KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

OBJEKTINIS PROGRAMAVIMAS I (P175B118)

Darbų aplankas

Atliko:
IFF-8/11 gr. studentai
Arnas Švenčionis
Gytis Budinas
Titas Černiauskas
Reinholdas Kondratavičius
2018 m. gruodžio 20 d.
Priėmė:
Asist. Andrius Lauraitis

KAUNAS 2018

TURINYS

1. Objektų rinkinys

1.1. Darbo užduotis

U1-12. **Kompiuterinis žaidimas**. Kuriate „fantasy“ tipo kompiuterinį žaidimą. Duomenų faile turite informacija apie žaidimo herojus: vardas, rasė, klasė, gyvybės taškai, mana, žalos taškai, gynybos taškai, jėga, vikrumas, intelektas, ypatinga galia.

- Raskite, kuris herojus pasižymi geriausiomis charakteristikomis (jėgos, vikrumo ir intelekto suma didžiausia), ekrane atspausdinkite herojaus vardą, rasę, klasę bei charakteristikas.
- Raskite, kokios rasės herojų daugiausia. Ekrane atspausdinkite rasės pavadinimą, bei visų tos rasės herojų vardus.
- Sudarykite visų elfų sąrašą, faile „Elfai.csv“ įrašykite visus jų duomenis.
- Sudarykite herojų, kurių gyvybės taškai bent 100, o gynybos taškai – bent 30, sąrašą. Į failą „Tankai.csv“ įrašykite herojų vardus, klases, rases bei ypatingas galias.

1.2. Programos tekstas

Fantasy.cs

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

namespace _1_Laboratorinis
{
    class Fantasy
    {
        public string Vardas { get; set; }
        public string Rase { get; set; }
        public string Klase { get; set; }
        public int GyvybesTaskai { get; set; }
        public int Mana { get; set; }
        public int ZalosTaskai { get; set; }
        public int GynybosTaskai { get; set; }
        public int Jega { get; set; }
        public int Vikrumas { get; set; }
        public int Intelektas { get; set; }
        public string YpatingaGalia { get; set; }

        public int Skaitkliukas { get; set; } // tarnybinis kintamasis

        /// <summary>
        /// rasiu pasikartojimo kiekiui skaiciuoti
        /// </summary>
        /// <param name="rase">rases pavadinimas</param>
        public Fantasy(string rase)
        {
            Rase = rase;
            Skaitkliukas = 0;
        }
        /// <summary>
        /// Duomenu priskyrimas
        /// </summary>
        /// <param name="vardas">herojaus vardas</param>
        /// <param name="rase">herojaus rase</param>
        /// <param name="klase">herojaus klase</param>
        /// <param name="gyvybestaskai">herojaus gyvybes taskai</param>
        /// <param name="mana">herojaus mana</param>
        /// <param name="zalostaskai">herojaus zalos taskai</param>
        /// <param name="gynybostaskai">herojaus gynybos taskai</param>
        /// <param name="jega">herojaus jega</param>
        /// <param name="vikrumas">herojaus vikrumas</param>
        /// <param name="intelektas">herojaus intelektas</param>
```

```
        /// <param name="ypatingagalia">herojaus ypatinga galia</param>
        public Fantasy(string vardas, string rase, string klase, int gyvybestaskai,
            int mana, int zalostaskai, int gynybostaskai, int jega,
            int vikrumas, int intelektas, string ypatingagalia )
        {
            Vardas = vardas;
            Rase = rase;
            Klase = klase;
            GyvybesTaskai = gyvybestaskai;
            Mana = mana;
            ZalosTaskai = zalostaskai;
            GynybosTaskai = gynybostaskai;
            Jega = jega;
            Vikrumas = vikrumas;
            Intelektas = intelektas;
            YpatingaGalia = ypatingagalia;

            Skaitkliukas = 0;
        }

    }
}
```

Program.cs

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

namespace _1_Laboratorinis
{
    class Program
    {
        void Main2()
        {
            List<Fantasy> herojai = Nuskaitymas();

            DuomenuFailas(herojai);

            //Pirmas punktas
            int stipriausioSk = Stipriausias(herojai);
            StipriausiuSpausdinimas(herojai, stipriausioSk);

            //Antras punktas
            List<Fantasy> r = DazniausiosRasesv2(herojai);
            string dazniausiaRase = DazniausiaRaseTest(r);
            List<Fantasy> dazniausi = DazniausiuVarduAtrinkimas(herojai,
             dazniausiaRase);
            SpausdiniTest(dazniausiaRase, dazniausi);

            //Trecias punktas
            List<Fantasy> a = ElfuRadimas(herojai);
            ElfuSpausdinimas(a);

            //Ketvirtas punktas
            List<Fantasy> b = TankuRadimas(herojai);
            TankuSpausdinimas(b);

        }
        static void Main(string[] args)
        {
            Program p = new Program();
            p.Main2();
        }
        /// <summary>
        /// Nuskaito duomenis is failo,
        /// paskirsto eilutes i duomenis
        /// </summary>
        /// <returns>grazina sukurta list'a</returns>
        List<Fantasy> Nuskaitymas()
```

```
{
    List<Fantasy> herojai = new List<Fantasy>();
    string[] lines = File.ReadAllLines(@"Duomenys1.csv");
    foreach (string line in lines)
    {
        string[] values = line.Split(';');
        string vardas = values[0];
        string rase = values[1];
        string klase = values[2];
        int gyvybestaskai = int.Parse(values[3]);
        int mana = int.Parse(values[4]);
        int zalostaskai = int.Parse(values[5]);
        int gynybostaksai = int.Parse(values[6]);
        int jega = int.Parse(values[7]);
        int vikrumas = int.Parse(values[8]);
        int intelektas = int.Parse(values[9]);
        string ypatingagalia = values[10];
        Fantasy herojus = new Fantasy(vardas, rase, klase, gyvybestaskai, mana,
           zalostaskai, gynybostaksai, jega, vikrumas, intelektas,
           ypatingagalia);
        herojai.Add(herojus);
    }
    return herojai;
}
/// <summary>
/// Spausdina pradinius duomenis lentele
/// </summary>
/// <param name="herojai">heroju duomenu list</param>
public void DuomenuFailas(List<Fantasy> herojai)
{
        int j = 1;
        string[] lines = new string[herojai.Count + 1];
        lines[0] = String.Format("|{0, -15}|{1, -7}|{2, -10}|{3, -15}|{4, -
           5}|{5, -13}|{6, -15}|{7, -5}|{8, -9}|{9, -11}|{10, -24}|", "Vardas",
           "Rase", "Klase", "Gyvybes Taskai", "Mana", "Zalos Taskai", "Gynybos
           Taskai", "Jega", "Vikrumas", "Intelektas", "Ypatinga Galia");
        for (int i = 0; i < herojai.Count; i++)
        {

            lines[j] = String.Format("|{0, -15}|{1, -7}|{2, -10}|{3, -15}|{4, -
                 5}|{5, -13}|{6, -15}|{7, -5}|{8, -9}|{9, -11}|{10, -24}|",
                 herojai[i].Vardas, herojai[i].Rase, herojai[i].Klase,
                 herojai[i].GyvybesTaskai, herojai[i].Mana,
                 herojai[i].ZalosTaskai, herojai[i].GynybosTaskai,
                 herojai[i].Jega, herojai[i].Vikrumas, herojai[i].Intelektas,
                 herojai[i].YpatingaGalia);
            j++;
        }
        File.WriteAllLines(@"Duomenys.txt", lines);
}

/// <summary>
/// Stipriausio herojaus skaiciaus List'e radimas
/// </summary>
/// <param name="herojai">heroju duomenu list</param>
/// <returns> Grazina stipriausio herojaus skaiciu list'e </returns>
int Stipriausias(List<Fantasy> herojai)
{
    int stiprumas = herojai[0].Jega + herojai[0].Vikrumas +
     herojai[0].Intelektas;
    int stipriausioSk = 0;
    int max = stiprumas;

    for (int i = 0; i < herojai.Count; i++)
    {
        stiprumas = herojai[i].Jega + herojai[i].Vikrumas +
           herojai[i].Intelektas;
        if (stiprumas > max)
        {
            max = stiprumas;
```

```
                stipriausioSk = i; // suzinomas stipriausio herojaus skaicius
                     List'e
            }
        }
        return stipriausioSk;
    }
    /// <summary>
    /// Consolėje spausdinamas stipriausias herojus pagal zinoma jo skaiciu list'e
    /// </summary>
    /// <param name="herojai">heroju duomenu list</param>
    /// <param name="stipriausioSk">stipriausio herojaus indeksas</param>
    public void StipriausiuSpausdinimas(List<Fantasy> herojai, int stipriausioSk)
    {
        Console.WriteLine("Stipriausias herojus yra: ");
        Console.WriteLine("-------------------------------------------------------
         ------------------------------------------------------------------------
         -----------");
        Console.WriteLine("|{0, -15}|{1, -7}|{2, -10}|{3, -15}|{4, -5}|{5, -13}|{6,
         -15}|{7, -5}|{8, -9}|{9, -11}|{10, -24}|", "Vardas", "Rase", "Klase",
         "Gyvybes Taskai", "Mana", "Zalos Taskai", "Gynybos Taskai", "Jega",
         "Vikrumas", "Intelektas", "Ypatinga Galia");
        Console.WriteLine("-------------------------------------------------------
         ------------------------------------------------------------------------
         -----------");
        Console.WriteLine("|{0, -15}|{1, -7}|{2, -10}|{3, -15}|{4, -5}|{5, -13}|{6,
         -15}|{7, -5}|{8, -9}|{9, -11}|{10, -24}|", herojai[stipriausioSk].Vardas,
         herojai[stipriausioSk].Rase, herojai[stipriausioSk].Klase,
         herojai[stipriausioSk].GyvybesTaskai, herojai[stipriausioSk].Mana,
         herojai[stipriausioSk].ZalosTaskai, herojai[stipriausioSk].GynybosTaskai,
         herojai[stipriausioSk].Jega, herojai[stipriausioSk].Vikrumas,
         herojai[stipriausioSk].Intelektas, herojai[stipriausioSk].YpatingaGalia);
        Console.WriteLine("-------------------------------------------------------
         ------------------------------------------------------------------------
         -----------");
        Console.WriteLine("");
    }

    /// <summary>
    /// Sukuriamas naujas list'as, kuriame yra suzinoma kiek kiiekviena rasė turi
         herojų
    /// </summary>
    /// <param name="herojai">heroju duomenu list</param>
    /// <returns>Gražina kiekvienos rasės herojų skaičių ir tos rasės pavadinimą
         </returns>
    public List<Fantasy> DazniausiosRasesv2(List<Fantasy> herojai)
    {
        List<Fantasy> unikalus_herojai = new List<Fantasy>();
        Fantasy rastas_h; // rastas herojus sarase


        for (int i = 0; i < herojai.Count; i++)
        {
            rastas_h = null;
            for (int j = 0; j < unikalus_herojai.Count; ++j) // ieskome sarase
               unikalus_herojai ar toks jau herojus yra
            {
                if (unikalus_herojai[j].Rase.CompareTo(herojai[i].Rase) == 0)
                {
                    rastas_h = unikalus_herojai[j];
                    break;
                }

            }
                if (rastas_h == null) // tokios rases nera, pridedame elementą į
                     sarašą
                {
                    rastas_h = new Fantasy(herojai[i].Rase);
                    unikalus_herojai.Add(rastas_h);
                }
            rastas_h.Skaitkliukas += 1;
```

```
        }
        return unikalus_herojai;
    }
    /// <summary>
    /// Tikriname kurios rasės herojų daugiausia
    /// </summary>
    /// <param name="unikalus_herojai">rasiu ir ju pasikartojimo list</param>
    /// <returns>Gražinamas dažniausios rasės pavadinimas</returns>
    public string DazniausiaRaseTest(List<Fantasy> unikalus_herojai)
    {

        string dazniausia_r = unikalus_herojai[0].Rase; //Tikriname kurios rasės
         herojų yra daugiausia
        int max = unikalus_herojai[0].Skaitkliukas;
        for (int u = 0; u < unikalus_herojai.Count; u++)
        {
            if (unikalus_herojai[u].Skaitkliukas > max)
            {
                dazniausia_r = unikalus_herojai[u].Rase;
                max = unikalus_herojai[u].Skaitkliukas;
            }
        }
        return dazniausia_r;
    }
    /// <summary>
    /// Į naują list'ą įdedami visi dažniausios rasės herojų vardai
    /// </summary>
    /// <param name="herojai">visu heroju list</param>
    /// <param name="dazniausia_r">dazniausios rases pavadinimas</param>
    /// <returns>Gražinamas vardų list'as</returns>
    public List<Fantasy> DazniausiuVarduAtrinkimas(List<Fantasy> herojai, string
         dazniausia_r)
    {
        List<Fantasy> vardai = new List<Fantasy>();
        for (int i = 0; i < herojai.Count; i++)
        {
            if (herojai[i].Rase == dazniausia_r) //Jei herojaus rasė sutampa su
               dažniausia rase, jo vardas pridedamas į vardų list'ą
            {
                vardai.Add(herojai[i]);
            }
        }
        return vardai;
    }
    /// <summary>
    /// Spausdinamas dažniausios rasės pavadinimas ir visi tos rasės herojų vardai
    /// </summary>
    /// <param name="dazniausiaRase">dazniausios rases pavadinimas</param>
    /// <param name="dazniausiuVardai">dazniauiu rasiu list</param>
    public void SpausdiniTest(string dazniausiaRase, List<Fantasy>
         dazniausiuVardai)
    {
        Console.WriteLine("Dazniausia rase: " + dazniausiaRase);
        Console.WriteLine("---------------------------------");
        Console.WriteLine("|Dazniausios rases heroju vardai|");
        Console.WriteLine("---------------------------------");
        foreach (var herojus in dazniausiuVardai)
        {
            Console.WriteLine("|{0, -31}|", herojus.Vardas);
            Console.WriteLine("---------------------------------");
        }
    }

    /// <summary>
    /// Einama per herojų sarašą ir ieškoma elfų. Elfai įrašomi į naują list'ą
    /// </summary>
    /// <param name="herojai">heroju duomenu list</param>
    /// <returns>Gražinamas sukurtas elfų list'as</returns>
    public List<Fantasy> ElfuRadimas(List<Fantasy> herojai)
```

```
{
    List<Fantasy> Elfai = new List<Fantasy>();


    for (int i = 0; i < herojai.Count; i++) // einama per visą herojų list'ą
    {
        if (herojai[i].Rase == "Elfas") // jei herojus yra elfas jis įrašomas į
           elfų sąrašą
        {
            Elfai.Add(herojai[i]);
        }
    }
    return Elfai;
}
/// <summary>
/// Į failą "Elfai.csv" įrašomi elfai iš elfų list'o
/// </summary>
/// <param name="Elfai">tik elfu listas</param>
public void ElfuSpausdinimas(List<Fantasy> Elfai)
{
    int j = 1;
    string[] lines = new string[Elfai.Count + 1];
    lines[0] = String.Format("Vardas; Rase; Klase; Gyvybes Taskai; Mana; Zalos
     Taskai; Gynybos Taskai; Jega; Vikrumas; Intelektas; Ypatinga Galia");
    for (int i = 0; i < Elfai.Count; i++)
    {
        lines[j] =
           String.Format("{0};{1};{2};{3};{4};{5};{6};{7};{8};{9};{10}",
           Elfai[i].Vardas, Elfai[i].Rase, Elfai[i].Klase,
           Elfai[i].GyvybesTaskai, Elfai[i].Mana, Elfai[i].ZalosTaskai,
           Elfai[i].GynybosTaskai, Elfai[i].Jega, Elfai[i].Vikrumas,
           Elfai[i].Intelektas, Elfai[i].YpatingaGalia);
        j++;
    }
    File.WriteAllLines(@"Elfai.csv", lines);
}

/// <summary>
/// Einama per herojų sarašą ir tikrina kiekvieno herojaus charakteristikas.
/// Jei jos atitinka tanko charakteristikas, herojus pridedamas į tankų sarašą
/// </summary>
/// <param name="herojai">heroju duomenu listas</param>
/// <returns>Gražina tankų list'ą</returns>
public List<Fantasy> TankuRadimas(List<Fantasy> herojai)
{
    List<Fantasy> Tankai = new List<Fantasy>();


    for (int i = 0; i < herojai.Count; i++) //Einama per visą herojų list'ą
    {
        if ((herojai[i].GyvybesTaskai >= 100)&&(herojai[i].GynybosTaskai >=
           30)) //jei herojaus charakteristikos atitinka tanko apibūdinimą,
        {
            Tankai.Add(herojai[i]); //herojus įrašomas į tankų listą
        }
    }
    return Tankai;
}
/// <summary>
/// Į failą "Tankai.csv" įrašomi tankai iš tankų list'o
/// </summary>
/// <param name="Tankai">tanku duomenu listas</param>
public void TankuSpausdinimas(List<Fantasy> Tankai)
{
    int j = 1;
    string[] lines = new string[Tankai.Count + 1];
    lines[0] = String.Format("Vardas; Klase; Rase; Ypatinga Galia");
    for (int i = 0; i < Tankai.Count; i++)
    {
```

```
                lines[j] = String.Format("{0};{1};{2};{3}", Tankai[i].Vardas,                          Tankai[i].Klase,
Tankai[i].Rase,  Tankai[i].YpatingaGalia);
                j++;
            }
            File.WriteAllLines(@"Tankai.csv", lines);
        }

    }
}
```

1.3. Pradiniai duomenys ir rezultatai

Pirmas bandymas:

Pradiniai duomenys:

Vardas	Rase	Klase	Gyvybes Taskai	Mana	Zalos Taskai	Gynybos Taskai	Jega	Vikrumas	Intelektas	Ypatinga Galia
Garen	Zmogus	Tank	600	0	30	150	40	20	20	Justice
Ziggs	Yordle	Mage	200	300	100	20	60	20	40	Bomb
Miss Fortune	Zmogus	Marksman	200	150	80	20	70	30	30	Guns Blazing
Jinx	Elfas	Marksman	200	160	75	25	75	30	35	Super Mega Death Rocket
Brand	Elfas	Mage	200	300	100	20	60	20	40	Cataclysm
Lucian	Zmogus	Marksman	200	160	75	27	75	30	35	The Culling
Tristana	Yordle	Marksman	270	170	80	25	60	25	15	Blast
Rumble	Yordle	Fighter	400	50	50	29	30	20	25	Volcano
Irelia	Zmogus	Tank	430	70	60	35	25	15	20	Divine

Rezultatai Consolėje:

```
Stipriausias herojus yra:
-----------------------------------------------------------------------------------------------------------------------------------
|Vardas         |Rase   |Klase     |Gyvybes Taskai |Mana |Zalos Taskai |Gynybos Taskai |Jega |Vikrumas |Intelektas |Ypatinga Galia          |
-----------------------------------------------------------------------------------------------------------------------------------
|Jinx           |Elfas  |Marksman  |300            |160  |75           |30             |75   |30       |35         |Super Mega Death Rocket |
-----------------------------------------------------------------------------------------------------------------------------------

Dazniausia rase: Zmogus
---------------------------------
```

Dazniausios rases heroju vardai
Garen
Miss Fortune
Lucian
Irelia

Elfai.csv

Vardas	Rase	Klase	Gyvybes Taskai	Mana	Zalos Taskai	Gynybos Taskai	Jega	Vikrumas	Intelektas	Ypatinga Galia
Jinx	Elfas	Marksman	300	160	75	30	75	30	35	Super Mega Death Rocket
Brand	Elfas	Mage	200	300	100	20	60	20	40	Cataclysm

Tankai.csv

Vardas	Klase	Rase	Ypatinga Galia
Garen	Tank	Zmogus	Justice
Irelia	Tank	Zmogus	Divine

Antras bandymas:
Pradiniai duomenys lentele:

Vardas	Rase	Klase	Gyvybes Taskai	Mana	Zalos Taskai	Gynybos Taskai	Jega	Vikrumas	Intelektas	Ypatinga Galia
Garen	Zmogus	Tank	600	0	30	150	40	20	20	Justice
Ziggs	Yordle	Mage	200	300	100	20	60	20	40	Bomb
Miss Fortune	Zmogus	Marksman	200	150	80	20	70	30	30	Guns Blazing
Jinx	Zmogus	Marksman	200	160	75	25	75	30	35	Super Mega Death Rocket
Brand	Demonas	Mage	200	300	100	20	60	20	40	Cataclysm
Lucian	Zmogus	Marksman	200	160	75	27	75	30	35	The Culling
Tristana	Yordle	Marksman	270	170	80	25	60	25	15	Blast
Rumble	Yordle	Fighter	400	50	50	29	30	20	25	Volcano
Irelia	Zmogus	Tank	430	70	60	35	25	15	20	Divine

Konsole:

```
Stipriausias herojus yra:
-------------------------------------------------------------------------------------------------------------------------------
---------------
|Vardas         |Rase   |Klase     |Gyvybes Taskai |Mana |Zalos Taskai |Gynybos Taskai |Jega |Vikrumas |Intelektas |Ypatinga
Galia          |
-------------------------------------------------------------------------------------------------------------------------------
---------------
|Jinx           |Zmogus |Marksman  |200            |160  |75           |25             |75   |30       |35         |Super Mega
Death Rocket |
-------------------------------------------------------------------------------------------------------------------------------
---------------

Dazniausia rase: Zmogus
---------------------------------
|Dazniausios rases heroju vardai|
---------------------------------
|Garen                          |
---------------------------------
|Miss Fortune                   |
---------------------------------
|Jinx                           |
---------------------------------
|Lucian                         |
---------------------------------
|Irelia                         |
---------------------------------
Press any key to continue . . .
```

Elfai.csv

Elfu Nera

Tankai.csv

Vardas; Klase; Rase;
Ypatinga Galia
Garen;Tank;Zmogus;Justice
Irelia;Tank;Zmogus;Divine

Trečias bandymas:
Duomenys lentele:

Vardas	Rase	Klase	Gyvybes Taskai	Mana	Zalos Taskai	Gynybos Taskai	Jega	Vikrumas	Intelektas	Ypatinga Galia
Garen	Zmogus	Tank	10	0	30	20	40	20	20	Justice
Ziggs	Elfas	Mage	200	300	100	20	60	20	40	Bomb
Miss Fortune	Zmogus	Marksman	200	150	80	20	70	30	30	Guns Blazing
Jinx	Zmogus	Marksman	200	160	75	25	75	30	35	Super Mega Death Rocket
Brand	Demonas	Mage	200	300	100	20	60	20	40	Cataclysm
Lucian	Zmogus	Marksman	200	160	75	27	75	30	35	The Culling
Tristana	Yordle	Marksman	270	170	80	25	60	25	15	Blast
Rumble	Yordle	Fighter	400	50	50	29	30	20	25	Volcano
Irelia	Zmogus	Tank	90	70	60	14	25	15	20	Divine
Destiny	Zmogus	Marksman	200	170	80	25	80	35	40	Escape

Konsole:

```
Stipriausias herojus yra:
```

Vardas	Rase	Klase	Gyvybes Taskai	Mana	Zalos Taskai	Gynybos Taskai	Jega	Vikrumas	Intelektas	Ypatinga Galia
Destiny	Zmogus	Marksman	200	170	80	25	80	35	40	Escape

```
Dazniausia rase: Zmogus
```

Dazniausios rases heroju vardai
Garen
Miss Fortune
Jinx
Lucian
Irelia
Destiny

Elfai.csv

Vardas; Rase; Klase; Gyvybes Taskai; Mana; Zalos Taskai; Gynybos Taskai; Jega; Vikrumas; Intelektas; Ypatinga Galia
Ziggs;Elfas;Mage;200;300;100;20;60;20;40;Bomb

Tankai.csv

Tanku nera

1.4. Dėstytojo pastabos

- P5
- P6
- P11
- P8

2. Konteineris

2.1. Darbo užduotis

U2-9. **IMBD**. Turite skirtingų kinomanų mėgėjų peržiūrėtus filmų sąrašus. Keičiasi duomenų formatas. Pirmoje eilutėje kino mėgėjo vardas pavardė, antroje - gimimo metai, trečioje - miestas. Toliau informacija apie filmus pateikta tokiu pačiu formatu kaip L1 užduotyje.

- Raskite ir atspausdinkite ekrane kiekvieno kino mėgėjo mėgstamiausią režisierių.
- Sudarykite filmų, kuriuos peržiūrėjo visi kino mėgėjai, sąrašą. Visus duomenis apie filmus įrašykite į failą „MatėVisi.csv“.
- Kiekvienam kino mėgėjui sudarykite rekomenduojamų peržiūrėti filmų sąrašą, į kurį įtraukite filmus, kurių jis nematė, tačiau matė kiti kino mėgėjai. Rekomendacijų sąrašus įrašykite į failus „Rekomendacija_vardas_pavardė.csv“.

