**Капсулация и модификатори за достъп**

**App1**

Тема 7. Капсулация. Зад 1. Сортиране на хора по име и възраст

class Person

**Person.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App1 |
|  | { |
|  | /// <summary> |
|  | /// Човек |
|  | /// </summary> |
|  | class Person |
|  | { |
|  | // Име |
|  | private string firstName; |
|  | public string FirstName { get { return this.firstName; }} |
|  |  |
|  | // Фамилия |
|  | private string lastName; |
|  |  |
|  | // Възраст |
|  | private int age; |
|  | public int Age { get { return this.age; } } |
|  |  |
|  | // Конструктор |
|  | public Person(string name, string family, int age) |
|  | { |
|  | this.firstName = name; |
|  | this.lastName = family; |
|  | this.age = age; |
|  | } |
|  |  |
|  | // Пренаписахме ToString |
|  | public override string ToString() |
|  | { |
|  | return $"{this.firstName} {this.lastName} is {this.age} years old."; |
|  | } |
|  | } |
|  | } |

**Program.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App1 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | var lines = int.Parse(Console.ReadLine()); |
|  | var persons = new List<Person>(); |
|  | for (int i = 0; i < lines; i++) |
|  | { |
|  | var cmdArgs = Console.ReadLine().Split(); |
|  | var person = new Person(cmdArgs[0], cmdArgs[1], int.Parse(cmdArgs[2])); |
|  | persons.Add(person); |
|  | } |
|  |  |
|  | persons.OrderBy(p => p.FirstName) |
|  | .ThenBy(p => p.Age) |
|  | .ToList() |
|  | .ForEach(p => Console.WriteLine(p.ToString())); |
|  |  |
|  | } |
|  | } |
|  | } |

**App2**

Тема 7. Капсулация. Зад 2. Правоъгълен паралелепипед

class Box

**Box.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App2 |
|  | { |
|  | class Box |
|  | { |
|  | // дължина |
|  | private float lenght; |
|  | public float Lenght |
|  | { |
|  | get { return this.lenght; } |
|  | set { this.lenght = value; } |
|  | } |
|  |  |
|  | // широчина |
|  | private float width; |
|  | public float Width |
|  | { |
|  | get { return this.width; } |
|  | set { this.width = value; } |
|  | } |
|  |  |
|  | // височина |
|  | private float height; |
|  | public float Height |
|  | { |
|  | get { return this.height; } |
|  | set { this.height = value; } |
|  | } |
|  |  |
|  | // Конструктор |
|  | public Box(float l, float h, float w) |
|  | { |
|  | this.height = h; |
|  | this.lenght = l; |
|  | this.width = w; |
|  | } |
|  |  |
|  | // Площ |
|  | public float SurfaceArea() |
|  | { |
|  | float l = this.lenght; |
|  | float h = this.height; |
|  | float w = this.width; |
|  | return 2 \* (l \* w + l \* h + w \* h); |
|  | } |
|  |  |
|  | // Околната повърхнина |
|  | public float LateralSurfaceArea() |
|  | { |
|  | float l = this.lenght; |
|  | float h = this.height; |
|  | float w = this.width; |
|  | return 2 \* (l \* h + w \* h); |
|  | } |
|  |  |
|  | // Обем |
|  | public float Volume() |
|  | { |
|  | return this.lenght \* this.height \* this.width; |
|  | } |
|  |  |
|  | } |
|  | } |

**Program.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App2 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | float l = float.Parse(Console.ReadLine()); |
|  | float w = float.Parse(Console.ReadLine()); |
|  | float h = float.Parse(Console.ReadLine()); |
|  |  |
|  | Box Mother = new Box(l, h, w); |
|  | Console.WriteLine("Surface Area – {0:f2}", Mother.SurfaceArea()); |
|  | Console.WriteLine("Lateral Surface Area – {0:f2}", Mother.LateralSurfaceArea()); |
|  | Console.WriteLine("Volume – {0:f2}", Mother.Volume()); |
|  | } |
|  | } |
|  | } |

