**C# Tips and Tricks**

**1.Минимални и максимални стойности на типовете данни:**

* **sbyte [-128 …127]: signed 8-bit [-27 … 27-1]**
* **byte [0 … 255]: unsigned 8-bit [0 … 28-1]**
* **short [-32 768 … 32 767]: signed 16-bit [-215 … 215-1]**
* **ushort [0 … 65 535]: unsigned 16-bit [0 … 216-1]**
* **int [-2 147 483 648 … 2 147 483 647]: signed 32-bit [-231 … 231-1]**
* **uint [0 … 4 294 967 295]: unsigned 32-bit [0 … 232-1]**
* **long [-9 223 372 036 854 775 808 … 9 223 372 036 854 775 807]: signed 64-bit [-263 … 263-1]**
* **ulong [0 … 18 446 744 073 709 551 615]: unsigned 64-bit [0 … 264-1]**
* **Floating-point types are:**
  + **float (±1.5 × 10−45 to ±3.4 × 1038)**
    - **32-bits, precision of 7 digits**
  + **double (±5.0 × 10−324 to ±1.7 × 10308)**
    - **64-bits, precision of 15-16 digits**
* **The default value of floating-point types:**
  + **Is 0.0F for the float type**
  + **Is 0.0D for the double type**
* **There is a special decimal floating-point real number type in C#:**
  + **decimal (±1,0 × 10-28 to ±7,9 × 1028)**
    - **128-bits, precision of 28-29 digits**
  + **Used for financial calculations**
  + **Almost no round-off errors**
  + **Almost no loss of precision**
* **The default value of decimal type is:**
  + **0.0M (M is the suffix for decimal numbers)**
* **BigInteger represents an arbitrarily large signed integer.**
  + **From the System.Numerics namespace.**
  + **Reference needs to be added.**
* **Needs special methods for simple actions like adding, subtracting, multiplying, powering etc.**

**2.За прочитане на няколко елемента и разделянето им по space ползвам:**

**for (int i = 0; i < N; i++)**

**{**

**var input = Console.ReadLine().Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).ToList();**

**string siteName = input.First();**

**long siteVisits = long.Parse(input.Skip(1).First());**

**decimal pricePerVisit = decimal.Parse(input.Last());**

**}**

**3.За отпечатване на елементи от списък и масив ползвам:**

**Console.WriteLine(string.Join("\n", nameOfAllSites));**

**Като сепаратора в кавичките може да е различен,в този случай всеки елемент е на нов ред!**

**4.За включване на BigInteger-Reference/add/System.Numerics след това пиша BigInteger и го включвам с Ctrl+.(или Alt+Enter).**

**5.Полезно е да ползвам и степен с BigInteger:**

**BigInteger securityToken = BigInteger.Pow(securityKey, N);**

**Като 1-вото число е числото,което ще повдигаме на степен,а 2-рото,колко пъти ще го повдигаме.Друг полезен метод за повдигане на степен е:**

**BigInteger securityToken = 1;**

**for (int i = 0; i < N; i++)**

**{**

**securityToken \*= securityKey;**

**}**

**Като тук числото може да ми бъде от всякакъв тип.Последният метод за повдигане на число на степен е с използването на Math.Pow:**

