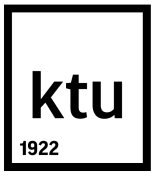

Kauno technologijos universitetas
Informatikos fakultetas

Objektinis programavimas I (P175B118)

Laboratorinių darbų ataskaita

Nojus Raškevičius, IFF-0/6

Studentas

Prof. Vacius Jusas

Dėstytojas

Kaunas 2020

Turinys

1 Duomenų klasė

1.1 Darbo užduotis

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 daugiausiai gyvybės taškų turintį herojų, ekrane atspausdinkite jo vardą, rasę, klasę ir gyvybės taškų kiekį. Jei yra keli, spausdinkite visus.
- Raskite žaidėją, kurio gynybos ir žalos taškų skirtumas yra mažiausias. Atspausdinkite informaciją apie žaidėją į ekraną. Jei yra keli, spausdinkite visus.
- Sudarykite visų herojų klasių sąrašą, klasių pavadinimus įrašykite į failą „Klasės.csv“ .

1.2 Programos tekstas

```
//Hero.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace U1_24_NR_ND
{
    /// <summary>
    /// the main hero class that is used throughout the program
    /// </summary>
    class Hero
    {
        public string Name { get; set; }
        public string Race { get; set; }
        public string Class { get; set; }
        public int LifePoints { get; set; }
        public int ManaPoints { get; set; }
        public int AtkPoints { get; set; }
        public int DefPoints { get; set; }
        public int StrPoints { get; set; }
        public int SpdPoints { get; set; }
        public int IntPoints { get; set; }
        public string Special { get; set; }

        /// <summary>
        /// the constructor method for this class
        /// </summary>
        public Hero(string name, string race, string _class, int lifePoints, int
        ↪  manaPoints, int atkPoints, int defPoints, int strPoints, int
        ↪  spdPoints, int intPoints, string special)
        {
            this.Name = name;
            this.Race = race;
            this.Class = _class;
            this.LifePoints = lifePoints;
            this.ManaPoints = manaPoints;
            this.AtkPoints = atkPoints;
            this.DefPoints = defPoints;
            this.StrPoints = strPoints;
            this.SpdPoints = spdPoints;
            this.IntPoints = intPoints;
            this.Special = special;
        }
    }
}
```

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

namespace U1_24_NR_ND
{
    static class IOUtils
    {
        /// <summary>
        /// reads heroes in from a filename
        /// </summary>
        /// <param name="fileName">the filename from which to read</param>
        /// <returns>a list of heroes</returns>
        public static List<Hero> ReadHeroes(string fileName)
        {
            List<Hero> output = new List<Hero>();

            string[] lines = new string[100];

            // file error handling
            if (System.IO.File.Exists(fileName))
            {
                lines = File.ReadAllLines(fileName, Encoding.UTF8);
            }
            else
            {
                Console.WriteLine("Failas nerastas. Programa negali veikti.");
                System.Environment.Exit(1); // exit code 1 means that the program
                ↪  did not run successfuly
            }

            if (lines.Length <= 0)
            {
                Console.WriteLine("Pateiktas tuščias failas. Programa negali
                ↪  veikti.");
                System.Environment.Exit(1); // exit code 1 means that the program
                ↪  did not run successfuly
            }

            foreach (string line in lines)
            {
                string[] values = line.Split(';');

                string name = values[0];
                string race = values[1];
                string _class = values[2];
                int lifePoints = int.Parse(values[3]);
                int manaPoints = int.Parse(values[4]);
                int atkPoints = int.Parse(values[5]);
                int defPoints = int.Parse(values[6]);
                int strPoints = int.Parse(values[7]);
                int spdPoints = int.Parse(values[8]);
                int intPoints = int.Parse(values[9]);
                string special = values[10];

                Hero heroToAdd = new Hero(
                    name,
                    race,
                    _class,
                    lifePoints,
                    manaPoints,
                    atkPoints,
```

```
                defPoints,
                strPoints,
                spdPoints,
                intPoints,
                special
            );

            output.Add(heroToAdd);

        }

        return output;
    }


    /// <summary>
    /// prints out a table of heroes when give a list of them as input
    /// </summary>
    /// <param name="input">the list of heroes to be used as input</param>
    public static void PrintHeroes(List<Hero> input)
    {
        // the amount of empty characters given for every value in the table
        List<int> tableSpacing = new List<int> {10, 14, 11, 4, 4, 4, 5, 2, 2,
        ↪  2, 16};


        PrintIndexedTableLine(tableSpacing, 11, '┌', '┬', '┐', '─');

        Console.WriteLine(
            "|{0,-10}|{1,-14}|{2,-11}|{3,-4}|{4,-4}|{5, -4}|{6, -5}|{7,
            ↪  -2}|{8, -2}|{9, -2}|{10, -16}|",
            "Vardas",
            "Rasė",
            "Klasė",
            "G.t.",
            "M.t.",
            "Ž.t.",
            "Gy.t.",
            "J.",
            "V.",
            "I.",
            "Ypat. galia"
        );

        PrintIndexedTableLine(tableSpacing, 11, '├', '┼', '┤', '─');

        for (int i = 0; i < input.Count; i++)
        {
            Hero hero = input[i];
            Console.WriteLine(
                "|{0,-10}|{1,-14}|{2,-11}|{3,-4}|{4,-4}|{5, -4}|{6, -5}|{7,
                ↪  -2}|{8, -2}|{9, -2}|{10, -16}|",
                hero.Name,
                hero.Race,
                hero.Class,
                hero.LifePoints,
                hero.ManaPoints,
                hero.AtkPoints,
                hero.DefPoints,
                hero.StrPoints,
                hero.SpdPoints,
                hero.IntPoints,
                Truncate(hero.Special, 12)
            );

            if (i == input.Count - 1)
```

```
            {
                PrintIndexedTableLine(tableSpacing, 11, '└', '┴', '┘', '─');
            }
            else
            {
                PrintIndexedTableLine(tableSpacing, 11, '├', '┼', '┤', '─');
            }
        }

