

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное образовательное учреждение  
высшего образования**  
**«Омский государственный технический университет»**  
Факультет информационных технологий и компьютерных систем  
Кафедра «Информатика и вычислительная техника»

**ОТЧЕТ**  
по лабораторной работе №2  
по дисциплине «Объектно-ориентированное программирование»

Выполнил студент  
группы ИВТ-212

Проверил ст. преподаватель  
Степанов П.П.

---

Омск, 2022

Задание:

Используя Visual Studio, создать проект по шаблону Console Application.

Требуется:

- создать класс, представляющий учебный класс Classroom;
- создать класс Pupil (ученик).
- В теле класса создать методы: Study(); Read(); Write(); Relax();
- создать три производных класса на основе базового класса Pupil: ExcelentPupil; GoodPupil; BadPupil;
- – преобразовать перечисленные три производных класса запечатанными и переопределить каждый из методов в зависимости от успеваемости ученика.

Конструктор класса Classroom принимает аргументы типа Pupil, класс должен состоять из четырех учеников. 45 Необходимо предусмотреть возможность того, что пользователь может передать два или три аргумента.

- Реализовать статическое поле «Количество учеников» в классе Classroom.

- К классу Pupil реализовать виртуальное свойство, доступное только для чтения GetCurrentGrade, позволяющее возвращать текущую оценку ученика (случайное число от 2 до 5).

- В классах-наследниках перегрузить свойство GetCurrentGrade так, чтобы вероятность получения хорошей оценки у способных учеников была выше.

- В классе Classroom реализовать свойство GetRoundGrade, вычисляющее средний балл по группе.

- Написать 10 unit-тестов, проверяющих корректность работы программы.

## Реализация класса Classroom:

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8
9 namespace ConsoleApp1;
10 class Classroom
11 {
12     Stack<Pupil> innerClass = new Stack<Pupil>();
13     static int pupilsCount = 4;
14     public int GetRoundGrade()
15     {
16         int roundedGrade = 0;
17         foreach(Pupil pupil in innerClass)
18         {
19             roundedGrade += pupil.GetCurrentGrade();
20         }
21         roundedGrade /= pupilsCount;
22         return roundedGrade;
23     }
24 }
25 public Classroom(Pupil p1, Pupil p2)
26 {
27     innerClass.Push(p1);
28     innerClass.Push(p2);
29     innerClass.Push(new BadPupil());
30     innerClass.Push(new GoodPupil());
31 }
32 public Classroom(Pupil p1, Pupil p2, Pupil p3)
33 {
```

Рис. 1 – Класс Classroom

