|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Полежаев Яков Вячеславович  **ЛАБОРАТОРНАЯ РАБОТА №10**  по направлению подготовки *38.03.05 Бизнес-информатика*  образовательная программа «Бизнес-информатика»   |  |  | | --- | --- | |  | Руководитель  Преподаватель кафедры ИТБ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Марквирер В.Д. |   Пермь, 2022 |

**Задача №1**

1. **Постановка задачи**
2. **Часть**

**Основное содержание работы.**

Написать программу, в которой создается иерархия классов. Записать объекты классов в массив, выполнить просмотр элементов массива. Показать использование виртуальных функций.

**Порядок выполнения работы.**

1. Определить иерархию классов (в соответствии с вариантом).
2. Реализовать классы.
3. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в массив, после чего массив просматривается.
4. Реализовать 2 варианта программы: с помощью виртуальных и не виртуальных методов.
5. Без виртуальных функций программа будет работать не правильно! Объяснить почему. Объяснить необходимость виртуальных функций

**1 вар.** (студент, преподаватель, **персона**, сотрудник;)

1. **Часть**

**Основное содержание работы.**

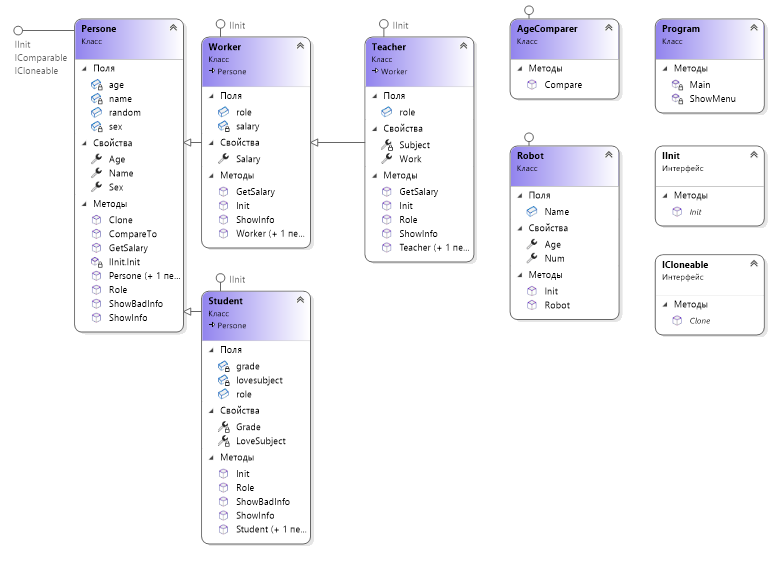
1. Реализовать метод для выполнения заданных запросов. При необходимости (для выполнения запроса) в класс могут быть добавлены новые поля (по сравнению с частью 1). В программе должно быть минимум ввода с клавиатуры. Поля объектов задаются в тексте программы. С клавиатуры вводятся только параметры запроса.
2. Реализовать не менее трех запросов, соответствующих иерархии классов (можно реализовать свои запросы).

**Запросы.**

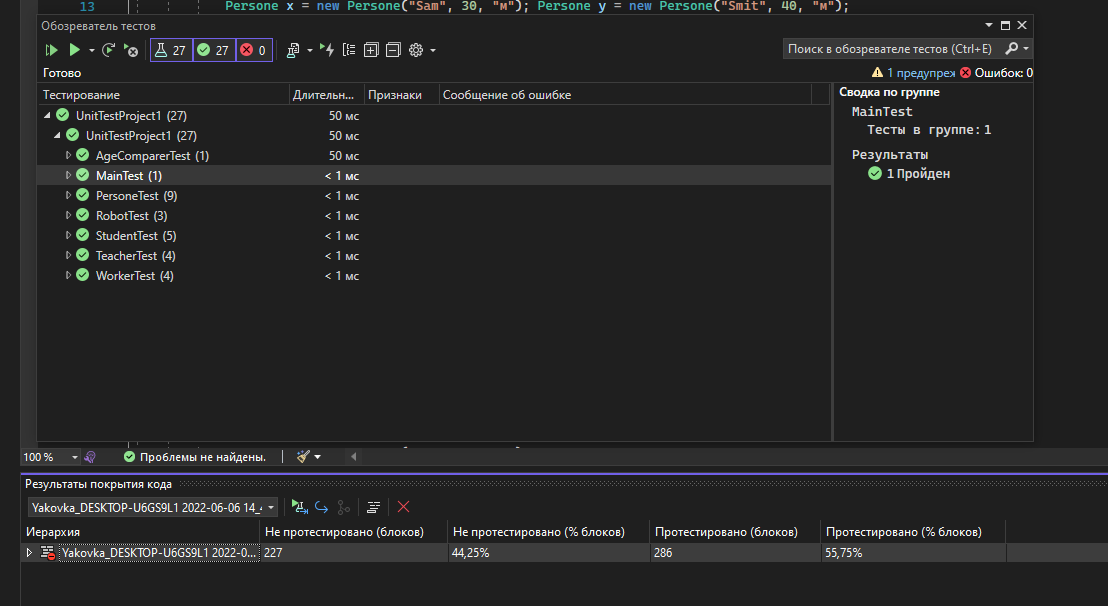
1) Имена всех лиц мужского (женского) пола.

