**АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ**

**ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ**

**(АКТ (ф) СПбГУТ)**

**Отчеты по лабораторным и практическим работам**

**ПиТПМ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИСПП-35 | |  |  | *Туйкова А.Е* |
|  | (Группа) | | (Подпись) | (Дата) | (И.О. Фамилия) |
| Преподаватель | | |  |  | *Садовский Р. В.* |
|  | |  | (Подпись) | (Дата) | (И.О. Фамилия) |

Архангельск 2024

# **Лабораторная работа №6**

**Модульное тестирование**

**1 Цель работы**

1.1 Тестирование программного кода по методологии белого ящика;

Репозиторий: <https://github.com/kkerra/UnitTesting>

**2 Контрольные вопросы**

2.1 Для чего используется атрибут [Fact]?

Атрибут [Fact] в xUnit используется для обозначения тестового метода, который не принимает параметров и предназначен для выполнения единственного тестового случая. Этот атрибут применяется к методам, которые проверяют конкретное поведение или результат без использования входных данных из вне.

2.2 Для чего используется атрибут [Theory]?

Атрибут [Theory] в xUnit используется для обозначения параметризованного тестового метода, который может принимать различные наборы входных данных. Метод, помеченный [Theory], обычно сопровождается атрибутами [InlineData], [MemberData] или другими атрибутами, предоставляющими данные для тестирования. Это позволяет выполнять один тест с разными наборами входных данных.

2.3 Какие методы используются в Assert и для чего они предназначены?

Assert - это статический класс в xUnit, предоставляющий множество методов для проверки условий в тестах. Вот некоторые из наиболее часто используемых методов:

- Assert.Equal(expected, actual): Проверяет, что два значения равны. Если они не равны, тест не проходит.

- Assert.NotEqual(expected, actual): Проверяет, что два значения не равны.

- Assert.True(condition): Проверяет, что условие истинно.

- Assert.False(condition): Проверяет, что условие ложно.

- Assert.Null(object): Проверяет, что объект равен null.

- Assert.NotNull(object): Проверяет, что объект не равен null.

- Assert.Throws<TException>(Action): Проверяет, что действие выбрасывает исключение заданного типа.

- Assert.Contains(substring, string): Проверяет, что строка содержит заданную подстроку.

**3 Выводы**

3.1 Протестировали программный код по методологии белого ящика;

# **Лабораторная работа №7**

**Интеграционное тестирование**

**1 Цель работы**

1.1 Интеграционное тестирование программного кода при помощи библиотеки xUnit и Moq;

Репозиторий: <https://github.com/kkerra/UnitTesting>

**2 Контрольные вопросы**

2.1 Какой метод используется для настройки зависимостей подменяемых Moq?

Для настройки зависимостей в Moq используется метод Setup(). С помощью этого метода можно указать, какое поведение нужно симулировать для определённого метода подменяемого объекта (например, установить ожидаемое возвращаемое значение).

2.2 Какой метод используется для проверки выполнялись ли какие-либо действия в зависимостях подменяемых Moq?

Для проверки выполнения действий используются методы Verify() и VerifyAll(). Метод Verify() позволяет удостовериться, что определённый метод был вызван с заданными параметрами и заданным количеством вызовов.

2.3 Что такое интеграционное тестирование?

Интеграционное тестирование — это этап тестирования программного обеспечения, на котором проверяется взаимодействие и интеграция компонентов системы. В отличие от модульного тестирования, которое фокусируется на отдельных модулях или компонентах приложения в изоляции, интеграционное тестирование проверяет, как эти модули работают вместе.

**3 Выводы**

3.1 Протестировали программный код при помощи библиотеки xUnit и Moq;

# **Практическая работа №2**

**Изучение процесса формирования набора тестовых данных**

**1 Цель работы**

1.1 Изучить основные методы и техники формирования тестовых данных для тестирования программного обеспечения;

1.2 Научиться генерировать наборы тестовых данных для различных типов тестирования.

**2 Задания**

**3 Контрольные вопросы**

3.1 Что такое набор тестовых данных?

Тестовый набор данных — это набор данных, который независим от обучающего набора данных, но который соответствует такому же распределению вероятностей, как и обучающий набор данных.

3.2 Что такое граничные значения?

Граничное значение - это минимальное (или максимальное) значение, которое находится на границе.

3.3 Что такое классы эквивалентности?

**Классы эквивалентности — это набор входных (или выходных) данных программного обеспечения, которые обрабатываются программой по одному алгоритму или приводят к одному результату.**

# **Практическая работа №3**

**Деление классов тестов по видам, типам и областям.**

1. **Цель работы**
   1. Научиться разделять тесты на различные виды и категории, а также понимать их назначение и особенности.
2. **Классификация тестов**

|  |  |
| --- | --- |
| Вид теста и его описание | Пример |
| По уровню тестирования | |
| Модульное | Тестирование отдельного модуля, программы |
| Интеграционное | Тестирование взаимодействия между модулями |
| Системное | Полное тестирование всей системы |
| Приемочное | Тестирование на соответствие требованиям, приближенным к реальным |
| По видам тестирования | |
| Функциональное | Проверка функционала приложения |
| Нефункциональное | Тестирование производительности или безопасности |
| Регрессионное | Проверка того, что изменения не привели к сбоям |
| Стресс-тестирование | Проверка системы под нагрузкой, превышающей нормальные условия экспуатации |
| Совместимости | Проверка работы приложения на разных устройствах и браузерах |
| По областям | |
| Пользовательского интерфейса | Проверка удобства и логики взаимодействия пользователя с интерфейсом |
| Безопасности | Проверка уязвимостей |
| Производительности | Измерение времени отклика и нагрузки на приложение |
| Локализации | Проверка корректности перевода и адаптации интерфейса |
| Тестирование веб-приложений | Проверка работоспособности и безопасности веб-сайтов |
| Тестирование мобильных приложений | Проверка функциональности и производительности мобильных устройств |

1. **Выводы**
   1. Научились разделять тесты на различные виды и категории, а также понимать их назначение и особенности.

