



**Kauno technologijos universitetas**

Informatikos fakultetas

## **Objektinis programavimas I (P175B118)**

Laboratorinių darbų ataskaita

---

**Aistis Jakutonis IFF-3/1**

Studentas

---

**Kęstutis Simonavičius**

Dėstytojas

## TURINYS

<b>1. Duomenų klasė .....</b>	<b>4</b>
1.1. Darbo užduotis .....	4
1.2. Programos tekstas.....	4
1.3. Pradiniai duomenys ir rezultatai.....	10
1.3.1 Duomenys ir rezultatai 1 .....	10
1.3.2 Duomenys ir rezultatai 2 .....	11
1.4. Dėstytojo pastabos.....	12
<b>2. Skaičiavimų klasė .....</b>	<b>13</b>
2.1. Darbo užduotis .....	13
2.2. Programos tekstas.....	13
2.3. Pradiniai duomenys ir rezultatai.....	23
2.3.1 Duomenys ir rezultatai 1 .....	23
2.3.2 Duomenys ir rezultatai 2 .....	25
2.4. Dėstytojo pastabos.....	27
<b>3. Konteineris .....</b>	<b>28</b>
3.1. Darbo užduotis .....	28
3.2. Programos tekstas.....	28
3.3. Pradiniai duomenys ir rezultatai.....	43
3.3.1 Duomenys ir rezultatai 1 .....	43
3.3.2 Duomenys ir rezultatai 2 .....	45
3.4. Dėstytojo pastabos.....	47
<b>4. Teksto analizė ir redagavimas .....</b>	<b>48</b>
4.1. Darbo užduotis .....	48
4.2. Programos tekstas.....	48
4.3. Pradiniai duomenys ir rezultatai.....	57
4.3.1 Duomenys ir rezultatai 1 .....	57

4.3.2	Duomenys ir rezultatai 2 .....	58
4.4.	Dėstytojo pastabos.....	58
<b>5.</b>	<b>Paveldėjimas.....</b>	<b>59</b>
5.1.	Darbo užduotis .....	59
5.2.	Programos tekstas.....	59
5.3.	Pradiniai duomenys ir rezultatai.....	79
5.3.1	Duomenys ir rezultatai 1 .....	79
5.3.2	Duomenys ir rezultatai 2 .....	82
5.4.	Dėstytojo pastabos.....	85

# 1. Duomenų klasė

## 1.1. Darbo užduotis

**U1-24. Kompiuterinis žaidimas.** Kuriate „fantasy“ kompiuterinį žaidimą. Duomenų faile turite informacija apie žaidimo herojus: vardas, rasė, klasė, gyvybės taškai, mana, žalos taškai, gynybos taškai, jéga, 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 išrašykite į failą „Klasės.csv“. Klasių pavadinimai neturi kartotis.

## 1.2. Programos tekstas

```
using System.Text; //Library used for text encoding

namespace U1_24KompiuterinisZaidimas
{
    /*U1-24. Kompiuterinis žaidimas. Kuriate „fantasy“ kompiuterinį žaidimą.
     *Duomenų faile turite informacija apie žaidimo herojus: vardas, rasė,
     *klasė, gyvybės taškai, mana, žalos taškai, gynybos taškai, jéga, 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 išrašykite į
     *failą „Klasės.csv“. Klasių pavadinimai neturi kartotis.
    */

    //The class in which the constructor is created
    public class Hero
    {
        public string name { get; }
        public string race { get; }
        public int number { get; }
        public int health { get; }
        public int mana { get; }
        public int damage { get; }
        public int defend { get; }
        public int strength { get; }
        public int speed { get; }
        public int intellect { get; }
        public string power { get; }

        //Creates a hero constructor
        public Hero(string name, string race, int number, int health, int mana,
                   int damage, int defend, int strength, int speed, int intellect,
                   string power)
        {
            this.name = name;
```

```

        this.race = race;
        this.number = number;
        this.health = health;
        this.mana = mana;
        this.damage = damage;
        this.defend = defend;
        this.strength = strength;
        this.speed = speed;
        this.intellect = intellect;
        this.power = power;
    }
}

//A class that performs scans and prints
public class InputOutput
{
    //A method that reads heroes and their data from the "Herojus.csv" file
    public static List<Hero> ReadHeroes(string fileName)
    {
        List<Hero> heroes = new List<Hero>();

        //Reads all lines from a file in UTF-8 encoding
        string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);

        //Parses each line
        foreach (string line in Lines)
        {
            string[] values = line.Split(";");
            string name = values[0];
            string race = values[1];
            int number = int.Parse(values[2]);
            int health = int.Parse(values[3]);
            int mana = int.Parse(values[4]);
            int damage = int.Parse(values[5]);
            int defend = int.Parse(values[6]);
            int strength = int.Parse(values[7]);
            int speed = int.Parse(values[8]);
            int intellect = int.Parse(values[9]);
            string power = values[10];

            //Creates new hero object
            Hero hero = new Hero(name, race, number, health, mana, damage,
                defend, strength, speed, intellect, power);

            //Adds the created Hero object to the list of heroes
            heroes.Add(hero);
        }

        return heroes;
    }

    //A method that prints all heroes and their data to the console
    public static void PrintAllHeroes(List<Hero> heroes)
    {
        //A table is created to store the data
        Console.WriteLine("Registro informacija:");
        Console.WriteLine(new string('-', 146));
        Console.WriteLine("| {0,-12} | {1,-15} | {2,-5} | {3,-14} | " +
            "{4,-4} | {5,-12} | {6,-14} | {7,-4} | {8,-8} | {9,-10} | " +
            "{10,-14} |", "Vardas", "Rasé", "Klasé", "Gyvybés taškai",
            "Mana", "Žalos taškai", "Gynybos taškai", "Jéga", "Vikrumas",

```

```

        "Intelektas", "Ypatinga galia");
Console.WriteLine(new string('-', 146));

foreach (Hero hero in heroes)
{
    Console.WriteLine("| {0,-12} | {1,-15} | {2,5} | {3,14} | " +
        "{4,4} | {5,12} | {6,14} | {7,4} | {8,8} | {9,10} | " +
        "{10,-14} |", hero.name, hero.race, hero.number,
        hero.health, hero.mana, hero.damage, hero.defend,
        hero.strength, hero.speed, hero.intellect, hero.power);
}

Console.WriteLine(new string('-', 146));
}

//A method that prints all heroes and their data to a txt file
//The data is stored in a table
public static void PrintAllHeroesToTxt(string fileNameTxt, List<Hero>
heroes)
{
    string[] lines = new string[heroes.Count + 5];

    lines[0] = String.Format("Registro informacija:");
    lines[1] = String.Format(new string('-', 146));
    lines[2] = String.Format("| {0,-12} | {1,-15} | {2,-5} | {3,-14}" +
        " | {4,-4} | {5,-12} | {6,-14} | {7,-4} | {8,-8} | " +
        "{9,-10} | {10,-14} |", "Vardas", "Rasé", "Klasé",
        "Gyvybés taškai", "Mana", "žalos taškai", "Gynybos taškai",
        "Jéga", "Vikrumas", "Intelektas", "Ypatinga galia");
    lines[3] = String.Format(new string('-', 146));

    int x = 4;

    foreach (Hero hero in heroes)
    {
        lines[x] = String.Format("| {0,-12} | {1,-15} | {2,5} | " +
            "{3,14} | {4,4} | {5,12} | {6,14} | {7,4} | {8,8} | " +
            "{9,10} | {10,-14} |", hero.name, hero.race, hero.number,
            hero.health, hero.mana, hero.damage, hero.defend,
            hero.strength, hero.speed, hero.intellect, hero.power);
        x++;
    }

    lines[x] = String.Format(new string('-', 146));

    //Prints on each line of the file
    File.WriteAllLines(fileNameTxt, lines, Encoding.UTF8);
}

//The method prints hero(s) with the most life points and their data
//(name, race, class, life points)
//The data is placed in a table
public static void PrintHealthiest(List<Hero> heroes)
{
    //Heroes with the most life points are listed in the
    //strongest list
    List<Hero> strongest = Tasks.FindMostHealth(heroes);

    Console.WriteLine("Daugiausiai gyvybés taškų:");
    Console.WriteLine(new string('-', 59));
    Console.WriteLine("| {0,-12} | {1,-15} | {2,5} | {3,14} |",

```

```

        "Vardas", "Rasé", "Klasé", "Gyvybés taškai");
Console.WriteLine(new string('-', 59));

    foreach (Hero hero in strongest)
    {
        Console.WriteLine("| {0,-12} | {1,-15} | {2,5} | {3,14} |",
            hero.name, hero.race, hero.number, hero.health);
    }

    Console.WriteLine(new string('-', 59));
}

//The method prints the hero(s) with the smallest difference
//(defense points - damage points)
//All hero data is printed
//The data is placed in a table
public static void PrintWithSmallestDifference(List<Hero> heroes)
{
    List<Hero> strongest = Tasks.FindBalance(heroes);

    Console.WriteLine("Mažiausias skirtumas tarp gynybos ir žalos " +
        "taškų:");
    Console.WriteLine(new string('-', 146));
    Console.WriteLine("| {0,-12} | {1,-15} | {2,-5} | {3,-14} | " +
        "| {4,-4} | {5,-12} | {6,-14} | {7,-4} | {8,-8} | {9,-10} | " +
        "| {10,-14} |", "Vardas", "Rasé", "Klasé", "Gyvybés taškai",
        "Mana", "Žalos taškai", "Gynybos taškai", "Jéga", "Vikrumas",
        "Intelektas", "Ypatinga galia");
    Console.WriteLine(new string('-', 146));

    foreach (Hero hero in strongest)
    {
        Console.WriteLine("| {0,-12} | {1,-15} | {2,5} | {3,14} | " +
            "| {4,4} | {5,12} | {6,14} | {7,4} | {8,8} | {9,10} | " +
            "| {10,-14} |", hero.name, hero.race, hero.number,
            hero.health, hero.mana, hero.damage, hero.defend,
            hero.strength, hero.speed, hero.intellect, hero.power);
    }

    Console.WriteLine(new string('-', 146));
}

//The method prints hero classes (without duplicates) to
//"/"Klasės.csv" file
public static void PrintNumbers(string fileName, List<Hero> heroes)
{
    //Different hero classes are taken from the Tasks class and drafted
    //to a numbers list
    List<int> numbers = Tasks.FindClasses(heroes);
    string[] lines = new string[numbers.Count + 1];

    lines[0] = String.Format("Heroju klasės:");

    for (int i = 0; i < numbers.Count; i++)
    {
        lines[i + 1] = String.Format("{0}", numbers[i]);
    }

    //Prints to all lines of the file
    File.WriteAllLines(fileName, lines, Encoding.UTF8);
}

```

```

}

//The class in which the calculations are performed
public class Tasks
{
    //A private method for calculating the maximum number of life points
    private static int FindHugeHealth(List<Hero> heroes)
    {
        //Life points are assumed to be greater than -1
        int strong = -1;

        foreach (Hero hero in heroes)
        {
            if (hero.health > strong)
            {
                strong = hero.health;
            }
        }
        return strong;
    }

    //A method that finds all heroes with the highest life points
    public static List<Hero> FindMostHealth(List<Hero> heroes)
    {
        //A new list is created to contain all the heroes
        //having the highest amount of life points
        List<Hero> healthiest = new List<Hero>();
        int strong = FindHugeHealth(heroes);

        foreach (Hero hero in heroes)
        {
            if (strong == hero.health)
            {
                //A hero object matching the condition is added to
                //the new list
                healthiest.Add(hero);
            }
        }
        return healthiest;
    }

    //A private method for finding the smallest difference
    //(defense points - damage points) between all heroes
    private static double FindSmallestDifference(List<Hero> heroes)
    {
        double difference = heroes[0].defend - heroes[0].damage;

        foreach (Hero hero in heroes)
        {
            if (difference > (hero.defend - hero.damage))
            {
                difference = hero.defend - hero.damage;
            }
        }

        return difference;
    }

    //A method to find all heroes with the smallest difference
    //(defense points - damage points) and put them into one list
    public static List<Hero> FindBalance(List<Hero> heroes)
}

```

```

    {
        //Creates a new list to hold the selected heroes
        List<Hero> difference = new List<Hero>();
        double strong = FindSmallestDifference(heroes);

        foreach (Hero hero in heroes)
        {
            if (strong == hero.defend - hero.damage)
            {
                //Heroes that meet the condition are added to the new list
                difference.Add(hero);
            }
        }
        return difference;
    }

    //A method that searches for hero classes and compares them
    //to find identical classes
    public static List<int> FindClasses(List<Hero> heroes)
    {
        // A new list is created, which will contain hero classes that do not
        //overlap with the previous ones

        List<int> numbers = new List<int>();

        foreach (Hero hero in heroes)
        {
            int nr = hero.number;

            if (!numbers.Contains(nr))
            {
                //Classes that match the condition are added to the new list
                numbers.Add(nr);
            }
        }
        return numbers;
    }
}

class Program
{
    static void Main(string[] args)
    {
        //Heroes and their data are read from the "Herojus.csv" file
        //and then are placed into the allHeroes list
        List<Hero> allHeroes =
        InputOutput.ReadHeroes(@"../../../../Herojus.csv");

        //All heroes and their data are printed to the console in a table
        InputOutput.PrintAllHeroes(allHeroes);

        //All heroes and their data in the table are printed to the
        //"Duomenys.txt" file
        string fileNameTxt = "Duomenys.txt";
        InputOutput.PrintAllHeroesToTxt(fileNameTxt, allHeroes);

        //A blank line is printed to avoid mixing results
        Console.WriteLine();

        //The results of the first task are printed to the console
        //(the heroes with the most health points)
    }
}

```

```

        InputOutput.PrintHealthiest(allHeroes);

        //A blank line is printed to avoid mixing results
        Console.WriteLine();

        // The results of the second task are printed to the console (the
        //heroes with the smallest difference (defense points - damage
        //points))
        InputOutput.PrintWithSmallestDifference(allHeroes);

        //Third task results (all different heroes classes) are printed to
        //"Klases.csv" file
        string fileName = "Klases.csv";
        InputOutput.PrintNumbers(fileName, allHeroes);
    }
}
}

```

### 1.3. Pradiniai duomenys ir rezultatai

#### 1.3.1 Duomenys ir rezultatai 1

Pradinis failas:

```

Aistis;Europietis;13;100;50;55;50;100;100;100;Lietimas
Rugile;Azijiete;12;150;60;70;60;150;150;150;Uostymas
Matas;Amerikietis;14;90;70;85;70;110;110;110;Klausa
Rokas;Indas;11;150;100;120;100;120;120;120;Rega
Tada;Australas;10;150;90;120;90;130;130;130;Liepsna

```

Šiais duomenimis tikrinama ar veikia daugiausiai gyvybės taškų turinčių herojų atpažinimas (trys herojai turi po 150 gyvybės taškų). Yra vienas herojus turintis mažiausią skirtumą (gynybos taškai – žalos taškai). Yra penkios skirtinges klasės.

Duomenys „.txt“ faile:

Duomenys.txt										
Registro informacija:										
Vardas	Rasė	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Aistis	Europietis	13	100	50	55	50	100	100	100	Lietimas
Rugile	Azijiete	12	150	60	70	60	150	150	150	Uostymas
Matas	Amerikietis	14	90	70	85	70	110	110	110	Klausa
Rokas	Indas	11	150	100	120	100	120	120	120	Rega
Tada	Australas	10	150	90	120	90	130	130	130	Liepsna

Rezultatai (Konsolėje):

Registro informacija:											
Vardas	Rasė	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	Europietis	13	100	50	55	50	100	100	100	Lietimas	
Rugile	Azijiete	12	150	60	70	60	150	150	150	Uostymas	
Matas	Amerikietis	14	90	70	85	70	110	110	110	Klausa	
Rokas	Indas	11	150	100	120	100	120	120	120	Rega	
Tada	Australas	10	150	90	120	90	130	130	130	Liepsna	

  

Daugiausiai gyvybės taškų:											
Vardas	Rasė	Klasė	Gyvybės taškai								
Rugile	Azijiete	12	150								
Rokas	Indas	11	150								
Tada	Australas	10	150								

  

Mažiausias skirtumas tarp gynybos ir žalos taškų:											
Vardas	Rasė	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Tada	Australas	10	150	90	120	90	130	130	130	130	Liepsna

Rezultatai (Klases.csv):

## Klases

Herojų klasės:	
	13
	12
	14
	11
	10

### 1.3.2 Duomenys ir rezultatai 2

Pradinis failas:

```
Aistis;Europietis;13;100;50;55;50;100;100;100;Lietimas
Rugile;Azijiete;11;150;60;65;60;150;150;150;Uostymas
Matas;Amerikietis;14;90;70;85;70;110;110;110;Klausa
Rokas;Indas;11;150;100;120;90;120;120;120;Rega
Tadas;Australas;10;100;90;120;90;130;130;130;Liepsna
```

Šiais duomenimis tikrinama ar tinkamai veikia herojų klasės atrinkimas bei skirtumo (gynybos taškai – žalos taškai) apskaičiavimo funkcijos. (Yra trys skirtingos klasės, yra 2 herojai su mažiausiu skirtumu, yra 2 herojai su didžiausiu gyvybių skaičiumi).

