /// <returns> A Row of cells</returns>

        public TableRow MakeRow(string cell1, string cell2, string cell3, string cell4)

        {

            TableRow row = new TableRow();

            TableCell Cell1 = new TableCell();

            Cell1.Text = cell1;

            row.Cells.Add(Cell1);

            TableCell Cell2 = new TableCell();

            Cell2.Text = cell2;

            row.Cells.Add(Cell2);

            TableCell Cell3 = new TableCell();

            Cell3.Text = cell3;

            row.Cells.Add(Cell3);

            TableCell Cell4 = new TableCell();

            Cell4.Text = cell4;

            row.Cells.Add(Cell4);

            return row;

        }

        /// <summary>

        /// Makes a table with 1 cells for workers.

        /// </summary>

        /// <param name="cell1">A cell</param>

        /// <returns> A Row of cells</returns>

        public TableRow MakeRow(string cell1)

        {

            TableRow row = new TableRow();

            TableCell Cell1 = new TableCell();

            Cell1.Text = cell1;

            row.Cells.Add(Cell1);

            return row;

        }

        /// <summary>

        /// Creates a table of workers starting data

        /// </summary>

        /// <param name="workers">List of workers</param>

        public void MakeWorkerTableStart(LinkedList<Worker> workers)

        {

            foreach (Worker worker in workers)

            {

                if (worker != null)

                {

                   Table1.Rows.Add(MakeRow(worker.Date, worker.Name, worker.DetailCode, worker.DetailProduced.ToString()));

                }

            }

        }

        /// <summary>

        /// Creates a table of Details data

        /// </summary>

        /// <param name="workers">List of details</param>

        public void MakeDetailTable(LinkedList<Detail> details)

        {

            foreach (Detail detail in details)

            {

                if (detail != null)

                {

                    Table2.Rows.Add(MakeRow(detail.Name, detail.Code, detail.Price.ToString()));

                }

            }

        }

        /// <summary>

        /// Creates a table of workers data after calculations

        /// </summary>

        /// <param name="workers">List of workers</param>

        public void MakeWorkerTable(LinkedList<Worker> workers)

        {

            foreach (Worker worker in workers)

            {

                if (worker != null)

                {

                    Table3.Rows.Add(MakeRow(worker.Date, worker.Name, worker.DetailCode, worker.DetailProduced.ToString(), worker.TotalDaysWorked.ToString(), worker.TotalDetailsProduced.ToString(), worker.TotalEarnings.ToString()));

                }

            }

        }

    }

}

**TaskUtils failas:**

using System;

namespace LD2\_10\_MKuliesius.AppCode

{

    public class TaskUtils

    {

        public static decimal CalculateEarningsForPart(LinkedList<Worker> workerList, LinkedList<Detail> detailList, string desiredWorker, string desiredPart)

        {

            decimal earnings = 0;

            foreach(Detail detail in detailList)

            {

                if(detail.Code == desiredPart)

                {

                    foreach(Worker worker in workerList)

                    {

                        if(worker.Name == desiredWorker)

                        {

                            earnings += worker.DetailProduced \* detail.Price;

                        }

                        // worker.TotalEarnings += earnings;

                    }

                }

            }

            return earnings;

        }

        /// <summary>

        /// Makes sepparate list with unique workers

        /// </summary>

        /// <param name="workerList"></param>

        /// <returns></returns>

        public static LinkedList<Worker> MakeSepparateWorkerList(LinkedList<Worker> workerList, LinkedList<Detail> detailList)

        {

            LinkedList<Worker> result = new LinkedList<Worker>();

            foreach (Worker worker in workerList)

            {

                if (!IsInList(result, worker))

                {

                    result.Add(worker);

                }

            }

            return result;

        }

        /// <summary>

        /// Updates workers information in list

        /// </summary>

        /// <param name="Uniqueworkers"></param>

        /// <param name="workers"></param>

        /// <param name="detailList"></param>

        public static void UpdateWorkers(LinkedList<Worker> Uniqueworkers,LinkedList<Worker>workers, LinkedList<Detail> detailList)

        {

            foreach (Worker worker in Uniqueworkers)

            {

                worker.TotalEarnings = TotalEarnings(workers, worker, detailList);

                //decimal total = TotalEarnings(workers, worker, detailList);

                worker.TotalDaysWorked = TotalWorkDays(workers, worker);

                worker.TotalDetailsProduced = TotalDetails(workers, worker);

            }

        }

        /// <summary>

        /// Checks if worker is in list

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <returns></returns>

        public static bool IsInList(LinkedList<Worker> workerList, Worker worker)

        {

            if (workerList.Empty)

            {

                return false;

            }

            else

            {

                foreach (Worker worker1 in workerList)

                {

                    if (worker.Name.Equals(worker1.Name))

                    {

                        return true;

                    }

                }

            }

            return false;

        }

        /// <summary>

        /// Calculates the total ammount of earnings for the worker

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <param name="detailList"></param>

        /// <returns></returns>

        public static decimal TotalEarnings(LinkedList<Worker> workerList, Worker worker, LinkedList<Detail> detailList)

        {

            decimal total = 0;

            foreach (Worker worker1 in workerList)

            {

                if (worker1.Name.Equals(worker.Name))

                {

                    foreach (Detail detail in detailList)

                    {

                        if (detail.Code.Equals(worker.DetailCode))

                        {

                            worker1.TotalEarnings += worker1.DetailProduced \* detail.Price;

                            total += worker1.DetailProduced \* detail.Price;

                        }

                    }

                }

            }

            return total;

        }

        /// <summary>

        /// Calculates how many details worker has made in total

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <returns></returns>

        public static int TotalDetails(LinkedList<Worker> workerList, Worker worker)

        {

            int total = 0;

            foreach (Worker worker1 in workerList)

            {

                if (worker1.Name.Equals(worker.Name))

                {

                    total+=worker1.DetailProduced;

                    worker1.TotalDetailsProduced += worker1.DetailProduced;

                }

            }

            return total;

        }

        /// <summary>

        /// Calculates how many work days worker worked in total

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="worker"></param>

        /// <returns></returns>

        public static int TotalWorkDays(LinkedList<Worker> workerList, Worker worker)

        {

            int total = 0;

            string date = "";

            foreach (Worker worker1 in workerList)

            {

                if (worker1.Name.Equals(worker.Name) && !worker1.Date.Equals(date))

                {

                    total++;

                    date = worker1.Date;

                }

            }

            return total;

        }

        /// <summary>

        /// finds best worker from unique workers list

        /// </summary>

        /// <param name="workerList"></param>

        /// <returns></returns>

        public static Worker BestWorker(LinkedList<Worker> workerList)

        {

            Worker worker = new Worker();

            decimal max = 0;

            foreach (Worker worker1 in workerList)

            {

                if (worker1.TotalEarnings.CompareTo(max) >0)

                {

                    max = worker1.TotalEarnings;

                    worker = worker1;

                }

            }

            return worker;

        }

        /// <summary>

        /// Makes a list of workers that worked on a certain part

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="partName"></param>

        /// <param name="detailList"></param>

        /// <returns></returns>

        public static LinkedList<Worker> MakeWorkerListByPartName(LinkedList<Worker> workerList, string partName, LinkedList<Detail> detailList)

        {

            LinkedList<Worker> result = new LinkedList<Worker>();

            foreach(Detail detail in detailList)

            {

                if (detail.Name.Equals(partName))

                {

                    string partCode = detail.Code;

                    foreach (Worker worker1 in workerList)

                    {

                        if(worker1.DetailCode.Equals(partCode))

                        {

                            result.Add(worker1);

                        }

                    }

                }

            }

            return result;

        }

        /// <summary>

        /// calculates total parts made for list.

        /// </summary>

        /// <param name="workerList"></param>

        /// <returns></returns>

        public static int CalculateTotalParts(LinkedList<Worker> workerList)

        {

            int total = 0;

            foreach (Worker worker in workerList)

            {

                total += worker.DetailProduced;

            }

            return total;

        }

        /// <summary>

        /// returns total ammount of money earned

        /// </summary>

        /// <param name="workerList"></param>

        /// <param name="detailList"></param>

        /// <returns></returns>

        public static decimal CalculateTotalEarned(LinkedList<Worker> workerList, LinkedList<Detail> detailList)

        {

            decimal total = 0;

            foreach (Worker worker in workerList)

            {

                foreach (Detail detail in detailList)

                {

                    if(detail.Code.Equals(worker.DetailCode))

                    {

                        total += detail.Price \* worker.DetailProduced;

                    }

                }

            }

            return total;

        }

        /// <summary>

        /// Makes list of workers that fit the user defined preferences

        /// </summary>

        /// <param name="workers"></param>

        /// <param name="details"></param>

        /// <param name="pref1"></param>

        /// <param name="pref2"></param>

        /// <returns></returns>

        public static LinkedList<Worker> MakeListByPreferences(LinkedList<Worker> workers, LinkedList<Detail> details, string pref1, string pref2)

        {

            LinkedList<Worker> result = new LinkedList<Worker>();

            foreach (Worker worker in workers)

            {

                foreach(Detail detail in details)

                {

                    if(detail.Code == worker.DetailCode)

                    {

                        if (worker.DetailProduced > Convert.ToInt64(pref1))

                        {

                            if (detail.Price \* worker.DetailProduced < Convert.ToInt64(pref2))

                            {

                                result.Add(worker);

                            }

                        }

                    }

                }

            }

            return result;

        }

/// <summary>

        /// Checks if inputed city contains only letters and whitespaces

        /// </summary>

        /// <param name="text">The input</param>

        /// <returns>True if input only contains letters and whitespaces</returns>

        public static bool Validation(string text)

        {

            foreach (char simb in text)

            {

                if (!char.IsLetter(simb) && !char.IsWhiteSpace(simb))

                {

                    return false;

                }

            }

            if (text.Length > 0)

            {

                return true;

            }

            return false;

        }

    }

**Forma.aspx failas:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma.aspx.cs" Inherits="LD3\_10\_MKuliesius.Forma" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<div>

<asp:Label ID="Label1" runat="server" CssClass="MainLabelTop" Font-Bold="True" Font-Size="XX-Large" Text="LD3 10"></asp:Label>

</div>

</div>

<div>

<asp:Label ID="Label2" runat="server" Text="Darbuotojai: "></asp:Label>

<asp:FileUpload ID="FileUpload1" runat="server" />

<br />

<br />

<asp:Label ID="Label3" runat="server" Text="Detales: "></asp:Label>

<asp:FileUpload ID="FileUpload2" runat="server" />

<br />

<br />

<asp:Button ID="Button1" runat="server" OnClick="Button1\_Click" Text="Skaityti" CausesValidation="False" BackColor="#99FF99" />

<br />

<asp:Label ID="Label4" runat="server"></asp:Label>

<br />

<br />

<asp:Label ID="Label5" runat="server"></asp:Label>

<br />

<br />

<asp:Table ID="Table1" runat="server" BorderWidth="1px" Visible="false" BackColor="Wheat"></asp:Table>

<br />

<asp:Table ID="Table2" runat="server" BorderWidth="1px" Visible="false" BackColor="Wheat"></asp:Table>

<br />

<br />

<asp:Label ID="Label6" runat="server"></asp:Label>

<br />

<asp:Table ID="Table3" runat="server" BorderWidth="1px" Visible="false" BackColor="Wheat"></asp:Table>

<br />

<br />

<br />

<br />

<asp:Label ID="Label9" runat="server" Visible="false"></asp:Label>

<br />

<asp:Label ID="Label8" runat="server"></asp:Label>

<asp:TextBox ID="TextBox5" runat="server" Visible="True"></asp:TextBox>

<br />

<asp:Button ID="Button2" runat="server" Text="Atrinkti" Visible="True" OnClick="Button2\_Click" BackColor="#99FF99" ForeColor="Black" />

<br />

<br />

<br />

<asp:Label ID="Label10" runat="server"> Iveskite kieki</asp:Label>

<br />

<asp:TextBox ID="TextBox4" runat="server" Visible="True"></asp:TextBox>

<br />

<asp:Label ID="Label11" runat="server">Iveskite ikaini</asp:Label>

<br />

<asp:TextBox ID="TextBox6" runat="server" Visible="True"></asp:TextBox>

<br />

<asp:Button ID="Button3" runat="server" Text="Atrinkti" Visible="True" OnClick="Button3\_Click" BackColor="#99FF99" ForeColor="Black" />

<br />

<br />

<br />

</div>

</form>

</body>

</html>

**Forma.aspx.cs failas:**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Text;

using System.Web.UI.WebControls;

namespace LD3\_10\_MKuliesius

{

    public partial class Forma : 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)

            {

                File.Delete(Server.MapPath("AppData/Rezultatai.txt"));

                string path1 = Server.MapPath(FileUpload1.FileName);

                string path2 = Server.MapPath(FileUpload2.FileName);

                string file1 = Server.HtmlEncode(FileUpload1.FileName);

                string extension1 = Path.GetExtension(file1);

                string file2 = Server.HtmlEncode(FileUpload2.FileName);

                string extension2 = Path.GetExtension(file2);

                // Checks if files are .txt

                if (extension1.Equals(".txt") && extension2.Equals(".txt"))

                {

                    // Saving files

                    FileUpload1.SaveAs(path1);

                    FileUpload2.SaveAs(path2);

                    // Status update

                    Label4.ForeColor = Color.Green;

                    Label4.Text = "Failai sukelti teisingai.";

                    //Read Files

                    LinkedList<Worker> workers = InOut.ReadWorkerFile(path1);

                    LinkedList<Detail> details = InOut.ReadDetailFile(path2);

                    Session["initial1"] = workers;

                    Session["initial2"] = details;

                    //path to result file.

                    string pathResults = Server.MapPath("AppData/Rezultatai.txt");

                    Session["Results"] = pathResults;

                    //Output initial data to file.

                    InOut.WriteInitialWorkerData(pathResults, "Pradiniai darbuotoju duomenys", workers);

                    InOut.WriteInitialDetailData(pathResults, "Pradiniai detaliu duomenys", details);

                    //Output initial data to textboxes.