**App3**

Тема 8. Модификатори за достъп. Зад 1. Увеличение на заплатата

class Person

**Person.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App3 |
|  | { |
|  | /// <summary> |
|  | /// Човек |
|  | /// </summary> |
|  | class Person |
|  | { |
|  | // Име |
|  | private string firstName; |
|  | public string FirstName { get { return this.firstName; }} |
|  |  |
|  | // Фамилия |
|  | private string lastName; |
|  |  |
|  | // Възраст |
|  | private int age; |
|  | public int Age { get { return this.age; } } |
|  |  |
|  | // Заплата |
|  | private double salary; |
|  | public double Salary { get { return this.salary; } } |
|  |  |
|  | // Увеличаване на заплатата |
|  | public void IncreaseSalary(double bonus) |
|  | { |
|  | double multiplyer = 1.0 + (bonus / 100); |
|  | this.salary \*= multiplyer; |
|  | } |
|  |  |
|  | // Конструктор |
|  | public Person(string name, string family, int age, double salary) |
|  | { |
|  | this.firstName = name; |
|  | this.lastName = family; |
|  | this.age = age; |
|  | this.salary = salary; |
|  | } |
|  |  |
|  | // Пренаписахме ToString |
|  | public override string ToString() |
|  | { |
|  | return $"{this.firstName} {this.lastName} get {this.salary:f2} leva"; |
|  | } |
|  | } |
|  | } |

**Program.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App3 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | // Колекция |
|  | var persons = new List<Person>(); |
|  |  |
|  | // Вход |
|  | var lines = int.Parse(Console.ReadLine()); |
|  | for (int i = 0; i < lines; i++) |
|  | { |
|  | var cmdArgs = Console.ReadLine().Split(); |
|  | var person = new Person(cmdArgs[0], cmdArgs[1], int.Parse(cmdArgs[2]), double.Parse(cmdArgs[3])); |
|  | persons.Add(person); |
|  | } |
|  |  |
|  | // Добавяне на Бонус |
|  | var bonus = double.Parse(Console.ReadLine()); |
|  | persons.ForEach(p => p.IncreaseSalary(bonus)); |
|  |  |
|  | // Изход |
|  | persons.ForEach(p => Console.WriteLine(p.ToString())); |
|  | } |
|  | } |
|  | } |

**App4**

Тема 8. Модификатори за достъп. Зад 2. Ферма за животни

class Chicken

**Chiken.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App4 |
|  | { |
|  | class Chicken |
|  | { |
|  | public const int MinAge = 0; |
|  | public const int MaxAge = 15; |
|  |  |
|  | // Име |
|  | protected string name; |
|  | public string Name |
|  | { |
|  | get { return this.name; } |
|  | set { this.name = value; } |
|  | } |
|  |  |
|  | // Възраст |
|  | internal int age; |
|  | public int Age |
|  | { |
|  | get { return this.age; } |
|  | set { this.age = value; } |
|  | } |
|  |  |
|  | // Конструктор |
|  | public Chicken(string name, int age) |
|  | { |
|  | this.name = name; |
|  | this.age = age; |
|  | } |
|  |  |
|  | // Продукция дневно |
|  | public double ProductPerDay |
|  | { |
|  | get { return this.CalculateProductPerDay(); } |
|  | } |
|  | public double CalculateProductPerDay() |
|  | { |
|  | switch (this.Age) |
|  | { |
|  | case 0: |
|  | case 1: |
|  | case 2: |
|  | case 3: return 1.5; |
|  | case 4: |
|  | case 5: |
|  | case 6: |
|  | case 7: return 2; |
|  | case 8: |
|  | case 9: |
|  | case 10: |
|  | case 11: return 1; |
|  | default: return 0.75; |
|  | } |
|  | } |
|  | } |

**Program.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App4 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | string name = Console.ReadLine(); |
|  | int age = int.Parse(Console.ReadLine()); |
|  |  |
|  | Chicken chicken = new Chicken(name, age); |
|  | Console.WriteLine( "Chicken {0} (age {1}) can produce {2} eggs per day.", |
|  | chicken.Name, chicken.Age, chicken.ProductPerDay); |
|  | } |
|  | } |
|  | } |

**App5**

Тема 9. Валидация. Упр 3.1. Проверка на данните.