Duomenys „.txt“ faile:

Duomenys.txt

Registro informacija:											
Vardas	Rasė	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	Europietis	13	100	50	55	50	100	100	100	Lietimas	
Rugile	Azijiete	11	150	60	65	60	150	150	150	Uostymas	
Matas	Amerikietis	11	90	70	85	70	110	110	110	Klausa	
Rokas	Indas	11	150	100	120	90	120	120	120	Rega	
Tadas	Australas	10	100	90	120	90	130	130	130	Liepsna	

Rezultatai (Konsolėje):

Registro informacija:											
Vardas	Rasė	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	Europietis	13	100	50	55	50	100	100	100	Lietimas	
Rugile	Azijiete	11	150	60	65	60	150	150	150	Uostymas	
Matas	Amerikietis	11	90	70	85	70	110	110	110	Klausa	
Rokas	Indas	11	150	100	120	90	120	120	120	Rega	
Tadas	Australas	10	100	90	120	90	130	130	130	Liepsna	

  

Daugiausiai gyvybės taškų:											
Vardas	Rasė	Klasė	Gyvybės taškai								
Rugile	Azijiete	11	150								
Rokas	Indas	11	150								

  

Mažiausias skirtumas tarp gynybos ir žalos taškų:											
Vardas	Rasė	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Rokas	Indas	11	150	100	120	90	120	120	120	Rega	
Tadas	Australas	10	100	90	120	90	130	130	130	Liepsna	

Rezultatai (Klases.csv):

Klases	
Herojų klasės:	
13	
11	
10	

#### 1.4. Dėstytojo pastabos

Ateityje geriau naudoti standartinį šabloną rašant komentarus

## 2. Skaičiavimų klasė

### 2.1. Darbo užduotis

**U2-24. Kompiuterinis žaidimas.** Sugrupavote herojus pagal dvi rases, ir surašėte jų duomenis į skirtinges failus. Duomenų formatas dabar tokis: pirmoje eilutėje – rasės pavadinimas. Antroje – pradinis miestas. Toliau pateikta informacija tokiu pačiu formatu kaip L1 užduotyje, tik nebéra rasės stulpelio.

- Sudarykite visų herojų klasės sąrašą, klasės pavadinimus išrašykite į failą „Klasės.csv“.
- Raskite, kokių klasės herojų „trūksta“ kiekvienai rasei. Į failą „Trūkstamų.csv“ išrašykite kiekvienos rasės pavadinimą, ir trūkstamų klasės sąrašą. Jei rasė turi bent po vieną kiekvienos klasės atstovą, parašykite žodį „VISI“.
- Raskite, kurioje rasėje yra stipriausias herojus: herojus stiprumą rodo gyvybės ir gynybos taškų suma sumažinta žalos taškais. Ekrane atspausdinkite visus herojaus duomenis ir jo rasę.

### 2.2. Programos tekstas

```
namespace U1_24KompiuterinisZaidimas
{
    /// <summary>
    /// The class in which the constructor is created
    /// </summary>
    public class Hero
    {
        public string race { get; }
        public string city { get; }
        public string name { get; }
        public int number { get; }
        public int health { get; }
        public int mana { get; }
        public int damage { get; }
        public int defend { get; }
        public int strength { get; }
        public int speed { get; }
        public int intellect { get; }
        public string power { get; }

        /// <summary>
        /// Creates a hero constructor
        /// </summary>
        /// <param name="race"></param>
        /// <param name="city"></param>
        /// <param name="name"></param>
        /// <param name="number"></param>
        /// <param name="health"></param>
        /// <param name="mana"></param>
        /// <param name="damage"></param>
        /// <param name="defend"></param>
        /// <param name="strength"></param>
        /// <param name="speed"></param>
        /// <param name="intellect"></param>
        /// <param name="power"></param>
        public Hero(string race, string city, string name, int number,
```

```

        int health, int mana, int damage, int defend, int strength,
        int speed, int intellect, string power)
{
    this.race = race;
    this.city = city;
    this.name = name;
    this.number = number;
    this.health = health;
    this.mana = mana;
    this.damage = damage;
    this.defend = defend;
    this.strength = strength;
    this.speed = speed;
    this.intellect = intellect;
    this.power = power;
}

/// <summary>
/// Overriding operator "<"
/// </summary>
/// <param name="strength"></param>
/// <param name="hero"></param>
/// <returns></returns>
public static bool operator <(double strength, Hero hero)
{
    return strength < hero.GetStrength();
}

/// <summary>
/// Overriding operator ">"
/// </summary>
/// <param name="strength"></param>
/// <param name="hero"></param>
/// <returns></returns>
public static bool operator >(double strength, Hero hero)
{
    return strength > hero.GetStrength();
}

/// <summary>
/// Overrides ToString method
/// </summary>
/// <returns></returns>
public override string ToString()
{
    string line;

    line = string.Format("| {0,-12} | {1,5} | {2,14} | " +
        "{3,4} | {4,12} | {5,14} | {6,4} | {7,8} | {8,10} | " +
        "{9,-14} |", this.name, this.number,
        this.health, this.mana, this.damage, this.defend,
        this.strength, this.speed, this.intellect, this.power);

    return line;
}

/// <summary>
/// Counts strength
/// </summary>
/// <returns></returns>
public double GetStrength()

```

```

        {
            return this.health + this.defend - this.damage;
        }
    }

using System.Text; //Library used for text encoding

namespace U1_24KompiuterinisZaidimas
{
    /// <summary>
    /// A class that performs scans and prints
    /// </summary>
    public class InputOutput
    {
        /// <summary>
        /// A method that reads heroes and their data from the
        /// "Herojus.csv" file
        /// </summary>
        /// <param name="fileName"></param>
        /// <returns></returns>
        public static HeroRegister ReadHeroes(string fileName)
        {
            HeroRegister heroes = new HeroRegister();

            //Reads all lines from a file in UTF-8 encoding
            string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);

            string race = Lines[0];
            string city = Lines[1];

            //Parses each line
            for (int i = 2; i < Lines.Length; i++)
            {
                string[] values = Lines[i].Split(";");
                string name = values[0];
                int number = int.Parse(values[1]);
                int health = int.Parse(values[2]);
                int mana = int.Parse(values[3]);
                int damage = int.Parse(values[4]);
                int defend = int.Parse(values[5]);
                int strength = int.Parse(values[6]);
                int speed = int.Parse(values[7]);
                int intellect = int.Parse(values[8]);
                string power = values[9];

                //Creates new hero object
                Hero hero = new Hero(race, city, name, number, health, mana,
                    damage, defend, strength, speed, intellect, power);

                //Adds the created Hero object to the list of heroes
                heroes.Add(hero);
            }

            return heroes;
        }

        /// <summary>
        /// A method that prints all heroes and their data to the console
        /// </summary>
        /// <param name="heroes"></param>
    }
}

```

```

public static void PrintAllHeroes(HeroRegister heroes)
{
    List<string> races = heroes.GetRaces();

    foreach (string race in races)
    {
        string city = heroes.GetCityByRace(race);

        //A table is created to store the data
        Console.WriteLine("Rasé: {0}; Miestas: {1}", race, city);
        Console.WriteLine(new string('-', 128));
        Console.WriteLine("| {0,-12} | {1,-5} | {2,-14} | " +
            "{3,-4} | {4,-12} | {5,-14} | {6,-4} | {7,-8} | " +
            "{8,-10} | {9,-14} |", "Vardas", "Klasé",
            "Gyvybés taškai", "Mana", "Žalos taškai",
            "Gynybos taškai", "Jéga", "Vikrumas", "Intelektas",
            "Ypatinga galia");
        Console.WriteLine(new string('-', 128));

        for (int i = 0; i < heroes.HeroCount(); i++)
        {
            Hero hero = heroes.WhichHero(i);

            //Selects a hero with exact race
            if (hero.race != race)
            {
                continue;
            }
            Console.WriteLine(hero.ToString());
        }

        Console.WriteLine(new string('-', 128));
        Console.WriteLine();
    }
}

/// <summary>
/// A method that prints all heroes and their data to a txt file
/// The data is stored in a table
/// </summary>
/// <param name="fileNameTxt"></param>
/// <param name="heroes"></param>
public static void PrintAllHeroesToTxt(string fileNameTxt,
    HeroRegister heroes)
{
    List<string> races = heroes.GetRaces();

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

    lines.Add("Registro informacija:");

    foreach (string race in races)
    {
        string city = heroes.GetCityByRace(race);

        lines.Add(String.Format("Rasé: {0}; Miestas: {1}", race, city));
        lines.Add(new string('-', 128));
        lines.Add(String.Format("| {0,-12} | {1,-5} | {2,-14} | " +
            "{3,-4} | {4,-12} | {5,-14} | {6,-4} | {7,-8} | " +
            "{8,-10} | {9,-14} |", "Vardas", "Klasé",

```

```

    "Gyvybės taškai", "Mana", "Žalos taškai", "Gynybos taškai",
    "Jéga", "Vikrumas", "Intelektas", "Ypatinga galia"));
lines.Add(new string('-', 128));

for (int i = 0; i < heroes.HeroCount(); i++)
{
    Hero hero = heroes.WhichHero(i);
    if (hero.race != race)
    {
        continue;
    }
    lines.Add(hero.ToString());
}

lines.Add(new string('-', 128));

lines.Add("");
}

//Prints on each line of the file
File.WriteAllLines(fileNameTxt, lines, Encoding.UTF8);
}

/// <summary>
/// The method prints hero classes (without duplicates) to
/// "Klases.csv" file
/// </summary>
/// <param name="fileName"></param>
/// <param name="numbers"></param>
public static void PrintNumbers(string fileName, List<int> numbers)
{
    List<string> lines = new List<string>();

    lines.Add("Heroju klasės:");

    for (int i = 0; i < numbers.Count; i++)
    {
        lines.Add(String.Format("{0}", numbers[i]));
    }

    //Prints to all lines of the file
    File.WriteAllLines(fileName, lines, Encoding.UTF8);
}

/// <summary>
/// The method prints heroes missing classes to csv file
/// <param name="fileName"></param>
/// <param name="numbers"></param>
public static void PrintMissingNumbers(string fileName,
    List<int> numbers)
{
    List<int> trols = new List<int>();
    List<int> elves = new List<int>();
    List<string> lines = new List<string>();

    lines.Add("Trükstamos klasės:");

    int x = numbers.IndexOf(-1)++;

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

```

```

        if (i > x)
        {
            trols.Add(numbers[i]);
        }
        else if (i < x)
        {
            elfs.Add(numbers[i]);
        }
    }

    lines.Add("Troliai:");
    if (trols.Count == 0)
    {
        lines.Add("VISI");
    }
    else
    {
        for (int i = 0; i < trols.Count(); i++)
        {
            lines.Add(String.Format("{0}", trols[i]));
        }
    }

    lines.Add("Elfai:");
    if (elfs.Count == 0)
    {
        lines.Add("VISI");
    }
    else
    {
        for (int i = 0; i < elfs.Count(); i++)
        {
            lines.Add(String.Format("{0}", elfs[i]));
        }
    }

    /// Prints to all lines of the file
    File.WriteAllLines(fileName, lines, Encoding.UTF8);
}

}

namespace U1_24KompiuterinisZaidimas
{
    /// <summary>
    /// Class which controls the list of heroes and logic around them
    /// </summary>
    public class HeroRegister
    {
        /// <summary>
        /// List of heroes
        /// </summary>
        private List<Hero> AllHeroes;

        /// <summary>
        /// Creating an empty register
        /// </summary>
        public HeroRegister()
        {
            AllHeroes = new List<Hero>();
        }
    }
}

```

```

}

/// <summary>
/// Creating a register with list of heroes
/// </summary>
/// <param name="heroes"></param>
public HeroRegister(List<Hero> heroes)
{
    AllHeroes = new List<Hero>();

    foreach (Hero hero in heroes)
    {
        this.AllHeroes.Add(hero);
    }
}

/// <summary>
/// Adds hero to the register
/// </summary>
/// <param name="hero"></param>
public void Add(Hero hero)
{
    AllHeroes.Add(hero);
}

/// <summary>
/// Counts how many heroes there are in register
/// </summary>
/// <returns></returns>
public int HeroCount()
{
    return this.AllHeroes.Count();
}

/// <summary>
/// Returns an exact hero from the register
/// </summary>
/// <param name="number"></param>
/// <returns></returns>
public Hero WhichHero(int number)
{
    return AllHeroes[number];
}

/// <summary>
/// Combines two hero registers into one
/// </summary>
/// <param name="heroes"></param>
/// <returns></returns>
public HeroRegister CombineRegisters(HeroRegister heroes)
{
    HeroRegister newRegister = new HeroRegister(this.AllHeroes);

    for (int i = 0; i < heroes.HeroCount(); i++)
    {
        newRegister.Add(heroes.WhichHero(i));
    }

    return newRegister;
}

```

```

    ///<summary>
    /// Returns different hero races
    ///</summary>
    ///<returns></returns>
    public List<string> GetRaces()
    {
        List<string> races = new List<string>();

        foreach (Hero hero in this.AllHeroes)
        {
            if (!races.Contains(hero.race))
            {
                races.Add(hero.race);
            }
        }

        return races;
    }

    ///<summary>
    /// Assigns city to its race
    ///</summary>
    ///<param name="race"></param>
    ///<returns></returns>
    public string GetCityByRace(string race)
    {
        foreach (Hero hero in this.AllHeroes)
        {
            if (race == hero.race)
            {
                return hero.city;
            }
        }

        return null;
    }

    ///<summary>
    /// Method which finds all the different classes of the heroes
    /// and places them in one list
    ///</summary>
    ///<returns></returns>
    public List<int> FindClasses()
    {
        //New list is created for the new list of the classes
        List<int> numbers = new List<int>();

        foreach (Hero hero in this.AllHeroes)
        {
            int nr = hero.number;

            if (!numbers.Contains(nr))
            {
                //Classes which met the conditions are added to
                //the new list
                numbers.Add(nr);
            }
        }

        numbers.Sort();
        return numbers;
    }

```

```

}

/// <summary>
/// Method which finds all missing classes and puts them in one list
/// </summary>
/// <returns></returns>
public List<int> MissingClasses()
{
    List<string> races = this.GetRaces();
    List<int> classes = new List<int>();
    List<int> other = new List<int>();

    for (int i = 0; i < 2; i++)
    {
        string race = races[i];
        for (int j = 0; j < this.HeroCount(); j++)
        {
            Hero hero = WhichHero(j);
            if (i == 0 && !classes.Contains(hero.number) && hero.race
                == race)
            {
                classes.Add(hero.number);
            }
            else if (i == 1 && !other.Contains(hero.number) && hero.race
                == race)
            {
                other.Add(hero.number);
            }
        }
    }

    var missesClasses = classes.Except(other).ToList();
    var missesOther = other.Except(classes).ToList();

    missesClasses.Add(-1);
    List<int> misses = missesClasses.Concat(missesOther).ToList();

    return misses;
}

/// <summary>
/// Method which finds the biggest strength of the heroes
/// </summary>
/// <returns></returns>
public double FindStrength()
{
    double strength = AllHeroes[0].health + AllHeroes[0].defend
        - AllHeroes[0].damage;

    foreach (Hero hero in this.AllHeroes)
    {
        if (strength < hero)
        {
            strength = hero.GetStrength();
        }
    }

    return strength;
}

/// <summary>

```

```

    /// Method which finds all the strongest heroes from the list
    /// </summary>
    /// <returns></returns>
    public HeroRegister FindAllStrongest()
    {
        HeroRegister strength = new HeroRegister();
        double powerfull = FindStrength();

        foreach (Hero hero in this.AllHeroes)
        {
            if (powerfull == hero.GetStrength())
            {
                strength.Add(hero);
            }
        }

        return strength;
    }
}

namespace U1_24KompiuterinisZaidimas
{
    /*U2-24. Kompiuterinis žaidimas. Sugrupavote herojus pagal dvi rases,
     * ir surašėte jų duomenis į skirtingus failus. Duomenų formatas dabar
     * tokis: pirmoje eilutėje - rasės pavadinimas. Antroje - pradinis miestas.
     * Toliau pateikta informacija tokiu pačiu formatu kaip L1 užduotyje,
     * tik nebéra rasės stulpelio.
     * • Sudarykite visų herojų klasių sąrašą, klasių pavadinimus išrašykite
     * į failą „Klasės.csv“ .
     * • Raskite, kokių klasių herojų „trūksta“ kiekvienai rasei. Į failą
     * „Trūkstamai.csv“ išrašykite kiekvienos rasės pavadinimą, ir trūkstamų
     * klasių sąrašą. Jei rasė turi bent po vieną kiekvienos klasės atstovą,
     * parašykite žodį „VISI“.
     * • Raskite, kurioje rasėje yra stipriausias herojus: herojus stiprumą
     * rodo gyvybės ir gynybos taškų suma sumažinta žalos taškais. Ekrane
     * atspausdinkite visus herojaus duomenis ir jo rasę.
    */
}

class Program
{
    static void Main(string[] args)
    {
        //Heroes and their data are read from the "Troliai.csv" and
        //"Elfai.csv" files and then are placed into the register
        HeroRegister registerT =
            InputOutput.ReadHeroes(@"../../../../Troliai.csv");
        HeroRegister registerE =
            InputOutput.ReadHeroes(@"../../../../Elfai.csv");

        HeroRegister register = registerT.CombineRegisters(registerE);

        //All heroes and their data are printed to the console
        //in a table
        Console.WriteLine("Registro informacija:");
        InputOutput.PrintAllHeroes(register);

        //All heroes and their data in the table are printed to
        //txt file
        InputOutput.PrintAllHeroesToTxt("Duomenys.txt", register);
    }
}

```

```

//First task result (all different heroes classes) is
//printed to the "Klases.csv" file
InputOutput.PrintNumbers("Klases.csv", register.FindClasses());

//Second task result (all missing each race classes) is printed
//to the "Trukstami.csv" file
List<int> missingClasses = register.MissingClasses();
InputOutput.PrintMissingNumbers("Trukstami.csv", missingClasses);

//Third task result: Searches in which race there is the
//strongest hero and prints it, but if there is more than
//one code prints others too
Console.WriteLine("Stipriausi herojai:");

HeroRegister strongT = registerT.FindAllStrongest();
HeroRegister strongE = registerE.FindAllStrongest();

HeroRegister strong = strongT.CombineRegisters(strongE);

if (strongT.WhichHero(0).GetStrength()
    == strongE.WhichHero(0).GetStrength())
{
    InputOutput.PrintAllHeroes(strong);
}
else if (strongT.WhichHero(0).GetStrength()
    > strongE.WhichHero(0).GetStrength())
{
    InputOutput.PrintAllHeroes(strong);
}
else
{
    InputOutput.PrintAllHeroes(strong);
}
}

}

}

```

## 2.3. Pradiniai duomenys ir rezultatai

### 2.3.1 Duomenys ir rezultatai 1

Pradiniai failai:

---

1	<b>Elfai</b>
2	<b>Atlanta</b>
3	<b>Astijus;1;100;50;55;50;100;100;100;Lietimas</b>
4	<b>Aurimas;2;150;60;65;60;150;150;150;Uostymas</b>
5	<b>Rimas;3;90;70;85;70;110;110;110;Klausa</b>
6	<b>Lukas;4;150;100;120;90;120;120;120;Rega</b>
7	<b>Arnas;7;100;90;120;90;130;130;130;Liepsna</b>

1 Troliai  
 2 Asgardas  
 3 Aistis;1;100;50;55;50;100;100;100;Lietimas  
 4 Rugilė;2;150;60;65;60;150;150;150;Uostymas  
 5 Matas;3;90;70;85;70;110;110;110;Klausa  
 6 Rokas;6;150;100;120;90;120;120;120;Rega  
 7 Tadas;6;100;90;120;90;130;130;130;Liepsna

Šiais duomenimis tikrinama ar veikia klasių atrūšiavimas (pirma užduotis), ar veikia trūkstamų klasių radimas (antra užduotis), ar veikia stipriausią herojų paieška (į jų šiuo atveju yra du: Trolis Rugilė ir Elfas Aurimas) (trečia užduotis).

