Практическая работа №10

Модульное тестирование

Цель лабораторной работы: Получить практические навыки модульного тестирования кода программных компонентов.

Создание проекта тестирования

Модульное тестирование используется разработчиками для проверки правильности работы методов классов. Модульные тесты сохраняются в системе контроля версий и выполняются при каждом построении приложения. Кроме того, модульные тесты являются основой регрессионного тестирования, которое выполняется при добавлении новых возможностей или модификации приложения.

Для реализации модульного тестирования разрабатываемого приложения создадим тестовый проект модульного тестирования. Для этого добавим в решение ProjectTeachingLoadOfTeachers проект модульного теста (рис. 1), выбрав последовательно ссылку Тест (1), шаблонПроект модульного теста (2) и введя имя проекта UnitTestProjectTeachersLoad (3).

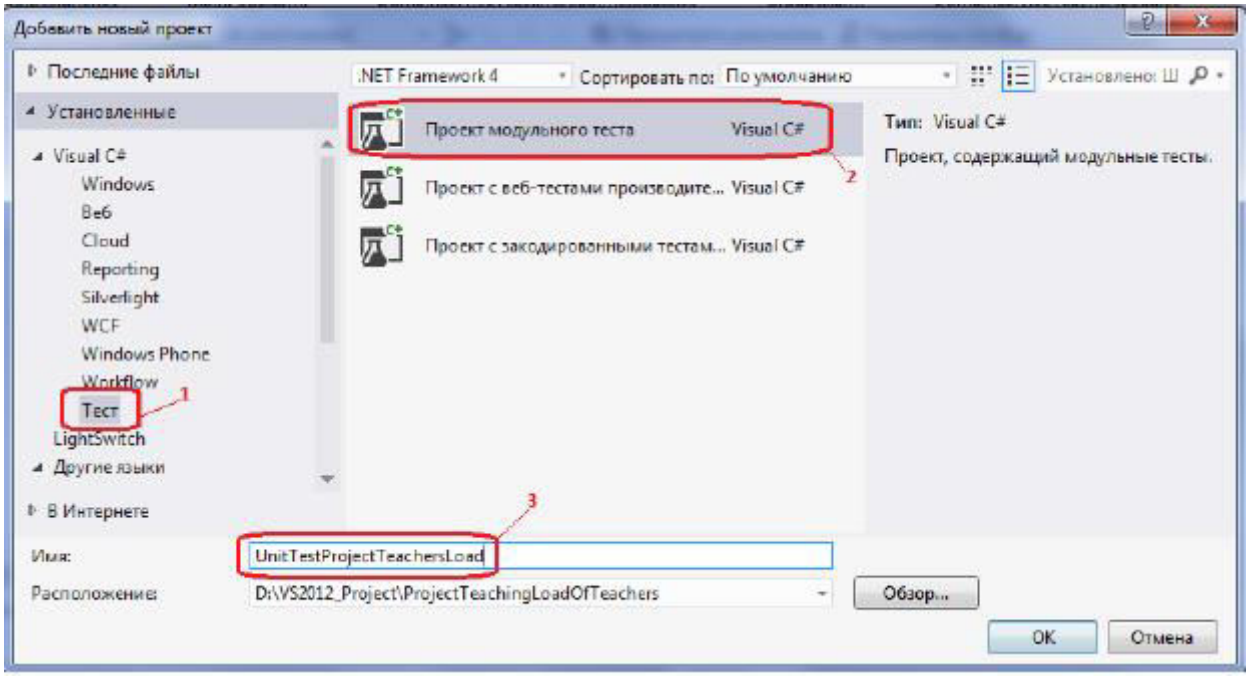

Рис. 1. Создание проекта модульного теста

Создание теста

При создании проекта модульного теста генерируется класс для тестирования UnitTest, который переименуем, задав имя UnitTestGroup.

```
{
```

```
    [TestClass]
    public class UnitTestGroup
    {
        [TestMethod]
        public void TestMethodGroup()
        {
            // Код теста
  }
    }
}
```

Предположим, что предполагается тестирование уровня представления для задачи Ввод, редактирование и удаление данных по студенческим группам.

Подготовим код теста.

```
[TestClass]
    public class UnitTestGroup
    {
        [TestMethod]
        public void TestMethodGroup()
        {
              // Completion flag
            bool completed = false;
            // Handle error
            viewModel.ErrorNotice += (s, ea) => Assert.Fail(ea.Data.Message);
            // Handle property change
            viewModel.PropertyChanged += (s, ea) =>
            {
                if (ea.PropertyName == "Employees"
                    && viewModel.Gpoups != null) completed = true;
            };
            // Call ViewModel method
            EnqueueCallback(() => viewModel.Load Gpoups());
            // Wait for completion with timeout
            int timeoutSeconds = 10; // Debugging: Timeout.Infinite
            this.EnqueueConditional(() => completed, timeoutSeconds);
            // Perform Asserts
            EnqueueCallback(() =>
                {
                    Assert.IsNotNull(viewModel.Gpoups);
                });
            // Complete test
            EnqueueTestComplete();
        }
    }
}
```

Результаты тестирования формируются классом Assert, который производит сравнение ожидаемых результатов с фактическими.

После построения тестового проекта созданный тест будет отображаться в обозревателе тестов (рис. 2).

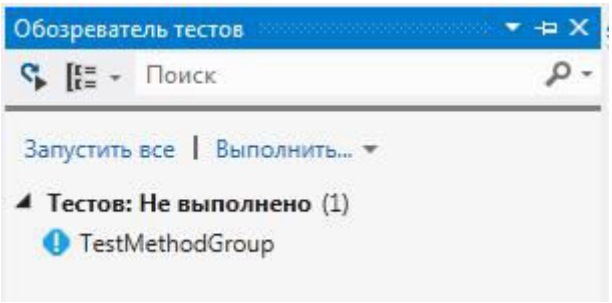

Рис. 2. Невыполненный тест в обозревателе тестов

При выполнении теста в обозревателе тестов отображаются параметры его выполнения (рис. 3).

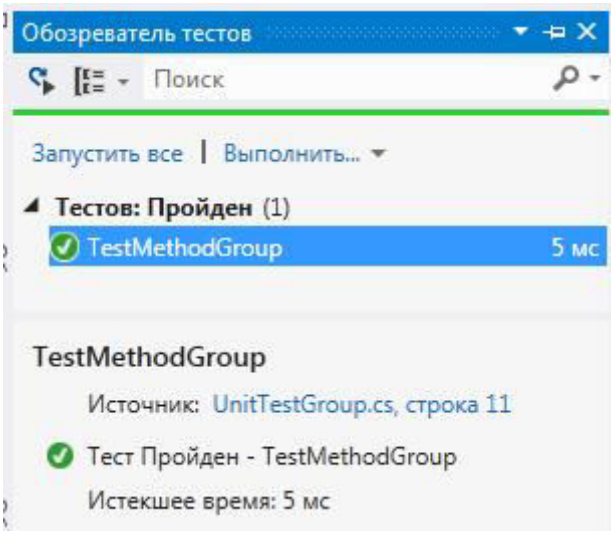

Рис. 3. Выполненный тест в обозревателе тестов

При разработке модульных тестов следует придерживаться следующих рекомендаций:

- каждый тест должен проверять небольшой срез функциональности;
- тесты должны быть автономными и изолированными;
- тестировать следует как ожидаемые, так и ошибочные ситуации.

Задание

1. Изучить теоретический материал.
2. По задания преподавателя провести модульное тестирование классов приложения.