2.2. Programos tekstas

Branch.cs

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

namespace U1_9
{
    class Branch
    {
        public const int MaxFilmuSk = 500;

        public string Vardas { get; set; }
        public FilmuKonteineris Filmai { get; private set; }

        /// <summary>
        /// sukuriamas naujas branch
        /// ziurovo informacija
        /// jo matyti filmai
        /// </summary>
        /// <param name="vardas">ziurovo vardas</param>
        public Branch (string vardas)
        {
            Vardas = vardas;
            Filmai = new FilmuKonteineris(MaxFilmuSk);

        }

        /// <summary>
        /// uzklojimas tostring
        /// leidzia formatuotai spausdinti informacija
        /// </summary>
        /// <returns>suformatuota informacija</returns>
        public override string ToString()
        {
            var builder = new StringBuilder();
            builder.AppendLine(Filmai.ToString());
            builder.AppendLine("Ziurovas: " + Vardas);
            builder.AppendLine("");
            return builder.ToString();
            //return String.Format("Pav: {0}, Metai: {1}, Zanras: {2}, Studija {3},
                  Rezisierius: {4} {5}, Aktoriai: {6}, {7}, Pajamos: {8}", Filmas,
Metai, Zanras,           Studija, RezisieriausVardas, RezisieriausPavarde, Aktorius1,
Aktorius2, Pajamos);
```

```
        }
    }
}
```

BranchuKonteineris.cs

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

namespace U1_9
{
    class BranchuKonteineris
    {
        private Branch[] branches;
        public int Count { get; private set; }

        /// <summary>
        /// sukuria nauja branchu konteineri
        /// </summary>
        /// <param name="size">konteinerio dydis</param>
        public BranchuKonteineris(int size)
        {
            branches = new Branch[size];
            Count = 0;
        }
        /// <summary>
        /// prideda paduota branch i konteineri
        /// </summary>
        /// <param name="branch">paduota informacija apie
        /// ziurova</param>
        public void PridetiBrancha(Branch branch)
        {
            branches[Count++] = branch;
        }
        /// <summary>
        /// prideda branch i nurodyta vieta konteineryje
        /// </summary>
        /// <param name="branch"></param>
        /// <param name="index"></param>
        public void PridetiBrancha(Branch branch, int index)
        {
            branches[index] = branch;
        }
        /// <summary>
        /// gauna branch is konteinerio
        /// nurodant jo vieta
        /// </summary>
        /// <param name="index">vieta konteineryje</param>
        /// <returns>branch informacija</returns>
        public Branch GautiBrancha(int index)
        {
            return branches[index];
        }

    }
}
```

FilmuKonteineris.cs

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

namespace U1_9
{
    class FilmuKonteineris
```

```
{
    private Sąrašas[] filmai;
    public int Count { get; private set; }

    /// <summary>
    /// sukuriamas naujas filmu konteineris
    /// </summary>
    /// <param name="size">filmu konteinerio dydis</param>
    public FilmuKonteineris (int size)
    {
        filmai = new Sąrašas[size];
        Count = 0;
    }
    /// <summary>
    /// prideti filma i konteineri
    /// </summary>
    /// <param name="filmas">filmo duomenys</param>
    public void PridetiFilma (Sąrašas filmas)
    {
        filmai[Count++] = filmas;
    }
    /// <summary>
    /// prideti filma i nustatyta vieta
    /// </summary>
    /// <param name="filmas">filmo duomenys</param>
    /// <param name="index">vieta konteineryje</param>
    public void PridetiFilma (Sąrašas filmas, int index)
    {
        filmai[index] = filmas;
    }
    /// <summary>
    /// randa filma konteyneryje
    /// padavus jo vieta
    /// </summary>
    /// <param name="index">filmo vieta konteineryje</param>
    /// <returns></returns>
    public Sąrašas RastiFilma (int index)
    {
        return filmai[index];
    }

    /// <summary>
    /// ToString uzklojimas,
    /// leidziantis formatuotai spausdinti
    /// filmu duomenis
    /// </summary>
    /// <returns></returns>
    public override string ToString()
    {
        var builder = new StringBuilder();
        foreach(var filmas in filmai)
        {

            if (filmas == null)
                break;
            builder.AppendLine(filmas.ToString());
        }
        return builder.ToString();

    }

    /// <summary>
    /// contains uzklojimas
    /// leidzia ziureti ar konteineris turi
    /// pauota filma
    /// </summary>
    /// <param name="filmas">paduotas filmas tikrinamas</param>
    /// <returns>grazina ar filmas yra komnteineryje</returns>
    public bool Contains(Sąrašas filmas)
    {
```

```
            return filmai.Contains(filmas);
        }

    }
}
```

Sąrašas.cs

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

namespace U1_9
{
    class Sąrašas
    {
        public string Filmas { get; set; }
        public int Metai { get; set; }
        public string Zanras { get; set; }
        public string Studija { get; set; }
        public string RezisieriausVardas { get; set; }
        public string RezisieriausPavarde { get; set; }
        public string Aktorius1 { get; set; }
        public string Aktorius2 { get; set; }
        public double Pajamos { get; set; }

        /// <summary>
        /// ziurovo ziuretu filmu informacija
        /// </summary>
        /// <param name="filmas">pavadinimas</param>
        /// <param name="metai">isleidimo metai</param>
        /// <param name="zanras">zanras</param>
        /// <param name="studija">studija</param>
        /// <param name="rezisieriausvardas">rezisieriaus vardas</param>
        /// <param name="rezisieriauspavarde">rezisieriaus pavarde</param>
        /// <param name="aktorius1">vienas aktorius filme</param>
        /// <param name="aktorius2">antras aktorius filme</param>
        /// <param name="pajamos">filmo pajamos</param>
        public Sąrašas(string filmas, int metai, string zanras, string studija,
               string rezisieriausvardas, string rezisieriauspavarde, string aktorius1,
               string aktorius2, double pajamos)
         {
            Filmas = filmas;
            Metai = metai;
            Zanras = zanras;
            Studija = studija;
            RezisieriausVardas = rezisieriausvardas;
            RezisieriausPavarde = rezisieriauspavarde;
            Aktorius1 = aktorius1;
            Aktorius2 = aktorius2;
            Pajamos = pajamos;
         }

        /// <summary>
        /// to string uzklojimas
        /// </summary>
        /// <returns>suformatuotai atspausdina filmuo duomenis</returns>
        public override string ToString()
        {
            return String.Format("{0, 12}|, {1, 6}|, {2, 12}|," +
                " {3, 16}|, {4, 20}|, {5, 21}|, {6, 12}|, {7, 13}|," +
                " {8, 8}|", Filmas, Metai, Zanras, Studija, RezisieriausVardas,
                RezisieriausPavarde, Aktorius1, Aktorius2, Pajamos);
        }

        /// <summary>
        /// equals uzklojimas
```

```
        /// lygina filmus
        /// </summary>
        /// <param name="obj">lyginamas objektas</param>
        /// <returns></returns>
        public override bool Equals(object obj)
        {
            return this.Equals(obj as Sąrašas);
        }

        public bool Equals (Sąrašas filmass)
        {
            //tikrina, ar objektas egzistuoja
            if(Object.ReferenceEquals(filmass, null))
            {
                return false;
            }
            //Tikrina, ar tokia pati klase
            if (this.GetType() != filmass.GetType())
                return false;
            //graziname true, jei objektu savybes sutampa
                return (Filmas == filmass.Filmas);
        }

        /// <summary>
        /// randa filmo hashcode
        /// </summary>
        /// <returns>filmo hashcode</returns>
        public override int GetHashCode()
        {
            return Filmas.GetHashCode();
        }

        /// <summary>
        /// uzlojimas, leidziantis lyginti filmus
        /// </summary>
        /// <param name="lhs"></param>
        /// <param name="rhs"></param>
        /// <returns></returns>
        public static bool operator == (Sąrašas lhs, Sąrašas rhs)
        {
            //Patikriname kaire puse (ar egzistuoja objektas)
            //negalima naudoti lhs==null.
            if(Object.ReferenceEquals(lhs, null))
            {
                if(Object.ReferenceEquals(rhs, null))
                {
                    //jei objektas neeigzistuoja nei kaireje puseje, nei desineje
puseje
                    //palyginimo operatoriaus puseje, graziname true (null == null =
true)
                    return true;
                }
                return false;// jei objetas neegzistuoja tik kaireje puseje
            }
            return lhs.Equals(rhs);
        }
        public static bool operator != (Sąrašas lhs, Sąrašas rhs)
        {
            return !(lhs == rhs);
        }

    }

}
```

Program.cs

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

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

namespace U1_9
{
    class Program
    {
        /// <summary>
        /// Duomenų nuskaitymas ir sudėjimas į sąrašą
        /// </summary>
        /// <returns>Struktūrinį sąrašą</returns>
        List<Sąrašas> Skaitymas()
        {
            List<Sąrašas> sąrašas = new List<Sąrašas>();
            string[] eilutes = File.ReadAllLines(@"Duota.csv");
            foreach (string eilute in eilutes)
            {
                string[] duomuo = eilute.Split(';');
                string filmas = duomuo[0];
                int metai = int.Parse(duomuo[1]);
                string zanras = duomuo[2];
                string studija = duomuo[3];
                string rezisieriausvardas = duomuo[4];
                string rezisieriauspavarde = duomuo[5];
                string aktorius1 = duomuo[6];
                string aktorius2 = duomuo[7];
                double pajamos = Convert.ToDouble(duomuo[8]);
                //Sukuriamas šalutinis sąrašas, kuriame pridedame nuskaitytus duomenis
                Sąrašas surašymas = new Sąrašas(filmas, metai, zanras, studija,
                    rezisieriausvardas, rezisieriauspavarde, aktorius1, aktorius2,
pajamos);
                //Sąrašą išsaugojame ir įdedame į struktūrinį sąrašą
                sąrašas.Add(surašymas);
            }
            return sąrašas;
        }
        /// <summary>
        /// Ieškome didžiausių pajamų
        /// </summary>
        /// <param name="sąrašas">Visų filmų sąrašas</param>
        /// <returns>Pelningiausio filmo suma</returns>
        double Pelningas(List<Sąrašas> sąrašas)
        {
            double pelningas = 0;
            foreach (Sąrašas sąraš in sąrašas)
            {
                if (sąraš.Pajamos > pelningas && sąraš.Metai == 2014)
                {
                    pelningas = sąraš.Pajamos;
                }
            }
            return pelningas;
        }
        /// <summary>
        /// Suradus didžiausias pajamas, išsaugojame filmus, kurios yra pelningiausios
        /// </summary>
        /// <param name="sąrašas">Visų filmų sąrašas</param>
        /// <param name="Pelningas">Pelningiausio filmo suma</param>
        /// <returns>Pelningiausius filmus</returns>
        List<Sąrašas> Pelningiausias(List<Sąrašas> sąrašas, double Pelningas)
        {
            List<Sąrašas> Pelningasis = new List<Sąrašas>();
            foreach (Sąrašas sąraš in sąrašas)
            {
                if (Pelningas == sąraš.Pajamos)
                {
                    Pelningasis.Add(sąraš);
                }
```

```
            }
            return Pelningasis;
        }
        /// <summary>
        /// Surandame filmus, kuriuose vaidino Nicolas Cage
        /// </summary>
        /// <param name="sąrašas">Visų filmų sąrašas</param>
        /// <returns>Filmai kuriuose vaidino Nicolas Cage</returns>
        List<Sąrašas> Cage(List<Sąrašas> sąrašas)
        {
            List<Sąrašas> Nicolas = new List<Sąrašas>();
            foreach (Sąrašas sąraš in sąrašas)
            {
                if (sąraš.Aktorius1 == "N. Cage" || sąraš.Aktorius2 == "N. Cage")
                {
                    Nicolas.Add(sąraš);
                }
            }
            return Nicolas;
        }
        /// <summary>
        /// Randame visus žanrus
        /// </summary>
        /// <param name="sąrašas">Visų filmų sąrašas</param>
        /// <returns>Visi filmų žanrai iš sąrašo</returns>
        List<Sąrašas> Zanrai(List<Sąrašas> sąrašas)
        {
            List<Sąrašas> zanrai = new List<Sąrašas>();
            foreach (Sąrašas saras in sąrašas)
            {
                bool Nera = true;
                foreach (Sąrašas zan in zanrai)
                {
                    if (saras.Zanras == zan.Zanras)
                    {
                        Nera = false;
                        break;
                    }
                }
                if (Nera)
                {
                    zanrai.Add(saras);
                }
            }
            return zanrai;
        }
        /// <summary>
        /// Rašome į koncolę pelningiausius filmus
        /// </summary>
        /// <param name="Pelningiausias">Pelningiausi filmai</param>
        /// <param name="Zanrai">Visi filmų žanrai iš sąrašo</param>
        void Rasymas(List<Sąrašas> Pelningiausias, List<Sąrašas> Zanrai)
        {
            Console.WriteLine("Pelningiausi(-as) filma(-i)(-as)");
            foreach (Sąrašas Peln in Pelningiausias)
            {
                Console.WriteLine(Peln.Filmas, "; Režisierius: ",
Peln.RezisieriausVardas, " ",
                    Peln.RezisieriausPavarde, "; Pajamos: ", Peln.Pajamos);
            }
        }
        /// <summary>
        /// Surandame daugiausiai filmų pastačius(-į)(ius) režisieriu(-s)
        /// </summary>
        /// <param name="vardai">Režisierių vardai</param>
        /// <param name="pavardes">Režisierių pavardes</param>
        /// <param name="Kiek">Kiek filmų surežisuota</param>
        /// <param name="sąrašas">Visų filmų sąrašas</param>
        void DaugSkaiciavimai(string[] vardai, string[] pavardes, int[] Kiek,
List<Sąrašas> sąrašas)
```

```
        {
            int kiekis = 0;
            foreach (Sąrašas sąraš in sąrašas)
            {
                bool yra = true;
                for (int i = 0; i < kiekis; i++)
                {
                    if (vardai[i] == sąraš.RezisieriausVardas && pavardes[i] ==
sąraš.RezisieriausPavarde)
                    {
                        yra = false;
                        Kiek[i]++;
                        break;
                    }
                }
                if (yra)
                {
                    vardai[kiekis] = sąraš.RezisieriausVardas;
                    pavardes[kiekis] = sąraš.RezisieriausPavarde;
                    Kiek[kiekis] = 1;
                    kiekis++;
                }
            }
        }
        /// <summary>
        /// Surašom visus žanrus į failą
        /// </summary>
        /// <param name="Zanrai">Visi filmų žanrai iš sąrašo</param>
        void ZanrasRasymas(List<Sąrašas> Zanrai)
        {
            using (StreamWriter sw = new StreamWriter("Žanrai.csv", false))
            {
                foreach (Sąrašas zan in Zanrai)
                {
                    sw.WriteLine(zan.Zanras);
                }
            }
        }
        /// <summary>
        /// Surašom visus filmus kuriuose vaidino Nicolas Cage
        /// </summary>
        /// <param name="Nicolas">Nicolo Cage vaidinti filmai</param>
        void CageRasymas(List<Sąrašas> Nicolas)
        {
            using (StreamWriter sw = new StreamWriter("Cage.csv", false))
            {
                foreach (Sąrašas nic in Nicolas)
                {
                    sw.WriteLine(nic.Filmas + "; " + nic.Metai + "; " + nic.Studija +
"; ");
                }
            }
        }
        /// <summary>
        /// Radimas kelintas režisierius daugiausiai surežisavo filmų
        /// </summary>
        /// <param name="vardai">Režisierių vardai</param>
        /// <param name="pavardes">Režisierių pavardė</param>
        /// <param name="Kiek">Kiek surežisuota filmų</param>
        /// <returns>Indeksą</returns>
        int Indeksas(string[] vardai, string[] pavardes, int[] Kiek)
        {

            int ind = 0;
            for (int i = 1; i < Kiek.Length; i++)
            {
                if (Kiek[i] > Kiek[ind])
                {
                    ind = i;
                }
```

```
            }
            return ind;
        }
        /// <summary>
        /// Į koncolę spausdinamas režisierius kuris surežisavo daugiausiai filmų
        /// </summary>
        /// <param name="vardai">Režisieriaus vardas</param>
        /// <param name="pavardes">Režisieriaus pavardė</param>
        /// <param name="Kiek">Kiek filmų surežisuota</param>
        void Spausdinimas(string vardai, string pavardes, int Kiek)
        {
            Console.WriteLine("Daugiausiai filmų pastatęs:");
            Console.WriteLine("");
            Console.WriteLine("Režisieriaus vardas: " + vardai);
            Console.WriteLine("Režisieriaus pavardė: " + pavardes);
            Console.WriteLine("Kiek filmų surežisuota: " + Kiek);
            Console.WriteLine("");
            Console.WriteLine("");
        }
        /// <summary>
        /// Metodas skirtas spausdinti pradinius duomenis lentelėje
        /// </summary>
        /// <param name="sąrašas">Visų filmų sąrašas</param>
        void SpausdinamPradiniusDuomenis(List<Sąrašas> sąrašas)
        {
            using (StreamWriter sw = new StreamWriter("duomenys.txt"))
            {
                sw.WriteLine("----------------------------------------" +
                    "----------------------------------------" +
                    "----------------------------------------" +
                    "----------------------------------------" +
                    "------------------");
                sw.WriteLine("{0, -10} | {1, -8} | {2, -15} | {3, -15}" +
                    " | {4, -21} | {5, -19} | {6, -15} | {7, -15} | {8, -15}",
                    "Filmas", "Metai", "Žanras",
                    "Studija", "Režisieriaus Vardas", "Režisierius Pavardė",
                    "1 Aktorius", "2 Aktorius", "Pajamos");
                sw.WriteLine("----------------------------------------" +
                    "----------------------------------------" +
                    "----------------------------------------" +
                    "----------------------------------------" +
                    "------------------");

                for (int i = 0; i < sąrašas.Count; i++)
                {
                    sw.WriteLine("{0, -10} | {1, -8} | {2, -15} | {3, -15}" +
                        " | {4, -21} | {5, -19} | {6, -15} | {7, -15} | {8, -25}",
                        sąrašas[i].Filmas, sąrašas[i].Metai, sąrašas[i].Zanras,
                    sąrašas[i].Studija, sąrašas[i].RezisieriausVardas,
                    sąrašas[i].RezisieriausPavarde, sąrašas[i].Aktorius1,
                    sąrašas[i].Aktorius2, sąrašas[i].Pajamos);
                }
                sw.WriteLine("----------------------------------------" +
                    "----------------------------------------" +
                    "----------------------------------------" +
                    "----------------------------------------" +
                    "-----------------");
            }
        }

        public const int MaxFilmuSk = 500;
        public const int MaxBranchSk = 3;
        static void Main(string[] args)
        {
            //-------------------------1Lab--------------------------
            Program p = new Program();
            //List<Sąrašas> sąrašas = p.Skaitymas();
            //List<Sąrašas> Pelningasis = p.Pelningiausias(sąrašas,
p.Pelningas(sąrašas));
            //string[] vardai = new string[99999];
```

```
            //string[] pavardes = new string[99999];
            //int[] Kiek = new int[99999];
            //p.DaugSkaiciavimai(vardai, pavardes, Kiek, sąrašas);
            //int ind = p.Indeksas(vardai, pavardes, Kiek);
            //p.Spausdinimas(vardai[ind], pavardes[ind], Kiek[ind]);
            //List<Sąrašas> Nicolas = p.Cage(sąrašas);
            //List<Sąrašas> Zanras = p.Zanrai(sąrašas);
            //p.Rasymas(Pelningasis, Zanras);
            //p.ZanrasRasymas(Zanras);
            //p.CageRasymas(Nicolas);
            //p.SpausdinamPradiniusDuomenis(sąrašas);

            //Console.Read();
            //--------------------------2Lab-------------------------------

            var branchai = p.BranchuKonteinerioSukurimas();

            p.MegstamiausioRezIeskojimas(branchai);

            var visumatytifilmai = p.VisuMatytiFilmai(branchai);
            p.VisuMatytuFilmuSpausdinimas(visumatytifilmai);

            p.RekomenduojamiFilmai(branchai);

            p.PradiniaiDuomenysLentele(branchai);

        }

        /// <summary>
        /// Sukuriamas Branch'u konteineris, branchams priskiriami filmu konteineriai
        /// </summary>
        /// <returns>grazina branchu konteineri</returns>
        public BranchuKonteineris BranchuKonteinerioSukurimas ()
        {
            string[] filePaths = Directory.GetFiles(Directory.GetCurrentDirectory(),
                "*Duom.csv");
            var branchai = new BranchuKonteineris(MaxBranchSk);
            foreach (string path in filePaths)
            {
                Branch branch = null;
                bool rado = DuomenuSkaitymas(path, ref branch, out string vardas);
                if (rado == false)
                {
                    Console.WriteLine("Nera ziurovo");
                }
                SpausdintiBrancha(branch);
                branchai.PridetiBrancha(branch);
            }
            return branchai;
        }

        /// <summary>
        /// Nuskaitomi duomenys is failu
        /// </summary>
        /// <param name="path">failo vieta</param>
        /// <param name="branchess">viena atsaka(ziurovas)</param>
        /// <param name="vardas">ziurovo vardas</param>
        /// <returns>grazina ziurovo varda, filmu konteineri</returns>
        private bool DuomenuSkaitymas(string path, ref Branch branchess, out string
vardas)
        {
            using (StreamReader reader = new StreamReader(path))
            {
                string line = null;
                vardas = reader.ReadLine();
                if (vardas != null)
                {
                    line = reader.ReadLine();
                    line = reader.ReadLine();
                }
```

```
            branchess = new Branch(vardas);
            while (null != (line = reader.ReadLine()))
            {
                string[] values = line.Split(';');
                string filmas = values[0];
                int metai = int.Parse(values[1]);
                string zanras = values[2];
                string studija = values[3];
                string rezisieriausvardas = values[4];
                string rezisieriauspavarde = values[5];
                string aktorius1 = values[6];
                string aktorius2 = values[7];
                double pajamos = double.Parse(values[8]);

                Sąrašas movie = new Sąrašas(filmas, metai, zanras, studija,
                    rezisieriausvardas, rezisieriauspavarde, aktorius1,
                    aktorius2, pajamos);
                branchess.Filmai.PridetiFilma(movie);
            }
        }
        return true;
    }

    /// <summary>
    /// pradiniai duomenys surasomi i lentele tekstiniame faile
    /// </summary>
    /// <param name="branchai">ziurovu duomenys</param>
    public void PradiniaiDuomenysLentele (BranchuKonteineris branchai)
    {
       int u = 1;
        //jei true, sukurti nauja faila, o jei false, prideti
        using (StreamWriter sw = new StreamWriter(@"Duomenys2.txt", false))
        {
            for (int i = 0; i < branchai.Count; i++)
            {
                //sw.WriteLine("{0}", )
                var laikinas = branchai.GautiBrancha(i);
                sw.WriteLine("{0}", laikinas.Vardas);
                if (laikinas == null)
                {
                    continue;
                }
                //string[] lines = new string[MaxFilmuSk + u];
                sw.WriteLine("{0, 12}| {1, 7}| {2, 13}| {3, 17}|" +
                    " {4, 21}| {5, 22}| {6, 13}| {7, 14}| {8, 9}|",
                    "Pavadinimas", "Metai", "Zanras", "Studija",
                    "Rezisieriaus Vardas", "Rezisieriaus Pavarde",
                    "Aktorius1", "Aktorius2", "Pajamos");
                sw.WriteLine("-----------------------------" +
                    "-----------------------------------" +
                    "-----------------------------------" +
                    "-------------------------------------------");

                for (int j = 0; j < laikinas.Filmai.Count; j++)
                {
                    if (laikinas.Filmai.RastiFilma(j) == null)
                    {
                        continue;
                    }
                    sw.WriteLine("{0}", laikinas.Filmai.RastiFilma(j).ToString());
                    u++;
                    //lines[u+1] = String.Format("{0}", branchai)
                }
                sw.WriteLine("-----------------------------" +
                    "-----------------------------------" +
                    "-----------------------------------" +
                    "-------------------------------------------");
                sw.WriteLine();

            }
```

```
            }
        }

        /// <summary>
        /// Ieskomas dazniausiai pasikartojantis rezisierius
        /// </summary>
        /// <param name="branchai">ziurovu duomenys</param>
        public void MegstamiausioRezIeskojimas (BranchuKonteineris branchai)
        {
            Console.WriteLine("{0,20} | {1, 26} |", "Ziurovo Vardas",
                "Megstamiausias rezisierius");
            for (int i = 0; i < branchai.Count; i++)
            {
                var laikinasbranch = branchai.GautiBrancha(i);
                var laikinifilmai = laikinasbranch.Filmai;
                string megstRezPav = MegstamiausiasRezisierius(laikinifilmai);
                SpausdintiMegstRez(laikinasbranch.Vardas, megstRezPav);
            }
        }