class Person

**Person.cs**

|  |
| --- |
|  |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App5 |
|  | { |
|  | class Person |
|  | { |
|  | // Име |
|  | private string firstName; |
|  | public string FirstName { get { return this.firstName; } } |
|  | // Фамилия |
|  | private string lastName; |
|  | // Възраст |
|  | private int age; |
|  | public int Age |
|  | { |
|  | get { return this.age; } |
|  | } |
|  | public double Salary |
|  | { |
|  | get { return this.Salary; } |
|  | set |
|  | { |
|  | if (value < 460) |
|  | { |
|  | throw new ArgumentException("Salary cannot be less than 460 leva"); |
|  | } |
|  | this.Salary = value; |
|  | } |
|  | } |
|  | // Конструктор |
|  | public Person(string name, string family, int age) |
|  | { |
|  | this.firstName = name; |
|  | this.lastName = family; |
|  | this.age = age; |
|  | } |
|  | public override string ToString() |
|  | { |
|  | return $"{this.firstName} {this.lastName} is {this.age} years old."; |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |

**Program.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App5 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | // TODO |
|  | } |
|  | } |
|  | } |

**App6**

Тема 9. Валидация. Упр 3.3. На пазара.

class Shopping

**Person.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App6 |
|  | { |
|  | class Person |
|  | { |
|  | private string name; |
|  | private double money; |
|  | private List<string> products; |
|  |  |
|  | public Person(string name, double money) |
|  | { |
|  | Name = name; |
|  | Money = money; |
|  | products = new List<string>(); |
|  | } |
|  | public double Money |
|  | { |
|  | get { return money; } |
|  | private set |
|  | { |
|  | if (value < 0) throw new ArgumentException("Money cannot be negative"); |
|  | money = value; |
|  | } |
|  | } |
|  | public string Name |
|  | { |
|  | get { return name; } |
|  | private set |
|  | { |
|  | if (name == string.Empty) throw new ArgumentException("Name cannot be empty"); |
|  | name = value; |
|  | } |
|  | } |
|  | public List<string> Products |
|  | { |
|  | get { return products; } |
|  | } |
|  | public void Buy(string product, List<Product> available) |
|  | { |
|  | int index = 0; |
|  | for (int i = 0; i < available.Count; i++) |
|  | if (available[i].Name == product) |
|  | index = i; |
|  |  |
|  | if (available[index].Price <= money) |
|  | { |
|  | products.Add(product); |
|  | money -= available[index].Price; |
|  | } |
|  | else Console.WriteLine($"{name} can't afford {product}"); |
|  | } |
|  | } |
|  | } |

**Product.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App6 |
|  | { |
|  | class Product |
|  | { |
|  | private string name; |
|  | private double price; |
|  | public string Name |
|  | { |
|  | get { return name; } |
|  | set { name = value; } |
|  | } |
|  | public double Price |
|  | { |
|  | get { return price; } |
|  | set { price = value; } |
|  | } |
|  | public Product(string name, double price) |
|  | { |
|  | Name = name; |
|  | Price = price; |
|  | } |
|  | } |
|  | }  **Shopping.cs** |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App6 |
|  | { |
|  | class Shopping |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | string[] inputPeople = Console.ReadLine().Split(';').ToArray(); |
|  | List<Person> people = new List<Person>(); |
|  | for (int i = 0; i < inputPeople.Length; i++) |
|  | { |
|  | string[] person = inputPeople[i].Split('=').ToArray(); |
|  | people.Add(new Person(person[0], int.Parse(person[1]))); |
|  | } |
|  |  |
|  | string[] inputProducts = Console.ReadLine().Split(';').ToArray(); |
|  | List<Product> products = new List<Product>(); |
|  | for (int i = 0; i < inputProducts.Length; i++) |
|  | { |
|  | string[] product = inputProducts[i].Split('=').ToArray(); |
|  | products.Add(new Product(product[0], double.Parse(product[1]))); |
|  | } |
|  |  |
|  | string[] command = Console.ReadLine().Split(' ').ToArray(); |
|  | while (command[0] != "END") |
|  | { |
|  | string nameOfPerson = command[0]; |
|  | string nameOfProduct = command[1]; |
|  | for (int i = 0; i < people.Count; i++) |
|  | { |
|  | if (people[i].Name == nameOfPerson) |
|  | people[i].Buy(nameOfProduct, products); |
|  | } |
|  | command = Console.ReadLine().Split(' ').ToArray(); |
|  | } |
|  |  |
|  | foreach (var person in people) |
|  | { |
|  | if (person.Products.Count == 0) |
|  | Console.WriteLine($"{person.Name} - Nothing bought"); |
|  | else |
|  | Console.WriteLine($"{person.Name} - {string.Join(", ", person.Products)}"); |
|  | } |
|  | } |
|  | } |
|  | } |