Duomenys „.txt“ faile:

Duomenys.txt										
Registro informacija:										
Rasé: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	1	100	50	55	50	100	100	100	Lietimas	
Rugilė	2	150	60	65	60	150	150	150	Uostymas	
Matas	3	90	70	85	70	110	110	110	Klausa	
Rokas	6	150	100	120	90	120	120	120	Rega	
Tadas	6	100	90	120	90	130	130	130	Liepsna	

  

Rasé: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Astijus	1	100	50	55	50	100	100	100	Lietimas	
Aurimas	2	150	60	65	60	150	150	150	Uostymas	
Rimas	3	150	70	85	70	110	110	110	Klausa	
Lukas	4	150	100	120	90	120	120	120	Rega	
Arnas	7	100	90	120	90	130	130	130	Liepsna	

Rezultatai (Konsolėje):

Registro informacija:										
Rasé: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	1	100	50	55	50	100	100	100	Lietimas	
Rugilė	2	150	60	65	60	150	150	150	Uostymas	
Matas	3	90	70	85	70	110	110	110	Klausa	
Rokas	6	150	100	120	90	120	120	120	Rega	
Tadas	6	100	90	120	90	130	130	130	Liepsna	

  

Rasé: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Astijus	1	100	50	55	50	100	100	100	Lietimas	
Aurimas	2	150	60	65	60	150	150	150	Uostymas	
Rimas	3	150	70	85	70	110	110	110	Klausa	
Lukas	4	150	100	120	90	120	120	120	Rega	
Arnas	7	100	90	120	90	130	130	130	Liepsna	

  

Stipriausiai herojai:										
Rasé: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Rugilė	2	150	60	65	60	150	150	150	Uostymas	

  

Rasé: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Aurimas	2	150	60	65	60	150	150	150	Uostymas	

Rezultatai (Klases.csv):

### Klases

Herojų klasės:
1
2
3
4
6
7

Rezultatai (Trukstami.csv):

### Trukstami

Trūkstamos klasės:
Troliai:
4
7
Elfai:
6

### 2.3.2 Duomenys ir rezultatai 2

Pradinis failas:

```
1 Elfai
2 Atlanta
3 Astijus;1;100;50;55;50;100;100;100;Lietimas
4 Aurimas;2;150;60;65;60;150;150;150;Uostymas
5 Rimas;3;150;70;65;60;110;110;110;Klausa
6 Lukas;4;150;100;120;90;120;120;120;Rega
7 Arnas;5;100;90;120;90;130;130;130;Liepsna
8
1 Troliai
2 Asgardas
3 Aistis;1;150;50;65;60;100;100;100;Lietimas
4 Rugilė;2;150;60;65;60;150;150;150;Uostymas
5 Matas;3;150;70;65;60;110;110;110;Klausa
6 Rokas;4;150;100;120;90;120;120;120;Rega
7 Tadas;6;100;90;120;90;130;130;130;Liepsna
8
```

Šiais duomenimis tikrinama ar veikia stipriausių herojų paieška (jų šiuo atveju yra penki: Troliai (Rugilė, Aistis, Matas) ir Elfai: (Aurimas, Rimas)).

Duomenys „.txt“ faile:

Duomenys.txt										
Registro informacija:										
Rasė: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	1	150	50	65	60	100	100	100	Lietimas	
Rugilė	2	150	60	65	60	150	150	150	Uostymas	
Matas	3	150	70	65	60	110	110	110	Klausa	
Rokas	4	150	100	120	90	120	120	120	Rega	
Tadas	6	100	90	120	90	130	130	130	Liepsna	

  

Rasė: Elfai; Miestas: Atlanta										
Registro informacija:										
Rasė: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Astijus	1	100	50	55	50	100	100	100	Lietimas	
Aurimas	2	150	60	65	60	150	150	150	Uostymas	
Rimas	3	150	70	65	60	110	110	110	Klausa	
Lukas	4	150	100	120	90	120	120	120	Rega	
Arnas	5	100	90	120	90	130	130	130	Liepsna	

Rezultatai (Konsolėje):

Registro informacija:										
Rasė: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	1	150	50	65	60	100	100	100	Lietimas	
Rugilė	2	150	60	65	60	150	150	150	Uostymas	
Matas	3	150	70	65	60	110	110	110	Klausa	
Rokas	4	150	100	120	90	120	120	120	Rega	
Tadas	6	100	90	120	90	130	130	130	Liepsna	

  

Rasė: Elfai; Miestas: Atlanta										
Registro informacija:										
Rasė: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Astijus	1	100	50	55	50	100	100	100	Lietimas	
Aurimas	2	150	60	65	60	150	150	150	Uostymas	
Rimas	3	150	70	65	60	110	110	110	Klausa	
Lukas	4	150	100	120	90	120	120	120	Rega	
Arnas	5	100	90	120	90	130	130	130	Liepsna	

  

Stipriausi herojai:										
Rasė: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Aistis	1	150	50	65	60	100	100	100	Lietimas	
Rugilė	2	150	60	65	60	150	150	150	Uostymas	
Matas	3	150	70	65	60	110	110	110	Klausa	

  

Rasė: Elfai; Miestas: Atlanta										
Registro informacija:										
Rasė: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jėga	Vikrumas	Intelektas	Ypatinga galia	
Aurimas	2	150	60	65	60	150	150	150	Uostymas	
Rimas	3	150	70	65	60	110	110	110	Klausa	

Rezultatai (Klases.csv):

#### Klases

Herojų klasės:
1
2
3
4
5
6

Rezultatai (Trukstami.csv):

#### Trukstami

Trūkstamos klasės:
Troliai:
5
Elfai:
6

## 2.4. Dėstytojo pastabos

### 3. Konteineris

#### 3.1. Darbo užduotis

**U3\_24. Kompiuterinis žaidimas.** Sugrupavote herojus pagal dvi rases, ir surašėte jų duomenis į skirtinges failus. Duomenų formatas dabar tokis: pirmoje eilutėje – rasės pavadinimas. Antroje – pradinis miestas. Toliau pateikta informacija tokiu pačiu formatu kaip L1 užduotyje, tik nebéra rasės stulpelio.

- Raskite kiekvienos rasės stipriausių herojų: herojus stiprumą rodo gyvybės ir gynybos taškų sumažinta žalos taškais. Ekrane atspausdinkite visus herojaus duomenis ir jo rasę.
- Sudarykite visų herojų klasių sąrašą, klasių pavadinimus atspausdinkite ekrane.
- Raskite, kokių klasių herojų „trūksta“ kiekvienai rasei. I failą „Trūkstami.csv“ išrašykite kiekvienos rasės pavadinimą, ir trūkstamų klasių sąrašą. Jei rasė turi bent po vieną kiekvienos klasės atstovą, parašykite žodį „VISI“.
- Sudarykite herojų, kurie turi gyvybės taškų daugiau nei gynybos taškų, sąrašą. Surikiuokite herojus pagal gyvybės taškus, gynybos taškus ir vardus. Rezultatus išrašykite į failą „Herojai.csv“.

#### 3.2. Programos tekstas

```
namespace U1_24KompiuterinisZaidimas
{
    /// <summary>
    /// The class in which the constructor is created
    /// </summary>
    public class Hero
    {
        public string name { get; }
        public int number { get; }
        public int health { get; }
        public int mana { get; }
        public int damage { get; }
        public int defend { get; }
        public int strength { get; }
        public int speed { get; }
        public int intellect { get; }
        public string power { get; }

        /// <summary>
        /// Creates a hero constructor
        /// </summary>
        /// <param name="name"></param>
        /// <param name="number"></param>
        /// <param name="health"></param>
        /// <param name="mana"></param>
        /// <param name="damage"></param>
        /// <param name="defend"></param>
        /// <param name="strength"></param>
        /// <param name="speed"></param>
        /// <param name="intellect"></param>
        /// <param name="power"></param>
        public Hero(string name, int number,
                    int health, int mana, int damage, int defend, int strength,
```

```

        int speed, int intellect, string power)
{
    this.name = name;
    this.number = number;
    this.health = health;
    this.mana = mana;
    this.damage = damage;
    this.defend = defend;
    this.strength = strength;
    this.speed = speed;
    this.intellect = intellect;
    this.power = power;
}

/// <summary>
/// Overriding operator "<"
/// </summary>
/// <param name="strength"></param>
/// <param name="hero"></param>
/// <returns></returns>
public static bool operator <(double strength, Hero hero)
{
    return strength < hero.GetStrength();
}

/// <summary>
/// Overriding operator ">"
/// </summary>
/// <param name="strength"></param>
/// <param name="hero"></param>
/// <returns></returns>
public static bool operator >(double strength, Hero hero)
{
    return strength > hero.GetStrength();
}

/// <summary>
/// Overrides ToString method
/// </summary>
/// <returns></returns>
public override string ToString()
{
    string line;

    line = String.Format("| {0,-12} | {1,5} | {2,14} | " +
        "{3,4} | {4,12} | {5,14} | {6,4} | {7,8} | {8,10} | " +
        "{9,-14} |", this.name, this.number,
        this.health, this.mana, this.damage, this.defend,
        this.strength, this.speed, this.intellect, this.power);

    return line;
}

/// <summary>
/// Counts strength
/// </summary>
/// <returns></returns>
public double GetStrength()
{
    return this.health + this.defend - this.damage;
}

```

```

    ///<summary>
    /// Counts health
    ///</summary>
    ///<returns></returns>
    public double GetHealth()
    {
        return this.health - this.defend;
    }

    ///<summary>
    /// Compares healths, defences and names. This information will
    /// be used in sorting
    ///</summary>
    ///<param name="hero"></param>
    ///<returns></returns>
    public int CompareTo(Hero hero)
    {
        int hp = this.health.CompareTo(hero.health);
        int defence = this.defend.CompareTo(hero.defend);
        if (hp != 0)
        {
            return hp;
        }
        else if (defence != 0)
        {
            return defence;
        }
        else
        {
            return hero.name.CompareTo(this.name);
        }
    }
}

namespace U1_24KompiuterinisZaidimas
{
    ///<summary>
    /// Class in which the container is made
    ///</summary>
    public class HeroContainer
    {
        private int Capacity;
        private Hero[] heroes;
        public int Count { get; private set; }

        ///<summary>
        /// Creating a container
        ///</summary>
        ///<param name="capacity"></param>
        public HeroContainer(int capacity = 16)
        {
            this.heroes = new Hero[capacity];
            this.Capacity = capacity;
        }

        ///<summary>
        /// Method which ensures if there is enough space in the container
        ///</summary>
        ///<param name="minimum"></param>

```

```

private void EnsureCapacity(int minimum)
{
    if (minimum > Capacity)
    {
        Hero[] temp = new Hero[minimum];
        for (int i = 0; i < this.Count; i++)
        {
            temp[i] = this.heroes[i];
        }

        this.Capacity = minimum;
        this.heroes = temp;
    }
}

/// <summary>
/// Method checks if it is not the same hero
/// </summary>
/// <param name="hero"></param>
/// <returns></returns>
public bool Contains(Hero hero)
{
    for (int i = 0; i < this.Count; i++)
    {
        if (this.heroes[i].Equals(hero))
        {
            return true;
        }
    }

    return false;
}

/// <summary>
/// Method adds Hero to the container
/// </summary>
/// <param name="hero"></param>
public void Add(Hero hero)
{
    if (this.Count == this.Capacity)
    {
        this.EnsureCapacity(this.Capacity * 2);
    }

    if (this.Contains(hero))
    {
        return;
    }

    this.heroes[this.Count++] = hero;
}

/// <summary>
/// Method returns the hero by the index
/// </summary>
/// <param name="index"></param>
/// <returns></returns>
public Hero Get(int index)
{
    return this.heroes[index];
}

```

```

/// <summary>
/// Method puts the hero in the place the index show
/// </summary>
/// <param name="hero"></param>
/// <param name="index"></param>
public void Put(Hero hero, int index)
{
    if (index >= 0 && index < this.Count)
    {
        this.heroes[index] = hero;
    }
}

/// <summary>
/// Method inserts hero into the place which the index show
/// </summary>
/// <param name="hero"></param>
/// <param name="index"></param>
public void Insert(Hero hero, int index)
{
    if (index >= 0 && index < this.Count)
    {
        if (this.Count + 1 > this.Capacity)
        {
            this.EnsureCapacity(this.Capacity * 2);
        }

        for (int i = this.Count; i > index + 1; i--)
        {
            this.heroes[i] = this.heroes[i - 1];
        }

        this.heroes[index] = hero;
        this.Count++;
    }
}

/// <summary>
/// Method removes hero which is in the index place
/// </summary>
/// <param name="index"></param>
public void RemoveAt(int index)
{
    if (index >= 0 && index < this.Count)
    {
        for (int i = index + 1; i < this.Count; i++)
        {
            this.heroes[i - 1] = this.heroes[i];
        }

        this.Count--;
    }
}

/// <summary>
/// Method removes hero which is equal to the given one
/// </summary>
/// <param name="hero"></param>
public void Remove(Hero hero)
{
}

```

```

        for (int i = 0; i < this.Count; i++)
    {
        if (hero == this.heroes[i])
        {
            this.RemoveAt(i);
            break;
        }
    }
}

/// <summary>
/// Method returns sorted heroes who has more health points
/// than defence points
/// </summary>
/// <returns></returns>
public HeroContainer MoreHealth()
{
    HeroContainer healthier = new HeroContainer();

    for (int i = 0; i < this.Count; i++)
    {
        Hero hero = this.Get(i);

        if (hero.health > hero.defend)
        {
            healthier.Add(hero);
        }
    }

    return healthier;
}

/// <summary>
/// Method is sorting heroes
/// </summary>
public void Sort()
{
    bool flag = true;

    while (flag)
    {
        flag = false;

        for (int i = 0; i < this.Count - 1; i++)
        {
            Hero one = this.heroes[i];
            Hero two = this.heroes[i + 1];

            if (one.CompareTo(two) < 0)
            {
                this.heroes[i] = two;
                this.heroes[i + 1] = one;
                flag = true;
            }
        }
    }
}

/// <summary>
/// Method which finds all the different classes of the heroes
/// and places them in one list

```

```

    ///</summary>
    ///<returns></returns>
    public List<int> FindClasses()
    {
        //New list is created for the new list of the classes
        List<int> numbers = new List<int>();

        for (int i = 0; i < this.Count; i++)
        {
            int nr = this.heroes[i].number;

            if (!numbers.Contains(nr))
            {
                //Classes which met the conditions are added to
                //the new list
                numbers.Add(nr);
            }
        }

        numbers.Sort();
        return numbers;
    }

    ///<summary>
    /// Method which finds the biggest strength of the heroes
    ///</summary>
    ///<returns></returns>
    public double FindStrength()
    {
        if (this.Count == 0)
        {
            return 0;
        }

        double strength = this.heroes[0].GetStrength();

        for (int i = 0; i < this.Count; i++)
        {
            double heroStrength = this.heroes[i].GetStrength();
            if (strength < heroStrength)
            {
                strength = heroStrength;
            }
        }

        return strength;
    }

    ///<summary>
    /// Method which finds all the strongest heroes from the list
    ///</summary>
    ///<returns></returns>
    public HeroContainer FindAllStrongest()
    {
        HeroContainer strongContainer = new HeroContainer();

        double powerfull = FindStrength();

        for (int i = 0; i < this.Count; i++)
        {
            Hero hero = this.heroes[i];