```
18         roundedGrade += pupil.GetCurrentGrade();
19     }
20 }
21 roundedGrade /= pupilsCount;
22 return roundedGrade;
23 }
24 }
25 public Classroom(Pupil p1, Pupil p2)
26 {
27     innerClass.Push(p1);
28     innerClass.Push(p2);
29     innerClass.Push(new BadPupil());
30     innerClass.Push(new GoodPupil());
31 }
32 public Classroom(Pupil p1, Pupil p2, Pupil p3)
33 {
34     innerClass.Push(p1);
35     innerClass.Push(p2);
36     innerClass.Push(p3);
37     innerClass.Push(new ExelentPupil());
38 }
39 }
40 }
41 }
42 }
```

Рис. 2 - Класс Classroom

## Реализация базового класса Pupil:

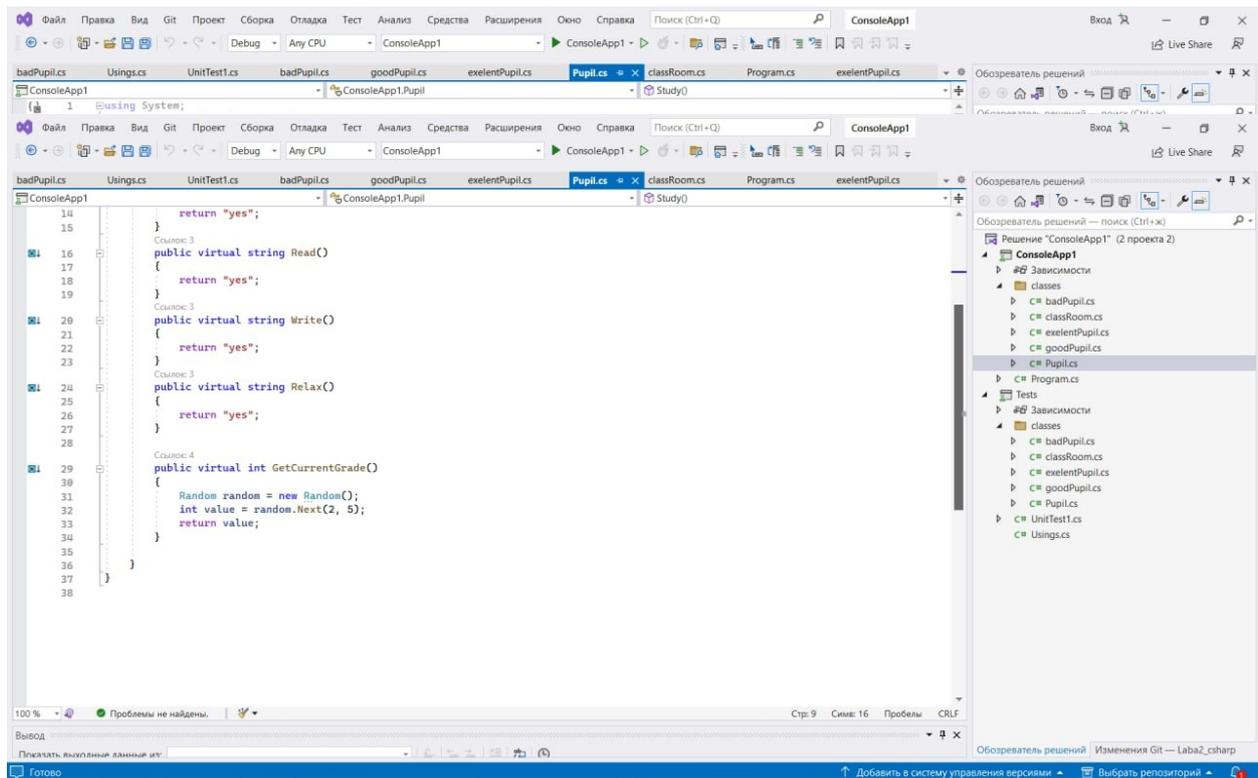


Рис.3 – Класс Pupil

Рис. 4 – Класс Pupil

## Реализация производного класса GoodPupil:

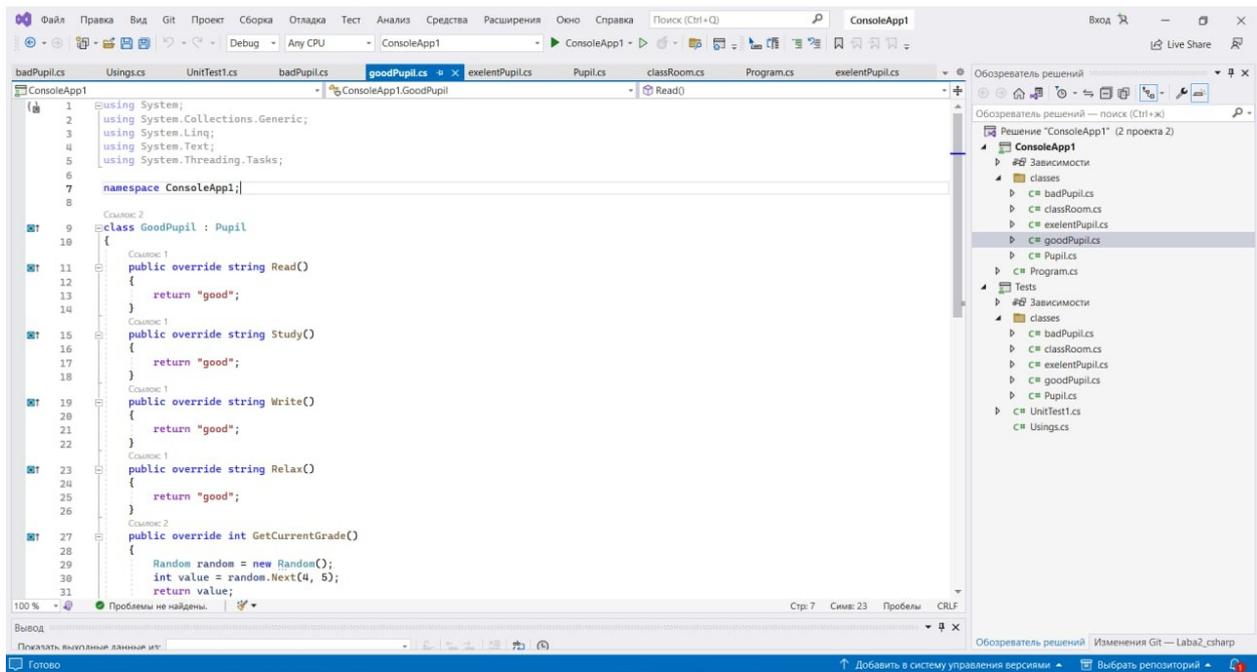


Рис. 5 – Класс GoodPupil

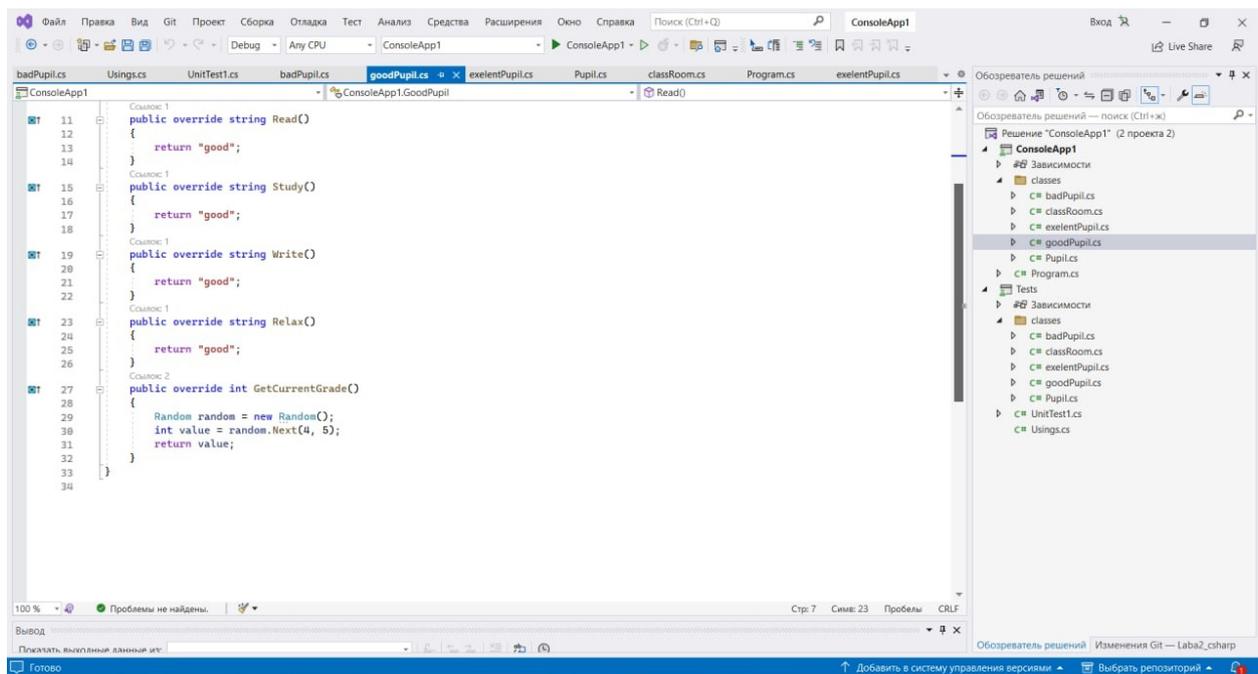


Рис. 6 – Класс GoodPupil

## Реализация производного класса BadPupil:

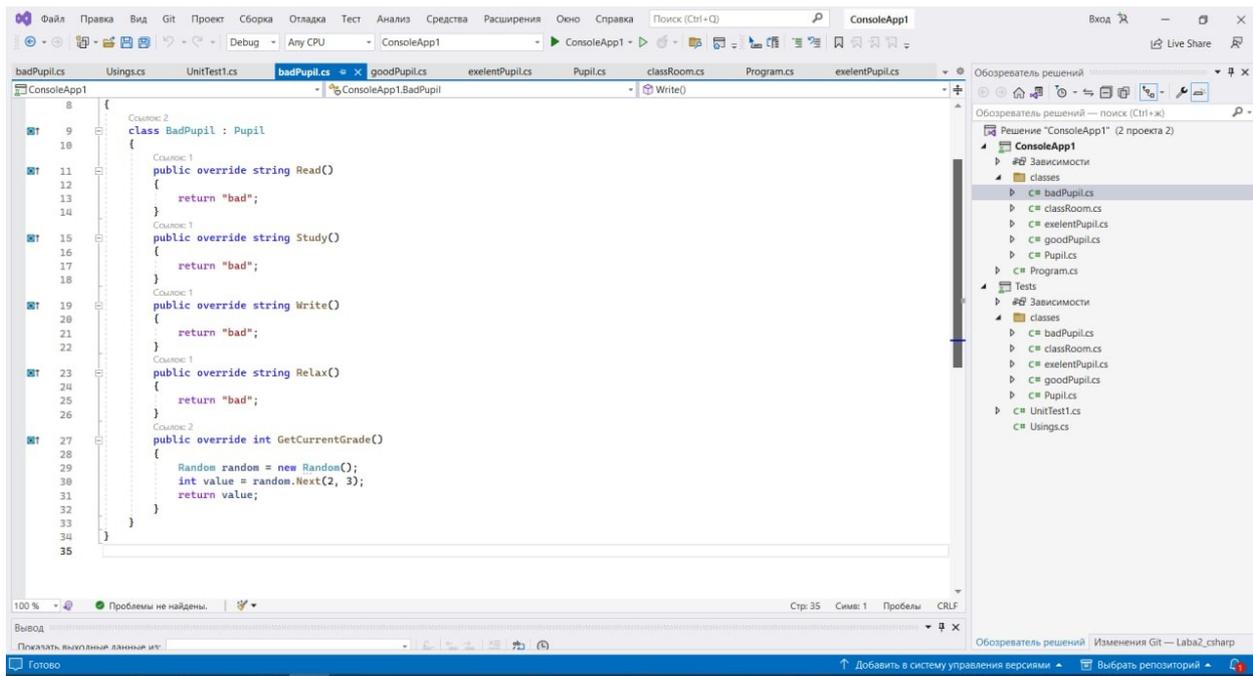