        /// <summary>
        /// Spausdinami ziurovai ir ju megstamiausi rezisieriai
        /// </summary>
        /// <param name="ziurovas">ziurovo vardas</param>
        /// <param name="megstRezPav">jo megstamiausio rezisieriaus vardas</param>
        public void SpausdintiMegstRez (string ziurovas, string megstRezPav)
        {
            Console.WriteLine("-----------------------------------------------------");
            Console.WriteLine("{0,20} | {1, 26} |", ziurovas, megstRezPav);

        }

        /// <summary>
        /// randa unikalius rezisierius ir ju kieki
        /// </summary>
        /// <param name="Filmai">ziurovo ziuretu filmu konteineris</param>
        /// <returns> grazina dazniausiai pasikartojancio rezisieriaus varda</returns>
        public string MegstamiausiasRezisierius(FilmuKonteineris Filmai)
        {
            var rezpasikartojimodictionary = new Dictionary<string, int>();
            for (int i = 0; i < Filmai.Count; i++)
            {
                string rezPavarde = Filmai.RastiFilma(i).RezisieriausVardas
                    + " " + Filmai.RastiFilma(i).RezisieriausPavarde;
                //rezPasDict.ContainsKey(rezPavarde); // grazina arba true arba false
                if(!rezpasikartojimodictionary.ContainsKey(rezPavarde)) // jei true,
reik pridet rezisieriu i dictionary
                {
                    rezpasikartojimodictionary.Add(rezPavarde, 1);
                }
                else // jei false, padidinam jo kieki
                {
                    rezpasikartojimodictionary.TryGetValue(rezPavarde, out int value);
// paimam value
                    value++; // ji padidinam
                    rezpasikartojimodictionary[rezPavarde] = value++; // grazinam
padidinta
                }
            }
            string rezultatas = DazniausiasRez(rezpasikartojimodictionary);
            return rezultatas;
        }

        /// <summary>
        /// suzinomas dazniausio rezisieriaus konteineryje vardas
        /// </summary>
        /// <param name="rezpasikartojimodictionary">unikalus rezisieriai
        /// ir ju pasikartojimo kiekis</param>
        /// <returns>grazina dazniausiai pasikartojancio rezisieriaus varda</returns>
```

```
        public string DazniausiasRez (Dictionary<string, int>
rezpasikartojimodictionary)
        {
            string dazniausiasrez = null;
            int maxvalue = 0;
            foreach (var entry in rezpasikartojimodictionary)
            {
                string localrezisierius = entry.Key;
                int localmax = entry.Value;
                if (localmax > maxvalue)
                {
                    maxvalue = localmax;
                    dazniausiasrez = localrezisierius;
                }
            }
            return dazniausiasrez;
        }

        /// <summary>
        /// kreipiamasi i metoda, kuris sukuria nauja visu matytu filmu konteineri
        /// </summary>
        /// <param name="branchai">visi ziurovu duomenys</param>
        /// <returns>spausdinimui grazinamas visu matytu filmu konteineris</returns>
        public FilmuKonteineris VisuMatytiFilmai (BranchuKonteineris branchai)
        {
            var branch1 = branchai.GautiBrancha(0).Filmai;
            var branch2 = branchai.GautiBrancha(1).Filmai;
            var branch3 = branchai.GautiBrancha(2).Filmai;
            var visumatytifilmai = RastiVisuMatytusFilmus(branch1, branch2, branch3);
            return visumatytifilmai;
        }

        /// <summary>
        /// sudaromas visu matytu filmu konteineris
        /// </summary>
        /// <param name="filmai1">tikrinamas filmu konteineris</param>
        /// <param name="filmai2">lyginamas filmu konteineris</param>
        /// <param name="filmai3">lyginamas filmu konteineris</param>
        /// <returns>grazina visu matytu filmu konteineri</returns>
        private FilmuKonteineris RastiVisuMatytusFilmus(FilmuKonteineris filmai1,
            FilmuKonteineris filmai2, FilmuKonteineris filmai3)
        {
            FilmuKonteineris VisuMatytiFilmai = new FilmuKonteineris(MaxFilmuSk);
            for (int i = 0; i < filmai1.Count; i++)
            {
                //Jei filmai3 konteineris i filmai 2 konteineris turi filma
                //is filmai1 konteinerio, jis pridedamas i matytu vilmu konteineri
                if (filmai3.Contains(filmai1.RastiFilma(i)) &&
                    filmai2.Contains(filmai1.RastiFilma(i)))
                {
                    VisuMatytiFilmai.PridetiFilma(filmai1.RastiFilma(i));
                }
            }
            return VisuMatytiFilmai;
        }

        /// <summary>
        /// Ieskoma rekomenduojamu filmu, kreipiamasi i metoda, pridedanti nematytus
filmus
        /// </summary>
        /// <param name="branchai">ziurovu duomenys</param>
        public void RekomenduojamiFilmai(BranchuKonteineris branchai)
        {
            //var rekfilmukont = new FilmuKonteineris(MaxFilmuSk);
            for (int i = 0; i < MaxBranchSk; i++)
            {
                var branch1 = branchai.GautiBrancha(i % 3);
                var branch2 = branchai.GautiBrancha((i + 1) % 3);
                var branch3 = branchai.GautiBrancha((i + 2) % 3);
```

```
                var rekfilmukont = RastiRekomenduojamusFilmus(branch1, branch2,
branch3);
                var vardas = branchai.GautiBrancha(i).Vardas;
                RekSpausdinimas(vardas, rekfilmukont);
            }
        }

        /// <summary>
        /// Pridedami rekomenduojami filmai i konteineri
        /// </summary>
        /// <param name="branch1">vienas ziurovas</param>
        /// <param name="branch2">antras ziurovas</param>
        /// <param name="branch3">trecias ziurovas</param>
        /// <returns></returns>
        private FilmuKonteineris RastiRekomenduojamusFilmus (Branch branch1, Branch
branch2,
            Branch branch3)
        {
            var RekFilmuKont = new FilmuKonteineris(Branch.MaxFilmuSk);
            for (int i = 0; i < branch2.Filmai.Count; i++)
            {
                if (!(branch1.Filmai.Contains(branch2.Filmai.RastiFilma(i))))
                {
                    RekFilmuKont.PridetiFilma(branch2.Filmai.RastiFilma(i));
                }
                if(!(branch1.Filmai.Contains(branch3.Filmai.RastiFilma(i))))
                {
                    RekFilmuKont.PridetiFilma(branch3.Filmai.RastiFilma(i));
                }
            }
            return RekFilmuKont;
        }

        /// <summary>
        /// Spausdinami rekomenduojami filmai i ziurovui atskirai sukurta faila
        /// </summary>
        /// <param name="vardas">ziurovo vardas</param>
        /// <param name="RekomenduojamiF">jam rekomenduojamu filmu spausdinimas</param>
        public void RekSpausdinimas (string vardas, FilmuKonteineris RekomenduojamiF)
        {
            int j = 1;
            string[] lines = new string[RekomenduojamiF.Count + 1];
            lines[0] = String.Format("Pavadinimas,Metai,Zanras,Studija," +
                "Rezisieriaus Vardas,Rezisieriaus Pavarde,Aktorius1," +
                "Aktorius 2,Pajamos");
            for (int i = 0; i < RekomenduojamiF.Count; i++)
            {
                lines[j] = String.Format("{0}",
RekomenduojamiF.RastiFilma(i).ToString());
                j++;
            }
            File.WriteAllLines("Rekomendacija " + vardas + ".csv", lines);
        }

        /// <summary>
        /// spausdinamas visu ziurovu matyti filmai i faila
        /// </summary>
        /// <param name="VisuMatytiFilmai">visu matytu filmu konteineris</param>
        public void VisuMatytuFilmuSpausdinimas(FilmuKonteineris VisuMatytiFilmai)
        {
            int j = 1;
            string[] lines = new string[VisuMatytiFilmai.Count + 1];
            lines[0] = String.Format("Pavadinimas,Metai,Zanras,Studija," +
                "Rezisieriaus Vardas,Rezisieriaus Pavarde,Aktorius1," +
                "Aktorius 2,Pajamos");
            for (int i = 0; i < VisuMatytiFilmai.Count; i++)
            {
                lines[j] = String.Format("{0}",
VisuMatytiFilmai.RastiFilma(i).ToString());
                j++;
```

```
            }
            File.WriteAllLines(@"MateVisi.csv", lines);
        }

        /// <summary>
        /// spausdinami branch'ai consoleje lenteles formoje
        /// </summary>
        /// <param name="branch">spausdina ziurovu informacija lentele</param>
        void SpausdintiBrancha(Branch branch)
        {
            Console.WriteLine("{0, 12}| {1, 7}| {2, 13}| {3, 17}| {4, 21}|" +
                " {5, 22}| {6, 13}| {7, 14}| {8, 9}|", "Pavadinimas", "Metai",
                "Zanras", "Studija" ,"Rezisieriaus Vardas", "Rezisieriaus Pavarde",
                "Aktorius1", "Aktorius2", "Pajamos");
            Console.WriteLine("-----------------------------------------------------" +
                "-----------------------------------------------------------------" +
                "----------------------------------");
            Console.WriteLine("{0}", branch.ToString());
            Console.WriteLine("-----------------------------------------------------" +
                "-----------------------------------------------------------------" +
                "----------------------------------");

        }

    }
}
```

2.3. Pradiniai duomenys ir rezultatai

Pirmas bandymas:

Duomenys lentele:

Vardas Pavarde

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McConaughey	J. Chastain	165000000
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Vardenis Pavardenis

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Vardaite Pavardaite

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
Kondyske	2014	Siaubo	Mano Gyvenimas	Degantis	Laidas	A. Vejelis	V. Oras	30000
Skrodimas	2014	Siaubo	Real Life	Kunas	Plautaitis	K. Inkstas	K. Gerkle	800000
Muziejus	2015	Fantastinis	Warner Bros	Lina	Kapyte	N. Cage	M. Papinigis	982580
Meile	2002	Melodrama	Microsoft	Kunas	Plautaitis	N. Cage	K. Gerkle	475210
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Konsoleje:

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McConaughey	J. Chastain	165000000
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Ziurovas: Vardas Pavarde

```
 Pavadinimas|   Metai|        Zanras|           Studija|   Rezisieriaus Vardas|   Rezisieriaus Pavarde|     Aktorius1|      Aktorius2|   Pajamos|
      North|,   2014|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   445200|
      South|,   2013|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   524500|
       East|,   2012|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   205400|
       West|,   2011|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   150400|
  Sausainis|,   2016|,       Siaubo|,          Cookie|,               Selga|,                Nolan|,      W. Oreo|,      C. Crisp|,  8888888|

Ziurovas: Vardenis Pavardenis


 Pavadinimas|   Metai|        Zanras|           Studija|   Rezisieriaus Vardas|   Rezisieriaus Pavarde|     Aktorius1|      Aktorius2|   Pajamos|
   Kondyske|,   2014|,       Siaubo|,   Mano Gyvenimas|,             Degantis|,              Laidas|,   A. Vejelis|,       V. Oras|,    30000|
  Skrodimas|,   2014|,       Siaubo|,        Real Life|,               Kunas|,           Plautaitis|,   K. Inkstas|,     K. Gerkle|,   800000|
   Muziejus|,   2015|, Fantastinis|,     Warner Bros|,                Lina|,               Kapyte|,      N. Cage|, M. Papinigis|,   982580|
      Meile|,   2002|,   Melodrama|,       Microsoft|,               Kunas|,           Plautaitis|,      N. Cage|,     K. Gerkle|,   475210|
  Sausainis|,   2016|,       Siaubo|,          Cookie|,               Selga|,                Nolan|,      W. Oreo|,      C. Crisp|,  8888888|

Ziurovas: Vardaite Pavardaite


     Ziurovo Vardas | Megstamiausias rezisierius |
----------------------------------------------------
     Vardas Pavarde |          Christopher Nolan |
     Ziurovo Vardas | Megstamiausias rezisierius |
----------------------------------------------------
Vardenis Pavardenis |          Kaunas Miestietis |
     Ziurovo Vardas | Megstamiausias rezisierius |
----------------------------------------------------
 Vardaite Pavardaite |           Kunas Plautaitis |
Press any key to continue . . .
```

MateVisi.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
						W.	C.	
Sausainis|	2016|	Siaubo|	Cookie|	Selga|	Nolan|	Oreo|	Crisp|	8888888|

Rekomendacija_Vardaite_Pavardaite.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McConaughey	J. Chastain	165000000
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400

Rekomendacija_Vardas_Pavarde.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200
Kondyske	2014	Siaubo	Mano Gyvenimas	Degantis	Laidas	A. Vejelis	V. Oras	30000
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
Skrodimas	2014	Siaubo	Real Life	Kunas	Plautaitis	K. Inkstas	K. Gerkle	800000
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
Muziejus	2015	Fantastinis	Warner Bros	Lina	Kapyte	N. Cage	M. Papinigis	982580
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400
Meile	2002	Melodrama	Microsoft	Kunas	Plautaitis	N. Cage	K. Gerkle	475210

Rekomendacija_Vardenis_Pavardenis.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
Kondyske	2014	Siaubo	Mano Gyvenimas	Degantis	Laidas	A. Vejelis	V. Oras	30000
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McConaughey	J. Chastain	165000000
Skrodimas	2014	Siaubo	Real Life	Kunas	Plautaitis	K. Inkstas	K. Gerkle	800000
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
Muziejus	2015	Fantastinis	Warner Bros	Lina	Kapyte	N. Cage	M. Papinigis	982580
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
Meile	2002	Melodrama	Microsoft	Kunas	Plautaitis	N. Cage	K. Gerkle	475210

Antras Bandymas

Duomenys

Katinas Kandilas

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McCoghey	J. Chastain	165000
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Tarakonas Zirmunelis

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McCughey	J. Chastain	165000
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Pica Ananase

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
Kondyske	2014	Siaubo	Mano Gyvenimas	Degantis	Laidas	A. Vejelis	V. Oras	30000
Skrodimas	2014	Siaubo	Real Life	Kunas	Plautaitis	K. Inkstas	K. Gerkle	800000
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McCohey	J. Chastain	165000
Muziejus	2015	Fantastinis	Warner Bros	Lina	Kapyte	N. Cage	M. Papinigis	982580
Meile	2002	Melodrama	Microsoft	Kunas	Plautaitis	N. Cage	K. Gerkle	475210
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Konsolele:

```
Pavadinimas|   Metai|        Zanras|           Studija|   Rezisieriaus Vardas|   Rezisieriaus Pavarde|     Aktorius1|      Aktorius2|   Pajamos|
-----------------------------------------------------------------------------------------------------------------------------------------------------
      North|,   2014|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   445200|
      South|,   2013|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   524500|
       East|,   2012|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   205400|
       West|,   2011|,   Animacinis|,          Disney|,              Kaunas|,           Miestietis|,  K. Poliaris|,    A. Biednas|,   150400|
Interstellar|,   2014|,       Sci-fi|,   Legindary Pic.|,         Christopher|,                Nolan|,  M. McCoghey|,   J. Chastain|,   165000|
   Sausainis|,   2016|,       Siaubo|,          Cookie|,               Selga|,                Nolan|,      W. Oreo|,      C. Crisp|,  8888888|

Ziurovas: Katinas Kandilas


-----------------------------------------------------------------------------------------------------------------------------------------------------
 Pavadinimas|   Metai|        Zanras|           Studija|   Rezisieriaus Vardas|   Rezisieriaus Pavarde|     Aktorius1|      Aktorius2|   Pajamos|
-----------------------------------------------------------------------------------------------------------------------------------------------------
Interstellar|,   2014|,       Sci-fi|,   Legindary Pic.|,         Christopher|,                Nolan|,  M. McCughey|,   J. Chastain|,   165000|
      Naktis|,   2014|,        Drama|,        Microsoft|,                Lina|,               Kapyte|,      N. Cage|,    G. Budinas|,   250000|
     Greitas|,   2011|,      Veiksmo|,          Dreams|,               Ponas|,            Pomidoras|,    K. Luopas|,     P. Walker|,   500000|
   Sausainis|,   2016|,       Siaubo|,          Cookie|,               Selga|,                Nolan|,      W. Oreo|,      C. Crisp|,  8888888|

Ziurovas: Tarakonas Zirmunelis


-----------------------------------------------------------------------------------------------------------------------------------------------------
 Pavadinimas|   Metai|        Zanras|           Studija|   Rezisieriaus Vardas|   Rezisieriaus Pavarde|     Aktorius1|      Aktorius2|   Pajamos|
-----------------------------------------------------------------------------------------------------------------------------------------------------
    Kondyske|,   2014|,       Siaubo|,   Mano Gyvenimas|,            Degantis|,               Laidas|,   A. Vejelis|,       V. Oras|,    30000|
   Skrodimas|,   2014|,       Siaubo|,        Real Life|,               Kunas|,           Plautaitis|,   K. Inkstas|,     K. Gerkle|,   800000|
Interstellar|,   2014|,       Sci-fi|,   Legindary Pic.|,         Christopher|,                Nolan|,   M. McCohey|,   J. Chastain|,   165000|
    Muziejus|,   2015|,  Fantastinis|,      Warner Bros|,                Lina|,               Kapyte|,      N. Cage|,  M. Papinigis|,   982580|
       Meile|,   2002|,    Melodrama|,        Microsoft|,               Kunas|,           Plautaitis|,      N. Cage|,     K. Gerkle|,   475210|
   Sausainis|,   2016|,       Siaubo|,          Cookie|,               Selga|,                Nolan|,      W. Oreo|,      C. Crisp|,  8888888|

Ziurovas: Pica Ananase


-----------------------------------------------------------------------------------------------------------------------------------------------------
      Ziurovo Vardas | Megstamiausias rezisierius |
-----------------------------------------------------
    Katinas Kandilas |          Kaunas Miestietis |
-----------------------------------------------------
Tarakonas Zirmunelis |          Christopher Nolan |
-----------------------------------------------------
        Pica Ananase |           Kunas Plautaitis |
Press any key to continue . . .
```

MateVisi.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos

Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McConaughey	J. Chastain	165000000
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Rekomendacija_Katinas_Kandilas.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
Kondyske	2014	Siaubo	Mano Gyvenimas	Degantis	Laidas	A. Vejelis	V. Oras	30000
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
Skrodimas	2014	Siaubo	Real Life	Kunas	Plautaitis	K. Inkstas	K. Gerkle	800000
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
Muziejus	2015	Fantastinis	Warner Bros	Lina	Kapyte	N. Cage	M. Papinigis	982580

Rekomendacija_Pica_Ananase.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400

Rekomendacija_Tarakonas_Zirmunelis.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
Kondyske	2014	Siaubo	Mano Gyvenimas	Degantis	Laidas	A. Vejelis	V. Oras	30000
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200

Skrodimas	2014	Siaubo	Real Life	Kunas	Plautaitis	K. Inkstas	K. Gerkle	800000
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
Muziejus	2015	Fantastinis	Warner Bros	Lina	Kapyte	N. Cage	M. Papinigis	982580
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400
Meile	2002	Melodrama	Microsoft	Kunas	Plautaitis	N. Cage	K. Gerkle	475210

Trečias bandymas:

Duomenys lentele:

Katinas Kandilas

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McCoghey	J. Chastain	165000

Tarakonas Zirmunelis

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Pica Ananase

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos
North	2014	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	445200
South	2013	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	524500
East	2012	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	205400
West	2011	Animacinis	Disney	Kaunas	Miestietis	K. Poliaris	A. Biednas	150400
Interstellar	2014	Sci-fi	Legindary Pic.	Christopher	Nolan	M. McCughey	J. Chastain	165000
Naktis	2014	Drama	Microsoft	Lina	Kapyte	N. Cage	G. Budinas	250000
Greitas	2011	Veiksmo	Dreams	Ponas	Pomidoras	K. Luopas	P. Walker	500000
Sausainis	2016	Siaubo	Cookie	Selga	Nolan	W. Oreo	C. Crisp	8888888

Konsole:

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius2	Pajamos

```
-------------------------------------------------------------------------------------------------------------------------------------------------------
      North|,   2014|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   445200|
      South|,   2013|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   524500|
       East|,   2012|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   205400|
       West|,   2011|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   150400|
Interstellar|,   2014|,       Sci-fi|,   Legindary Pic.|,        Christopher|,               Nolan|,  M. McCoghey|,   J. Chastain|,   165000|

Ziurovas: Katinas Kandilas


-------------------------------------------------------------------------------------------------------------------------------------------------------
 Pavadinimas|   Metai|        Zanras|           Studija|   Rezisieriaus Vardas|   Rezisieriaus Pavarde|     Aktorius1|      Aktorius2|   Pajamos|
-------------------------------------------------------------------------------------------------------------------------------------------------------
     Naktis|,   2014|,        Drama|,        Microsoft|,               Lina|,              Kapyte|,      N. Cage|,    G. Budinas|,   250000|
    Greitas|,   2011|,      Veiksmo|,           Dreams|,              Ponas|,           Pomidoras|,    K. Luopas|,     P. Walker|,   500000|
  Sausainis|,   2016|,       Siaubo|,          Cookie|,              Selga|,              Nolan|,      W. Oreo|,      C. Crisp|,  8888888|

Ziurovas: Tarakonas Zirmunelis


-------------------------------------------------------------------------------------------------------------------------------------------------------
 Pavadinimas|   Metai|        Zanras|           Studija|   Rezisieriaus Vardas|   Rezisieriaus Pavarde|     Aktorius1|      Aktorius2|   Pajamos|
-------------------------------------------------------------------------------------------------------------------------------------------------------
      North|,   2014|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   445200|
      South|,   2013|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   524500|
       East|,   2012|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   205400|
       West|,   2011|,   Animacinis|,          Disney|,             Kaunas|,          Miestietis|,  K. Poliaris|,    A. Biednas|,   150400|
Interstellar|,   2014|,       Sci-fi|,   Legindary Pic.|,        Christopher|,               Nolan|,  M. McCughey|,   J. Chastain|,   165000|
     Naktis|,   2014|,        Drama|,        Microsoft|,               Lina|,              Kapyte|,      N. Cage|,    G. Budinas|,   250000|
    Greitas|,   2011|,      Veiksmo|,           Dreams|,              Ponas|,           Pomidoras|,    K. Luopas|,     P. Walker|,   500000|
  Sausainis|,   2016|,       Siaubo|,          Cookie|,              Selga|,              Nolan|,      W. Oreo|,      C. Crisp|,  8888888|

