**1 Классы. Библиотеки FCL. Классы как типы и объекты этих типов**

Задание 1. Создать класс А с целочисленными полями а и b и двумя методами согласно варианту. Внутри класса реализовать конструктор для инициализации a и b. Создать объект класса и продемонстрировать работу со

всеми элементами класса.

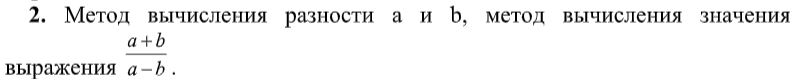


Рисунок 1.1 – Уравнение функции

Листинг программы:

using System;

using static System.Console;

namespace zad2

{

class Program

{

public static void Main()

{

Console.Write("Введите число а: ");

int num1 = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите число b: ");

int num2 = Convert.ToInt32(Console.ReadLine());

A num = new A(num1, num2);

num.raznost();

num.func();

Console.ReadLine();

}

}

class A

{

public int a;

public int b;

public A(int a, int b)

{

this.a = a;

this.b = b;

}

public void raznost()

{

Console.WriteLine($"Разность чисел {a}-{b} равно {a - b}");

}

public void func()

{

Console.WriteLine($"Результат вычисления заданной функции равен {(a+b) / (a-b)}");

}

}

}

Таблица 1.1 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5,3 | 2, 4 |

Анализ результатов:

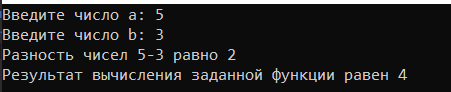


Рисунок 1.2 – Результат работы программы

Источник: собственная разработка

Задание 2. Построить иерархию классов в соответствии с вариантом задания:

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace StructApp

{

class Employee : Worker

{

public void make()

{

Console.WriteLine("Рабочий ждёт когда ему дадут работу.");

}

}

class Engineer

{

public string place;

public void make()

{

Console.Write("Инженер работает на " + place);

}

}

class Worker

{

public string instrument;

public void make()

{

Console.WriteLine(", его инструмент это " + instrument + ".");

}

}

class Personal : Engineer

{

public void make()

{

Console.WriteLine("Персонал работает в " + place + ".");

}

}

class Program

{

static void Main(string[] args)

{

Employee Employees = new Employee();

Employees.make();

Engineer Engineers= new Engineer();

Engineers.place = "заводе";

Engineers.make();

Worker workers= new Worker();

workers.instrument = "молоток";

workers.make();

Personal Personals = new Personal();

Personals.place = "офисе";

Personals.make();

Console.ReadKey();

}

}

}

Таблица 1.2 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Рабочий ждёт когда ему дадут работу.  Инженер работает на заводе, его инструмент это молоток.  Персонал работает в офисе. |

Анализ результатов:



Рисунок 1.3 – Результат работы программы

Источник: собственная разработка

Задание 3. Описать класс, реализующий шестнадцатеричный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, — и свойство, позволяющее получить его текущее состояние.

Листинг программы:

internal class Program

{

private static void Main()

{

Console.WriteLine("Введите минимум для счетчика");

var min = int.Parse(Console.ReadLine() ?? throw new InvalidOperationException());

Console.WriteLine("Введите максимум для счетчика");

var max = int.Parse(Console.ReadLine() ?? throw new InvalidOperationException());

Console.WriteLine("Введите значение для счетчика");

var v = int.Parse(Console.ReadLine() ?? throw new InvalidOperationException());

var counter = new Counter(max, min, v);

Console.WriteLine("Введите + для увеличение и - для уменьшения, чтобы выйти нажмите любую клавишу!");

do

{

var c = Console.ReadKey();

if (c.KeyChar == '+') counter.Increase();

else if (c.KeyChar == '-') counter.Decrease();

else break;

Console.WriteLine($" => {counter.Value}");

} while (true);

Console.ReadKey();

}

}

public class Counter

{

public readonly int Maximum;

public readonly int Minimum;

public int Value { private set; get; }

public Counter(int maximum, int minimum, int counter)

{

this.Maximum = maximum;

this.Minimum = minimum;

counter = Math.Min(this.Maximum, counter);

counter = Math.Max(this.Minimum, counter);

this.Value = counter;

}

private Counter()

{

this.Maximum = 10;

this.Minimum = 0;

this.Value = 5;

}

public void Increase()

{

var value = this.Value + 1;

if (value > this.Maximum || value < this.Minimum)

return;

this.Value++;

}

public void Decrease()

{

var value = this.Value - 1;

if (value > this.Maximum || value < this.Minimum)

return;

this.Value--;

}

}

Таблица 1.3 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1, 10, 5 | 6 |

Анализ результатов:

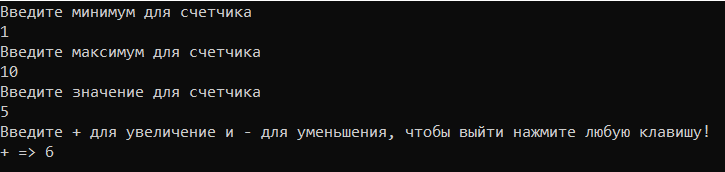


Рисунок 1.4 – Результат работы программы