Тимофеева Наталья

М8О-202Б-19

ООП C#

Лабораторная работа № 8

**Описание работы**

КОНКРЕТИЗАЦИЯ

1. Изучите категорию конкретизация. Какие разновидности этой категории есть. Вершины графа, соединенные горизонтальной связью, определите на графе в виде категории использование, для этого поставьте соответствующие стрелки. Для каждого вида категории составьте свой граф.

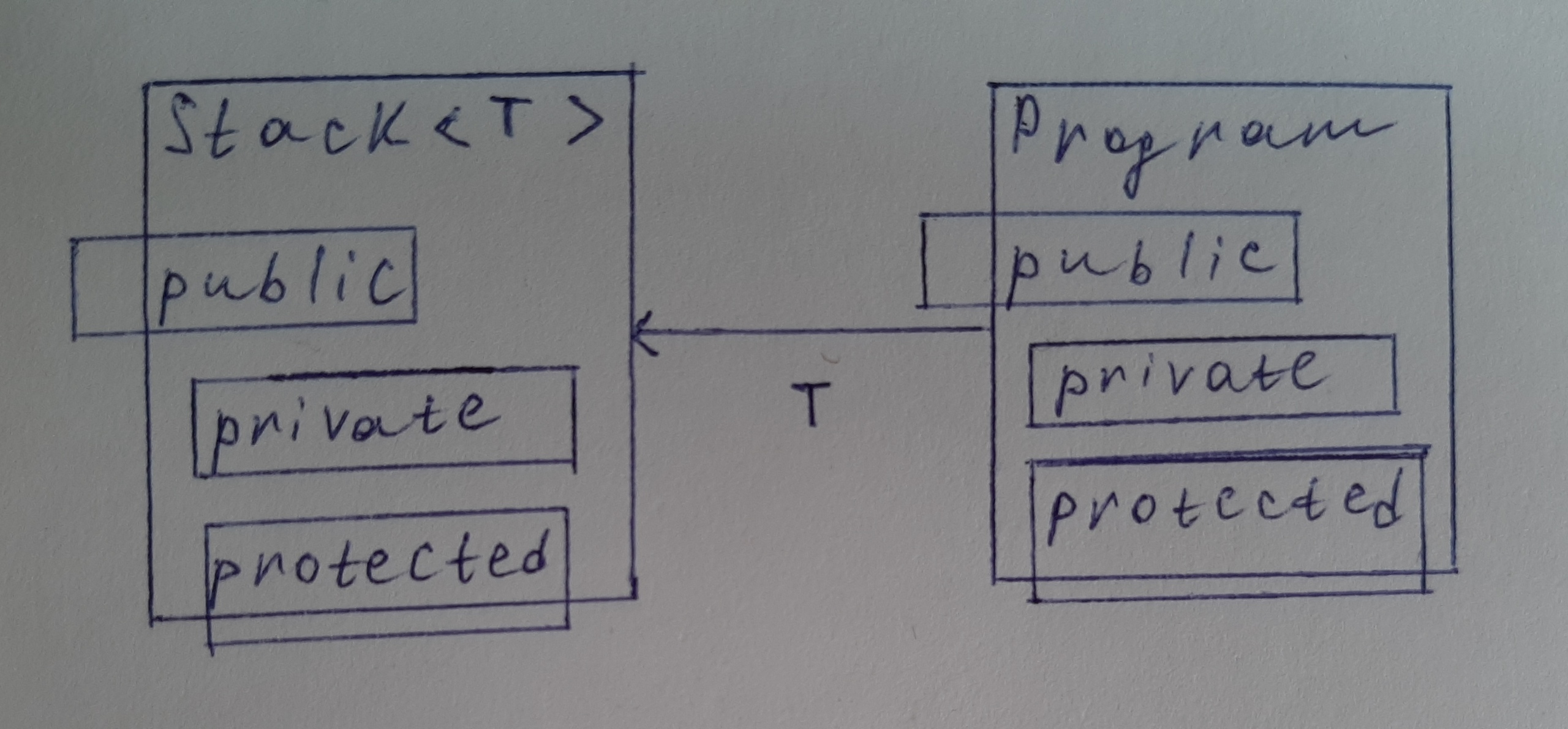
2. Реализуйте категорию конкретизация функции в отдельном проекте. Функция позволяет перемещаться по графу A-B-C, представленному категорией агрегация по ссылке и наследованием. Реализуемая функция должна двигаться по любому отрезку иерархии и реализуется в цикле.

3. Реализуйте категорию конкретизация класса в отдельном проекте.

4. Сравните полученные варианты. Напишите какими свойствами обладает категория конкретизация? В каком случае ее целесообразно использовать?

5. Оформите работу. Проектам должны соответствовать графы. Напишите ответы на вопросы. Сохраните результаты лабораторной.