**App7**

Тема 10. Допълнителни задачи. Упр 4.1. Първи и резервен отбор

class Person

class Team

**Person.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App7 |
|  | { |
|  | class Person |
|  | { |
|  | private string firstName; |
|  | public string FirstName { get { return this.firstName; } } |
|  | private string lastName; |
|  | private int age; |
|  | public int Age { get { return this.age; } } |
|  | public Person(string name, string family, int age) |
|  | { |
|  | this.firstName = name; |
|  | this.lastName = family; |
|  | this.age = age; |
|  | } |
|  | } |
|  | } |

**Program.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App7 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | var team = new Team("Kremikovci"); |
|  | var lines = int.Parse(Console.ReadLine()); |
|  | for (int i = 0; i < lines; i++) |
|  | { |
|  | var playersInfo = Console.ReadLine().Split(); |
|  | var person = new Person(playersInfo[0], playersInfo[1], int.Parse(playersInfo[2])); |
|  | team.AddPlayer(person); |
|  | } |
|  | Console.WriteLine("First team have {0} players", team.FirstTeam.Count()); |
|  | Console.WriteLine("First team have {0} players", team.ReserveTeam.Count()); |
|  |  |
|  | } |
|  | } |
|  | } |

**Team.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App7 |
|  | { |
|  | class Team |
|  | { |
|  | private string name; |
|  | private List<Person> firstTeam; |
|  | private List<Person> reserveTeam; |
|  | public Team(string name) |
|  | { |
|  | this.name = name; |
|  | this.firstTeam = new List<Person>(); |
|  | this.reserveTeam = new List<Person>(); |
|  | } |
|  | public IReadOnlyCollection<Person> FirstTeam |
|  | { |
|  | get { return this.firstTeam.AsReadOnly(); } |
|  | } |
|  | public IReadOnlyCollection<Person> ReserveTeam |
|  | { |
|  | get { return this.reserveTeam.AsReadOnly(); } |
|  | } |
|  | public void AddPlayer(Person player) |
|  | { |
|  | if (player.Age < 40) { firstTeam.Add(player); } |
|  | else { reserveTeam.Add(player); } |
|  | } |
|  | } |
|  | } |

**App8**

Тема 10. Допълнителни задачи. Упр 4.2. Създаване на футболен отбор

class FootboolPlayers

class FootboolTeam

**Player.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App8 |
|  | { |
|  | class Player |
|  | { |
|  | // Конструктор |
|  | public Player(string name, int du, int sp, int dr, int p, int sh) |
|  | { |
|  | this.Name = name; |
|  | this.Durablility = du; |
|  | this.Sprint = sp; |
|  | this.Dribble = dr; |
|  | this.Passing = p; |
|  | this.Shooting = sh; |
|  | } |
|  |  |
|  | // Име |
|  | private string name; |
|  | public string Name |
|  | { |
|  | get { return name; } |
|  | set |
|  | { |
|  |  |
|  | if (value.Length == 0 || String.IsNullOrEmpty(value) || String.IsNullOrWhiteSpace(value)) |
|  | { |
|  | Console.WriteLine("A name should not be empty"); |
|  | } |
|  | else |
|  | { |
|  | name = value; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // Райтинг на играч |
|  | public double Rating() |
|  | { |
|  | return (durability+sprint+dribble+passing+shooting)/5.0; |
|  | } |
|  |  |
|  | // Издръжливост, Спринт, дрибъл, подавания и стрелба. |
|  | private int durability; |
|  | private int sprint; |
|  | private int dribble; |
|  | private int passing; |
|  | private int shooting; |
|  |  |
|  | // Методи |
|  | public int Durablility |
|  | { |
|  | get { return durability; } |
|  | set |
|  | { |
|  | if (value > 100 || value < 0) |
|  | { |
|  | Console.WriteLine("Durability should be between 0 and 100"); |
|  | } |
|  | else |
|  | { |
|  | durability = value; |
|  | } |
|  | } |
|  | } |
|  | public int Sprint |
|  | { |
|  | get { return sprint; } |
|  | set |
|  | { |
|  | if (value > 100 || value < 0) |
|  | { |
|  | Console.WriteLine("Sprint should be between 0 and 100"); |
|  | } |
|  | else |
|  | { |
|  | sprint = value; |
|  | } |
|  | } |
|  | } |
|  | public int Dribble |
|  | { |
|  | get { return dribble; } |
|  | set |
|  | { |
|  | if (value > 100 || value < 0) |
|  | { |
|  | Console.WriteLine("Dribble should be between 0 and 100"); |
|  | } |
|  | else |
|  | { |
|  | dribble = value; |
|  | } |
|  | } |
|  | } |
|  | public int Passing |
|  | { |
|  | get { return passing; } |
|  | set |
|  | { |
|  | if (value > 100 || value < 0) |
|  | { |
|  | Console.WriteLine("Passing should be between 0 and 100"); |
|  | } |
|  | else |
|  | { |
|  | passing = value; |
|  | } |
|  | } |
|  | } |
|  | public int Shooting |
|  | { |
|  | get { return shooting; } |
|  | set |
|  | { |
|  | if (value > 100 || value < 0) |
|  | { |
|  | Console.WriteLine("Shooting should be between 0 and 100"); |
|  | } |
|  | else |
|  | { |
|  | shooting = value; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |

**Program.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App8 |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | // Data Structure |
|  | List<Team> team = new List<Team>(); |
|  |  |
|  | // Input |
|  | string line = String.Empty; |
|  | while(true) |
|  | { |
|  | line = Console.ReadLine(); |
|  | if (line == "END") break; |
|  | var cmd = line.Split(';'); |
|  | switch (cmd[0]) |
|  | { |
|  | case "Team": |
|  | { |
|  | team.Add(new Team(cmd[1])); |
|  | break; |
|  | } |
|  | case "Add": |
|  | { |
|  | var tm = team.Where(x => x.Name == cmd[1]).First(); |
|  | tm.Players.Add ( new Player |
|  | ( |
|  | cmd[2], |
|  | int.Parse(cmd[3]), |
|  | int.Parse(cmd[4]), |
|  | int.Parse(cmd[5]), |
|  | int.Parse(cmd[6]), |
|  | int.Parse(cmd[7]) |
|  | )); |
|  | break; |
|  | } |
|  | case "Remove": |
|  | { |
|  | var tm = team.Where(x => x.Name == cmd[1]).First(); |
|  | var pl = tm.Players.Where(x => x.Name == cmd[2]).First(); |
|  | tm.Players.Remove(pl); |
|  | break; |
|  | } |
|  | case "Rating": |
|  | { |
|  | var tm = team.Where(x => x.Name == cmd[1]).Take(1).First(); |
|  | Console.WriteLine("{0} - {1}", tm.Name, tm.Rating()); |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |

**Team.cs**

|  |
| --- |
| using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace App8 |
|  | { |
|  | class Team |
|  | { |
|  | // Конструктор |
|  | public Team(string name) |
|  | { |
|  | this.Name = name; |
|  | this.Players = new List<Player>(); |
|  | } |
|  |  |
|  | // Играчи |
|  | private List<Player> players; |
|  | public List<Player> Players |
|  | { |
|  | get { return players; } |
|  | set { players = value; } |
|  | } |
|  |  |
|  | // Име |
|  | private string name; |
|  | public string Name |
|  | { |
|  | get { return name; } |
|  | set |
|  | { |
|  | if (value.Length == 0 || String.IsNullOrEmpty(value) || String.IsNullOrWhiteSpace(value)) |
|  | { |
|  | Console.WriteLine("A name should not be empty"); |
|  | } |
|  | else |
|  | { |
|  | name = value; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | // Рейтинг |
|  | public int Rating() |
|  | { |
|  | double sum = this.Players.Sum(x => x.Rating()); |
|  | int count = this.Players.Count(); |
|  | return (int)Math.Ceiling(sum / count); |
|  | } |
|  | } |
|  | } |

**App9**

Тема 10. Допълнителни задачи. Упр 5.1. Калории на Pizza

class Dough

class Topping

class Pizza