    }

    /// <summary>
    /// prints out a list of heroes with some of their info omitted
    /// </summary>
    /// <param name="input">a list of heroes to be used as input</param>
    public static void PrintHeroesCompressed(List<Hero> input)
    {
        // the amount of empty characters given for every value in the table
        List<int> tableSpacing = new List<int> {18, 18, 18, 18};


        PrintIndexedTableLine(tableSpacing, 4, '┌', '┬', '┐', '─');

        Console.WriteLine(
            "| {0,-16} | {1,-16} | {2,-16} | {3,-16} |",
            "Vardas",
            "Rasė",
            "Klasė",
            "Gyvybės t."
        );

        PrintIndexedTableLine(tableSpacing, 4, '├', '┼', '┤', '─');

        for (int i = 0; i < input.Count; i++)
        {
            Hero hero = input[i];
            Console.WriteLine(
                "| {0,-16} | {1,-16} | {2,-16} | {3,-16} |",
                hero.Name,
                hero.Race,
                hero.Class,
                hero.LifePoints
            );

            if (i == input.Count - 1)
            {
                PrintIndexedTableLine(tableSpacing, 4, '└', '┴', '┘', '─');
            }
            else
            {
                PrintIndexedTableLine(tableSpacing, 4, '├', '┼', '┤', '─');
            }
        }

    }

    /// <summary>
    /// a method to truncate strings that are too long
    /// </summary>
    /// <param name="value">the string to truncate</param>
    /// <param name="maxChars">the maximum amount of chars to use before
    ↪  truncating</param>
    /// <returns></returns>
    private static string Truncate(string value, int maxChars)
    {
```

```
            return value.Length <= maxChars ? value : value.Substring(0, maxChars)
            ↪  + "...";
        }


        /// <summary>
        ///  a simple method to assist in creating text character based tables
        /// </summary>
        /// <param name="spacing">a list of ints which defines the amount of
        ↪  <paramref name="line"/> chars to put in between any of the other
        ↪  chars</param>
        /// <param name="columnCount">the amount of columns in the</param>
        /// <param name="leftEdge">the char used at the left edge of the
        ↪  table</param>
        /// <param name="middleEdge">the char used inbetween lines</param>
        /// <param name="rightEdge">the char used at the right edge or end of the
        ↪  line</param>
        /// <param name="line">the char used inbetween any and all other
        ↪  chars</param>
        private static void PrintIndexedTableLine(List<int> spacing, int
        ↪  columnCount, char leftEdge, char middleEdge, char rightEdge, char
        ↪  line)
        {

            Console.Write(leftEdge);

            for (int i = 0; i < columnCount; i++) {

                Console.Write(new string(line, spacing[i]));

                if (i == columnCount - 1)
                {
                    Console.WriteLine(rightEdge);
                }
                else
                {
                    Console.Write(middleEdge);
                }
            }
        }

        /// <summary>
        /// outputs a list of classes to a csv file
        /// </summary>
        /// <param name="fileName">the filename to which to output</param>
        /// <param name="classes">the list of classes</param>
        public static void OutputClassesToCSV(string fileName, List<String>
        ↪  classes)
        {
            string[] lines = classes.ToArray();

            File.WriteAllLines(fileName, lines, Encoding.UTF8);
        }

    }
}

//TaskUtils.cs
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace U1_24_NR_ND
```

```
{
    class TaskUtils
    {
        /// <summary>
        /// finds all heroes with the highest amount of health.
        /// </summary>
        /// <param name="input">a list of heroes</param>
        /// <returns>a list of heroes who have the highest amount of
        ↪  health</returns>
        public static List<Hero> FindHeroesWithHighestHealth(List<Hero> input)
        {
            List<Hero> output = new List<Hero>();

            foreach (Hero hero in input)
            {
                if (output.Count == 0)
                {
                    output.Add(hero);
                    continue;
                }

                Hero heroToCompare = output[0];

                if (hero.LifePoints > heroToCompare.LifePoints)
                {
                    output.Clear();
                    output.Add(hero);
                }
                else if (hero.LifePoints == heroToCompare.LifePoints)
                {
                    output.Add(hero);
                }

            }


            return output;
        }

        /// <summary>
        /// a method to find the heroes who have the smallest difference
        /// between their attack and defence points
        /// </summary>
        /// <param name="input">a list of heroes</param>
        /// <returns>the heroes who have the smallest difference between atk and
        ↪  def</returns>
        public static List<Hero> FindHeroesWithSmallestDifference(List<Hero>
        ↪  input)
        {
            List<Hero> output = new List<Hero>();

            foreach (Hero hero in input)
            {
                if (output.Count == 0)
                {
                    output.Add(hero);
                    continue;
                }

                Hero heroToCompare = output[0];

                int aDiff = Math.Abs(hero.AtkPoints-hero.DefPoints);
                int bDiff = Math.Abs(heroToCompare.AtkPoints -
                ↪  heroToCompare.DefPoints);

                if (aDiff < bDiff)
```

```
                {
                    output.Clear();
                    output.Add(hero);
                }
                else if (aDiff == bDiff)
                {
                    output.Add(hero);
                }
            }

            return output;
        }

        /// <summary>
        /// finds all unique hero classes when given a list of them as input
        /// </summary>
        /// <param name="input">the list of heroes</param>
        /// <returns>the unique classes</returns>
        public static List<String> FindUniqueClasses(List<Hero> input)
        {
            List<String> output = new List<String>();

            foreach (Hero hero in input)
            {
                if (!output.Contains(hero.Class))
                {
                    output.Add(hero.Class);
                }
            }
            return output;
        }
    }
}

//Program.cs
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace U1_24_NR_ND
{
    class Program
    {
        /// <summary>
        /// the main method for this program
        /// </summary>
        public static void Main(string[] args)
        {
            // read from file
            List<Hero> allHeroes = IOUtils.ReadHeroes("herojai.csv");

            // print out all heroes
            Console.WriteLine("Visi herojai:");
            IOUtils.PrintHeroes(allHeroes);

            // print out heroes with highest [LifePoints]
            Console.WriteLine("Herojai su didžiausiu kiekiu gyvybės taškų:");

            ↪  IOUtils.PrintHeroesCompressed(TaskUtils.FindHeroesWithHighestHealth(allHeroes

            // print out all heroes with the smallest difference between
            // [AtkPoints] and [DefPoints]
```