1. **Часть**
2. Создать интерфейс IInit, который содержит метод Init() для формирования элементов класса с помощью ДСЧ. Реализовать этот интерфейс в иерархии классов (часть 1 и часть 2).
3. Добавить еще один класс, не относящийся к этой иерархии классов, но реализующий интерфейс IInit.
4. Создать массив элементов типа IInit и поместить в него экземпляры различных классов иерархии и класса, созданного в задании 2. Выполнить просмотр массива.
5. Создать массив из элементов, относящихся к иерархии классов (часть 1 и часть 2).
6. Реализовать сортировку элементов массива, используя стандартный интерфейс IComparable и метод Sort класса Array.
7. Реализовать сортировку и поиск элемента в массиве, используя стандартный интерфейс ICompare и метод Sort класса Array.
8. Реализовать метод клонирования объектов из интерфейса IClonable. Показать разницу между клонированием и поверхностным копированием объектов.

**2. Диаграмма классов (общая)**



1. **Код программы (общий)**
2. using System;
3. using System.Collections.Generic;
4. using System.Linq;
5. using System.Text;
6. using System.Threading.Tasks;
7. namespace лаб\_10
8. {
9. public interface ICloneable
10. {
11. object Clone();
12. }
13. public interface IInit
14. {
15. void Init();
16. }
17. public class Persone:IInit, IComparable, ICloneable
18. {
19. public object Clone() => MemberwiseClone();
20. public int CompareTo(object obj)//для сортировки по первым буквам
21. {
22. Persone temp = obj as Persone;
23. char[] first1 = this.Name.ToCharArray();
24. char[] first2 = temp.Name.ToCharArray();
25. if(temp != null)
26. {
27. if(first1[0].GetHashCode() == first2[0].GetHashCode())
28. {
29. return 0;
30. }
31. else if(first1[0].GetHashCode() > first2[0].GetHashCode())
32. {
33. return 1;
34. }
35. else if (first1[0].GetHashCode() < first2[1].GetHashCode()){ return -1;}
36. }
37. else { throw new Exception("параметр не преобразуется в Persone"); }
38. return 0;
39. }
40. public static Random random = new Random();
41. public Persone(string name,int Age,string Sex)
42. {
43. this.Name = name;
44. this.Age = Age;
45. this.Sex = Sex;
46. }
47. public Persone()
48. {
49. this.Name = "NO name";
50. this.Age = 0;
51. this.Sex = "No Sex :( ";
52. }
53. private string name;
54. public string Name { get => name; set { name = value; } }
55. private int age;
56. public int Age
57. {
58. get => age;
59. set
60. {
61. if(value > 0)
62. {
63. age = value;
64. }
65. else
66. {
67. Console.WriteLine("нереальное значение");
68. throw new ArgumentException();
69. }
70. }
71. }
72. private string sex;
73. public string Sex
74. {
75. get => sex;
76. set
77. {
78. if(value == "м" || value == "ж")
79. {
80. sex = value;
81. }
82. else{ Console.WriteLine("ошибка при вводе пола"); throw new ArgumentException();}
83. }
84. }
85. public virtual void ShowInfo()
86. {
87. Console.WriteLine($"имя:{this.Name} возраст:{this.Age}");
88. }
89. public void ShowBadInfo()
90. {
91. Console.WriteLine($"имя:{this.Name} возраст:{this.Age}");
92. }
93. public virtual int GetSalary()
94. {
95. return 0;
96. }
97. public virtual string Role()
98. {
99. return "NoRole";
100. }
101. void IInit.Init()
102. {
103. this.Age = random.Next(8, 100);
104. }
105. }
106. internal class Robot: IInit
107. {
108. public Robot()
109. {
110. Name = "Robot";
111. Age = 0;
112. }
113. public string Name = "Robot";
114. public int Num { get; set; }
115. public int Age { get; set; }
116. public void Init()
117. {
118. this.Num = Persone.random.Next(1,1000);
119. this.Age = Persone.random.Next(1, 300);
120. }
121. }
122. internal class Student : Persone, IInit
123. {
124. public Student() : base()
125. {
126. Grade = 0;
127. LoveSubject = "No subject";
128. }