**double value4 = Math.Pow(5, 2);**

**той връща double!!!**

**6.Хитър начин за направа на while loop докато се върти и да намалява е:**

**var amount=int.Parse(Console.ReadLine());**

**while(amount-- > 0)**

**{**

**}**

**Така цикъла хем се върти,хем намалява;**

**7.Използване на TimeSpan за време:**

**long secondsAll = secondsLeaving + secondsWithOutDays;**

**TimeSpan time = TimeSpan.FromSeconds(secondsAll);**

**string totalTime = time.ToString(@"hh\:mm\:ss");**

**Console.Write("Time Arrival: ");**

**Console.WriteLine(totalTime);**

**8.Метод за намиране на всички цифри,малки и големи букви и тяхната сума в string:**

**var allLetters = new string**

**(text.Where(x=>char.IsLetter(x)&&char.IsLower(x)).ToArray());**

**int sumLetters = 0;**

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

**{**

**sumLetters += allLetters[i];**

**}**

**9.Метод за намиране на сбора на chars в string:**

**string pattern = @"(\<.+?\>)";**

**string input = Console.ReadLine();**

**MatchCollection matches = Regex.Matches(input, pattern);**

**if (matches.Count == 0)**

**{**

**Console.WriteLine("Better luck next time");**

**return;**

**}**

**foreach (Match m in matches)**

**{**

**var allDigits =new string(m.ToString().Where(x => char.IsDigit(x)).ToArray());**

**var digitsToChar = allDigits.ToCharArray();**

**var sum = digitsToChar.Sum(x => int.Parse(x.ToString()));**

**Console.WriteLine($"Found {sum} carat diamond");**

**}**

**Ако ми трябват букви или символи само замествам char.IsLetter IsSymbol,ако не ги parsn-a c# отчита кода на ASCII таблицата,което е много удобно за букви и символи.**

**10.Бърз начин за сплитване по дадени елементи:**

**var placeholders = Console.ReadLine()**

**.Split("{}".ToCharArray(), StringSplitOptions.RemoveEmptyEntries).ToArray();**

**Мога да изброя всички елементи в двойните кавички и да ги сплитна(по-лесно е отколкото с char[])**

**11.Бърз начин за мачване и replace на текст е:**

**string pattern = @"([A-Za-z]+)(.+)\1";**

**string text = Console.ReadLine();**

**var placeholders = Console.ReadLine()**

**.Split("{}".ToCharArray(), StringSplitOptions.RemoveEmptyEntries).ToArray();**

**MatchCollection matches = Regex.Matches(text, pattern);**

**int i = 0;**

**foreach (Match item in matches)**

**{**

**string newValue = item.Groups[1] + placeholders[i++] + item.Groups[1];**

**text = text.Replace(item.ToString(), newValue);**

**}**

**Console.WriteLine(text);**

**12.Бърз начин за принтиране на елементи от лист или масив,всеки на нов ред е:**

**Console.WriteLine("Broadcasts:");**

**if (listOfBroadcasts.Count==0)**

**{**

**Console.WriteLine("None");**

**}**

**else**

**{**

**listOfBroadcasts.ForEach(x => Console.WriteLine(x));**

**}**

**13.Метод за Reverse na string:**

**string recepCode = matchMessage.Groups[1].Value;**

**recepCode = string.Join("", recepCode.ToCharArray().Reverse().ToArray());**

**14.Бърз начин за добавяне на елементи в дадено Dictionary е:**

**Dictionary<char, int> chars = new Dictionary<char, int>();**

**foreach (char c in currentWord)**

**{**

**if (chars.ContainsKey(c)) chars[c]++;**

**else chars.Add(c, 1);**

**}**

**15.Начин за разделяне и парсване на цифри от конзолата,които не са разделени:**

**var inputKey = Console.ReadLine().ToCharArray()**

**.Select(x => int.Parse(x.ToString())).ToList();**

**16.Ако имам много неща в едно Dictionary,а накрая трябва да изпринтирам малка част от тях,най-добре е да си направя ново Dictionary за резултата и да елиминирам елементите,които не ми трябват, иначе мога да сбъркам с LINQ(Hornet Armada).**

**17.Метода Substring:**

**int startIndexFirst = firtTeamName.IndexOf(key);**

**int lastIndexFirst = firtTeamName.LastIndexOf(key);**

**firtTeamName = firtTeamName**

**.Substring(startIndexFirst + key.Length, lastIndexFirst - startIndexFirst - key.Length);**

**firtTeamName = string.Join("", firtTeamName.ToUpper().ToCharArray().Reverse().ToArray());**

**Ползва се когато имам даден string между други 2 stringa и искам да го извадя.Вземат се стартовия индекс и последният индекс на stringowete.Зад Football League(2016/06/12)**