Ziurovas: Pica Ananase


-------------------------------------------------------
      Ziurovo Vardas | Megstamiausias rezisierius |
-------------------------------------------------------
    Katinas Kandilas |          Kaunas Miestietis |
-------------------------------------------------------
Tarakonas Zirmunelis |                Lina Kapyte |
-------------------------------------------------------
        Pica Ananase |          Kaunas Miestietis |
Press any key to continue . . .
```

MateVisi.csv

Visu matytu filmu nera

Rekomendacija Katinas Kandilas.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
Naktis|	2014|	Drama|	Microsoft|	Lina|	Kapyte|	N. Cage|	G. Budinas|	250000|
Greitas|	2011|	Veiksmo|	Dreams|	Ponas|	Pomidoras|	K. Luopas|	P. Walker|	500000|
Sausainis|	2016|	Siaubo|	Cookie|	Selga|	Nolan|	W. Oreo|	C. Crisp|	8888888|

Rekomendacija Pica Ananase.csv

Rekomenduojamu filmu nera

Rekomendacija Taraoknas Zirmunelis.csv

Pavadinimas	Metai	Zanras	Studija	Rezisieriaus Vardas	Rezisieriaus Pavarde	Aktorius1	Aktorius 2	Pajamos
North|	2014|	Animacinis|	Disney|	Kaunas|	Miestietis|	K. Poliaris|	A. Biednas|	445200|
South|	2013|	Animacinis|	Disney|	Kaunas|	Miestietis|	K. Poliaris|	A. Biednas|	524500|
East|	2012|	Animacinis|	Disney|	Kaunas|	Miestietis|	K. Poliaris|	A. Biednas|	205400|
West|	2011|	Animacinis|	Disney|	Kaunas|	Miestietis|	K. Poliaris|	A. Biednas|	150400|
Interstellar|	2014|	Sci-fi|	Legindary Pic.|	Christopher|	Nolan|	M. McCoghey|	J. Chastain|	165000|

2.4. Dėstytojo pastabos

- P5
- P8

3. Paveldėjimas

3.1. Darbo užduotis

U3_10. Buitinės technikos parduotuvė. Turite informaciją apie skirtingose buitinės technikos parduotuvėse esančius šaldytuvus. Keičiasi duomenų formatas. Pirmoje eilutėje pavadinimas, antroje – adresas, trečioje – telefonas. Parduotuvėje be šaldytuvų galima įsigyti mikrobangų krosnelių ir elektrinis virdulys. Sukurkite klasę „BuitinisPrietaisas“ (laukai - gamintojas, modelis, energijos klasė, spalva, kaina), kurią paveldės “Šaldytuvas” (papildomi laukai - talpa, montavimo tipas, požymis „turi šaldiklį“, aukštis, plotis, gylis), “MikrobangųKrosnelė” (papildomi laukai – galingumas, programų skaičius) ir “ElektrinisVirdulys” (papildomi laukai – 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. Išrikiuokite šaldytuvus nuo pigiausio iki brangiausio. Ekrane atspausdinkite šaldytuvo gamintoją, modelį, talpą ir kainą.
- Sudarykite visų buitinių prietaisų, kurių energijos klasė yra A+ ir didesnė, sąrašą. Visą informaciją apie juos įrašykite į failą „A+.csv“.
- Ar yra tokių buitinių prietaisų, kuriuos galima įsigyti tik vienoje parduotuvėje? Atspausdinkite tokių prietaisų sąrašą faile „TikTen.csv“.

3.2. Programos tekstas

Branch.cs

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

namespace _3_Laboras
{
    class Branch
    {
        public string Pavadinimas { get; set; }
        public string Adresas { get; set; }
        public string Telefonas { get; set; }
        public PrietaisuKonteineris Saldytuvai { get; set; }
        public PrietaisuKonteineris MikrobanguKrosneles { get; set; }
        public PrietaisuKonteineris ElektriniaiViduliai { get; set; }

        /// <summary>
        /// naujas issisakojimas
        /// </summary>
        /// <param name="pavadinimas">parduouves pavadinimas</param>
        /// <param name="adresas">parduotuves adresas</param>
        /// <param name="telefonas">parduotuves telefono nr</param>
        public Branch (string pavadinimas, string adresas, string telefonas)
        {
            Pavadinimas = pavadinimas;
            Adresas = adresas;
            Telefonas = telefonas;
            Saldytuvai = new PrietaisuKonteineris();
            MikrobanguKrosneles = new PrietaisuKonteineris();
            ElektriniaiViduliai = new PrietaisuKonteineris();
        }

        /// <summary>
        /// prideda saldytuva i prietaisu konteineri
        /// </summary>
        /// <param name="saldytuvas">saldytuvo info</param>
```

```
        public void PridetiSaldytuva (Saldytuvas saldytuvas)
        {
            Saldytuvai.PridetiPrietaisa(saldytuvas);
        }

        /// <summary>
        /// prideda mikrobangu krosnele i prietaisu konteineri
        /// </summary>
        /// <param name="krosnele">mikrobangu k. info</param>
        public void PridetiMikrobanguKr (MikrobanguKrosnele krosnele)
        {
            MikrobanguKrosneles.PridetiPrietaisa(krosnele);
        }

        /// <summary>
        /// prideda elektrini virduli i prietaisu konteineri
        /// </summary>
        /// <param name="virdulys">elektrinio virdulio info</param>
        public void PridetiElektriniVir (ElektrinisVidrulys virdulys)
        {
            ElektriniaiViduliai.PridetiPrietaisa(virdulys);
        }
    }
}
```

BranchuKonteineris.cs

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

namespace _3_Laboras
{
    class BranchuKonteineris
    {
        private Branch[] branches;
        public int Count { get; private set; }

        /// <summary>
        /// sukuriamas naujas issisakojimu konteineris
        /// </summary>
        /// <param name="size">konteinerio dydis</param>
        public BranchuKonteineris(int size)
        {
            branches = new Branch[size];
            Count = 0;
        }
        /// <summary>
        /// prideda parduotuve i konteineri
        /// </summary>
        /// <param name="branch">parduotuve</param>
        public void PridetiBrancha(Branch branch)
        {
            branches[Count++] = branch;
        }
        /// <summary>
        /// prideda parduotuve i konteineri i nurodyta vieta
        /// </summary>
        /// <param name="branch">parduotuve</param>
        /// <param name="index">vieta</param>
        public void PridetiBrancha(Branch branch, int index)
        {
            branches[index] = branch;
        }
        /// <summary>
        /// gauna parduotuves inforamcija
        /// </summary>
        /// <param name="index">parduotuves vieta</param>
```

```
        /// <returns>grazina parduotuves duomenis</returns>
        public Branch GautiBrancha(int index)
        {
            return branches[index];
        }
    }
}
```

BuitinisPrietaisas.cs

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

namespace _3_Laboras
{
    class BuitinisPrietaisas
    {
        public string Gamintojas { get; set; }
        public string Modelis { get; set; }
        public string EKlase { get; set; }
        public string Spalva { get; set; }
        public double Kaina { get; set; }

        public BuitinisPrietaisas (string gamintojas, string modelis,
            string eKlase, string spalva, double kaina)
        {
            Gamintojas = gamintojas;
            Modelis = modelis;
            EKlase = eKlase;
            Spalva = spalva;
            Kaina = kaina;
        }


        public override bool Equals(object obj)
        {
            return this.Equals(obj as BuitinisPrietaisas);
        }
        public bool Equals(BuitinisPrietaisas prietaisas)
        {
            if (Object.ReferenceEquals(prietaisas, null))
            {
                return false;
            }
            if (this.GetType() != prietaisas.GetType())
            {
                return false;
            }
            return (Gamintojas == prietaisas.Gamintojas) && (Modelis ==
             prietaisas.Modelis);
        }
        public override int GetHashCode()
        {
            return Gamintojas.GetHashCode() ^ Modelis.GetHashCode();
        }
        /// <summary>
        /// == uzklojimas. Sulygina desine ir kaire reiksmes, jei jos lygios, grazina
             true
        /// </summary>
        /// <param name="lhs"></param>
        /// <param name="rhs"></param>
        /// <returns></returns>
        public static bool operator ==(BuitinisPrietaisas lhs,  BuitinisPrietaisas rhs)
        {
            if (Object.ReferenceEquals(lhs, null))
            {
                if (Object.ReferenceEquals(rhs, null))
```

```
                {
                    return true;
                }
                return false;
            }
            return lhs.Equals(rhs);
        }
        public static bool operator !=(BuitinisPrietaisas lhs, BuitinisPrietaisas rhs)
        {
            return !(lhs == rhs);
        }

        /// <summary>
        /// Tikrina kairi ir desini prietaisus pagal kaina
        /// </summary>
        /// <param name="lhs"></param>
        /// <param name="rhs"></param>
        /// <returns></returns>
        public static bool operator <=(BuitinisPrietaisas lhs, BuitinisPrietaisas rhs)
        {
            return (lhs.Kaina <= rhs.Kaina);
        }
        public static bool operator >=(BuitinisPrietaisas lhs, BuitinisPrietaisas rhs)
        {
            return (lhs.Kaina >= rhs.Kaina);
        }
    }
}
```

PrietaisuKonteineris.cs

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

namespace _3_Laboras
{
    class PrietaisuKonteineris
    {
        private BuitinisPrietaisas[] Prietaisai;
        public int Count { get; private set; }

        /// <summary>
        /// naujas prietaisu konteineris
        /// </summary>
        public PrietaisuKonteineris ()
        {
            Prietaisai = new BuitinisPrietaisas[100];
            Count = 0;
        }
        /// <summary>
        /// prideda prietaisa i konteineri
        /// </summary>
        /// <param name="prietaisas">prietaiso informacija</param>
        public void PridetiPrietaisa (BuitinisPrietaisas prietaisas)
        {
            Prietaisai[Count++] = prietaisas;
        }
        /// <summary>
        /// prideda prietaisa i nurodyta konteinerio vieta
        /// </summary>
        /// <param name="prietaisas">prietaiso info</param>
        /// <param name="index">konteinerio vieta</param>
        public void NustatytiPrietaisa (BuitinisPrietaisas prietaisas, int index)
        {
            Prietaisai[index] = prietaisas;
        }
        /// <summary>
```

```
        /// gauna prietaisa is konteinerio
        /// </summary>
        /// <param name="index">vieta konteineryje</param>
        /// <returns>prietaiso info</returns>
        public BuitinisPrietaisas GautiPrietaisa (int index)
        {
            return Prietaisai[index];
        }

        /// <summary>
        /// contains metodas. Tikrina ar konteineryje yra duodamas metodui prietaisas
        /// </summary>
        /// <param name="prietaisas">prietaiso info</param>
        /// <returns></returns>
        public bool Contains(BuitinisPrietaisas prietaisas)
        {
            return Prietaisai.Contains(prietaisas);
        }

        /// <summary>
        /// rusiuoja prietaisus (pagal ka rusiuoja galima pakeisti uzklojimuose)
        /// </summary>
        public void RusiuotiPrietaisus()
        {
            for (int i = 0; i < Count - 1; i++)
            {
                BuitinisPrietaisas mazReiksmesPrietaisas = Prietaisai[i];
                int mazReiksmesIndeksas = i;
                for (int j = i + 1; j < Count; j++)
                {
                    if(Prietaisai[j] <= mazReiksmesPrietaisas)
                    {
                        mazReiksmesPrietaisas = Prietaisai[j];
                        mazReiksmesIndeksas = j;
                    }
                }
                Prietaisai[mazReiksmesIndeksas] = Prietaisai[i];
                Prietaisai[i] = mazReiksmesPrietaisas;
            }
        }

        /// <summary>
        /// istrina prietaisa
        /// </summary>
        /// <param name="prietaisas">prietaiso info</param>
        public void IstrintiPrietaisa(BuitinisPrietaisas prietaisas)
        {
            int i = 0;
            while (i < Count)
            {
                if (Prietaisai[i].Equals(prietaisas))
                {
                    Count--;
                    for (int j = i; j < Count; j++)
                    {
                        Prietaisai[j] = Prietaisai[j + 1];
                    }
                    break;
                }
                i++;
            }
        }
    }
}
```

Saldytuvas.cs

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

```
using System.Text;
using System.Threading.Tasks;

namespace _3_Laboras
{
    class Saldytuvas : BuitinisPrietaisas
    {
        public int Talpa { get; set; }
        public string MontavimoTipas { get; set; }
        public bool Saldiklis { get; set; }
        public double Aukstis { get; set; }
        public double Plotis { get; set; }
        public double Gylis { get; set; }

        /// <summary>
        /// saldytuvo informacija
        /// </summary>
        /// <param name="gamintojas">gamintojas</param>
        /// <param name="modelis">modelis</param>
        /// <param name="eKlase">energijos klase</param>
        /// <param name="spalva">spalva</param>
        /// <param name="kaina">kaina</param>
        /// <param name="talpa">talpa</param>
        /// <param name="montavimoTipas">montavimo tipas</param>
        /// <param name="saldiklis">ar yra saldiklis</param>
        /// <param name="aukstis">aukstis</param>
        /// <param name="plotis">plotis</param>
        /// <param name="gylis">gylis</param>
        public Saldytuvas(string gamintojas, string modelis, string eKlase,
            string spalva, double kaina, int talpa, string montavimoTipas,
            bool saldiklis, double aukstis, double plotis, double gylis) :
            base(gamintojas, modelis, eKlase, spalva, kaina)
        {
            Talpa = talpa;
            MontavimoTipas = montavimoTipas;
            Saldiklis = saldiklis;
            Aukstis = aukstis;
            Plotis = plotis;
            Gylis = gylis;
        }

        /// <summary>
        /// Metodas, i kuri kreipiamasi norint spausdinti saldytuvo duomenis
        /// </summary>
        /// <returns>suformatuotus saldytuvo duomenis</returns>
        public override string ToString()
        {
            return String.Format("{0, -10}, {1, -7}, {2, -8}, {3, -10}, {4, -5}," +
                " {5, -5}, {6, -15}, {7, -8}, {8, -6}, {9, -5}, {10, -5}", Gamintojas,
                Modelis, EKlase, Spalva, Kaina, Talpa, MontavimoTipas, Saldiklis,
                   Aukstis,
                Plotis, Gylis);

        }

        /// <summary>
        /// leizia lyginti saldytuvus
        /// </summary>
        /// <param name="obj"></param>
        /// <returns></returns>
        public override bool Equals(object obj)
        {
            return this.Equals(obj as Saldytuvas);
        }

        public bool Equals(Saldytuvas saldytuvas)
        {
            return base.Equals(saldytuvas);
        }
```

```
        public override int GetHashCode()
        {
            return Gamintojas.GetHashCode() ^ Modelis.GetHashCode();
        }

        /// <summary>
        /// Lyginamos kaires ir desines reiksmes, jei lygiios, grazinama true
        /// </summary>
        /// <param name="lhs">kairys lyginamasis</param>
        /// <param name="rhs">desinysis lyginamasis</param>
        /// <returns></returns>
        public static bool operator ==(Saldytuvas lhs, Saldytuvas rhs)
        {
            if (Object.ReferenceEquals(lhs, null))
            {
                if (Object.ReferenceEquals(rhs, null))
                {
                    return true;
                }
                return false;
            }
            return lhs.Equals(rhs);
        }

        public static bool operator !=(Saldytuvas lhs, Saldytuvas rhs)
        {
            return !(lhs == rhs);
        }
    }
}
```

ElektrinisVirdulys.cs

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

namespace _3_Laboras
{
    class ElektrinisVidrulys : BuitinisPrietaisas
    {
        public int Galia { get; set; }
        public int Turis { get; set; }

        /// <summary>
        /// naujas elektrinis vird.
        /// </summary>
        /// <param name="gamintojas">gamintojas</param>
        /// <param name="modelis">modelis</param>
        /// <param name="eKlase">energijos klase</param>
        /// <param name="spalva">spalva</param>
        /// <param name="kaina">kaina</param>
        /// <param name="galia">galia</param>
        /// <param name="turis">turis</param>
        public ElektrinisVidrulys (string gamintojas, string modelis, string eKlase
            , string spalva, double kaina, int galia, int turis) :
            base (gamintojas, modelis, eKlase, spalva, kaina)
        {
            Galia = galia;
            Turis = turis;
        }

        /// <summary>
        /// Metodas, i kuri kreipiamasi norint spausdinti elektrinio virdulio duomenis
        /// </summary>
        /// <returns>suformatuota informacija</returns>
        public override string ToString()
```

```
        {
            return String.Format("{0, -10}, {1, -9}, {2, -8}, {3, -7}, {4, -3}, {5, -
                   5}," +
                " {6, -5}", Gamintojas, Modelis, EKlase, Spalva, Kaina, Galia, Turis);

        }

        /// <summary>
        /// leidzia lyginti du virdulius
        /// </summary>
        /// <param name="obj">tikrinamas objektas/param>
        /// <returns></returns>
        public override bool Equals(object obj)
        {
            return this.Equals(obj as ElektrinisVidrulys);
        }

        public bool Equals(ElektrinisVidrulys elektrinisVirdulys)
        {
            return base.Equals(elektrinisVirdulys);
        }

        public override int GetHashCode()
        {
            return Gamintojas.GetHashCode() ^ Modelis.GetHashCode();
        }

        /// <summary>
        /// lygina desine ir kaire reiksmes, jei jos lygios, grazina true, o jei ne,
             grazina            false
        /// </summary>
        /// <param name="lhs">kairysis lyginamasis</param>
        /// <param name="rhs">desinysis lyginamasis</param>
        /// <returns></returns>
        public static bool operator ==(ElektrinisVidrulys lhs, ElektrinisVidrulys rhs)
        {
            if (Object.ReferenceEquals(lhs, null))
            {
                if (Object.ReferenceEquals(rhs, null))
                {
                    return true;
                }
                return false;
            }
            return lhs.Equals(rhs);
        }

        public static bool operator !=(ElektrinisVidrulys lhs, ElektrinisVidrulys rhs)
        {
            return !(lhs == rhs);
        }
    }
}
```

MikrobanguKrosnele.cs

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

namespace _3_Laboras
{
    class MikrobanguKrosnele : BuitinisPrietaisas
    {
        public int Galingumas { get; set; }
        public int ProgSkaicius { get; set; }

        /// <summary>
```

```
/// nauja krosnele
/// </summary>
/// <param name="gamintojas">gamintojas</param>
/// <param name="modelis">modelis</param>
/// <param name="eKlase">energijos klase</param>
/// <param name="spalva">slapva</param>
/// <param name="kaina">kaina</param>
/// <param name="galingumas">galingumas</param>
/// <param name="progSkaicius">programu skaicius</param>
public MikrobanguKrosnele (string gamintojas, string modelis, string eKlase,
    string spalva, double kaina, int galingumas, int progSkaicius) :
    base(gamintojas, modelis, eKlase, spalva, kaina)
{
    Galingumas = galingumas;
    ProgSkaicius = progSkaicius;
}

/// <summary>
/// Metodas, i kuri kreipiamasi norint spausdinti mikrobangu krosneles duomenis
/// </summary>
/// <returns>suformatuota informacija</returns>
public override string ToString()
{
    return String.Format("{0, -10}, {1, -7}, {2, -10}, {3, -7}, {4, -5}, {5, -
           9}," +
        " {6, -5}", Gamintojas, Modelis, EKlase, Spalva, Kaina, Galingumas,
        ProgSkaicius);

}

/// <summary>
/// leidzia lyginti dvi krosneles
/// </summary>
/// <param name="obj"></param>
/// <returns></returns>
public override bool Equals(object obj)
{
    return this.Equals(obj as MikrobanguKrosnele);
}

public bool Equals(MikrobanguKrosnele mikrobanguKrosnele)
{
    return base.Equals(mikrobanguKrosnele);
}

public override int GetHashCode()
{
    return Gamintojas.GetHashCode() ^ Modelis.GetHashCode();
}

/// <summary>
/// sulygina kaire ir desine reiksmes, jei jos lygios grazina true
/// </summary>
/// <param name="lhs">kairysis lyginamasis</param>
/// <param name="rhs">desinysis lyginamasis</param>
/// <returns></returns>
public static bool operator ==(MikrobanguKrosnele lhs, MikrobanguKrosnele rhs)
{
    if (Object.ReferenceEquals(lhs, null))
    {
        if (Object.ReferenceEquals(rhs, null))
        {
            return true;
        }
        return false;
    }
    return lhs.Equals(rhs);
}

public static bool operator !=(MikrobanguKrosnele lhs, MikrobanguKrosnele rhs)
```