```
            Console.WriteLine("Herojai su mažiausiu skirtumu tarp žalos ir gynybos
            ↪  taškų:");

            ↪  IOUtils.PrintHeroes(TaskUtils.FindHeroesWithSmallestDifference(allHeroes));

            // find and output all unique classes
            List<String> uniqueClasses = TaskUtils.FindUniqueClasses(allHeroes);
            IOUtils.OutputClassesToCSV("Klasės.csv", uniqueClasses);
        }
    }
}
```

1.3 Pradiniai duomenys ir rezultatai

Pradiniai duomenys:

```
Aloyzas;Lokys;Kunigas;97;72;38;35;7;7;1;Gerai gamina maistą
Aloyzas;Žmogus;Dainininkas;42;69;82;73;4;5;8;Labai laimingas
Antanas;Varliažmogis;Kunigas;18;27;25;51;1;9;9;Labai gražios akys
Antanas;Varliažmogis;Kunigas;66;87;99;25;4;2;4;Ugnies valdymas
Petras;Driežažmogis;Burtininkas;21;55;20;17;5;1;1;Labai laimingas
Vardėnis;Driežažmogis;Magas;59;33;40;55;0;0;8;Labai gražios akys
Motiejus;Elfas;Lankininkas;35;44;45;40;3;7;5;Labai laimingas
Motiejus;Tamsusis elfas;Kunigas;86;70;26;69;8;9;7;Labai gražios akys
Motiejus;Elfas;Karys;57;71;95;51;4;7;2;Labai laimingas
Antanas;Driežažmogis;Riteris;33;6;91;80;3;4;7;Ugnies valdymas
Vardėnis;Lokys;Gydytojas;18;34;43;12;1;3;5;Ugnies valdymas
Motiejus;Tamsusis elfas;Lankininkas;75;63;18;22;1;2;7;Labai laimingas
Juozas;Šuo;Karys;12;56;86;38;6;7;4;Gerai gamina maistą
Vardėnis;Varliažmogis;Burtininkas;37;47;14;75;1;6;6;Gerai gamina maistą
Motiejus;Žmogus;Riteris;28;23;61;81;3;1;9;Labai gražios akys
Antanas;Elfas;Riteris;56;4;16;91;9;7;5;Labai laimingas
Juozas;Žmogus;Karys;97;17;74;69;5;7;5;Ugnies valdymas
Motiejus;Varliažmogis;Magas;39;74;21;31;7;5;0;Labai gražios akys
Petras;Žmogus;Riteris;46;64;92;83;8;6;9;Labai laimingas
Vardėnis;Elfas;Karys;61;77;81;26;1;1;9;Gerai gamina maistą
Petras;Driežažmogis;Gydytojas;92;35;99;37;3;4;0;Labai gražios akys
Juozas;Driežažmogis;Riteris;32;60;48;83;7;6;6;Labai laimingas
Aloyzas;Driežažmogis;Lankininkas;65;21;98;68;6;6;9;Labai gražios akys
Juozas;Driežažmogis;Gydytojas;91;51;63;23;4;5;4;Ugnies valdymas
Juozas;Zombis;Gydytojas;93;95;82;23;9;9;5;Labai gražios akys
```

Rezultatai:

```
Visi herojai:
```

Vardas	Rasė	Klasė	G.t.	M.t.	Ž.t.	Gy.t.	J.	V.	I.	Ypat. galia
Aloyzas	Lokys	Kunigas	97	72	38	35	7	7	1	Gerai gamina...
Aloyzas	Žmogus	Dainininkas	42	69	82	73	4	5	8	Labai laimin...
Antanas	Varliažmogis	Kunigas	18	27	25	51	1	9	9	Labai gražio...
Antanas	Varliažmogis	Kunigas	66	87	99	25	4	2	4	Ugnies valdy...
Petras	Driežažmogis	Burtininkas	21	55	20	17	5	1	1	Labai laimin...
Vardėnis	Driežažmogis	Magas	59	33	40	55	0	0	8	Labai gražio...
Motiejus	Elfas	Lankininkas	35	44	45	40	3	7	5	Labai laimin...
Motiejus	Tamsusis elfas	Kunigas	86	70	26	69	8	9	7	Labai gražio...
Motiejus	Elfas	Karys	57	71	95	51	4	7	2	Labai laimin...
Antanas	Driežažmogis	Riteris	33	6	91	80	3	4	7	Ugnies valdy...

Vardėnis	Lokys	Gydytojas	18	34	43	12	1	3	5	Ugnies valdy...
Motiejus	Tamsusis elfas	Lankininkas	75	63	18	22	1	2	7	Labai laimin...
Juozas	Šuo	Karys	12	56	86	38	6	7	4	Gerai gamina...
Vardėnis	Varliažmogis	Burtininkas	37	47	14	75	1	6	6	Gerai gamina...
Motiejus	Žmogus	Riteris	28	23	61	81	3	1	9	Labai gražio...
Antanas	Elfas	Riteris	56	4	16	91	9	7	5	Labai laimin...
Juozas	Žmogus	Karys	97	17	74	69	5	7	5	Ugnies valdy...
Motiejus	Varliažmogis	Magas	39	74	21	31	7	5	0	Labai gražio...
Petras	Žmogus	Riteris	46	64	92	83	8	6	9	Labai laimin...
Vardėnis	Elfas	Karys	61	77	81	26	1	1	9	Gerai gamina...
Petras	Driežažmogis	Gydytojas	92	35	99	37	3	4	0	Labai gražio...
Juozas	Driežažmogis	Riteris	32	60	48	83	7	6	6	Labai laimin...
Aloyzas	Driežažmogis	Lankininkas	65	21	98	68	6	6	9	Labai gražio...
Juozas	Driežažmogis	Gydytojas	91	51	63	23	4	5	4	Ugnies valdy...
Juozas	Zombis	Gydytojas	93	95	82	23	9	9	5	Labai gražio...

Herojai su didžiausiu kiekiu gyvybės taškų:

Vardas	Rasė	Klasė	Gyvybės t.
Aloyzas	Lokys	Kunigas	97
Juozas	Žmogus	Karys	97

Herojai su mažiausiu skirtumu tarp žalos ir gynybos taškų:

Vardas	Rasė	Klasė	G.t.	M.t.	Ž.t.	Gy.t.	J.	V.	I.	Ypat. galia
Aloyzas	Lokys	Kunigas	97	72	38	35	7	7	1	Gerai gamina...
Petras	Driežažmogis	Burtininkas	21	55	20	17	5	1	1	Labai laimin...

Klasės.csv:

```
Kunigas
Dainininkas
Burtininkas
Magas
Lankininkas
Karys
Riteris
Gydytojas
```

1.4 Dėstytojo pastabos

- Ataskaitos pavadinimas buvo ne pagal taisykles
- Dėstytotojo pareigos buvo ne tos
- Metodai nebuvo tinkamai bei visiškai aprašyti
- Nėra pateikti komentarai apie pradinių duomenų prasmę

Galutinis pažymys: 8

2 Skaičiavimų klasė

2.1 Darbo užduotis

Krepšinio rinktinė. urite ne tik šių, bet ir vienų ankstesniųjų metų į stovyklas pakviestų krepšininkų sąrašus. Keičiasi duomenų failų formatas. Pirmoje eilutėje metai, antroje –stovyklos pradžios data, trečioje –stovyklos pabaigos data. Toliau informacija apie krepšininkus pateikta tokiu pačiu formatu kaip L1 užduotyje.

- Sudarykite visų puolėjų, dalyvavusių rinktinės stovyklose, sąrašą ir ekrane atspausdinkite jų vardus, pavardes bei ūgį.
- Raskite aukščiausiąkrepšininką, ir ekrane atspausdinkite jo vardą, pavardę bei amžių. Jei yra keli, spausdinkite visus.
- Sudarykite sąrašą klubų, kuriuose žaidė kandidatai į rinktinę, ir įrašykite į failą „Klubai.csv“.

2.2 Programos tekstas

```
//Player.cs
using System;

namespace L2_ND
{
    class Player
    {
        public string Name { get; set; }
        public string Surname { get; set; }
        public int Age { get; set; }
        public int Height { get; set; }
        public string Position { get; set; }
        public string Club { get; set; }
        public bool IsPicked { get; set; }
        public bool IsCaptain { get; set; }
        public DateTime startDate { get; set; }
        public DateTime endDate { get; set; }

        /// <summary>
        /// a method to represent a player object with a string
        /// </summary>
        /// <returns>a string that describes a player</returns>
        public override string ToString()
        {
            return String.Format(
                "Žaidėjas: {0} {1}, Metai: {2}, Aukštis: {3}, Pozicija: {4},
                ↪  Klubas: {5}, Ar parinktas: {6}, Ar kapitonas: {7}, Stovyklos
                ↪  pradžia: {8}, Stovyklos pabaiga: {9}",
                this.Name,
                this.Surname,
                this.Age,
                this.Height,
                this.Position,
                this.Club,
                this.IsPicked,
                this.IsPicked,
                this.startDate.ToShortDateString(),
                this.endDate.ToShortDateString()
            );
        }

        /// <summary>
        /// compares this object to any other object
        /// </summary>
        /// <param name="other">object to compare against</param>
        /// <returns>True if other is equal to this object; False if object isn't
        ↪  equal.</returns>
        public override bool Equals(object other)
```

```
    {
        if (other == null || other.GetType() != this.GetType())
        {
            return false;
        }

        if (((Player)other).GetHashCode() != this.GetHashCode())
        {
            return false;
        }

        return true;
    }

    /// <summary>
    /// a rudimentary hashing method of this class
    /// </summary>
    /// <returns>an integer hash of this class</returns>
    public override int GetHashCode()
    {
        int hash = 0;

        string hString = String.Format(
            "{0}{1}{2}{3}{4}{5}",
            this.Name,
            this.Surname,
            this.Position,
            this.Club,
            this.startDate.ToShortDateString(),
            this.endDate.ToShortDateString()
        );


        foreach (char c in hString)
        {
            hash += (int)c;
        }

        hash += this.Age + this.Height;

        if (this.IsPicked) {
            hash += 5;
        }
        if (this.IsCaptain) {
            hash += 7;
        }

        return hash;
    }

    public Player(string name, string surname, int age, int height, string
    ↪  position, string club, bool isPicked, bool isCaptain, DateTime start,
    ↪  DateTime end) {
        this.Name = name;
        this.Surname = surname;
        this.Age = age;
        this.Height = height;
        this.Position = position;
        this.Club = club;
        this.IsPicked = isPicked;
        this.IsCaptain = isCaptain;
        this.startDate = start;
        this.endDate = end;
    }

}
```

```
}

//PlayerRegister.cs
using System;
using System.Collections.Generic;
using System.Linq;

namespace L2_ND
{
    class PlayerRegister
    {
        /// <summary>
        /// the main list of players to be manipulated in the register
        /// </summary>
        private List<Player> allPlayers;

        /// <summary>
        /// constructor method with a player list as an argument
        /// creates a new list of players and adds the argument to the register's
        ↪  list
        /// </summary>
        /// <param name="players">a list of players</param>
        public PlayerRegister(List<Player> players)
        {
            if (this.allPlayers == null)
            {
                this.allPlayers = new List<Player>();
            }

            this.allPlayers.AddRange(players);
        }

        /// <summary>
        /// constructor method without any arguments -- creates an empty list of
        ↪  players in the register
        /// </summary>
        public PlayerRegister()
        {
            this.allPlayers = new List<Player>();
        }

        /// <summary>
        /// adds a list of players to this register's player list
        /// </summary>
        /// <param name="playersToAdd">a list of players</param>
        public void AddRange(List<Player> playersToAdd)
        {
            this.allPlayers.AddRange(playersToAdd);
        }

        /// <summary>
        /// adds a player to this register's all players
        /// </summary>
        /// <param name="player">the player to add</param>
        public void Add(Player player)
        {
            this.allPlayers.Add(player);
        }


        /// <summary>
        /// turns the register to a string
        /// </summary>
        /// <returns>a string that describes the data held in the
        ↪  register</returns>
        public override string ToString()
```