                    Label5.Text = "Įvesti pradiniai duomenys. 1 lentelė - Darbuotojai, 2 lentelė - detalių kainoraštis.";

                    Table1.Visible = true;

                    Table2.Visible = true;

                    Table3.Visible = true;

                    Label5.Font.Size = 26;

                    Label5.Font.Bold = true;

                    Table1.Rows.Add(MakeRow("<b>Data</b>", "<b>Vardas</b>", "<b>Det. Kodas</b>", "<b>Pagamino</b>"));

                    MakeWorkerTableStart(workers);

                    Table2.Rows.Add(MakeRow("<b>Pavadinimas</b>", "<b>Det. Kodas</b>", "<b>Kaina</b>"));

                    MakeDetailTable(details);

                    // Daugiausiai uzdirbusio darbuotojo paieska, darbo dienu skaicius, detaliu kiekis bei viso uzdirbta suma

                    //Pirma sugeneruojam atskira sarasa su unikaliais darbuotojais, kuriame suskaiciuosime kiekvieno darbuotojo uzdirbtus pinigus, visas detales bei darbo dienu skaiciu

                    LinkedList<Worker> UniqueWorkers = TaskUtils.MakeSepparateWorkerList(workers, details);

                    TaskUtils.UpdateWorkers(UniqueWorkers, workers, details);

                    Label6.Font.Size = 26;

                    Label6.Font.Bold = true;

                    Label6.Visible = true;

                    Label6.Text = "Unikalūs darbuotojai";

                    Table3.Rows.Add(MakeRow("<b>Data</b>", "<b>Vardas</b>", "<b>Det. Kodas</b>", "<b>Pagamino</b>", "<b>Viso dirbo</b>", "<b>Viso pagamino</b>", "<b>Viso uždirbo</b>"));

                    MakeWorkerTable(UniqueWorkers);

                    Worker BestWorker = TaskUtils.BestWorker(UniqueWorkers);

                    Table3.Rows.Add(MakeRow("Geriausias darbuotojas: "));

                    Table3.Rows.Add(MakeRow("", BestWorker.Name, BestWorker.DetailCode, "", BestWorker.TotalDaysWorked.ToString(), BestWorker.TotalDetailsProduced.ToString(), BestWorker.TotalEarnings.ToString()));

                    Label9.Visible = true;

                    Label9.Font.Size = 26;

                    Label9.Font.Bold = true;

                    Label9.Visible = true;

                    Label9.Text = "Įrašykite detalę, pagal kurią norite atrinkti darbuotojus (pavadinimą)";

                }

                else

                {

                    // At least one of the extensions isn't .txt

                    Label4.ForeColor = Color.Red;

                    Label4.Text = "Netinkamo tipo failai (tinka tik .txt).";

                }

            }

            else

            {

                // FileUpload.HasFile = false

                Label4.ForeColor = Color.Red;

                Label4.Text = "Nepasirinkti visi reikiami failai.";

            }

        }

        protected void Button2\_Click(object sender, EventArgs e)

        {

            //Uzkraunami is sesijos pradiniai duomenys

            LinkedList<Worker> workers = (LinkedList<Worker>)Session["Initial1"];

            LinkedList<Detail> details = (LinkedList<Detail>)Session["Initial2"];

            //Sukuriamas pagal norima atributa naudotoju sarasas

            LinkedList<Worker> partWorkers = TaskUtils.MakeWorkerListByPartName(workers, TextBox5.Text, details);

            string pathResults = (string)Session["Results"];

            Label5.Text = "Įvesti pradiniai duomenys. 1 lentelė - Darbuotojai, 2 lentelė - detalių kainoraštis.";

            Table1.Visible = true;

            Table2.Visible = true;

            Table3.Visible = true;

            Label5.Font.Size = 26;

            Label5.Font.Bold = true;

            Label6.Font.Size = 26;

            Label6.Font.Bold = true;

            Label6.Visible = true;

            Table1.Rows.Add(MakeRow("<b>Data</b>", "<b>Vardas</b>", "<b>Det. Kodas</b>", "<b>Pagamino</b>"));

            MakeWorkerTableStart(workers);

            Table2.Rows.Add(MakeRow("<b>Pavadinimas</b>", "<b>Det. Kodas</b>", "<b>Kaina</b>"));

            MakeDetailTable(details);

            if (TaskUtils.Validation(TextBox5.Text))

            {

                // Viskas įvesta teisingai.

                Label4.ForeColor = Color.Green;

                Label4.Text = "Paieškos laukai užpildyti teisingai.";

                Label6.Text = "Darbuotojai atrinkti bei surikiuoti pagal pasirinktą detalę: " + TextBox5.Text;

                partWorkers.Sort();

                Table3.Rows.Add(MakeRow("<b>Data</b>", "<b>Vardas</b>", "<b>Det. Kodas</b>", "<b>Pagamino</b>", "<b>Viso dirbo</b>", "<b>Viso pagamino</b>", "<b>Viso uždirbo</b>"));

                MakeWorkerTable(partWorkers);

                Table3.Rows.Add(MakeRow("Viso pagaminta detaliu:" + TaskUtils.CalculateTotalParts(partWorkers)));

                Table3.Rows.Add(MakeRow("Is viso uzdirbta:" + TaskUtils.CalculateEarningsForPart(partWorkers, details, TextBox5.Text)));

            }

            else

            {

                // FileUpload.HasFile = false

                Label4.ForeColor = Color.Red;

                Label4.Text = "Atrinkimui pagal detalę galima vesti tik raides ir tik vieną detalę!";

            }

        }

        protected void Button3\_Click(object sender, EventArgs e)

        {

            string path1 = Server.MapPath("U10a.txt");

            //Uzkraunami is sesijos pradiniai duomenys

            LinkedList<Worker> workers = (LinkedList<Worker>)Session["Initial1"];

            LinkedList<Detail> details = (LinkedList<Detail>)Session["Initial2"];

            //Sukuriamas atrinktu pagal kriteriju darbuotoju sarasas ir tolau surikiuojamas

            LinkedList<Worker> PreferenceWorkers = TaskUtils.MakeListByPreferences(workers, details, TextBox4.Text, TextBox6.Text);

            PreferenceWorkers.Sort();

            Label5.Visible = false;

            Label6.Visible = false;

            Button2.Visible = false;

            Label8.Visible = false;

            Label9.Text = "Atrinktu pagal kriterijus darbuotoju sarasas Isvestas i faila" + path1;

            InOut.WriteInitialWorkerData(path1, "Atrinktu pagal kriterijus darbuotoju sarasas", PreferenceWorkers);

            //textbox4 ir textbox6

        }

        /// <summary>

        /// Papildomas metodas isvedimo i tekstini langa supaprastinimui

        /// </summary>

        /// <param name="list"></param>

        /// <param name="textBox"></param>

        protected void DisplayInTextBox(List<string> list, TextBox textBox)

        {

            StringBuilder sb = new StringBuilder();

            foreach (string item in list)

            {

                sb.Append(item);

            }

            textBox.Text = sb.ToString();

        }

    }

 }

**Forma.aspx.designer.cs failas:**

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

// <auto-generated>

//     This code was generated by a tool.

//

//     Changes to this file may cause incorrect behavior and will be lost if

//     the code is regenerated.

// </auto-generated>

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

namespace LD3\_10\_MKuliesius

{

    public partial class Forma

    {

        /// <summary>

        /// form1 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.HtmlControls.HtmlForm form1;

        /// <summary>

        /// Label1 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label1;

        /// <summary>

        /// Label2 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label2;

        /// <summary>

        /// FileUpload1 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.FileUpload FileUpload1;

        /// <summary>

        /// Label3 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label3;

        /// <summary>

        /// FileUpload2 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.FileUpload FileUpload2;

        /// <summary>

        /// Button1 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Button Button1;

        /// <summary>

        /// Label4 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label4;

        /// <summary>

        /// Label5 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label5;

        /// <summary>

        /// Table1 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Table Table1;

        /// <summary>

        /// Table2 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Table Table2;

        /// <summary>

        /// Label6 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label6;

        /// <summary>

        /// Table3 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Table Table3;

        /// <summary>

        /// Label9 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label9;

        /// <summary>

        /// Label8 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label8;

        /// <summary>

        /// TextBox5 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.TextBox TextBox5;

        /// <summary>

        /// Button2 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Button Button2;

        /// <summary>

        /// Label10 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label10;

        /// <summary>

        /// TextBox4 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.TextBox TextBox4;

        /// <summary>

        /// Label11 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Label Label11;

        /// <summary>

        /// TextBox6 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.TextBox TextBox6;

        /// <summary>

        /// Button3 control.

        /// </summary>

        /// <remarks>

        /// Auto-generated field.

        /// To modify move field declaration from designer file to code-behind file.

        /// </remarks>

        protected global::System.Web.UI.WebControls.Button Button3;

    }

}

**Style.css failas:**

body {

background-color:peachpuff;

}

table {

background-color: palegoldenrod;

border: 1px solid;

}

td {

padding: 2px;

border: 1px solid;

}

.MainLabelTop {

color: black;

font-family: 'Times New Roman';

font-size: 20px;

padding: 20px;

}

**UnitTest.cs failas:**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using LD3\_10\_MKuliesius;

using System;

using System.Configuration;

using System.Diagnostics;

namespace LD3\_10\_MKuliesius.UnitTest

{

    [TestClass]

    public class TaskUtilsTest

    {

        #region variables

        LinkedList<Worker> workers = InOut.ReadWorkerFile("U10a1.txt");

        LinkedList<Detail> details = InOut.ReadDetailFile("U10B.txt");

        #endregion

        [TestMethod]

        public void Validation\_CalculatesCorrectEarningForPrice()

        {

            var detail = "DetaleA";

            decimal expection = 1942.5m;

            decimal result = TaskUtils.CalculateEarningsForPart(workers,details,detail);

            Assert.AreEqual(expection, result);

        }

        [TestMethod]

        public void Validation\_CalculatesCorrectEarningForPriceWorker()

        {

            var name = "Smith John";

            var detail = "A123";

            decimal result = TaskUtils.CalculateEarningsForPart(workers, details, name, detail);

            Assert.AreEqual(1470.00m, result);

        }

        [TestMethod]

        public void Validation\_IsInList\_True()

        {

            var name = "Smith John";

            var detail = "A123";

            Worker worker = new Worker(name, "2024 - 04 - 01", "A123", 50);

            bool result = TaskUtils.IsInList(workers, worker);

            Assert.IsTrue(result);

        }

        [TestMethod]

        public void Validation\_IsInList\_False()

        {

            var name = "SAAA";

            var detail = "A123";

            Worker worker = new Worker(name, "2024 - 04 - 01", "A122", 20);

            bool result = TaskUtils.IsInList(workers, worker);

            Assert.IsFalse(result);

        }

        [TestMethod]

        public void Validation\_TotalPartsForWorker()

        {

            var name = "Smith John";

            var detail = "A123";

            Worker worker = new Worker(name, "2024 - 04 - 01", "A123", 50);

            var result = TaskUtils.TotalDetails(workers, worker);

            Assert.AreEqual(140,result);

        }

        [TestMethod]

        public void Validation\_OnlySymbolsAndSpaces\_ReturnsTrue()

        {

            //Arrange act assert

            var sub = "Naujoji Akmenė";

            bool result = TaskUtils.Validation(sub);

            Assert.IsTrue(result);

        }

        [TestMethod]

        public void Validation\_WithSpecialCharacters\_ReturnsFalse()

        {

            var sub = "Kaunas!";

            bool result = TaskUtils.Validation(sub);

            Assert.IsFalse(result);

        }

        [TestMethod]

        public void Validation\_Empty\_ReturnsFalse()

        {

            var sub = "";

            bool result = TaskUtils.Validation(sub);

            Assert.IsFalse(result);

        }

        [TestMethod]

        public void CalculateEarningsForPart\_WithValidData\_ReturnsCorrectEarnings()

        {

            // Expected earnings calculation: 50 \* 10.50 = 525.00

            decimal expected = 1942.50m;

            decimal result = TaskUtils.CalculateEarningsForPart(workers, details, "DetaleA");

            Assert.AreEqual(expected, result, "The calculated earnings do not match expected.");

        }

        [TestMethod]

        public void CalculateEarningsForPart\_WithValidWorkerAndPart\_ReturnsCorrectEarnings()

        {

            // Expected earnings calculation: 50 \* 10.50 = 525.00

            decimal expected = 1470.00m;

            decimal result = TaskUtils.CalculateEarningsForPart(workers, details, "Smith John", "A123");

            Assert.AreEqual(expected, result, "The calculated earnings for specific worker and part do not match expected.");

        }

        [TestMethod]

        public void IsInList\_ExistingWorker\_ReturnsTrue()

        {

            Worker testWorker = new Worker("Smith John", "2024-04-01", "A123", 50);

            bool isInList = TaskUtils.IsInList(workers, testWorker);

            Assert.IsTrue(isInList, "Existing worker should be found in list.");

        }

        [TestMethod]

        public void IsInList\_NonExistingWorker\_ReturnsFalse()

        {

            Worker testWorker = new Worker("Non Existent", "2024-04-01", "XYZ987", 10);

            bool isInList = TaskUtils.IsInList(workers, testWorker);

            Assert.IsFalse(isInList, "Non-existent worker should not be found in list.");

        }

        [TestMethod]

        public void TotalDetails\_ForExistingWorker\_ReturnsCorrectTotal()

        {

            int expected = 140; // Smith John produced 50 details

            Worker testWorker = new Worker("Smith John", "2024-04-01", "A123", 50);

            int totalDetails = TaskUtils.TotalDetails(workers, testWorker);

            Assert.AreEqual(expected, totalDetails, "Total details produced by the worker does not match expected.");

        }

        [TestClass]

        public class LinkedListTests

        {

            LinkedList<Worker> workers = InOut.ReadWorkerFile("U10a1.txt");

            LinkedList<Detail> details = InOut.ReadDetailFile("U10B.txt");

            [TestMethod]

            public void Sort\_BasicCorrectData\_SortsCorrectly()

            {

                LinkedList<Detail> TestList = new LinkedList<Detail>();

                bool correct = false;

                var det1 = new Detail("AAA", "ADATA", 2.5m);

                var det2 = new Detail("BBB", "BamBaliys", 2.6m);

                var det3 = new Detail("CCC", "CAdiLavaas", 3.7m);

                TestList.Add(det1);

                TestList.Add(det2);

                TestList.Add(det3);

                TestList.Sort();

                if (TestList.Get(0).Name == "AAA" && TestList.Get(1).Name == "BBB" && TestList.Get(2).Name == "CCC")

                {

                    correct = true;

                }

                string message = "AAA " + TestList.Get(0).Code + " BBB " + TestList.Get(1).Code + " CCC " + TestList.Get(2).Code;

                Debug.WriteLine(message);

                Assert.IsTrue(correct, message);

            }

            [TestMethod]

            public void Add\_BasicCorrectData\_AddsToTheEndCorrectly()

            {

                LinkedList<Detail> TestList = new LinkedList<Detail>();

                bool correct = false;

                var det1 = new Detail("AAA", "ADATA", 2.5m);

                var det2 = new Detail("BBB", "BamBaliys", 2.6m);

                var det3 = new Detail("CCC", "CAdiLavaas", 3.7m);

                TestList.Add(det2);

                TestList.Add(det1);

                TestList.Add(det3);

                if (TestList.Get(2).Code == "CCC")

                {

                    correct = true;

                }

                Assert.IsTrue(TestList.Get(2).Code.Equals("CCC"));

            }

            [TestMethod]

            public void Sort\_BasicCorrectData\_SortsInt()

            {

                LinkedList<int> AllSubs = new LinkedList<int>();

                bool sorted = false;

                var sub1 = 10; //2nd

                var sub2 = 1; //1st

                var sub3 = 25; //3rd

                AllSubs.Add(sub1);

                AllSubs.Add(sub2);

                AllSubs.Add(sub3);

                AllSubs.Sort();

                if (AllSubs.Get(0) <= AllSubs.Get(1) && AllSubs.Get(1) <= AllSubs.Get(2))

                {

                    sorted = true;

                }

                Assert.IsTrue(sorted);

            }

        }

    }

}

## Pradiniai duomenys ir rezultatai

Testavimam sukūriau 3 darbuotojų duomenų failus bei 1 detalių informacijos failą.

