KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

OBJEKTINIO PROGRAMAVIMO ĮVADAS (P175B157)

Laboratorinių darbų ataskaita

Atliko:
MGTM-3/2 gr. studentas
Antanas Dičkus
2023 m. lapkričio 14 d.
Priėmė: lekt. Rima Sturienė

KAUNAS 2023

TURINYS

1. Pažintis su klase

1.1. Darbo užduotis

U2–2. Liftas

• Sukurkite klasę Studentas, kuri turėtų kintamuosius amžiui ir ūgiui saugoti. Trys studentai nutarė treniruotis žaisti krepšinį. Raskite, koks aukščiausio studento amžius ir koks jauniausio studento ūgis.

• Papildykite klasę Studentas kintamuoju, skirtu studento svoriui saugoti. Sukurkite klasę Liftas, kuri turėtų kintamuosius lifto keliamosios galios reikšmei ir talpai saugoti. Per kelis kartus visi studentai pakils liftu į reikiamą aukštą?

• Papildykite klasę Liftas metodais Dėti(), kurie leistų keisti lifto keliamąją galią ir talpą. Ar visi studentai vienu metu bus pakelti į reikiamą aukštį, jeigu lifto keliamoji galia bus padvigubinta? O jeigu talpa bus padvigubinta?

1.2. Programos tekstas

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Security;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;

namespace MGTM_3_2_Antanas_Dičkus_L_1
{
    internal class Program
    {
        /// <summary>
        /// Klasė studento parametrams aprašyti
        /// </summary>
        class Studentas
        {
            private int ugis, // stuento ugis
                        svoris, // stuento svoris
                        amzius; // studento amžius

            /// <summary>
            /// Konstruktorius su parametrais
            /// </summary>
            /// <param name="ugis"></param>
            /// <param name="svoris"></param>
            /// <param name="amzius"></param>
            public Studentas(int ugis, int amzius, int svoris)
            {
                this.ugis = ugis;
                this.amzius = amzius;
                this.svoris = svoris;
            }
            /// <summary>
            /// Grąžina studento ūgį
            /// </summary>
            /// <returns> Studento ūgį </returns>
            public int DuotiŪgį()
            {
                return ugis;
            }
            /// <summary>
            /// Grąžina studento amžių
            /// </summary>
            /// <returns> Studento amžius </returns>
            public int DuotiAmžių()
```

```
        {
            return amzius;
        }
        /// <summary>
        /// Grąžina studneto svorį
        /// </summary>
        /// <returns> Studento svoris </returns>
        public int DuotiSvorį()
        {
            return svoris;
        }

    }

    /// <summary>
    /// Klasė lifto parametrams aprašyti
    /// </summary>
    class Liftas
    {
        private int talpa,
                    galia;

        /// <summary>
        /// Konstruktorius su parametrais
        /// </summary>
        /// <param name="talpa"></param>
        /// <param name="galia"></param>
        public Liftas(int galia, int talpa)
        {
            this.talpa = talpa;
            this.galia = galia;
        }
        /// <summary>
        /// Grąžina lifto galią
        /// </summary>
              /// <returns> Lifto galią </returns>
public int DuotiGalią()
{
    return galia;
}
/// <summary>
/// Grąžina lifto talpą
/// </summary>
/// <returns> Lifto talpą </returns>
public double DuotiTalpą()
{
    return talpa;
}

//talpa ir galia yra kintamieji argumentams
/// <summary>
/// Padvigubina lifto talpą
/// </summary>
/// <param name="talpa"></param>
public int DėtiTalpą()
{
    return talpa = talpa * 2;
}
/// <summary>
/// Padvigubina lifto galią
/// </summary>
/// <param name="galia"></param>
public int DėtiGalią()
{
    return galia = galia * 2;
}
```

```
}

static void Main(string[] args)
{
    Console.OutputEncoding = Encoding.UTF8;

    //Dialogai studentų parametrų įvedimui
    Console.WriteLine("Duomenys apie studentus:");
    Studentas s1 = duomenu_ivedimas_apie_studenta(1);
    Studentas s2 = duomenu_ivedimas_apie_studenta(2);
    Studentas s3 = duomenu_ivedimas_apie_studenta(3);

    //randama, kuris studentas yra pats aukščiausias ir išspausdinamas jo amžius
    Aukščiausias_Studentas(s1, s2, s3);

    //randama, kuris studentas yra pats jauniausias ir išspausdinamas jo ūgis
    Jauniausias_Studentas(s1, s2, s3);

    // Lentele spausdina Studentų duomenis palyginimui
    spausdinimas(s1, s2, s3);

    //Dialogai lifto parametrų įvedimui
    Liftas liftas = duomenu_ivedimas_apie_lifta();

    //Randama, per kiek kartų visi studentai pakils liftu į reikiamą aukštą
    galios_pakelimo_kartai(liftas, s1, s2, s3);
    liftas.DėtiTalpą();
    Console.WriteLine("Lifto našumas padvigubinus talpą:");
    //Randama, per kiek kartų visi studentai pakils liftu į reikiamą aukštą
padvigubinus galią
    galios_pakelimo_kartai(liftas, s1, s2, s3);
    liftas.DėtiGalią();
    Console.WriteLine("Lifto našumas padvigubinus galią:");
    //Randama, per kiek kartų visi studentai pakils liftu į reikiamą aukštą
padvigubinus talpą
    galios_pakelimo_kartai(liftas, s1, s2, s3);
    Console.ReadLine();
        }
/// <summary>
/// Studento parametrų įvedimas
/// </summary>
/// <param name="numeris"></param>
/// <returns> Studentą su įvestais parametrais </returns>
static Studentas duomenu_ivedimas_apie_studenta(int numeris)
{
    int studento_ugis = 0, studento_svoris = 0, stuento_amzius = 0;
    Studentas s;

    Console.Write("Įveskite {0} studento ūgį (centimetrais): ", numeris);
    studento_ugis = Convert.ToInt32(Console.ReadLine());

    Console.Write("Įveskite {0} studento svorį (kilogramais): ", numeris);
    studento_svoris = Convert.ToInt32(Console.ReadLine());

    Console.Write("Įveskite {0} studento amžių (metais): ", numeris);
    stuento_amzius = Convert.ToInt32(Console.ReadLine());
    s = new Studentas(studento_ugis, stuento_amzius, studento_svoris);
    return s;
}
/// <summary>
/// Lifto parametrų įvedimas
/// </summary>
/// <returns> Liftą su įvestais duomenimis </returns>
static Liftas duomenu_ivedimas_apie_lifta()
{
```

```
    int galia = 0, talpa = 0;
    Liftas liftas;

    Console.Write("Įveskite kiekį, kurį talpina liftas (vienetais): ");
    talpa = Convert.ToInt32(Console.ReadLine());