129. public Student(string Name,int Age, string Sex,int Grade,string LoveSubject): base(Name, Age,Sex)
130. {
131. this.Grade = Grade;
132. this.LoveSubject = LoveSubject;
133. }
134. private int grade;
135. private int Grade
136. {
137. get => grade;
138. set
139. {
140. if (value > 0 && value <= 11)
141. {
142. grade = (int)value;
143. }
144. else
145. {
146. Console.WriteLine("нереальное значение");
147. throw new ArgumentException();
148. }
149. }
150. }
151. private string lovesubject;
152. private string LoveSubject { get =>lovesubject; set { lovesubject = value; } }
153. public void Init()
154. {
155. this.Grade = (int)random.Next(1, 11);
156. this.Age = (int)random.Next(8, 23);
157. }
158. public override void ShowInfo()
159. {
160. Console.WriteLine($"имя:{this.Name} возраст:{this.Age} занятость:ученик класс:{Grade} любимый предмет:{LoveSubject} ");
161. }
162. public void ShowBadInfo()
163. {
164. Console.WriteLine($"имя:{this.Name} возраст:{this.Age} занятость:ученик класс:{Grade} любимый предмет:{LoveSubject} ");
165. }
166. public string role = "Студент";
167. public override string Role()
168. {
169. return this.role;
170. }
171. }
172. internal class Teacher: Worker, IInit
173. {
174. public Teacher() : base()
175. {
176. Salary = 0;
177. Subject = "No Subject";
178. }
179. public Teacher(string Name, int Age, string Sex, int Salary, string Subject) : base(Name, Age, Sex, Salary)
180. {
181. this.Salary = Salary;
182. this.Subject = Subject;
183. this.Work = "Учитель";
184. }
185. public string Work { get; set; }
186. private string Subject { get; set; }
187. public override void ShowInfo()
188. {
189. Console.WriteLine($"имя:{this.Name} возраст:{this.Age} занятость:учитель зарплата:{Salary}руб/мес предмет:{Subject} ");
190. }
191. public override int GetSalary()
192. {
193. return this.Salary;
194. }
195. new public string role = "Учитель";
196. public override string Role()
197. {
198. return this.role;
199. }
200. new public void Init()
201. {
202. this.Salary = random.Next(10000, 30000);
203. this.Age = random.Next(21, 80);
204. }
205. }
206. public class Worker : Persone, IInit
207. {
208. public string role = "Простой рабочий";
209. public Worker() : base()
210. {
211. Salary = 0;
212. }
213. public Worker(string Name,int Age, string Sex, int Salary):base(Name,Age,Sex)
214. {
215. this.Salary = Salary;
216. }
217. private int salary;
218. public int Salary
219. {
220. get => salary;
221. set
222. {
223. if (value > 0)
224. {
225. salary = (int)value;
226. }
227. else
228. {
229. Console.WriteLine("нереальное значение");
230. throw new ArgumentException();
231. }
232. }
233. }
234. public override void ShowInfo()
235. {
236. Console.WriteLine($"имя:{this.Name} возраст:{this.Age} занятость:работник зарплата:{Salary}руб/мес ");
237. }
238. public override int GetSalary()
239. {
240. return this.Salary;
241. }
242. public void Init()
243. {
244. this.Salary = random.Next(10000, 30000);
245. this.Age = random.Next(21, 80);
246. }
247. }
248. internal class Program
249. {
250. static void ShowMenu()
251. {
252. Console.WriteLine("1. мужчины\n" +
253. "2. женщины\n" +
254. "3. работающие\n" +
255. "4. связанные со школой\n" +
256. "5. ученики\n" +
257. "6. закончить программу\n" +
258. "7. Задать данные с ДСЧ\n");
259. }
260. static void Main(string[] args)
261. {
262. Persone[] people = new Persone[5];
263. people[0] = new Persone("Abraham" , 60,"м");
264. people[1] = new Student("Lina", 20, "м", 11, "Math");
265. people[2] = new Teacher("Vasily", 40, "м", 30000,"PE");
266. people[3] = new Worker("Vlad", 21, "м", 20000);
267. people[4] = new Student("Sam", 18, "м", 10, "English");
268. Console.WriteLine("Список людей:");
269. int i = 1;
270. foreach(Persone item in people)
271. {