Pirmasis darbuotojų failas:

U10a1.txt:

2024-04-01;Smith John;A123;50

2024-04-01;Smith John;B456;30

2024-04-01;Johnson Alice;A123;40

2024-04-02;Smith John;A123;60

2024-04-02;Johnson Alice;C789;20

2024-04-03;Johnson Alice;B456;25

2024-04-03;Johnson Alice;A123;35

Antrasis darbuotojų failas:

U10a2.txt

2024-04-01;Brown Bob;A123;40

2024-04-01;Brown Bob;B456;30

2024-04-01;Brown Bob;C789;20

2024-04-02;Brown Bob;A123;50

2024-04-02;Brown Bob;C789;15

2024-04-02;White Wendy;B456;35

2024-04-03;White Wendy;A123;25

2024-04-03;White Wendy;C789;30

Trečiasis darbuotojų failas:

U10a3.txt

2024-04-01;Green Gary;A123;60

2024-04-01;Green Gary;B456;20

2024-04-01;Green Gary;C789;40

2024-04-02;Green Gary;B456;25

2024-04-02;Green Gary;A123;45

2024-04-02;Green Gary;C789;30

2024-04-03;Green Gary;A123;35

2024-04-03;Green Gary;B456;30

Detalių failas:

U10b.txt

A123;DetaleA;10.50

B456;DetaleB;15.75

C789;DetaleC;20.00

**Testavimas:**

* Pirmasis testas

Pirmam testui turime du asmenis: Smith John ir Johnson Alice. John dirbo 3 dienas, o Alice dirbo 4 dienas. Kaip matoma duomenų failuose viršuje, naudojame pirmą darbuotojų testavimo failą.

Paleidus programą ir atidarius failus gauname tokį vaizdą:

A screenshot of a computer

Description automatically generated

Kaip matome, apskaičiavo kuris darbuotojas geriausiai pasirodė darbe, atvaizdavo pradinius duomenis, bei atrinko unikalius darbuotojus.

Toliau pasirenkame pagal kokią detalę norime filtruoti ir šio testo atveju pasirenkame DetaleA:

A screenshot of a computer

Description automatically generated

Kaip matome, atrinko visus iš sąrašo darbuotojus, kurie gamino šią dalį, parodo datą kada dirbo, surikiavo iš eilės darbuotojus. Suskaičiavo kiek iš viso detalių pagamino bei kiek uždirbo iš gamybos

Žemiau esančiuose tekstiniuose laukuose pasirenkame norimus kriterijus kaip atsirinkti darbuotojus:

A close-up of a computer screen

Description automatically generated

Atrenkame, kad pagamintų daugiau nei 20 detalių ir kainuotų mažiau nei 500.

Atrinkus, išveda rezultatus į failą U10a.txt:

Atrinktu pagal kriterijus darbuotoju sarasas

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Johnson Alice | A123 | 40 | 3 | 120 | 1260.00 |

| 2024-04-03 | Johnson Alice | B456 | 25 | 0 | 25 | 262.50 |

| 2024-04-03 | Johnson Alice | A123 | 35 | 0 | 35 | 367.50 |

| 2024-04-01 | Smith John | B456 | 30 | 0 | 30 | 315.00 |

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

Rezultatų failas:

Pradiniai darbuotoju duomenys

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Smith John | A123 | 50 | 0 | 0 | 0 |

| 2024-04-01 | Smith John | B456 | 30 | 0 | 0 | 0 |

| 2024-04-01 | Johnson Alice | A123 | 40 | 0 | 0 | 0 |

| 2024-04-02 | Smith John | A123 | 60 | 0 | 0 | 0 |

| 2024-04-02 | Johnson Alice | C789 | 20 | 0 | 0 | 0 |

| 2024-04-03 | Johnson Alice | B456 | 25 | 0 | 0 | 0 |

| 2024-04-03 | Johnson Alice | A123 | 35 | 0 | 0 | 0 |

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

Pradiniai detaliu duomenys

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

| Pavadinimas | Det. Kodas | Kaina |

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

DetaleA | A123 | 10.50

DetaleB | B456 | 15.75

DetaleC | C789 | 20.00

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

* Antrasis testas

Antram testui turime du asmenis: Brown Bob ir White Wendy. Bob dirbo 2 dienas, o Wendy dirbo 2 dienas. Kaip matoma duomenų failuose viršuje, naudojame antrą darbuotojų testavimo failą.

Paleidus programą ir atidarius failus gauname tokį vaizdą:

A screenshot of a computer

Description automatically generated

Kaip matome, apskaičiavo kuris darbuotojas geriausiai pasirodė darbe, atvaizdavo pradinius duomenis, bei atrinko unikalius darbuotojus.

Toliau pasirenkame pagal kokią detalę norime filtruoti ir šio testo atveju pasirenkame DetaleA:

A screenshot of a computer

Description automatically generated

Kaip matome, atrinko visus iš sąrašo darbuotojus, kurie gamino šią dalį, parodo datą kada dirbo, surikiavo iš eilės darbuotojus. Suskaičiavo kiek iš viso detalių pagamino bei kiek uždirbo iš gamybos

Žemiau esančiuose tekstiniuose laukuose pasirenkame norimus kriterijus kaip atsirinkti darbuotojus:

A close-up of a computer screen

Description automatically generated

Atrenkame, kad pagamintų daugiau nei 20 detalių ir kainuotų mažiau nei 300.

Atrinkus, išveda rezultatus į failą U10a.txt:

Atrinktu pagal kriterijus darbuotoju sarasas

Atrinktu pagal kriterijus darbuotoju sarasas

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-03 | White Wendy | A123 | 25 | 0 | 25 | 393.75 |

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

Rezultatų failas:

Pradiniai darbuotoju duomenys

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Brown Bob | A123 | 40 | 0 | 0 | 0 |

| 2024-04-01 | Brown Bob | B456 | 30 | 0 | 0 | 0 |

| 2024-04-01 | Brown Bob | C789 | 20 | 0 | 0 | 0 |

| 2024-04-02 | Brown Bob | A123 | 50 | 0 | 0 | 0 |

| 2024-04-02 | Brown Bob | C789 | 15 | 0 | 0 | 0 |

| 2024-04-02 | White Wendy | B456 | 35 | 0 | 0 | 0 |

| 2024-04-03 | White Wendy | A123 | 25 | 0 | 0 | 0 |

| 2024-04-03 | White Wendy | C789 | 30 | 0 | 0 | 0 |

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

Pradiniai detaliu duomenys

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

| Pavadinimas | Det. Kodas | Kaina |

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

DetaleA | A123 | 10.50

DetaleB | B456 | 15.75

DetaleC | C789 | 20.00

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

* Trečiasis testas

Trečiam testui turime vieną asmenį: Green Gary, kuris dirbo 3 dienas. Kaip matoma duomenų failuose viršuje, naudojame antrą darbuotojų testavimo failą.

Paleidus programą ir atidarius failus gauname tokį vaizdą:

A screenshot of a computer

Description automatically generated

Kaip matome, apskaičiavo kuris darbuotojas geriausiai pasirodė darbe, atvaizdavo pradinius duomenis, bei atrinko unikalius darbuotojus.

Toliau pasirenkame pagal kokią detalę norime filtruoti ir šio testo atveju pasirenkame DetaleA:

A screenshot of a computer

Description automatically generated

Kaip matome, atrinko visus iš sąrašo darbuotojus, kurie gamino šią dalį, parodo datą kada dirbo, surikiavo iš eilės darbuotojus. Suskaičiavo kiek iš viso detalių pagamino bei kiek uždirbo iš gamybos

Žemiau esančiuose tekstiniuose laukuose pasirenkame norimus kriterijus kaip atsirinkti darbuotojus:

A close-up of a computer screen

Description automatically generated

Atrenkame, kad pagamintų daugiau nei 30 detalių ir kainuotų mažiau nei 400.

Atrinkus, išveda rezultatus į failą U10a.txt:

Atrinktu pagal kriterijus darbuotoju sarasas

Atrinktu pagal kriterijus darbuotoju sarasas

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-03 | Green Gary | A123 | 35 | 0 | 35 | 367.50 |

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

Rezultatų failas:

Pradiniai darbuotoju duomenys

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

| Data | Vardas | Det. Kodas | Pagamino | Viso dirbo | Viso pagamino | Viso uzdirbo |

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

| 2024-04-01 | Green Gary | A123 | 60 | 0 | 0 | 0 |

| 2024-04-01 | Green Gary | B456 | 20 | 0 | 0 | 0 |

| 2024-04-01 | Green Gary | C789 | 40 | 0 | 0 | 0 |

| 2024-04-02 | Green Gary | B456 | 25 | 0 | 0 | 0 |

| 2024-04-02 | Green Gary | A123 | 45 | 0 | 0 | 0 |

| 2024-04-02 | Green Gary | C789 | 30 | 0 | 0 | 0 |

| 2024-04-03 | Green Gary | A123 | 35 | 0 | 0 | 0 |

| 2024-04-03 | Green Gary | B456 | 30 | 0 | 0 | 0 |

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

Pradiniai detaliu duomenys

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

| Pavadinimas | Det. Kodas | Kaina |

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

DetaleA | A123 | 10.50

DetaleB | B456 | 15.75

DetaleC | C789 | 20.00

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

UnitTest testavimas:

Apsirašiau kelis testus TaskUtils klasės funkcionalumui patestuoti. Kadangi ne visus metodus išeina tokiu būdu ištestuoti, apsirašiau kelis pačius reikalingiausius metodus, kuriuos testavau:  
A screenshot of a computer

Description automatically generated

Kaip matome iš rezultatų, visi testai praeina gerai išskyrus Sort\_BasicCorrectData\_SortsCorrectly:

A screenshot of a computer

Description automatically generated

Šio testo rezultatai parodo, kad metodas rikiuoti duomenis rikiuoja, tačiau tikrinimo metodai neveikia kaip turėtų arba tiesiog kažką ne taip aprašiau kode

## Dėstytojo pastabos

LD3 ataskaita: P4 (-0.2), P6 (-0.2), P7 (-0.2), P11 (-0.2), P13 (-0.2). Įv.: 0

Už programą: 6

Testukas: 0/3

# Polimorfizmas ir išimčių valdymas (L4)

## Darbo užduotis

**U4\_10. Buitinės technikos parduotuvė**. Turite informaciją apie skirtingose buitinės technikos parduotuvėse (>=3) esančius šaldytuvus, mikrobangų krosneles ir elektrinius virdulius. Pirmoje eilutėje yra pavadinimas, antroje – adresas, trečioje – telefonas. Sukurkite abstrakčią klasę „Device“ (savybės - gamintojas, modelis, energijos klasė, spalva, kaina), kurią paveldės “Fridge” (savybės - talpa, montavimo tipas, požymis „turi šaldiklį“, aukštis, plotis, gylis), “Oven” (savybės – galingumas, programų skaičius) ir “Kettle” (savybės – galia, tūris).

* Suskaičiuokite, kiek skirtingų „Siemens“ šaldytuvų, mikrobangų krosnelių ir virdulių modelių siūlo kiekviena parduotuvė, rezultatą atspausdinkite ekrane.
* Sudarykite dešimties pigiausių pastatomų šaldytuvų, kurių talpa 80 litrų ar didesnė, sąrašą. Ekrane atspausdinkite šaldytuvo gamintoją, modelį, talpą ir kainą.
* Ar yra tokių buitinių prietaisų, kuriuos galima įsigyti tik vienoje parduotuvėje? Atspausdinkite tokių prietaisų sąrašą faile „TikTen.csv“.
* Sudarykite ir surikiuokite brangių buitinių prietaisų sąrašą, pateikdami pilną informaciją apie juos. Šaldytuvas yra brangus, jei jo kaina viršija 1000€. Mikrobangų krosnelė yra brangi, jei jos kaina viršija 500€. Virdulys yra brangus, jei jo kaina viršija 50€. Šaldytuvus rikiuokite pagal aukštį, mikrobangų krosneles – pagal galingumą, o virdulius – pagal galią. Rezultatus įrašykite į failą „Brangus.csv“.

## Grafinės vartotojo sąsajos schema

A screenshot of a computer

Description automatically generated

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Label | Text | LD4\_10 Martynas Kuliešius |
| FileUpload | Height\*Width, AllowMultiple | 69\*420, True |
| Button1 | Text, Height\*Width, Font-Size | Atlikti užduotį, 50\*160,X-Large |

## Klasių diagrama

A diagram of a program

Description automatically generated

## Programos vartotojo vadovas

Paleidus programą naudotoją pasitinka toks vaizdas:

A green box with black text

Description automatically generated

Pasirenkami norimi naudoti failai, kuriuos nori skaityti vartotojas:

A green rectangular sign with black border

Description automatically generated

Paspaudžiamas mygtukas atlikti užduotį. Programa atlieka savo darbus ir tai, ką reikia, išveda į lenteles, o kitais atvejais pasako, kokiame faile išvedė užduoties rezultatus:

A screenshot of a computer

Description automatically generated

## Programos tekstas

**Device.cs failas:**

using System;

namespace LD4\_10\_MKuliesius.AppCode

{

    public abstract class Device: IComparable<Device>, IEquatable<Device>

    {

        public string Maker {  get; set; }

        public string Model { get; set; }

        public string EnergyClass { get; set; }

        public string Colour { get; set; }

        public decimal Price { get; set; }

        public string Type { get; set; }

        public Device() { }

        public Device(string maker, string model, string energyClass, string colour, decimal price, string type)

        {

            Maker = maker;

            Model = model;

            EnergyClass = energyClass;

            Colour = colour;

            Price = price;

            Type = type;

        }

        public abstract int CompareTo(Device other);

        public abstract bool Equals(Device other);

        public override int GetHashCode()

        {

            return base.GetHashCode();

        }

    }

}

**Fridge.cs failas:**

using System;

namespace LD4\_10\_MKuliesius.AppCode

{

    public class Fridge : Device, IComparable<Device>, IEquatable<Fridge>

    {

        //Saldytuvas brangus jei kainuoja virs 1000 . Šaldytuvus rikiuokite pagal aukštį

        public int Capacity { get; set; }

        public string MountType { get; set; }

        public bool HasFreezer { get; set; }

        public double Height { get; set; }

        public double Width { get; set; }

        public double Depth { get; set; }

        public Fridge() :base() { }

        public Fridge(string maker, string model, string energyClass, string colour, decimal price, string type, int capacity, string mountType, bool hasFreezer, double height, double width, double depth) : base(maker, model,energyClass,colour,price, type)

        {

            Capacity = capacity;

            MountType = mountType;

            HasFreezer = hasFreezer;

            Height = height;

            Width = width;

            Depth = depth;

        }

        public override int CompareTo(Device other)

        {

            if(other is Fridge fridge)

            {

                return Height.CompareTo(fridge.Height);

            }

            return 0;

        }

       public override bool Equals(Device other)

       {

           return Maker.Equals(other.Maker);

       }

        public bool Equals(Fridge other)

        {

            return MountType.Equals(other.MountType);

        }

        public override string ToString()

        {

            return Maker + " " + Model + " " + EnergyClass + " " + Colour + " " + Price + "e " + Capacity + "L " + MountType + " " + HasFreezer + " " + Height + "m " + Width + "m " + Depth + "m ";

        }

    }

}

**Kettle.cs failas:**

using System;

namespace LD4\_10\_MKuliesius.AppCode

{

    public class Kettle: Device, IComparable<Device>, IEquatable<Kettle>

    {

        // virdulys brangus jei kainuoja virs 50, o virdulius rikiuokite pagal galią

        public int Power { get; set; }

        public double Capacity {  get; set; }

        public Kettle() : base() { }

        public Kettle(string maker, string model, string energyClass, string colour, decimal price, string type, int power, double capacity) : base(maker, model, energyClass, colour,price, type)

        {

            Power = power;

            Capacity = capacity;

        }

        public override int CompareTo(Device other)

        {

            if (other is Kettle kettle)

            {

                return Power.CompareTo(kettle.Power);

            }

            return 0;