    Console.Write("Įveskite galią, kuria kelti gali liftas (kilogramais): ");
    galia = Convert.ToInt32(Console.ReadLine());


    liftas = new Liftas(galia, talpa);
    return liftas;
}
/// <summary>
/// Randa, kuris studentas yra aukščiausias
/// </summary>
/// <param name="s1"></param>
/// <param name="s2"></param>
/// <param name="s3"></param>
static void Aukščiausias_Studentas(Studentas s1, Studentas s2, Studentas s3)
{
    if (s1.DuotiŪgį() > s2.DuotiŪgį() && s1.DuotiŪgį() > s3.DuotiŪgį())
        Console.WriteLine("pirmasis studentas yra aukščiausias, jo amžius yra
{0}", s1.DuotiAmžių());
    else if (s2.DuotiŪgį() > s1.DuotiŪgį() && s2.DuotiŪgį() > s3.DuotiŪgį())
        Console.WriteLine("antrasis studentas yra aukščiausias, jo amžius yra
{0}", s2.DuotiAmžių());
    else if (s3.DuotiŪgį() > s1.DuotiŪgį() && s3.DuotiŪgį() > s2.DuotiŪgį())
        Console.WriteLine("trečiasis studentas yra aukščiausias, jo amžius
yra{0}", s3.DuotiAmžių());
    // else Console.WriteLine("Visų studentų ūgiai yra lygūs");
}

/// <summary>
/// Išveda lentelę su duomenimis apie studentus
/// </summary>
/// <param name="s1"></param>
/// <param name="s2"></param>
/// <param name="s3"></param>
static void spausdinimas(Studentas s1, Studentas s2, Studentas s3)
{
    Console.WriteLine("-------------------------------------------------------
---");
    Console.WriteLine("Studento:\t\t amžius |\t ūgis |\t       svoris|");

    Console.WriteLine("-------------------------------------------------------
---");
    Console.WriteLine(" Pirmas studentas:  {0, 10:d}  |{1, 13:d}| {2, 13:d}|",
        s1.DuotiAmžių(), s1.DuotiŪgį(), s1.DuotiSvorį());

    Console.WriteLine("-------------------------------------------------------
---");
    Console.WriteLine(" Antras studentas:  {0, 10:d}  |{1, 13:d}| {2, 13:d}|",
        s2.DuotiAmžių(), s2.DuotiŪgį(), s2.DuotiSvorį());

    Console.WriteLine("-------------------------------------------------------
---");
    Console.WriteLine(" Trecias studentas: {0, 10:d}  |{1, 13:d}| {2, 13:d}|",
        s3.DuotiAmžių(), s3.DuotiŪgį(), s3.DuotiSvorį());

    Console.WriteLine("-------------------------------------------------------
---");
}
```

```
/// <summary>
/// Randa, kuris studentas yra jauniausias
/// </summary>
/// <param name="s1"></param>
/// <param name="s2"></param>
/// <param name="s3"></param>
static void Jauniausias_Studentas(Studentas s1, Studentas s2, Studentas s3)
{
    if (s1.DuotiAmžių() < s2.DuotiAmžių() && s1.DuotiAmžių() < s3.DuotiAmžių())
        Console.WriteLine("Pirmasis studentas yra jauniausias, jo ūgis yra {0}",
s1.DuotiŪgį());
    else if (s2.DuotiAmžių() < s1.DuotiAmžių() && s2.DuotiAmžių() <
s3.DuotiAmžių())
        Console.WriteLine("Antrasis studentas yra jauniausias, jo ūgis yra {0}",
s2.DuotiŪgį());
    else if (s3.DuotiAmžių() < s2.DuotiAmžių() && s3.DuotiAmžių() <
s1.DuotiAmžių())
        Console.WriteLine("Trečiasis studentas yra jauniausias, jo ūgis yra {0}",
s3.DuotiŪgį());
}
/// <summary>
/// Apskaičiuoja, koks yra studentų bendras svoris
/// </summary>
/// <param name="pirmas_svoris"></param>
/// <param name="antras_svoris"></param>
/// <param name="trecias_svoris"></param>
/// <returns> Studentų bendrą svorį</returns>
        static int svoriu_suma(int pirmas_svoris, int antras_svoris, int
trecias_svoris)
        {
            int suma = pirmas_svoris + antras_svoris + trecias_svoris;
            return suma;
        }
        /// <summary>
        /// Apskaičiuoja per kartų kartus visi studentai pakils liftu į reikiamą
aukštą
        /// </summary>
        /// <param name="liftas"></param>
        /// <param name="s1"></param>
        /// <param name="s2"></param>
        /// <param name="s3"></param>
        static void galios_pakelimo_kartai(Liftas liftas, Studentas s1, Studentas
s2, Studentas s3)
        {
            if (liftas.DuotiTalpą() >= 3)
            {
                if (liftas.DuotiGalią() >= svoriu_suma(s1.DuotiSvorį(),
s2.DuotiSvorį(), s3.DuotiSvorį()))
                    Console.WriteLine("Studentai užkils per vieną kartą");
                else if (liftas.DuotiGalią() >= (s1.DuotiSvorį() +
s2.DuotiSvorį()) && liftas.DuotiGalią() >= (s2.DuotiSvorį() + s3.DuotiSvorį()) &&
liftas.DuotiGalią() >= s1.DuotiSvorį() + s3.DuotiSvorį())
                    Console.WriteLine("Studentai užkils per du kartus");
                else if (liftas.DuotiGalią() >= s1.DuotiSvorį() &&
liftas.DuotiGalią() >= s2.DuotiSvorį() && liftas.DuotiGalią() >= s3.DuotiSvorį())
                    Console.WriteLine("Studentai užkils po vieną per tris
kartus");
                else
                    Console.WriteLine("Bent vienas iš studentų sveria perdaug, kad
liftas galėtų pakelti jį, todėl visi studentai negalės užkilti");
            }

            else if (liftas.DuotiTalpą() == 2)
            {
```

```
                if (liftas.DuotiGalią() >= (s1.DuotiSvorį() + s2.DuotiSvorį()) &&
liftas.DuotiGalią() >= (s2.DuotiSvorį() + s3.DuotiSvorį()) && liftas.DuotiGalią()
>= s1.DuotiSvorį() + s3.DuotiSvorį())
                    Console.WriteLine("Studentai užkils per du kartus");
                else if (liftas.DuotiGalią() >= s1.DuotiSvorį() &&
liftas.DuotiGalią() >= s2.DuotiSvorį() && liftas.DuotiGalią() >= s3.DuotiSvorį())
                    Console.WriteLine("Studentai užkils po vieną per tris
kartus");
                else
                    Console.WriteLine("Bent vienas iš studentų sveria perdaug, kad
liftas galėtų pakelti jį, todėl visi studentai negalės užkilti");
            }
            else if (liftas.DuotiTalpą() == 1)
            {
                if (liftas.DuotiGalią() >= s1.DuotiSvorį() && liftas.DuotiGalią()
>= s2.DuotiSvorį() && liftas.DuotiGalią() >= s3.DuotiSvorį())
                    Console.WriteLine("Studentai užkils po vieną per tris
kartus");
                else
                    Console.WriteLine("Bent vienas iš studentų sveria perdaug, kad
liftas galėtų pakelti jį, todėl visi studentai negalės užkilti");
            }
        }
    }
}
```

1.3. Pradiniai duomenys ir rezultatai

Pavyzdys nr.1:

```
Duomenys apie studentus:
Įveskite 1 studento ūgį (centimetrais): 185
Įveskite 1 studento svorį (kilogramais): 85
Įveskite 1 studento amžių (metais): 18
Įveskite 2 studento ūgį (centimetrais): 199
Įveskite 2 studento svorį (kilogramais): 99
Įveskite 2 studento amžių (metais): 19
Įveskite 3 studento ūgį (centimetrais): 158
Įveskite 3 studento svorį (kilogramais): 58
Įveskite 3 studento amžių (metais): 15
antrasis studentas yra aukščiausias, jo amžius yra 19
------------------------------------------------------------
Studento:               amžius |       ūgis |       svoris|
------------------------------------------------------------
 Pirmas studentas:          18  |         185|           85|
------------------------------------------------------------
 Antras studentas:          19  |         199|           99|
------------------------------------------------------------
 Trecias studentas:         15  |         158|           58|
------------------------------------------------------------
Įveskite kiekį, kurį talpina liftas (vienetais): 3
Įveskite galią, kuria kelti gali liftas (kilogramais): 450
Su lifto standartiniais parametrais:
Studentai užkils per vieną kartą
Padvigubinus lifto talpa:
Studentai užkils per vieną kartą
Padvigubinus lifto galia:
Studentai užkils per vieną kartą
```

Pavyzdys nr.2:

```
Duomenys apie studentus:
Įveskite 1 studento ūgį (centimetrais): 188
Įveskite 1 studento svorį (kilogramais): 88
Įveskite 1 studento amžių (metais): 18
Įveskite 2 studento ūgį (centimetrais): 212
Įveskite 2 studento svorį (kilogramais): 112
Įveskite 2 studento amžių (metais): 22
Įveskite 3 studento ūgį (centimetrais): 158
Įveskite 3 studento svorį (kilogramais): 58
Įveskite 3 studento amžių (metais): 15
antrasis studentas yra aukščiausias, jo amžius yra 22
------------------------------------------------------------
Studento:               amžius |        ūgis |        svoris|
------------------------------------------------------------
 Pirmas studentas:          18  |          188|            88|
------------------------------------------------------------
 Antras studentas:          22  |          212|           112|
------------------------------------------------------------
 Trecias studentas:         15  |          158|            58|
------------------------------------------------------------
Įveskite kiekį, kurį talpina liftas (vienetais): 2
Įveskite galią, kuria kelti gali liftas (kilogramais): 250
Su lifto standartiniais parametrais:
Studentai užkils per du kartus
Padvigubinus lifto talpa:
Studentai užkils per du kartus
Padvigubinus lifto galia:
Studentai užkils per vieną kartą
```

Pavyzdys nr.3:

```
Duomenys apie studentus:
Įveskite 1 studento ūgį (centimetrais): 158
Įveskite 1 studento svorį (kilogramais): 75
Įveskite 1 studento amžių (metais): 15
Įveskite 2 studento ūgį (centimetrais): 202
Įveskite 2 studento svorį (kilogramais): 100
Įveskite 2 studento amžių (metais): 20
Įveskite 3 studento ūgį (centimetrais): 188
Įveskite 3 studento svorį (kilogramais): 88
Įveskite 3 studento amžių (metais): 18
antrasis studentas yra aukščiausias, jo amžius yra 20
Pirmasis studentas yra jauniausias, jo ūgis yra 158
------------------------------------------------------------
Studento:               amžius |        ūgis |        svoris|
------------------------------------------------------------
 Pirmas studentas:          15  |          158|            75|
------------------------------------------------------------
 Antras studentas:          20  |          202|           100|
------------------------------------------------------------
 Trecias studentas:         18  |          188|            88|
------------------------------------------------------------
Įveskite kiekį, kurį talpina liftas (vienetais): 1
Įveskite galią, kuria kelti gali liftas (kilogramais): 95
Su lifto standartiniais parametrais:
Bent vienas iš studentų sveria perdaug, kad liftas galėtų pakelti jį, todėl visi studentai negalės užkilti
Padvigubinus lifto talpa:
Bent vienas iš studentų sveria perdaug, kad liftas galėtų pakelti jį, todėl visi studentai negalės užkilti
Padvigubinus lifto galia:
Studentai užkils per du kartus
```

1.4. Dėstytojo pastabos

2. Objektų rinkinys

2.1. Darbo užduotis

U3–2. Krepšinis

• Krepšinio mokykloje treniruotes lankančių sąrašas yra tekstiniame faile: būsimo krepšininko vardas ir pavardė, amžius ir ūgis. Pirmoje eilutėje yra krepšinio mokyklos pavadinimas. Sukurkite klasę Krepšininkas, kuri turėtų kintamuosius vardui su pavarde, amžiui bei ūgiui saugoti. Raskite, koks būsimų krepšininkų amžiaus vidurkis ir koks ūgio vidurkis.

• Papildykite programą veiksmais su dviejų krepšinio mokyklų duomenimis. Kiekvienos mokyklos duomenys saugomi atskiruose failuose. Kurioje mokykloje aukščiausias sportinkas? Surašykite į atskirą rinkinį visus abiejų mokyklų sportininkus, kurių ūgis didesnis už vidurkį.

2.2. Programos tekstas

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

namespace MGTM_32_Dičkus_Antanas_L2
{
    /// <summary>
    /// Klasė Krepšininko parametrams aprašyti
    /// </summary>
    class Krepsininkas
    {
        private string vardas_pavarde;// Krepšininko vardas ir pavardė
        private int amzius;//Krepšininko amžius
        private double ugis;//Krepšininko ūgis
        /// <summary>
        /// Konstruktorius su parametrais
        /// </summary>
        /// <param name="vardas_pavarde"></param>
        /// <param name="amzius"></param>
        /// <param name="ugis"></param>
        public Krepsininkas(string vardas_pavarde, int amzius, double ugis)
        {
            this.vardas_pavarde = vardas_pavarde;
            this.amzius = amzius;
            this.ugis = ugis;
        }
        /// <summary>
        /// Metodas, gražinantis Krepšininko vardą ir pavardę
        /// </summary>
        /// <returns>Krepšininko vardą ir pavardę</returns>
        public string Duoti_vardas_pavarde()
        {
            return vardas_pavarde;
        }
        /// <summary>
        /// Metodas, gražinantis Krepšininko amžių
        /// </summary>
        /// <returns>Krepšininko amžių</returns>
        public int Duoti_amzius()
        {
            return amzius;
        }
        /// <summary>
        /// Metodas, gražinantis Krepšininko ūgį
        /// </summary>
        /// <returns>Krepšininko ūgį</returns
        public double Duoti_ugis()
```

```
        {
            return ugis;
        }

    }
    internal class Program
    {
        const string fd1 = "..\\..\\Duomenys1.txt";// pirmasis duomenų failas
        const string fd2 = "..\\..\\Duomenys2.txt";// antrasis duomenų failas
        const string fr = "..\\..\\Rezultatai.txt";// rezultatų failas
        const int CMax = 1000;
        static void Main(string[] args)
        {
            /*
             n1 - pirmo duomenų failo krepsininkų skaičius;
             n2 - antrojo duomenų failo krepsininkų skaičius;
            */
            int n1, n2;
            /*
             mokykla1 - pirmosios krepšinio mokyklos pavadinimas
             mokykla2 - antrosios krepšinio mokyklos pavadinimas
             */
            string mokykla1, mokykla2;
            /*
             K1 - pirmosios krepšinio mokyklos krepšininkų sąrašas
             K2 - antrosios krepšinio mokyklos krepšininkų sąrašas
            */
            Krepsininkas[] K1 = new Krepsininkas[CMax];
            Krepsininkas[] K2 = new Krepsininkas[CMax];

            /*
             Bendras - pirmosios ir antrosios krepšinio mokyklos krepšininkų sąrašas
             Auksciausias - aukštesnių nei ūgio vidurkis krepšininkų sąrašas
            */
            Krepsininkas[] Bendras = new Krepsininkas[CMax];
            Krepsininkas[] Auksciausi = new Krepsininkas[CMax];
            /*
             nAuksciausi - Aukštesnių nei vidurkis krepšininkų skaičius
             nBendras - pirmosios ir antrosios krepšinio mokyklos krepšininkų skaičius
             */
            int nAuksciausi;
            int nBendras = 0;

            nuskaitymas(out n1, K1, fd1, out mokykla1);
            pradiniu_duomenu_spaudinimas(n1, K1, fr, mokykla1);
            nuskaitymas(out n2, K2, fd2, out mokykla2);
            pradiniu_duomenu_spaudinimas(n2, K2, fr, mokykla2);

            if (File.Exists(fr))
                File.Delete(fr);

            Bendras_sarasas(ref nBendras, Bendras, K1, n1);
            Bendras_sarasas(ref nBendras, Bendras, K2, n2);
            Auksciausiu_sarasas(nBendras, out nAuksciausi, Bendras, Auksciausi);


            isvedimas_vidurkiu(n1, K1, fr, mokykla1);
            isvedimas_vidurkiu(n2, K2, fr, mokykla2);

            Galutinis_isvedimas(n1, n2, K1, K2, mokykla1, mokykla2, fr, Auksciausi,
nAuksciausi, nBendras, Bendras);

            Console.ReadLine();
        }
        /// <summary>
        /// Metodas, nuskaitantis pradinius duomenis iš failo
        /// </summary>
        /// <param name="n"></param>
        /// <param name="K"></param>
        /// <param name="fd"></param>
```

```
        /// <param name="mokykla"></param>
        private static void nuskaitymas(out int n, Krepsininkas[] K, string fd,
out string mokykla)
        {
            n = 0;
            string[] dalys;
            string vardasPavarde;
            int amzius;
            double ugis;
            bool galima = true;
            using (StreamReader skaitymas = new StreamReader(fd))
            {
                string laikinas = skaitymas.ReadLine();
                mokykla = laikinas;
                laikinas = skaitymas.ReadLine();
                while (galima == true)
                {
                    if (laikinas != null)
                    {
                        dalys = laikinas.Split(' ');
                        vardasPavarde = dalys[0] + " " + dalys[1];
                        amzius = Convert.ToInt32(dalys[2]);
                        ugis = Convert.ToDouble(dalys[3]);
                        K[n] = new Krepsininkas(vardasPavarde, amzius, ugis);
                        n++;
                        laikinas = skaitymas.ReadLine();
                    }
                    else { galima = false; }
                }
            }
        }
        /// <summary>
        /// Metodas, skaičiuojantis ūgio vidurkį
        /// </summary>
        /// <param name="K"></param>
        /// <param name="n"></param>
        /// <returns>ūgio vidurkį</returns>
        static double ugio_vidurkis(Krepsininkas[] K, int n)
        {
            double suma = 0;
            for (int i = 0; i < n; i++)
            {
                suma = suma + K[i].Duoti_ugis();
            }
            suma = suma/n;
            return suma;