**18.Метод за reverse на елементи от лист по даден старт и брой:**

**var listOfElements = Console.ReadLine()**

**.Split(" ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries)**

**.ToList();**

**int elementsLength = listOfElements.Count();**

**int lastIndexOfElements = elementsLength - 1;**

**string input = Console.ReadLine();**

**if (input == "end")**

**break;**

**string[] command = input.Split().ToArray();**

**string operation = command.First();**

**if (operation == "reverse")**

**{**

**int startIndex = int.Parse(command.Skip(2).First());**

**int count = int.Parse(command.Last());**

**long lastIndexReversed = startIndex + count;**

**if (startIndex < 0 || startIndex > lastIndexOfElements**

**|| count < 0 || count > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**if (lastIndexReversed > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**listOfElements.Reverse(startIndex, count);**

**}**

**19.Метод за sort на елементи от лист по даден старт и брой:**

**else if (operation == "sort")**

**{**

**int startIndex = int.Parse(command.Skip(2).First());**

**int count = int.Parse(command.Last());**

**long lastIndexSort = startIndex + count;**

**if (startIndex < 0 || startIndex > lastIndexOfElements**

**|| count < 0 || count > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**if (lastIndexSort > elementsLength)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**listOfElements.Sort(startIndex, count, null);**

**}**

**20.Методи за RollLeft and RollRight:**

**else if (operation == "rollLeft")**

**{**

**int count = int.Parse(command.Skip(1).First());**

**int timesRow = count % elementsLength;**

**if (count<0)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**for (int i = 0; i < timesRow; i++)**

**{**

**listOfElements.Add(listOfElements.ElementAt(0));**

**listOfElements.RemoveAt(0);**

**}**

**}**

**else if(operation=="rollRight")**

**{**

**int count = int.Parse(command.Skip(1).First());**

**int timesRow = count % elementsLength;**

**if (count < 0)**

**{**

**Console.WriteLine("Invalid input parameters.");**

**continue;**

**}**

**listOfElements.Reverse();**

**for (int i = 0; i < timesRow; i++)**

**{**

**listOfElements.Add(listOfElements.ElementAt(0));**

**listOfElements.RemoveAt(0);**

**}**

**listOfElements.Reverse();**

**}**

**21.Метод за намиране на дните в даден месец и година:**

decimal pricePercapsule = decimal.Parse(Console.ReadLine());

var orderDate = Console.ReadLine().Split('/').ToArray();

int capsulesCount = int.Parse(Console.ReadLine());

var day = orderDate.First();

var month = int.Parse(orderDate.Skip(1).First());

var year = int.Parse(orderDate.Last());

decimal countDays =DateTime.DaysInMonth(year, month);

priceCoffee = countDays \* pricePercapsule \* capsulesCount;

**Датата се чете като string и се ползва опцията DateTime.DaysInMonth**

**22.Метод за позлване на TimeSpan и дата към string:**

var leavingTime = Console.ReadLine().Split(new char[] { ':' }).ToArray();

long stepsToHome = long.Parse(Console.ReadLine());

long secondsForStep = long.Parse(Console.ReadLine());

long allTimeInSeconds = stepsToHome \* secondsForStep;

int hours = int.Parse(leavingTime[0]);

int minutes = int.Parse(leavingTime[1]);

int seconds = int.Parse(leavingTime[2]);

int secondsLeaving = seconds + (60 \* minutes) + (3600 \* hours);

long secondsWithOutDays = allTimeInSeconds % (24\*3600);

long secondsAll = secondsLeaving + secondsWithOutDays;

TimeSpan time = TimeSpan.FromSeconds(secondsAll);

string totalTime = time.ToString(@"hh\:mm\:ss");