        }

        public override bool Equals(Device other)

        {

            return Price.Equals(other.Price);

        }

        public bool Equals(Kettle other)

        {

            return Capacity.Equals(other.Capacity);

        }

        public override string ToString()

        {

            return Maker + " " + Model + " " + EnergyClass + " " + Colour + " " + Price + "e "+ Power +"W " + Capacity +"L ";

        }

    }

}

**Oven.cs failas:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace LD4\_10\_MKuliesius.AppCode

{

    public class Oven: Device, IComparable<Device>, IEquatable<Oven>

    {

        // Orkaite brangi jei kainuoja virs 500, mikrobangų krosneles rikiuokite pagal galingumą

        public int Power { get; set; }

        public int ProgramNum { get; set; }

        public Oven() : base() { }

        public Oven(string maker, string model, string energyClass, string colour, decimal price, string type, int power, int programNum) : base(maker, model, energyClass, colour, price, type)

        {

            Power = power;

            ProgramNum = programNum;

        }

        public override int CompareTo(Device other)

        {

            if (other is Oven oven)

            {

                return Power.CompareTo(oven.Power);

            }

            return 0;

        }

        public override bool Equals(Device other)

        {

            return Maker.Equals(other.Maker);

        }

        public bool Equals(Oven other)

        {

            return ProgramNum.Equals(other.ProgramNum);

        }

        public override string ToString()

        {

            return Maker + " " + Model + " " + EnergyClass + " " + Colour + " " + Price + "e " + Power + "W " + ProgramNum;

        }

    }

}

**DevicesContainer.cs failas:**

using System;

using System.Collections.Generic;

namespace LD4\_10\_MKuliesius.AppCode

{

    /// <summary>

    /// container/register of a shop and all of its items

    /// </summary>

    public class DevicesContainer

    {

        public string Name { get; set; }

        public string Address { get; set; }

        public string PhoneNum { get; set; }

        private List<Device> AllDevices;

        public DevicesContainer()

        {

            AllDevices = new List<Device>();

        }

        public DevicesContainer(string name, string address, string phoneNum, List<Device> devices)

        {

            Name = name;

            Address = address;

            PhoneNum = phoneNum;

            AllDevices = new List<Device>(devices);

        }

        public string GetName()

        {

            return Name;

        }

        public void Add(Device device)

        {

            AllDevices.Add(device);

        }

        public int ItemsCount()

        {

            return this.AllDevices.Count;

        }

        public Device GetByIndex(int index)

        {

            return this.AllDevices[index];

        }

        public List<Device> GetAll()

        {

            return AllDevices;

        }

        public bool IsEmpty()

        {

            if (AllDevices.Count > 0)

            {

                return false;

            }

            else return true;

        }

        public int Count()

        {

            return AllDevices.Count;

        }

        public Device Get(int index)

        {

            if (index >= 0 && index < AllDevices.Count)

            {

                return AllDevices[index];

            }

            throw new IndexOutOfRangeException("Index is out of range for device list.");

        }

    }

}

**FormExtention.cs failas:**

using LD4\_10\_MKuliesius.AppCode;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web;

using System.Web.UI.WebControls;

namespace LD4\_10\_MKuliesius

{

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

    {

        /// <summary>

        /// Checks if correct files were uploaded

        /// </summary>

        /// <param name="fileUpload"></param>

        /// <returns></returns>

        public bool ProperFiles(FileUpload fileUpload)

        {

            foreach(HttpPostedFile file in fileUpload.PostedFiles)

            {

                if(!(file.FileName.EndsWith(".txt") || file.FileName.EndsWith(".csv")))

                {

                    return false;

                }

            }

            return fileUpload.HasFiles;

        }

        /// <summary>

        /// Copies files to the folder

        /// </summary>

        /// <param name="fileUpload"></param>

        /// <param name="path"></param>

        public void CopyFilesToServer(FileUpload fileUpload, string path)

        {

            IList<HttpPostedFile> files = fileUpload.PostedFiles;

            Directory.Delete(path, true);

            Directory.CreateDirectory(path);

            for (int i = 0; i < fileUpload.PostedFiles.Count; i++)

            {

                files[i].SaveAs(path + files[i].FileName);

            }

        }

        /// <summary>

        /// Isveda siemens gaminiu kieki i lentele ekrane

        /// </summary>

        /// <param name="table"></param>

        /// <param name="siemensCount"></param>

        public void AddSiemensCountToTable(Table table, int siemensCount)

        {

            TableCell text= new TableCell()

            {

                Text="Is viso Siemens gaminiu yra: "

            };

            TableCell text2 = new TableCell()

            {

                Text = siemensCount.ToString()

            };

            TableRow tableRow = new TableRow()

            {

                Cells =

                {

                    text,

                    text2

                }

            };

            table.Rows.Add(tableRow);

        }

        /// <summary>

        /// Isveda pigiausius saldytuvus i lentele

        /// </summary>

        /// <param name="table"></param>

        /// <param name="fridges"></param>

        public void CreateCheapFridgeTable(Table table, List<Fridge> fridges)

        {

            ///gamintojas modelis talpa kaina

            ///

            TableCell Title1 = new TableCell()

            {

                Text = "Gamintojas"

            };

            TableCell Title2 = new TableCell()

            {

                Text = "Modelis"

            };

            TableCell Title3 = new TableCell()

            {

                Text = "Talpa"

            };

            TableCell Title4 = new TableCell()

            {

                Text = "Kaina"

            };

            TableRow Row = new TableRow()

            {

                Cells =

                {

                    Title1,

                    Title2,

                    Title3,

                    Title4

                }

            };

            table.Rows.Add(Row);

            foreach (Fridge fridge in fridges)

            {

                TableCell Maker = new TableCell()

                {

                    Text = fridge.Maker

                };

                TableCell Model = new TableCell()

                {

                    Text = fridge.Model

                };

                TableCell Capacity = new TableCell()

                {

                    Text = fridge.Capacity.ToString()

                };

                TableCell Price = new TableCell()

                {

                    Text = fridge.Price.ToString()

                };

                TableRow Row2 = new TableRow()

                {

                    Cells =

                    {

                        Maker,

                        Model,

                        Capacity,

                        Price

                    }

                };

                table.Rows.Add(Row2);

            }

        }

    }

}

**InOut.cs failas:**

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace LD4\_10\_MKuliesius.AppCode

{

    public class InOut

    {

        public static List<DevicesContainer> ReadDevices(string path)

        {

            List<DevicesContainer> allShops =  new List<DevicesContainer>();

            foreach (string file in Directory.GetFiles(path))

            {

                List<Device> devices = new List<Device>();

                string[] Lines = File.ReadAllLines(file); // Correct file reading

                if (Lines.Length >= 3)

                {

                    string shopName = Lines[0];

                    string address = Lines[1];

                    string phone = Lines[2];

                    for (int i = 3; i < Lines.Length; i++) // Adjusted to iterate correctly

                    {

                        string[] Bits = Lines[i].Split(new[] { ";" }, StringSplitOptions.RemoveEmptyEntries);

                        if (Bits.Length < 6)

                        {

                            throw new Exception($"Data incorrectly formatted at line {i + 1}");

                        }

                        string type = Bits[0];

                        string maker = Bits[1];

                        string model = Bits[2];

                        string energyClass = Bits[3];

                        string color = Bits[4];

                        decimal price = Convert.ToDecimal(Bits[5]);

                        switch (type)

                        {

                            case "Fridge":

                                int capacity = Convert.ToInt32(Bits[6]);

                                string mountType = Bits[7];

                                string booly = Bits[8];

                                bool hasFreezer = false;

                                if (booly == "Yes")

                                {

                                    hasFreezer = true;

                                }

                                double height = Convert.ToDouble(Bits[9]);

                                double width = Convert.ToDouble(Bits[10]);

                                double depth = Convert.ToDouble(Bits[10]);

                                Fridge fridge = new Fridge(maker, model, energyClass, color, price, type, capacity, mountType, hasFreezer, height, width, depth);

                                devices.Add(fridge);

                                break;

                            case "Kettle":

                                int power = Convert.ToInt32(Bits[6]);

                                double capacity1 = Convert.ToDouble(Bits[7]);

                                Kettle kettle = new Kettle(maker, model, energyClass, color,price,type,power,capacity1);

                                devices.Add(kettle);

                                break;

                            case "Oven":

                                int power1 = Convert.ToInt32(Bits[6]);

                                int programNum = Convert.ToInt32(Bits[7]);

                                Oven oven = new Oven(maker, model, energyClass, color, price, type, power1, programNum);

                                devices.Add(oven);

                                break;

                            default: throw new Exception($"Ivestas netinkamo tipo narys {type}");

                        }

                    }

                    DevicesContainer container = new DevicesContainer(shopName, address, phone, devices);

                    allShops.Add(container);

                }

                else throw new Exception($"Per mažai duomenų");

            }

            return allShops;

        }

        /// <summary>

        /// Isveda  visus duomenis i startiniu duomenu faila

        /// </summary>

        /// <param name="shops"></param>

        /// <param name="fin"></param>

        /// <param name="header"></param>

        /// <param name="Format"></param>

        public static void PrintStarting(List<DevicesContainer> shops, string fin, string header, string Format)

        {

            string dashes = new string('-', 115);

            using (var file = new StreamWriter(fin, false, Encoding.UTF8))

            {

                foreach (DevicesContainer shop in shops)

                {

                    file.WriteLine(header);

                    file.WriteLine(dashes);

                    if (!shop.IsEmpty())

                    {

                        file.WriteLine(shop.Name);

                        file.WriteLine(shop.Address);

                        file.WriteLine(shop.PhoneNum);

                        file.WriteLine(dashes);

                        file.WriteLine(Format);

                        file.WriteLine(dashes);

                        for (int i = 0; i < shop.Count(); i++)

                        {

                            file.WriteLine(shop.Get(i).ToString());

                        }

                    }

                    else file.WriteLine("Sąrašas tusčias");

                    file.WriteLine(dashes);

                    file.WriteLine();

                    file.WriteLine();

                }

            }

        }

        /// <summary>

        /// Spausdinimas i csv faila tik tom prekem, kurias galima rasti tik vienoje parduotuveje.

        /// </summary>

        /// <param name="shops"></param>

        /// <param name="fin"></param>

        public static void PrintToCsvOnlyOneShop(List<DevicesContainer> shops, string fin)

        {

            using (StreamWriter sw = new StreamWriter(fin))

            {

                foreach (DevicesContainer shop in shops)

                {

                    sw.WriteLine("{0};",shop.Name);

                    sw.WriteLine("{0};",shop.Address);

                    for (int i = 0;i < shop.Count();i++)

                    {

                        Device device = shop.Get(i);

                        if (device is Fridge)

                        {

                            Fridge fr = (Fridge)device;

                            sw.WriteLine("Saldytuvas");

                            sw.WriteLine("{0};{1}", fr.Maker, fr.Model);

                        }

                        if (device is Kettle)

                        {

                            Kettle ke = (Kettle)device;

                            sw.WriteLine("Virdulys");

                            sw.WriteLine("{0};{1}", ke.Maker, ke.Model);

                        }

                        if (device is Oven)

                        {

                            Oven ov = (Oven)device;

                            sw.WriteLine("Mikrobange");

                            sw.WriteLine("{0};{1}", ov.Maker, ov.Model);

                        }

                    }

                }

            }

        }

        /// <summary>

        /// Spausdinimas i csv faila visiems brangiems produktams

        /// </summary>

        /// <param name="devices"></param>

        /// <param name="fin"></param>

        public static void PrintToCsvExpensive(List<Device> devices, string fin)

        {

            using (StreamWriter sw = new StreamWriter(fin))

            {

                foreach (Device device in devices)

                {

                    if (device is Fridge fr)

                    {

                        sw.WriteLine("Saldytuvas");

                        sw.WriteLine("{0};{1};{2};{3};{4};{5};{6};{7};{8};{9};{10};", fr.Maker,

                            fr.Model, fr.EnergyClass, fr.Colour, fr.Price.ToString(), fr.Capacity.ToString(), fr.MountType, fr.HasFreezer.ToString(), fr.Height.ToString(), fr.Width.ToString(), fr.Depth.ToString());

                    }

                    if (device is Kettle ke)

                    {

                        sw.WriteLine("Virdulys");

                        sw.WriteLine("{0};{1};{2};{3};{4};{5};{6}", ke.Maker, ke.Model, ke.EnergyClass, ke.Colour, ke.Price.ToString(), ke.Power.ToString(), ke.Capacity.ToString());

                    }

                    if (device is Oven ov)

                    {

                        sw.WriteLine("Mikrobange");

                        sw.WriteLine("{0};{1};{2};{3};{4};{5};{6}", ov.Maker, ov.Model, ov.EnergyClass, ov.Colour, ov.Price.ToString(), ov.Power.ToString(), ov.ProgramNum.ToString());

                    }

                }

            }

        }

        /// <summary>

        /// Prints siemens count to result file

        /// </summary>

        /// <param name="fin"></param>

        /// <param name="count"></param>

        /// <param name="header"></param>

        public static void PrintToTxt(string fin, int count, string header)

        {

            string dashes = new string('-', 115);

            File.AppendAllText(fin, dashes);

            File.AppendAllText(fin, "\n");

            File.AppendAllText(fin, header + count.ToString());

            File.AppendAllText(fin, "\n");

            File.AppendAllText(fin, dashes);

            File.AppendAllText(fin, "\n");

        }

        /// <summary>

        /// Prints siemens count to result file

        /// </summary>

        /// <param name="fin"></param>

        /// <param name="count"></param>

        /// <param name="header"></param>

        public static void PrintToTxt(string fin, List<Fridge> fridges, string header)

        {

            string dashes = new string('-', 115);

            File.AppendAllText(fin, header);

            File.AppendAllText(fin, "\n");

            File.AppendAllText(fin, dashes);

            File.AppendAllText(fin, "\n");

            foreach (Fridge fridge in fridges)

            {

                File.AppendAllText(fin, fridge.Maker + " " + fridge.Model + " " + fridge.Price.ToString() + " " + fridge.Capacity.ToString());

                File.AppendAllText(fin, "\n");

            }

            File.AppendAllText(fin, dashes);

            File.AppendAllText(fin, "\n");

        }

    }

}

**TaskUtils.cs failas:**

using System.Collections.Generic;

using System.Linq;

namespace LD4\_10\_MKuliesius.AppCode

{

    public class TaskUtils

    {

        /// <summary>

        /// Task 1, calculate how many siemens devices there are

        /// </summary>

        /// <param name="shops"></param>

        /// <returns></returns>

        public static int SiemensCount(List<DevicesContainer> shops)

        {

            int count = 0;

            foreach (DevicesContainer shop in shops)

            {

                for (int i = 0; i < shop.Count(); i++)

                {

                    Device device = shop.Get(i);

                    if (device.Maker.Equals("Siemens"))

                    {

                        count++;

                    }

                }

            }

            return count;

        }

        /// <summary>

        /// Task 2, returns the cheapest standing fridges that are freestadnign and have more than 80l capacity

        /// </summary>

        /// <param name="shops"></param>

        /// <returns></returns>

        public static List<Fridge> StandingCheapestFridges(List<DevicesContainer> shops)

        {

            return shops

                .SelectMany(shop => shop.GetAll())

                .OfType<Fridge>() // Ensure the device is a Fridge before checking further

                .Where(fridge => fridge.MountType == "Freestanding" && fridge.Capacity > 80) // Filters for Freestanding mount type and capacity > 80

