Министерство образования Самарской области

АССОЦИАЦИЯ ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ "ИНСТИТУТ НАПРАВО.РУ"

УТВЕРЖДЕНО
Приказ директора
от 07.08.2024 № 012/2/с

Воробьев К.А.

УЧЕБНО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ПМ.03 Тестирование информационных систем

Специальность 09.02.10 Разработка компьютерных игр, дополненной и виртуальной реальности
Квалификация: разработчик компьютерных игр, дополненной и виртуальной реальности
Форма обучения: очная
На базе основного общего образования
Срок получения образования: 3 года 10 месяцев

Тольятти, 2024г.

Содержание

1. Назначение и организация практикума 3

2. Состав работ и представление результатов 3

3. Практические работы 4

Работа 1. Классы эквивалентности и границы 4

Работа 2. Таблица решений 4

Работа 3. Отчёт о дефекте 5

Работа 4. Регрессионный набор 6

Работа 5. Статический анализ фрагмента кода 6

Работа 6. Проверка производительности 7

Развёрнутая лабораторная работа. Автоматические и ручные проверки модуля урона 7

4. Комплексное учебное задание 9

5. Термины и понятия 9

6. Исходные данные и учебные примеры 9

7. Использованные источники 10

1. Назначение и организация практикума

Планирование, проектирование, выполнение и документирование проверок игрового программного обеспечения.

Указания предназначены для обучающихся по специальности 09.02.10 «Разработка компьютерных игр, дополненной и виртуальной реальности», квалификация — разработчик компьютерных игр, дополненной и виртуальной реальности.

Компонент	Учебная нагрузка
Индекс и наименование	ПМ.03 Тестирование информационных систем
Общий объём	266 академических часов
Практические занятия	100 академических часов
Промежуточная аттестация	Экзамен — семестр 8; Дифференцированный зачёт — семестр 8

Перед работой изучаются её исходные данные и проверяемые результаты. Затем выполняются последовательные действия, сопоставляется учебный пример и решаются самостоятельные варианты. Ответ включает ход работы и подтверждение результата. Использование готового примера без выполнения своего варианта не подтверждает освоение действия.

Подготовленные материалы: План и тестовые данные; Тест-кейсы, дефекты и автоматические проверки; Итоговый отчёт с ограничениями.

2. Состав работ и представление результатов

Работа	Тема и результат
1	Классы эквивалентности и границы. Ожидаемый результат не выводится из текущего поведения ошибочной программы.
2	Таблица решений. Побочные изменения проверены вместе с сообщением.
3	Отчёт о дефекте. Другой участник повторяет дефект.
4	Регрессионный набор. Набор связан с изменением.
5	Статический анализ фрагмента кода. Приведение типа выполняется до деления.
6	Проверка производительности. Сырые данные сохранены.

Профессиональные результаты модуля

Код	Профессиональная компетенция
ПК 3.1	Осуществлять подготовку к проведению тестирования кода или информационной системы.
ПК 3.2	Выполнять процесс измерения характеристик компонент программного продукта для определения соответствия заданным критериям.
ПК 3.3	Производить исследование созданного программного кода с использованием специализированных программных средств с целью выявления ошибок и отклонения от алгоритма.
ПК 3.4	Осуществлять тестирование информационной системы на этапе опытной эксплуатации с фиксацией выявленных ошибок в разрабатываемых модулях информационной системы.

В отчёте указываются тема, вариант, исходные данные, ход работы, результаты проверки и вывод. Оценивание проводится по критериям фонда оценочных средств. Приложите тестовые сценарии, протокол и файлы автоматических тестов.

3. Практические работы

Работа 1. Классы эквивалентности и границы

Краткие теоретические сведения

Тесты выбирают по правилам области ввода. Номинальный успешный случай дополняется граничными и недопустимыми значениями.

Исходные данные

Имя игрока содержит от 3 до 12 символов; разрешены латинские буквы и цифры; пробелы запрещены.

Порядок выполнения

1. Выделить допустимые и недопустимые классы.

2. Выбрать длины 2,3,12,13.

3. Добавить пустое значение, пробел и запрещённый символ.

4. Для каждого случая заранее записать ожидаемый результат.

Разбор учебного примера

Имена длиной 3 и 12 с допустимыми символами принимаются; 2 и 13 отклоняются. Строка из трёх пробелов не является допустимым именем.

Задания для самостоятельного выполнения

А: разрешить внутренний пробел.

Б: диапазон длины изменить на 1–20.

Проверка результата

— Ожидаемый результат не выводится из текущего поведения ошибочной программы.

— Изменение требования меняет набор проверок.

Вопросы для самоконтроля

1. Почему один пример не покрывает весь класс без предпосылок?

2. Какие проверки связаны с разными правилами?

Работа 2. Таблица решений

Краткие теоретические сведения

Таблица решений проверяет сочетания условий и позволяет увидеть пропущенное правило. Для каждого столбца задают результат до выполнения.