```
{
    string output = "";

    output += String.Format(
        "Žaidėjų registras: Žaidėjų kiekis: {0}\n",
        this.PlayerCount()
    );

    output += "Visi Žaidėjai: \n";

    foreach (Player p in this.allPlayers)
    {
        output += String.Format(
            "{0}\n",
            p.ToString()
        );
    }

    return output;
}

/// <summary>
/// gets the amount of players in the register
/// </summary>
/// <returns>the player count</returns>
public int PlayerCount()
{
    return this.allPlayers.Count();
}

/// <summary>
/// gets a list of players by their year
/// </summary>
/// <returns>a list of players</returns>
public List<Player> GetPlayersByYear(int year)
{
    return this.allPlayers.Where((Player p) => p.startDate.Year ==
    ↪  year).ToList();
}

/// <summary>
/// get a list of all players who have been invited
/// </summary>
/// <returns>a list of players who have been invited</returns>
public List<Player> GetInvitedPlayers()
{
    List<Player> output = new List<Player>();

    foreach (Player player in this.allPlayers)
    {
        if (player.IsPicked == true)
        {
            output.Add(player);
        }
    }
    return output;
}

/// <summary>
/// get the tallest player(s) in the register
/// </summary>
/// <returns>a list of the tallest players</returns>
public List<Player> GetTallestPlayers()
{

    List<Player> output = new List<Player>();
```

```
            foreach (Player player in this.allPlayers)
            {
                if (output.Count == 0)
                {
                    output.Add(player);
                    continue;
                }

                Player playerToCompare = output[0];

                if (player > playerToCompare)
                {
                    output.Clear();
                    output.Add(player);
                }
                else if (player == playerToCompare)
                {
                    output.Add(player);
                }
            }
            return output;
        }


        /// <summary>
        /// gets a list of all of the attackers
        /// </summary>
        /// <returns>a list of all attackers</returns>
        public List<Player> GetAllAttackers()
        {
            List<Player> output = new List<Player>();

            foreach (Player player in this.allPlayers)
            {
                if (player.Position == "Attacker")
                {
                    output.Add(player);
                }
            }
            return output;
        }

        /// <summary>
        /// gets a list of all unique clubs from every invited player
        /// </summary>
        /// <returns>a list of strings</returns>
        public List<String> GetUniqueInvitedClubs()
        {
            List<string> output = new List<string>();

            foreach (Player player in this.GetInvitedPlayers())
            {
                string club = player.Club;

                if (output.Contains(club) == false)
                {
                    output.Add(club);
                }
            }

            return output;
        }
    }
}
```

```
//IOUtils.cs
using System;
using System.IO;
using System.Collections.Generic;
using System.Text;

namespace L2_ND
{
    static class IOUtils
    {
        /// <summary>
        /// reads players in from a filename
        /// </summary>
        /// <param name="fileName">the filename from which to read</param>
        /// <returns>a list of players</returns>
        public static List<Player> ReadPlayersFromFile(string fileName)
        {
            List<Player> output = new List<Player>();

            string[] lines = new string[150];

            DateTime startDate = new DateTime();
            DateTime endDate = new DateTime();

            // file error handling
            if (System.IO.File.Exists(fileName))
            {
                lines = File.ReadAllLines(fileName, Encoding.UTF8);
            }
            else
            {
                Console.WriteLine("Failas nerastas. Programa negali veikti.");
                System.Environment.Exit(1); // exit code 1 means that the program
                ↪  did not run successfuly
            }

            if (lines.Length <= 0)
            {
                Console.WriteLine("Pateiktas tuščias failas. Programa negali
                ↪  veikti.");
                System.Environment.Exit(1); // exit code 1 means that the program
                ↪  did not run successfuly
            }


            string year = lines[0];

            startDate = DateTime.Parse(String.Format("{0}-{1}", year, lines[1]));
            endDate = DateTime.Parse(String.Format("{0}-{1}", year, lines[2]));


            for (int i = 3; i < lines.Length; i++)
            {
                string line = lines[i];

                // basic support for comments
                // if a line in the input file starts with
                // '//', then ignore it
                if (line.StartsWith("//"))
                {
                    continue;
                }

                string[] values = line.Split(';');
```

```
            string name = values[0];
            string surname = values[1];
            int age = int.Parse(values[2]);
            int height = int.Parse(values[3]);
            string position = values[4];
            string club = values[5];
            bool isPicked = bool.Parse(values[6]);
            bool isCaptain = bool.Parse(values[7]);

            Player PlayerToAdd = new Player(
                name,
                surname,
                age,
                height,
                position,
                club,
                isPicked,
                isCaptain,
                startDate,
                endDate
            );

            output.Add(PlayerToAdd);

        }

        return output;
    }

    /// <summary>
    /// prints out a table of Players when give a list of them as input
    /// </summary>
    /// <param name="input">the list of Players to be used as input</param>
    public static void PrintPlayers(List<Player> input)
    {
        // the amount of empty characters given for every value in the table
        List<int> tableSpacing = new List<int> {10, 14, 3, 3, 10, 10, 10, 10};



        PrintIndexedTableLine(tableSpacing, 8, '┌', '┬', '┐', '─');

        Console.WriteLine(
            "|{0,-10}|{1,-14}|{2,-3}|{3,-3}|{4,-10}|{5, -10}|{6, -10}|{7,
            ↪  -10}|",
            "Vardas",
            "Pavardė",
            "Amž",
            "Au.",
            "Pozicija",
            "Klubas",
            "Išrinktas",
            "Kapitonas"
        );

        PrintIndexedTableLine(tableSpacing, 8, '├', '┼', '┤', '─');

        for (int i = 0; i < input.Count; i++)
        {
            Player player = input[i];

            Console.WriteLine(
                "|{0,-10}|{1,-14}|{2,-3}|{3,-3}|{4,-10}|{5, -10}|{6, -10}|{7,
                ↪  -10}|",
                player.Name,
                player.Surname,
```

```
                player.Age,
                player.Height,
                player.Position,
                player.Club,
                player.IsPicked,
                player.IsCaptain
            );

            if (i == input.Count - 1)
            {
                PrintIndexedTableLine(tableSpacing, 8, '└', '┴', '┘', '─');
            }
            else
            {
                PrintIndexedTableLine(tableSpacing, 8, '├', '┼', '┤', '─');
            }
        }
    }