272. Console.WriteLine($"{i} человек");
273. item.ShowInfo();
274. Console.WriteLine();
275. i++;
276. }
277. Console.WriteLine();
278. Console.WriteLine("Плохой пример");
279. people[0].ShowBadInfo();
280. people[1].ShowBadInfo();
281. Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");
282. Console.ReadKey();
283. Console.Clear();
284. bool megaFlag = true;
285. do
286. {
287. Console.WriteLine("Выберите категорию и напишите ее номер:");
288. int option = 0;
289. Console.WriteLine();
290. Program.ShowMenu();
291. bool flag = false;
292. do
293. {
294. try
295. {
296. option = Int32.Parse(Console.ReadLine()); flag = true;
297. }
298. catch (FormatException) { Console.WriteLine("Ошибка ввода"); flag = false; }
299. } while (flag == false);
301. switch (option)
302. {
303. case 1:
304. Console.Clear();
305. Console.WriteLine("Вы выбрали категорию Мужчины\n");
306. int manCount = 0;
307. foreach (var item in people)
308. {
309. if (item.Sex == "м") { Console.WriteLine($"{manCount + 1}. имя:{item.Name} возраст:{item.Age}"); manCount++; }
310. }
311. Console.WriteLine($"Всего в списке {manCount} мужчин\n");
312. Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");
313. Console.ReadKey();
314. Console.Clear();
315. break;
316. case 2:
317. Console.Clear();
318. Console.WriteLine("Вы выбрали категорию Жещины\n");
319. int womanCount = 0;
320. foreach (var item in people)
321. {
322. if (item.Sex == "ж") { Console.WriteLine($"{womanCount + 1}. имя:{item.Name} возраст:{item.Age}"); womanCount++; }
323. }
324. Console.WriteLine($"Всего в списке {womanCount} женщин\n");
325. Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");
326. Console.ReadKey();
327. Console.Clear();
328. break;
329. case 3:
330. Console.Clear();
331. Console.WriteLine("Вы выбрали категорию Работающие\n");
332. int workerCount = 0;
333. foreach (var item in people)
334. {
336. if ( item is Worker) { item.ShowInfo(); workerCount++; }
337. }
338. Console.WriteLine($"Всего в списке {workerCount} работающих\n");
339. Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");
340. Console.ReadKey();
341. Console.Clear();
342. break;
343. case 4:
344. Console.Clear();
345. Console.WriteLine("Вы выбрали категорию Связанные со школой\n");
346. int schoolerCount = 0;
347. foreach (Persone item in people)
348. {
349. if (item is Teacher || item is Student) { Console.WriteLine($"{schoolerCount + 1}. имя:{item.Name} возраст:{item.Age} роль:{item.Role()}"); schoolerCount++; }
350. }
351. Console.WriteLine($"Всего в списке {schoolerCount} связанных со школой\n");
352. Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");
353. Console.ReadKey();
354. Console.Clear();
355. break;
356. case 5:
357. Console.Clear();
358. Console.WriteLine("Вы выбрали категорию ученики\n");
359. int studentCount = 0;
360. foreach (Persone item in people)
361. {
362. if (item is Student) { Console.WriteLine($"{studentCount + 1}. имя:{item.Name} возраст:{item.Age}"); studentCount++; }
363. }
364. Console.WriteLine($"Всего в списке {studentCount} учащихся\n");
365. Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");
366. Console.ReadKey();
367. Console.Clear();
368. break;
369. case 6:
370. megaFlag = false;
371. break;
372. case 7:
373. IInit[] inits = new IInit[6];// 3 часть
374. for (int j = 0; j < people.Length; j++)
375. {
376. inits[j] = people[j];
377. }
378. Robot robot = new Robot();
379. inits[5] = robot;
380. foreach(IInit item in inits)
381. {
382. item.Init();
383. Console.WriteLine(item.GetType());
384. }
385. Console.ReadKey();
386. Console.Clear();
387. Console.WriteLine("Реализация IIComparable");
388. Console.WriteLine("Сортировка по имени");
389. Array.Sort(people);
390. Console.WriteLine();
391. foreach (Persone item in people)