                .OrderBy(fridge => fridge.Price)

                .Take(10)

                .ToList();

        }

        /// <summary>

        /// Finds devices that are only sold in a single store

        /// </summary>

        /// <param name="shops"></param>

        /// <returns></returns>

        public static List<DevicesContainer> OnlyOneShop(List<DevicesContainer> shops)

        {

            // Dictionary to count devices

            var deviceCounts = new Dictionary<(string Manufacturer, string Model), int>();

            // Fill the dictionary with device counts across all shops

            foreach (var shop in shops)

            {

                foreach (var device in shop.GetAll())

                {

                    var key = (device.Maker, device.Model);

                    if (deviceCounts.ContainsKey(key))

                    {

                        deviceCounts[key]++;

                    }

                    else

                    {

                        deviceCounts[key] = 1;

                    }

                }

            }

            // Now, create a list of containers to store only unique devices

            List<DevicesContainer> uniqueShops = shops.Select(\_ => new DevicesContainer()).ToList();

            // Iterate through shops and add only unique items

            for (int i = 0; i < shops.Count; i++)

            {

                foreach (var device in shops[i].GetAll())

                {

                    var key = (device.Maker, device.Model);

                    if (deviceCounts[key] == 1)  // Only add device if it appears exactly once across all shops

                    {

                        uniqueShops[i].Add(device);

                    }

                }

            }

            return uniqueShops;  // This list will include empty containers if no unique items found in corresponding shop

        }

        /// <summary>

        /// Finds Most expensive items that are for sale in each store

        /// </summary>

        /// <param name="shops"></param>

        /// <returns></returns>

        public static List<Device> ExpensiveDevices(List<DevicesContainer> shops)

        {

            List<Device> expensiveDevices = new List<Device>();

            // Aggregate all devices from all shops into a single list for easier processing

            List<Device> allDevices = shops.SelectMany(shop => shop.GetAll()).ToList();

            // Filtering and sorting Fridges

            var expensiveFridges = allDevices.OfType<Fridge>()

                                            .Where(f => f.Price > 1000)

                                            .OrderBy(f => f) // Using IComparable<Fridge>

                                            .ToList();

            // Filtering and sorting Ovens

            var expensiveOvens = allDevices.OfType<Oven>()

                                           .Where(o => o.Price > 500)

                                           .OrderBy(o => o) // Using IComparable<Oven>

                                           .ToList();

            // Filtering and sorting Kettles

            var expensiveKettles = allDevices.OfType<Kettle>()

                                             .Where(k => k.Price > 50)

                                             .OrderBy(k => k) // Using IComparable<Kettle>

                                             .ToList();

            // Add sorted lists to the final list

            expensiveDevices.AddRange(expensiveFridges);

            expensiveDevices.AddRange(expensiveOvens);

            expensiveDevices.AddRange(expensiveKettles);

            return expensiveDevices;

        }

    }

}

**Forma.aspx failas:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma.aspx.cs" Inherits="LD4\_10\_MKuliesius.Forma" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

        <link href="Style.css" rel="stylesheet" type="text/css" />

    <style type="text/css">

        #form1 {

            margin-left: 40px;

        }

    </style>

</head>

<body>

    <form id="form1" runat="server">

        <div>

        <asp:Label runat="server" Text="LD4\_10 Martynas Kuliešius"></asp:Label></div> <br/>

        <asp:FileUpload runat="server" ID="FileUpload1" AllowMultiple="True" BackColor="#CCFF99" BorderColor="Black" BorderStyle="Groove" Height="69px" Width="420px"></asp:FileUpload> <br/>

        <asp:Button runat="server" Text="Atlikti užduotį" OnClick="Button1\_Click" ID="Button1" Font-Size="X-Large" Height="50px" Width="160px"></asp:Button> <br/><br/>

        <asp:Label runat="server" ID="Label1" Text="Label" Visible="false"></asp:Label> <br/>

        <asp:Table runat="server" ID="Table1" Visible="false"></asp:Table> <br/>

        <asp:Label runat="server" ID="Label2" Text="Label" Visible="false"></asp:Label> <br/>

        <asp:Table runat="server" ID="Table2" Visible="false"></asp:Table> <br/>

        <asp:Label runat="server" ID="Label3" Text="Label" Visible="false"></asp:Label> <br/>

        <asp:Label runat="server" ID="Label4" Text="Label" Visible="false"></asp:Label> <br/>

        <asp:Label runat="server" ID="Label5" Text="Label" Visible="false"></asp:Label> <br/>

    </form>

</body>

</html>

**Forma.aspx.cs failas:**

using LD4\_10\_MKuliesius.AppCode;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

namespace LD4\_10\_MKuliesius

{

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

    {

        string DataFolder;

        string ResultFolder;

        List<DevicesContainer> AllDataContainers = new List<DevicesContainer>();

        List<DevicesContainer> AllContainers = new List<DevicesContainer>();

        protected void Page\_Load(object sender, EventArgs e)

        {

            DataFolder = Server.MapPath("/AppData/DataFolder/");

            ResultFolder = Server.MapPath("/AppData/ResultFolder/");

        }

        protected void Button1\_Click(object sender, EventArgs e)

        {

            File.Delete(Server.MapPath("AppData/Rezultatai.txt"));

            if (ProperFiles(FileUpload1))

            {

                CopyFilesToServer(FileUpload1, DataFolder);

                Directory.Delete(ResultFolder, true);

                Directory.CreateDirectory(ResultFolder);

                try

                {

                    Debug.WriteLine("ABOBA");

                    AllContainers = InOut.ReadDevices(DataFolder);

                    AllDataContainers = InOut.ReadDevices(DataFolder);

                }

                catch (Exception ex)

                {

                    Label5.Visible = true;

                    Debug.WriteLine("ABOBA2");

                    Label5.BackColor = System.Drawing.Color.Red;

                    Debug.WriteLine("Nepavyko nuskaityti bent vieno is duomenu failu:" + ex.Message);

                    Label5.Text = "Nepavyko nuskaityti bent vieno is duomenu failu:" + ex.Message;

                    //Controls.Add(new Label() { Text = "Nepavyko nuskaityti bent vieno is duomenu failu:" + ex.Message }); testavimui

                }

                if (AllContainers.Count > 0)

                {

                    string DataFormat = string.Format("| {0,20} | {1,20} | {2, 4} | {3, 20} | {4,-8} |",

                        "Gamintojas", "Modelis", "Energ. Klase", "Spalva", "Kaina");

                    InOut.PrintStarting(AllContainers, ResultFolder + "Rezultatai.txt", "Pradiniai duomenys", DataFormat);

                    //Task 1

                    Table1.Visible= true;

                    Label1.Visible= true;

                    Label1.Text = "Siemens firmos gaminiu viso rasta:";

                    int SiemensCount = TaskUtils.SiemensCount(AllContainers);

                    AddSiemensCountToTable(Table1, SiemensCount);

                    InOut.PrintToTxt(ResultFolder + "Rezultatai.txt", SiemensCount, "Siemens firmos gaminiu viso rasta: ");

                    //Task2

                    Label2.Visible= true;

                    Table2.Visible= true;

                    Label2.Text = "10 Pigiausių pastatomų šaldytuvų su talpa didesne nei 80 sąrašas:";

                    List<Fridge> fridges = TaskUtils.StandingCheapestFridges(AllContainers);

                    CreateCheapFridgeTable(Table2, fridges);

                    InOut.PrintToTxt(ResultFolder + "Rezultatai.txt", fridges, "10 Pigiausių pastatomų šaldytuvų su talpa didesne nei 80 sąrašas:");

                    //Task3

                    Label3.Visible= true;

                    Label3.Text = "Išskirtinai parduodamų prekių sąrašas faile TikTen.csv";

                    List<DevicesContainer> exclusive = TaskUtils.OnlyOneShop(AllContainers);

                    InOut.PrintToCsvOnlyOneShop(exclusive, ResultFolder + "TikTen.csv");

                    //Task4

                    Label4.Visible= true;

                    Label4.Text = "Brangių buitinių prietaisų sąrašas sudarytas ir išvestas į Brangus.csv";

                    List<Device> expensive = TaskUtils.ExpensiveDevices(AllContainers);

                    InOut.PrintToCsvExpensive(expensive, ResultFolder + "Brangus.csv");

                }

                else

                {

                    Label5.Visible = true;

                    Label5.BackColor = System.Drawing.Color.Red;

                    Label5.Text = "Sąrašas tusčias";

                }

            }

        }

    }

}

**Style.css failas:**

body {

    background-color: ghostwhite;

}

Button {

    Font-Size : X-Large;

    Height : 50px;

    Width : 160px;

}

table {

    background-color: linen;

    border: 1px dotted;

    border-color: black;

}

td {

    padding: 2px;

    border: 1px solid;

}

.Label {

    color: red;

    font-size: 16px;

}

.MainLabelTop {

    color: black;

    font-family: 'Times New Roman';

    font-size: 20px;

}

## Pradiniai duomenys ir rezultatai

Pradinius duomenis pasidariau tokius, kuriuose saugoma parduotuves pavadinimas, adresas, telefono numeris ir tada kiekviename faile yra bent po 5 kiekvieno tipo produkto: Šaldytuvas, Mikrobangė, Virdulys

**U10a.csv failas:**

Euronics

Mindaugo g. 11, Vilnius

867745032

Fridge;Haier;HRF-521DS6;A+;Silver;1850.00;500;Freestanding;Yes;192;90;70;

Fridge;Indesit;LI8 S2E X;A++;Inox;1600.00;375;Freestanding;No;189;60;68;

Fridge;Miele;KFN 29142 D;A++;White;2200.00;390;Freestanding;Yes;201;60;66;

Fridge;Hisense;RB412N4AF1;A+;Black;1099.00;350;Freestanding;Yes;185;60;55;

Fridge;Electrolux;EN3481AOX;A+;Silver;1700.00;380;Freestanding;Yes;200;60;60;

Oven;Zanussi;ZOP37982XK;A+;Stainless Steel;850.00;3200;9;

Oven;Beko;BIM24300BS;A;Black;700.00;3400;8;

Oven;Candy;FCXP615X;A+;Stainless Steel;650.00;3300;10;

Oven;Hotpoint;SI6864SHIX;A+;Stainless Steel;780.00;3650;13;

Oven;Samsung;NV75N5641RS;A;Dual Cook;1100.00;3500;7;

Kettle;Tefal;KO3308;A;Black;50.00;2400;1.5;

Kettle;Morphy Richards;102033;A;White;55.00;2200;1.7;

Kettle;De'Longhi;KBOV2001.BG;A;Beige;60.00;2000;1.5;

Kettle;Bosch;TWK7805;A;Grey;40.00;3100;1.7;

Kettle;Russell Hobbs;23912;A;Silver;45.00;2400;1.7;

**U10b.csv failas:**

Euronics

Savanorių pr. 321, Klaipėda

867965498

Fridge;GE;GNE25JSKSS;A+;Stainless Steel;2100.00;510;Freestanding;Yes;175;70;67;

Fridge;Bosch;KGV39VW31;A++;White;1700.00;340;Freestanding;Yes;201;60;65;

Fridge;Liebherr;CNel 4313;A++;Silver;1950.00;310;Freestanding;No;185;60;63;

Fridge;LG;GBB72PZEFN;A+++;Graphite;2200.00;384;Freestanding;Yes;203;60;68;

Fridge;Samsung;RT38K5030S8;A+;Silver;1500.00;375;Freestanding;Yes;178;67;66;

Oven;Electrolux;EOF5C70X;A;Stainless Steel;950.00;3500;7;

Oven;Siemens;HB578G0S00;A+;Silver;1200.00;3600;15;

Oven;Samsung;NV70K1340BS/EU;A;Black;800.00;3450;6;

Oven;Whirlpool;AKZ96290IX;A++;Stainless Steel;1000.00;3400;16;

Oven;Beko;BIE22300X;A;Stainless Steel;650.00;3200;8;

Kettle;KitchenAid;5KEK1222;A;Red;65.00;1500;1.25;

Kettle;Smeg;KLF03PBUK;A;Blue;70.00;2400;1.7;

Kettle;Hamilton Beach;40998;A;Stainless Steel;45.00;1500;1.0;

Kettle;Kenwood;ZJP05.A0GY;A;Grey;50.00;2200;1.7;

Kettle;Russell Hobbs;20415;A;Black;55.00;2400;1.7;

**U10c.csv failas:**

Euronics

Žirmūnų g. 64, Šiauliai

867850940

Fridge;LG;GSX961NSVZ;A++;Black;2600.00;601;Freestanding;Yes;179;91;74;

Fridge;Samsung;RB38T602DSA;A++;Graphite;1700.00;385;Built-in;No;203;59;65;

Fridge;Fisher & Paykel;RF522ADX5;A+;Stainless Steel;2300.00;475;Freestanding;Yes;172;79;70;

Fridge;Beko;RCNE560E40ZXBN;A+++;Inox;1900.00;505;Freestanding;Yes;192;70;68;

Fridge;AEG;RMB76121NX;A+;Stainless Steel;2500.00;560;Freestanding;Yes;177;91;73;

Oven;Whirlpool;AKZ96270IX;A++;Stainless Steel;700.00;3650;16;

Oven;AEG;BPE556320M;A+;Stainless Steel;1150.00;3500;10;

Oven;Bosch;HBA534BS0B;A;Black;900.00;3600;8;

Oven;Zanussi;ZOP37982XU;A++;Stainless Steel;850.00;3200;10;

Oven;Neff;B3ACE4HN0B;A;Stainless Steel;1000.00;3400;12;

Kettle;Russell Hobbs;20412;A;Silver;35.00;2400;1.7;

Kettle;Breville;VKJ956;A;Stainless Steel;45.00;3000;1.7;

Kettle;Panasonic;NC-GK1;A;White;55.00;2200;1.7;

Kettle;Morphy Richards;108262;A;Cream;50.00;3000;1.5;

Kettle;Braun;WK500;A;Black;60.00;3000;1.7;

Pirmas testas:

Pasileidžiu programą ir teisingai pasirenku visus tris duomenų failus. Juos užkrauna ir viską ko reikia atlieka korektiškai. Išveda rezultatų failus ir atvaizduoja ekrane rezultatus:

A screenshot of a computer

Description automatically generated

**Rezultatai.txt failas:**

Pradiniai duomenys

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

Euronics

Mindaugo g. 11, Vilnius

867745032

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

| Gamintojas | Modelis | Energ. Klase | Spalva | Kaina |

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

Haier HRF-521DS6 A+ Silver 1850.00e 500L Freestanding True 192m 90m 90m

Indesit LI8 S2E X A++ Inox 1600.00e 375L Freestanding False 189m 60m 60m

Miele KFN 29142 D A++ White 2200.00e 390L Freestanding True 201m 60m 60m

Hisense RB412N4AF1 A+ Black 1099.00e 350L Freestanding True 185m 60m 60m

Electrolux EN3481AOX A+ Silver 1700.00e 380L Freestanding True 200m 60m 60m

Zanussi ZOP37982XK A+ Stainless Steel 850.00e 3200W 9

Beko BIM24300BS A Black 700.00e 3400W 8

Candy FCXP615X A+ Stainless Steel 650.00e 3300W 10

Hotpoint SI6864SHIX A+ Stainless Steel 780.00e 3650W 13

Samsung NV75N5641RS A Dual Cook 1100.00e 3500W 7

Tefal KO3308 A Black 50.00e 2400W 1.5L

Morphy Richards 102033 A White 55.00e 2200W 1.7L

De'Longhi KBOV2001.BG A Beige 60.00e 2000W 1.5L

Bosch TWK7805 A Grey 40.00e 3100W 1.7L

Russell Hobbs 23912 A Silver 45.00e 2400W 1.7L

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

Pradiniai duomenys

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

Euronics

Savanorių pr. 321, Klaipėda

867965498

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

| Gamintojas | Modelis | Energ. Klase | Spalva | Kaina |

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

GE GNE25JSKSS A+ Stainless Steel 2100.00e 510L Freestanding True 175m 70m 70m

Bosch KGV39VW31 A++ White 1700.00e 340L Freestanding True 201m 60m 60m

Liebherr CNel 4313 A++ Silver 1950.00e 310L Freestanding False 185m 60m 60m

LG GBB72PZEFN A+++ Graphite 2200.00e 384L Freestanding True 203m 60m 60m

Samsung RT38K5030S8 A+ Silver 1500.00e 375L Freestanding True 178m 67m 67m

Electrolux EOF5C70X A Stainless Steel 950.00e 3500W 7

Siemens HB578G0S00 A+ Silver 1200.00e 3600W 15

Samsung NV70K1340BS/EU A Black 800.00e 3450W 6

Whirlpool AKZ96290IX A++ Stainless Steel 1000.00e 3400W 16

Beko BIE22300X A Stainless Steel 650.00e 3200W 8

KitchenAid 5KEK1222 A Red 65.00e 1500W 1.25L

Smeg KLF03PBUK A Blue 70.00e 2400W 1.7L

Hamilton Beach 40998 A Stainless Steel 45.00e 1500W 1L

Kenwood ZJP05.A0GY A Grey 50.00e 2200W 1.7L

Russell Hobbs 20415 A Black 55.00e 2400W 1.7L

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

Pradiniai duomenys

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

Euronics

Žirmūnų g. 64, Šiauliai

867850940

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

| Gamintojas | Modelis | Energ. Klase | Spalva | Kaina |

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

LG GSX961NSVZ A++ Black 2600.00e 601L Freestanding True 179m 91m 91m

Samsung RB38T602DSA A++ Graphite 1700.00e 385L Built-in False 203m 59m 59m

Fisher & Paykel RF522ADX5 A+ Stainless Steel 2300.00e 475L Freestanding True 172m 79m 79m

Beko RCNE560E40ZXBN A+++ Inox 1900.00e 505L Freestanding True 192m 70m 70m

AEG RMB76121NX A+ Stainless Steel 2500.00e 560L Freestanding True 177m 91m 91m

Whirlpool AKZ96270IX A++ Stainless Steel 700.00e 3650W 16

AEG BPE556320M A+ Stainless Steel 1150.00e 3500W 10

Bosch HBA534BS0B A Black 900.00e 3600W 8

Zanussi ZOP37982XU A++ Stainless Steel 850.00e 3200W 10

Neff B3ACE4HN0B A Stainless Steel 1000.00e 3400W 12

Russell Hobbs 20412 A Silver 35.00e 2400W 1.7L

Breville VKJ956 A Stainless Steel 45.00e 3000W 1.7L

Panasonic NC-GK1 A White 55.00e 2200W 1.7L

Morphy Richards 108262 A Cream 50.00e 3000W 1.5L

Braun WK500 A Black 60.00e 3000W 1.7L

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

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

Siemens firmos gaminiu viso rasta: 1

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

10 Pigiausių pastatomų šaldytuvų su talpa didesne nei 80 sąrašas:

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

Hisense RB412N4AF1 1099.00 350

Samsung RT38K5030S8 1500.00 375

Indesit LI8 S2E X 1600.00 375

Electrolux EN3481AOX 1700.00 380

Bosch KGV39VW31 1700.00 340

Haier HRF-521DS6 1850.00 500

Beko RCNE560E40ZXBN 1900.00 505

Liebherr CNel 4313 1950.00 310

GE GNE25JSKSS 2100.00 510

Miele KFN 29142 D 2200.00 390

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

**TikTen.csv failas:**

**;**

**;**

**Saldytuvas**

**Haier;HRF-521DS6**

**Saldytuvas**

**Indesit;LI8 S2E X**

**Saldytuvas**

**Miele;KFN 29142 D**

**Saldytuvas**

**Hisense;RB412N4AF1**

**Saldytuvas**

**Electrolux;EN3481AOX**

**Mikrobange**

**Zanussi;ZOP37982XK**

**Mikrobange**

**Beko;BIM24300BS**

**Mikrobange**

**Candy;FCXP615X**

**Mikrobange**

**Hotpoint;SI6864SHIX**

**Mikrobange**

**Samsung;NV75N5641RS**

**Virdulys**

**Tefal;KO3308**

**Virdulys**

**Morphy Richards;102033**

**Virdulys**

**De'Longhi;KBOV2001.BG**

**Virdulys**

**Bosch;TWK7805**

**Virdulys**

**Russell Hobbs;23912**

**;**

**;**

**Saldytuvas**

**GE;GNE25JSKSS**

**Saldytuvas**

**Bosch;KGV39VW31**

**Saldytuvas**

**Liebherr;CNel 4313**

**Saldytuvas**

**LG;GBB72PZEFN**

**Saldytuvas**

**Samsung;RT38K5030S8**

**Mikrobange**

**Electrolux;EOF5C70X**

**Mikrobange**

**Siemens;HB578G0S00**

**Mikrobange**

**Samsung;NV70K1340BS/EU**

**Mikrobange**

**Whirlpool;AKZ96290IX**

**Mikrobange**

**Beko;BIE22300X**

**Virdulys**

**KitchenAid;5KEK1222**

**Virdulys**

**Smeg;KLF03PBUK**

**Virdulys**

**Hamilton Beach;40998**

**Virdulys**

**Kenwood;ZJP05.A0GY**

**Virdulys**

**Russell Hobbs;20415**

**;**

**;**

**Saldytuvas**

**LG;GSX961NSVZ**

**Saldytuvas**

**Samsung;RB38T602DSA**

**Saldytuvas**

**Fisher & Paykel;RF522ADX5**

**Saldytuvas**

**Beko;RCNE560E40ZXBN**

**Saldytuvas**

**AEG;RMB76121NX**

**Mikrobange**

**Whirlpool;AKZ96270IX**

**Mikrobange**

**AEG;BPE556320M**

**Mikrobange**

**Bosch;HBA534BS0B**

**Mikrobange**

**Zanussi;ZOP37982XU**

**Mikrobange**

**Neff;B3ACE4HN0B**

**Virdulys**

**Russell Hobbs;20412**

**Virdulys**

**Breville;VKJ956**

**Virdulys**

**Panasonic;NC-GK1**

**Virdulys**

**Morphy Richards;108262**

**Virdulys**

**Braun;WK500**

**Brangus.csv failas:**

**Saldytuvas**

**Fisher & Paykel;RF522ADX5;A+;Stainless Steel;2300.00;475;Freestanding;True;172;79;79;**

**Saldytuvas**

**GE;GNE25JSKSS;A+;Stainless Steel;2100.00;510;Freestanding;True;175;70;70;**

**Saldytuvas**

**AEG;RMB76121NX;A+;Stainless Steel;2500.00;560;Freestanding;True;177;91;91;**

**Saldytuvas**

**Samsung;RT38K5030S8;A+;Silver;1500.00;375;Freestanding;True;178;67;67;**

**Saldytuvas**

**LG;GSX961NSVZ;A++;Black;2600.00;601;Freestanding;True;179;91;91;**

**Saldytuvas**

**Hisense;RB412N4AF1;A+;Black;1099.00;350;Freestanding;True;185;60;60;**

**Saldytuvas**

**Liebherr;CNel 4313;A++;Silver;1950.00;310;Freestanding;False;185;60;60;**

**Saldytuvas**

**Indesit;LI8 S2E X;A++;Inox;1600.00;375;Freestanding;False;189;60;60;**

**Saldytuvas**

**Haier;HRF-521DS6;A+;Silver;1850.00;500;Freestanding;True;192;90;90;**

**Saldytuvas**

**Beko;RCNE560E40ZXBN;A+++;Inox;1900.00;505;Freestanding;True;192;70;70;**

**Saldytuvas**

**Electrolux;EN3481AOX;A+;Silver;1700.00;380;Freestanding;True;200;60;60;**

**Saldytuvas**

**Miele;KFN 29142 D;A++;White;2200.00;390;Freestanding;True;201;60;60;**

**Saldytuvas**

**Bosch;KGV39VW31;A++;White;1700.00;340;Freestanding;True;201;60;60;**

**Saldytuvas**

**LG;GBB72PZEFN;A+++;Graphite;2200.00;384;Freestanding;True;203;60;60;**

**Saldytuvas**

**Samsung;RB38T602DSA;A++;Graphite;1700.00;385;Built-in;False;203;59;59;**

**Mikrobange**

**Zanussi;ZOP37982XK;A+;Stainless Steel;850.00;3200;9**

**Mikrobange**

**Beko;BIE22300X;A;Stainless Steel;650.00;3200;8**

**Mikrobange**

**Zanussi;ZOP37982XU;A++;Stainless Steel;850.00;3200;10**

**Mikrobange**

**Candy;FCXP615X;A+;Stainless Steel;650.00;3300;10**

**Mikrobange**

**Beko;BIM24300BS;A;Black;700.00;3400;8**

**Mikrobange**

**Whirlpool;AKZ96290IX;A++;Stainless Steel;1000.00;3400;16**

**Mikrobange**

**Neff;B3ACE4HN0B;A;Stainless Steel;1000.00;3400;12**

**Mikrobange**

**Samsung;NV70K1340BS/EU;A;Black;800.00;3450;6**

**Mikrobange**

**Samsung;NV75N5641RS;A;Dual Cook;1100.00;3500;7**

**Mikrobange**

**Electrolux;EOF5C70X;A;Stainless Steel;950.00;3500;7**

**Mikrobange**

**AEG;BPE556320M;A+;Stainless Steel;1150.00;3500;10**

**Mikrobange**

**Siemens;HB578G0S00;A+;Silver;1200.00;3600;15**

**Mikrobange**

**Bosch;HBA534BS0B;A;Black;900.00;3600;8**

**Mikrobange**

**Hotpoint;SI6864SHIX;A+;Stainless Steel;780.00;3650;13**

**Mikrobange**

**Whirlpool;AKZ96270IX;A++;Stainless Steel;700.00;3650;16**

**Virdulys**

**KitchenAid;5KEK1222;A;Red;65.00;1500;1.25**

**Virdulys**

**De'Longhi;KBOV2001.BG;A;Beige;60.00;2000;1.5**

**Virdulys**

**Morphy Richards;102033;A;White;55.00;2200;1.7**

**Virdulys**

**Panasonic;NC-GK1;A;White;55.00;2200;1.7**

**Virdulys**

**Smeg;KLF03PBUK;A;Blue;70.00;2400;1.7**

**Virdulys**

**Russell Hobbs;20415;A;Black;55.00;2400;1.7**

**Virdulys**

**Braun;WK500;A;Black;60.00;3000;1.7**

**Antras Testas:**

Antruoju testu, pabandau paleisti programą pasirinkęs neteisingą duomenų failą. Tokį pranešimą gaunu ekrane:

A screenshot of a computer

Description automatically generated

Kadangi programa negalėjo teisingai nuskaityti duomenų, todėl parodo, kad sąrašas tuščias, bei kode meta klaidos pranešimą(exception)

## Dėstytojo pastabos

6 už programą

Ataskaita: LD4 ataskaita: (pak.) P4 (-0.2), P6 (-0.2), P7 (-0.2), P13 (-0.2). Įv.: 0.2

Testukas 0/3

# Deklaratyvusis programavimas (L5)

## Darbo užduotis

**U5\_10. Pajamos.** Žmonės užsisako spaudą. Užsakymas vyksta metų ribose. Pirmoje failo eilutėje nurodyta įvedimo data (failų daug), o tolesnėse eilutėse nurodyta prenumeratoriaus pavardė, adresas, laikotarpio pradžia (nurodyta sveiku skaičiumi 1..12), laikotarpio ilgis, leidinio kodas, leidinių kiekis. Atskirame 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. Sudarykite to paties nurodyto mėnesio prenumeratorių sąrašą.

## Grafinės vartotojo sąsajos schema

A screenshot of a computer

Description automatically generated

## Sąsajoje panaudotų komponentų keičiamos savybės

|  |  |  |
| --- | --- | --- |
| Komponentas | Savybė | Reikšmė |
| Label | Text | LD5\_10 Martynas Kuliešius |
| Label | Text | Pasirinkite užsakovų failus |
| FileUpload | ID, AllowMultiple, BackColor, BorderColor, BorderStyle, Height, Width | FileUpload1, True, #CCFF99, Black, Groove, 20px, 210px |
| Label | Text | Pasirinkite leidėjų failą |
| FileUpload | ID, BackColor, BorderColor, BorderStyle, Height, Width | FileUpload2, cyan, Black, Groove, 20px, 210px |
| Button | Text, ID, Font-Size, Height, Width | Skaityti failus, Button1, X-Large, 50px, 160px |
| Label | Id, Visible | Label1, False |
| Button | Text, ID, Width, Visible | Rodyti pasirinkto mėnesio pajamas, Button2, 170px, false |
| Button | Text, ID, Width, Visible | Surikiuoti leidėjų pajamas, Button3, 170px, false |
| Button | Text, ID, Width, Visible | Nurodyto mėnesio prenumeratorių sąrašas, Button4, 170px, false |
| DropDownList | ID, Visible, ListItem (text, value) | DropDown1, false, [(Sausis, 1)(Vasaris, 2)(Kovas, 3)(Balandis, 4)(Gegužė, 5)(Birželis, 6)(Liepa, 7)(Rugpjūtis, 8)(Rugsėjis, 9)(Spalis, 10)(Lapkritis, 11)(Gruodis, 12)] |
| Table | ID, Visible | Table1, false |

## Klasių diagrama

A diagram of a workflow

Description automatically generated with medium confidence

## Programos vartotojo vadovas

Pasileidus programą matomas toks vaizdas(matomas apačioje).

A screenshot of a computer

Description automatically generated

Pasirenkami norimi failai skaitymui. Paspaudus mygtuką „Skaityti failus“ programa nuskaito pasirinktus failus ir jį informaciją išsaugo atitinkamuose sąrašuose. Ir programa praneša ar failai nuskaityti teisingai, taip pat programa vykdo išimčių valdymą šiuo momentu, jeigu neteisingi failai arba nepasirinkti failai, pranešama vartotojui. Tuo pačiu atveria programos funkcionalumą: atskleidžia tris mygtukus bei pasirinkimo sąrašą.

A screenshot of a computer

Description automatically generated

Programos naudojimas labai paprastas. Pasirenkamas norimas mėnuo iš sąrašo (numeriukas dropdown sąraše) ir spaudžiamas mygtukas Rodyti mėnesio pajamas.