**Граф**



**Программа**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Lab8 {

class Stack<T> {

List<T> obj = new List<T>();

public void Push(T a) {

if (a == null) {

throw new ArgumentNullException(nameof(a));

}

obj.Add(a);

}

public int count => obj.Count;

public T Pop() {

T a = obj.LastOrDefault();

if (a == null) {

throw new NullReferenceException("Stack is empty");

}

obj.RemoveAt(count - 1);

return a;

}

public T Pop(int n) {

T a = obj[n];

obj.RemoveAt(n);

return a;

}

}

class Program {

static void Main() {

Stack<int> st1 = new Stack<int>();

st1.Push(202);

st1.Push(67);

st1.Push(-80);

st1.Push(208);

Console.WriteLine("There is {0} int objects", st1.count);

int p1 = st1.Pop(2);

Console.WriteLine("Element {0} was removed", p1);

p1 = st1.Pop(0);

Console.WriteLine("Element {0} was removed", p1);

Console.WriteLine("There is {0} int objects", st1.count);

Stack<double> st2 = new Stack<double>();

st2.Push(46.8);

st2.Push(-3.1);

st2.Push(-6.6);

st2.Push(381.86);

st2.Push(0.001);

Console.WriteLine("There is {0} double objects", st2.count);

double p2;

int c = st2.count;

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

p2 = st2.Pop(2);

Console.WriteLine("Element {0} was removed", p2);

}

Console.WriteLine("There is {0} double objects", st2.count);

Stack<string> st3 = new Stack<string>();

st3.Push("Timofeeva");

st3.Push("Merculov");

st3.Push("Pavlovich");

Console.WriteLine("There is {0} string objects", st3.count);

string p3;

c = st3.count;

for (int i = 0; i < c; ++i) {

p3 = st3.Pop();

Console.WriteLine("Element {0} was removed", p3);

}

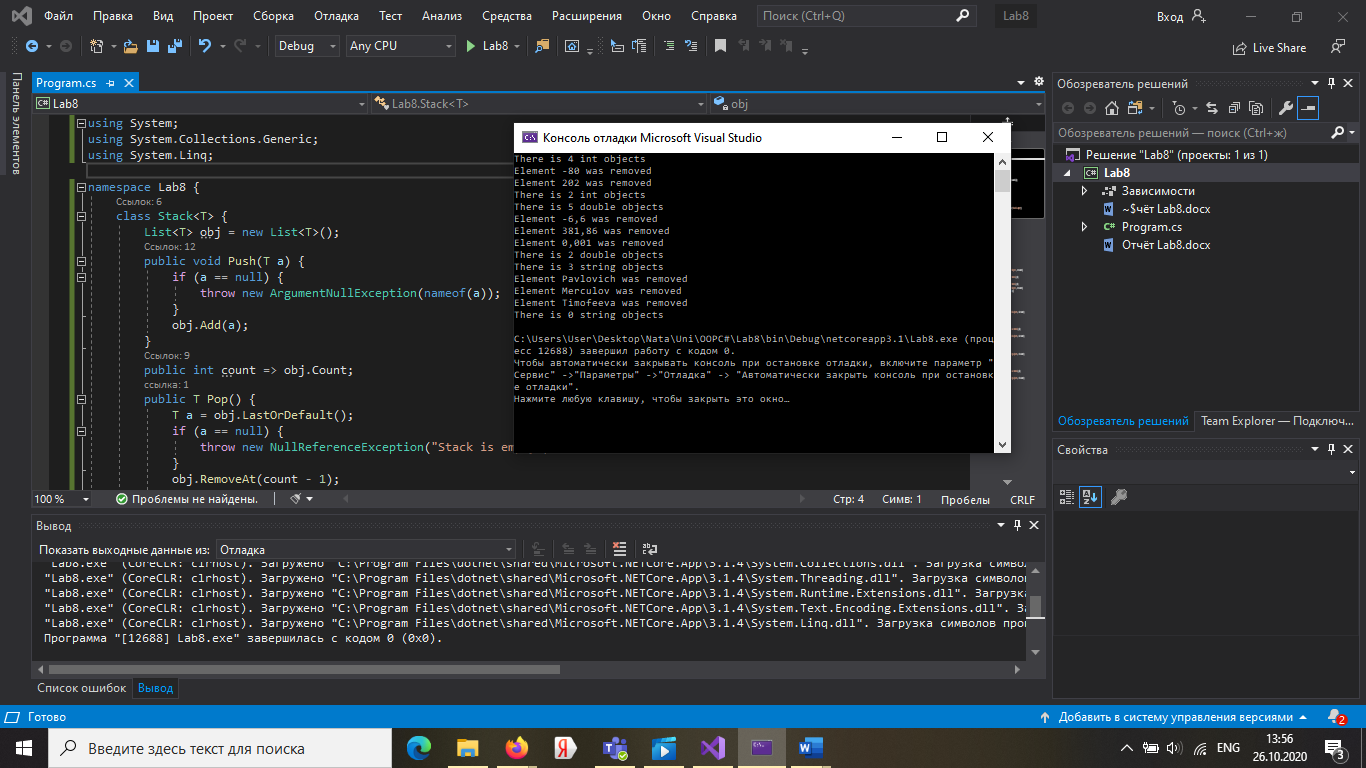
Console.WriteLine("There is {0} string objects", st3.count);

}

}

}

**Вывод с консоли**



**Вывод**

Конкретизация позволяет описывать классы и функции для работы с объектами, типы которых заранее не определены. С каким(и) именно типам(и) придется работать определяется в момент вызова – имя типа настраивает функцию\класс для работы.