**Практическая работа №4**

**Инспектирование кода программы.**

1. **Цель работы**
   1. Изучить методы инспектирования кода программы и научиться применять их на практике для повышения качества программного обеспечения.
2. **Контрольные вопросы**
   1. Какие основные техники инспектирования кода существуют?

Существуют следующие основные техники инспектирования кода:

**Формальная инспекция кода** — формализированная процедура просмотра кода.

**Неформальная инспекция кода** (анализ кода) — не имеет чётких правил.

**Чтение кода** — самостоятельное изучение разработчиком чужого кода без присутствия автора.

**Парное программирование** — обзор, осуществляемый постоянно: два разработчика за одним компьютером вместе решают одну задачу.

* 1. Каковы основные критерии при инспектировании кода?

При инспектировании кода можно использовать следующие критерии:

**Соответствие лучшим практикам.**

**Правильность форматирования кода.**

**Актуальность проводимых тестов и версий.**

**Реализация командой правил и политик написания кода.**

* 1. Какие ошибки можно предотвратить при помощи инспектирования кода?

**При помощи инспектирования кода можно предотвратить следующие ошибки:**

Неправильное понимание задач.

Неправильное решение задач.

Неправильный перенос решений в код.

1. **Выводы**
   1. Изучены методы инспектирования кода программы и научиться применять их на практике для повышения качества программного обеспечения.

**Лабораторная работа №8**

**Тестирование ПО методом «черного ящика»**

1. **Цель работы**
   1. Освоить процесс тестирования методом «черного ящика»
2. **Наборы тестов приложений**

|  |  |  |
| --- | --- | --- |
| Действия | Ожидаемый результат | Полученный результат |
| При прохождении теста, не выбирать ответ | Не перелистывает следующую страницу с вопросом | Получилось пролистнуть весь тест не ответив не на один вопрос |
| Создать тесты с одинаковыми именами | Ошибка | Перезаписывает тест |
| В редакторе тестов выбрать тип вопроса «Один ответ» | Выбрать один вариант ответа | Можно выбрать все варианты |
| Изменение числа баллов за ответ | Разбалловка начинается от нуля | Можно выбрать отрицательные числа |
| Изменение типа вопроса во время редактирования вопроса | Изменение типа вопроса с соответствующим вариантом выбора | Тип не изменяется, для его изменения требуется пересоздавать вопрос |
| Видоизменение шрифта | Применение шрифта без изменений остального текста | Сбиваются абзацы |
| Снятие полужирного выделения | Убрать с выделенного текста полужирный | В некоторых лекциях в содержании не убирается полужирный |
| Удаление всех лекций | Продолжение работы приложения | Вылет приложения |
| Выбор открытого вопроса, задать таймер для ответа на вопроса | Функция таймера для ответа на вопрос | Данной функции нет |
| Создание вопроса после удаления всех вопросов в тесте | Свободное редактирование | Ошибка о выходе индекса из диапазона, тест невозможно больше редактировать и восстановить |
| Удаление несохраненного/несуществующего теста | Продолжение использования редактора | Вылет приложения |
| Изменение значения максимального балла за вопрос с помощью клавиатуры | Запрет изменения значения на большее, чем количество вопросов |  |
| Начисление баллов в редактировании вопроса | Интуитивно понятный интерфейс для начисления баллов | Не понятно, как начисляются баллы за вопрос или ответ |

1. **Контрольные вопросы**
   1. Что такое тестирование черного ящика?

Тестирование по стратегии чёрного ящика, часто называемое функциональным тестированием — это методика, изучающая функциональность ПО без необходимости знания внутренней структуры кода.

* 1. Какие преимущества и недостатки у тестирования черного ящика?

**Преимущества тестирования «чёрного ящика»:**

1. Нет необходимости анализировать внутреннюю организацию ПО.
2. Тестирование ориентировано на пользователя.
3. Раннее обнаружение проблем с интерфейсом.
4. Эффективное интеграционное тестирование.
5. Гибкость при разработке тест-кейсов.
6. Эффективность при проверке требований.
7. Подходит для крупных проектов.
8. Доступная автоматизация.
9. Простота масштабируемости.

**Недостатки тестирования «чёрного ящика»:**

1. Ограниченный охват кода.
2. Неспособность тестировать сложные алгоритмы.
3. Избыточное тестирование.
4. Неэффективность при выполнении повторяющихся задач.
5. Отсутствие возможности полноценно оценить производительность и масштабируемость.
6. Сложность локализации ошибок.
7. Ограничения при тестировании безопасности
   1. Какие типы ошибок чаще всего обнаруживаются при тестировании черного ящика?

Тестирование черного ящика обеспечивает поиск следующих категорий ошибок:

некорректных или отсутствующих функций

ошибок интерфейса

ошибок во внешних структурах данных или в доступе к внешней базе данных

ошибок характеристик аппаратных устройств

ошибок инициализации и завершения

1. **Выводы**
   1. Освоен процесс тестирования методом «черного ящика»