Рис. 7 – Класс BadPupil

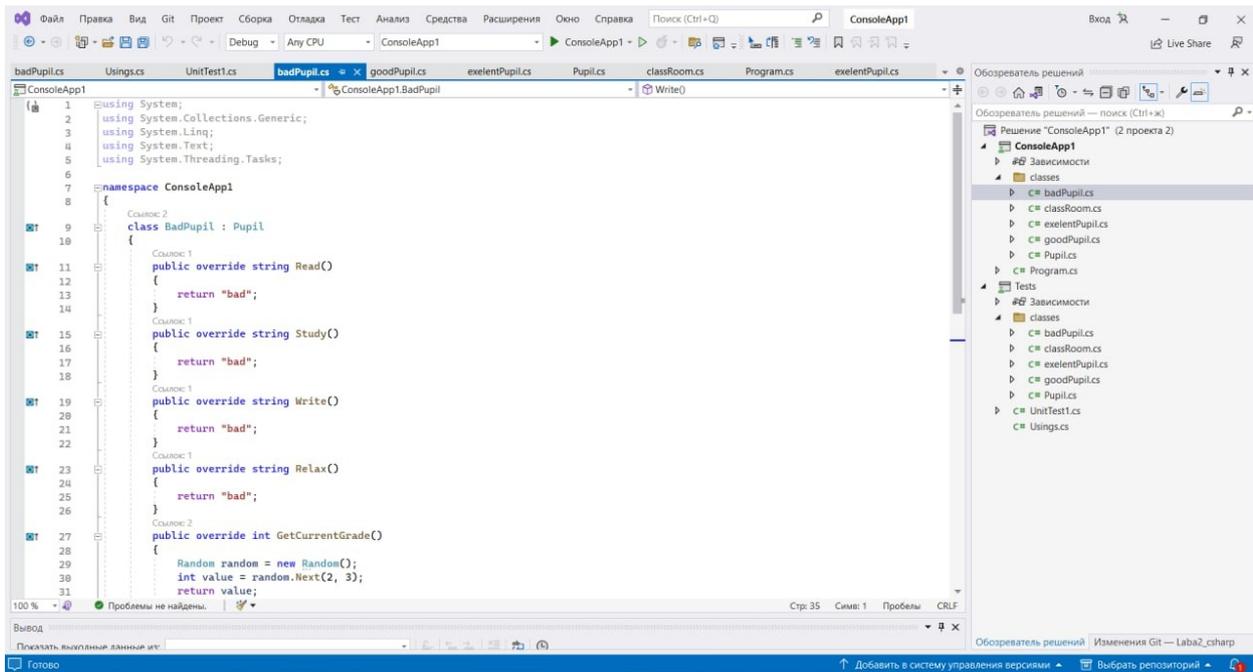


Рис. 8 – Класс BadPupil

## Реализация производного класса ExcelentPupil:

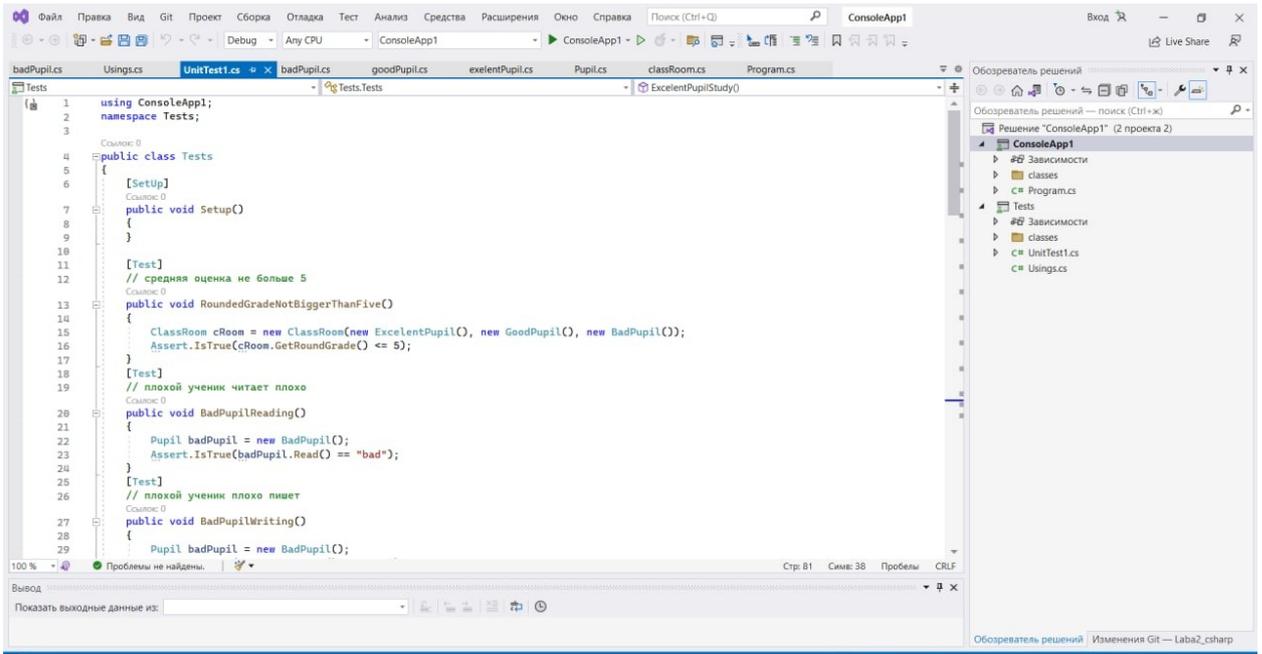
```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ConsoleApp1
8 {
9     class ExcelentPupil : Pupil
10    {
11        public override string Read()
12        {
13            return "exelent";
14        }
15        public override string Study()
16        {
17            return "exelent";
18        }
19        public override string Write()
20        {
21            return "exelent";
22        }
23        public override string Relax()
24        {
25            return "exelent";
26        }
27        public override int GetCurrentGrade()
28        {
29            Random random = new Random();
30            int value = random.Next(5, 5);
31            return value;
32        }
33    }
34 }
35
```

Рис. 9 – Класс ExcelentPupil