```
        {
            return !(lhs == rhs);
        }
    }
}
```

Program.cs

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

namespace _3_Laboras
{
    class Program
    {
        public const int MaxBranchNr = 3;
        public const int MaxPrietaisuSk = 20;
        static void Main(string[] args)
        {
            Program p = new Program();

            BranchuKonteineris branches = p.BranchuKonteinerioSukurimas();
            p.PradiniaiDuomenys(branches, "Duomenys.txt");

            //Pirmas Punktas
            p.SiemensProduktai(branches);

            //Antras Punktas
            var Saldytuvai = p.SaldytuvuSarasas(branches);
            Saldytuvai.RusiuotiPrietaisus();
            p.TinkamuSaldytuvuS(Saldytuvai);

            //Trecias Punktas
            var APlius = p.EnergijosKlase(branches);
            p.SpausdinimasIFaila(APlius, "A+.csv");

            //Ketvirtas Punktas
            var Nesikartoja = p.KurieNesikartoja(branches);
            p.SpausdinimasIFaila(Nesikartoja, "TikTen.csv");

            Console.ReadLine();

        }

        /// <summary>
        /// Pradiniai duomenys lentele txt faile
        /// </summary>
        /// <param name="branch">visa parduotuviu informacija</param>
        /// <param name="file">failo vieta</param>
        private void PradiniaiDuomenys(BranchuKonteineris branch, string file)
        {
            using (StreamWriter sw = new StreamWriter(@file, false))
            {
                for (int i = 0; i < branch.Count; i++)
                {
                    sw.WriteLine("-----------------------------------------------------
                    ---");
                    sw.WriteLine("{0} {1} {2}", branch.GautiBrancha(i).Pavadinimas,
                         branch.GautiBrancha(i).Adresas,
                         branch.GautiBrancha(i).Telefonas);
                    sw.WriteLine("-----------------------------------------------------
                    ---");
                    //SpausdinimasIFaila(branch.GautiBrancha(i).Saldytuvai, file);
                    //SpausdinimasIFaila(branch.GautiBrancha(i).MikrobanguKrosneles,
                         file);
```

```
                //SpausdinimasIFaila(branch.GautiBrancha(i).ElektriniaiViduliai,
               file);

                sw.WriteLine("Saldytuvai");
                sw.WriteLine("Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa
                     Montavimo Tipas Saldiklis Aukstis Plotis Gylis");
                for (int j = 0; j < branch.GautiBrancha(i).Saldytuvai.Count; j++)
                {

                     sw.WriteLine(branch.GautiBrancha(i).Saldytuvai.
                     GautiPrietaisa(j).ToString());
                }
                sw.WriteLine();

                sw.WriteLine("Mikrobangu Krosneles");
                sw.WriteLine("Gamintojas Modelis   E.  Tipas  Spalva  Kaina
                     Galingumas Programu Sk.");
                for (int j = 0; j <
                     branch.GautiBrancha(i).MikrobanguKrosneles.Count; j++)
                {
                    sw.WriteLine(branch.GautiBrancha(i).MikrobanguKrosneles.
                     GautiPrietaisa(j).ToString());
                }
                sw.WriteLine();

                sw.WriteLine("Elektriniai Virduliai");
                sw.WriteLine("Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia
                     Turis");
                for (int j = 0; j <
                     branch.GautiBrancha(i).ElektriniaiViduliai.Count; j++)
                {
                    sw.WriteLine(branch.GautiBrancha(i).ElektriniaiViduliai.
                     GautiPrietaisa(j).ToString());
                }
                sw.WriteLine();
            }
        }
    }

    /// <summary>
    /// Iesko prietaisu, kurie nesikartoja ir juos iraso i nauja prietaisu
         konteineri
    /// </summary>
    /// <param name="Parduotuves">parduotuviu informacija</param>
    /// <returns>nesikartojanciu prietaisu konteineris</returns>
    private PrietaisuKonteineris KurieNesikartoja(BranchuKonteineris Parduotuves)
    {
        PrietaisuKonteineris Nauji = new PrietaisuKonteineris();

        for(int i = 0; i < Parduotuves.Count; i++)
        {
            for(int j = 0; j < Parduotuves.GautiBrancha(i).Saldytuvai.Count; j++)
            {
                if (!AryYraParduotuvese(Parduotuves, i,
                     Parduotuves.GautiBrancha(i).
               Saldytuvai.GautiPrietaisa(j), Nauji))
                {
                    Nauji.PridetiPrietaisa(Parduotuves.GautiBrancha(i).
                     Saldytuvai.GautiPrietaisa(j));
                }
            }
            for (int j = 0; j <
               Parduotuves.GautiBrancha(i).MikrobanguKrosneles.Count; j++)
            {
                if (!AryYraParduotuvese(Parduotuves, i,
                     Parduotuves.GautiBrancha(i).MikrobanguKrosneles
               .GautiPrietaisa(j), Nauji))
                {
                    Nauji.PridetiPrietaisa(Parduotuves.GautiBrancha(i).
                     MikrobanguKrosneles.GautiPrietaisa(j));
```

```
                }
            }
            for (int j = 0; j <
               Parduotuves.GautiBrancha(i).ElektriniaiViduliai.Count; j++)
            {
                if (!AryYraParduotuvese(Parduotuves, i,
                      Parduotuves.GautiBrancha(i).ElektriniaiViduliai.
               GautiPrietaisa(j), Nauji))
                {
                    Nauji.PridetiPrietaisa(Parduotuves.
               GautiBrancha(i).ElektriniaiViduliai.GautiPrietaisa(j));
                }
            }
        }
        return Nauji;
    }

    /// <summary>
    /// Tikrina ar kiekviena parduotuve turi tikrinama prietaisa
    /// </summary>
    /// <param name="Parduotuves">parduotuviu informacija</param>
    /// <param name="ind">tikrinamo prietaiso vieta konteineryje</param>
    /// <param name="prietaisas">prietaiso informacija</param>
    /// <returns>true arba false(yra arba nera)</returns>
    bool AryYraParduotuvese(BranchuKonteineris Parduotuves, int ind,
         BuitinisPrietaisas
         prietaisas, PrietaisuKonteineris nauji)
    {
        for(int i = ind + 1 ; i < Parduotuves.Count; i++)
        {
            if (Parduotuves.GautiBrancha(i).Saldytuvai.Contains(prietaisas) ||
                    (nauji.Contains(prietaisas)))
                return true;
            if (Parduotuves.GautiBrancha(i).
               MikrobanguKrosneles.Contains(prietaisas) ||
               (nauji.Contains(prietaisas)))
                return true;
            if
               (Parduotuves.GautiBrancha(i).ElektriniaiViduliai.
               Contains(prietaisas) || (nauji.Contains(prietaisas)))
                return true;
        }
        return false;
    }

    /// <summary>
    /// Spausdina pirmus 10 tinkamu saldytuvu
    /// </summary>
    /// <param name="saldytuvai">saldytuvu konteineris</param>
    private void TinkamuSaldytuvuS (PrietaisuKonteineris saldytuvai)
    {
        int skaicius = 0;
        //Saldytuvas saldyt = saldytuvai.GautiPrietaisa(i) as Saldytuvas;
        Console.WriteLine("Saldytuvai, kuriu talpa didesne uz 80 surusiuoti pagal
         kaina: ");
        Console.WriteLine();
        Console.WriteLine("Gamintojas Modelis   Talpa  Kaina");
        Console.WriteLine("---------------------------------");
        for (int i = 0; i < saldytuvai.Count; i++)
        {
            Saldytuvas saldyt = saldytuvai.GautiPrietaisa(i) as Saldytuvas;
            if (skaicius == 10)
                break;
            Console.WriteLine("{0,-10} {1,-10} {2,-5} {3,-5}",
                     saldytuvai.GautiPrietaisa(i).Gamintojas,
                           saldytuvai.GautiPrietaisa(i).Modelis, saldyt.Talpa,
                                 saldytuvai.GautiPrietaisa(i).Kaina);
            Console.WriteLine("---------------------------------");
            skaicius++;
        }
```

```
}

/// <summary>
/// is visu saldytuvu atrenka tinkamus ir kurie nesikartoja
/// </summary>
/// <param name="saldytuvai">sadytuvu konteineris</param>
/// <param name="prietaisai">pildomas prietaisu konteineris</param>
/// <returns>tinkamu saldytuvu sarasas</returns>
private PrietaisuKonteineris TinkamiSaldytuvai (PrietaisuKonteineris
     saldytuvai, PrietaisuKonteineris prietaisai)
{

    for (int i = 0; i < saldytuvai.Count; i++)
    {
        Saldytuvas saldyt = saldytuvai.GautiPrietaisa(i) as Saldytuvas;

        if ((saldyt.Talpa >= 80) && (saldyt.MontavimoTipas == "Pastatomas") &&
                  (!prietaisai.Contains(saldytuvai.GautiPrietaisa(i))))
        {
            prietaisai.PridetiPrietaisa(saldytuvai.GautiPrietaisa(i));
        }
    }
    return prietaisai;
}

/// <summary>
/// sukuria saldytuvu sarasa, kuriame yra visu parduotuviu saldytuvai
/// </summary>
/// <param name="branch">parduotuviu informacija</param>
/// <returns>Visu saldytuvu sarasas</returns>
private PrietaisuKonteineris SaldytuvuSarasas (BranchuKonteineris branch)
{
    //branch.GautiBrancha(0).Saldytuvai.GautiPrietaisa(0).
    var prietaisai = new PrietaisuKonteineris();

    for (int i = 0; i < branch.Count; i++)
    {
        var S = branch.GautiBrancha(i).Saldytuvai;
        prietaisai = TinkamiSaldytuvai(S, prietaisai);

    }
    return prietaisai;
}

/// <summary>
/// Spausdinima paduoto prietaiso konteinerio duomenis i paduoto failo
     pavadinima
/// </summary>
/// <param name="prietaisas">prietaisu konteineris, prietaisu
     informacija</param>
/// <param name="fileName">failo pavadinimas</param>
private void SpausdinimasIFaila (PrietaisuKonteineris prietaisas, string
     fileName)
{
    int j = 1;

    using (StreamWriter sw = new StreamWriter(@fileName, false))
    {
        sw.WriteLine("Gamintojas, Modelis, Energijos Klase, Spalva, Kaina,
                  Talpa/Galingumas/Galia, Montavimo t./Programu sk. /Turis,
                  Saldiklis, Aukstis, Plotis, Gylis");
        for (int i = 0; i < prietaisas.Count; i++)
        {
            sw.WriteLine("{0}", prietaisas.GautiPrietaisa(i).ToString());
            j++;
        }
    }
}

/// <summary>
```

```
/// sukuria konteineri prietaisu, kuriu energijos klase yra A+ arba didesne
/// </summary>
/// <param name="prietaisai">prietaisu informacija</param>
/// <param name="aPlius">pildomas konteineris tinkamais prietaisais</param>
/// <returns>Tinkamos energijos klases prietaisu konteineris</returns>
private PrietaisuKonteineris PrietaisaiEK(PrietaisuKonteineris prietaisai,
           PrietaisuKonteineris aPlius)
{

    for (int i = 0; i < prietaisai.Count; i++)
    {
        if ((prietaisai.GautiPrietaisa(i).EKlase == "A+") ||
           (prietaisai.GautiPrietaisa(i).EKlase == "A++") ||
           (prietaisai.GautiPrietaisa(i).EKlase == "A+++"))
        {
            if(!aPlius.Contains(prietaisai.GautiPrietaisa(i)))
            {
                aPlius.PridetiPrietaisa(prietaisai.GautiPrietaisa(i));
            }
            //aPlius.PridetiPrietaisa(prietaisai.GautiPrietaisa(i));
        }
    }
    return aPlius;
}

/// <summary>
/// Surenka visus prietaisus i viena konteineri
/// </summary>
/// <param name="branch">parduotuviu informacija</param>
/// <returns>visu prietaisu konteineris</returns>
private PrietaisuKonteineris EnergijosKlase (BranchuKonteineris branch)
{
    var APlius = new PrietaisuKonteineris();
    for (int i = 0; i < branch.Count; i++)
    {
        var Saldytuvai = branch.GautiBrancha(i).Saldytuvai;
        APlius = PrietaisaiEK(Saldytuvai, APlius);

        var MikrobanguK = branch.GautiBrancha(i).MikrobanguKrosneles;
        APlius = PrietaisaiEK(MikrobanguK, APlius);

        var ElektrinisV = branch.GautiBrancha(i).ElektriniaiViduliai;
        APlius = PrietaisaiEK(ElektrinisV, APlius);

    }
    return APlius;
}

/// <summary>
///
/// </summary>
/// <param name="parduotuvesDuom">parduotuves duomenys</param>
/// <param name="saldytuvuK">parduotuves saldytuvu kiekis</param>
/// <param name="MikrobanguKK">parduotuves mikrobangu k. kiekis</param>
/// <param name="eleVK">parduotuves elektriniu virduliu kiekis</param>
private void SiemensSpausdinimas (string parduotuvesDuom, int saldytuvuK, int
           MikrobanguKK, int eleVK)
{
    Console.WriteLine("{0}", parduotuvesDuom);
    Console.WriteLine("----------------------------------------");
    Console.WriteLine("Siemens saldytuvu: {0}", saldytuvuK);
    Console.WriteLine("----------------------------------------");
    Console.WriteLine("Siemens Mikrobangu K.: {0}", MikrobanguKK);
    Console.WriteLine("----------------------------------------");
    Console.WriteLine("Siemens Elektriniu V.: {0}", eleVK);
    Console.WriteLine("----------------------------------------");
    Console.WriteLine();
}

/// <summary>
```

```
/// Randa kiekvieno produkto tipo siemens produktu kieki, atspausdina atsakyma
     i konsole
/// </summary>
/// <param name="branch">parduotuviu informacija</param>
private void SiemensProduktai(BranchuKonteineris branch)
{
    for (int i = 0; i < branch.Count; i++)
    {
        string parduotuvesDuomenys = (branch.GautiBrancha(i).Pavadinimas + ", "
           + branch.GautiBrancha(i).Adresas + ", " +
           branch.GautiBrancha(i).Telefonas);

        var saldytuvai = branch.GautiBrancha(i).Saldytuvai;
        var siemensSaldytuvai = SiemensKiekis(saldytuvai);

        var mikrobanguK = branch.GautiBrancha(i).MikrobanguKrosneles;
        var siemensMikrobanguK = SiemensKiekis(mikrobanguK);

        var eleVirduliai = branch.GautiBrancha(i).ElektriniaiViduliai;
        var siemensEleVirduliai = SiemensKiekis(eleVirduliai);

        SiemensSpausdinimas(parduotuvesDuomenys, siemensSaldytuvai,
           siemensMikrobanguK, siemensEleVirduliai);

    }
}

/// <summary>
/// Tikrina kiek siemens vieno produkto tipo turi
/// </summary>
/// <param name="prietaisai">simens produktu konteineris</param>
/// <returns> siemens produkto tipo kieki</returns>
private int SiemensKiekis (PrietaisuKonteineris prietaisai)
{
    int kiekis = 0;
    for (int i = 0; i < prietaisai.Count; i++)
    {
        if (prietaisai.GautiPrietaisa(i).Gamintojas.Contains("Siemens"))
        {
            kiekis++;
        }
    }
    return kiekis;
}

/// <summary>
/// sukuria branchu konteineri, i ji iraso parduotuves ir prekiu duomenis
/// </summary>
/// <returns>Branchu konteineti</returns>
private BranchuKonteineris BranchuKonteinerioSukurimas ()
{
    string[] filePaths = Directory.GetFiles(Directory.GetCurrentDirectory(),
           "*Parduotuve.csv");
    var branchai = new BranchuKonteineris(MaxBranchNr);
    foreach (var path in filePaths)
    {
        Branch branch = null;
        bool rado = ParduotuviuDuomenuSkaitymas(path, ref branch, out string
           vardas, out string adresas, out string numeris);
        if (rado == false)
        {
            Console.WriteLine("Nera Parduotuves");
        }
        //SausdintiBrancha(branch);
        branchai.PridetiBrancha(branch);
    }
    return branchai;
}

/// <summary>
```

```
/// Nuskaito parduotuves ir jos prekiu duomenis
/// </summary>
/// <param name="path">failo vardas</param>
/// <param name="branchai">naujas issisakojimas (parduotuve)</param>
/// <param name="pavadinimas">parduotuves pavadinimas</param>
/// <param name="adresas">parduotuves adresas</param>
/// <param name="numeris">parduotuves telefono nr</param>
/// <returns></returns>
public bool ParduotuviuDuomenuSkaitymas(string path, ref Branch branchai, out
     string pavadinimas, out string adresas, out string numeris)
{
    using (StreamReader sr = new StreamReader(path))
    {
        string line = null;
        pavadinimas = sr.ReadLine();
        adresas = sr.ReadLine();
        numeris = sr.ReadLine();
        line = pavadinimas;
        if (line == null)
            return false;
        branchai = new Branch(pavadinimas, adresas, numeris);
        while(null != (line = sr.ReadLine()))
        {
            string[] values = line.Split(';');
            char type = line[0];
            string gamintojas = values[1];
            string modelis = values[2];
            string eneKlase = values[3];
            string spalva = values[4];
            double kaina = double.Parse(values[5]);

            switch(type)
            {
                case 'S':
                    int talpa = int.Parse(values[6]);
                    string montavimoTipas = values[7];
                    bool saldiklis = bool.Parse(values[8]);
                    double aukstis = double.Parse(values[9]);
                    double plotis = double.Parse(values[10]);
                    double gylis = double.Parse(values[11]);
                    Saldytuvas saldytuvas = new Saldytuvas(gamintojas, modelis,
                        eneKlase, spalva, kaina, talpa, montavimoTipas,
                        saldiklis, aukstis, plotis, gylis);
                    if (!branchai.Saldytuvai.Contains(saldytuvas))
                    {
                        branchai.PridetiSaldytuva(saldytuvas);
                    }
                    //branchai.PridetiSaldytuva(saldytuvas);
                    break;

                case 'M':
                    int galingumas = int.Parse(values[6]);
                    int progSkaicius = int.Parse(values[7]);
                    MikrobanguKrosnele mKrosnele = new
                        MikrobanguKrosnele(gamintojas, modelis, eneKlase,
                        spalva, kaina, galingumas, progSkaicius);
                    if (!branchai.MikrobanguKrosneles.Contains(mKrosnele))
                    {
                        branchai.PridetiMikrobanguKr(mKrosnele);
                    }
                    //branchai.PridetiMikrobanguKr(mKrosnele);
                    break;

                case 'E':
                    int galia = int.Parse(values[6]);
                    int turis = int.Parse(values[7]);
                    ElektrinisVidrulys eVirdulys = new
                        ElektrinisVidrulys(gamintojas, modelis, eneKlase,
                        spalva, kaina, galia, turis);
                    if (!branchai.ElektriniaiViduliai.Contains(eVirdulys))
```

```
                                {
                                    branchai.PridetiElektriniVir(eVirdulys);
                                }
                                //branchai.PridetiElektriniVir(eVirdulys);
                                break;
                            }
                        }
                        return true;
                    }
                }
            }
        }
```

3.3. Pradiniai duomenys ir rezultatai

1 Bandymas

```
-----------------------------------------------------
Elektromarkt Adreso g. 3 861948205
-----------------------------------------------------
SALDYTUVAI
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Snaige    , A980   , A+      , Balta     , 900  , 85   , Pastatomas     , True    , 15    , 5    , 8
Siemens   , C 10-25, A++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Samsung   , 10-75  , A       , Balta     , 700  , 80   , Pastatomas     , True    , 15    , 8    , 6
LG        , A900   , B       , Balta     , 500  , 60   , Pastatomas     , False   , 1     , 1    , 7
Siemens   , 100B   , A+++    , Raudona   , 2000 , 100  , Pastatomas     , True    , 2     , 2    , 18

MIKROBANGES
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , A++       , Balta  , 200  , 220      , 6
Samsung   , 2120   , B         , Juoda  , 150  , 230      , 5

VIRDULIAI
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , A++     , Juoda  , 40 , 220  , 5

-----------------------------------------------------
Avitela Gatves g. 30 862561221
-----------------------------------------------------
SALDYTUVAI
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Siemens   , C 10-25, A++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Snaige    , A980   , A+      , Balta     , 900  , 75   , Pastatomas     , True    , 15    , 5    , 8

MIKROBANGES
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , A++       , Balta  , 200  , 220      , 6

VIRDULIAI
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , A++     , Juoda  , 40 , 220  , 5

-----------------------------------------------------
Technorama Alejos g. 12 8625112521
-----------------------------------------------------
SALDYTUVAI
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Siemens   , 100B   , A+++    , Raudona   , 2000 , 100  , Pastatomas     , True    , 2     , 2    , 18
Snaige    , A980   , A+      , Balta     , 900  , 85   , Pastatomas     , True    , 15    , 5    , 8
Siemens   , C 10-25, A++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18

MIKROBANGES
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , A++       , Balta  , 200  , 220      , 6

VIRDULIAI
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , A++     , Juoda  , 40 , 220  , 5
```

Konsole:

```
Elektromarkt, Adreso g. 3, 861948205
----------------------------------------
Siemens saldytuvu: 2
```

```
----------------------------------------
Siemens Mikrobangu K.: 1
----------------------------------------
Siemens Elektriniu V.: 0
----------------------------------------

Avitela, Gatves g. 30, 862561221
----------------------------------------
Siemens saldytuvu: 1
----------------------------------------
Siemens Mikrobangu K.: 1
----------------------------------------
Siemens Elektriniu V.: 0
----------------------------------------

Technorama, Alejos g. 12, 8625112521
----------------------------------------
Siemens saldytuvu: 2
----------------------------------------
Siemens Mikrobangu K.: 1
----------------------------------------
Siemens Elektriniu V.: 0
----------------------------------------

Saldytuvai, kuriu talpa didesne uz 80 surusiuoti pagal kaina:

Gamintojas Modelis   Talpa  Kaina
---------------------------------
Samsung    10-75      80    700
---------------------------------
Snaige     A980       85    900
---------------------------------
Siemens    C 10-25    90    1200
---------------------------------
Siemens    100B       100   2000
---------------------------------
```

A+.csv

Gamintojas	Modelis	Energijos Klase	Spalva	Kaina					
Snaige	A980	A+	Balta	900	85	Pastatomas	True	15	5
Siemens	C 10-25	A++	Juoda	1200	90	Pastatomas	True	2	1
Siemens	100B	A+++	Raudona	2000	100	Pastatomas	True	2	2
Siemens	B-200	A++	Balta	200	220	6			
Keprel	20-1	A++	Juoda	40	220	5			

TikTen.csv

Gamintojas	Modelis	Energijos Klase	Spalva	Kaina					
Samsung	10-75	A	Balta	700	80	Pastatomas	True	15	8
LG	A900	B	Balta	500	60	Pastatomas	False	1	1
Samsung	2120	B	Juoda	150	230	5			
Siemens	100B	A+++	Raudona	2000	100	Pastatomas	True	2	2
Snaige	A980	A+	Balta	900	85	Pastatomas	True	15	5

Siemens	C 10-25	A++	Juoda	1200	90	Pastatomas	True	2	1
Siemens	B-200	A++	Balta	200	220	6			
Keprel	20-1	A++	Juoda	40	220	5			

2 Bandymas:

```
-------------------------------------------------------
Topo Centras Dauksio g. 3 861251205
-------------------------------------------------------
SALDYTUVAI
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Snaige    , A980   , A+      , Balta     , 900  , 85   , Pastatomas     , True    , 15    , 5    , 8
Siemens   , C 10-25, A++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Samsung   , 10-75  , A       , Balta     , 700  , 80   , Pastatomas     , True    , 15    , 8    , 6
LG        , A900   , B       , Balta     , 500  , 60   , Pastatomas     , False   , 1     , 1    , 7
Siemens   , 100B   , A+++    , Raudona   , 2000 , 100  , Pastatomas     , True    , 2     , 2    , 18
Siemens   , 20A    , A+      , Balta     , 500  , 85   , Pastatomas     , True    , 2     , 1    , 0.5

MIKROBANGES
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , A++       , Balta  , 200  , 220      , 6
Samsung   , 2120   , B         , Juoda  , 150  , 230      , 5

VIRDULIAI
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , A++     , Juoda  , 40 , 220  , 5
Siemens   , 22B      , A       , Melyna , 50 , 220  , 6

-------------------------------------------------------
Jono Uzkampis Opiolio g. 30 869119112
-------------------------------------------------------
SALDYTUVAI
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Siemens   , C 10-25, A++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Snaige    , A980   , A+      , Balta     , 900  , 75   , Pastatomas     , True    , 15    , 5    , 8
LG        , 10G    , B       , Balta     , 100  , 40   , Kabinamas      , False   , 3     , 2    , 2
Siemens   , K21    , A       , Juoda     , 300  , 80   , Pastatomas     , True    , 2     , 2    , 1

MIKROBANGES
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , A++       , Balta  , 200  , 220      , 6

VIRDULIAI
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , A++     , Juoda  , 40 , 220  , 5

-------------------------------------------------------
Varlyte Torjos g. 12 8625124521
-------------------------------------------------------
SALDYTUVAI
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Siemens   , 100B   , A+++    , Raudona   , 2000 , 100  , Pastatomas     , True    , 2     , 2    , 18
Snaige    , A980   , A+      , Balta     , 900  , 85   , Pastatomas     , True    , 15    , 5    , 8
Siemens   , C 10-25, A++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Samsung   , W-112  , B       , Balta     , 600  , 60   , Sieninis       , False   , 1     , 1    , 8

MIKROBANGES
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , A++       , Balta  , 200  , 220      , 6

VIRDULIAI
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , A++     , Juoda  , 40 , 220  , 5
Siemens   , 10S      , A+      , Juoda  , 25 , 220  , 4
```

Konsoleje:

```
Topo Centras, Dauksio g. 3, 861251205
----------------------------------------
Siemens saldytuvu: 3
----------------------------------------
Siemens Mikrobangu K.: 1
----------------------------------------
Siemens Elektriniu V.: 1
----------------------------------------

Jono Uzkampis, Opiolio g. 30, 869119112
```

```
----------------------------------------
Siemens saldytuvu: 2
----------------------------------------
Siemens Mikrobangu K.: 1
----------------------------------------
Siemens Elektriniu V.: 0
----------------------------------------

Varlyte, Torjos g. 12, 8625124521
----------------------------------------
Siemens saldytuvu: 2
----------------------------------------
Siemens Mikrobangu K.: 1
----------------------------------------
Siemens Elektriniu V.: 1
----------------------------------------
```

Saldytuvai, kuriu talpa didesne uz 80 surusiuoti pagal kaina:

Gamintojas	Modelis	Talpa	Kaina
Siemens	K21	80	300
Siemens	20A	85	500
Samsung	10-75	80	700
Snaige	A980	85	900
Siemens	C 10-25	90	1200
Siemens	100B	100	2000

A+.csv

Gamintojas	Modelis	Energijos Klase	Spalva	Kaina					
Snaige	A980	A+	Balta	900	85	Pastatomas	True	15	5
Siemens	C 10-25	A++	Juoda	1200	90	Pastatomas	True	2	1
Siemens	100B	A+++	Raudona	2000	100	Pastatomas	True	2	2
Siemens	20A	A+	Balta	500	85	Pastatomas	True	2	1
Siemens	B-200	A++	Balta	200	220	6			
Keprel	20-1	A++	Juoda	40	220	5			
Siemens	10S	A+	Juoda	25	220	4			

TikTen.csv

Gamintojas	Modelis	Energijos Klase	Spalva	Kaina					
Samsung	10-75	A	Balta	700	80	Pastatomas	True	15	8
LG	A900	B	Balta	500	60	Pastatomas	False	1	1
Siemens	20A	A+	Balta	500	85	Pastatomas	True	2	1
Samsung	2120	B	Juoda	150	230	5			
Siemens	22B	A	Melyna	50	220	6			
LG	10G	B	Balta	100	40	Kabinamas	False	3	2
Siemens	K21	A	Juoda	300	80	Pastatomas	True	2	2

Siemens	100B	A+++	Raudona	2000	100	Pastatomas	True	2	2
Snaige	A980	A+	Balta	900	85	Pastatomas	True	15	5
Siemens	C 10-25	A++	Juoda	1200	90	Pastatomas	True	2	1
Samsung	W-112	B	Balta	600	60	Sieninis	False	1	1
Siemens	B-200	A++	Balta	200	220	6			
Keprel	20-1	A++	Juoda	40	220	5			
Siemens	10S	A+	Juoda	25	220	4			

Trečias bandymas:

Duomenys lentele:

```
-----------------------------------------------------------
Topo Centras Dauksio g. 3 861251205
-----------------------------------------------------------
Saldytuvai
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Snaige    , A980   , B+      , Balta     , 900  , 85   , Pastatomas     , True    , 15    , 5    , 8
Siemens   , C 10-25, C++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Samsung   , 10-75  , B       , Balta     , 700  , 80   , Pastatomas     , True    , 15    , 8    , 6
LG        , A900   , B       , Balta     , 500  , 60   , Pastatomas     , False   , 1     , 1    , 7
Siemens   , 100B   , C+++    , Raudona   , 2000 , 100  , Pastatomas     , True    , 2     , 2    , 18
Siemens   , 20A    , B+      , Balta     , 500  , 85   , Pastatomas     , True    , 2     , 1    , 0.5

Mikrobangu Krosneles
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , B++       , Balta  , 200  , 220      , 6
Samsung   , 2120   , B         , Juoda  , 150  , 230      , 5

Elektriniai Virduliai
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , C++     , Juoda  , 40 , 220  , 5
Siemens   , 22B      , C       , Melyna , 50 , 220  , 6

-----------------------------------------------------------
Jono Uzkampis Opiolio g. 30 869119112
-----------------------------------------------------------
Saldytuvai
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Siemens   , C 10-25, C++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Snaige    , A980   , B+      , Balta     , 900  , 75   , Pastatomas     , True    , 15    , 5    , 8
LG        , 10G    , B       , Balta     , 100  , 40   , Kabinamas      , False   , 3     , 2    , 2
Siemens   , K21    , C       , Juoda     , 300  , 80   , Pastatomas     , True    , 2     , 2    , 1

Mikrobangu Krosneles
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , C++       , Balta  , 200  , 220      , 6

Elektriniai Virduliai
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , C++     , Juoda  , 40 , 220  , 5

-----------------------------------------------------------
Varlyte Torjos g. 12 8625124521
-----------------------------------------------------------
Saldytuvai
Gamintojas Modelis  E. Tipas  Spalva     Kaina  Talpa   Montavimo Tipas Saldiklis Aukstis Plotis Gylis
Snaige    , A980   , B+      , Balta     , 900  , 85   , Pastatomas     , True    , 15    , 5    , 8
Siemens   , C 10-25, C++     , Juoda     , 1200 , 90   , Pastatomas     , True    , 2     , 1    , 18
Samsung   , 10-75  , B       , Balta     , 700  , 80   , Pastatomas     , True    , 15    , 8    , 6
LG        , A900   , B       , Balta     , 500  , 60   , Pastatomas     , False   , 1     , 1    , 7
Siemens   , 100B   , C+++    , Raudona   , 2000 , 100  , Pastatomas     , True    , 2     , 2    , 18
Siemens   , 20A    , B+      , Balta     , 500  , 85   , Pastatomas     , True    , 2     , 1    , 0.5
LG        , 10G    , B       , Balta     , 100  , 40   , Kabinamas      , False   , 3     , 2    , 2
Siemens   , K21    , C       , Juoda     , 300  , 80   , Pastatomas     , True    , 2     , 2    , 1

Mikrobangu Krosneles
Gamintojas Modelis   E.  Tipas  Spalva  Kaina  Galingumas Programu Sk.
Siemens   , B-200  , B++       , Balta  , 200  , 220      , 6
Samsung   , 2120   , B         , Juoda  , 150  , 230      , 5

Elektriniai Virduliai
Gamintojas Modelis   E.  Tipas  Spalva  Kaina Galia Turis
Keprel    , 20-1     , C++     , Juoda  , 40 , 220  , 5
Siemens   , 22B      , C       , Melyna , 50 , 220  , 6
```

Konsole:

```
Topo Centras, Dauksio g. 3, 861251205
---------------------------------------
Siemens saldytuvu: 3
---------------------------------------
Siemens Mikrobangu K.: 1
---------------------------------------
Siemens Elektriniu V.: 1
---------------------------------------

Jono Uzkampis, Opiolio g. 30, 869119112
---------------------------------------
Siemens saldytuvu: 2
---------------------------------------
Siemens Mikrobangu K.: 1
---------------------------------------
Siemens Elektriniu V.: 0
---------------------------------------

Varlyte, Torjos g. 12, 8625124521
---------------------------------------
Siemens saldytuvu: 4
---------------------------------------
Siemens Mikrobangu K.: 1
---------------------------------------
Siemens Elektriniu V.: 1
---------------------------------------

Saldytuvai, kuriu talpa didesne uz 80 surusiuoti pagal kaina:

Gamintojas Modelis   Talpa  Kaina
---------------------------------
Siemens    K21        80    300
---------------------------------
Siemens    20A        85    500
---------------------------------
Samsung    10-75      80    700
---------------------------------
Snaige     A980       85    900
---------------------------------
Siemens    C 10-25    90    1200
---------------------------------
Siemens    100B       100   2000
---------------------------------
```

A++.csv

Prietaisu nera

TikTen.csv

Prietaisu nera

3.4. Dėstytojo pastabos

- P8
- P5

4. Teksto analizė ir redagavimas

4.1. Darbo užduotis

U4-12. Pasikartojantys žodžiai

Tekstiniame faile Knyga.txt duotas tekstas sudarytas iš žodžių, atskirtų skyrikliais. Skyriklių aibė žinoma. Raskite ir spausdinkite faile Rodikliai.txt:

- Nurodytą kiekį dažniausiai pasikartojančių žodžių (ne daugiau nei 10 žodžių), surikiuotą pagal pasikartojimo skaičių mažėjimo tvarka, o kai pasikartojimų skaičius sutampa – pagal abėcėlę;
- ilgiausią sakinį (didžiausias žodžių kiekis), jo ilgį (simboliais ir žodžiais) ir vietą (sakinio pradžios eilutės numerį).