    /// <summary>
    /// prints only the name, surname and age of a list of players
    /// </summary>
    /// <param name="input">a list of players</param>
    public static void PrintCondensedPlayersWithAge(List<Player> input)
    {
        // the amount of empty characters given for every value in the table
        List<int> tableSpacing = new List<int> {10, 14, 3};

        PrintIndexedTableLine(tableSpacing, 3, '┌', '┬', '┐', '─');

        Console.WriteLine(
            "|{0,-10}|{1,-14}|{2,-3}|",
            "Vardas",
            "Pavardė",
            "Amž"
        );

        PrintIndexedTableLine(tableSpacing, 3, '├', '┼', '┤', '─');

        for (int i = 0; i < input.Count; i++)
        {
            Player player = input[i];
            Console.WriteLine(
                "|{0,-10}|{1,-14}|{2,-3}|",
                player.Name,
                player.Surname,
                player.Age
            );

            if (i == input.Count - 1)
            {
                PrintIndexedTableLine(tableSpacing, 3, '└', '┴', '┘', '─');
            }
            else
            {
                PrintIndexedTableLine(tableSpacing, 3, '├', '┼', '┤', '─');
            }
        }
    }

    /// <summary>
    /// prints only the name, surname and height of a list of players
    /// </summary>
    /// <param name="input">a list of players</param>
    public static void PrintCondensedPlayersWithHeight(List<Player> input)
    {
```

```
    // the amount of empty characters given for every value in the table
    List<int> tableSpacing = new List<int> {10, 14, 3};


    PrintIndexedTableLine(tableSpacing, 3, '┌', '┬', '┐', '─');

    Console.WriteLine(
        "|{0,-10}|{1,-14}|{2,-3}|",
        "Vardas",
        "Pavardė",
        "Au."
    );

    PrintIndexedTableLine(tableSpacing, 3, '├', '┼', '┤', '─');

    for (int i = 0; i < input.Count; i++)
    {
        Player player = input[i];
        Console.WriteLine(
            "|{0,-10}|{1,-14}|{2,-3}|",
            player.Name,
            player.Surname,
            player.Height
        );

        if (i == input.Count - 1)
        {
            PrintIndexedTableLine(tableSpacing, 3, '└', '┴', '┘', '─');
        }
        else
        {
            PrintIndexedTableLine(tableSpacing, 3, '├', '┼', '┤', '─');
        }
    }
}

/// <summary>
/// a method to truncate strings that are too long
/// </summary>
/// <param name="value">the string to truncate</param>
/// <param name="maxChars">the maximum amount of chars to use before
↪  truncating</param>
/// <returns>a truncated string</returns>
private static string Truncate(string value, int maxChars)
{
    return value.Length <= maxChars ? value : value.Substring(0, maxChars)
    ↪  + "...";
}

/// <summary>
///  a simple method to assist in creating text character based tables
/// </summary>
/// <param name="spacing">a list of ints which defines the amount of
↪  <paramref name="line"/> chars to put in between any of the other
↪  chars</param>
/// <param name="columnCount">the amount of columns in the</param>
/// <param name="leftEdge">the char used at the left edge of the
↪  table</param>
/// <param name="middleEdge">the char used inbetween lines</param>
/// <param name="rightEdge">the char used at the right edge or end of the
↪  line</param>
/// <param name="line">the char used inbetween any and all other
↪  chars</param>
private static void PrintIndexedTableLine(List<int> spacing, int
↪  columnCount, char leftEdge, char middleEdge, char rightEdge, char
↪  line)
```

```
    {
        Console.Write(leftEdge);

        for (int i = 0; i < columnCount; i++) {

            Console.Write(new string(line, spacing[i]));

            if (i == columnCount - 1)
            {
                Console.WriteLine(rightEdge);
            }
            else
            {
                Console.Write(middleEdge);
            }
        }
    }


    /// <summary>
    /// outputs a list of strings to a csv file
    /// </summary>
    /// <param name="fileName">the filename to which to output</param>
    /// <param name="input">a list of strings</param>
    public static void OutputStringListToCSV(string fileName, List<String>
    ↪  input)
    {
        string[] lines = input.ToArray();

        File.WriteAllLines(fileName, lines, Encoding.UTF8);
    }


    /// <summary>
    /// outputs a list of players into a csv file
    /// </summary>
    /// <param name="fileName">the filename to which to write</param>
    /// <param name="players">a list of players</param>
    public static void OutputPlayersToCSV(string fileName, List<Player>
    ↪  players)
    {

        List<string> output = new List<string>();

        foreach (Player p in players)
        {
            string line;

            line = String.Format(
                "{0};{1};{2};{3};{4};{5};{6}",
                p.Name,
                p.Surname,
                p.Age,
                p.Height,
                p.Position,
                p.IsPicked,
                p.IsCaptain
            );

            output.Add(line);
        }

        File.WriteAllLines(fileName, output.ToArray(), Encoding.UTF8);
    }
}
```

```
}

//Program.cs
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace L2_ND
{
    class Program
    {
        static void Main(string[] args)
        {
            List<Player> allPlayers = new List<Player>();


             ↪  allPlayers.AddRange(IOUtils.ReadPlayersFromFile("2020-krepsininkai.csv"));

             ↪  allPlayers.AddRange(IOUtils.ReadPlayersFromFile("2019-krepsininkai.csv"));

            PlayerRegister reg = new PlayerRegister(allPlayers);

            // print out all players
            Console.Write(reg.ToString());

            // get and print out all attackers
            List<Player> allAttackers = reg.GetAllAttackers();
            Console.WriteLine("Visi puolėjai:");
            IOUtils.PrintCondensedPlayersWithHeight(allAttackers);

            // get and print out the tallest player(s)
            List<Player> tallestPlayers = reg.GetTallestPlayers();
            Console.WriteLine("Aukščiausi žaidėjai:");
            IOUtils.PrintCondensedPlayersWithHeight(tallestPlayers);