```
10 {
11    public override string Read()
12    {
13        return "exelent";
14    }
15    public override string Study()
16    {
17        return "exelent";
18    }
19    public override string Write()
20    {
21        return "exelent";
22    }
23    public override string Relax()
24    {
25        return "exelent";
26    }
27    public override int GetCurrentGrade()
28    {
29        Random random = new Random();
30        int value = random.Next(5, 5);
31        return value;
32    }
33 }
34 }
35
```

Рис. 10 – Класс ExcelentPupil

## Unit – тесты:



The screenshot shows the Visual Studio IDE with the 'UnitTest1.cs' file open. The code defines a 'Tests' class with several test methods. The first method, 'RoundedGradeNotBiggerThanFive()', tests a 'ClassRoom' object with 'ExcelentPupil', 'GoodPupil', and 'BadPupil' instances, asserting that the rounded grade is less than or equal to 5. The second method, 'BadPupilReading()', tests a 'BadPupil' instance, asserting that its 'Read()' method returns 'bad\*'. The third method, 'BadPupilWriting()', tests a 'BadPupil' instance, asserting that its 'Write()' method returns 'bad\*'. The interface includes a menu bar, a toolbar, a file explorer on the right, and a status bar at the bottom.

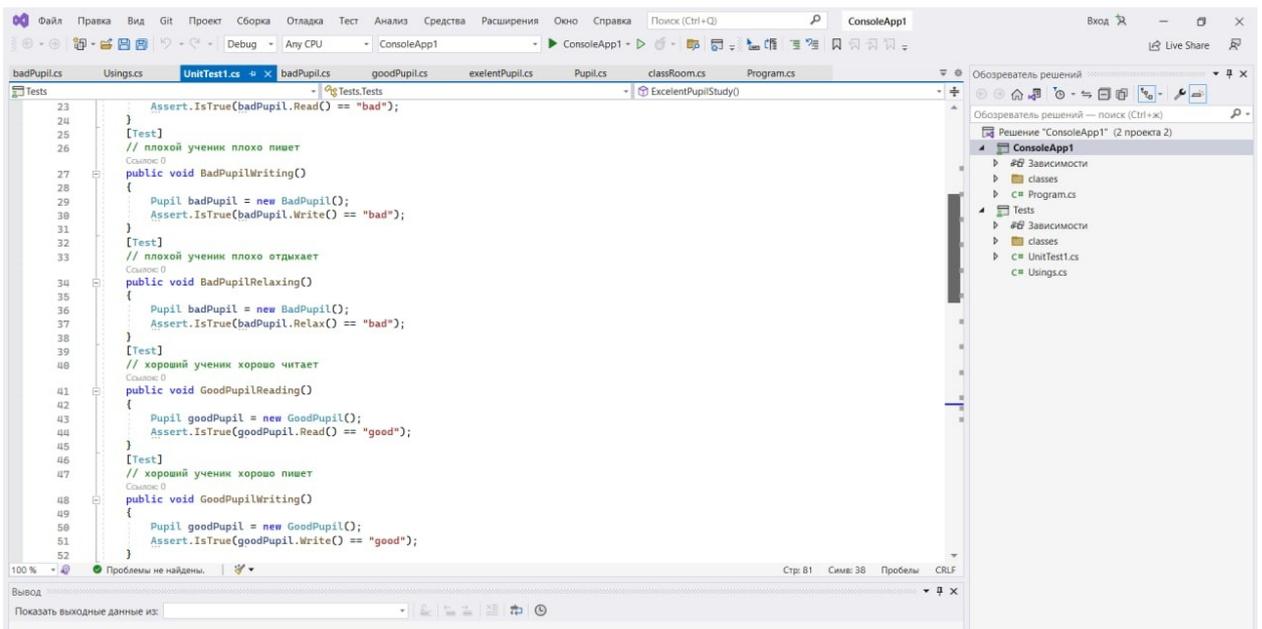
```
using ConsoleApp1;
namespace Tests
{
    public class Tests
    {
        [SetUp]
        public void Setup()
        {
        }