Reikia teksto žodžius sulygiuoti, kad kiekvienos eilutės kiekvienas žodis prasidėtų fiksuotoje toje pačioje pozicijoje. Galima įterpti tik minimalų būtiną tarpų skaičių. Galima šalinti kelis iš eilės einančius vienodus skyriklius, paliekant tik vieną jų atstovą. Įterpimo ir šalinimo taisykles taikome, siekdami gauti lygiuotą minimalų tekstą. Šalinimo taisyklės netaikome, jei nėra poreikio. Pradinio teksto eilutės ilgis neviršija 80 simbolių.

Spausdinkite faile ManoKnyga.txt pertvarkytą tekstą pagal tokias taisykles:

- kiekvienos eilutės pirmasis žodis turi prasidėti pozicijoje p1=1.
- antrasis kiekvienos eilutės žodis turi prasidėti minimalioje galimoje pozicijoje p2, tokioje, kad kiekvienos eilutės pirmasis žodis kartu su už jo esančiais skyrikliais baigiasi iki p2-2 arba p2-1.
- trečiasis kiekvienos eilutės žodis turi prasidėti minimalioje galimoje pozicijoje p3, tokioje, kad kiekvienos eilutės antrasis žodis kartu su už jo esančiais skyrikliais baigiasi iki p3-2 arba p3-1.
- ir t.t.

4.2. Programos tekstas

Sentence.cs

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

namespace U4_12
{
    class Sentence
    {
        public string Sentencee { get; set; }
        public int Beginning { get; set; }
        public int WordAmount { get; set; }

        /// <summary>
        /// adds a new sentence
        /// </summary>
        /// <param name="sentence">sentence in a string</param>
        /// <param name="beginning">the beginning of a sentence</param>
        /// <param name="wordAmount">the end of the sentence</param>
        public Sentence (string sentence, int beginning, int wordAmount)
        {
            Sentencee = sentence;
            Beginning = beginning;
            WordAmount = wordAmount;
        }
    }
}
```

SentenceContainer.cs

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

```
using System.Threading.Tasks;

namespace U4_12
{
    class SentenceContainer
    {
        private Sentence[] Sentences;
        public int Count { get; private set; }

        /// <summary>
        /// creates  new sentence container
        /// </summary>
        /// <param name="size">size of container</param>
        public SentenceContainer(int size)
        {
            Sentences = new Sentence[size];
            Count = 0;
        }
        /// <summary>
        /// adds a sentence to the container
        /// </summary>
        /// <param name="sentence">given sentence</param>
        public void AddSentence (Sentence sentence)
        {
            Sentences[Count++] = sentence;
        }
        /// <summary>
        /// adds a sentence to the given place
        /// </summary>
        /// <param name="sentence">the sentehce</param>
        /// <param name="index">the place in the container</param>
        public void AddSentence (Sentence sentence, int index)
        {
            Sentences[index] = sentence;
        }
        /// <summary>
        /// gets a sentence from the container
        /// </summary>
        /// <param name="index">index of sentence</param>
        /// <returns>the sentence</returns>
        public Sentence GetSentence (int index)
        {
            return Sentences[index];
        }
    }
}
```

Program.cs

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

namespace U4_12
{
    class Program
    {
        static void Main(string[] args)
        {
            Program p = new Program();
            const string Duom = "..\\..\\Knyga.txt";
            const string Rez = "..\\..\\Rodikliai.txt";
            const string Rez2 = "..\\..\\ManoKnyga.txt";
            var Sentences = new SentenceContainer(100);
            var lines = new List<string>();
            var words = new List<string>();
            char[] skyrikliai = { ' ', '.', ',', '!', '?', ':', ';', '(', ')', '\t' };
```

```
    p.FindSentences(Duom, ref Sentences);
    //p.PrintSentences(Rez, Sentences);
    p.Checking(Sentences, Rez);

    var UniqueW = p.Words(Sentences, skyrikliai);
    p.PrintDictionary(Rez, UniqueW);

    p.Processing(Duom, ref lines);
    //int LongestL = p.LongestLine(lines);
    var LongestW = p.AllWords(lines, skyrikliai, ref words);
    var NewLines = p.Task(lines, LongestW, skyrikliai);
    p.PrintList(NewLines, Rez2);
    Console.WriteLine();
}

/// <summary>
/// Prints a given list
/// </summary>
/// <param name="lines">a list of lines</param>
/// <param name="fn">file name</param>
void PrintList (List<string> lines, string fn)
{
    using (StreamWriter sw = new StreamWriter(@fn, false,
     Encoding.GetEncoding(1257)))
    {
        for (int i = 0; i < lines.Count; i++)
        {
            sw.WriteLine(lines[i]);
        }
    }
}

/// <summary>
/// Goes through every line and word
/// adds the amount of spaces needed to every word
/// </summary>
/// <param name="lines">a list of lines</param>
/// <param name="LongestW">the longest word</param>
/// <param name="seperators">word seperators</param>
/// <returns>a modified list with spaces</returns>
List <string> Task (List<string> lines, string LongestW,
    char[] seperators)
{
    var NewLines = new List<string>();
    int SymbolAmount = LongestW.Length;
    for (int i = 0; i < lines.Count; i++)
    {
        string newline = "";
        var line = lines[i];
        string[] zodziai = line.Split(seperators,
           StringSplitOptions.RemoveEmptyEntries);
        // eilutes zodziu masyvas
        for (int j = 0; j < zodziai.Length; j++)
        {
            var zodis = zodziai[j];
            newline +=zodis + new string(' ', SymbolAmount -
                  zodziai[j].Length);

        }
        NewLines.Add(newline);
    }
    return NewLines;
}

/// <summary>
/// Finds the longest word and returns it
/// </summary>
/// <param name="words">a list of words</param>
/// <returns>longest word</returns>
```

```
string LongestWo (List<string> words)
{
    string LongWord = words[0];
    int LongWordsInd = 0;
    for (int i = 0; i < words.Count; i++)
    {
        if (words[i] == null)
            continue;
        if(words[i].Length >= LongWord.Length)
        {
            LongWord = words[i];
            LongWordsInd = i;
        }
    }
    return words[LongWordsInd];
}

/// <summary>
/// finds all of the words and gets the longes one
/// </summary>
/// <param name="lines">a list of lines</param>
/// <param name="seperators">word seperators</param>
/// <param name="words">a list of words</param>
/// <returns>the longest word</returns>
string AllWords(List <string> lines, char[] seperators,
    ref List<string> words)
{
    char ch;
    string word = null;
    for (int i = 0; i < lines.Count; i++) // eina per kiekviena eilute
    {
        var line = lines[i];
        for (int j = 0; j < line.Length; j++) //eina per vienos eilutes
           simbolius
        {
            ch = line[j];
            if (seperators.Contains(ch))
            {
                words.Add(word);
                word = null;
            }
            else
            word += ch;
        }
    }
    var LongestWord = LongestWo(words);
    return LongestWord;
}

/// <summary>
/// finds the longest line
/// </summary>
/// <param name="lines">a list of lines</param>
/// <returns>the longest line's index</returns>
int LongestLine (List<string> lines)
{
    string LongestL = lines[0];
    int LongestLIndex = 0;
    for (int i = 0; i < lines.Count; i++)
    {
        if(lines[i].Length > LongestL.Length)
        {
            LongestL = lines[i];
            LongestLIndex = i;
        }
    }
    return LongestLIndex;
}

/// <summary>
```

```
/// makes a list of lines in the text
/// </summary>
/// <param name="fn">file name</param>
/// <param name="lines">a list of lines</param>
void Processing (string fn, ref List<string> lines)
{
    using (StreamReader sr = new StreamReader(@fn, Encoding.GetEncoding(1257)))
    {
        while (!sr.EndOfStream)
        {
            var line = sr.ReadLine();
            lines.Add(line);
        }
    }
}

/// <summary>
/// Prints unique words and the amount of them in the text
/// </summary>
/// <param name="fn">file name</param>
/// <param name="words">dictionary of the unique words</param>
void PrintDictionary (string fn, Dictionary <string, int> words)
{
    var wordsS = from entry in words orderby entry.Value descending select
     entry;
    using (StreamWriter sw = new StreamWriter(@fn, true,
     Encoding.GetEncoding(1257)))
    {
        int j = 0;
        sw.WriteLine("Word    | Amount ");
        foreach (var entry in wordsS)
        {
            if (j == 10)
                break;

            sw.WriteLine(entry.Key + " | " + entry.Value + "|");
            sw.WriteLine("----------");
            j++;
        }
    }
}

/// <summary>
/// Prints the longest sentence, it's amount of words,
/// symbols and whitch line it starts in
/// </summary>
/// <param name="sentences">sentence container</param>
/// <param name="longestSIndex">longest sentence index</param>
/// <param name="fn">file name</param>
void PrintLongestS (SentenceContainer sentences, int longestSIndex, string fn)
{
    using (StreamWriter sr = new StreamWriter(@fn, false,
     Encoding.GetEncoding(1257)))
    {
        var lSentence = sentences.GetSentence(longestSIndex);
        sr.WriteLine("Longest sentence: ");
        sr.WriteLine(lSentence.Sentencee);
        sr.WriteLine("Amount of symbols: {0} ",
            lSentence.Sentencee.Length);
        sr.WriteLine("Amount of words: {0} ",
            lSentence.WordAmount);
        sr.WriteLine("The sentence begins in line {0}",
            lSentence.Beginning);
        sr.WriteLine();
    }
}

/// <summary>
/// finds the longest sentence in the sentence container
/// </summary>
```

```
/// <param name="sentences">sentence container</param>
/// <returns>the index of the longest sentence</returns>
int LongestS (SentenceContainer sentences)
{
    int LongestSen = sentences.GetSentence(0).WordAmount;
    int LongestSenIndex = 0;
    for (int i = 0; i < sentences.Count; i++)
    {
        if(sentences.GetSentence(i).WordAmount> LongestSen)
        {
            LongestSen = sentences.GetSentence(i).WordAmount;
            LongestSenIndex = i;
        }
    }
    return LongestSenIndex;
}

/// <summary>
/// Reads file char by char. Builds sentences
/// and puts them into a sentence container
/// </summary>
/// <param name="fn">file name</param>
/// <param name="sentences">sentence container</param>
public void FindSentences (string fn, ref SentenceContainer sentences)
{
    using (StreamReader sr = new StreamReader(@fn, Encoding.GetEncoding(1257)))
    {
        char[] sentenceEnd = { '.', '?', '!' };
        char[] seperators = { ' ', '-', ',', ':', ';', '(', ')', '\t' };
        string sentence = null;
        int eilute = 0;
        int eil = 0;
        while (!sr.EndOfStream)
        {
            char ch = (char)sr.Read();
            sentence += ch;
            if(sentenceEnd.Contains(ch))
            {
                if (sentences.Count == 0)
                {
                    eilute = EnterAmount(eilute, sentence);
                    eil = eilute;
                    var wordA = WordAmount(sentence, seperators);
                    //Eliminuojam naujos eilutes simboli
                    sentence = sentence.Replace(System.Environment.NewLine,
                        "").Trim(seperators);
                    var SE = new Sentence(sentence, 0, wordA);
                    sentences.AddSentence(SE);
                    sentence = null;
                }
                else
                {
                    eilute = EnterAmount(eilute, sentence);
                    var wordA = WordAmount(sentence, seperators);
                    //Eliminuojam naujos eilutes simboli
                    sentence = sentence.Replace(System.Environment.NewLine,
                        "").Trim(seperators);
                    var SE = new Sentence(sentence, eil + 1, wordA);
                    sentences.AddSentence(SE);
                    eil = eilute;
                    sentence = null;
                }
            }
        }
        //Checking(sentences);
    }

}

/// <summary>
```

```
/// checks if there are any sentences in the file
/// </summary>
/// <param name="Sentences">sentence container</param>
/// <param name="fn">file name</param>
void Checking (SentenceContainer Sentences, string fn)
{
    int LongestSenIndex = 0;
    if (Sentences.Count != 0)
    {
        LongestSenIndex = LongestS(Sentences);
        PrintLongestS(Sentences, LongestSenIndex, fn);
    }
    else
        Console.WriteLine("There are no sentences");
}

/// <summary>
/// Calculates the amount of \n used in the given string
/// </summary>
/// <param name="eilute">a line</param>
/// <param name="sentence">sentence</param>
/// <returns>amount of \n</returns>
public int EnterAmount (int eilute, string sentence)
{
    for (int i = 0; i < sentence.Length; i++)
    {
        if (sentence[i] == '\n')
            eilute++;
    }
    return eilute;
}

/// <summary>
/// Calculates and returns the amount of words that are in
/// the given sentence
/// </summary>
/// <param name="sentence">sentehce in a string</param>
/// <param name="seperators">word sepetators</param>
/// <returns>amount of words in a sentence</returns>
public int WordAmount (string sentence, char[] seperators)
{
    int zodziuK = 0;
    for (int i = 0; i < sentence.Length; i++)
    {
        if (seperators.Contains(sentence[i]))
            zodziuK++;
    }
    return zodziuK;
}

/// <summary>
/// Searches for words. If the word is unique, adds it to the dictionary
/// otherwise, ads 1 to it's value
/// </summary>
/// <param name="sentences">gives a sentence</param>
/// <param name="seperators">word seperators</param>
/// <returns>dictionary of words</returns>
public Dictionary<string, int> Words(SentenceContainer sentences, char[]
     seperators)
{
    var ComWords = new Dictionary<string, int>();
    for (int i = 0; i < sentences.Count; i++)
    {
        var sentence = sentences.GetSentence(i).Sentencee;
        string[] words = sentence.Split(seperators,
           StringSplitOptions.RemoveEmptyEntries);
        for (int j = 0; j < words.Length; j++)
        {
            var word = words[j];
            if (!ComWords.ContainsKey(word))
```

```
                    {
                        ComWords.Add(word, 1);
                    }
                    else
                    {
                        ComWords.TryGetValue(word, out int value);
                        value++;
                        ComWords[word] = value++;
                    }
                }
            }
            return ComWords;
        }

    }
}
```

4.3. Pradiniai duomenys ir rezultatai

Knyga.txt

```
Sakinys yra gramatiškai susijusių žodžių grupė.
Laimingas žmogus valandų neskaito
Sakinį gali. sudaryti ir vienas žodis:
Vakaras Vėsu Palaukit
Sakiniai į vientisinius ir sudėtinius skirstomi pagal gramatinių centrų skaičių.
Sakiniai, turintys du ir daugiau gramatinių centrų, yra sudėtiniai.
Malūno užtvankoje vienodai šniokščia vanduo, už miestelio pukši traktorius.
```

Rodikliai.txt

```
Longest sentence:
sudaryti ir vienas þodis: Vakaras Vësu Palaukit Sakiniai á vientisinius ir sudëtinius
skirstomi pagal gramatiniø centrø skaièiø.
Amount of symbols: 128
Amount of words: 18
The sentence begins in line 3

Word           | Amount |
ir             |3       |
----------
yra            |2       |
----------
Sakiniai       |2       |
----------
gramatiniø     |2       |
----------
centrø         |2       |
----------
Sakinys        |1       |
----------
gramatiðkai    |1       |
----------
susijusiø      |1       |
----------
þodþiø         |1       |
----------
grupë          |1       |
----------
```

ManoKnyga.txt

```
Sakinys       yra           gramatiðkai   susijusiø     þodþiø        grupë
Laimingas     þmogus        valandø       neskaito
Sakiná        gali          sudaryti      ir            vienas        þodis
Vakaras       Vësu          Palaukit
Sakiniai      á             vientisinius  ir            sudëtinius    skirstomi     pagal         gramatiniø    centrø
skaièiø
```

```
Sakiniai      turintys      du            ir            daugiau       gramatiniø    centrø        yra
sudëtiniai
Malûno        uþtvankoje    vienodai      ðniokðèia     vanduo        uþ            miestelio     pukði
traktorius
```

Antras bandymas:

Knyga.txt

```
Naujausias eksponatas milžiniškas atlantinis paltusas,
antras pagal dydį iš visų pasaulyje sugautų šios rūšies žuvų.
Šį paltusą muziejui padovanojo Norvegijos jūroje jį sugavęs lietuvis žvejas. Net trejus
metus triūsė patyrę muziejaus taksidermijos meistrai, kol padarė iškamšą muziejaus.
Viena svarbiausių permainų muziejuje ne tik gausesnis
eksponatų skaičius, bet ir pakeistos bei modernizuotos
ekspozicinės erdvės, padidėję muziejaus jų plotai, šiuolaikiškos vitrinos
eksponatams. Šiems darbams, kaip sakė muziejaus direktorius
Ramūnas Grigonis, išleista daugiau muziejaus kaip 400 tūkst. eurų.
```

Rodikliai.txt

```
Longest sentence:
Viena svarbiausiø permainø muziejuje ne tik gausesniseksponatø skaièius, bet ir
pakeistos bei modernizuotosekspozicinës erdvës, padidëjæ muziejaus jø plotai,
ðiuolaikiðkos vitrinoseksponatams.
Amount of symbols: 192
Amount of words: 22
The sentence begins in line 4

Word          | Amount |
muziejaus     |5       |
----------
kaip          |2       |
----------
Naujausias    |1       |
----------
eksponatas    |1       |
----------
milþiniðkas   |1       |
----------
atlantinis    |1       |
----------
paltusas      |1       |
----------
antras        |1       |
----------
pagal         |1       |
----------
dydá          |1       |
----------
```

4.4. Dėstytojo pastabos

- P5
- P13
- P11

5. Polimorfizmas

5.1. Darbo užduotis

U5_11. Juvelyrikos parduotuvė. Turite informaciją apie skirtingose juvelyrikos parduotuvėse esančius žiedus. Pirmoje eilutėje yra pavadinimas, antroje – adresas, trečioje – telefonas. Parduotuvėje galima įsigyti žiedų, auskarų, grandinėlių. Sukurkite abstrakčią klasę „Juvelyrinis“ (laukai - gamintojas, pavadinimas, metalas, svoris, praba, kaina), kurią paveldės “Žiedas” (papildomas laukas – dydis), “Auskarai” (papildomas laukas – užsegimo tipas) ir “Grandinėlė” (papildomas laukas – ilgis).

- Raskite ir atspausdinkite ekrane, kurioje parduotuvėje yra parduodamas brangiausias žiedas, brangiausi auskarai ir brangiausia grandinėlė.
- Ar yra tokių juvelyrinių gaminių, kurių galima įsigyti tik vienoje juvelyrinėje parduotuvėje? Atspausdinkite jų ir parduotuvių sąrašą faile „Unikalūs.csv“.
- Sudarykite juvelyrinių dirbinių, pigesnių nei 300 eurų, sąrašą. Visus duomenis apie juos įrašykite į failą „300.csv“.
- Sudarykite ir surikiuokite brangių juvelyrinių dirbinių sąrašą, pateikdami pilną informaciją apie juos. Žiedas yra brangus, jei jo kaina viršija 500€. Auskarai yra brangūs, jei jų kaina viršija 300€. Grandinėlė yra brangi, jei jos kaina viršija 150€ Žiedus rikiuokite pagal dydį, auskarus – pagal svorį, o grandinėles – pagal ilgį. Rezultatus įrašykite į failą „Brangus.csv“.