        }
        /// <summary>
        /// Metodas, skaičiuojantis amžiaus vidurkį
        /// </summary>
        /// <param name="K"></param>
        /// <param name="n"></param>
        /// <returns></returns>
        static int amziaus_vidurkis(Krepsininkas[] K, int n)
        {
            int suma = 0;
            for (int i = 0; i < n; i++)
            {
                suma = suma + K[i].Duoti_amzius();
            }
            suma = suma/n;
            return suma;
        }
        /// <summary>
        /// Metodas, sukuriantis pirmosios ir antrosios krepšinio mokyklos krepšininkų
sąrašą
        /// </summary>
        /// <param name="n"></param>
        /// <param name="Bendras"></param>
        /// <param name="K"></param>
```

```
        /// <param name="m"></param>
        private static void Bendras_sarasas(ref int n, Krepsininkas[] Bendras,
Krepsininkas[] K, int m)
        {
            for (int i = 0; i < m; i++)
            {
                Bendras[n] = K[i];
                n++;
            }
        }
        /// <summary>
        /// Metodas, surandantis aukščiausio sąrašo žaidėjo indeksą
        /// </summary>
        /// <param name="n"></param>
 /// <param name="K"></param> /// <returns> aukščiusio žaidėjo indeksą</returns>
 static int Auksciausias_zaidejas(int n, Krepsininkas[] K)
 {
     int did = 0;
     for (int i = 0; i < n; i++)
     {
         if (K[did].Duoti_ugis() < K[i].Duoti_ugis())
         {
             did = i;
         }
     }
     return did;
 }
 /// <summary>
 /// Metodas, gražinantis mokyklos, kurioje mokosi pats aukščiausias žaidėjas, vardą
 /// </summary>
 /// <param name="n1"></param>
 /// <param name="n2"></param>
 /// <param name="K1"></param>
 /// <param name="K2"></param>
 /// <param name="mokykla1"></param>
 /// <param name="mokykla2"></param>
 /// <returns>mokyklos, kurioje mokosi pats aukščiausias žaidėjas, vardą</returns>
 static string Auksciausia_zaideja_turinti_mokykla(int n1, int n2, Krepsininkas[] K1,
    Krepsininkas[] K2, string mokykla1, string mokykla2)
 {
     string mokykla;
     if (K1[Auksciausias_zaidejas(n1, K1)].Duoti_ugis()
         > K2[Auksciausias_zaidejas(n2, K2)].Duoti_ugis())
     {
         mokykla = mokykla1;
     }
     else if (K1[Auksciausias_zaidejas(n1, K1)].Duoti_ugis()
         < K2[Auksciausias_zaidejas(n2, K2)].Duoti_ugis())
     {
         mokykla = mokykla2;
     }
     else mokykla = "Abiejose mokyklose yra vienodo ūgio aukščiausi žaidėjai";
     return mokykla;
 }
 /// <summary>
 /// Metodas, gražinantis Bendro sąrašo krepšininkų ūgio vidurkį
 /// </summary>
 /// <param name="Bendras"></param>
 /// <param name="nBendras"></param>
 /// <returns> visų krepšininkų ūgio vidurkis</returns>
 static double ugiu_vidurkis(Krepsininkas[] Bendras, int nBendras)
 {
     double vidurkis = 0;
     for (int i = 0; i < nBendras; i++)
     {
         vidurkis = vidurkis + Bendras[i].Duoti_ugis();
     }
     vidurkis = vidurkis / nBendras;
     return vidurkis;
 }
```

```
 /// <summary>
 /// Metodas, sukuriantis atskirą rinkinį visus abiejų mokyklų sportininkus, kurių ūgis
didesnis už vidurkį
 /// </summary>
 /// <param name="nBendras"></param>
 /// <param name="nAukstu"></param>
 /// <param name="Bendras"></param>
 /// <param name="Auksti"></param>
 private static void Auksciausiu_sarasas(int nBendras, out int nAukstu,
 Krepsininkas[] Bendras, Krepsininkas[] Auksti)
 {
     nAukstu = 0;
     for (int i = 0; i < nBendras; i++)
     {
         if (Bendras[i].Duoti_ugis() > ugiu_vidurkis(Bendras, nBendras))
         {
             Auksti[nAukstu] = Bendras[i];
             nAukstu++;
         }
     }

 }
 /// <summary>
 /// Metodas, spausdinantis atskirų mokyklų amžiaus ir ūgio vidurkius
 /// </summary>
 /// <param name="n"></param>
 /// <param name="K"></param>
 /// <param name="fr"></param>
 /// <param name="mokykla"></param>
 private static void isvedimas_vidurkiu(int n, Krepsininkas[] K, string fr, string
mokykla)
 {
     const string virsus =
     "|-------------------------------------|----------------------|----" +
"-------------""|\r\n"
   + "| Mokyklos pavadinimas pavadinimas    |   Amžiaus vidurkis    |   Ūgio vidurkis
|\r\n"
   + "|-------------------------------------|----------------------|----" +
"-------------""|\r\n"
     using (var rez = System.IO.File.AppendText(fr))
     {
         rez.WriteLine(virsus);
         rez.WriteLine("| {0, 35} | {1, 21} | {2, 18:f2} |",
         mokykla, amziaus_vidurkis(K, n), ugio_vidurkis(K, n));
         rez.WriteLine("-----------------------------------------------------" +
"--------------------------------");
         rez.WriteLine();

     }
       }
/// <summary>
/// Metodas, spaudinantis pradinius duomenis
/// </summary>
/// <param name="n"></param>
/// <param name="K"></param>
/// <param name="fr"></param>
/// <param name="mokykla"></param>
private static void pradiniu_duomenu_spaudinimas(int n, Krepsininkas[] K, string fr,
string mokykla)
{
    const string virsus =
    "|-------------------------------------|----------------------|-----------"  +
"--------|\r\n"
  + "| Krepšininko vardas                  |   Amžius              |   Ūgis
|\r\n"
  + "|-------------------------------------|----------------------|-------------" +
"-------|";
    using (var rez = System.IO.File.AppendText(fr))
    {
        rez.WriteLine("mokyklos pavadinimas: {0}", mokykla);
```

```
            rez.WriteLine(virsus);
            for(int i= 0; i<n; i++)
            {
            rez.WriteLine("| {0, 35} | {1, 21} | {2, 18:f2} |",
            K[i].Duoti_vardas_pavarde(), K[i].Duoti_amzius(), K[i].Duoti_ugis());
            rez.WriteLine("-----------------------------------------------------” +
“----------------------------");

            }
            rez.WriteLine();
        }
       }
            /// <summary>
            /// Metodas, spausdinantis Aukščiausią žaidėją turinčios mokyklos vardą ir
lentelę, kurioje yra visų žaidėjų, aukštesnių nei vidurkis, parametrai
            /// </summary>
            /// <param name="n1"></param>
            /// <param name="n2"></param>
            /// <param name="K1"></param>
            /// <param name="K2"></param>
            /// <param name="mokykla1"></param>
            /// <param name="mokykla2"></param>
            /// <param name="fr"></param>
            /// <param name="Auksciausi"></param>
            /// <param name="nAuksciausiu"></param>
            private static void Galutinis_isvedimas(int n1, int n2, Krepsininkas[] K1,
               Krepsininkas[] K2, string mokykla1, string mokykla2, string fr,
               Krepsininkas[] Auksciausi, int nAuksciausiu)
            {

                const string virsus =
 "|-----------------------------------|----------------------|---------------” +
“-----|\r\n"
              + "| Vardas Pavardė                    |        Amžiaus        |    “ +
    “Ūgis        |\r\n"
            + "|-----------------------------------|----------------------|---------“ +
”----------|";
                using (var rez = System.IO.File.AppendText(fr))
                {
                    rez.WriteLine("Aukščiausią žaidėją turinti mokykla: {0}",
                        Auksciausia_zaideja_turinti_mokykla(n1, n2, K1, K2, mokykla1,
mokykla2));
                    rez.WriteLine();
rez.WriteLine("Žaidėjų, kurių ūgis yra didesnis negu vidutinis ({0})” +” sąrašas:",
ugiu_vidurkis(Bendras, nBendras));                rez.WriteLine(virsus);
                    for (int i = 0; i < nAuksciausiu; i++)
                    {
                        rez.WriteLine("| {0, 35} | {1, 21} | {2, 18:f2} |",
                            Auksciausi[i].Duoti_vardas_pavarde(),
                            Auksciausi[i].Duoti_amzius(), Auksciausi[i].Duoti_ugis());

                        rez.WriteLine("-----------------------------------------” +
“---------------------------------------------");
                    }


                }
            }
        }
}
```