Console.Write("Time Arrival: ");

Console.WriteLine(totalTime);

**23.Метод за игнориране на празен string (ползва се Where(a=>a!=””)):**

var stackNumbers=new Stack<int>(input.Split().Where(a=>a!="").Select(int.Parse));

**24.Стекове и опашки-определения,функции и начини на употреба:**

**1.**

**Stack<T> - структура от данни която пази елементи.Ако искаме да извадим елемент от структурата,то изваденият ще бъде последният вкаран елемент.**

**Пример: 1,2,3,34,56565,5 5 ще бъде изваденият елемент**

**Stack<T>.Push – добавяме елемент към стека(добавя се на последно място)**

**Stack<T>.Pop – изтриваме елемент от стека(последният добавен)**

**Stack<T>.Peek – показва последният добавен елемент в стека,но не го маха!**

**Stack<T>.Count – брой на елементите в стека**

**Stack<T>.ToArray – превръща стека в масив като реда на вкараните елементи се запазва**

**Stack<T>.Contains(n) – true or false взависимост дали стека съдържа n**

**Stack<T>.Clear – изтрива всички елементи от стека**

**Stack<T>.TrimExcess – прави стека в дължина,колкото са елементите(маха всички празни елементи)**

**Хитър начин за вадене на елементи от стек,докато не остане нито един е:**

**while (stack.Count != 0)**

**{**

**Console.Write(stack.Pop());**

**}**

**2.**

**Queue<T> - структура от данни която пази елементи.Ако искаме да извадим елемент от структурата,то изваденият ще бъде първият вкаран елемент.**

**Пример: 1,2,3,34,56565,5 1 ще бъде изваденият елемент**

**Queue<T>.Enqueue – добавяме елемент към опашката**

**Queue <T>.Dequeue – изтриваме елемент от опашката(първият добавен)**

**Queue <T>.Peek – показва последният добавен елемент в опашката,но не го маха!**

**Queue <T>.Count – брой на елементите в опашката**

**Queue <T>.ToArray – превръща опашката в масив като реда на вкараните елементи се запазва**

**Queue <T>.Contains(n) – true or false взависимост дали опашката съдържа n**

**Queue <T>.Clear – изтрива всички елементи от опашката**

**Queue <T>.TrimExcess – прави опашката с дължина,колкото са елементите(маха всички празни елементи)**

**25.Стартиране и спиране на таймер за скорост:**

var watch = Stopwatch.StartNew();

.

.функция,която се изпълнява

.

.

watch.Stop();

Console.WriteLine(watch.ElapsedTicks); или Console.WriteLine(watch.ElapsedMilliseconds);

**26.Сума на всички елементи в 2D Array**

**Метод 1:**

int sum = 0;

for (int rows = 0; rows < matrix.Length; rows++)

{

string[] input = Console.ReadLine()

.Split(", ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

matrix[rows] = new int[input.Length];

for (int cols = 0; cols < matrix[rows].Length; cols++)

{

matrix[rows][cols] = int.Parse(input[cols]);

sum += matrix[rows][cols];

}

}

**Метод 2:**

for (int rows = 0; rows < matrix.Length; rows++)

{

string[] input = Console.ReadLine()

.Split(", ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

matrix[rows] = new int[input.Length];

for (int cols = 0; cols < matrix[rows].Length; cols++)

{

matrix[rows][cols] = int.Parse(input[cols]);

}

}

var sumTwo = matrix.Sum(i => i.Sum());

**27.Когато търся остатък от деление(reminder 10 % 2==0),винаги трябва да превръщам делимото(10),в положително число,иначе функцията дава грешен резултат:**

int[] zeroReminder = sequance.Where(x => Math.Abs(x) % 3 == 0).ToArray();

int[] oneReminder = sequance.Where(x => Math.Abs(x) % 3 == 1).ToArray();

int[] twoReminder = sequance.Where(x => Math.Abs(x) % 3 == 2).ToArray();