5.2. Programos tekstas

BranchuKonteineris.cs

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

namespace Darbas
{
    class BranchuKonteineris
    {
        private Branch[] branches;
        public int Count { get; private set; }

        /// <summary>
        /// suuria nauja branchu konteineri
        /// </summary>
        /// <param name="size">konteinerio dydis</param>
        public BranchuKonteineris(int size)
        {
            branches = new Branch[size];
            Count = 0;
        }
        /// <summary>
        /// Prideti issisakojima i konteineri
        /// </summary>
        /// <param name="branch">issisakojimo informacija</param>
        public void PridetiBrancha(Branch branch)
        {
            branches[Count++] = branch;
        }
        /// <summary>
        /// prideti issisakojima i nurodyta vieta
        /// </summary>
        /// <param name="branch">issisakojimo informacija</param>
        /// <param name="index">vieta konteineryje</param>
        public void PridetiBrancha(Branch branch, int index)
        {
            branches[index] = branch;
        }
        /// <summary>
        /// gauti issisakojima padavus jo vieta konteineryje
        /// </summary>
```

```
        /// <param name="index">reikeiamos parduotuves index
        /// konteineryje</param>
        /// <returns>issisakojimo (parduotuves) informacija</returns>
        public Branch GautiBrancha(int index)
        {
            return branches[index];
        }


    }
}
```

Branch.cs

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

namespace Darbas
{
    class Branch
    {
        public string Pavadinimas { get; set; }
        public string Adresas { get; set; }
        public string Telefonas { get; set; }
        //private Irasas[] Irasai;
        private GaminioKonteineris Gaminiai;
        public int Count { get; private set; }

        public string Tekstas { get; set; }

        /// <summary>
        /// prieti branch su parduotuves pavadinimu,
        /// adresu telefonu ir papuosalu konteineriu
        /// </summary>
        /// <param name="pavadinimas">parduotuves pavadinimas</param>
        /// <param name="adresas">parduotuves adresas</param>
        /// <param name="telefonas">parduotues telefono nr.</param>
        public Branch(string pavadinimas, string adresas, string telefonas)
        {
            Pavadinimas = pavadinimas;
            Adresas = adresas;
            Telefonas = telefonas;
            //Irasai = new Irasas[Program.MaxNrOfBranches];
            Gaminiai = new GaminioKonteineris();
        }

        /// <summary>
        /// prideti branch su tekstu ir konteineiu
        /// </summary>
        /// <param name="tekstas">brancho tekstas</param>
        public Branch (string tekstas)
        {
            Tekstas = tekstas;
            Gaminiai = new GaminioKonteineris();
        }

        /// <summary>
        /// prideti gamini
        /// Metodas kreipiasi i klasiu
        /// sub-metodus ir ten juos isskirsto
        /// </summary>
        /// <param name="gaminys">gaminio informacija</param>
        public void PridetiGamini(JuvelyrinisGaminys gaminys)
        {
            //Irasai[Count] = irasas;
            Gaminiai.Prideti(gaminys);
            Count++;
        }
```

```
/// <summary>
/// gauna gamini pagal indexa
/// </summary>
/// <param name="index">gaminio vieta konteineryje</param>
/// <returns>grazina indexo gamini</returns>
public JuvelyrinisGaminys GautiGamini(int index)
{
    //return Irasai[index];
    return Gaminiai.GautiGamini(index);
}

/// <summary>
/// iesko pigiu gaminiu,
/// juos ideda i branch klase
/// </summary>
/// <param name="tekstas">branch tekstas<param>
/// <returns>sukurta branch su pigiais gaminiais</returns>
public Branch PigusGaminiai (string tekstas)
{
    var b = new Branch(tekstas);
    for (int i = 0; i < Count; i++)
    {
        if(Gaminiai.GautiGamini(i).Kaina <= 300)
        {
            b.PridetiGamini(Gaminiai.GautiGamini(i));
        }
    }
    return b;
}

/// <summary>
/// sukurti branch klase be teksto
/// </summary>
public Branch ()
{
    Gaminiai = new GaminioKonteineris();
}

/// <summary>
/// grazina parduotuves ziedu konteineri
/// </summary>
/// <returns>ziedu konteineris</returns>
public GaminioKonteineris Ziedai()
{
    GaminioKonteineris a = new GaminioKonteineris();
    for (int i = 0; i < Count; i++)
    {
        if((GautiGamini(i) is Žiedas) && (!a.Contains(GautiGamini(i))))
        {
            a.Prideti(GautiGamini(i) as Žiedas);
        }
    }
    return a;
}

/// <summary>
/// grazina parduotuves auskaru konteineri
/// </summary>
/// <returns>auskaru konteineris</returns>
public GaminioKonteineris Auskarai()
{
    GaminioKonteineris a = new GaminioKonteineris();
    for (int i = 0; i < Count; i++)
    {
        if ((GautiGamini(i) is Auskarai) && (!a.Contains(GautiGamini(i))))
        {
            a.Prideti(GautiGamini(i) as Auskarai);
        }
    }
```

```
    return a;
}

/// <summary>
/// grazina parduotuves grandineliu konteineri
/// </summary>
/// <returns>grandineiu konteineris</returns>
public GaminioKonteineris Grandinele()
{
    GaminioKonteineris a = new GaminioKonteineris();
    for (int i = 0; i < Count; i++)
    {
        if ((GautiGamini(i) is Grandinėlė) && (!a.Contains(GautiGamini(i))))
        {
            a.Prideti(GautiGamini(i) as Grandinėlė);
        }
    }
    return a;
}

/// <summary>
/// randa brangiausia papuosala parduotuveje
/// lygina su kitais
/// </summary>
/// <returns>brangiausia gamini</returns>
public JuvelyrinisGaminys GautiBrangiausia ()
{
    double maxKaina = this.GautiGamini(0).Kaina;
    JuvelyrinisGaminys a = this.GautiGamini(0);
    for (int i = 0; i < Count; i++)
    {
        if(this.GautiGamini(i).Kaina > maxKaina)
        {
            maxKaina = this.GautiGamini(i).Kaina;
            a = this.GautiGamini(i);
        }
    }
    return a;
}

/// <summary>
/// randa parduotuves unikalius produktus
/// skaiciuoja produkto pasikartojimu sk
/// </summary>
/// <param name="ba">visu parduotuviu informacija</param>
/// <returns>parduotuves unkalius produktus</returns>
public GaminioKonteineris Unikalus(BranchuKonteineris ba)
{
    GaminioKonteineris naujas = new GaminioKonteineris();
    for (int j = 0; j < Count; j++) // eina per specifinio brancho juvelyrikos
     konteineri
    {
        int counter = 0;
        for (int u = 0; u < ba.Count; u++) //eina per kitus branchus
        {
            for (int i = 0; i < ba.GautiBrancha(u).Count; i++) // eina per ju
                 juvelyrikos konteineri
            {
                if(this.GautiGamini(j) == ba.GautiBrancha(u).GautiGamini(i))
                 //jei randa toki pati produkta, prideda prie counter 1
                {
                    counter++;
                }
            }
        }
        if(counter == 1) // jei tikrinimo pabaigoje produkto pasikartojimo
           skaicius yra 1, jis yra unikalus
        {
            naujas.Prideti(this.GautiGamini(j));
        }
```

```
            }
            return naujas;
        }

        /// <summary>
        /// uzklojimas norint sudeti du branchus
        /// </summary>
        /// <param name="a">kairysis branch</param>
        /// <param name="b">desinysis branch</param>
        /// <returns>sudetus branches</returns>
        public static Branch operator +(Branch a, Branch b)
        {
            Branch c = new Branch();
            for (int i = 0; i < a.Count; i++)
            {
                c.PridetiGamini(a.GautiGamini(i));
            }
            for (int i = 0; i < b.Count; i++)
            {
                c.PridetiGamini(b.GautiGamini(i));
            }
            return c;
        }
    }
}
```

JuvelyrinisGaminys.cs

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

namespace Darbas
{
    abstract class JuvelyrinisGaminys
    {
        public string Gamintojas { get; set; }
        public string Pavadinimas { get; set; }
        public string Metalas { get; set; }
        public double Svoris { get; set; }
        public int Praba { get; set; }
        public double Kaina { get; set; }
        /// <summary>
        /// Prilyginame kintamuosius
        /// </summary>
        /// <param name="gamintojas">Gaminio gamintojas</param>
        /// <param name="pavadinimas">Gaminio pavadinimas</param>
        /// <param name="metalas">Gaminio metalas</param>
        /// <param name="svoris">Gaminio svoris</param>
        /// <param name="praba">Gaminio praba</param>
        /// <param name="kaina">Gaminio kaina</param>
        public JuvelyrinisGaminys(string gamintojas, string pavadinimas, string
             metalas, double svoris, int praba, double kaina)
        {
            Gamintojas = gamintojas;
            Pavadinimas = pavadinimas;
            Metalas = metalas;
            Svoris = svoris;
            Praba = praba;
            Kaina = kaina;
        }

        //public JuvelyrinisGaminys GautiPigu ()
        //{
        //    if this.Kaina <= 300;
        //}

        /// <summary>
```

```
/// Kreipiasi i SetData metoda
/// </summary>
/// <param name="data">eilute su duomenim</param>
public JuvelyrinisGaminys (string data)
{
    SetData(data);
}

/// <summary>
/// paskirsto paduota eilute i elementus
/// </summary>
/// <param name="line">informacijos eilute</param>
public virtual void SetData (string line)
{
    string[] values = line.Split(',');
    Gamintojas = values[1];
    Pavadinimas = values[2];
    Metalas = values[3];
    Svoris = double.Parse(values[4]);
    Praba = int.Parse(values[5]);
    Kaina = double.Parse(values[6]);
}

/// <summary>
/// Metodas, naudojamas rasyti gamini i csv faila
/// </summary>
/// <returns>suformatuota gaminio
/// informacija csv failui</returns>
public virtual string ICsvFaila ()
{
    return String.Format("{0,-12}, {1,-16}, {2,-17}, {3,-14}, {4,-10}, {5,-
     10},", Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina);
}

/// <summary>
/// ToString uzklojimas
/// </summary>
/// <returns>suformatuota gaminio informacija</returns>
public override string ToString ()
{
    return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10}",
     Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina);
}

public abstract bool SortByKey(JuvelyrinisGaminys gaminys);

/// <summary>
/// Sulyginimo metodas
/// </summary>
/// <param name="obj">Objektas</param>
/// <returns>True arba false reikšme</returns>
public override bool Equals(object obj)
{
    return this.Equals(obj as JuvelyrinisGaminys);
}
/// <summary>
/// Sulyginimo metodas
/// </summary>
/// <param name="juvelyrinis">Gaminys</param>
/// <returns>True arba false reikšme</returns>
public bool Equals(JuvelyrinisGaminys juvelyrinis)
{
    if (Object.ReferenceEquals(juvelyrinis, null))
    {
        return false;
    }
    //if (this.GetType() != juvelyrinis.GetType())
    //{
    //    return false;
    //}
```

```
            return (Gamintojas == juvelyrinis.Gamintojas) && (Pavadinimas ==
             juvelyrinis.Pavadinimas) && (Metalas == juvelyrinis.Metalas);
        }
        /// <summary>
        /// Užklojamas hashcode metodas
        /// </summary>
        /// <returns></returns>
        public override int GetHashCode()
        {
            return Gamintojas.GetHashCode() ^ Pavadinimas.GetHashCode() ^
             Metalas.GetHashCode();
        }

        public static bool operator ==(JuvelyrinisGaminys lhs, JuvelyrinisGaminys rhs)
        {
            if (Object.ReferenceEquals(lhs, null))
            {
                if (Object.ReferenceEquals(rhs, null))
                {
                    return true;
                }
                return false;
            }
            return lhs.Equals(rhs);
        }

        public static bool operator !=(JuvelyrinisGaminys lhs, JuvelyrinisGaminys rhs)
        {
            return !(lhs == rhs);
        }

        /// <summary>
        /// grazina antraste
        /// </summary>
        /// <returns>suformuota antraste</returns>
        public virtual string Header()
        {
            return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10}",
             "Gamintojas", "Pavadinimas", "Metalas", "Svoris", "Praba", "Kaina");
        }
    }
}
```

Žiedas.cs

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

namespace Darbas
{
    class Žiedas : JuvelyrinisGaminys
    {
        public double Dydis { get; set; }
        /// <summary>
        /// Žiedo papildoma savybė
        /// </summary>
        /// <param name="gamintojas">Gaminio gamintojas</param>
        /// <param name="pavadinimas">Gaminio pavadinimas</param>
        /// <param name="metalas">Gaminio metalas</param>
        /// <param name="svoris">Gaminio svoris</param>
        /// <param name="praba">Gaminio praba</param>
        /// <param name="kaina">Gaminio kaina</param>
        /// <param name="dydis">Žiedo dydis</param>
        public Žiedas (string gamintojas, string pavadinimas, string metalas, double
             svoris, int praba, double kaina, double dydis) :
            base (gamintojas, pavadinimas, metalas, svoris, praba, kaina)
        {
            Dydis = dydis;
```

```
}
/// <summary>
/// Užklojamas žiedo spausdinimas
/// </summary>
/// <returns>String formatas</returns>
public override string ToString()
{
    return base.ToString() + String.Format("{0, 5}", Dydis);
    //return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10}
     {6,-10}", Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina, Dydis);
}

/// <summary>
/// metodas, naudojamas spausdinimui i csv faila
/// </summary>
/// <returns>suforamatuota informacija csv failui</returns>
public override string ICsvFaila()
{
    return String.Format("{0,-12}, {1,-16}, {2,-17}, {3,-14}, {4,-7}, {5,-7},
     {6, 5}", Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina, Dydis);
    //return base.ICsvFaila(); + String.Format("{0, 5}", Dydis);
}

/// <summary>
/// antraste informacijai
/// </summary>
/// <returns>suformatuota informacija apie profukta</returns>
public override string Header()
{
    return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10} {6,
     5}", "Gamintojas", "Pavadinimas", "Metalas", "Svoris", "Praba", "Kaina",
     "Dydys");
    //return base.ToString() + String.Format("{0,-12}", "Dydis/lgis/Tipas");
}

/// <summary>
/// rusiuoja pagal dydi
/// </summary>
/// <param name="gaminys">gaminys ir jo info</param>
/// <returns>palyginta informacija</returns>
public override bool SortByKey(JuvelyrinisGaminys gaminys)
{
    //return this >= (animal as GuineaPig);
    return this.Dydis.CompareTo((gaminys as Žiedas).Dydis) >= 0;
}

/// <summary>
/// kreipiasi i skirstymo metoda
/// priima informacija
/// </summary>
/// <param name="data">eilute su informacija</param>
public Žiedas (string data) : base (data)
{
    SetData(data);
}

/// <summary>
/// isskirsto line i duomenis
/// </summary>
/// <param name="line">infomracija su eilute</param>
public override void SetData(string line)
{
    base.SetData(line);
    string[] values = line.Split(',');
    Dydis = double.Parse(values[7]);
}

/// <summary>
/// lyginimo uzklojimas hashcode
/// </summary>
```

```
        /// <param name="obj"></param>
        /// <returns></returns>
        public override bool Equals(object obj)
        {
            return this.Equals(obj as Žiedas);
        }
        public bool Equals(Žiedas ziedas)
        {
            return base.Equals(ziedas);
        }
        public override int GetHashCode()
        {
            return Pavadinimas.GetHashCode() ^ Pavadinimas.GetHashCode();
        }

        /// <summary>
        /// uzklojimas ==
        /// leidzia lyginti du klases elementus
        /// </summary>
        /// <param name="lhs">kairysis elementas</param>
        /// <param name="rhs">desinysis elementas</param>
        /// <returns></returns>
        public static bool operator ==(Žiedas lhs, Žiedas rhs)
        {
            if (Object.ReferenceEquals(lhs, null))
            {
                if (Object.ReferenceEquals(rhs, null))
                {
                    return true;
                }
                return false;
            }
            return lhs.Equals(rhs);
        }
        public static bool operator !=(Žiedas lhs, Žiedas rhs)
        {
            return !(lhs == rhs);
        }
    }
}
```

Auskarai.cs

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

namespace Darbas
{
    class Auskarai : JuvelyrinisGaminys
    {
        public string Tipas { get; set; }
        /// <summary>
        /// Auskarų papildoma savybė
        /// </summary>
        /// <param name="gamintojas">Gaminio gamintojas</param>
        /// <param name="pavadinimas">Gaminio pavadinimas</param>
        /// <param name="metalas">Gaminio metalas</param>
        /// <param name="svoris">Gaminio svoris</param>
        /// <param name="praba">Gaminio praba</param>
        /// <param name="kaina">Gaminio kaina</param>
        /// <param name="tipas">Auskarų užsisegimo tipas</param>
        public Auskarai (string gamintojas, string pavadinimas, string metalas, double
             svoris, int praba, double kaina, string tipas): base(gamintojas,
             pavadinimas, metalas, svoris, praba, kaina)
        {
            Tipas = tipas;
        }
```

```
/// <summary>
/// metodas skirtas spausdinimui i csv faila
/// </summary>
/// <returns>suformatuota informacija</returns>
public override string ICsvFaila()
{
    return String.Format("{0,-12}, {1,-16}, {2,-17}, {3,-14}, {4,-7}, {5,-7},
     {6, 5}", Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina, Tipas);
}

/// <summary>
/// rusiuoja pagal svori
/// </summary>
/// <param name="gaminys">lyginamas objektas</param>
/// <returns>true arba false</returns>
public override bool SortByKey(JuvelyrinisGaminys gaminys)
{
    //return this >= (animal as GuineaPig);
    return this.Svoris.CompareTo((gaminys as Auskarai).Svoris) >= 0;
}

/// <summary>
/// Užklojamas Auskarų spausdinimas
/// </summary>
/// <returns>Suformatuota informacija</returns>
public override string ToString()
{
    return base.ToString() + String.Format("{0, 5}", Tipas);
    //return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10}
     {6,-10}", Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina, Dydis);
}

/// <summary>
/// grazina antraste
/// </summary>
/// <returns>suformatuota antraste</returns>
public override string Header()
{
    return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10} {6,
     5}", "Gamintojas", "Pavadinimas", "Metalas", "Svoris", "Praba", "Kaina",
     "Tipas");
    //return base.ToString() + String.Format("{0,-12}", "Dydis/lgis/Tipas");
}

/// <summary>
/// priima informacija,
/// kreipiasi i skirstyma
/// </summary>
/// <param name="data">eilute su informacija</param>
public Auskarai(string data) : base (data)
{
    SetData(data);
}

/// <summary>
/// skirsto linija pagal tipa
/// </summary>
/// <param name="line">eilute su informacija</param>
public override void SetData(string line)
{
    base.SetData(line);
    string[] values = line.Split(',');
    Tipas = values[7];
}

/// <summary>
/// leidzia lyginti du klases objektus
/// </summary>
/// <param name="obj">lygina su obj</param>
```

```
        /// <returns>true arba false</returns>
        public override bool Equals(object obj)
        {
            return this.Equals(obj as Auskarai);
        }
        public bool Equals(Auskarai auskarai)
        {
            return base.Equals(auskarai);
        }
        public override int GetHashCode()
        {
            return Pavadinimas.GetHashCode() ^ Pavadinimas.GetHashCode();
        }

        /// <summary>
        /// leidzia lyginti du klases objektus
        /// </summary>
        /// <param name="lhs">kairysis elementas</param>
        /// <param name="rhs">desinysis elementas</param>
        /// <returns>ar lygu ar ne</returns>
        public static bool operator ==(Auskarai lhs, Auskarai rhs)
        {
            if (Object.ReferenceEquals(lhs, null))
            {
                if (Object.ReferenceEquals(rhs, null))
                {
                    return true;
                }
                return false;
            }
            return lhs.Equals(rhs);
        }
        public static bool operator !=(Auskarai lhs, Auskarai rhs)
        {
            return !(lhs == rhs);
        }
    }
}
```

Grandinėlė.cs

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

namespace Darbas
{
    class Grandinėlė : JuvelyrinisGaminys
    {
        public double Ilgis { get; set; }
        /// <summary>
        /// Grandinėlės papildoma savybė
        /// </summary>
        /// <param name="gamintojas">Gaminio gamintojas</param>
        /// <param name="pavadinimas">Gaminio pavadinimas</param>
        /// <param name="metalas">Gaminio metalas</param>
        /// <param name="svoris">Gaminio svoris</param>
        /// <param name="praba">Gaminio praba</param>
        /// <param name="kaina">Gaminio kaina</param>
        /// <param name="ilgis">Grandinėlės ilgis</param>
        public Grandinėlė(string gamintojas, string pavadinimas, string metalas, double
             svoris, int praba, double kaina, double ilgis) : base(gamintojas,
             pavadinimas, metalas, svoris, praba, kaina)
        {
            Ilgis = ilgis;
        }

        /// <summary>
        /// rusiuoja pagal ilgi
        /// </summary>
```

```
/// <param name="gaminys">gaminys ir jo informacija</param>
/// <returns>palyginta informacija</returns>
public override bool SortByKey(JuvelyrinisGaminys gaminys)
{
    //return this >= (animal as GuineaPig);
    return this.Ilgis.CompareTo((gaminys as Grandinėlė).Ilgis) >= 0;
}

/// <summary>
/// spausdinimui i csv faila
/// </summary>
/// <returns>suforamtuoja informacija csv failo spausdinimui</returns>
public override string ICsvFaila()
{
    return String.Format("{0,-12}, {1,-16}, {2,-17}, {3,-14}, {4,-7}, {5,-7},
     {6, 5}", Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina, Ilgis);
}

/// <summary>
/// to string metodas. atspausdina duomenis
/// </summary>
/// <returns>suformatuota informacija</returns>
public override string ToString()
{
    return base.ToString() + String.Format("{0, 5}", Ilgis);
    //return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10}
     {6,-10}", Gamintojas, Pavadinimas, Metalas, Svoris, Praba, Kaina, Dydis);
}

/// <summary>
/// kreipiantis gaunama antraste
/// </summary>
/// <returns>suformatuota antraste</returns>
public override string Header()
{
    return String.Format("{0,-12} {1,-16} {2,-17} {3,-14} {4,-10} {5,-10} {6,
     5}", "Gamintojas", "Pavadinimas", "Metalas", "Svoris", "Praba", "Kaina",
     "Ilgis");
    //return base.ToString() + String.Format("{0,-12}", "Dydis/lgis/Tipas");
}

/// <summary>
/// kreipiasi i skirstyma,
/// priima informacija
/// </summary>
/// <param name="data">eilute su informacija</param>
public Grandinėlė(string data) : base (data)
{
    SetData(data);
}

/// <summary>
/// Skirsto duomenis grandineles
/// </summary>
/// <param name="line">eilute su informacija</param>
public override void SetData(string line)
{
    base.SetData(line);
    string[] values = line.Split(',');
    Ilgis = double.Parse(values[7]);
}

/// <summary>
/// leidzia lyginti du klases objektus
/// </summary>
/// <param name="obj">objektas su kuriuo lyginam</param>
/// <returns>true arba false</returns>
public override bool Equals(object obj)
{
    return this.Equals(obj as Grandinėlė);
```

```
        }
        public bool Equals(Grandinėlė grandinele)
        {
            return base.Equals(grandinele);
        }
        public override int GetHashCode()
        {
            return Pavadinimas.GetHashCode() ^ Pavadinimas.GetHashCode();
        }

        /// <summary>
        /// leidzia lyginti du klases objektus
        /// </summary>
        /// <param name="lhs">kairysis elementas</param>
        /// <param name="rhs">desinysis elementas</param>
        /// <returns>ar lygu ar ne</returns>
        public static bool operator ==(Grandinėlė lhs, Grandinėlė rhs)
        {
            if (Object.ReferenceEquals(lhs, null))
            {
                if (Object.ReferenceEquals(rhs, null))
                {
                    return true;
                }
                return false;
            }
            return lhs.Equals(rhs);
        }
        public static bool operator !=(Grandinėlė lhs, Grandinėlė rhs)
        {
            return !(lhs == rhs);
        }
    }
}
```

Program.cs

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

namespace Darbas
{
    class Program
    {
        public const int MaxNrOfBranches = 20;
        static void Main(string[] args)
        {
            Program p = new Program();
            //ParduotuveKonteineris parduotuves =
                   p.SkaitytiDuomenis("..\\..\\Duomenys");
            //p.SpausdinimasLentele(parduotuves);
            //GaminioKonteineris gaminiai = p.AddAll(parduotuves);
            //JuvelyrinisGaminys BrangiausiasGaminys =
             p.RastiBrangiausiaGamini(gaminiai);
            //Console.WriteLine("Brangiausias gaminys: \n{0} {1} {2} {3} {4} {5} \n",
             BrangiausiasGaminys.Gamintojas, BrangiausiasGaminys.Pavadinimas,
             BrangiausiasGaminys.Metalas, BrangiausiasGaminys.Svoris,
             BrangiausiasGaminys.Praba,
            //    BrangiausiasGaminys.Kaina);
            //p.RastiBrangiausiusParduotuvesGaminiai(parduotuves);
            //p.Unikalus(gaminiai);
            //gaminiai.Rikiavimas();
            //GaminioKonteineris pigiausi = p.SkaiciuotiPigiausi(gaminiai, 10);
            //p.SpausdintiPigiausius(pigiausi);
            //Console.ReadKey();
```

```
        //------------------------------------------------------------------------
         ---------------------------------------

        const string DataDir = @"..\..\Data";
        const string Punktas2 = "Unikalus.csv";
        const string Punktas3 = "300.csv";
        const string Punktas4 = "Brangus.csv";
        const string Pradiniai = "PradiniaiDuomenys.txt";
        File.Delete(Punktas2);
        File.Delete(Punktas4);

        BranchuKonteineris branchai = p.BranchuKonteinerioSukurimas(DataDir);

        p.DuomenuSpausdinimas(Pradiniai, branchai);
        //var a  = p.GautiGaminius("yoyo", branchai.GautiBrancha(1), 'Z');
        p.Brangiausi(branchai);

        p.Unikalus(branchai, Punktas2);

        var Pigus = p.Pigus(branchai, "Juvelyrika uz 300 ir maziau");
        p.PigiausiuSpausdinimas(Pigus, Punktas3);

        p.BrangiJuvelyrika(branchai, Punktas4);

        Console.ReadLine();
    }

    /// <summary>
    /// Pradiniu duomenu skapsdinimas txt faile
    /// </summary>
    /// <param name="file">failo pavadinimas</param>
    /// <param name="ba">parduotuviu konteineris</param>
    public void DuomenuSpausdinimas(string file, BranchuKonteineris ba)
    {
        using (StreamWriter sw = new StreamWriter(file, false,
         Encoding.GetEncoding(1257)))
        {
            if (ba.Count == 0)
            {
                sw.WriteLine("Nera parduotuviu");
            }
            else
                for (int i = 0; i < ba.Count; i++)
                {
                    GaminioKonteineris Ziedai = new GaminioKonteineris();
                    GaminioKonteineris Auskarai = new GaminioKonteineris();
                    GaminioKonteineris Grandineles = new GaminioKonteineris();
                    string s = new string('-',
                     ba.GautiBrancha(i).GautiGamini(0).ToString().Length);
                    Paskirstymas(ba.GautiBrancha(i), ref Ziedai, ref Auskarai, ref
                     Grandineles);
                    if (ba.GautiBrancha(i) == null)
                        continue;
                    sw.WriteLine("{0} \n {1} \n {2}",
                     ba.GautiBrancha(i).Pavadinimas, ba.GautiBrancha(i).Adresas,
                     ba.GautiBrancha(i).Telefonas);
                    sw.WriteLine("Ziedai");
                    Print(sw, Ziedai);
                    sw.WriteLine("Auskarai");
                    Print(sw, Auskarai);
                    sw.WriteLine("Grandineles");
                    Print(sw, Grandineles);
                    sw.WriteLine();
                    sw.WriteLine();
                }
        }
    }