            // write all unique clubs to a file
            List<string> uniqueClubs = reg.GetUniqueInvitedClubs();
            IOUtils.OutputStringListToCSV("Klubai.csv", uniqueClubs);
        }
    }
}
```

2.3 Pradiniai duomenys ir rezultatai

2.3.1 Pirmas tikrinimas

krepsininkai-2020.csv:

```
2020
07-01
07-30
Aloyzas;Valančiūnas;33;182;Attacker;L. Rytas;True;False
Petras;Pavardėnis;28;197;Attacker;Kruojos;False;False
Antanas;Žukauskas;31;198;Attacker;Šaulys;True;False
Vardėnis;Valančiūnas;34;190;Striker;Žalgiris;False;False
Antanas;Jasikevičius;21;195;Attacker;L. Rytas;False;False
Vardėnis;Žukauskas;29;190;Sniper;Žalgiris;True;False
Aloyzas;Valančiūnas;29;195;Attacker;Žalgiris;True;False
```

krepsininkai-2019.csv:

```
2019
08-01
09-01
Motiejus;Sabonis;31;192;Striker;Šaulys;True;False
Antanas;Valančiūnas;22;187;Attacker;L. Rytas;True;False
Aloyzas;Pavardėnis;30;206;Defender;L. Rytas;False;False
Juozas;Valančiūnas;30;179;Striker;Šaulys;False;False
Juozas;Pavardėnis;33;207;Attacker;L. Rytas;True;True
Petras;Valančiūnas;27;191;Striker;Kruojos;True;False
```

Šitie duomenys yra skirti bendram tikrinimui atlikti. Išvestyje turėtų būti tik vienas aukščiausias žaidėjas, (Juozas Pavardėnis), turėtų būti 7 puolėjai, turėtų būti 4 unikalūs klubai (Šaulys, L. Rytas, Žalgiris, Kruojos).

Programos išvestis:

```
Žaidėjų registras: Žaidėjų kiekis: 13
Visi Žaidėjai:
Žaidėjas: Aloyzas Valančiūnas, Metai: 33, Aukštis: 182, Pozicija: Attacker, Klubas: L.
↪  Rytas, Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Petras Pavardėnis, Metai: 28, Aukštis: 197, Pozicija: Attacker, Klubas: Kruojos,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Antanas Žukauskas, Metai: 31, Aukštis: 198, Pozicija: Attacker, Klubas: Šaulys, Ar
↪  parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 7/1/2020, Stovyklos pabaiga:
↪  7/30/2020
Žaidėjas: Vardėnis Valančiūnas, Metai: 34, Aukštis: 190, Pozicija: Striker, Klubas:
↪  Žalgiris, Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020,
↪  Stovyklos pabaiga: 7/30/2020
Žaidėjas: Antanas Jasikevičius, Metai: 21, Aukštis: 195, Pozicija: Attacker, Klubas: L.
↪  Rytas, Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Vardėnis Žukauskas, Metai: 29, Aukštis: 190, Pozicija: Sniper, Klubas: Žalgiris,
↪  Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 7/1/2020, Stovyklos pabaiga:
↪  7/30/2020
Žaidėjas: Aloyzas Valančiūnas, Metai: 29, Aukštis: 195, Pozicija: Attacker, Klubas:
↪  Žalgiris, Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Motiejus Sabonis, Metai: 31, Aukštis: 192, Pozicija: Striker, Klubas: Šaulys, Ar
↪  parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Antanas Valančiūnas, Metai: 22, Aukštis: 187, Pozicija: Attacker, Klubas: L.
↪  Rytas, Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos
↪  pabaiga: 9/1/2019
Žaidėjas: Aloyzas Pavardėnis, Metai: 30, Aukštis: 206, Pozicija: Defender, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos
↪  pabaiga: 9/1/2019
Žaidėjas: Juozas Valančiūnas, Metai: 30, Aukštis: 179, Pozicija: Striker, Klubas: Šaulys, Ar
↪  parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Juozas Pavardėnis, Metai: 33, Aukštis: 207, Pozicija: Attacker, Klubas: L. Rytas,
↪  Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Petras Valančiūnas, Metai: 27, Aukštis: 191, Pozicija: Striker, Klubas: Kruojos,
↪  Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
```

Visi puolėjai:

Vardas	Pavardė	Au.
Aloyzas	Valančiūnas	182
Petras	Pavardėnis	197
Antanas	Žukauskas	198
Antanas	Jasikevičius	195
Aloyzas	Valančiūnas	195
Antanas	Valančiūnas	187
Juozas	Pavardėnis	207

Aukščiausi žaidėjai:

Vardas	Pavardė	Au.
Juozas	Pavardėnis	207

Klubai.csv:

```
L. Rytas
Šaulys
Žalgiris
Kruojos
```

2.3.2 Antras tikrinimas

krepsininkai-2020.csv:

```
2020
07-01
07-30
Antanas;Žukauskas;21;177;Defender;L. Rytas;False;False
Aloyzas;Sabonis;19;193;Striker;Šaulys;False;False
Petras;Jasikevičius;22;204;Sniper;Šaulys;False;False
Petras;Pavardėnis;28;209;Defender;L. Rytas;False;False
Motiejus;Žukauskas;20;205;Defender;L. Rytas;False;False
Vardėnis;Sabonis;26;193;Attacker;L. Rytas;False;False
Juozas;Valančiūnas;23;198;Sniper;Šaulys;True;False
Juozas;Žukauskas;25;185;Defender;L. Rytas;False;False
Aloyzas;Pavardėnis;20;205;Defender;Žalgiris;True;False
Vardėnis;Pavardėnis;19;178;Striker;L. Rytas;False;False
Aloyzas;Valančiūnas;29;208;Striker;L. Rytas;False;False
Antanas;Valančiūnas;32;191;Attacker;Žalgiris;True;False
Antanas;Sabonis;27;209;Sniper;L. Rytas;False;False
```

krepsininkai-2019.csv:

```
2019
08-01
09-01
Juozas;Žukauskas;21;191;Striker;Žalgiris;True;True
Juozas;Pavardėnis;29;194;Attacker;Žalgiris;True;False
Vardėnis;Valančiūnas;28;182;Sniper;L. Rytas;False;False
Aloyzas;Jasikevičius;32;179;Defender;L. Rytas;False;False
Motiejus;Sabonis;32;179;Striker;Šaulys;False;False
Petras;Sabonis;23;191;Attacker;Žalgiris;False;False
Juozas;Pavardėnis;18;184;Sniper;L. Rytas;False;False
Vardėnis;Pavardėnis;19;201;Defender;Žalgiris;True;False
Juozas;Pavardėnis;28;202;Defender;Šaulys;True;False
Petras;Jasikevičius;33;177;Defender;Kruojos;False;False
Vardėnis;Sabonis;25;192;Striker;Šaulys;True;False
```

Šie įvesties duomenys yra skirti tikrinti „Klubai.csv“ išvestį. Yra įvesti tokie patys klubai kaip ir pirmajame tikrinime, tačiau jie yra pakeisti taip, kad žaidėjai yra priimami tik iš „Žalgirio“ ir „Šaulio“ klubų. Kitaip tariant, Į „Klubai.csv“ turėtų buti išvesti tik du klubai: „Šaulys“ ir „Žalgiris“.

Programos išvestis:

```
Žaidėjų registras: Žaidėjų kiekis: 24
Visi Žaidėjai:
Žaidėjas: Antanas Žukauskas, Metai: 21, Aukštis: 177, Pozicija: Defender, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Aloyzas Sabonis, Metai: 19, Aukštis: 193, Pozicija: Striker, Klubas: Šaulys, Ar
↪  parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos pabaiga:
↪  7/30/2020
Žaidėjas: Petras Jasikevičius, Metai: 22, Aukštis: 204, Pozicija: Sniper, Klubas: Šaulys, Ar
↪  parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos pabaiga:
↪  7/30/2020
Žaidėjas: Petras Pavardėnis, Metai: 28, Aukštis: 209, Pozicija: Defender, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Motiejus Žukauskas, Metai: 20, Aukštis: 205, Pozicija: Defender, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Vardėnis Sabonis, Metai: 26, Aukštis: 193, Pozicija: Attacker, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Juozas Valančiūnas, Metai: 23, Aukštis: 198, Pozicija: Sniper, Klubas: Šaulys, Ar
↪  parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 7/1/2020, Stovyklos pabaiga:
↪  7/30/2020
Žaidėjas: Juozas Žukauskas, Metai: 25, Aukštis: 185, Pozicija: Defender, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Aloyzas Pavardėnis, Metai: 20, Aukštis: 205, Pozicija: Defender, Klubas: Žalgiris,
↪  Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 7/1/2020, Stovyklos pabaiga:
↪  7/30/2020
Žaidėjas: Vardėnis Pavardėnis, Metai: 19, Aukštis: 178, Pozicija: Striker, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Aloyzas Valančiūnas, Metai: 29, Aukštis: 208, Pozicija: Striker, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Antanas Valančiūnas, Metai: 32, Aukštis: 191, Pozicija: Attacker, Klubas:
↪  Žalgiris, Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 7/1/2020, Stovyklos
↪  pabaiga: 7/30/2020
Žaidėjas: Antanas Sabonis, Metai: 27, Aukštis: 209, Pozicija: Sniper, Klubas: L. Rytas, Ar
↪  parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 7/1/2020, Stovyklos pabaiga:
↪  7/30/2020
Žaidėjas: Juozas Žukauskas, Metai: 21, Aukštis: 191, Pozicija: Striker, Klubas: Žalgiris, Ar
↪  parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Juozas Pavardėnis, Metai: 29, Aukštis: 194, Pozicija: Attacker, Klubas: Žalgiris,
↪  Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Vardėnis Valančiūnas, Metai: 28, Aukštis: 182, Pozicija: Sniper, Klubas: L. Rytas,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos
↪  pabaiga: 9/1/2019
Žaidėjas: Aloyzas Jasikevičius, Metai: 32, Aukštis: 179, Pozicija: Defender, Klubas: L.
↪  Rytas, Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos
↪  pabaiga: 9/1/2019
Žaidėjas: Motiejus Sabonis, Metai: 32, Aukštis: 179, Pozicija: Striker, Klubas: Šaulys, Ar
↪  parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Petras Sabonis, Metai: 23, Aukštis: 191, Pozicija: Attacker, Klubas: Žalgiris, Ar
↪  parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Juozas Pavardėnis, Metai: 18, Aukštis: 184, Pozicija: Sniper, Klubas: L. Rytas, Ar
↪  parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Vardėnis Pavardėnis, Metai: 19, Aukštis: 201, Pozicija: Defender, Klubas:
↪  Žalgiris, Ar parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos
↪  pabaiga: 9/1/2019
Žaidėjas: Juozas Pavardėnis, Metai: 28, Aukštis: 202, Pozicija: Defender, Klubas: Šaulys, Ar
↪  parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
Žaidėjas: Petras Jasikevičius, Metai: 33, Aukštis: 177, Pozicija: Defender, Klubas: Kruojos,
↪  Ar parinktas: False, Ar kapitonas: False, Stovyklos pradžia: 8/1/2019, Stovyklos
↪  pabaiga: 9/1/2019
```

```
Žaidėjas: Vardėnis Sabonis, Metai: 25, Aukštis: 192, Pozicija: Striker, Klubas: Šaulys, Ar
↪  parinktas: True, Ar kapitonas: True, Stovyklos pradžia: 8/1/2019, Stovyklos pabaiga:
↪  9/1/2019
```

Visi puolėjai:

Vardas	Pavardė	Au.
Vardėnis	Sabonis	193
Antanas	Valančiūnas	191
Juozas	Pavardėnis	194
Petras	Sabonis	191

Aukščiausi žaidėjai:

Vardas	Pavardė	Au.
Petras	Pavardėnis	209
Antanas	Sabonis	209

Klubai.csv:

```
Šaulys
Žalgiris
```

2.4 Dėstytojo pastabos

3 Konteineris

3.1 Darbo užduotis

3.2 Programos tekstas

3.3 Pradiniai duomenys ir rezultatai

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 Paveldėjimas

5.1 Darbo užduotis

5.2 Programos tekstas

5.3 Pradiniai duomenys ir rezultatai

5.4 Dėstytojo pastabos