**28.Методи за намиране сумата на диагоналите в квадратна матрица:**

**Метод 1:**

int sumFirst = 0;

int sumLast = 0;

int indexFirst = 0;

int indexLast = sizeMatrix - 1;

foreach (var row in matrix)

{

sumFirst += row.ElementAt(indexFirst);

sumLast += row.ElementAt(indexLast);

indexFirst++;

indexLast--;

}

int absoluteDeviation = Math.Abs(sumFirst - sumLast);

**Метод 2:**

long sumPrimaryDiagonal = 0;

long sumSecondaryDiagonal = 0;

for (int row = 0; row < matrix.Length; row++)

{

sumPrimaryDiagonal += matrix[row][row];

sumSecondaryDiagonal += matrix[row][matrix.Length - 1 - row];

}

return Math.Abs(sumPrimaryDiagonal - sumSecondaryDiagonal);

**29.Бърз метод за четене на елементите не дадена матрица:**

**За String[][]:**

string[][] matrix = new string[rowsCount][];

for (int row = 0; row < rowsCount; row++)

{

matrix[row] = Console.ReadLine()

.Split(" ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

}

**За Char[][]:**

char[][] matrix = new Char[rowsCount][];

for (int row = 0; row < rowsCount; row++)

{

matrix[row] = Console.ReadLine()

.Split(" ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries)

.Select(x => x[0])

.ToArray();

}

**За Int[][]:**

int[][] matrix = new int[rowsCount][];

for (int row = 0; row < rowsCount; row++)

{

matrix[row] = Console.ReadLine()

.Split(" ".ToCharArray(), StringSplitOptions.RemoveEmptyEntries)

.Select(int.Parse)

.ToArray();

}

**30.Метод за максимална сума на матрица 3х3 в друга по-голяма матрица:**

long maxSum = long.MinValue;

long[][] searchMatrix = new long[3][];

long[][] currentMatrix = new long[3][];

for (int i = 0; i < rowsCount-2; i++)

{

for (int j = 0; j < colsCount-2; j++)

{

for (int row = 0; row < 3; row++)

{

currentMatrix[row] = matrix[row+i].Skip(j).Take(3).ToArray();

}

long currentSum = currentMatrix.Sum(x => x.Sum());

if (currentSum>maxSum)

{

searchMatrix[0] = currentMatrix[0];

searchMatrix[1] = currentMatrix[1];

searchMatrix[2] = currentMatrix[2];

maxSum = currentMatrix.Sum(x => x.Sum());

}

}

}

**31.Използване на функцията Predicate:**

[**https://stackoverflow.com/questions/1710301/what-is-a-predicate-in-c**](https://stackoverflow.com/questions/1710301/what-is-a-predicate-in-c)

**32.Метод за заместване на всички whitespaces с единичен:**

string encryptedText = builder.ToString();

var list = encryptedText.Split(' ').Where(s => !string.IsNullOrWhiteSpace(s));

encryptedText = string.Join(" ", list);

**33.Parse по String[] :**

string[] regionData = input

.Split(new[] {" -> "}, StringSplitOptions.RemoveEmptyEntries)

.ToArray();

**34.Добавяне на допълнителни елементи към дадено Dictionary и отпечатване на елементи чрез List:**

var regions = regionsAndArmy

.Select(m => new

{

RegionName = m.Key,

BlackMeteors = m.Value["Black"]

})

.OrderByDescending(r => r.BlackMeteors)

.ThenBy(r => r.RegionName.Length);

foreach (var r in regions)

{

Console.WriteLine(r.RegionName);

regionsAndArmy[r.RegionName]

.OrderByDescending(x => x.Value)

.ThenBy(x=>x.Key)

.ToList()

.ForEach(c => Console.WriteLine($"-> {c.Key} : {c.Value}"));

