Базовые техники тест дизайна

Классы эквивалентности

Разбиение на эквивалентные классы — это метод тест-дизайна, который позволяет уменьшить количество тестовых случаев, необходимых для эффективного тестирования приложения, без потери качества. Суть метода заключается в разделении всех возможных входных данных на несколько групп (классов), которые предполагают одинаковое поведение системы. Предполагается, что если один из представителей класса работает корректно, то и остальные будут работать так же.

Шаги применения метода:

1 Анализ спецификации: изучение требований и определение всех возможных входных данных.
2 Идентификация классов эквивалентности:
2.1 Допустимые классы: значения, которые система должна принимать и обрабатывать.
2.2 Недопустимые классы: значения, которые система должна отвергать или обрабатывать особым образом.
3 Разработка тестовых случаев: выбор представителя из каждого класса для тестирования.

Анализ граничных значений

Анализ граничных значений **(Boundary Value Analysis, BVA)** — это одна из основных техник тест-дизайна, используемая для определения тестовых случаев, направленных на проверку поведения системы на границах допустимых и недопустимых значений входных данных. Основная идея этой техники заключается в том, что ошибки чаще всего возникают именно на границах диапазонов значений, а не внутри них.

Принципы анализа граничных значений
1 Идентификация границ диапазонов:
Определите минимальные и максимальные значения для каждого входного параметра или условия.
Выявите все переходные точки, где изменяется поведение системы.
2 Выбор граничных значений:
2.1 Нижняя граница:
Минимальное допустимое значение **(min)**.
Значение сразу ниже минимального **(min - 1)**.
Значение сразу выше минимального **(min + 1)**.
2.2 Верхняя граница:
Максимальное допустимое значение **(max)**.
Значение сразу ниже максимального **(max - 1)**.
Значение сразу выше максимального **(max + 1)**.
3 Разработка тестовых случаев:
Создайте тесты, использующие выбранные граничные значения.
Проверьте, как система обрабатывает значения на границах и за их пределами.

Таблица решений (Decision Table)

Таблица решений представляет собой табличную форму представления логики принятия решений, позволяя систематизировать и визуализировать различные комбинации входных данных и соответствующих им ожидаемых результатов. Это особенно полезно при тестировании сложных систем с множеством условий и возможных исходов.

Что такое Таблица решений?
Таблица решений представляет собой матрицу, где строки соответствуют условиям и действиям, а столбцы — правилам или комбинациям этих условий. Она визуально отображает, какие действия должны быть предприняты при определенных наборах условий.

Когда использовать Таблицу решений?
1 **Сложная логика системы:** Когда система имеет множество условий, которые могут комбинироваться различными способами.
2 **Необходимость полного покрытия:** Чтобы убедиться, что все возможные комбинации условий протестированы.
3 **Выявление пропусков и конфликтов:** Помогает обнаружить противоречивые или отсутствующие бизнес-правила.

Таблица решений (Decision Table)

Как составить Таблицу решений:

1 Идентификация условий:
Составьте полный список всех условий, влияющих на принятие решений.
Условия обычно выражаются в бинарной форме: Да/Нет или Истина/Ложь.

2 Определение действий:
Определите все возможные действия или результаты, которые может выполнить система.

3 Построение матрицы:
Создайте таблицу, где строки — это условия и действия, а столбцы — комбинации условий (правила).

4 Заполнение комбинаций:
Перечислите все возможные комбинации условий.
Для каждого правила укажите, какие действия должны быть выполнены.

5 Оптимизация:
Удалите дублирующие или несовместимые правила.
Объедините похожие правила для упрощения таблицы.

Пример таблицы решений

	1	2	3	4	5	6	7	8	9
C1: Треугольник?	Y								N
C2: a=b?	Y				N				-
C3: a=c?	Y		N		Y		N		-
C4: b=c?	Y	N	Y	N	Y	N	Y	N	-
A1: Не треугольник									X
A2: Равносторонний	X								
A3: Равнобедренный				X		X	X		
A4: Разносторонний								X	
A5: Невозможно		X	X		X				

Описание приложения:
Вы работаете над тестированием приложения "Калькулятор налогов", которое рассчитывает сумму налога на доход физического лица в зависимости от введенного годового дохода. Правила расчета следующие:

Доход ≤ 100 000 рублей: налоговая ставка **10%.**
100 001 – 500 000 рублей: налоговая ставка **15%.**
500 001 – 1 000 000 рублей: налоговая ставка **20%.**
> 1 000 000 рублей: налоговая ставка **25%.**

Приложение должно также проверять корректность введенных данных и отображать соответствующие сообщения об ошибках в следующих случаях:
Отрицательное значение дохода
Нenumeric значение **(**например**,** текстовые символы**)**
Пустое поле ввода.

Требуется выполнить следующие задачи:

Разбиение на эквивалентные классы:
a) Определите эквивалентные классы для ввода годового дохода, включая как допустимые, так и недопустимые значения.
b) Определите эквивалентные классы для валидации ввода данных.