    /// <summary>
    /// spausdina konteinerius
```

```
/// </summary>
/// <param name="sw">streamwriter reikiamam faile</param>
/// <param name="konteineris">paduotas konteineris</param>
private void Print(StreamWriter sw, GaminioKonteineris konteineris)
{
    sw.WriteLine(konteineris.GautiGamini(0).Header());
    for (int i = 0; i < konteineris.Count; i++)
    {
        sw.WriteLine(konteineris.GautiGamini(i).ToString());
        string s = new string('-',
           konteineris.GautiGamini(0).ToString().Length);
        sw.WriteLine(s);
    }
    sw.WriteLine();
}

/// <summary>
/// Sudeda parduotuves papuosalus i konteineri
/// ir ji paskirsto i papuosalu tipus
/// </summary>
/// <param name="ba">vienos parduotuves info</param>
/// <param name="Ziedai">ziedu konteineris</param>
/// <param name="Auskarai">auskaru konteineris</param>
/// <param name="Grandineles">grandineliu konteineris</param>
private void Paskirstymas(Branch ba, ref GaminioKonteineris Ziedai, ref
     GaminioKonteineris Auskarai, ref GaminioKonteineris Grandineles)
{
    GaminioKonteineris visi = new GaminioKonteineris();
    for (int j = 0; j < ba.Count; j++)
    {
        if (!visi.Contains(ba.GautiGamini(j)))
            visi.Prideti(ba.GautiGamini(j));
    }
    AtrinktiPagalTipa(visi, typeof(Žiedas), ref Ziedai);
    AtrinktiPagalTipa(visi, typeof(Auskarai), ref Auskarai);
    AtrinktiPagalTipa(visi, typeof(Grandinėlė), ref Grandineles);
}

/// <summary>
/// Spausdina parduotuviu informacija
/// ir jos unikaliu produktu informacija
/// </summary>
/// <param name="unikalus">unikaliu gaminiu konteineris</param>
/// <param name="file">failo vardas</param>
/// <param name="pavadinimas">parduotuves pavadinimas</param>
/// <param name="adresas">parduotuves adresas</param>
/// <param name="telefonas">patduotuves telefono nr</param>
private void UnikaliuSpausdinimas(GaminioKonteineris unikalus, string file,
     string pavadinimas, string adresas, string telefonas)
{
    using (StreamWriter sw = new StreamWriter(file, true,
     Encoding.GetEncoding(1257)))
    {
        sw.WriteLine("{0} \n {1} \n {2}", pavadinimas, adresas, telefonas);
        if (unikalus.Count == 0)
        {
            sw.WriteLine("Parduotuve neturi unikaliu papuosalu.");
            sw.WriteLine();
        }
        else
        {
            sw.WriteLine("Gamintojas, Pavadinimas, Metalas, Svoris, Praba,
                 Kaina, Dydis/Ilgis/Uzsegimo tipas");
            for (int i = 0; i < unikalus.Count; i++)
            {
                sw.WriteLine(unikalus.GautiGamini(i).ToString());
            }
            sw.WriteLine();
        }
    }
```

```
}

/// <summary>
/// sukuria nauja gaminiu konteineri,
/// naudoja brancho metoda, randa unikalius papuosalus
/// </summary>
/// <param name="ba">visu parduotuviu info</param>
/// <param name="file">failo info</param>
private void Unikalus(BranchuKonteineris ba, string file)
{
    GaminioKonteineris unikalus = new GaminioKonteineris();
    for (int i = 0; i < ba.Count; i++)
    {
        unikalus = ba.GautiBrancha(i).Unikalus(ba);
        //if (unikalus.Count != 0)
        UnikaliuSpausdinimas(unikalus, file, ba.GautiBrancha(i).Pavadinimas,
           ba.GautiBrancha(i).Adresas, ba.GautiBrancha(i).Telefonas);
    }
}

/// <summary>
/// Sukuria atskirus konteinerius
/// brangiem ziedam, grandinelem ir auskaram
/// juos isrikiuoja
/// </summary>
/// <param name="ba">visu parduotuviu info</param>
/// <param name="file">failo vardas</param>
private void BrangiJuvelyrika(BranchuKonteineris ba, string file)
{
    GaminioKonteineris Ziedai = new GaminioKonteineris();
    GaminioKonteineris Auskarai = new GaminioKonteineris();
    GaminioKonteineris Grandineles = new GaminioKonteineris();
    GaminioKonteineris visi = new GaminioKonteineris();
    for (int i = 0; i < ba.Count; i++)
    {

        visi += ba.GautiBrancha(i).Ziedai().TikBrangus();
        visi += ba.GautiBrancha(i).Auskarai().TikBrangus();
        visi += ba.GautiBrancha(i).Grandinele().TikBrangus();
    }

    AtrinktiPagalTipa(visi, typeof(Žiedas), ref Ziedai);
    Ziedai.Rusiuoti();
    PrangiosJuvelyrikosSpausdinimas(Ziedai, file, "Brangus Ziedai:");

    AtrinktiPagalTipa(visi, typeof(Auskarai), ref Auskarai);
    Auskarai.Rusiuoti();
    PrangiosJuvelyrikosSpausdinimas(Auskarai, file, "Brangus Auskarai:");

    AtrinktiPagalTipa(visi, typeof(Grandinėlė), ref Grandineles);
    Grandineles.Rusiuoti();
    PrangiosJuvelyrikosSpausdinimas(Grandineles, file, "Brangios
     Grandineles:");

}

/// <summary>
/// Duomenu atrinkimas pagal tipa
/// </summary>
/// <param name="paduoti">paduotas pradinis konteineris</param>
/// <param name="a">reikiamas tipas</param>
/// <param name="naujas">sudarytas naujas konteineris</param>
private void AtrinktiPagalTipa(GaminioKonteineris paduoti, Type a, ref
     GaminioKonteineris naujas)
{
    for (int i = 0; i < paduoti.Count; i++)
    {
        if ((paduoti.GautiGamini(i).GetType() == a) &&
           (!naujas.Contains(paduoti.GautiGamini(i))))
            naujas.Prideti(paduoti.GautiGamini(i));
```

```
    }
}

/// <summary>
/// papildoma konteineri papildo esamo elementais
/// </summary>
/// <param name="esamas">imami elementai is sio konteinerio</param>
/// <param name="pildomas">pildomas konteineris</param>
private void Pridejimas(GaminioKonteineris esamas, ref GaminioKonteineris
    pildomas)
{
    for (int i = 0; i < esamas.Count; i++)
    {
        pildomas.Prideti(esamas.GautiGamini(i));
    }
}

/// <summary>
/// spausdina brangios juvelyrikos papuosalus
/// </summary>
/// <param name="juvelyrika">brangios juvelyrikos konteineris</param>
/// <param name="file">failo vardas</param>
/// <param name="antraste">antraste duomenims</param>
private void PrangiosJuvelyrikosSpausdinimas(GaminioKonteineris juvelyrika,
    string file, string antraste)
{
    using (StreamWriter sw = new StreamWriter(file, true,
    Encoding.GetEncoding(1257)))
    {
        sw.WriteLine("{0}", antraste);
        sw.WriteLine(juvelyrika.GautiGamini(0).Header());
        for (int i = 0; i < juvelyrika.Count; i++)
        {
            sw.WriteLine(juvelyrika.GautiGamini(i).ICsvFaila());
        }
        sw.WriteLine();
    }
}

/// <summary>
/// spausdina parduotuves, turincias
/// brangiausius papuosalus
/// </summary>
/// <param name="bk">visu parduotuviu informacija</param>
/// <param name="Ziedas">Parduotuves, turincios brangiausia zieda,
    index</param>
/// <param name="Auskarai">parduotuves, turincios brangiausius auskarus,
    index</param>
/// <param name="Grandinele">parduotuves, turincios brangiausia grandinle,
    index</param>
private void BrangiausiuSpausdinimas(BranchuKonteineris bk, int Ziedas, int
    Auskarai, int Grandinele)
{
    Console.WriteLine("Brangiausias ziedas yra {0} parduotuveje.",
    bk.GautiBrancha(Ziedas).Pavadinimas);
    Console.WriteLine("Brangiausi auskarai yra {0} parduotuveje.",
    bk.GautiBrancha(Auskarai).Pavadinimas);
    Console.WriteLine("Brangiausia grandinele yra {0} parduotuveje.",
    bk.GautiBrancha(Grandinele).Pavadinimas);
}

/// <summary>
/// Isrenka brangiausius brancho papuosalus
/// ir juos lygina su kitu branchu
/// branchiausiais papuosalais
/// </summary>
/// <param name="ba">visu parduotuviu info</param>
private void Brangiausi(BranchuKonteineris ba)
{
    int MaxZiedasI = 0;
```

```
        int MaxAuskaraiI = 0;
        int MaxGrandinelI = 0;
        double MaxZiedoK = 0;
        double MaxAuskaruK = 0;
        double MaxGrandinelesK = 0;
        for (int i = 0; i < ba.Count; i++)
        {
            MaxZiedasI = 0;
            MaxAuskaraiI = 0;
            MaxGrandinelI = 0;
            MaxZiedoK = 0;
            MaxAuskaruK = 0;
            MaxGrandinelesK = 0;

            var Ziedas = ba.GautiBrancha(i).Ziedai().GautiBrangiausia();
            var Auskarai = ba.GautiBrancha(i).Auskarai().GautiBrangiausia();
            var Grandinele = ba.GautiBrancha(i).Grandinele().GautiBrangiausia();

            Tikrinimas(ba, i, Ziedas, Auskarai, Grandinele, ref MaxZiedasI, ref
               MaxAuskaraiI, ref MaxGrandinelI, ref MaxZiedoK, ref MaxAuskaruK, ref
               MaxGrandinelesK);

        }
        BrangiausiuSpausdinimas(ba, MaxZiedasI, MaxAuskaraiI, MaxGrandinelI);
    }

    /// <summary>
    /// lygina paduotus papuosalus
    /// su kitu klasiu brangiausiais papuosalais
    /// </summary>
    /// <param name="ba">visu parduotuviu informacija</param>
    /// <param name="index">tikrinamos parduotuves indeksas</param>
    /// <param name="Ziedas">tikrinamos parduotuves brangiausias ziedas</param>
    /// <param name="Auskarai">tikrinamos parduotuves brangiausi auskarai</param>
    /// <param name="Grandinele">tikrinamos parduotuves bragiausia
         grandinele</param>
    /// <param name="MaxZiedasI">Brangiausio ziedo indeksas</param>
    /// <param name="MaxAuskaraiI">Brangiausiu auskaru indeksas</param>
    /// <param name="MaxGrandinelI">Brangiausios grandineles indeksas</param>
    /// <param name="MaxZiedoK">Brangiausio ziedo kaina</param>
    /// <param name="MaxAuskaruK">Brangiausiu auskaru kaina</param>
    /// <param name="MaxGrandinelesK">Brangiausios grandineles kaina</param>
    private void Tikrinimas(BranchuKonteineris ba, int index, JuvelyrinisGaminys
         Ziedas,
        JuvelyrinisGaminys Auskarai, JuvelyrinisGaminys Grandinele, ref int
         MaxZiedasI,
        ref int MaxAuskaraiI, ref int MaxGrandinelI, ref double MaxZiedoK,
        ref double MaxAuskaruK, ref double MaxGrandinelesK)
    {
        for (int j = 0; j < ba.Count; j++)
        {
            var Ziedas2 = ba.GautiBrancha(j).Ziedai().GautiBrangiausia();
            var Auskarai2 = ba.GautiBrancha(j).Auskarai().GautiBrangiausia();
            var Grandinele2 = ba.GautiBrancha(j).Grandinele().GautiBrangiausia();
            if ((Ziedas.Kaina > Ziedas2.Kaina) || (Ziedas != null) || (Ziedas2 !=
               null))
            {
                MaxZiedasI = index;
                MaxZiedoK = Ziedas.Kaina;
            }
            if ((Auskarai != null) || (Auskarai2 != null) || (Auskarai.Kaina >
               Auskarai2.Kaina))
            {
                MaxAuskaraiI = index;
                MaxAuskaruK = Auskarai.Kaina;
            }
            if ((Grandinele != null) || (Grandinele2 != null) || (Grandinele.Kaina
               > Grandinele2.Kaina))
            {
                MaxGrandinelI = index;
```

```
                MaxGrandinelesK = Grandinele.Kaina;
            }
        }
    }

    /// <summary>
    /// Sukuria branch'a su tekstu,
    /// kuriame kuriamas gaminiu konteineris
    /// su paposalais, pigesniais nei 300 eur
    /// </summary>
    /// <param name="ba">visu parduotuviu informacija</param>
    /// <param name="tekstas">tekstas, irasomas i nauja sukurta branch</param>
    /// <returns>grazina branch klase, pripildyta pigiu papuosalu</returns>
    private Branch Pigus(BranchuKonteineris ba, string tekstas)
    {
        Branch Pigus = new Branch(tekstas);
        for (int i = 0; i < ba.Count; i++)
        {
            Pigus += ba.GautiBrancha(i).PigusGaminiai(tekstas);
        }
        return Pigus;
    }

    /// <summary>
    /// spausdina i faila
    /// pigausius produktus ir ju informacija
    /// </summary>
    /// <param name="br">parduotuves duomenys</param>
    /// <param name="file">failo pavadinimas</param>
    private void PigiausiuSpausdinimas(Branch br, string file)
    {
        using (StreamWriter sw = new StreamWriter(file, false,
         Encoding.GetEncoding(1257)))
        {
            sw.WriteLine("Juvelyriniai gaminiai, pigesni nei 300 euru:");
            if (br.Count != 0)
                for (int i = 0; i < br.Count; i++)
                {
                    sw.WriteLine(br.GautiGamini(i).ICsvFaila());
                }
            else
                sw.WriteLine("Pigiu papuosalu nera.");
        }
    }

    /// <summary>
    /// Sukuria branchu konteineri, skaito duomenu failus
    /// </summary>
    /// <param name="file">failo pavadinimas</param>
    /// <returns>sukurta branhu konteineri</returns>
    private BranchuKonteineris BranchuKonteinerioSukurimas(string file)
    {
        string[] filePaths = Directory.GetFiles(file, "*.csv");
        var branchai = new BranchuKonteineris(MaxNrOfBranches);
        foreach (var path in filePaths)
        {
            Branch branch = null;
            bool rado = GaminiuDuomenuSkaitymas(path, ref branch, out string
               vardas, out string adresas, out string numeris);
            if (rado == false)
            {
                Console.WriteLine("Nera Parduotuves");
            }
            //SausdintiBrancha(branch);
            branchai.PridetiBrancha(branch);
        }
        return branchai;
    }

    /// <summary>
```

```
/// Eina per duomenu faila, kviecia branch klase, perduoda duomenis
/// ,kur jie suskirstomi
/// </summary>
/// <param name="path">failo vieta</param>
/// <param name="branch">parduotuves informacija</param>
/// <param name="pavadinimas">parduotuves pavadinimas</param>
/// <param name="adresas">parduotuves adresas</param>
/// <param name="telefonas">parduotuves telefono nr</param>
/// <returns></returns>
private bool GaminiuDuomenuSkaitymas(string path, ref Branch branch, out string
           pavadinimas, out string adresas, out string telefonas)
{
    using (StreamReader sr = new StreamReader(path))
    {
        string line = null;
        pavadinimas = sr.ReadLine();
        adresas = sr.ReadLine();
        telefonas = sr.ReadLine();
        line = pavadinimas;
        if (line == null)
            return false;
        branch = new Branch(pavadinimas, adresas, telefonas);
        while (null != (line = sr.ReadLine()))
        {
            switch (line[0])
            {
                case 'Z':
                    branch.PridetiGamini(new Žiedas(line));
                    break;
                case 'A':
                    branch.PridetiGamini(new Auskarai(line));
                    break;
                case 'G':
                    branch.PridetiGamini(new Grandinėlė(line));
                    break;
            }
        }
        return true;
    }
}

/// <summary>
/// Gauna gaminius pagal tipa
/// </summary>
/// <param name="tekstas">brancho tekstas</param>
/// <param name="ba">branch</param>
/// <param name="type">elemento tipas</param>
/// <returns>suurta nauja to elemento branch</returns>
private Branch GautiGaminius(string tekstas, Branch ba, char type)
{
    Branch Gaminiai = new Branch(tekstas);
    for (int i = 0; i < ba.Count; i++)
    {
        switch (type)
        {
            case 'Z':
            case 'z':
                if (ba.GautiGamini(i) is Žiedas)
                    Gaminiai.PridetiGamini(ba.GautiGamini(i));
                break;
            case 'A':
            case 'a':
                if (ba.GautiGamini(i) is Auskarai)
                    Gaminiai.PridetiGamini(ba.GautiGamini(i));
                break;
            case 'G':
            case 'g':
                if (ba.GautiGamini(i) is Grandinėlė)
                    Gaminiai.PridetiGamini(ba.GautiGamini(i));
                break;
```

```
                }
            }
            return Gaminiai;
        }
    }
}
```

5.3. Pradiniai duomenys ir rezultatai

Pirmas Bandymas.
PradiniaiDuomenys.txt

```
Adidas papuosalai
 Gatvynas
 825612581
Ziedai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Dydys
Fancy        Didysis          Auksas           0.1            20         600         0.2
-------------------------------------------------------------------------------------------
Nike         Stripe           Plienas          0.1            30         500         0.1
-------------------------------------------------------------------------------------------
Adidas       Speed            Sidabras         0.2            40         550         0.2
-------------------------------------------------------------------------------------------

Auskarai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Tipas
Asiai        Rubiai           Ruby             0.5            40         500       Kitoksai
-------------------------------------------------------------------------------------------

Grandineles
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Ilgis
Fine         Grandinine       Sidabras         0.1            14         350           2
-------------------------------------------------------------------------------------------
Fancy        Ilgoji           Sidabras         0.4            20         349         1.8
-------------------------------------------------------------------------------------------



Papuosalai jums
 Judesio g.
 862512512
Ziedai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Dydys
Fancy        Didysis          Auksas           0.1            20         600         0.2
-------------------------------------------------------------------------------------------
Adidas       Speed            Sidabras         0.2            40         200         0.2
-------------------------------------------------------------------------------------------

Auskarai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Tipas
Rich         Kakline          Auksas           0.3            30         300       Toksai
-------------------------------------------------------------------------------------------

Grandineles
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Ilgis
Fine         Grandinine       Sidabras         0.1            14         350           2
-------------------------------------------------------------------------------------------



Ziedine
 Areso g.
 865259124
Ziedai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Dydys
Fancy        Mazylis          Sidabras         0.1            20         310         0.2
-------------------------------------------------------------------------------------------

Auskarai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Tipas
Rich         Kakline          Auksas           0.3            30         600       Toksai
-------------------------------------------------------------------------------------------

Grandineles
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Ilgis
Fine         Grandinine       Sidabras         0.1            14         600           2
-------------------------------------------------------------------------------------------
```

Console:

```
Brangiausias ziedas yra Ziedine parduotuveje.
Brangiausi auskarai yra Ziedine parduotuveje.
Brangiausia grandinele yra Ziedine parduotuveje.
```

Unikalus.csv

Adidas papuosalai
Gatvynas
8.26E+08

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fancy	Ilgoji	Sidabras	0.4	20	349	1.8
Asiai	Rubiai	Ruby	0.5	40	500	Kitoksai
Nike	Stripe	Plienas	0.1	30	500	0.1

Papuosalai jums
Judesio g.
8.63E+08
Parduotuve neturi unikaliu papuosalu.

Ziedine
Areso g.
8.65E+08

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Fancy	Mazylis	Sidabras	0.1	20	310	0.2

300.csv

Juvelyriniai gaminiai pigesni nei 300 euru:

Adidas	Speed	Sidabras	0.2	40	200	0.2
Rich	Kakline	Auksas	0.3	30	300	Toksai

Brangus.csv

Brangus Ziedai:

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Adidas	Speed	Sidabras	0.2	40	550	0.2
Fancy	Didysis	Auksas	0.1	20	600	0.2
Nike	Stripe	Plienas	0.1	30	500	0.1

Brangus Auskarai:

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Tipas
Asiai	Rubiai	Ruby	0.5	40	500	Kitoksai
Rich	Kakline	Auksas	0.3	30	300	Toksai

Brangios Grandineles:

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fine	Grandinine	Sidabras	0.1	14	350	2
Fancy	Ilgoji	Sidabras	0.4	20	349	1.8

2 bandymas
PradiniaiDuomenys.txt

Adidas papuosalai
 Gatvynas
 825612581
Ziedai

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Fancy	Didysis	Auksas	0.1	20	600	0.2
Nike	Stripe	Plienas	0.1	30	500	0.1
Adidas	Speed	Sidabras	0.2	40	550	0.2

Auskarai

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Tipas
Asiai	Rubiai	Ruby	0.5	40	500	Kitoksai
Rich	Kakline	Auksas	0.3	30	600	Toksai

Grandineles

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fine	Grandinine	Sidabras	0.1	14	350	2
Fancy	Ilgoji	Sidabras	0.4	20	349	1.8

Nike
 Alytaus g.
 862541531
Ziedai

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Nike	Stripe	Plienas	0.1	30	500	0.1
Adidas	Speed	Sidabras	0.2	40	550	0.2
Engl	Cold	Silver	0.2	40	400	0.2

Auskarai

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Tipas
Rich	Kakline	Auksas	0.3	30	310	Toksai
Pimp	Swag	Auksas	0.3	40	500	Fancy

Grandineles

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fine	Grandinine	Sidabras	0.1	14	350	2

Papuosalai jums
 Judesio g.
 862512512
Ziedai

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Fancy	Didysis	Auksas	0.1	20	600	0.2
Adidas	Speed	Sidabras	0.2	40	450	0.2

Auskarai

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Tipas
Rich	Kakline	Auksas	0.3	30	600	Toksai

Grandineles

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fine	Grandinine	Sidabras	0.1	14	350	2
Fully	Grandus	Sidabras	0.1	20	400	1.3

Ziedine

```
 Areso g.
 865259124
Ziedai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Dydys
Fancy        Mazylis          Sidabras         0.1            20         310         0.2
-------------------------------------------------------------------------------------

Auskarai
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina       Tipas
Rich         Kakline          Auksas           0.3            30         600       Toksai
-------------------------------------------------------------------------------------
Kaujokas     Svajone          Sidabras         0.2            25         350       Saugus
-------------------------------------------------------------------------------------

Grandineles
Gamintojas   Pavadinimas      Metalas          Svoris         Praba      Kaina      Ilgis
Fine         Grandinine       Sidabras         0.1            14         600           2
-------------------------------------------------------------------------------------
```

300.csv

Juvelyriniai gaminiai
Pigiu papuosalu nera.

Brangus.csv

Brangus Ziedai:

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Adidas	Speed	Sidabras	0.2	40	550	0.2
Fancy	Didysis	Auksas	0.1	20	600	0.2
Nike	Stripe	Plienas	0.1	30	500	0.1

Brangus Auskarai:

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Tipas
Asiai	Rubiai	Ruby	0.5	40	500	Kitoksai
Pimp	Swag	Auksas	0.3	40	500	Fancy
Rich	Kakline	Auksas	0.3	30	600	Toksai
Kaujokas	Svajone	Sidabras	0.2	25	350	Saugus

Brangios Grandineles:

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fine	Grandinine	Sidabras	0.1	14	350	2
Fancy	Ilgoji	Sidabras	0.4	20	349	1.8
Fully	Grandus	Sidabras	0.1	20	400	1.3

Unikalus.csv

Adidas papuosalai
Gatvynas
8.26E+08

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fancy	Ilgoji	Sidabras	0.4	20	349	1.8
Asiai	Rubiai	Ruby	0.5	40	500	Kitoksai

Nike
Alytaus g.
8.63E+08

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Engl	Cold	Silver	0.2	40	400	0.2
Pimp	Swag	Auksas	0.3	40	500	Fancy

Papuosalai jums
Judesio g.
8.63E+08

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Ilgis
Fully	Grandus	Sidabras	0.1	20	400	1.3

Ziedine
Areso g.
8.65E+08

Gamintojas	Pavadinimas	Metalas	Svoris	Praba	Kaina	Dydys
Fancy	Mazylis	Sidabras	0.1	20	310	0.2
Kaujokas	Svajone	Sidabras	0.2	25	350	Saugus

5.4. Dėstytojo pastabos

- P5
- P6