2.3. Pradiniai duomenys ir rezultatai

1 duomenys ir rezultai:

```
Aukščiausią žaidėją turinti mokykla: Abiejų mokyklų aukščiausi žaidėjai yra vienodo ūgio, todėl iš abiejų

Žaidėjų, kurių ūgis yra didesnis negu vidutinis, sąrašas:
|-----------------------------------|---------------------|--------------------|
| Vardas Pavardė                    |        Amžiaus      |        Ūgis        |
|-----------------------------------|---------------------|--------------------|
|                      Tanoka Beard |                  27 |             212.00 |
-------------------------------------------------------------------------------
|                     Marius Grigonis |                  27 |             199.00 |
-------------------------------------------------------------------------------
|                    Paulius Jankūnas |                  22 |             203.00 |
-------------------------------------------------------------------------------
|                    Artūras Milaknis |                  21 |             199.14 |
-------------------------------------------------------------------------------
|                     Arvydas Sabonis |                  35 |             212.00 |
-------------------------------------------------------------------------------

mokyklos pavadinimas: zalgiris1
|-----------------------------------|---------------------|--------------------|
| Krepšininko vardas                |   Amžius            |   Ūgis             |
|-----------------------------------|---------------------|--------------------|
|                    Lukas Lekavičius |                  29 |             181.00 |
-------------------------------------------------------------------------------
|                      Tanoka Beard |                  27 |             212.00 |
-------------------------------------------------------------------------------
|                     Dainius Šalenga |                  37 |             188.20 |
-------------------------------------------------------------------------------
|                Valdemaras Chomičius |                  14 |             179.50 |
-------------------------------------------------------------------------------
|                     Marius Grigonis |                  27 |             199.00 |
-------------------------------------------------------------------------------

mokyklos pavadinimas: zalgiris2
|-----------------------------------|---------------------|--------------------|
| Krepšininko vardas                |   Amžius            |   Ūgis             |
|-----------------------------------|---------------------|--------------------|
|                    Paulius Jankūnas |                  22 |             203.00 |
-------------------------------------------------------------------------------
|                    Artūras Milaknis |                  21 |             199.14 |
-------------------------------------------------------------------------------
|                   Rimas Kurtinaitis |                  27 |             189.70 |
-------------------------------------------------------------------------------
|                    Edgaras Ulanovas |                  31 |             180.00 |
-------------------------------------------------------------------------------
|                     Arvydas Sabonis |                  35 |             212.00 |
-------------------------------------------------------------------------------

|-----------------------------------|---------------------|--------------------|
| Mokyklos pavadinimas pavadinimas  |   Amžiaus vidurkis  |   Ūgio vidurkis    |
|-----------------------------------|---------------------|--------------------|
|                          zalgiris1 |                  26 |             191.94 |
-------------------------------------------------------------------------------

|-----------------------------------|---------------------|--------------------|
| Mokyklos pavadinimas pavadinimas  |   Amžiaus vidurkis  |   Ūgio vidurkis    |
|-----------------------------------|---------------------|--------------------|
|                          zalgiris2 |                  27 |             196.77 |
-------------------------------------------------------------------------------
```

2 duomenys ir rezultatai:

mokyklos pavadinimas: zalgiris1

Krepšininko vardas	Amžius	Ūgis
Lukas Lekavičius	29	181.00
Tanoka Beard	27	212.00
Dainius Šalenga	37	188.20
Valdemaras Chomičius	14	179.50
Marius Grigonis	27	199.00

mokyklos pavadinimas: zalgiris2

Krepšininko vardas	Amžius	Ūgis
Paulius Jankūnas	22	203.00
Artūras Milaknis	21	199.14
Rimas Kurtinaitis	27	189.70
Edgaras Ulanovas	31	180.00
Arvydas Sabonis	35	215.00

Mokyklos pavadinimas pavadinimas	Amžiaus vidurkis	Ūgio vidurkis
zalgiris1	26	191.94

Mokyklos pavadinimas pavadinimas	Amžiaus vidurkis	Ūgio vidurkis
zalgiris2	27	197.37

Auksčiausią žaidėją turinti mokykla: zalgiris2

Žaidėjų, kurių ūgis yra didesnis negu vidutinis, sąrašas:

Vardas Pavardė	Amžiaus	Ūgis
Tanoka Beard	27	212.00
Marius Grigonis	27	199.00
Paulius Jankūnas	22	203.00
Artūras Milaknis	21	199.14
Arvydas Sabonis	35	215.00

2.4. Dėstytojo pastabos

3. Konteinerinė klasė

3.1. Darbo užduotis

U4 - 2.Mobiliojo ryšio kortelės
Norėdamas palyginti mobiliojo ryšio operatorių siūlomas išankstinio mokėjimo korteles Sirvydas surinko šią informaciją į tekstinį failą. Faile eilutėmis yra kortelių duomenys: kortelės(tinklo) pavadinimas, pradinė suma kortelėje, tarifas savame tinkle, tarifas į kitus tinklus, SMS žinučių tarifas savame tinkle ir į kitus tinklus. Parašykite programą, kuri spausdintų kortelių duomenis lentele, surastų kortelę, kurios SMS žinučių tarifai į kitus tinklus mažiausi. Papildykite programą veiksmais, kurie leistų atrinkti korteles, kurios leidžia skambinti ir siųsti SMS žinutes savame tinkle nemokamai, ir šį sąrašą surikiuoti pagal pradinę sumą mažėjimo tvarka ir kortelės pavadinimą abėcėliškai.