        [Test]
        // средняя оценка не больше 5
        public void RoundedGradeNotBiggerThanFive()
        {
            Classroom cRoom = new Classroom(new ExcelentPupil(), new GoodPupil(), new BadPupil());
            Assert.IsTrue(cRoom.GetRoundGrade() <= 5);
        }

        [Test]
        // плохой ученик читает плохо
        public void BadPupilReading()
        {
            Pupil badPupil = new BadPupil();
            Assert.IsTrue(badPupil.Read() == "bad*");
        }

        [Test]
        // плохой ученик плохо пишет
        public void BadPupilWriting()
        {
            Pupil badPupil = new BadPupil();
            Assert.IsTrue(badPupil.Write() == "bad*");
        }
    }
}
```

Рис. 11 – Unit – тесты



The screenshot shows the Visual Studio IDE with the 'UnitTest1.cs' file open, displaying the continuation of the test methods. The fourth method, 'BadPupilRelaxing()', tests a 'BadPupil' instance, asserting that its 'Relax()' method returns 'bad\*'. The fifth method, 'GoodPupilReading()', tests a 'GoodPupil' instance, asserting that its 'Read()' method returns 'good\*'. The sixth method, 'GoodPupilWriting()', tests a 'GoodPupil' instance, asserting that its 'Write()' method returns 'good\*'. The interface includes a menu bar, a toolbar, a file explorer on the right, and a status bar at the bottom.

