

Kauno technologijos universitetas

Informatikos fakultetas

Objektinis programavimas I (P175B118)

Laboratorinių darbų ataskaita

Aistis Jakutonis IFF-3/1

Studentas

Lekt. Kęstutis Simonavičius

Dėstytojas

TURINYS

1.	Duc	Duomenų klasė4			
	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		
2.	Skaičiavimų klasė				
	2.1.	Darbo užduotis	13		
	2.2.	Programos tekstas	13		
	2.3.	Pradiniai duomenys ir rezultatai	13		
	2.4.	Dėstytojo pastabos	13		
3.	Konteineris				
	3.1.	Darbo užduotis	14		
	3.2.	Programos tekstas	14		
	3.3.	Pradiniai duomenys ir rezultatai	14		
	3.4.	Dėstytojo pastabos	14		
4.	Tek	sto analizė ir redagavimas	15		
	4.1.	Darbo užduotis	15		
	4.2.	Programos tekstas	15		
	4.3.	Pradiniai duomenys ir rezultatai	15		
	4.4.	Dėstytojo pastabos	15		
5.	Paveldėjimas				
	5.1.	Darbo užduotis	16		
	5.2.	Programos tekstas	16		
	5.3.	Pradiniai duomenys ir rezultatai	16		

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, jėga, vikrumas, intelektas, ypatinga galia.

- Raskite daugiausiai gyvybės taškų turintį herojų, ekrane atspausdinkite jo vardą, rasę, klasę ir gyvybės taškų kiekį. Jei yra keli, spausdinkite visus.
- Raskite žaidėją, kurio gynybos ir žalos taškų skirtumas yra mažiausias. Atspausdinkite informaciją apie žaidėją į ekraną. Jei yra keli, spausdinkite visus.
- Sudarykite visų herojų klasių sąrašą, klasių pavadinimus įrašykite į failą "Klasės.csv".
 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, jėga, vikrumas,
     *intelektas, ypatinga galia.
        · Raskite daugiausiai gyvybės taškų turintį herojų, ekrane
        atspausdinkite jo vardą, rasę, klasę ir gyvybės taškų kiekį. Jei yra
        keli, spausdinkite visus.
        · Raskite žaidėją, kurio gynybos ir žalos taškų skirtumas yra
        mažiausias. Atspausdinkite informaciją apie žaidėją į ekraną. Jei yra
        keli, spausdinkite visus.
        · Sudarykite visu heroju klasių saraša, klasių pavadinimus įrašykite 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", "Rasė", "Klasė", "Gyvybės taškai", "Mana", "Žalos taškai", "Gynybos taškai", "Jėga", "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\} \mid \{1,-15\} \mid \{2,-5\} \mid \{3,-14\}" +
                " | {4,-4} | {5,-12} | {6,-14} | {7,-4} | {8,-8} | " +
                "{9,-10} | {10,-14} |", "Vardas", "Rasė", "Klasė",
                "Gyvybės taškai", "Mana", "Žalos taškai", "Gynybos taškai",
                "Jėga", "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 gyvybės taškų:");
            Console.WriteLine(new string('-', 59));
            Console.WriteLine("| \{0,-12\} \mid \{1,-15\} \mid \{2,5\} \mid \{3,14\} \mid",
```

```
"Vardas", "Rasė", "Klasė", "Gyvybės 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 " +
        "tašku:");
    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", "Rasė", "Klasė", "Gyvybės taškai", "Mana", "Žalos taškai", "Gynybos taškai", "Jėga", "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\} \mid \{5,12\} \mid \{6,14\} \mid \{7,4\} \mid \{8,8\} \mid \{9,10\} \mid "+
             "{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
//"Klases.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("Herojų klasės:");
    for (int i = 0; i < numbers.Count; i++)</pre>
        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)
        // {
m 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

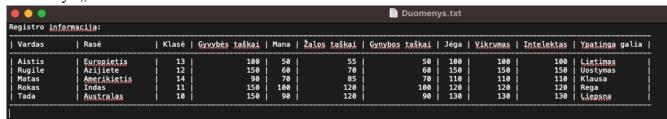
1.3.1 Duomenys ir rezultatai 1

Pradinis failas:

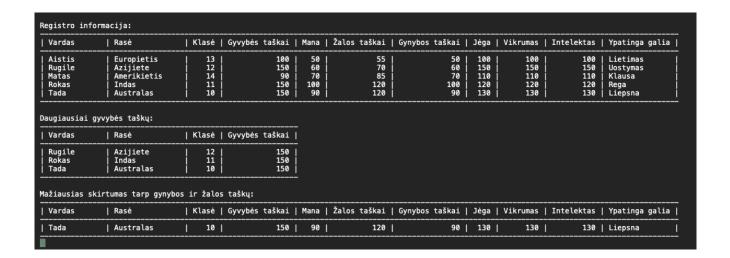
```
Aistis;Europietis;13;100;50;55;50;100;100;100;Lietimas
Rugile;Azijiete;12;150;60;70;60;150;150;150;Uostymas
Matas;Amerikietis;14;90;70;85;70;110;110;110;Klausa
Rokas;Indas;11;150;100;120;100;120;120;120;Rega
Tada;Australas;10;150;90;120;90;130;130;130;Liepsna
```

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 skirtingos klasės.

Duomenys ".txt" faile:



Rezultatai (Konsolėje):



Rezultatai (Klases.csv):

Klases

Herojų klas	ės:
	13
	12
	14
	11
	10

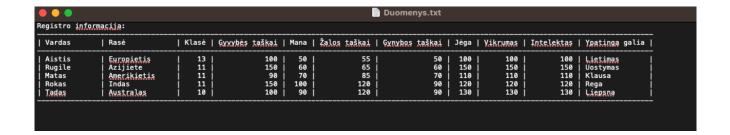
1.3.2 Duomenys ir rezultatai 2

Pradinis failas:

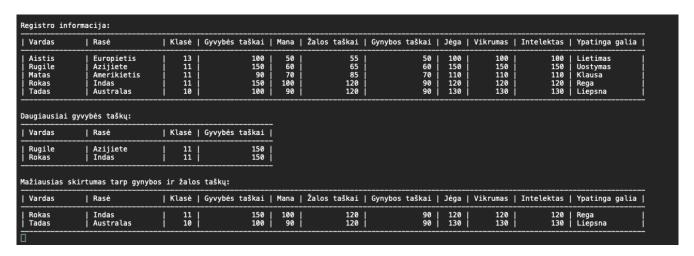
```
Aistis;Europietis;13;100;50;55;50;100;100;100;Lietimas
Rugile;Azijiete;11;150;60;65;60;150;150;150;Uostymas
Matas;Amerikietis;11;90;70;85;70;110;110;110;Klausa
Rokas;Indas;11;150;100;120;90;120;120;120;Rega
Tadas;Australas;10;100;90;120;90;130;130;130;Liepsna
```

Šiais duomenimis tikrinama ar tinkamai veikia herojų klasių 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:



Rezultatai (Konsolėje):



Rezultatai (Klases.csv):

Klases

Herojų klas	ės:
	13
	11
	10

1.4. Dėstytojo pastabos

Ateityje geriau naudoti standartinį šabloną rašant komentarus

2. Skaičiavimų klasė

- 2.2. Programos tekstas
- 2.3. Pradiniai duomenys ir rezultatai
- 2.4. Dėstytojo pastabos

3. Konteineris

- 3.2. Programos tekstas
- 3.3. Pradiniai duomenys ir rezultatai
- 3.4. Dėstytojo pastabos

4.	Teksto	analizė	ir reda	agavimas
	1011010	allalled		uga viiiiac

- 4.2. Programos tekstas
- 4.3. Pradiniai duomenys ir rezultatai
- 4.4. Dėstytojo pastabos

5. Paveldėjimas

- 5.2. Programos tekstas
- 5.3. Pradiniai duomenys ir rezultatai
- 5.4. Dėstytojo pastabos