A screenshot of a computer

Description automatically generated

Tada galima paspausti mygtuką „Surikiuoti leidėjų pajamas“ tam, kad parodytų surikiuotas leidėjų pajamas:

A screenshot of a computer

Description automatically generated

Taip pat galima atvaizduoti nurodyto mėnesio prenumeratorių sąrašą paspaudus mygtuką „Nurodyto mėnesio prenumeratorių sąrašas“:

A screenshot of a computer

Description automatically generated

## Programos tekstas

**Order.cs failas:**

namespace LD5\_10\_MKuliesius.AppCode

{

    /// <summary>

    /// Class of each order(persons order information)

    /// </summary>

    public class Order

    {

        public string Name { get; set; }

        public string Address { get; set; }

        public int StartMonth { get; set; }

        public int Duration { get; set; }

        public string PublicationCode { get; set; }

        public Order() { }

        public Order(string name, string address, int startMonth, int duration, string publicationCode)

        {

            this.Address = address;

            this.Name = name;

            this.StartMonth = startMonth;

            this.Duration = duration;

            this.PublicationCode = publicationCode;

        }

        public override string ToString()

        {

            return Name;

        }

    }

}

**Publication.cs failas:**

namespace LD5\_10\_MKuliesius.AppCode

{

    /// <summary>

    /// Class of publication publisher. Contains the publication code, publisher name, pricePerMoth and profit for month variables.

    /// </summary>

    public class Publication

    {

        public string Code { get; set; }

        public string Name { get; set; }

        public string PublisherName { get; set; }

        public decimal PricePerMonth { get; set; }

        public decimal ProfitMonth { get; set; }

        public Publication() { }

        public Publication(string code, string name, string publisherName, decimal price)

        {

            this.Code = code;

            this.Name = name;

            this.PublisherName = publisherName;

            this.PricePerMonth = price;

            ProfitMonth = 0;

        }

    }

}

**OrderContainer.cs failas:**

using System;

using System.Collections.Generic;

namespace LD5\_10\_MKuliesius.AppCode

{

    /// <summary>

    /// A container to store all orders for a given date

    /// </summary>

    public class OrderContainer

    {

        public DateTime Date;

        public List<Order> Orders;

        public OrderContainer() { }

        public OrderContainer(DateTime date, List<Order> orders)

        {

            this.Date = date;

            this.Orders = orders;

        }

        public Order GetOrder(int id)

        {

            return Orders[id];

        }

        public List<Order> GetAllOrders()

        {

            return Orders;

        }

    }

}

**FormExtention.cs failas:**

using LD5\_10\_MKuliesius.AppCode;

using System.Collections.Generic;

using System.IO;

using System.Web;

using System.Web.UI.WebControls;

namespace LD5\_10\_MKuliesius

{

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

    {

        /// <summary>

        /// Checks if correct files were uploaded

        /// </summary>

        /// <param name="fileUpload"> fileUpload element from form</param>

        /// <returns></returns>

        public bool ProperFiles(FileUpload fileUpload)

        {

            foreach (HttpPostedFile file in fileUpload.PostedFiles)

            {

                if (!(file.FileName.EndsWith(".txt") || file.FileName.EndsWith(".csv")))

                {

                    return false;

                }

            }

            return fileUpload.HasFiles;

        }

        /// <summary>

        /// Copies files to the folder

        /// </summary>

        /// <param name="fileUpload"> fileupload element from form</param>

        /// <param name="path"> path to create/delete</param>

        public void CopyFilesToServer(FileUpload fileUpload, string path)

        {

            IList<HttpPostedFile> files = fileUpload.PostedFiles;

            Directory.Delete(path, true);

            Directory.CreateDirectory(path);

            for (int i = 0; i < fileUpload.PostedFiles.Count; i++)

            {

                files[i].SaveAs(path + files[i].FileName);

            }

        }

        /// <summary>

        /// Binds months to dropdownList

        /// </summary>

        private void BindMonthsDropDown()

        {

            // Clear existing items

            DropDown1.Items.Clear();

            // Add items to the dropdown

            for (int i = 1; i <= 12; i++)

            {

                DropDown1.Items.Add(new ListItem(i.ToString(), i.ToString()));

            }

        }

        /// <summary>

        /// Method to display selected months revenues to table

        /// </summary>

        /// <param name="table"> table variable</param>

        /// <param name="publications"> list of publishers</param>

        /// <param name="selectedMonth"> selected month variable</param>

        public static void AddRevenueToTable(Table table, List<Publication> publications, int selectedMonth)

        {

            TableCell text = new TableCell()

            {

                Text = $"Pasirinktam mėnesiui {selectedMonth} leidėjų uždarbiai:"

            };

            TableRow tablerow = new TableRow()

            {

                Cells = { text }

            };

            table.Rows.Add(tablerow);

            foreach (Publication publication in publications)

            {

                string publisherInfo = $"Leidėjas {publication.PublisherName} už {publication.Name} uždirbo {publication.ProfitMonth}";

                table.Rows.Add(new TableRow { Cells = { new TableCell { Text = publisherInfo } } });

            }

        }

        /// <summary>

        /// Adds subscribers to table

        /// </summary>

        /// <param name="table"> table variable </param>

        /// <param name="orders"> List of subscribers </param>

        /// <param name="selectedMonth"> selected month variable </param>

        public static void AddSubscribersToTable(Table table, List<Order> orders, int selectedMonth)

        {

            TableCell text = new TableCell()

            {

                Text = $"Pasirinkto mėnesio {selectedMonth} prenumeratoriai:"

            };

            TableRow tablerow = new TableRow()

            {

                Cells = { text }

            };

            table.Rows.Add(tablerow);

            foreach(Order order in orders)

            {

                TableCell name = new TableCell()

                {

                    Text = order.Name

                };

                TableCell address = new TableCell()

                {

                    Text = order.Address

                };

                TableCell publication = new TableCell()

                {

                    Text = order.PublicationCode

                };

                TableRow tableRow = new TableRow()

                {

                    Cells =

                    {

                        name,

                        address,

                        publication

                    }

                };

                table.Rows.Add(tableRow);

            }

        }

    }

}

**InOut.cs failas:**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

namespace LD5\_10\_MKuliesius.AppCode

{

    public class InOut

    {

        /// <summary>

        /// Methods reads all files in filepath given by the user. Files contain information about orders. And also handles exceptions

        /// </summary>

        /// <param name="filePath"> File path string</param>

        /// <returns></returns>

        /// <exception cref="Exception">possible exceptions for missreading files</exception>

        public static List<OrderContainer> ReadOrders(string filePath)

        {

            List<OrderContainer> allOrders = new List<OrderContainer>();

            try

            {

                foreach (string file in Directory.GetFiles(filePath))

                {

                    List<Order> orders = new List<Order>();

                    string[] Lines = File.ReadAllLines(file);

                    if (Lines.Length >= 2)

                    {

                        DateTime date = Convert.ToDateTime(Lines[0]);

                        for (int i = 1; i < Lines.Length; i++)

                        {

                            string[] Bits = Lines[i].Split(new[] { ";" }, StringSplitOptions.RemoveEmptyEntries);

                            if (Bits.Length < 6 || Bits.Length > 6)

                            {

                                Debug.WriteLine("ABOBA DUOMENYS NE TOKIO TIPO");

                                throw new Exception($"Data incorrectly formatted at line {i + 1}");

                            }

                            string Name = Bits[0];

                            string Address = Bits[1];

                            int startMonth = Convert.ToInt32(Bits[2]);

                            int duration = Convert.ToInt32(Bits[3]);

                            string PubCode = Bits[4];

                            Order newOrder = new Order(Name, Address, startMonth, duration, PubCode);

                            orders.Add(newOrder);

                        }

                        OrderContainer newOrderContainer = new OrderContainer(date, orders);

                        allOrders.Add(newOrderContainer);

                    }

                    else

                    {

                        Debug.WriteLine("ABOBA PER MAZAI DUOMENU FAILE");

                        throw new Exception($"Per mažai duomenų");

                    }

                }

            }

            catch (Exception ex)

            {

                Debug.WriteLine("ABOBA NEPAVYKO NUSKAITYTI");

                throw new Exception(ex.Message + " Klaida");

            }

            return allOrders;

        }

        /// <summary>

        /// Method reads all lines in a given file. And also handles exceptions

        /// </summary>

        /// <param name="filePath">string variable with the path to file</param>

        /// <returns></returns>

        /// <exception cref="Exception">possible exceptions for missreading files</exception>

        public static List<Publication> ReadPublications(string filePath)

        {

            List<Publication> allPublications = new List<Publication>();

            try

            {

                string[] lines = File.ReadAllLines(filePath);

                foreach (string line in lines)

                {

                    string[] parts = line.Split(';');

                    string code = parts[0];

                    string name = parts[1];

                    string publisherName = parts[2];

                    decimal price = Convert.ToDecimal(parts[3]);

                    Publication newPub = new Publication(code, name, publisherName, price);

                    allPublications.Add(newPub);

                }

                return allPublications;

            }

            catch(Exception ex)

            {

                throw new Exception(ex.Message);

            }

        }

    }

}

**TaskUtils.cs failas:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace LD5\_10\_MKuliesius.AppCode

{

    public class TaskUtils

    {

        /// <summary>

        /// Method calculates revenue for every publisher for selected month

        /// </summary>

        /// <param name="orderContainers"> container of all orders </param>

        /// <param name="publications"> list of all publishers </param>

        /// <param name="specifiedMonth"> selected month variable</param>

        /// <returns></returns>

        public static List<Publication> CalculateRevenueByPublisher(List<OrderContainer> orderContainers, List<Publication> publications, int specifiedMonth)

        {

            // Reset ProfitMonth for all publications

            foreach (var publication in publications)

            {

                publication.ProfitMonth = 0;

            }

            // Aggregate all orders from all order containers

            var allOrders = orderContainers.SelectMany(oc => oc.GetAllOrders()).ToList();

            foreach (var order in allOrders)

            {

                if (order.StartMonth <= specifiedMonth && (order.StartMonth + order.Duration - 1) >= specifiedMonth)

                {

                    var publication = publications.FirstOrDefault(p => p.Code == order.PublicationCode);

                    if (publication != null)

                    {

                        int months = Math.Min(order.Duration, specifiedMonth - order.StartMonth + 1);

                        decimal revenue = months \* publication.PricePerMonth;

                        publication.ProfitMonth += revenue;

                    }

                }

            }

            return publications;

        }

        /// <summary>

        /// Sorts given list by revenue and name

        /// </summary>

        /// <param name="publications"> list of publishers that need to be sorted</param>

        /// <returns></returns>

        public static void SortPublicationsByRevenueAndName(List<Publication> publications)

        {

            if (publications == null)

            {

                // If the publications list is null, return immediately

                return;

            }

            // Sort publications by monthly revenue (descending) and then by name (ascending)

            publications.Sort((p1, p2) =>

            {

                // First, compare by profit month (descending)

                int compareResult = p2.ProfitMonth.CompareTo(p1.ProfitMonth);

                // If profit months are equal, compare by name (ascending)

                if (compareResult == 0)

                {

                    compareResult = p1.Name.CompareTo(p2.Name);

                }

                return compareResult;

            });

        }

        /// <summary>

        /// Method selects users for selected month

        /// </summary>

        /// <param name="orderContainers"> Container of all orders </param>

        /// <param name="selectedMonth"> selected month variable </param>

        /// <returns></returns>

        public static List<Order> GetSubscriptionsForSelectedMonth(List<OrderContainer> orderContainers, int selectedMonth)

        {

            // Get all orders for the selected month

            var selectedMonthOrders = orderContainers.SelectMany(oc => oc.Orders)

                                                     .Where(order => order.StartMonth <= selectedMonth &&

                                                                      order.StartMonth + order.Duration - 1 >= selectedMonth)

                                                     .ToList();

            return selectedMonthOrders;

        }

    }

}

**Forma.aspx failas:**

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma.aspx.cs" Inherits="LD5\_10\_MKuliesius.Forma" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

     <link rel="stylesheet" type="text/css" href="Style.css" />

</head>

<body>

    <form id="form1" runat="server">

        <div>

        <asp:Label runat="server" Text="LD5\_10 Martynas Kuliešius"></asp:Label>  <br /><br />

            <asp:Label runat="server" Text="Pasirinkite Užsakovų failus"></asp:Label> <br />

            <asp:FileUpload runat="server" ID="FileUpload1" AllowMultiple="True" BackColor="#CCFF99" BorderColor="Black" BorderStyle="Groove" Height="20px" Width="210px"></asp:FileUpload>  <br />

            <asp:Label runat="server" Text="Pasirinkite leidėjų failą"></asp:Label>  <br />

            <asp:FileUpload runat="server" ID="FileUpload2" BackColor="cyan" BorderColor="Black" BorderStyle="Groove" Height="20px " Width="210px"></asp:FileUpload> <br />

            <asp:Button runat="server" Text="Skaityti failus" OnClick="Button1\_Click" ID="Button1" Font-Size="X-Large" Height="50px" Width="160px"></asp:Button> <br />

            <asp:Label runat="server" Text="Label" ID="Label1" Visible="false"></asp:Label>

            <br />

            <br />

            <asp:Button runat="server" Text="Rodyti mėnesio pajamas" ID="Button2" Width="170px" Visible="false" OnClick="Button2\_Click"></asp:Button>          <asp:Button runat="server" Text="Surikiuoti leidėjų pajamas" ID="Button3" Visible="false" OnClick="Button3\_Click"></asp:Button>          <asp:Button runat="server" Text="Nurodyto mėnesio prenumeratorių sąrašas" ID="Button4" Visible="false" OnClick="Button4\_Click"></asp:Button>

            <br />

            <asp:DropDownList ID="DropDown1" runat="server" Visible="false">

                <asp:ListItem Text="Sausis" Value="1"></asp:ListItem>

                <asp:ListItem Text="Vasaris" Value="2"></asp:ListItem>

                <asp:ListItem Text="Kovas" Value="3"></asp:ListItem>

                <asp:ListItem Text="Balandis" Value="4"></asp:ListItem>

                <asp:ListItem Text="Gegužė" Value="5"></asp:ListItem>

                <asp:ListItem Text="Birželis" Value="6"></asp:ListItem>

                <asp:ListItem Text="Liepa" Value="7"></asp:ListItem>

                <asp:ListItem Text="Rugpjūtis" Value="8"></asp:ListItem>

                <asp:ListItem Text="Rugsėjis" Value="9"></asp:ListItem>

                <asp:ListItem Text="Spalis" Value="10"></asp:ListItem>

                <asp:ListItem Text="Lapkritis" Value="11"></asp:ListItem>

                <asp:ListItem Text="Gruodis" Value="12"></asp:ListItem>

            </asp:DropDownList><br />

            <br />

            <br />

            <asp:Table runat="server" ID="Table1" Visible="false"></asp:Table>

        </div>

    </form>

</body>

</html>

**Forma.aspx.cs failas:**

using LD5\_10\_MKuliesius.AppCode;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

namespace LD5\_10\_MKuliesius

{

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

    {

        string DataFolder;

        string ResultFolder;

        List<OrderContainer> OrdersContainer;

        List<Publication> Publications;

        /// <summary>

        /// Method active only on page start-up

        /// </summary>

        protected void Page\_Load(object sender, EventArgs e)

        {

            DataFolder = Server.MapPath("/AppData/DataFolder/");

            ResultFolder = Server.MapPath("/AppData/ResultFolder/");

            if (!IsPostBack)

            {

                BindMonthsDropDown();

            }

        }

        /// <summary>

        /// Method that activates when user presses first button

        /// </summary>

        protected void Button1\_Click(object sender, EventArgs e)

        {

            File.Delete(Server.MapPath("AppData/Rezultatai.txt"));

            if (ProperFiles(FileUpload1))

            {

                CopyFilesToServer(FileUpload1, DataFolder);

                //CopyFilesToServer(FileUpload2 , DataFolder); when using this, program breaks, so i just read the publication file in the normal way