```
    }
}

    [Test]
    // плохой ученик плохо пишет
    public void BadPupilWriting()
    {
        Pupil badPupil = new BadPupil();
        Assert.IsTrue(badPupil.Write() == "bad*");
    }

    [Test]
    // плохой ученик плохо отдыхает
    public void BadPupilRelaxing()
    {
        Pupil badPupil = new BadPupil();
        Assert.IsTrue(badPupil.Relax() == "bad*");
    }

    [Test]
    // хороший ученик хорошо читает
    public void GoodPupilReading()
    {
        Pupil goodPupil = new GoodPupil();
        Assert.IsTrue(goodPupil.Read() == "good*");
    }

    [Test]
    // хороший ученик хорошо пишет
    public void GoodPupilWriting()
    {
        Pupil goodPupil = new GoodPupil();
        Assert.IsTrue(goodPupil.Write() == "good*");
    }
}
```

Рис.12 – Unit – тесты

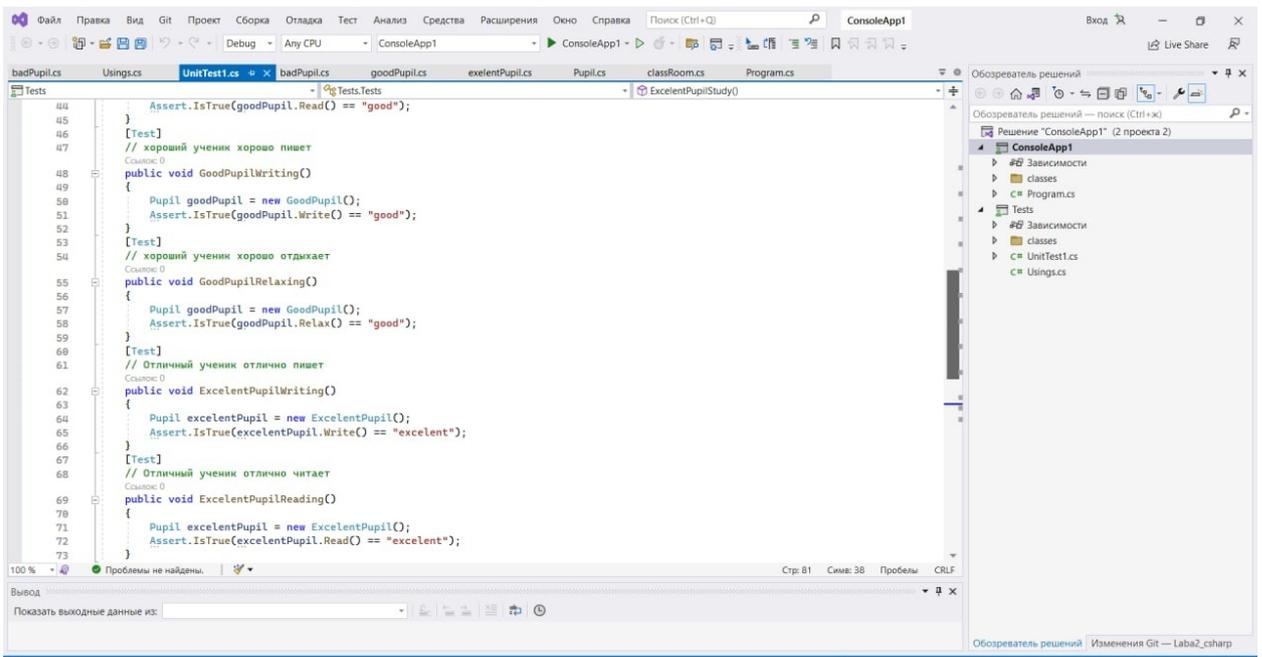


Рис. 13 - Unit – тесты

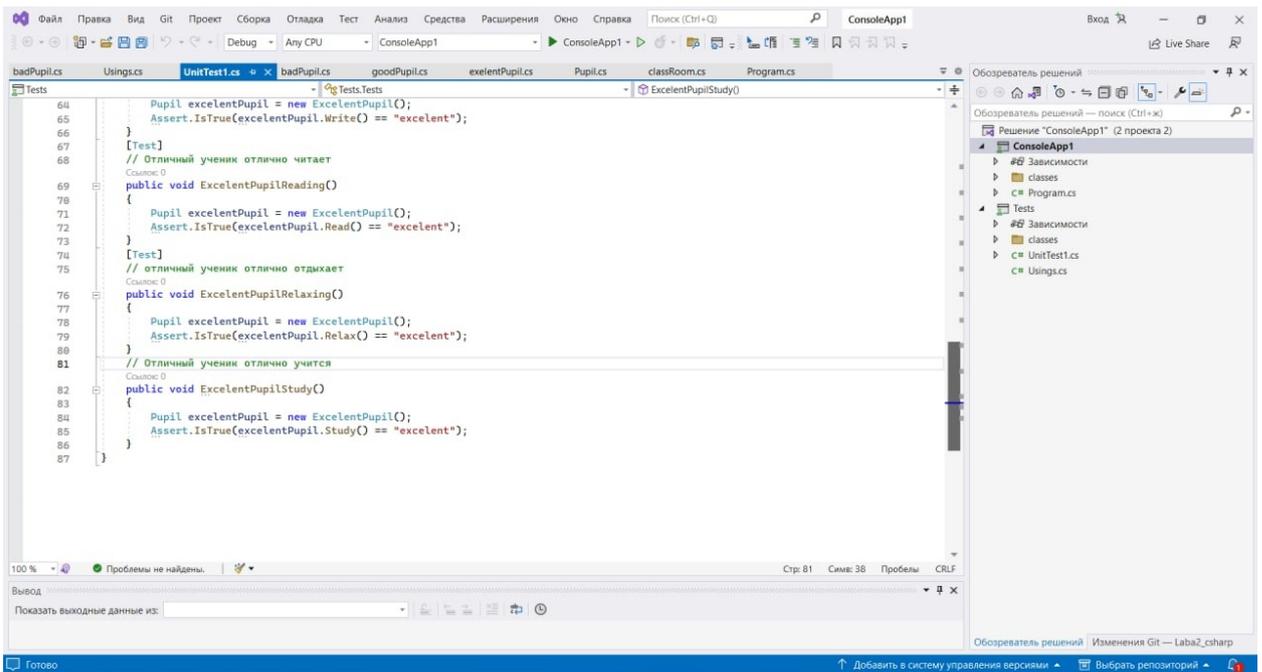


Рис. 14 - Unit – тесты