3.2. Programos tekstas

```
using System;
using System.Collections.Generic;
using System.ComponentModel.Design;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
namespace MGTM_32_Antanas_Dickus_L3
{
    /// <summary>
    /// Klase aprašanti korteles duomenis
    /// </summary>
    class Kortele
    {
        private string pavadinimas;//korteles pavadinimas
        private int suma;//korteleje esanti pinigu suma
        private double tarifas_sav,// skambuciu tarifas savame tinkle
            tarifas_sve, //skambuciu tarifas svetimame tinkle
            sms_sav, //SMS tarifas savame tinkle
            sms_sve; //SMS tarifas svetimame tinkle


        /// <summary>
        /// klases "korteles" konstruktorius be parametru
        /// </summary>
        public Kortele()
        {
            this.pavadinimas = "Pavadinimas nepateiktas";
            this.suma = 0;
            this.tarifas_sav = 0;
            this.tarifas_sve = 0;
            this.sms_sav = 0;
            this.sms_sve = 0;
        }
        /// <summary>
        /// klases "korteles" konstruktorius su parametrais
        /// </summary>
        /// <param name="pavadinimas"></param>
        /// <param name="suma"></param>
        /// <param name="tarifas_sav"></param>
        /// <param name="tarifas_sve"></param>
        /// <param name="sms_sav"></param>
        /// <param name="sms_sve"></param>
        public Kortele(string pavadinimas, int suma,
            double tarifas_sav, double tarifas_sve,
            double sms_sav, double sms_sve)
        {
            this.pavadinimas = pavadinimas;
            this.suma = suma;
```

```
            this.tarifas_sav = tarifas_sav;
            this.tarifas_sve = tarifas_sve;
            this.sms_sav = sms_sav;
            this.sms_sve = sms_sve;
        }
        /// <summary>
        /// metodas, sukuriantis lentele
        /// korteles duomenims
        /// </summary>
        /// <returns></returns>
        public override string ToString()
        {
            string eilute;
            eilute = string.Format("|{0, -17}|{1, -17}|{2, -17}|" +
                 "{3, -19}|{4, -18}|{5, -21}|",
                pavadinimas, suma,
                tarifas_sav, tarifas_sve,
                sms_sav, sms_sve);
            return eilute;
        }
        /// <summary>
        /// metodai, grazinantys korteles duomenis
        /// </summary>
        public double saviems_tarifas() { return tarifas_sav; }
        public double saviems_sms() { return sms_sav; }
        public double svetimiems() { return sms_sve + tarifas_sve; }
        public string Pavadinimas() { return pavadinimas; }
        public static bool operator <=(Kortele st1, Kortele st2)
        {
            int p = String.Compare(st1.pavadinimas,
                st2.pavadinimas,
            StringComparison.CurrentCulture);
            int s = st1.suma.CompareTo(st2.suma);
            return (s < 0 || (s == 0 && p > 0));
        }
        public static bool operator >=(Kortele st1, Kortele st2)
        {
            int p = String.Compare(st1.pavadinimas,
                st2.pavadinimas,
            StringComparison.CurrentCulture);
            int s = st2.suma.CompareTo(st1.suma);
            return (s > 0 || (s == 0 && p < 0));
        }
        public override bool Equals(object obj)
        {
            return obj is Kortele kortele &&
                   pavadinimas == kortele.pavadinimas &&
                   suma == kortele.suma &&
                   tarifas_sav == kortele.tarifas_sav &&
                   tarifas_sve == kortele.tarifas_sve &&
                   sms_sav == kortele.sms_sav &&
                   sms_sve == kortele.sms_sve;
               }
    public override int GetHashCode()
    {
        int hashCode = 612615005;
        hashCode = hashCode * -1521134295 +
EqualityComparer<string>.Default.GetHashCode(pavadinimas);
        hashCode = hashCode * -1521134295 + suma.GetHashCode();
        hashCode = hashCode * -1521134295 + tarifas_sav.GetHashCode();
        hashCode = hashCode * -1521134295 + tarifas_sve.GetHashCode();
        hashCode = hashCode * -1521134295 + sms_sav.GetHashCode();
        hashCode = hashCode * -1521134295 + sms_sve.GetHashCode();
        return hashCode;
    }
}
/// <summary>
/// konteineris, saugantis korteles duomenis
```

```
/// </summary>
class operatorius
{
    const int Cn = 100; // studentų masyvo dydis
    private Kortele[] Korteles; // studentų objektų masyvas
    private int kiek; // studentų skaičius

    /// <summary>
    /// konstruktorius pe parametru
    /// </summary>
    public operatorius()
    {
        kiek = 0;
        Korteles = new Kortele[Cn];
    }
    /// <summary>
    /// metodas, grazinantis i-tąją kortele
    /// </summary>
    /// <param name="i"></param>
    /// <returns> kortele </returns>
    public Kortele ImtiKortele(int i)
    { return Korteles[i]; }
    /// <summary>
    /// metodas, grazinantis korteliu kieki
    /// </summary>
    /// <returns></returns>
    public int ImtiKiek()
    { return kiek; }
    /// <summary>
    /// metodas, idedantis nauja kortele
    /// i oepratoriaus n-tąją vieta
    /// </summary>
    /// <param name="kt"></param>
    public void DetiKortele(Kortele kt)
    { Korteles[kiek++] = kt; }
    /// <summary>
    /// metodas, pakeiciantis korteliu
    /// indeksus masyve
    /// </summary>
    /// <param name="i"></param>
    /// <param name="j"></param>
    public void Rikiavimas()
    {
        for (int i = 0; i < kiek; i++)
        {
            Kortele min = Korteles[i];
            int im = i;
            for (int j = i + 1; j < kiek; j++)
            {
                if (min <= Korteles[j])
                {
                    min = Korteles[j];
                    im = j;
                    Korteles[im] = Korteles[i];
                    Korteles[i] = min;
                }
            }
        }
    }
    public Kortele pigiausia()
    {
        Kortele Pigiausia = Korteles[0];
        for (int i = 1; i < kiek; i++)
        {
            if (Korteles[i].svetimiems() <
                Pigiausia.svetimiems())
            {
                Pigiausia = Korteles[i];
```

```
            }
        }
        return Pigiausia;
    }

}
internal class Program
{//programoje naudojamos konstantos:
    const int Cn = 100;
    const string CFd1 = "..//..//Korteles.txt";
    const string CFr = "..//..//Rezultatai.txt";
    static void Main(string[] args)
    {
        /*
         * TestasMas - operatorius, saugantis pradinius duomenis
         * Nemokamos - operatorius, saugantis korteles,
         kurios leidzia nemokamai
         susisiekti su tame paciame
         tinkle esanciais numeriais
          */
        operatorius TestasMas = new operatorius();
         operatorius Nemokamos = new operatorius();
            string[] Pigiausios = new string[Cn];
            int pigios_kiekis = 0;
            Skaityti(CFd1, TestasMas);

            if (File.Exists(CFr)) File.Delete(CFr);

            Spausdinti(CFr, TestasMas,
                "Pradinis korteliu sarasas:",
                "Nera pradiniu duomenu");


            formuoti_pigiausias(TestasMas,
                ref Pigiausios, out pigios_kiekis);
            spaudinimas_pigiausiu(pigios_kiekis,
                Pigiausios, CFr);

            nemokamos_savame(TestasMas, Nemokamos);
            Nemokamos.Rikiavimas();

            Spausdinti(CFr, Nemokamos,
                "Korteles, leidziancios nemokamai" +
                " bendrauti savame tinkle:",
                "Nera korteliu, kurios leistu" +
                " nemokamai bendrauti savame tinkle");

        }
        /// <summary>
        /// metodas, nuskaitantis duomenis is pradinio failo
        /// </summary>
        /// <param name="fv"></param>
        /// <param name="kort"></param>
        static void Skaityti(string fv, operatorius kort)
        {
            using (StreamReader reader = new StreamReader(fv))
            {
                string line;
                while (reader.Peek() > -1)
                {
                    line = reader.ReadLine();
                    string[] parts = line.Split(' ');
                    string pav = parts[0];
                    int sum = Convert.ToInt32(parts[1]);
                    double tar_sav = double.Parse(parts[2]);
                    double tar_sve = double.Parse(parts[3]);
                    double sms_sav = double.Parse(parts[4]);
                    double sms_sve = double.Parse(parts[5]);
```

```
            Kortele kortele = new Kortele(pav, sum,
                tar_sav, tar_sve, sms_sav, sms_sve);
            kort.DetiKortele(kortele);
        }
    }
}
/// <summary>
/// metodas, spausdinantis lentele
/// </summary>
/// <param name="fv"></param>
/// <param name="kort"></param>
/// <param name="antraste"></param>
/// <param name="antraste2"></param>
static void Spausdinti(string fv,
    operatorius kort,
    string antraste,
    string antraste2)
{
    const string virsus =
        "-------------------------" +
        "-----------------------------------" +
        "-----------------------------------" +
        "-----------------\n" +
        "   Nr|Pavadinimas      |Suma           " +
        "  |Tarifas tinkle   |Tarifas ne tikle  " +
        " |SMS tarifas tinkle|SMS tarifas ne tinkle|\n" +
        "-------------------------" +
        "-----------------------------------" +
        "-----------------------------------" +
        "-----------------";
    using (var fr = File.AppendText(fv))
    {
        if (kort.ImtiKiek() > 0)
        {
            fr.WriteLine(antraste);
            fr.WriteLine(virsus);
            for (int i = 0; i < kort.ImtiKiek(); i++)
            {
                Kortele kt = kort.ImtiKortele(i);
                fr.WriteLine("{0, 4:d} {1}", i + 1,
                    kort.ImtiKortele(i).ToString());
            }
            fr.WriteLine("-------------------------" +
            "-----------------------------------" +
            "-----------------------------------" +
            "-----------------");
        }
        else fr.WriteLine(antraste2);

    }
}
/// <summary>
/// metodas, formauojantis pigiausia
/// tarifa svetimiems turinciu korteliu sarasa
/// </summary>
/// <param name="op"></param>
/// <param name="pigios"></param>
/// <param name="n"></param>
static void formuoti_pigiausias(operatorius op,
    ref string[] pigios, out int n)
{
    n = 0;
    for (int i = 0; i < op.ImtiKiek(); i++)
    {
        if (op.ImtiKortele(i).svetimiems() ==
            op.pigiausia().svetimiems())
        {
            pigios[n] = op.ImtiKortele(i).Pavadinimas();
```

```
                    n++;
                }
            }
        }
        static void spaudinimas_pigiausiu(int n,
            string[] pigios, string fv)
        {
            using (var fr = File.AppendText(fv))
            {
                fr.Write("Pigiausias tarifa " +
                    "svetimiems tinklams turi:");
                for (int i = 0; i < n; i++)
                {
                    fr.Write(" {0} ", pigios[i]);
                }
                fr.Write('\n');
            }
        }
        /// <summary>
        /// metodas, sukuriantis nemokama tarifa
        /// savame tinkle turinciu korteliu sarasa
        /// </summary>
        /// <param name="op"></param>
        /// <param name="nem"></param>
        static void nemokamos_savame(operatorius op,
            operatorius nem)
        {
            for (int i = 0; i < op.ImtiKiek(); i++)
            {
                if (op.ImtiKortele(i).saviems_tarifas() == 0 &&
                    op.ImtiKortele(i).saviems_sms() == 0)
                {
                    nem.DetiKortele(op.ImtiKortele(i));
                }
            }
        }
    }
}
```