}

**35.Regex който хваща определена дума,и след него точна дължина,без други букви:**

string patternMatchName = $@"{patternForNames}([a-zA-Z]{{{lengthName}}})(?![a-zA-Z])";

**36.Хитър начин за запълване на матрица с числата от 0 до последният елемент на матрицата:**

for (int row = 0; row < rows; row++)

{

matrix[row] = new int[cols];

for (int col = 0; col < cols; col++)

{

matrix[row][col] = row \* cols + col;

}

}

**37.Метод за проверка дали даден елемент е вътре в матрицата:**

private static bool IsInMatrix(int row, int col, int rows, int cols)

{

return row >= 0 && row < rows && col >= 0 && col < cols;

}

**38.Принтиране редовете на матрица:**

foreach (var row in matrix)

{

Console.WriteLine(string.Join(" ",row));

}

**39.Метод за намиране на дните между 2 дати (дати се подават като стринг : 2015 03 25):**

using System;

public class DateModifier

{

public static int GetDaysDifference(string firstDate, string secondDate)

{

TimeSpan difference = DateTime.Parse(firstDate) - DateTime.Parse(secondDate);

return Math.Abs(difference.Days);

}

}

using System;

using System.Globalization;

public class StartUp

{

public static void Main()

{

string firstDate = Console.ReadLine();

string secondDate = Console.ReadLine();

int daysValue=DateModifier.GetDaysDifference(firstDate, secondDate);

Console.WriteLine(daysValue);

}

}

**40.Мога да си създавам нови променливи с командата .Select:**

allEmployes = GetListOfAllEmployees(numberOfLines);

var searchedDept = allEmployes

.GroupBy(e => e.Department)

.Select(p => new

{

Department = p.Key,

AverageSalary = p.Average(sal => sal.Salary),

Employees = p.OrderByDescending(e => e.Salary).ToList()

})

.OrderByDescending(x => x.AverageSalary)

.FirstOrDefault();

**41.Когато създавам нов клас,1-во трябва да си направя всички полета,след това конструкторите,които ще използвам и най-накрая да направя съответните propertity-a:**

public class Employee

{

private string name;

private decimal salary;

private string position;

private string department;

private string email;

private int age;

public Employee(string name, decimal salary, string position, string department)

{

this.name = name;

this.salary = salary;

this.position = position;

this.department = department;

this.email = "n/a";

this.age = -1;

}

public string Name

{

get => this.name;

set => this.name = value;

}

public string Department

{

get => this.department;

set => this.department = value;

}

public decimal Salary

{

get => this.salary;

set => this.salary = value;

}

public string Email

{

get => this.email;

set => this.email = value;

}

public int Age

{

get => this.age;

set => this.age = value;

}

public override string ToString()

{

return $"{this.name} {this.salary:f2} {this.email} {this.age}";

}

}

**42.Принцип на работа на IEnumerator() –извиква за да се обходи дадена колекция по зададен от нас критерии(колекцията трябва да е Generic):**

List<string> collection = new List<string>()

{

"Pesho",

"Gosho",

"Ivan",

"Stamat"

};

IEnumerator<string> enumerator = collection.GetEnumerator();

enumerator.Reset();

while (enumerator.MoveNext())

{

Console.WriteLine(enumerator.Current);

}

**43.За да можем да итерираме дадена колекция през foreach loop,тя трябва да е IEnumerable от някакъв тип(в случаят от студенти) или цялата колекция да е IEnumerable:**

public class University : IEnumerable<Student>

{

private readonly List<Student> students;

public University()

{

this.students=new List<Student>();

}

public void AddStudent(Student student)

{

this.students.Add(student);

}

public IEnumerator<Student> GetEnumerator()

{

return this.students.GetEnumerator();