                Directory.Delete(ResultFolder, true);

                Directory.CreateDirectory(ResultFolder);

                string path2 = Server.MapPath(FileUpload2.FileName);

                string file2 = Server.HtmlEncode(FileUpload2.FileName);

                string extension2 = Path.GetExtension(file2);

                try

                {

                    Debug.WriteLine("ABOBA");

                    OrdersContainer = InOut.ReadOrders(DataFolder);

                    FileUpload2.SaveAs(path2);

                    Publications = InOut.ReadPublications(path2);

                }

                catch (Exception ex)

                {

                    Debug.WriteLine("ABOBA FAILED READ FILES");

                    Label1.Visible = true;

                    Label1.BackColor = System.Drawing.Color.Red;

                    Label1.Text = "Nepavyko nuskaityti bent vieno is duomenu failu:" + ex.Message;

                    throw new Exception("Nepavyko nuskaityti bent vieno is duomenu failu:" + ex.Message);

                }

                if (OrdersContainer.Count > 0 && Publications.Count > 0)

                {

                    Label1.Visible = true;

                    Label1.BackColor = System.Drawing.Color.Green;

                    Label1.Text = "Failai nuskaityti teisingai";

                    Button2.Visible = true;

                    Button3.Visible = true;

                    Button4.Visible = true;

                    DropDown1.Visible = true;

                    // Storing lists to session

                    Session["OrdersContainer"] = OrdersContainer;

                    Session["Publications"] = Publications;

                }

                else

                {

                    Debug.WriteLine("ABOBA FAILED READ FILES");

                    Label1.Visible = true;

                    Label1.BackColor = System.Drawing.Color.Red;

                    Label1.Text = "Nepavyko nuskaityti failų";

                    throw new Exception("Programa nenuskaitė teisingai failų, prašome perkrauti");

                }

            }

            else

            {

                Debug.WriteLine("ABOBA FAILED UPLOAD FILES");

                Label1.Visible = true;

                Label1.BackColor = System.Drawing.Color.Red;

                Label1.Text = "Nepasirinkti failai arba netinkami failai";

            }

        }

        /// <summary>

        /// Method for button 2 handling and doing task

        /// </summary>

        /// <param name="sender"></param>

        /// <param name="e"></param>

        protected void Button2\_Click(object sender, EventArgs e)

        {

            Table1.Visible = true;

            // Retrieving OrdersContainer from Session

            List<OrderContainer> ordersContainer = Session["OrdersContainer"] as List<OrderContainer>;

            // Retrieving Publications from Session

            List<Publication> publications = Session["Publications"] as List<Publication>;

            if (DropDown1.SelectedValue != null)

            {

                int selectedMonth = int.Parse(DropDown1.SelectedValue);

                var updatedPublications = TaskUtils.CalculateRevenueByPublisher(ordersContainer, publications, selectedMonth);

                Session["UpdatedPublications"] = updatedPublications;

                Label1.BackColor = System.Drawing.Color.Green;

                Label1.Text = $"Leidėjų pasirinkto mėnesio {selectedMonth} uždirbtas pelnas atvaizduotas lentelėje";

                AddRevenueToTable(Table1, updatedPublications, selectedMonth );

            }

            else

            {

                Label1.BackColor = System.Drawing.Color.Red;

                Label1.Text = "Pasirinkite norimą mėnesį sąraše.";

            }

        }

        /// <summary>

        /// Method for button 3 handling and doing task

        /// </summary>

        /// <param name="sender"></param>

        /// <param name="e"></param>

        protected void Button3\_Click(object sender, EventArgs e)

        {

            Table1.Visible = true;

            // Retrieving Publications from Session

            List<Publication> publications = Session["UpdatedPublications"] as List<Publication>;

            int selectedMonth = int.Parse(DropDown1.SelectedValue);

            TaskUtils.SortPublicationsByRevenueAndName(publications);

            Label1.BackColor = System.Drawing.Color.Green;

            Label1.Text = $"Surikiuota leidėjų informacija atvaizduota lentelėje";

            AddRevenueToTable(Table1, publications, selectedMonth);

        }

        /// <summary>

        /// Method for button 4 handling and doing task

        /// </summary>

        /// <param name="sender"></param>

        /// <param name="e"></param>

        protected void Button4\_Click(object sender, EventArgs e)

        {

            Table1.Visible = true;

            // Retrieving OrdersContainer from Session

            List<OrderContainer> ordersContainer = Session["OrdersContainer"] as List<OrderContainer>;

            if (DropDown1.SelectedValue != null)

            {

                int selectedMonth = int.Parse(DropDown1.SelectedValue);

                var selectedSubscriptions = TaskUtils.GetSubscriptionsForSelectedMonth(ordersContainer, selectedMonth);

                Label1.BackColor = System.Drawing.Color.Green;

                Label1.Text = $"Leidėjų pasirinkto mėnesio {selectedMonth} uždirbtas pelnas atvaizduotas lentelėje";

                AddSubscribersToTable(Table1 , selectedSubscriptions, selectedMonth);

            }

            else

            {

                Label1.BackColor = System.Drawing.Color.Red;

                Label1.Text = "Pasirinkite norimą mėnesį sąraše.";

            }

        }

    }

}

**Style.css failas:**

body {

    background-color: ghostwhite;

}

Button {

    Font-Size: X-Large;

    Height: 50px;

    Width: 160px;

}

table {

    background-color: linen;

    border: 1px dotted;

    border-color: black;

}

td {

    padding: 2px;

    border: 1px solid;

}

.Label {

    color: red;

    font-size: 16px;

}

.MainLabelTop {

    color: black;

    font-family: 'Times New Roman';

    font-size: 20px;

}

## Pradiniai duomenys ir rezultatai

**Programos testavimui turiu susikūręs 8 (a-b) užsakovų failus ir 1 leidinių/leidyklų (l) failą:**

**U10a.txt:**

2024-05-13

Kuliesius;Kauno g.14 Kaunas;1;3;A123;5;

Borisas;Vilniaus g.15 Vilnius;2;3;A123;6;

Arelis;Kauno g.13 Kaunas;1;3;B456;6;

Bubelis;Klaipedos g.16 Klaipeda;1;8;C789;9;

Simanavicius;Laisves al.10 Kaunas;3;6;A123;7;

Petrauskas;Gedimino pr.9 Vilnius;4;12;B456;10;

Kazlauskiene;Saules g.11 Siauliai;5;5;C789;4;

Urbonas;Vytauto pr.12 Kaunas;6;7;A123;3;

**U10b.txt;**

2024-06-14

Jonaitis;Zemaitijos g.15 Siauliai;7;6;B456;8;

Petraitis;Naujoji g.16 Alytus;8;4;C789;2;

Kazlauskaite;Vilties g.17 Marijampole;9;5;A123;5;

Lukauskas;Ukmerges g.18 Vilnius;10;3;B456;6;

Grigaitis;Kestucio g.19 Kaunas;11;9;C789;7;

Januskaite;Savanoriu pr.20 Vilnius;12;8;A123;1;

Zukauskas;Taikos pr.21 Klaipeda;1;12;B456;9;

Navickas;Kalvariju g.22 Vilnius;2;2;C789;3;

**U10c.txt:**

2024-07-15

Jasaitis;Daukanto g.23 Panevezys;3;6;A123;4;

Tamasauskas;Liepų g.24 Siauliai;4;4;B456;5;

Rakauskiene;Jonavos g.25 Jonava;5;3;C789;7;

Matulionis;Sodų g.26 Palanga;6;8;A123;6;

Sukackas;Šiaurės pr.27 Vilnius;7;12;B456;2;

Pranskietis;Vilniaus g.28 Vilnius;8;2;C789;1;

Kairys;Ateities g.29 Kaunas;9;5;A123;9;

Butkevicius;Žirmūnų g.30 Vilnius;10;7;B456;3;

**U10d.txt;**

2024-08-16

Kavaliauskas;Žirmūnų g.31 Vilnius;11;6;C789;8;

Daukantas;Šermukšnių g.32 Kaunas;12;4;A123;5;

Mickevicius;Birutės g.33 Klaipeda;1;9;B456;2;

Kalanta;Ramunių g.34 Panevezys;2;12;C789;7;

Sernas;Žalgirio g.35 Vilnius;3;5;A123;6;

Petrosius;Pilies g.36 Kaunas;4;3;B456;4;

Stankevičius;Pylimo g.37 Vilnius;5;7;C789;9;

Arlauskas;Kęstučio g.38 Siauliai;6;8;A123;3;

**U10e.txt:**

2024-09-17

Jankauskas;Basanavičiaus g.1 Vilnius;1;6;A123;10;

Petraitytė;Gedimino pr.2 Kaunas;2;3;B456;8;

Zemaitis;Neries krantinė 3 Kaunas;3;4;C789;7;

Vaitkus;Trakų g.4 Vilnius;4;5;A123;6;

Girdenis;Birštono g.5 Kaunas;5;2;B456;9;

Šalkauskas;Nemuno g.6 Klaipeda;6;8;C789;5;

Ragauskas;Laisvės pr.7 Vilnius;7;12;A123;4;

Burokaitė;Šiaurės g.8 Panevezys;8;7;B456;3;

**U10f.txt;**

2024-10-18

Pavilonis;Žalgirio g.9 Siauliai;9;6;C789;10;

Kazlauskis;Liepų g.10 Marijampole;10;4;A123;8;

Šidlauskas;Vilniaus g.11 Vilnius;11;5;B456;7;

Dovydaitis;Vasario 16-osios g.12 Kaunas;12;3;C789;6;

Paulauskas;Kęstučio g.13 Vilnius;1;8;A123;5;

Banys;Vytauto pr.14 Kaunas;2;12;B456;9;

Matelis;Savanorių pr.15 Vilnius;3;5;C789;4;

Starkus;Kauno g.16 Klaipeda;4;7;A123;3;

**U10g.txt:**

2024-11-19

Giedraitis;J. Basanavičiaus g.17 Vilnius;5;6;B456;10;

Kvedaras;Alyvų g.18 Kaunas;6;4;C789;8;

Lukėnas;Gėlių g.19 Panevezys;7;5;A123;7;

Vasilauskas;Naujoji g.20 Klaipeda;8;3;B456;6;

Dvareckas;Žemaitės g.21 Marijampole;9;8;C789;5;

Paukštė;Tilto g.22 Siauliai;10;12;A123;4;

Sakalauskas;Ateities g.23 Kaunas;11;2;B456;9;

Žemkalnis;Kovo 11-osios g.24 Vilnius;12;7;C789;3;

**U10h.txt;**

2024-12-20

Ambrazevičius;Vilniaus g.25 Vilnius;1;6;A123;10;

Grigonis;Kauno g.26 Kaunas;2;4;B456;8;

Juodaitis;Jonavos g.27 Jonava;3;5;C789;7;

Kairys;Ukmergės g.28 Vilnius;4;3;A123;6;

Valickas;Ramunių g.29 Palanga;5;8;B456;5;

Masiulis;Nemuno g.30 Klaipeda;6;12;C789;4;

Baranauskas;Saulės g.31 Siauliai;7;2;A123;9;

Šiaulys;Žemaitijos g.32 Siauliai;8;7;B456;3;

**U10l.txt:**

A123;Lietuvos Rytas;UAB Martonas;5.55;

B456;Kauno Diena;KTU ZINIOS;2.69;

C789;Klaipėdos diena; UAB Uostas; 8.89;

**Atliksim Penkis testus:**

**Pirmas testas:**

Pirmam testui pasirinksime a-d užsakovų failus. Ir atliksime visus testus su mėnesiais su šiais failais:

A screenshot of a computer

Description automatically generated

Failus nuskaitė teisingai, dėl to atskleidė tolimesnius mygtukus

A screenshot of a computer

Description automatically generated

Pirmo mėnesio leidėjų pelnus atvaizduoja teisingai.

A screenshot of a computer

Description automatically generated

Pirmo mėnesio pajamas surikiavo teisingai

A screenshot of a computer

Description automatically generated

Pirmo mėnesio užsakovus atvaizdavo teisingai

A screenshot of a computer

Description automatically generated

Šešto mėnesio leidėjų uždarbius suskaičiavo teisingai

A screenshot of a computer

Description automatically generated

Šešto mėnesio pajamas surikiavo teisingai

A screenshot of a computer

Description automatically generated

Šešto mėnesio užsakovus atvaizdavo teisingai

Atlikau testus 1 ir 6 mėnesiui. Kaip matome kiekvienoje nuotraukoje, programa darbus atlieka taisyklingai. Todėl manau, kad šis testas pavyko.

**Antras testas:**

Antram testui pasirinksime e-h užsakovų failus ir atliksime kelis testus su mėnesiais su šiais failais:

A screenshot of a computer

Description automatically generated

Failus nuskaitė, todėl atvėrė likusius mygtukus.

A screenshot of a computer

Description automatically generated

Pirmo mėnesio uždarbius apskaičiavo teisingai

A screenshot of a computer

Description automatically generated

Pirmo mėnesio pajamas surikiavo teisingai

A screenshot of a computer

Description automatically generated

Pirmo mėnesio prenumeratorius atvaizdavo teisingai.

A screenshot of a computer

Description automatically generated

Šešto mėnesio pajamas apskaičiavo teisingai ir jų kelis kart kviečiant programą nepridėjinėja, todėl veikia tvarkingai

A screenshot of a computer

Description automatically generated

Leidėjų pajamas surikiavo teisingai.

A screenshot of a computer

Description automatically generated

Šešto mėnesio prenumeratorius atvaizdavo korektiškai.

Po šio testo galiu teigti, kad programa veikia tinkamai, jeigu naudojami teisingi failai.

**Trečias testas:**

Trečiam testui pasirinksime visus užsakovų failus ir atliksime kelis testus su mėnesiais su šiais failais:

A screenshot of a computer

Description automatically generated

Visus failus pasirinkus, juos nuskaito taisyklingai, dėl to atvėrė likusius mygtukus.

A screenshot of a computer

Description automatically generated

Pirmo mėnesio pajamas apskaičiavo teisingai.

A screenshot of a computer

Description automatically generated

Pirmo mėnesio pajamas surikiavo teisingai.

A screenshot of a computer

Description automatically generated

Pirmo mėnesio prenumeratorius atvaizdavo teisingai.

A screenshot of a computer

Description automatically generated

Šešto mėnesio pajamas apskaičiavo teisingai ir kelis kartus perskaičiavus, jų nesusumavo, todėl programa vykdo darbą teisingai.

A screenshot of a computer

Description automatically generated

Šešto mėnesio pajamas surikiavo teisingai.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Šešto mėnesio prenumeratorius atvaizdavo teisingai.

Pagal šių testų rezultatus, galiu teigti, kad programa veikia taisyklingai.

**Ketvirtas testas:**

Ketvirtam testui sukeisime vietomis failus, vietoj leidėjų pasirinksime užsakovų failus, vietoj užsakovų failų pasirinksiu leidėjų failą:

A screenshot of a computer error

Description automatically generated

Paspaudus skaityti, programa atlieka išimčių valdymą ir dėl to, išmeta klaidą kai bandome skaityti netinkamus failus:

A computer screen shot of a program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Penktas testas:**

Penktam testui nepasirinksiu failų:

Atveriame programą ir spaudžiama skaityti failus nepasirinkus failų:

A screenshot of a computer error

Description automatically generated

A screenshot of a computer error

Description automatically generated

Programa atvaizduoja, kad jokių failų nepasirinko.

## Dėstytojo pastabos

Testukas: 1/3