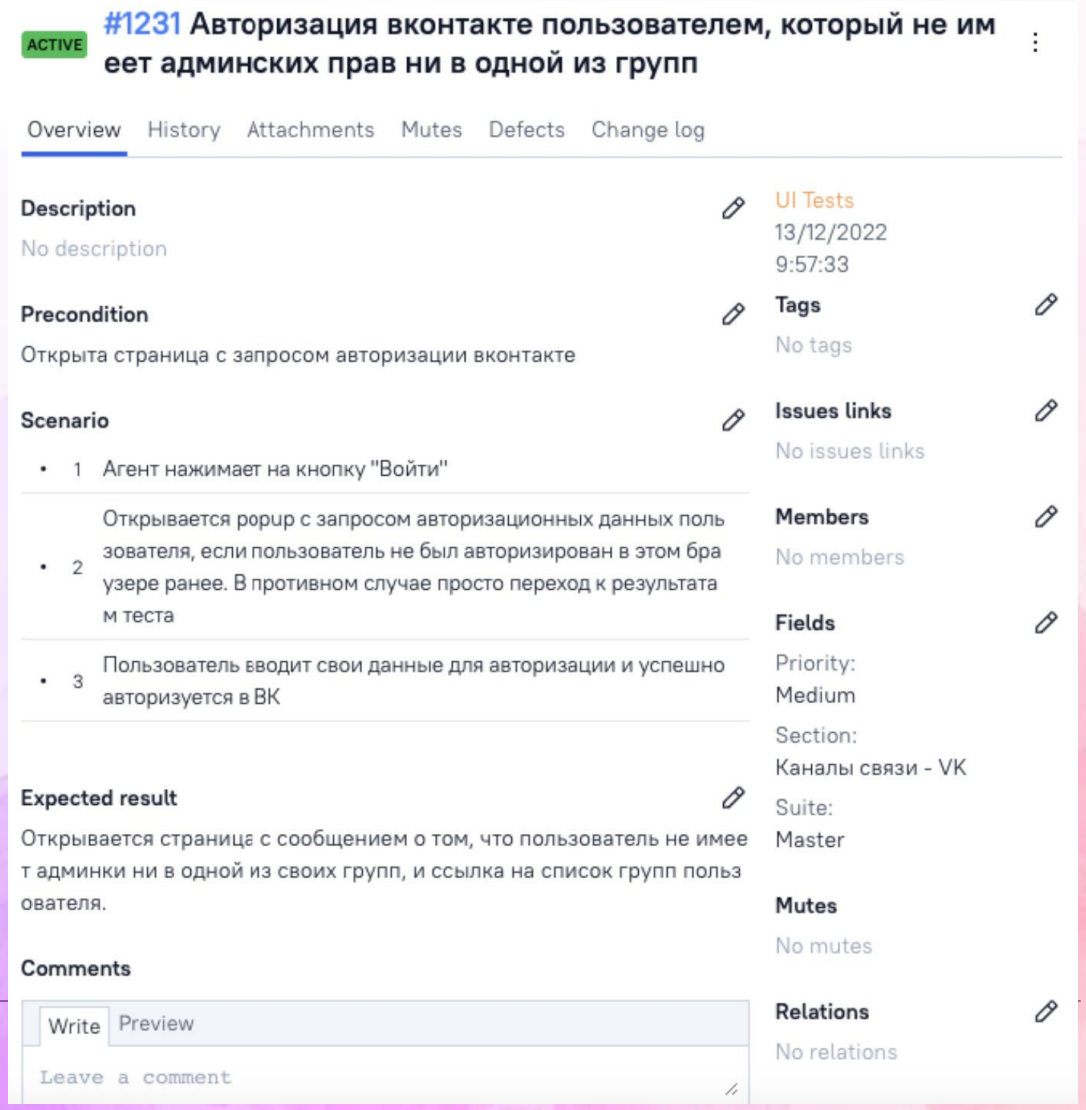
Анализ граничных значений:
a) Определите граничные значения для каждого диапазона доходов.
b) Разработайте тестовые случаи, проверяющие эти граничные значения.

Таблица решений (Decision Table):
a) Составьте таблицу, отображающую все возможные комбинации входных данных и ожидаемых результатов.

Разработка тесткейсов

На основе полученных эквивалентных классов и граничных значений разработайте набор тестовых случаев, указав для каждого:

1 **Номер тестового случая.**
2 **Цель теста.**
3 **Предусловия.**
4 **Входные данные.**
5 **Шаги выполнения.**
6 **Ожидаемый результат.**



```python
import locale
locale.setlocale(locale.LC_ALL, '')

def calculate_tax(income):
    if income <= 100_000:
        tax_rate = 0.10
    elif 100_001 <= income <= 500_000:
        tax_rate = 0.15
    elif 500_001 <= income <= 1_000_000:
        tax_rate = 0.20
    else:
        tax_rate = 0.25
    tax_amount = income * tax_rate
    return tax_amount

def main():
    print("Калькулятор налога на доход физического лица")
    try:
        income_str = input("Введите годовой доход в рублях: ")
        income = locale.atof(income_str.replace(' ', ''))
        if income < 0:
            print("Доход не может быть отрицательным.")
        else:
            tax = calculate_tax(income)
            print(f"Сумма налога: {tax:,.2f} рублей")
    except ValueError:
        print("Ошибка ввода. Пожалуйста, введите числовое значение.")

if __name__ == "__main__":
    main()
```

```python
# Тесты с использованием pytest
@pytest.mark.parametrize("income, expected_tax", [
    (50_000, 5_000.0),
    (150_000, 22_500.0),
    (600_000, 120_000.0),
    (1_500_000, 375_000.0),
    (0, 0.0),
    (-50_000, -5_000.0),
    (-1, -0.10)
])
def test_calculate_tax(income, expected_tax):
    assert calculate_tax(income) == expected_tax


if __name__ == "__main__":
    pytest.main()
```