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

}

**44.Ако искаме да напишем наша логики за итериране на елементи(примерно през 1) ползваме вмъкнат клас в основния.Той съдържа 5 метода,но 3 са основни:Reset,Current MoveNext:**

public class University : IEnumerable<Student>

{

private readonly List<Student> students;

public University()

{

this.students=new List<Student>();

}

public void AddStudent(Student student)

{

this.students.Add(student);

}

public IEnumerator<Student> GetEnumerator()

{

return new UniIterator();

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

private class UniIterator:IEnumerator<Student>

{

public bool MoveNext()

{

throw new System.**NotImplementedException**();

}

public void Reset()

{

throw new System.**NotImplementedException**();

}

public Student Current { get; }

object IEnumerator.Current

{

get { return Current; }

}

public void Dispose()

{

throw new System.**NotImplementedException**();

}

}

}

**Чрез Reset() сетваме нулевият елемент(индекс):**

public void Reset()

{

this.currentIndex = 0;

}

**Чрез MoveNext() се местим от елемент на елемент(ако има такъв):**

public bool MoveNext()

{

return (currentIndex += 2) < this.students.Count;

}

**Чрез Current() връщаме текущият елемент в колекцията:**

public Student Current => this.students[currentIndex];

така изглежда готовият помощен клас:

private class UniIterator:IEnumerator<Student>

{

private int currentIndex;

private readonly IList<Student> students;

public UniIterator(IList<Student> students)

{

this.Reset();

this.students = students;

}

public bool MoveNext()

{

return (currentIndex += 2) < this.students.Count;

}

public void Reset()

{

this.currentIndex = -2;

}

public Student Current => this.students[currentIndex];

object IEnumerator.Current

{

get { return Current; }

}

public void Dispose()

{

}

}

**45.За да не правя нов допълнителен клас,ползвам ключовата дума yield(върши същата работа каккто горният клас):**

public IEnumerator<Student> GetEnumerator()

{

// return new UniIterator(this.students);

for (int i = 0; i < this.students.Count; i+=2)

{

yield return this.students[i];

}

}

**46.Ключовата дума params – чрез нея можем да добавяме колкото си искаме аргументи от някакъв тип,само най-важното е тя да се подава последна в метода:**

var test = new string[] {"gosho", "dani", "ani", "ivan"};

PrintData("prefix",test);

}

public static void PrintData(string prefix, params string[] students)

{

foreach (var student in students)

{

Console.WriteLine($"{prefix} ---{student}");

}

}

**47.Използване на IComparable – връща -1,ако ел1 < ел 2; 0 ако ел1=ел2 и 1 ако ел1>ел2(фабрично):**

public class Student : IComparable<Student>

{

public Student(string name, int age, string facultyNumber)

{

this.Name = name;

this.Age = age;

this.FacultyNumber = facultyNumber;

}

public string Name { get; set; }

public int Age { get; set; }

public string FacultyNumber { get; set; }

public int CompareTo(Student otherStudent)

{

if (this.Age!=otherStudent.Age)

{

return this.Age - otherStudent.Age;

}

if (this.Name!=otherStudent.Name)

{

return this.Name.CompareTo(otherStudent.Name);

}

return 0;

}

}static void Main()

{

Student st1 = new Student("pesho", 24, "ab231");

Student st2 = new Student("gosho", 24, "assss231");

var sortStud = new SortedSet<Student>();

sortStud.Add(st1);

sortStud.Add(st2);

Console.WriteLine(st1.CompareTo(st2));

foreach (var st in sortStud)

{

Console.WriteLine(st.Name);

}

}

**48. Използване на IComparer – сравнението вече не фабрично,а ние задаваме критерият,като го подаваме в колелцията с new!**

static void Main()

{

Student st1 = new Student("pesho", 24, "ab231");

Student st2 = new Student("gosho", 24, "assss231");

var sortStud = new SortedSet<Student>(new StudentComparator());

sortStud.Add(st1);

sortStud.Add(st2);

foreach (var st in sortStud)