```

```

        if (powerfull == hero.GetStrength())
        {
            strongContainer.Add(hero);
        }
    }

    return strongContainer;
}
}

namespace U1_24KompiuterinisZaidimas
{
    /// <summary>
    /// Class which controls the list of heroes and logic around them
    /// </summary>
    public class HeroRegister
    {
        /// <summary>
        /// Calls the container
        /// </summary>
        public HeroContainer AllHeroes { get; }
        public string race { get; private set; }
        public string city { get; private set; }

        /// <summary>
        /// Creating an empty container, gets race and city
        /// </summary>
        public HeroRegister(string race, string city)
        {
            this.AllHeroes = new HeroContainer();
            this.race = race;
            this.city = city;
        }

        /// <summary>
        /// Gets container, race and city
        /// </summary>
        public HeroRegister(string race, string city, HeroContainer container)
        {
            this.AllHeroes = container;
            this.race = race;
            this.city = city;
        }
    }
}

using System.Text; //Library used for text encoding

namespace U1_24KompiuterinisZaidimas
{
    /// <summary>
    /// A class that performs scans and prints
    /// </summary>
    public class InputOutput
    {
        /// <summary>
        /// A method that reads heroes and their data from the files
        /// </summary>

```

```

    ///<param name="fileName"></param>
    ///<returns></returns>
    public static HeroRegister ReadHeroes(string fileName)
    {
        //Reads all lines from a file in UTF-8 encoding
        string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);

        string race = Lines[0];
        string city = Lines[1];

        HeroRegister heroes = new HeroRegister(race, city);

        //Parses each line
        for (int i = 2; i < Lines.Length; i++)
        {
            string[] values = Lines[i].Split(";");
            string name = values[0];
            int number = int.Parse(values[1]);
            int health = int.Parse(values[2]);
            int mana = int.Parse(values[3]);
            int damage = int.Parse(values[4]);
            int defend = int.Parse(values[5]);
            int strength = int.Parse(values[6]);
            int speed = int.Parse(values[7]);
            int intellect = int.Parse(values[8]);
            string power = values[9];

            //Creates new hero object
            Hero hero = new Hero(name, number, health, mana,
                damage, defend, strength, speed, intellect, power);

            //Adds the created Hero object to the list of heroes
            heroes.AllHeroes.Add(hero);
        }

        return heroes;
    }

    ///<summary>
    /// A method that prints all heroes and their data to the console
    ///</summary>
    ///<param name="register"></param>
    public static void PrintHeroes(HeroRegister register)
    {
        //A table is created to store the data
        Console.WriteLine($"Rasé: {register.race}; " +
            $"Miestas: {register.city}");
        Console.WriteLine(new string('-', 128));
        Console.WriteLine("| {0,-12} | {1,-5} | {2,-14} | " +
            "{3,-4} | {4,-12} | {5,-14} | {6,-4} | {7,-8} | " +
            "{8,-10} | {9,-14} | ", "Vardas", "Klasė",
            "Gyvybės taškai", "Mana", "Žalos taškai",
            "Gynybos taškai", "Jėga", "Vikrumas", "Intelektas",
            "Ypatinga galia");
        Console.WriteLine(new string('-', 128));

        for (int i = 0; i < register.AllHeroes.Count; i++)
        {
            Hero hero = register.AllHeroes.Get(i);

```

```

        Console.WriteLine(hero.ToString());
    }

    Console.WriteLine(new string('-', 128));
    Console.WriteLine();
}

/// <summary>
/// A method that prints all heroes and their data to a txt file
/// The data is stored in a table
/// </summary>
/// <param name="fileNameTxt"></param>
/// <param name="registers"></param>
public static void PrintHeroesToTxt(string fileNameTxt,
    HeroRegister[] registers)
{
    File.AppendAllText(
        fileNameTxt,
        "Registro informacija:\r\n",
        Encoding.UTF8);

    foreach (HeroRegister register in registers)
    {
        List<string> lines = new List<string>();

        lines.Add(String.Format("Rasé: {0}; Miestas: {1}",
            register.race, register.city));
        lines.Add(new string('-', 128));
        lines.Add(String.Format("| {0,-12} | {1,-5} | {2,-14} | " +
            "{3,-4} | {4,-12} | {5,-14} | {6,-4} | {7,-8} | " +
            "{8,-10} | {9,-14} |", "Vardas", "Klasé",
            "Gyvybės taškai", "Mana", "Žalos taškai", "Gynybos taškai",
            "Jéga", "Vikrumas", "Intelektas", "Ypatinga galia"));
        lines.Add(new string('-', 128));

        for (int i = 0; i < register.AllHeroes.Count; i++)
        {
            Hero hero = register.AllHeroes.Get(i);

            lines.Add(hero.ToString());
        }

        lines.Add(new string('-', 128));

        lines.Add("");
        //Prints on each line of the file
        File.AppendAllLines(fileNameTxt, lines, Encoding.UTF8);
    }
}

/// <summary>
/// A method that passes one heroes register to the printing method
/// </summary>
/// <param name="fileName"></param>
/// <param name="registers"></param>
public static void PrintHealthiest(string fileName,
    List<HeroRegister> registers)
{
    foreach (HeroRegister register in registers)
}

```

```

    {
        if (register.AllHeroes.Count > 0)
        {
            PrintAllHeroesToCSV(fileName, register);
        }
        else
        {
            List<string> lines = new List<string>();
            lines.Add("Néra tokiu heroju");
            File.WriteAllText("Herojai.csv", lines, Encoding.UTF8);
        }
    }

    /**
     * A method that passes one heroes register to the printing method
     */
    /**
     * @param name="registers"></param>
     */
    public static void PrintStrongest(List<HeroRegister> registers)
    {
        Console.WriteLine("Stipriausi herojai:");
        foreach (HeroRegister register in registers)
        {
            PrintHeroes(register);
        }
    }

    /**
     * A method that prints all heroes and their data to a csv file
     * The data is stored in a table
     */
    /**
     * @param name="fileName"></param>
     * @param name="heroes"></param>
     */
    public static void PrintAllHeroesToCSV(string fileName,
                                           HeroRegister heroes)
    {
        List<string> lines = new List<string>();

        lines.Add(String.Format($"Rasė: {heroes.race}; " +
                               $"Miestas: {heroes.city}"));
        lines.Add(String.Format($"{"Vardas"}; {"Klasė"}; " +
                               $"{"Gyvybės taškai"}; {"Mana"}; {"žalos taškai"}; " +
                               $"{"Gynybos taškai"}; {"Jėga"}; {"Vikrumas"}; " +
                               $"{"Intelektas"}; {"Ypatinga galia"}"));

        for (int i = 0; i < heroes.AllHeroes.Count; i++)
        {
            Hero hero = heroes.AllHeroes.MoreHealth().Get(i);

            lines.Add(String.Format($"{hero.name}; {hero.number}; " +
                                   $"{hero.health}; {hero.mana}; {hero.damage}; " +
                                   $"{hero.defend}; {hero.strength}; {hero.speed}; " +
                                   $"{hero.intellect}; {hero.power}"));
        }

        lines.Add("");

        //Prints on each line of the file
        File.AppendAllLines(fileName, lines, Encoding.UTF8);
    }
}

```

```

    }

    /// <summary>
    /// The method prints hero classes (without duplicates) to console
    /// </summary>
    /// <param name="numbers"></param>
    public static void PrintNumbers(List<int> numbers)
    {
        Console.WriteLine("Herojų klasės:");

        for (int i = 0; i < numbers.Count; i++)
        {
            Console.WriteLine(String.Format("{0}", numbers[i]));
        }
    }

    /// <summary>
    /// The method prints heroes missing classes to csv file
    /// </summary>
    /// <param name="fileName"></param>
    /// <param name="numbers"></param>
    /// <param name="registers"></param>
    public static void PrintMissingNumbers(
        string fileName,
        List<int[]> numbers,
        HeroRegister[] registers)
    {
        List<string> lines = new List<string>();

        lines.Add("Trūkstamos klasės:");

        for (int i = 0; i < registers.Count(); i++)
        {
            lines.Add(registers[i].race + ":");

            if (numbers[i].Length == 0)
            {
                lines.Add("VIZI");
            }
            else
            {
                foreach (int clas in numbers[i])
                {
                    lines.Add(String.Format("{0}", clas));
                }
            }
        }

        //Prints to all lines of the file
        File.WriteAllLines(fileName, lines, Encoding.UTF8);
    }
}

using System.ComponentModel;
using System.Text;

namespace U1_24KompiuterinisZaidimas
{
    /*U3_24. Kompiuterinis žaidimas.

```

\* Sugrupavote herojus pagal dvi rases, ir surašete jų duomenis i  
 \* skirtingus failus. Duomenų formatas dabar toks: pirmoje  
 \* eilutėje – rasės pavadinimas. Antroje – pradinis miestas. Toliau  
 \* pateikta informacija tokiu pačiu formatu kaip L1 užduotyje, tik  
 \* nebéra rasės stulpelio.  
     • Raskite kiekvienos rasės stipriausią herojų: herojus stiprumą rodo  
     gyvybės ir gynybos taškų suma sumažinta žalos taškais. Ekrane  
     atspausdinkite visus herojaus duomenis ir jo rasę.  
     • Sudarykite visų herojų klasių sąrašą, klasių pavadinimus  
     atspausdinkite ekrane.  
     • Raskite, kokių klasių herojų „trūksta“ kiekvienai rasei. I  
     failą „Trūkstami.csv“ išrašykite kiekvienos rasės pavadinimą, ir  
     trūkstamų klasių sąrašą. Jei rasė turi bent po vieną kiekvienos klasės  
     atstovą, parašykite žodį „VISI“.  
     • Sudarykite herojų, kurie turi gyvybės tašką daugiau nei gynybos  
     taškų, sąrašą. Surikiuokite herojus pagal gyvybės taškus, gynybos taškus  
     ir vardus. Rezultatus išrašykite į failą „Herojai.csv“.  
 \*/

```

class Program
{
    static void Main(string[] args)
    {
        //Heroes and their data are read from the "Troliai.csv" and
        //"Elfai.csv" files and then are placed into the register
        HeroRegister registerT =
            InputOutput.ReadHeroes(@"../../../../Troliai.csv");
        HeroRegister registerE =
            InputOutput.ReadHeroes(@"../../../../Elfai.csv");

        //Deletes file so there would be no text from the past
        File.Delete("Duomenys.txt");
        File.Delete("Herojai.csv");

        HeroRegister[] registers
            = new HeroRegister[] { registerT, registerE };

        //All heroes and their data in the table are printed to
        //txt file
        InputOutput.PrintHeroesToTxt("Duomenys.txt", registers);

        //All heroes and their data are printed to the console
        //in a table
        Console.WriteLine("Registro informacija:");
        InputOutput.PrintHeroes(registerT);
        InputOutput.PrintHeroes(registerE);

        //First task result: Searches in which race there is the
        //strongest hero and prints it, but if there is more than
        //one prints others too
        List<HeroRegister> strongest = CollectStrongest(registers);
        InputOutput.PrintStrongest(strongest);

        //Second task result (all different heroes classes) is
        //printed to the console
        List<int> classes = CollectClasses(registers);
        InputOutput.PrintNumbers(classes);

        //Third task result (all missing each race classes) are printed
        //to the "Trukstami.csv" file
        List<int[]> missingClasses = MissingClasses(registers);
    }
}
  
```

```

        InputOutput.PrintMissingNumbers("Trukstami.csv",
            missingClasses, registers);

        //Forth task: selected and sorted heroes are printed to
        //"Herojai.csv" file
        List<HeroRegister> healthiest = CollectHealthiest(registers);
        InputOutput.PrintHealthiest("Herojai.csv", healthiest);
    }

    /// <summary>
    /// Method collects all different classes
    /// </summary>
    /// <param name="registers"></param>
    /// <returns></returns>
    public static List<int> CollectClasses(HeroRegister[] registers)
    {
        List<int> results = new List<int>();

        foreach (HeroRegister register in registers)
        {
            List<int> registerClasses = register.AllHeroes.FindClasses();

            foreach (int clas in registerClasses)
            {
                if (!results.Contains(clas))
                {
                    results.Add(clas);
                }
            }
        }

        results.Sort();

        return results;
    }

    /// <summary>
    /// Method which finds all missing classes and puts them in one list
    /// </summary>
    /// <param name="registers"></param>
    /// <returns></returns>
    public static List<int[]> MissingClasses(HeroRegister[] registers)
    {
        List<int[]> result = new List<int[]>();
        List<int> allClasses = CollectClasses(registers);

        foreach (HeroRegister register in registers)
        {
            result.Add(allClasses.Except(
                register.AllHeroes.FindClasses()).ToArray());
        }

        return result;
    }

    /// <summary>
    /// Method collects all the strongest heroes into one list
    /// </summary>
    /// <param name="registers"></param>
    /// <returns></returns>

```

```

public static List<HeroRegister> CollectStrongest(
    HeroRegister[] registers)
{
    HeroContainer[] strongest = new HeroContainer[registers.Length];
    double strength = 0;

    for (int i = 0; i < registers.Length; i++)
    {
        strongest[i] = registers[i].AllHeroes.FindAllStrongest();

        if (strongest[i].Get(0).GetStrength() > strength)
        {
            strength = strongest[i].Get(0).GetStrength();
        }
    }

    return CollectStrongest(registers, strongest, strength);
}

/// <summary>
/// Mthod collects strongest heroes into the List
/// </summary>
/// <param name="registers"></param>
/// <param name="strongest"></param>
/// <param name="strength"></param>
/// <returns></returns>
public static List<HeroRegister> CollectStrongest(
    HeroRegister[] registers,
    HeroContainer[] strongest,
    double strength)
{
    List<HeroRegister> result = new List<HeroRegister>();

    for (int i = 0; i < strongest.Length; i++)
    {
        if (strongest[i].Get(0).GetStrength() == strength)
        {
            result.Add(new HeroRegister(
                registers[i].race,
                registers[i].city,
                strongest[i]
            ));
        }
    }

    return result;
}

/// <summary>
/// Method collects all healthiest heroes into one list
/// </summary>
/// <param name="registers"></param>
/// <returns></returns>
public static List<HeroRegister> CollectHealthiest(
    HeroRegister[] registers)
{
    List<HeroRegister> result = new List<HeroRegister>();
    double health = 0;

    for (int i = 0; i < registers.Length; i++)

```

```

    {
        HeroContainer healthiest = registers[i].AllHeroes.MoreHealth();

        if (healthiest.Count == 0)
        {
            continue;
        }

        if (healthiest.Get(0).GetHealth() > health)
        {
            result = new List<HeroRegister>();
            health = healthiest.Get(0).GetHealth();
        }
        else if (healthiest.Get(0).GetHealth() < health)
        {
            continue;
        }

        healthiest.Sort();
        result.Add(new HeroRegister(
            registers[i].race,
            registers[i].city,
            healthiest));
    };

    return result;
}
}
}

```

### 3.3. Pradiniai duomenys ir rezultatai

#### 3.3.1 Duomenys ir rezultatai 1

Pradiniai failai:

---

```

1 Troliai
2 Asgardas
3 Aldas;1;150;10;60;60;100;100;100;Lietimas
4 Rugis;2;100;10;60;60;150;100;100;Uostymas
5 Matas;3;150;10;60;60;110;100;100;Klausa
6 Rokas;4;150;10;120;90;120;100;100;Rega
7 Tadas;5;100;10;120;90;130;100;100;Liepsna
8

1 Elfai
2 Atlanta
3 Jonas;1;150;10;55;60;100;100;100;Lietimas
4 Petra;2;150;10;60;60;100;100;100;Uostymas
5 Rimas;3;150;10;60;60;100;100;100;Klausa
6 Lukas;4;150;10;120;90;100;100;100;Rega
7 Arnas;5;100;10;120;90;100;100;100;Liepsna
8

```

Šiais duomenimis tikrinama pirma užduotis(ar randami stipriausi herojai), antra užduotis(ar išrašomos visos herojų klasės be pasikartojimų), trečia užduotis(ar atspausdina, kad netrūksta nei vienos klasės abejoms rasėms), ketvirta užduotis(ar yra surūšiuojami herojai, kurių gyvybės taškai yra didesni už gynybos (šiuo atveju visi didesni, tik rūšiavimas)).