392. {
393. item.ShowInfo();
394. Console.WriteLine(new String ('-',10));
395. }
396. Console.WriteLine();
397. Console.WriteLine("Сортировка по возрасту");
398. Console.WriteLine();
399. Array.Sort(people,new AgeComparer());
400. foreach(Persone item in people)
401. {
402. item.ShowInfo();
403. Console.WriteLine(new String('-', 10));
404. }
405. Console.ReadKey();
406. Console.Clear();
407. Console.WriteLine("Копирование");
408. Persone [] clonePersone = new Persone[3];
409. clonePersone[0] = new Persone("Smit", 20, "м");
410. clonePersone[1] = (Persone)clonePersone[0].Clone();
411. clonePersone[2] = new Persone("Smit", 20, "м");
412. foreach(Persone item in clonePersone) { item.ShowInfo(); }
413. clonePersone[0].Age = 30;
414. Console.WriteLine(new string('-', 10));
415. foreach (Persone item in clonePersone) { item.ShowInfo(); }
416. Console.WriteLine(new string('-', 10));
417. Console.WriteLine("Клонирование");
418. clonePersone[1] = clonePersone[0];
419. foreach (Persone item in clonePersone) { item.ShowInfo(); }
420. Console.WriteLine(new string('-', 10));
421. clonePersone[0].Age = 40;
422. foreach (Persone item in clonePersone) { item.ShowInfo(); }
423. Console.ReadKey();
424. Console.Clear();
425. break;
426. default:
427. Console.WriteLine("Ошибка ввода");
428. Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");
429. Console.ReadKey();
430. continue;
431. }
432. }while(megaFlag);
434. }
435. }
436. }
437. **Код юнит теста.**
438. using Microsoft.VisualStudio.TestTools.UnitTesting;
439. using System;
440. using лаб\_10;
441. namespace UnitTestProject1
442. {
443. [TestClass]
444. public class AgeComparerTest
445. {
446. [TestMethod]
447. public void Compare\_30\_to\_40\_Age ()
448. {
449. Persone x = new Persone("Sam", 30, "м"); Persone y = new Persone("Smit", 40, "м");
450. int expected = -1;
451. AgeComparer c = new AgeComparer();
452. int actual = c.Compare(x,y);
453. Assert.AreEqual(expected, actual);
454. }
455. }
456. [TestClass]
457. public class PersoneTest
458. {
459. [TestMethod]
460. public void Clone\_Test()
461. {
462. Persone x = new Persone("Sam", 30, "м");
463. Persone actual = x;
464. Persone expected = (Persone)x.Clone();
465. actual.Age += 20;
466. Assert.AreNotEqual(expected.Age,actual.Age);
467. }
468. [TestMethod]
469. public void Init\_Test()
470. {
471. Persone x = new Persone("Sam", 30, "м");
472. x.Init();
473. }
474. [TestMethod]
475. public void CompareTo\_30\_30()
476. {
477. Persone x = new Persone("Sam", 30, "м"); Persone y = new Persone("Smit", 30, "м");
478. int actual = x.CompareTo(y);
479. int expected = 0;
480. Assert.AreEqual(expected, actual);
481. }
482. [TestMethod]
483. public void Name\_Test()
484. {
485. Persone x = new Persone();
486. x.Name = "Smit";
487. string actual = x.Name;
488. string expected = "Smit";
489. Assert.AreEqual(expected,actual);
491. }
492. [TestMethod]
493. public void Age\_Test()
494. {
495. Persone x = new Persone();
496. x.Age = 5;
497. int actual = x.Age;
498. int expected = 5;
499. Assert.AreEqual(expected, actual);
500. }
501. [TestMethod]
502. public void Sex\_Test()
503. {
504. Persone x = new Persone();
505. x.Sex = "вап";
506. string actual = x.Sex;
507. string expected = null;
508. Assert.AreEqual(expected, actual);
509. }
510. [TestMethod]
511. public void Salary\_Test()
512. {
513. Persone x = new Persone();
514. int actual = x.GetSalary();
515. int expected = 0;
516. Assert.AreEqual(expected, actual);
517. }
518. [TestMethod]
519. public void Role\_Test()
520. {
521. Persone x = new Persone();
522. string actual = x.Role();
523. string expected = "NoRole";
524. Assert.AreEqual(expected, actual);
525. }
526. [TestMethod]
527. public void ShowInfo\_Test()
528. {