3.3. Pradiniai duomenys ir rezultatai

1 duomenys ir rezultai:

```
Tele2 15 0.00 0.03 0.00 0.1
Telia 10 0.00 0.07 0.00 0.10
Bite 25 0.00 0.01 0.00 0.05
Labas 5 0.00 0.1 0.00 0.00
Ezys 25 0.00 0.1 0.00 0.00
```

```
Pradinis korteliu sarasas:
-------------------------------------------------------------------------------------------------------------------
  Nr|Pavadinimas      |Suma           |Tarifas tinkle   |Tarifas ne tikle   |SMS tarifas tinkle|SMS tarifas ne tinkle|
-------------------------------------------------------------------------------------------------------------------
   1 |Tele2           |15             |0                |0.03               |0                 |0.1                  |
   2 |Telia           |10             |0                |0.07               |0                 |0.1                  |
   3 |Bite            |25             |0                |0.01               |0                 |0.05                 |
   4 |Labas           |5              |0                |0.1                |0                 |0                    |
   5 |Ezys            |25             |0                |0.1                |0                 |0                    |
-------------------------------------------------------------------------------------------------------------------
Pigiausias tarifa svetimiems tinklams turi: Bite
Korteles, leidziancios nemokamai bendrauti savame tinkle:
-------------------------------------------------------------------------------------------------------------------
  Nr|Pavadinimas      |Suma           |Tarifas tinkle   |Tarifas ne tikle   |SMS tarifas tinkle|SMS tarifas ne tinkle|
-------------------------------------------------------------------------------------------------------------------
   1 |Bite            |25             |0                |0.01               |0                 |0.05                 |
   2 |Ezys            |25             |0                |0.1                |0                 |0                    |
   3 |Tele2           |15             |0                |0.03               |0                 |0.1                  |
   4 |Telia           |10             |0                |0.07               |0                 |0.1                  |
   5 |Labas           |5              |0                |0.1                |0                 |0                    |
-------------------------------------------------------------------------------------------------------------------
```

2 duomenys ir rezultai:

```
Tele2 15 0.01 0.03 0.00 0.1
Telia 10 0.70 0.07 0.00 0.10
Bite 25 0.80 0.01 0.00 0.05
Labas 5 0.10 0.01 0.00 0.00
Ezys 25 0.70 0.01 0.00 0.00
Pradinis korteliu sarasas:
-------------------------------------------------------------------------------------------------------------------
  Nr|Pavadinimas      |Suma           |Tarifas tinkle   |Tarifas ne tikle   |SMS tarifas tinkle|SMS tarifas ne tinkle|
-------------------------------------------------------------------------------------------------------------------
   1 |Tele2           |15             |0.01             |0.03               |0                 |0.1                  |
   2 |Telia           |10             |0.7              |0.07               |0                 |0.1                  |
   3 |Bite            |25             |0.8              |0.01               |0                 |0.05                 |
   4 |Labas           |5              |0.1              |0.01               |0                 |0                    |
   5 |Ezys            |25             |0.7              |0.01               |0                 |0                    |
-------------------------------------------------------------------------------------------------------------------
Pigiausias tarifa svetimiems tinklams turi: Labas  Ezys
Nera korteliu, kurios leistu nemokamai bendrauti savame tinkle
```

3.4. Dėstytojo pastabos

4. Teksto analizė ir redagavimas

4.1. Darbo užduotis

4.2. Programos tekstas

4.3. Pradiniai duomenys ir rezultatai

4.4. Dėstytojo pastabos

5. Susieti rinkiniai

5.1. Darbo užduotis

5.2. Programos tekstas

5.3. Pradiniai duomenys ir rezultatai

5.4. Dėstytojo pastabos