Duomenys txt faile:

Duomenys.txt										
Registro informacija:										
Rasė: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Aldas	1	150	10	60	60	100	100	100	Lietimas	
Rugis	2	100	10	60	60	150	100	100	Uostymas	
Matas	3	150	10	60	60	110	100	100	Klausa	
Rokas	4	150	10	120	90	120	100	100	Rega	
Tadas	5	100	10	120	90	130	100	100	Liepsna	

  

Rasė: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Jonas	1	150	10	55	60	100	100	100	Lietimas	
Petra	2	150	10	60	60	100	100	100	Uostymas	
Rimas	3	150	10	60	60	100	100	100	Klausa	
Lukas	4	150	10	120	90	100	100	100	Rega	
Arnas	5	100	10	120	90	100	100	100	Liepsna	

Rezultatai (Konsolėje):

Registro informacija:										
Rasė: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Aldas	1	150	10	60	60	100	100	100	Lietimas	
Rugis	2	100	10	60	60	150	100	100	Uostymas	
Matas	3	150	10	60	60	110	100	100	Klausa	
Rokas	4	150	10	120	90	120	100	100	Rega	
Tadas	5	100	10	120	90	130	100	100	Liepsna	

  

Rasė: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Jonas	1	150	10	55	60	100	100	100	Lietimas	
Petra	2	150	10	60	60	100	100	100	Uostymas	
Rimas	3	150	10	60	60	100	100	100	Klausa	
Lukas	4	150	10	120	90	100	100	100	Rega	
Arnas	5	100	10	120	90	100	100	100	Liepsna	

  

Stipriausi herojai:										
Rasė: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Jonas	1	150	10	55	60	100	100	100	Lietimas	

  

Herojų klasės:										
1										
2										
3										
4										
5										

Rezultatai (Trukstami.csv):

### Trukstami

Trūkstamos klasės:
Troliai:
VISI
Elfai:
VISI

Rezultatai (Herojai.csv) :

Herojai

Rasė: Troliai	Miestas: Asgardas								
<b>Vardas</b>	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
<b>Rokas</b>	4	150	10	120	90	120	100	100	Rega
<b>Aldas</b>	1	150	10	60	60	100	100	100	Lietimas
<b>Matas</b>	3	150	10	60	60	110	100	100	Klausa
<b>Tadas</b>	5	100	10	120	90	130	100	100	Liepsna
<b>Rugis</b>	2	100	10	60	60	150	100	100	Uostymas
<b>Rasė: Elfai</b>	Miestas: Atlanta								
<b>Vardas</b>	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
<b>Lukas</b>	4	150	10	120	90	100	100	100	Rega
<b>Jonas</b>	1	150	10	55	60	100	100	100	Lietimas
<b>Petra</b>	2	150	10	60	60	100	100	100	Uostymas
<b>Rimas</b>	3	150	10	60	60	100	100	100	Klausa
<b>Arnas</b>	5	100	10	120	90	100	100	100	Liepsna

### 3.3.2 Duomenys ir rezultatai 2

Pradiniai failai:

---

```

1  Troliai
2  Asgardas
3  Aldas;1;150;10;55;60;100;100;100;Lietimas
4  Rugis;2;100;10;60;60;150;100;100;Uostymas
5  Matas;3;150;10;55;60;110;100;100;Klausa
6  Rokas;4;150;10;120;90;120;100;100;Rega
7  Tadas;5;100;10;120;90;130;100;100;Liepsna
8

```

---

```

1  Elfai
2  Atlanta
3  Jonas;1;150;10;55;60;100;100;100;Lietimas
4  Petra;2;150;10;60;60;100;100;100;Uostymas
5  Rimas;3;150;10;60;60;100;100;100;Klausa
6  Lukas;4;15;10;120;90;100;100;100;Rega
7  Arnas;4;10;10;120;90;100;100;100;Liepsna
8

```

Šiais duomenimis tikrinama trečia užduotis(ar atspausdina, kad yra trūkstamų klasių), ketvirta užduotis(ar yra surūšiuojami herojai, kurių gyvybės taškai yra didesni už gynybos).

Duomenys txt failė:

● ● ●

**Registro informacija:**  
Rasė: Troliai; Miestas: Asgardas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Aldas	1	150	10	55	60	100	100	100	Lietimas
Rugis	2	100	10	60	60	150	100	100	Uostymas
Matas	3	150	10	55	60	110	100	100	Klausa
Rokas	4	150	10	120	90	120	100	100	Rega
Tadas	5	100	10	120	90	130	100	100	Liepsna

  

Rasė: Elfai; Miestas: Atlanta

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Jonas	1	150	10	55	60	100	100	100	Lietimas
Petra	2	150	10	60	60	100	100	100	Uostymas
Rimas	3	150	10	60	60	100	100	100	Klausa
Lukas	4	15	10	120	90	100	100	100	Rega
Arnas	4	10	10	120	90	100	100	100	Liepsna

Rezultatai (Konsolėje):

**Registro informacija:**  
Rasė: Troliai; Miestas: Asgardas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Aldas	1	150	10	55	60	100	100	100	Lietimas
Rugis	2	100	10	60	60	150	100	100	Uostymas
Matas	3	150	10	55	60	110	100	100	Klausa
Rokas	4	150	10	120	90	120	100	100	Rega
Tadas	5	100	10	120	90	130	100	100	Liepsna

  

Rasė: Elfai; Miestas: Atlanta

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Jonas	1	150	10	55	60	100	100	100	Lietimas
Petra	2	150	10	60	60	100	100	100	Uostymas
Rimas	3	150	10	60	60	100	100	100	Klausa
Lukas	4	15	10	120	90	100	100	100	Rega
Arnas	4	10	10	120	90	100	100	100	Liepsna

  

**Stipriausi herojai:**  
Rasė: Troliai; Miestas: Asgardas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Aldas	1	150	10	55	60	100	100	100	Lietimas
Matas	3	150	10	55	60	110	100	100	Klausa

  

Rasė: Elfai; Miestas: Atlanta

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Jonas	1	150	10	55	60	100	100	100	Lietimas

  

Herojų klasės:

1  
2  
3  
4  
5

Rezultatai (Trukstami.csv):

## Trukstami

**Trūkstamos klasės:**

Troliai:
VISI
Elfai:

5

Rezultatai (Herojai.csv) :

### Herojai

Rasė: Troliai	Miestas: Asgardas								
<b>Vardas</b>	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
<b>Rokas</b>	4	150	10	120	90	120	100	100	Rega
<b>Aidas</b>	1	150	10	55	60	100	100	100	Lietimas
<b>Matas</b>	3	150	10	55	60	110	100	100	Klausa
<b>Tadas</b>	5	100	10	120	90	130	100	100	Liepsna
<b>Rugis</b>	2	100	10	60	60	150	100	100	Uostymas
<b>Rasė: Elfai</b>	Miestas: Atlanta								
<b>Vardas</b>	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
<b>Jonas</b>	1	150	10	55	60	100	100	100	Lietimas
<b>Petra</b>	2	150	10	60	60	100	100	100	Uostymas
<b>Rimas</b>	3	150	10	60	60	100	100	100	Klausa

### 3.4. Dėstytojo pastabos

## 4. Teksto analizė ir redagavimas

### 4.1. Darbo užduotis

**U4H-24. Pasikartojantys žodžiai.** Tekstiniame faile Knyga.txt duotas tekstas sudarytas iš žodžių, atskirtų skyrikliais. Skyriklių aibė žinoma. Raskite ir spausdinkite faile Rodikliai.txt:

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

Reikia teksto žodžius sulygiuoti, kad kiekvienos eilutės kiekvienas žodis prasidėtų fiksuojoje toje pačioje pozicijoje. Galima įterpti tik minimalų būtiną tarpą skaičių. Reikia šalinti iš pradinio teksto kelis iš eilės einančius vienodus skyriklius, paliekant tik vieną jų atstovą. Įterpimo taisykłę taikome, siekdam gauti lygiuotą minimalų tekstą. Pradinio teksto eilutės ilgis neviršija 80 simbolių. Spausdinkite faile ManoKnyga.txt pertvarkytą tekstą pagal tokias taisykles:

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

### 4.2. Programos tekstas

```
namespace U4H_24_Pasikartojantys_zdz
{
    /* U4H-24. Pasikartojantys žodžiai.
       Tekstiniame faile Knyga.txt duotas tekstas sudarytas iš žodžių,
       atskirtų skyrikliais. Skyriklių aibė žinoma. Raskite ir spausdinkite
       faile Rodikliai.txt:
       Nurodyta kiekį dažniausiai pasikartojančių žodžių (ne daugiau
       nei 10 žodžių), surikiuota pagal pasikartojimo skaičių mažėjimo tvarka,
       o kai pasikartojimų skaičius sutampa – pagal abécéle;
       Ilgiausią sakinį (didžiausias žodžių kiekis), jo ilgį (simboliais
       ir žodžiais) ir vietą (sakinio pradžios eilutės numerį).

       Reikia teksto žodžius sulygiuoti, kad kiekvienos eilutės kiekvienas
       žodis prasidėtų fiksuojoje toje pačioje pozicijoje. Galima įterpti
       tik minimalų būtiną tarpą skaičių. Reikia šalinti iš pradinio teksto
       kelis iš eilės einančius vienodus skyriklius, paliekant tik vieną jų
       atstovą. Įterpimo taisykłę taikome, siekdam gauti lygiuotą minimalų
       tekstą. Pradinio teksto eilutės ilgis neviršija 80 simbolių.
       Spausdinkite faile ManoKnyga.txt pertvarkytą tekstą pagal tokias
       taisykles:
       Kiekvienos eilutės pirmasis žodis turi prasidėti pozicijoje p1=1.
       Antrasis kiekvienos eilutės žodis turi prasidėti minimalioje galimoje
       pozicijoje p2, tokioje, kad kiekvienos eilutės pirmasis žodis kartu
       su už jo esančiais skyrikliais baigiasi iki p2-2 arba p2-1.
       Trečiasis kiekvienos eilutės žodis turi prasidėti minimalioje
       galimoje pozicijoje p3, tokioje, kad kiekvienos eilutės antrasis žodis
       kartu su už jo esančiais skyrikliais baigiasi iki p3-2 arba p3-1.
```

```

        Ir t.t.
    */

    /// <summary>
    /// Main class
    /// </summary>
    class Program
    {
        static void Main(string[] args)
        {
            const string CFd = @"../../Knyga.txt";
            const string CFr8 = "Rodikliai.txt";
            const string CFr10 = "ManoKnyga.txt";
            char[] punctuation = { ' ', '.', ',', '!', '?', ':', ';',
                '(', ')', '\t' };
            string punct = @"\s,.;!?:()\\-";
            string pattern = @"([\p{P}\s])\1+";
            string endOfSentence = ".!?";

            //Solves first two tasks
            InOut.ProcessFirstTasks(CFd, CFr8, punctuation, endOfSentence);

            //Solves the last task
            InOut.ProcessSecondTask(CFd, CFr10, pattern, punct);
        }
    }

    using System.Text;
    using System.IO;
    using System.Text.RegularExpressions;
    using System.Collections.Generic;
    using System;
    using System.ComponentModel;

    namespace U4H_24_Pasikartojantys_zdz
    {
        /// <summary>
        /// Class that performs scans and prints
        /// </summary>
        public class InOut
        {
            /// <summary>
            /// Method reads from the file and prints the results
            /// </summary>
            /// <param name="fin"></param>
            /// <param name="fout"></param>
            /// <param name="punctuation"></param>
            public static void ProcessFirstTasks(string fin, string fout,
                char[] punctuation, string endOfSentence)
            {
                using (StreamReader reader = new StreamReader(fin, Encoding.UTF8))
                {
                    string line;
                    int numberofLine = 0;
                    string[] longestSentece = new string[2];

                    longestSentece[0] = "";
                    longestSentece[1] = "";

                    List<Word> mostRepetitive = new List<Word>();

```

```

// Properties array is used instead of a new class
// In it the results are being saved
int[] properties = new int[6];

// Each properties element is set to zero
SetPropertiesToZero(properties);

using (var writer = File.CreateText(fout))
{
    while ((line = reader.ReadLine()) != null)
    {
        // Finds words with their repetition
        TaskUtils.FindMostRepetitive(line, mostRepetitive,
            punctuation);

        // Finds the longest sentence
        TaskUtils.FindLongestSentence(line, punctuation,
            properties, numberofLine, endOfSentence,
            longestSentece);

        numberofLine++;
    }

    // Sorts repetitive words
    Sort(mostRepetitive);

    // Prints repetitive words
    PrintMostRepetitive(mostRepetitive, writer);

    writer.WriteLine($"Ilgiausias sakinys:");
    writer.WriteLine($"{longestSentece[0]}");
    writer.WriteLine($"Zodziai: {properties[0]}; " +
        $"Simboliai: {properties[1]}; Sakinio pradzia: " +
        $"{properties[4]}");
}
}

/// <summary>
/// Each properties element is set to zero
/// </summary>
/// <param name="properties"></param>
private static void SetPropertiesToZero(int[] properties)
{
    for (int i = 0; i < 6; i++)
    {
        properties[i] = 0;
    }
}

/// <summary>
/// Sorting
/// </summary>
/// <param name="mostRepetitive"></param>
private static void Sort(List<Word> mostRepetitive)
{
    bool flag = true;

    while (flag)
    {

```

```

        flag = false;

        for (int i = 0; i < mostRepetitive.Count - 1; i++)
        {
            Word one = mostRepetitive[i];
            Word two = mostRepetitive[i + 1];

            if (one.CompareTo(two) < 0)
            {
                mostRepetitive[i] = two;
                mostRepetitive[i + 1] = one;
                flag = true;
            }
        }
    }

    /// <summary>
    /// Printing most repetitive first 10 words
    /// </summary>
    /// <param name="mostRepetitive"></param>
    /// <param name="writer"></param>
    private static void PrintMostRepetitive(List<Word> mostRepetitive,
                                             StreamWriter writer)
    {
        writer.WriteLine("Dazniausiai pasikartojantys zodziai: ");

        for (int i = 0; i < Math.Min(10, mostRepetitive.Count); i++)
        {
            writer.WriteLine(mostRepetitive[i].word + " "
                            + mostRepetitive[i].count);
        }
    }

    /// <summary>
    /// Reads from the file and prints the results
    /// </summary>
    /// <param name="fin"></param>
    /// <param name="fout"></param>
    /// <param name="punct"></param>
    /// <param name="pattern"></param>
    public static void ProcessSecondTask(string fin, string fout,
                                         string pattern, string punct)
    {
        using (StreamReader reader = new StreamReader(fin, Encoding.UTF8))
        {
            string line;
            Word[] wordsInLine = new Word[40];

            // Each array elemnt is set to empty
            SetArrayToEmpty(wordsInLine);

            using (var writer = File.CreateText(fout))
            {
                while ((line = reader.ReadLine()) != null)
                {
                    // Removes repetitive punctuation
                    line = Regex.Replace(line, pattern, "$1");

                    // Finds the longest word in the column
                    TaskUtils.FindBiggestWordInColumn(line, wordsInLine,

```

```

                punct);
}

// Sets the start for the reader to zero so it would read
// it again from the start
reader.BaseStream.Seek(0, SeekOrigin.Begin);

// Prints aligned text
PrintAlignedLines(reader, writer, pattern, wordsInLine,
                  punct);
}
}

/// <summary>
/// Each array elemnt is set to empty
/// </summary>
/// <param name="wordsInLine"></param>
private static void SetArrayToEmpty(Word[] wordsInLine)
{
    for (int i = 0; i < 40; i++)
    {
        wordsInLine[i] = new Word("");
    }
}

/// <summary>
/// Prints aligned text
/// </summary>
/// <param name="reader"></param>
/// <param name="writer"></param>
/// <param name="pattern"></param>
/// <param name="wordsInLine"></param>
/// <param name="punct"></param>
private static void PrintAlignedLines(StreamReader reader,
                                     StreamWriter writer, string pattern, Word[] wordsInLine,
                                     string punct)
{
    string line;

    while ((line = reader.ReadLine()) != null)
    {
        if (line.Length > 0)
        {
            // Removes repetitive punctuation
            line = Regex.Replace(line, pattern, "$1");

            string[] words = Regex.Split(line, "[" + punct + "]+");

            // Aligning
            for (int i = 0; i < words.Length; i++)
            {
                if (!string.IsNullOrWhiteSpace(words[i]))
                {
                    //Mathces the word in line and adds to
                    //it punctuation
                    Match withPunctuation = Regex.Match(line,
                                                       words[i] + "([[" + punct + "]+)?");
                    writer.Write(withPunctuation);
                }
            }
        }
    }
}

```

```

        line = line.Substring(withPunctuation.Length);

        //Adds spaces if needed
        if (withPunctuation.Length
            != wordsInLine[i].word.Length)
        {
            int temp = wordsInLine[i].word.Length
                - withPunctuation.Length + 1;

            writer.Write(new string(' ', temp));
        }
        else
        {
            writer.Write(' ');
        }
    }

    writer.WriteLine();
}
}
}

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

namespace U4H_24_Pasikartojantys_zdz
{
    /// <summary>
    /// Class in which all the logic is done
    /// </summary>
    public class TaskUtils
    {
        /// <summary>
        /// Finds repetitive words and their frequency
        /// </summary>
        /// <param name="line"></param>
        /// <param name="mostRepetitive"></param>
        /// <param name="punctuation"></param>
        public static void FindMostRepetitive(string line,
List<Word> mostRepetitive, char[] punctuation)
        {
            if (line.Length > 0)
            {
                string[] words = line.Split(punctuation,
StringSplitOptions.RemoveEmptyEntries);

                foreach (string word in words)
                {
                    string wordLower = word.ToLower();

                    Word w = mostRepetitive.Find(x => x.word == wordLower);

                    if (w == null)
                    {
                        w = new Word(wordLower);
                        mostRepetitive.Add(w);
                    }
                }
            }
        }
    }
}
```

```

        }

        w.Increase();
    }
}

/// <summary>
/// First finds how many there is sentence endings in the line and
/// gets the longest information from another method
/// </summary>
/// <param name="line"></param>
/// <param name="punctuation"></param>
/// <param name="properties"></param>
/// <param name="numberOfLine"></param>
public static void FindLongestSentence(string line,
char[] punctuation, int[] properties, int numberOfLine,
string endOfSentence, string[] longestSentence)
{
    int x = 0;

    for (int i = 0; i < line.Length; i++)
    {
        if (endOfSentence.Contains(line[i]))
        {
            x++;
        }
    }

    // Finds the longest sentence
    CollectLongestSentences(line, punctuation, properties, x,
    numberOfLine, endOfSentence, longestSentence);
}

/// <summary>
/// Finds the longest sentece
/// </summary>
/// <param name="line"></param>
/// <param name="punctuation"></param>
/// <param name="properties"></param>
/// <param name="x"></param>
/// <param name="numberOfLine"></param>
private static void CollectLongestSentences(string line,
char[] punctuation, int[] properties, int x, int numberOfLine,
string endOfSentence, string[] longestSentence)
{
    int start = 0;

    for (int i = 0; i < line.Length; i++)
    {
        StringBuilder.newLine = new StringBuilder();

        if (endOfSentence.Contains(line[i])
        && x != 0)
        {
            newLine.Append(line, start, i - start + 1);

            int temp = newLine.Length;
            string[] words = newLine.ToString().Split(punctuation,
StringSplitOptions.RemoveEmptyEntries);
        }
    }
}