529. Persone x = new Persone("Sam", 30, "м");
530. x.ShowInfo();
531. x.ShowBadInfo();
532. }
533. }
534. [TestClass]
535. public class RobotTest
536. {
537. [TestMethod]
538. public void Init\_Test()
539. {
540. Robot robot = new Robot();
541. robot.Init();
542. }
543. [TestMethod]
544. public void Num\_Test()
545. {
546. Robot robot = new Robot();
547. robot.Num = 1;
548. int actual = robot.Num;
549. int expected = 1;
550. Assert.AreEqual(expected, actual);
551. }
552. [TestMethod]
553. public void Age\_Test()
554. {
555. Robot robot = new Robot();
556. robot.Age = -5;
557. int actual = robot.Age;
558. int expected = 0;
559. Assert.AreEqual(expected, actual);
560. robot.Age = 10;
561. actual = robot.Age;
562. expected = 10;
563. Assert.AreEqual(expected, actual);
564. }
565. }
566. [TestClass]
567. public class StudentTest
568. {
569. [TestMethod]
570. public void Init\_Test()
571. {
572. Student x = new Student("Sam", 30, "м", 11, "Math");
573. x.Init();
574. }
575. [TestMethod]
576. public void ShowInfo\_Test()
577. {
578. Student x = new Student("Sam", 30, "м",11,"Math");
579. x.ShowInfo();
580. x.ShowBadInfo();
581. }
582. [TestMethod]
583. public void Grade\_Test()
584. {
585. Student student = new Student();
586. student.Grade = 20;
587. int actual = student.Grade;
588. int expected = 0;
589. Assert.AreEqual(expected, actual);
590. student.Grade = 11;
591. actual = student.Grade;
592. expected = 11;
593. Assert.AreEqual(expected, actual);
594. student.Grade = -1;
595. actual = student.Grade;
596. expected = 0;
597. Assert.AreEqual(expected, actual);
598. }
599. [TestMethod]
600. public void LoveSubject\_Test()
601. {
602. Student student = new Student();
603. student.LoveSubject = "Math";
604. string actual = student.LoveSubject;
605. string expected = "Math";
606. Assert.AreEqual(expected, actual);
607. }
608. [TestMethod]
609. public void Role\_Test()
610. {
611. Student student = new Student();
612. string actual = student.Role();
613. string expected = "Студент";
614. Assert.AreEqual(expected, actual);
615. }
616. }
617. [TestClass]
618. public class TeacherTest
619. {
620. Teacher teacher = new Teacher("Sam", 25, "м", 30000, "Физра");
621. [TestMethod]
622. public void Init\_Test()
623. {
624. teacher.Init();
625. }
626. [TestMethod]
627. public void ShowInfo\_Test()
628. {
629. teacher.ShowInfo();
630. teacher.ShowBadInfo();
631. }
632. [TestMethod]
633. public void Salary\_Test()
634. {
636. int actual = teacher.GetSalary();
637. int expected = 30000;
638. Assert.AreEqual(expected, actual);
639. }
640. [TestMethod]
641. public void Role\_Test()
642. {
643. string actual = teacher.Role();
644. string expected = "Учитель";
645. Assert.AreEqual(expected, actual);
646. }
647. }
648. [TestClass]
649. public class WorkerTest
650. {
651. [TestMethod]
652. public void Init\_Test()
653. {
654. Worker worker = new Worker("Sam", 27, "м", 15000);
655. worker.Init();
656. }
657. Worker worker = new Worker("Sam",27,"м",15000);
658. [TestMethod]
659. public void ShowInfo\_Test()
660. {
661. worker.ShowInfo();
662. worker.ShowBadInfo();
663. }
664. [TestMethod]
665. public void GetSalary\_Test()
666. {
667. int actual = worker.GetSalary();
668. int expected = 15000;
669. Assert.AreEqual(expected, actual);
670. }
671. [TestMethod]
672. public void Salary\_Test()
673. {
674. worker.Salary = 50000;
675. int actual = worker.Salary;
676. int expected = 50000;
677. Assert.AreEqual(expected, actual);
678. }
679. }
680. [TestClass]
681. public class MainTest
682. {
683. [TestMethod]
684. public void ShowMenu\_Test()
685. {
686. Program.ShowMenu();
687. }
688. }
689. }
690. **Анализ покрытия кода тестами**

****

**Во всех классах были сделаны проверки, но в классе програм я не понял, что проверять.**