{

Console.WriteLine(st.Name);

}

}

public class StudentComparator : IComparer<Student>

{

public int Compare(Student x, Student y)

{

if (x.Age!=y.Age)

{

return x.Age - y.Age;

}

if (x.Name!=y.Name)

{

return x.Name.CompareTo(y.Name);

}

return 0;

}

}

**49.Основни принципи на Reflection-чрез нея може да изследваме даден клас и неговите характеристики!**

1 var type = typeof(TestReflection); - основна команда за класа

2 Console.WriteLine(type.FullName); - дава пълното име на класа

3 var interfaces = type.GetInterfaces(); - връща всички интерфейси,които съдържа класа

foreach (var inter in interfaces)

{

Console.WriteLine(inter.FullName);

}

4 var baseType = type.BaseType; - връща класът който се наследява(той е само 1!)

Console.WriteLine(baseType.FullName);

5 var constructors=type.GetConstructors(); - връща всички конструкторите,които съдържа класа(няма значение дали те са public или private)

**6**  var test = Activator.CreateInstance(typeof(TestReflection)); - още 1 начин за създаване на обект от инстанцията на класа

7 var test = Activator.CreateInstance(typeof(TestReflection),true); - ако конструкторите са private така може да ги достъпим пак!

8 var fields = type.GetFields(BindingFlags.Static | BindingFlags.NonPublic |

BindingFlags.Public | BindingFlags.Instance);

връща броят на полетата,техният тип и име(ако имаме и property,което няма поле ще ни върне и автоматично създаденото поле за property-то)

9 Метод за ръчно достъпване на някой от конструкторите на StringBUilder:

var typeBuilder = typeof(StringBuilder);

var constructor = typeBuilder.GetConstructor(new Type[] {typeof(string), typeof(int)});

var sb = constructor.Invoke(new object[] {"georgi", 100});

**50.UnitTests – след ката съм си създал класа и съм добавил неговите полета,пропъртита и методи трябва да направя нов проект Library(.NetCore) , който ще носи името на класа който ще тествам + Tests(BankAccountTests).След това то NugetManager трябва да инсталирам NUnit,NUnit3TestAdapter и** Microsoft.NET.Test.Sdk. **След това давам rebuild,пиша си теста и го тествам-ако в TestExplorer e зелен значи теста е минал,ако е червено – изписва каква е грешката:**

public class BankAccount

{

public int Balance { get; private set; }

public void Deposit(int amount)

{

this.Balance += amount;

}

public void Withdraw(int amount)

{

if (Balance < amount)

{

throw new Exception("Not enougth money");

}

this.Balance -= amount;

}

}

public class BankAccountTests

{

[Test]

public void BankAccountShouldIncreaseBalance()

{

var bankAccount=new BankAccount();

bankAccount.Deposit(10);

Assert.That(bankAccount.Balance,Is.EqualTo(9)); - така теста е червен-очаква се 9,а то е 10

Assert.AreEqual(9,bankAccount.Balance); - може да се запише и така

}

}

**51.Начин за проверка на вида Execption който се получава:**

[Test]

public void WithdrawTestException()

{

var bankaccount = new BankAccount();

Assert.Throws<Exception>(() => bankaccount.Withdraw(20));

}

**52.Начин за проверка на няколко стойности за 1 тест:**

[TestCase(10)]

[TestCase(100)]

[TestCase(-10)]

public void WithdrawTestException(int amount)

{

var bankaccount = new BankAccount();

Assert.Throws<ArgumentException>(() => bankaccount.Withdraw(amount));

}

**53.Начин за виждане на точният exception и грешката след него:**

[Test]

public void BrokenAxe()

{

var axe = new Axe(5, 0);

var dummy = new Dummy(10, 10);

//axe.Attack(dummy);

Assert.That(()=>axe.Attack(dummy),Throws.ArgumentException.With.Message.EqualTo("Axe is broken."));

}