```

```

                if (properties[0] < words.Length +
properties[2]
                && start == 0)
                {
                    properties[0] = words.Length +
properties[2];
                    properties[1] = temp +
properties[3];
                    longestSentence[0] = longestSentence[1] + " "
+ newLine.ToString();

                    if (properties[2] != 0)
                    {
                        properties[4] = properties[5];
                    }
                    else
                    {
                        properties[4] = lineNumber + 1;
                    }
                    properties[2] = 0;
                    properties[3] = 0;
                    longestSentence[1] = "";
                }
                else if (properties[0] < words.Length)
                {
                    properties[0] = words.Length;
                    properties[1] = temp;
                    properties[4] = lineNumber + 1;
                    longestSentence[0] = newLine.ToString();
                }

                start = i + 1;
                x--;
            }
        }
        else if (x == 0)
        {
            newLine.Append(line.Substring(start));

            int temp = newLine.Length;
            string[] words = newLine.ToString().Split(punctuation,
StringSplitOptions.RemoveEmptyEntries);

            if (properties[2] == 0)
            {
                properties[5] = lineNumber + 1;
            }

            if (start == 0)
            {
                properties[2] += words.Length;
                properties[3] += temp;
                longestSentence[1] += newLine.ToString();
            }
            else
            {
                properties[2] = words.Length;
                properties[3] = temp;
                longestSentence[1] = newLine.ToString();
            }
        }
    }

    break;
}

```

```

        }
    }

/// <summary>
/// Finds the longest words in the column
/// </summary>
/// <param name="line"></param>
/// <param name="wordsInLine"></param>
public static void FindBiggestWordInColumn(string line,
    Word[] wordsInLine, string punct)
{
    if (line.Length > 0)
    {
        //Splits line into words with no punctuation
        string[] words = Regex.Split(line, "[" + punct + "]+");
        int position = 0;

        for (int i = 0; i < words.Length; i++)
        {
            if (!string.IsNullOrWhiteSpace(words[i]))
            {
                //Adds punctuation if there was any to the
                //matched word
                Match withPunctuation = Regex.Match(line,
                    words[i] + "([" + punct + "]+)?");

                line = line.Substring(withPunctuation.Length);

                if (withPunctuation.Length
                    > wordsInLine[i].word.Length)
                {
                    wordsInLine[i].word =
                        withPunctuation.ToString().TrimEnd(' ');
                }
            }
        }
    }
}

namespace U4H_24_Pasikartojantys_zdz
{
    /// <summary>
    /// Constructor class
    /// </summary>
    public class Word
    {
        public string word { get; set; }
        public int count { get; private set; }

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="word"></param>
        public Word(string word)
        {
            this.word = word;
            this.count = 0;
        }
    }
}

```

```

    ///<summary>
    /// Increases the count
    ///</summary>
    public void Increase()
    {
        count++;
    }

    ///<summary>
    /// Compares first the count than by the alphabet
    ///</summary>
    ///<param name="other"></param>
    ///<returns></returns>
    public int CompareTo(Word other)
    {
        int bigger = this.count.CompareTo(other.count);
        if (bigger != 0)
        {
            return bigger;
        }
        else
        {
            return other.word.CompareTo(this.word);
        }
    }
}

```

## 4.3. Pradiniai duomenys ir rezultatai

### 4.3.1 Duomenys ir rezultatai 1

Pradinis failas:

```

1 Aa aa bb aa bb? Bb
2 aa bb bb aa!!!
3 laba|

```

Šie duomenys tikrina ar gerai yra nuskaitomi žodžiai ir jų kiekis, tikrinama ar veikia rūšiavimas tarp dažniausiai pasikartojančių žodžių. Tikrinama ar randamas teisingas ilgiausias sakinys. Tikrinama ar yra pašalinami pasikartojantys skyrikliai bei ar yra sulygiuojamas tekstas.

Rezultatai (Rodikliai.txt):

```

Dazniausiai pasikartojančios žodžiai:
aa 5
bb 5
laba 1
Ilgiausias sakinys:
Aa aa bb aa bb?
Žodžiai: 5; Simboliai: 15; Šakinio pradzia: 1

```

Skaičiai prie dažniausiai pasikartojančių žodžių (žodžio pasikartojimas) spausdinami norint paprasčiau patikrinti ar teisingai yra atliekamas rūšiavimas.

Rezultatai (ManoKnyga.txt) :

```
Aa aa bb aa bb? Bb  
aa bb bb aa!  
laba
```

#### 4.3.2 Duomenys ir rezultatai 2

Pradiniai failai:

- 1 Aa aa bb aa bb? Bb bb bb bb cc
- 2 aa bb bb xxxx aa!!!
- 3 laba

Šie duomenys tikrina ar teisingai randamas ilgiausias sakinys nors jis ir nesibaigia toje pačioje eilutėje kaip prasidėjo. Tikrinama ar teisingai veikia lygiavimas.

Rezultatai (Rodikliai.txt) :

```
Dazniausiai pasikartojuantys zodziai:  
bb 8  
aa 5  
cc 1  
laba 1  
xxxxx 1  
Ilgiausias sakinys:  
Bb bb bb cc aa bb bb xxxx aa!  
Zodziai: 10; Simboliai: 33; Sakinio pradzia: 1
```

Rezultatai (ManoKnyga.txt) :

```
Aa aa bb aa bb? Bb bb bb bb cc  
aa bb bb xxxx aa!  
laba
```

#### 4.4. Dėstytojo pastabos

## 5. Paveldėjimas

### 5.1. Darbo užduotis

**U5\_24. Kompiuterinis žaidimas.** Žaidimo pasaulyje yra dviejų tipų veikėjai – žaidėjo valdomi herojai bei kompiuterio valdomi „NPC“ (non playable character). Sugrupavote žaidimo veikėjus pagal tris rases ir surašėte jų duomenis į skirtingus failus. Duomenų formatas tokis: pirmoje eilutėje – rasės pavadinimas. Antroje – pradinis miestas. Toliau informacija apie žaidimo veikėjus. Sukurkite klasę „Player“ (savybės - vardas, klasė, gyvybės taškai, mana, žalos taškai, gynybos taškai), kurią paveldės klasės „Hero“ (savybės - jėga, vikumas, intelektas), „NPC“ (savybė – ypatinga galia).

- Raskite kiekvienos klasės daugiausiai gyvybės taškų turintį veikėją, ekrane atspausdinkite jo vardą, rasę, klasę ir gyvybės taškų kiekį.
- Sudarykite bendrą veikėjų, kurie turi gynybos taškų daugiau, nei žalos taškų, sąrašą. Surikiuokite šį sąrašą pagal gynybos ir žalos taškų skaičių didėjimo kryptimi bei įrašykite į failą „StiprusVeikejai.csv“. Suraskite gynybos taškų medianą ir papildykite į tą patį failą.
- Norite, jog kiekviena rasė turėtų bent po vieną kiekvienos klasės herojų ir NPC. Raskite, kokių klasių herojų ar NPC „trūksta“ kiekvienai rasei. Į failą „Trūkstami.csv“ įrašykite kiekvienos rasės pavadinimą, trūkstamų herojų klasių sąrašą, trūkstamų NPC klasių sąrašą.
- Raskite, kurioje rasėje yra stipriausias herojus, jei herojus stiprumą rodo gyvybės ir gynybos taškų sumažinta žalos taškais. Ekrane atspausdinkite visus herojaus duomenis ir jo rasę.

### 5.2. Programos tekstas

```
namespace U5_24KompiuterinisZaidimas
{
    /* U5_24. Kompiuterinis žaidimas.
     * Žaidimo pasaulyje yra dviejų tipų veikėjai – žaidėjo valdomi herojai
     * bei kompiuterio valdomi „NPC“ (non playable character). Sugrupavote
     * žaidimo veikėjus pagal tris rases ir surašėte jų duomenis į skirtingus
     * failus. Duomenų formatas tokis: pirmoje eilutėje – rasės pavadinimas.
     * Antroje – pradinis miestas. Toliau informacija apie žaidimo veikėjus.
     * Sukurkite klasę „Player“ (savybės - vardas, klasė, gyvybės taškai,
     * mana, žalos taškai, gynybos taškai), kurią paveldės klasės „Hero“
     * (savybės - jėga, vikumas, intelektas), „NPC“ (savybė – ypatinga galia).
     * Raskite kiekvienos klasės daugiausiai gyvybės taškų turintį veikėją,
     * ekrane atspausdinkite jo vardą, rasę, klasę ir gyvybės taškų kiekį.
     * Sudarykite bendrą veikėjų, kurie turi gynybos taškų daugiau, nei
     * žalos taškų, sąrašą. Surikiuokite šį sąrašą pagal gynybos ir žalos taškų
     * skaičių didėjimo kryptimi bei įrašykite į failą „StiprusVeikejai.csv“.
     * Suraskite gynybos taškų medianą ir papildykite į tą patį failą.
     * Norite, jog kiekviena rasė turėtų bent po vieną kiekvienos klasės
     * herojų ir NPC. Raskite, kokių klasių herojų ar NPC „trūksta“ kiekvienai
     * rasei. Į failą „Trūkstami.csv“ įrašykite kiekvienos rasės pavadinimą,
     * trūkstamų herojų klasių sąrašą, trūkstamų NPC klasių sąrašą.
     * Raskite, kurioje rasėje yra stipriausias herojus, jei herojus
     * stiprumą rodo gyvybės ir gynybos taškų sumažinta žalos taškais.
     * Ekrane atspausdinkite visus herojaus duomenis ir jo rasę.
    */
}

class Program
{
    static void Main(string[] args)
```

```

{
    //Players and their data are read from the csv files
    HeroRegister registerT =
        InputOutput.ReadHeroes(@"../../../../Troliai.csv");
    HeroRegister registerE =
        InputOutput.ReadHeroes(@"../../../../Elfai.csv");
    HeroRegister registerZ =
        InputOutput.ReadHeroes(@"../../../../Zmones.csv");

    //Deletes file so there would be no text from the past
    File.Delete("Duomenys.txt");
    File.Delete("StiprusVeikejai.csv");

    HeroRegister[] registers
        = new HeroRegister[] { registerT, registerE, registerZ};

    //All heroes and their data are printed to txt file
    InputOutput.PrintHeroesToTxt("Duomenys.txt", registers);

    //All heroes and their data are printed to the console
    Console.WriteLine("Registro informacija:");

    InputOutput.PrintHeroes(registers);

    //First task result is printed
    List<HeroRegister> mostHealth =
        TaskUtils.CollectWithMostHealth(registers);
    InputOutput.PrintMostHealth(mostHealth);

    //Second task result is printed to the csv file
    List<HealthiestPlayer> healthiest =
        TaskUtils.CollectHealthiest(registers);
    decimal mediana = TaskUtils.DefenceMediana(healthiest);
    InputOutput.PrintHealthiest("StiprusVeikejai.csv",
        healthiest, mediana);

    //All existing classes of players are printed to the console
    int x = 0;
    Console.WriteLine("Heroju klasės:");
    Console.WriteLine("Heroju:");
    InputOutput.PrintNumbers(TaskUtils.CollectClasses(registers, x));
    List<int[]> missingHeroClasses =
        TaskUtils.MissingClasses(registers, x);
    x++;
    Console.WriteLine();
    Console.WriteLine("NPC:");
    InputOutput.PrintNumbers(TaskUtils.CollectClasses(registers, x));
    List<int[]> missingNPCClasses =
        TaskUtils.MissingClasses(registers, x);
    Console.WriteLine();

    //Third task result is printed into csv file
    InputOutput.PrintMissingNumbers("Trukstami.csv",
        missingHeroClasses, missingNPCClasses, registers);

    //Forth task result is printed to the console
    List<HeroRegister> strongest =
        TaskUtils.CollectStrongest(registers);
    InputOutput.PrintStrongest(strongest);
}
}

```

```

}

using System.Text; //Library used for text encoding

namespace U5_24KompiuterinisZaidimas
{
    /// <summary>
    /// A class that performs scans and prints
    /// </summary>
    public class InputOutput
    {
        /// <summary>
        /// A method that reads players and their data from the files
        /// </summary>
        /// <param name="fileName"></param>
        /// <returns></returns>
        public static HeroRegister ReadHeroes(string fileName)
        {
            //Reads all lines from a file in UTF-8 encoding
            string[] Lines = File.ReadAllLines(fileName, Encoding.UTF8);

            string race = Lines[0];
            string city = Lines[1];

            HeroRegister players = new HeroRegister(race, city);

            for (int i = 2; i < Lines.Length; i++)
            {
                string[] values = Lines[i].Split(":");

                string name = values[0];
                int number = int.Parse(values[1]);
                int health = int.Parse(values[2]);
                int mana = int.Parse(values[3]);
                int damage = int.Parse(values[4]);
                int defend = int.Parse(values[5]);

                if (values.Count() == 7)
                {
                    string power = values[6];
                    NPC npc = new NPC(name, number, health, mana, damage,
                                      defend, power);
                    players.AllHeroes.Add(npc);
                }
                else
                {
                    int strength = int.Parse(values[6]);
                    int speed = int.Parse(values[7]);
                    int intellect = int.Parse(values[8]);

                    Hero hero = new Hero(name, number, health, mana, damage,
                                      defend, strength, speed, intellect);
                    players.AllHeroes.Add(hero);
                }
            }

            return players;
        }

        /// <summary>
        /// A method that prints all players and their data to the console
    }
}

```

```

    ///</summary>
    ///<param name="registers"></param>
    public static void PrintHeroes(HeroRegister[] registers)
    {
        foreach (HeroRegister register in registers)
        {
            //A table is created to store the data
            Console.WriteLine($"Rasé: {register.race}; " +
                $"Miestas: {register.city}");
            Console.WriteLine(new string('-', 128));
            Console.WriteLine($"| {0,-12} | {1,-5} | {2,-14} | " +
                "{3,-4} | {4,-12} | {5,-14} | {6,-4} | {7,-8} | " +
                "{8,-10} | {9,-14} |", "Vardas", "Klasé",
                "Gyvybés taškai", "Mana", "Žalos taškai",
                "Gynybos taškai", "Jéga", "Vikrumas", "Intelektas",
                "Ypatinga galia");
            Console.WriteLine(new string('-', 128));

            for (int i = 0; i < register.AllHeroes.Count; i++)
            {
                Player player = register.AllHeroes.Get(i);

                Console.WriteLine(player.ToString());
            }

            Console.WriteLine(new string('-', 128));
            Console.WriteLine();
        }
    }

    ///<summary>
    /// A method that prints all players and their data to a txt file
    /// The data is stored in a table
    ///</summary>
    ///<param name="fileNameTxt"></param>
    ///<param name="registers"></param>
    public static void PrintHeroesToTxt(string fileNameTxt,
        HeroRegister[] registers)
    {
        File.AppendAllText(
            fileNameTxt,
            "Registro informacija:\r\n",
            Encoding.UTF8);

        foreach (HeroRegister register in registers)
        {
            List<string> lines = new List<string>();

            lines.Add(String.Format("Rasé: {0}; Miestas: {1}",
                register.race, register.city));
            lines.Add(new string('-', 128));
            lines.Add(String.Format("| {0,-12} | {1,-5} | {2,-14} | " +
                "{3,-4} | {4,-12} | {5,-14} | {6,-4} | {7,-8} | " +
                "{8,-10} | {9,-14} |", "Vardas", "Klasé",
                "Gyvybés taškai", "Mana", "Žalos taškai", "Gynybos taškai",
                "Jéga", "Vikrumas", "Intelektas", "Ypatinga galia"));
            lines.Add(new string('-', 128));

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

```

```

        {
            Player player = register.AllHeroes.Get(i);

            lines.Add(player.ToString());
        }

        lines.Add(new string('-', 128));

        lines.Add("");

        File.AppendAllLines(fileNameTxt, lines, Encoding.UTF8);
    }
}

/// <summary>
/// Method prints all players with the most health
/// </summary>
/// <param name="registers"></param>
public static void PrintMostHealth(List<HeroRegister> registers)
{
    Console.WriteLine("Herojai turintys daugiausiai gyvybės taškų:");

    foreach (HeroRegister register in registers)
    {
        Console.WriteLine(new string('-', 54));
        Console.WriteLine("| {0,-12} | {1,-10} | {2,-5} | " +
            "{3,-14} |", "Vardas", "Rasé", "Klasé", "Gyvybės taškai");
        Console.WriteLine(new string('-', 54));

        for (int i = 0; i < register.AllHeroes.Count; i++)
        {
            Player player = register.AllHeroes.Get(i);

            Console.WriteLine($"| {player.name,-12} | " +
                $"|{register.race,-10} | {player.number,5} | " +
                $"|{player.health,14} |");
        }

        Console.WriteLine(new string('-', 54));
    }

    Console.WriteLine();
}
}

/// <summary>
/// The method prints player classes (without duplicates) to console
/// </summary>
/// <param name="numbers"></param>
public static void PrintNumbers(List<int> numbers)
{
    for (int i = 0; i < numbers.Count; i++)
    {
        Console.WriteLine(String.Format("{0}", numbers[i]));
    }
}

/// <summary>
/// The method prints players missing classes to csv file
/// </summary>
/// <param name="fileName"></param>

```

```

/// <param name="numbersOfHero"></param>
/// <param name="numbersOfNPC"></param>
/// <param name="registers"></param>
public static void PrintMissingNumbers(
    string fileName,
    List<int[]> numbersOfHero, List<int[]> numbersOfNPC,
    HeroRegister[] registers)
{
    List<string> lines = new List<string>();

    lines.Add("Trūkstamos klasės:");

    for (int i = 0; i < registers.Count(); i++)
    {
        lines.Add(registers[i].race + ":");

        if (numbersOfHero[i].Length == 0)
        {
            lines.Add("VIZI");
        }
        else
        {
            foreach (int clas in numbersOfHero[i])
            {
                lines.Add(String.Format("{0}", clas));
            }
        }
    }

    lines.Add("NPC:");

    if (numbersOfNPC[i].Length == 0)
    {
        lines.Add("VIZI");
    }
    else
    {
        foreach (int clas in numbersOfNPC[i])
        {
            lines.Add(String.Format("{0}", clas));
        }
    }
}

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

/// <summary>
/// A method that passes one players register to the printing method
/// for it to print all strongest players
/// </summary>
/// <param name="registers"></param>
public static void PrintStrongest(List<HeroRegister> registers)
{
    Console.WriteLine("Stipriausi herojai:");

    foreach (HeroRegister register in registers)
    {
        //A table is created to store the data
        Console.WriteLine($"Rasė: {register.race}; " +

```

```

        $"Miestas: {register.city}");
Console.WriteLine(new string('-', 128));
Console.WriteLine("| {0,-12} | {1,-5} | {2,-14} | " +
    "{3,-4} | {4,-12} | {5,-14} | {6,-4} | {7,-8} | " +
    "{8,-10} | {9,-14} |", "Vardas", "Klasė",
    "Gyvybės taškai", "Mana", "Žalos taškai",
    "Gynybос taškai", "Jéga", "Vikrumas", "Intelektas",
    "Ypatinga galia");
Console.WriteLine(new string('-', 128));

    for (int i = 0; i < register.AllHeroes.Count; i++)
    {
        Player player = register.AllHeroes.Get(i);

        Console.WriteLine(player.ToString());
    }

    Console.WriteLine(new string('-', 128));

    Console.WriteLine();
}
}

/// <summary>
/// A method that passes one players register to the csv printing
/// method for it to print results to csv file
/// </summary>
/// <param name="fileName"></param>
/// <param name="healthiest"></param>
/// <param name="mediana"></param>
public static void PrintHealthiest(string fileName,
    List<HealthiestPlayer> healthiest, decimal mediana)
{
    if (healthiest.Count > 0)
    {
        PrintAllHeroesToCSV(fileName, healthiest, mediana);
    }
    else
    {
        List<string> lines = new List<string>();

        lines.Add("Néra tokiu heroju");

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

/// <summary>
/// A method that prints all players and their data to a csv file
/// The data is stored in a table
/// </summary>
/// <param name="fileName"></param>
/// <param name="healthiest"></param>
/// <param name="mediana"></param>
public static void PrintAllHeroesToCSV(string fileName,
    List<HealthiestPlayer> healthiest, decimal mediana)
{
    List<string> lines = new List<string>();

    foreach (HealthiestPlayer health in healthiest)
    {

```

```

        lines.Add(String.Format($"Rasé: {health.register.race}; " +
            $"Miestas: {health.register.city}"));
        lines.Add(String.Format($"{"Vardas"}; {"Klasė"}; " +
            $"{"Gyvybės taškai"}; {"Mana"}; {"Žalos taškai"}; " +
            $"{"Gynybos taškai"}; {"Jéga"}; {"Vikrumas"}; " +
            $"{"Intelektas"}; {"Ypatinga galia"}));
    }

    lines.Add(lines.Add(health.player.ToCSV()));
}

lines.Add("");
lines.Add(String.Format($"Stipriausių veikėjų gynybos " +
    $"taškų mediana: {mediana}"));

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

namespace U5_24KompiuterinisZaidimas
{
    public static class TaskUtils
    {
        /// <summary>
        /// Method collects all players with biggest health points
        /// </summary>
        /// <param name="registers"></param>
        /// <returns></returns>
        public static List<HeroRegister> CollectWithMostHealth(
            HeroRegister[] registers)
        {
            HeroContainer[] mostHealth = new HeroContainer[registers.Length];
            List<HeroRegister> result = new List<HeroRegister>();

            for (int i = 0; i < registers.Length; i++)
            {
                mostHealth[i] = registers[i].AllHeroes.FindAllMostHealth();
                result.Add(new HeroRegister(registers[i].race, registers[i].city,
                    mostHealth[i]));
            }

            return result;
        }

        /// <summary>
        /// Method collects all different classes
        /// </summary>
        /// <param name="registers"></param>
        /// <param name="x"></param>
        /// <returns></returns>
        public static List<int> CollectClasses(HeroRegister[] registers, int x)
        {
            List<int> results = new List<int>();

            foreach (HeroRegister register in registers)
            {
                List<int> registerClasses = register.AllHeroes.FindClasses(x);

                foreach (int clas in registerClasses)
                {
                    if (!results.Contains(clas))

```

```

        {
            results.Add(clas);
        }
    }

    results.Sort();

    return results;
}

/// <summary>
/// Method finds all missing classes and puts them in one list
/// </summary>
/// <param name="registers"></param>
/// <returns></returns>
public static List<int[]> MissingClasses(HeroRegister[] registers, int x)
{
    List<int[]> result = new List<int[]>();
    List<int> allClasses = CollectClasses(registers, x);

    foreach (HeroRegister register in registers)
    {
        result.Add(allClasses.Except(
            register.AllHeroes.FindClasses(x)).ToArray());
    }

    return result;
}

/// <summary>
/// Method collects all the strongest players into one list
/// </summary>
/// <param name="registers"></param>
/// <returns></returns>
public static List<HeroRegister> CollectStrongest(
    HeroRegister[] registers)
{
    HeroContainer[] strongest = new HeroContainer[registers.Length];
    double strength = 0;

    for (int i = 0; i < registers.Length; i++)
    {
        strongest[i] = registers[i].AllHeroes.FindAllStrongest();

        if (strongest[i].Get(0).GetStrength() > strength)
        {
            strength = strongest[i].Get(0).GetStrength();
        }
    }

    return CollectStrongest(registers, strongest, strength);
}

/// <summary>
/// Mthod collects strongest players into the List
/// </summary>
/// <param name="registers"></param>
/// <param name="strongest"></param>
/// <param name="strength"></param>

```

```

/// <returns></returns>
public static List<HeroRegister> CollectStrongest(
    HeroRegister[] registers,
    HeroContainer[] strongest,
    double strength)
{
    List<HeroRegister> result = new List<HeroRegister>();

    for (int i = 0; i < strongest.Length; i++)
    {
        if (strongest[i].Get(0).GetStrength() == strength)
        {
            result.Add(new HeroRegister(
                registers[i].race,
                registers[i].city,
                strongest[i]
            ));
        }
    }

    return result;
}

/// <summary>
/// Method collects all players which health is greater than damage
/// </summary>
/// <param name="registers"></param>
/// <returns></returns>
public static List<HealthiestPlayer> CollectHealthiest(
    HeroRegister[] registers)
{
    List<HealthiestPlayer> result = new List<HealthiestPlayer>();

    for (int i = 0; i < registers.Length; i++)
    {
        HeroContainer container = registers[i].AllHeroes.MoreHealth();

        for (int j = 0; j < container.Count; j++)
        {
            result.Add(new HealthiestPlayer(container.Get(j),
                registers[i]));
        }
    }

    Sort(result);

    return result;
}

/// <summary>
/// Method is sorting players
/// </summary>
private static void Sort(List<HealthiestPlayer> list)
{
    bool flag = true;

    while (flag)
    {
        flag = false;

```

```

        for (int i = 0; i < list.Count - 1; i++)
    {
        HealthiestPlayer one = list[i];
        HealthiestPlayer two = list[i + 1];

        if (one.CompareTo(two) > 0)
        {
            list[i] = two;
            list[i + 1] = one;
            flag = true;
        }
    }
}

/// <summary>
/// Method finds the median
/// </summary>
/// <param name="healthiest"></param>
/// <returns></returns>
public static decimal DefenceMediana(List<HealthiestPlayer> healthiest)
{
    if (healthiest.Count % 2 == 0 && healthiest.Count > 0)
    {
        int temp = healthiest.Count / 2;
        decimal rez = healthiest[temp - 1].player.defend +
        healthiest[temp].player.defend;

        return rez / 2;
    }
    else if (healthiest.Count > 0)
    {
        int temp = healthiest.Count / 2 + 1;

        return healthiest[temp - 1].player.defend;
    }
    else return 0;
}
}

namespace U5_24KompiuterinisZaidimas
{
    /// <summary>
    /// Class which holds container of players and their race and city
    /// </summary>
    public class HeroRegister
    {
        //Konteineris čia neturėtu būti, bet nebereikia keisti
        public HeroContainer AllHeroes { get; set; }
        public string race { get; private set; }
        public string city { get; private set; }

        /// <summary>
        /// Creating constructor with empty container, race and city
        /// </summary>
        public HeroRegister(string race, string city)
        {
            this.AllHeroes = new HeroContainer();
            this.race = race;
            this.city = city;
        }
    }
}

```

```

    ///<summary>
    /// Constructor with container container, race and city
    ///</summary>
    public HeroRegister(string race, string city, HeroContainer container)
    {
        this.AllHeroes = container;
        this.race = race;
        this.city = city;
    }
}

namespace U5_24KompiuterinisZaidimas
{
    ///<summary>
    /// Class in which the container is made
    ///</summary>
    public class HeroContainer
    {
        private int Capacity;
        private Player[] players;
        public int Count { get; private set; }

        ///<summary>
        /// Creating a container
        ///</summary>
        ///<param name="capacity"></param>
        public HeroContainer(int capacity = 16)
        {
            this.players = new Player[capacity];
            this.Capacity = capacity;
        }

        ///<summary>
        /// Method which ensures if there is enough space in the container
        ///</summary>
        ///<param name="minimum"></param>
        private void EnsureCapacity(int minimum)
        {
            if (minimum > Capacity)
            {
                Player[] temp = new Player[minimum];
                for (int i = 0; i < this.Count; i++)
                {
                    temp[i] = this.players[i];
                }

                this.Capacity = minimum;
                this.players = temp;
            }
        }

        ///<summary>
        /// Method checks if it is not the same player
        ///</summary>
        ///<param name="player"></param>
        ///<returns></returns>
        public bool Contains(Player player)
        {
            for (int i = 0; i < this.Count; i++)
            {

```

```

        if (this.players[i].Equals(player))
        {
            return true;
        }
    }

    return false;
}

/// <summary>
/// Method adds Player to the container
/// </summary>
/// <param name="player"></param>
public void Add(Player player)
{
    if (this.Count == this.Capacity)
    {
        this.EnsureCapacity(this.Capacity * 2);
    }

    if (this.Contains(player))
    {
        return;
    }

    this.players[this.Count++] = player;
}

/// <summary>
/// Method returns the Player by the index
/// </summary>
/// <param name="index"></param>
/// <returns></returns>
public Player Get(int index)
{
    return this.players[index];
}

/// <summary>
/// Method puts the player in the place the index show
/// </summary>
/// <param name="player"></param>
/// <param name="index"></param>
public void Put(Player player, int index)
{
    if (index >= 0 && index < this.Count)
    {
        this.players[index] = player;
    }
}

/// <summary>
/// Method inserts player into the place which the index show
/// </summary>
/// <param name="player"></param>
/// <param name="index"></param>
public void Insert(Player player, int index)
{
    if (index >= 0 && index < this.Count)
    {
        if (this.Count + 1 > this.Capacity)

```

```

        {
            this.EnsureCapacity(this.Capacity * 2);
        }

        for (int i = this.Count; i > index + 1; i--)
        {
            this.players[i] = this.players[i - 1];
        }

        this.players[index] = player;
        this.Count++;
    }
}

/// <summary>
/// Method removes player which is in the index place
/// </summary>
/// <param name="index"></param>
public void RemoveAt(int index)
{
    if (index >= 0 && index < this.Count)
    {
        for (int i = index + 1; i < this.Count; i++)
        {
            this.players[i - 1] = this.players[i];
        }

        this.Count--;
    }
}

/// <summary>
/// Method removes player which is equal to the given one
/// </summary>
/// <param name="player"></param>
public void Remove(Player player)
{
    for (int i = 0; i < this.Count; i++)
    {
        if (player == this.players[i])
        {
            this.RemoveAt(i);
            break;
        }
    }
}

/// <summary>
/// Method finds all players who has the most health points
/// </summary>
/// <returns></returns>
public HeroContainer FindAllMostHealth()
{
    HeroContainer mostHealthiest = new HeroContainer();

    int healthiest = FindMostHealth();

    for (int i = 0; i < this.Count; i++)
    {
        if (healthiest == this.players[i].health)
        {

```

```

                mostHealthiest.Add(this.players[i]);
            }
        }

        return mostHealthiest;
    }

/// <summary>
/// Method searches for the highest health points
/// </summary>
/// <returns></returns>
private int FindMostHealth()
{
    int healthiest = -1;

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

        if (healthiest < player.health)
        {
            healthiest = player.health;
        }
    }

    return healthiest;
}

/// <summary>
/// Method returns players who has more defence points than damage
/// points
/// </summary>
/// <returns></returns>
public HeroContainer MoreHealth()
{
    HeroContainer healthier = new HeroContainer();

    for (int i = 0; i < this.Count; i++)
    {
        Player player = this.Get(i);

        if (player.defend > player.damage)
        {
            healthier.Add(player);
        }
    }

    return healthier;
}

/// <summary>
/// Method combines two containers into one
/// </summary>
/// <param name="container"></param>
/// <returns></returns>
public HeroContainer CombineContainers(HeroContainer container)
{
    HeroContainer result = new HeroContainer(this.Capacity
        + container.Capacity);

    result.AddAll(this.players);
}

```

```

        result.AddAll(container.players);

        return result;
    }

    /// <summary>
    /// Method adds each player from the array
    /// </summary>
    /// <param name="players"></param>
    public void AddAll(Player[] players)
    {
        foreach (Player player in players)
        {
            Add(player);
        }
    }

    /// <summary>
    /// Method which finds all the different classes of the players
    /// and places them in one list
    /// </summary>
    /// <returns></returns>
    public List<int> FindClasses(int who)
    {
        //New list is created for the new list of the classes
        List<int> numbers = new List<int>();

        for (int i = 0; i < this.Count; i++)
        {
            if (who == 1 && players[i] is NPC)
            {
                int nr = this.players[i].number;

                if (!numbers.Contains(nr))
                {
                    //Classes which met the conditions are added to
                    //the new list
                    numbers.Add(nr);
                }
            }
            else if (who == 0 && players[i] is Hero)
            {
                int nr = this.players[i].number;

                if (!numbers.Contains(nr))
                {
                    //Classes which met the conditions are added to
                    //the new list
                    numbers.Add(nr);
                }
            }
        }

        numbers.Sort();
        return numbers;
    }

    /// <summary>
    /// Method which finds all the strongest players from the list
    /// </summary>

```

```

    /// <returns></returns>
    public HeroContainer FindAllStrongest()
    {
        HeroContainer strongContainer = new HeroContainer();

        double powerfull = FindStrength();

        for (int i = 0; i < this.Count; i++)
        {
            Player hero = this.players[i];

            if (powerfull == hero.GetStrength())
            {
                strongContainer.Add(hero);
            }
        }

        return strongContainer;
    }

    /// <summary>
    /// Method which finds the biggest strength of the players
    /// </summary>
    /// <returns></returns>
    public double FindStrength()
    {
        if (this.Count == 0)
        {
            return 0;
        }

        double strength = this.players[0].GetStrength();

        for (int i = 0; i < this.Count; i++)
        {
            double heroStrength = this.players[i].GetStrength();
            if (strength < heroStrength)
            {
                strength = heroStrength;
            }
        }

        return strength;
    }
}

namespace U5_24KompiuterinisZaidimas
{
    /// <summary>
    /// Class for holding each players seperately and register for each
    /// player to know its race and city
    /// </summary>
    public class HealthiestPlayer
    {
        public Player player { get; }
        public HeroRegister register { get; }

        /// <summary>
        /// Creates a constructor
        /// </summary>
        /// <param name="player"></param>

```

```

    ///> <param name="register"></param>
    public HealthiestPlayer(Player player, HeroRegister register)
    {
        this.player = player;
        this.register = register;
    }

    ///> <summary>
    ///> Compares players defence points for sorting if those match than
    ///> compares by the damage points
    ///> </summary>
    ///> <param name="other"></param>
    ///> <returns></returns>
    public int CompareTo(HealthiestPlayer other)
    {
        int defence = this.player.defend.CompareTo(other.player.defend);
        int damage = this.player.damage.CompareTo(other.player.damage);

        if (defence != 0)
        {
            return defence;
        }
        else
        {
            return damage;
        }
    }
}

namespace U5_24KompiuterinisZaidimas
{
    ///> <summary>
    ///> The class in which the constructor is created
    ///> </summary>
    public abstract class Player
    {
        public string name { get; set; }
        public int number { get; set; }
        public int health { get; set; }
        public int mana { get; set; }
        public int damage { get; set; }
        public int defend { get; set; }

        ///> <summary>
        ///> Creates a Player constructor
        ///> </summary>
        ///> <param name="name"></param>
        ///> <param name="number"></param>
        ///> <param name="health"></param>
        ///> <param name="mana"></param>
        ///> <param name="damage"></param>
        ///> <param name="defend"></param>
        public Player(string name, int number,
                      int health, int mana, int damage, int defend)
        {
            this.name = name;
            this.number = number;
            this.health = health;
            this.mana = mana;
            this.damage = damage;
            this.defend = defend;
        }
    }
}

```

```

    }

    /// <summary>
    /// Overrides ToString method
    /// </summary>
    /// <returns></returns>
    public override string ToString()
    {
        string line;

        line = String.Format("| {0,-12} | {1,5} | {2,14} | " +
            "{3,4} | {4,12} | {5,14} | ", this.name, this.number,
            this.health, this.mana, this.damage, this.defend);

        return line;
    }

    /// <summary>
    /// Method for printing results to the csv file
    /// </summary>
    /// <returns></returns>
    public abstract string ToCSV();

    /// <summary>
    /// Counts strength
    /// </summary>
    /// <returns></returns>
    public double GetStrength()
    {
        return this.health + this.defend - this.damage;
    }
}

namespace U5_24KompiuterinisZaidimas
{
    /// <summary>
    /// Hero class inherits Player class
    /// </summary>
    public class Hero : Player
    {
        public int strength { get; set; }
        public int speed { get; set; }
        public int intellect { get; set; }

        /// <summary>
        /// Creates a hero constructor with base class elements
        /// </summary>
        /// <param name="name"></param>
        /// <param name="number"></param>
        /// <param name="health"></param>
        /// <param name="mana"></param>
        /// <param name="damage"></param>
        /// <param name="defend"></param>
        /// <param name="strength"></param>
        /// <param name="speed"></param>
        /// <param name="intellect"></param>
        public Hero(string name, int number,
            int health, int mana, int damage, int defend, int strength,
            int speed, int intellect) : base(name, number,
                health, mana, damage, defend)
    }
}

```

```

        this.strength = strength;
        this.speed = speed;
        this.intellect = intellect;
    }

    /// <summary>
    /// Overrides ToString method
    /// </summary>
    /// <returns></returns>
    public override string ToString()
    {
        string line;

        line = base.ToString() + String.Format(" {0,4} | {1,8} | " +
            "{2,10} | {3,-14} | ", this.strength,
            this.speed, this.intellect, "---");

        return line;
    }

    /// <summary>
    /// Overrides ToCSV method for printing results to csv file
    /// </summary>
    /// <returns></returns>
    public override string ToCSV()
    {
        string line;

        line = String.Format("${}");  

  

        line = String.Format("${this.name}; {this.number}; " +
            "${this.health}; {this.mana}; {this.damage}; " +
            "${this.defend}; {this.strength}; " +
            "${this.speed}; {this.intellect}; {"---"}");

        return line;
    }
}

namespace U5_24KompiuterinisZaidimas
{
    /// <summary>
    /// NPC inherits Player class
    /// </summary>
    public class NPC : Player
    {
        public string power { get; set; }

        /// <summary>
        /// Creates constructor with base class elements
        /// </summary>
        /// <param name="name"></param>
        /// <param name="number"></param>
        /// <param name="health"></param>
        /// <param name="mana"></param>
        /// <param name="damage"></param>
        /// <param name="defend"></param>
        /// <param name="power"></param>
        public NPC(string name, int number,

```

```

        int health, int mana, int damage, int defend,
        string power) : base(name, number, health,
            mana, damage, defend)
    {
        this.power = power;
    }

    /// <summary>
    /// Overrides ToString method for printing results
    /// </summary>
    /// <returns></returns>
    public override string ToString()
    {
        string line;

        line = base.ToString() + String.Format(" {0,4} | {1,8} | " +
            "{2,10} | {3,-14} |", "----", "----", "----", this.power);

        return line;
    }

    /// <summary>
    /// Overrides ToCSV method for printing results to csv file
    /// </summary>
    /// <returns></returns>
    public override string ToCSV()
    {
        string line;

        line = String.Format($"'{this.name}; {this.number}; " +
            $"'{this.health}; {this.mana}; {this.damage}; " +
            $"'{this.defend}; {"---"}; " +
            $"'{---}; {"---"}; {this.power}'");

        return line;
    }
}

```

### 5.3. Pradiniai duomenys ir rezultatai

#### 5.3.1 Duomenys ir rezultatai 1

Pradiniai failai:

---

```

1 Troliai
2 Asgardas
3 Aldas;1;280;10;55;55;100;100;100
4 Rugis;2;100;10;60;60;150;100;100
5 Matas;3;150;10;60;60;110;100;100
6 Rokas;4;170;10;55;50;Rega
7 Tadas;5;100;10;120;90;Liepsna
8 Lopas;4;170;10;55;50;Rega
9 Grybas;3;170;10;55;50;110;100;100

```

```

1 Elfai
2 Atlanta
3 Jonas;1;175;10;60;55;100;100;100
4 Petra;2;150;10;60;60;100;100;100
5 Rimas;3;140;10;60;60;100;100;100
6 Lukas;4;15;10;120;90;Rega
7 Arnas;5;10;10;120;90;Liepsna
8 Karolis;2;170;10;55;50;110;100;100
9 Simas;3;170;10;55;50;110;100;100
--
```

```

1 Žmones
2 Kaunas
3 Mikas;1;170;10;55;55;100;100;100
4 Binkis;2;100;10;60;55;150;100;100
5 Justas;3;160;10;55;50;110;100;100
6 Justinas;4;150;10;120;90;Paslaptis
7 Kristupas;5;100;10;120;90;Dangus|
8
```

Šie duomenys tikrina ar veikia pirma užduotis ir yra atspausdinamas kiekvienos rasės daugiausiai gyvybės taškų turintis žaidėjas. Tikrinama antra užduotis - ar atspausdinamas į csv failą vienas stiprus žaidėjas. Tikrinama trečia užduotis - ar teisingai atrenkamos ir į csv failą atspausdinamos trūkstamos veikėjų klasės (kai tokiu nėra). Tikrinama ketvirta užduotis - ar atspausdinami stipriausi žaidėjai.

Rezultatai (Duomenys.txt) :

Duomenys.txt										
Registro informacija:										
Rasė: Troliai; Miestas: Asgardas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Aldas	1	280	10	55	55	100	100	100	100	---
Rugis	2	100	10	60	60	150	100	100	100	---
Matas	3	150	10	60	60	110	100	100	100	---
Rokas	4	170	10	55	50	---	---	---	---	Rega
Tadas	5	100	10	120	90	---	---	---	---	Liepsna
Lopas	4	170	10	55	50	---	---	---	---	Rega
Grybas	3	170	10	55	50	110	100	100	100	---

  

Duomenys.txt										
Registro informacija:										
Rasė: Elfai; Miestas: Atlanta										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Jonas	1	175	10	60	55	100	100	100	100	---
Petra	2	150	10	60	60	100	100	100	100	---
Rimas	3	140	10	60	60	100	100	100	100	---
Lukas	4	15	10	120	90	---	---	---	---	Rega
Arnas	5	10	10	120	90	---	---	---	---	Liepsna
Karolis	2	170	10	55	50	110	100	100	100	---
Simas	3	170	10	55	50	110	100	100	100	---

  

Duomenys.txt										
Registro informacija:										
Rasė: Žmones; Miestas: Kaunas										
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	
Mikas	1	170	10	55	55	100	100	100	100	---
Binkis	2	100	10	60	55	150	100	100	100	---
Justas	3	160	10	55	50	110	100	100	100	---
Justinas	4	150	10	120	90	---	---	---	---	Paslaptis
Kristupas	5	100	10	120	90	---	---	---	---	Dangus

Rezultatai (Konsolėje) :

Registro informacija:  
Rasé: Troliai; Miestas: Asgardas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Aldas	1	280	10	55	55	100	100	100	---
Rugis	2	100	10	60	60	150	100	100	---
Matas	3	150	10	60	60	110	100	100	---
Rokas	4	170	10	55	50	---	---	---	Rega
Tadas	5	100	10	120	90	---	---	---	Liepsna
Lopas	4	170	10	55	50	---	---	---	Rega
Grybas	3	170	10	55	50	110	100	100	---

Rasé: Elfai; Miestas: Atlanta

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Jonas	1	175	10	60	55	100	100	100	---
Petra	2	150	10	60	60	100	100	100	---
Rimas	3	140	10	60	60	100	100	100	---
Lukas	4	15	10	120	90	---	---	---	Rega
Arnas	5	10	10	120	90	---	---	---	Liepsna
Karolis	2	170	10	55	50	110	100	100	---
Simas	3	170	10	55	50	110	100	100	---

Rasé: Žmones; Miestas: Kaunas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Mikas	1	170	10	55	55	100	100	100	---
Binkis	2	100	10	60	55	150	100	100	---
Justas	3	160	10	55	50	110	100	100	---
Justinas	4	150	10	120	90	---	---	---	Paslaptis
Kristupas	5	100	10	120	90	---	---	---	Dangus

Herojai turintys daugiausiai gyvybės taškų:

Vardas	Rasé	Klasė	Gyvybės taškai
Aldas	Troliai	1	280

Vardas	Rasé	Klasė	Gyvybės taškai
Jonas	Elfai	1	175

Vardas	Rasé	Klasė	Gyvybės taškai
Mikas	Žmones	1	170

Herojų klasės:

Herojų:

1  
2  
3

NPC:

4  
5

Stipriausi herojai:

Rasé: Troliai; Miestas: Asgardas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Aldas	1	280	10	55	55	100	100	100	---

Rezultatai (StiprusVeikejai.csv) :

Nėra tokių herojų

Rezultatai (Trukstami.csv) :

Trūkstamos klasės:
Troliai:
Hero:
VISI
NPC:
VISI
Elfai:
Hero:
VISI
NPC:
VISI
Žmones:
Hero:
VISI
NPC:
VISI

### 5.3.2 Duomenys ir rezultatai 2

Pradiniai failai:

---

```
1 Troliai
2 Asgardas
3 Aldas;1;150;10;55;50;100;100;100
4 Rugis;2;100;10;60;60;150;100;100
5 Matas;3;150;10;60;60;110;100;100
6 Rokas;4;170;10;55;60;Rega
7 Tadas;5;100;10;120;90;Liepsna
8 Lopas;4;170;10;55;60;Rega
9 Grybas;3;170;10;55;60;110;100;100
10 |
```

---

```
1 Elfai
2 Atlanta
3 Jonas;1;150;10;60;60;100;100;100
4 Petra;2;150;10;60;60;100;100;100
5 Rimas;2;140;10;60;60;100;100;100
6 Lukas;4;15;10;120;90;Rega
7 Arnas;5;10;10;120;90;Liepsna
8 Karolis;2;170;10;55;60;110;100;100
9 Simas;2;170;10;55;60;110;100;100
10 |
```

```

1 Žmones
2 Kaunas
3 Mikas;1;150;10;55;50;100;100;100
4 Binkis;1;100;10;60;60;160;100;100
5 Justas;1;170;10;55;60;110;100;100
6 Justinas;4;150;10;120;90;Paslaptis
7 Kristupas;6;100;10;120;90;Dangus
8

```

Šie duomenys tikrina ar veikia pirmą užduotis atrenka daugiau nei viena kiekvienos rasės žaidėja. Tikrinama antra užduotis - ar atspausdinamas iš csv failą daugiau vienas stiprus žaidėjas ir ar jie yra surūšiuojami. Tikrinama trečia užduotis - ar teisingai atrenkamos ir iš csv failą atspausdinamos trūkstamos veikėjų klasės (kai tokiai yra). Tikrinama ketvirta užduotis - ar atspausdinami stipriausi žaidėjai, kai jų yra daugiau nei vienas.

Rezultatai (Duomenys.txt) :

Duomenys.txt											
Registro informacija:											
Rasė: Troliai; Miestas: Asgardas											
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	Rega	Liepsna
Aldas	1	150	10	55	50	100	100	100	---		
Rugis	2	100	10	60	60	150	100	100	---		
Matas	3	150	10	60	60	110	100	100	---		
Rokas	4	170	10	55	60	---	---	---	---	Rega	
Tadas	5	100	10	120	90	---	---	---	---	Liepsna	
Lopas	4	170	10	55	60	---	---	---	---	Rega	
Grybas	3	170	10	55	60	110	100	100	---		

  

Duomenys.txt											
Registro informacija:											
Rasė: Elfai; Miestas: Atlanta											
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	Rega	Liepsna
Jonas	1	150	10	60	60	100	100	100	---		
Petra	2	150	10	60	60	100	100	100	---		
Rimas	2	140	10	60	60	100	100	100	---		
Lukas	4	15	10	120	90	---	---	---	---	Rega	
Arnas	5	10	10	120	90	---	---	---	---	Liepsna	
Karolis	2	170	10	55	60	110	100	100	---		
Simas	2	170	10	55	60	110	100	100	---		

  

Duomenys.txt											
Registro informacija:											
Rasė: Žmones; Miestas: Kaunas											
Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia	Rega	Liepsna
Mikas	1	150	10	55	50	100	100	100	---		
Binkis	1	100	10	60	60	160	100	100	---		
Justas	1	170	10	55	60	110	100	100	---		
Justinas	4	150	10	120	90	---	---	---	---	Paslaptis	
Kristupas	6	100	10	120	90	---	---	---	---	Dangus	

Rezultatai (Konsolėje):

Registro informacija:  
Rasé: Troliai; Miestas: Asgardas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Aldas	1	150	10	55	50	100	100	100	---
Rugis	2	100	10	60	60	150	100	100	---
Matas	3	150	10	60	60	110	100	100	---
Rokas	4	170	10	55	60	---	---	---	Rega
Tadas	5	100	10	120	90	---	---	---	Liepsna
Lopas	4	170	10	55	60	---	---	---	Rega
Grybas	3	170	10	55	60	110	100	100	---

Rasé: Elfai; Miestas: Atlanta

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Jonas	1	150	10	60	60	100	100	100	---
Petra	2	150	10	60	60	100	100	100	---
Rimas	2	140	10	60	60	100	100	100	---
Lukas	4	15	10	120	90	---	---	---	Rega
Arnas	5	10	10	120	90	---	---	---	Liepsna
Karolis	2	170	10	55	60	110	100	100	---
Simas	2	170	10	55	60	110	100	100	---

Rasé: Žmones; Miestas: Kaunas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Mikas	1	150	10	55	50	100	100	100	---
Binkis	1	100	10	60	60	160	100	100	---
Justas	1	170	10	55	60	110	100	100	---
Justinas	4	150	10	120	90	---	---	---	Paslaptis
Kristupas	6	100	10	120	90	---	---	---	Dangus

Herojai turintys daugiausiai gyvybės taškų:

Vardas	Rasé	Klasė	Gyvybės taškai
Rokas	Troliai	4	170
Lopas	Troliai	4	170
Grybas	Troliai	3	170

Vardas	Rasé	Klasė	Gyvybės taškai
Karolis	Elfai	2	170
Simas	Elfai	2	170

Vardas	Rasé	Klasė	Gyvybės taškai
Justas	Žmones	1	170

Herojų klasės:

Heroju:

1  
2  
3

NPC:

4  
5  
6

Stipriausi herojai:

Rasé: Troliai; Miestas: Asgardas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Rokas	4	170	10	55	60	---	---	---	Rega
Lopas	4	170	10	55	60	---	---	---	Rega
Grybas	3	170	10	55	60	110	100	100	---

Rasé: Elfai; Miestas: Atlanta

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Karolis	2	170	10	55	60	110	100	100	---
Simas	2	170	10	55	60	110	100	100	---

Rasé: Žmones; Miestas: Kaunas

Vardas	Klasė	Gyvybės taškai	Mana	Žalos taškai	Gynybos taškai	Jéga	Vikrumas	Intelektas	Ypatinga galia
Justas	1	170	10	55	60	110	100	100	---

Rezultatai (StiprusVeikejai.csv) :

A	B	C	D	E	F	G	H	I	J	K
1 Rasé: Troliai Miestas: Asgardas										
2 Vardas Klasé Gyvybės taš Mana Žalos taškai Gynybos taš Jéga Vikrumas Intelektas Ypatinga galia										
3 Rokas 4 170 10 55 60 --- --- --- Rega										
4 Rasé: Troliai Miestas: Asgardas										
5 Vardas Klasé Gyvybės taš Mana Žalos taškai Gynybos taš Jéga Vikrumas Intelektas Ypatinga galia										
6 Lopas 4 170 10 55 60 --- --- --- Rega										
7 Rasé: Troliai Miestas: Asgardas										
8 Vardas Klasé Gyvybės taš Mana Žalos taškai Gynybos taš Jéga Vikrumas Intelektas Ypatinga galia										
9 Grybas 3 170 10 55 60 110 100 100 ---										
10 Rasé: Elfai Miestas: Atlanta										
11 Vardas Klasé Gyvybės taš Mana Žalos taškai Gynybos taš Jéga Vikrumas Intelektas Ypatinga galia										
12 Karolis 2 170 10 55 60 110 100 100 ---										
13 Rasé: Elfai Miestas: Atlanta										
14 Vardas Klasé Gyvybės taš Mana Žalos taškai Gynybos taš Jéga Vikrumas Intelektas Ypatinga galia										
15 Simas 2 170 10 55 60 110 100 100 ---										
16 Rasé: Žmone Miestas: Kaunas										
17 Vardas Klasé Gyvybės taš Mana Žalos taškai Gynybos taš Jéga Vikrumas Intelektas Ypatinga galia										
18 Justas 1 170 10 55 60 110 100 100 ---										
19										
20 Stipriausių veikėjų gynybos taškų mediana: 60										
21										

Rezultatai (Trukstami.csv) :

Trūkstamos klasės:
Troliai:
Hero:
VISI
NPC:
6
Elfai:
Hero:
3
NPC:
6
Žmones:
Hero:
2
3
NPC:
5

## 5.4. Dėstytojo pastabos