Исходные данные

Покупка разрешена, если предмет доступен, средств достаточно и инвентарь не заполнен.

Порядок выполнения

1. Перечислить восемь сочетаний трёх условий.

2. Указать разрешение или отказ.

3. Добавить приоритет сообщения при нескольких причинах отказа.

4. Проверить отсутствие списания при отказе.

Разбор учебного примера

Разрешена только комбинация true,true,false для условий «доступен, достаточно, заполнен». В остальных случаях баланс и инвентарь неизменны.

Задания для самостоятельного выполнения

А: предмет бесплатный.

Б: инвентарь имеет отдельный слот для ключей.

Проверка результата

— Побочные изменения проверены вместе с сообщением.

— Нет неописанного сочетания.

Вопросы для самоконтроля

1. Чем таблица решений отличается от списка экранов?

2. Почему надо проверять состояние после отказа?

Работа 3. Отчёт о дефекте

Краткие теоретические сведения

Дефект воспроизводится через исходное состояние и действия. Серьёзность последствий и срочность исправления описывают разные характеристики.

Исходные данные

При закрытии игры во время сохранения следующий запуск завершает загрузку ошибкой.

Порядок выполнения

1. Создать копию учебных данных.

2. Описать условия и шаги.

3. Записать ожидаемый и фактический результаты.

4. Приложить журнал и указать область проверенного воспроизведения.

Разбор учебного примера

Название: повреждённое сохранение препятствует запуску. Не следует заявлять «все данные теряются всегда», если выполнена одна конфигурация.

Задания для самостоятельного выполнения

А: ошибка повторяется только на пустом профиле.

Б: временно недоступна запись в папку.

Проверка результата

— Другой участник повторяет дефект.

— В отчёте нет паролей и реальных персональных данных.

Вопросы для самоконтроля

1. Чем воспроизводимость отличается от частоты?

2. Как определить полезное вложение к отчёту?

Работа 4. Регрессионный набор

Краткие теоретические сведения

После исправления проверяют исходный дефект и связанные функции. Повтор только успешного сценария не подтверждает отсутствие побочных нарушений.

Исходные данные

Исправлено сохранение громкости; тот же файл хранит уровень и раскладку управления.

Порядок выполнения

1. Выделить затронутые поля и операции.

2. Составить минимальный регрессионный набор.

3. Выполнить перезапуск и загрузку старой записи.

4. Зафиксировать статусы passed, failed, blocked.

Разбор учебного примера

Проверяются громкость, уровень, управление, отсутствующий и повреждённый файл. Blocked означает невозможность проверки, а не положительный результат.

Задания для самостоятельного выполнения

А: формат сохранения изменён.

Б: обязательный тест недоступен из-за отсутствия устройства.

Проверка результата

— Набор связан с изменением.

— Итог не скрывает заблокированные проверки.

Вопросы для самоконтроля

1. Как обосновать выбор регрессии?

2. Когда нужен полный повтор набора?

Работа 5. Статический анализ фрагмента кода

Краткие теоретические сведения

Проверка кода без запуска выявляет нарушения контрактов и опасные предположения. Замечание должно содержать место, риск и предложение проверки.

Исходные данные

Фрагмент: int Average(int sum,int count){ return sum/count; }. Требуется среднее как вещественное число.

Порядок выполнения

1. Определить два возможных дефекта.

2. Сформулировать контракт для count.

3. Предложить исправленный вариант.

4. Подготовить тесты для count=0 и дробного результата.

Разбор учебного примера

Проблемы: деление на ноль и целочисленное деление. При sum=5,count=2 ожидается 2,5;

при count=0 нужен определённый отказ, а не случайный сбой.

Задания для самостоятельного выполнения

А: count отрицателен.

Б: результат округляется по отдельному требованию.

Проверка результата

— Приведение типа выполняется до деления.

— Отказ описан в контракте и проверен.

Вопросы для самоконтроля

1. Почему cast после деления не восстанавливает дробную часть?

2. Как отличить ошибку реализации от неясного требования?

Работа 6. Проверка производительности

Краткие теоретические сведения

Измерение задаёт нагрузку, устройство, прогрев, повторения и метрику. Среднее время кадра не описывает все задержки; нужно сохранить исходные измерения.

Исходные данные

Время пяти кадров: 10,10,10,10,60 мс.

Порядок выполнения

1. Вычислить среднее и медиану.

2. Найти максимальную задержку.

3. Сравнить с набором 20,20,20,20,20.

4. Объяснить, что скрывает одна средняя величина.

Разбор учебного примера

У обоих наборов среднее 20 мс. В первом медиана 10 и максимум 60, во втором медиана и максимум 20. Ощущение плавности не следует считать одинаковым только по среднему.

Задания для самостоятельного выполнения

А: добавить 100 одинаковых быстрых кадров.

Б: сравнить измерения до и после исправления на одной сцене.

Проверка результата

— Сырые данные сохранены.

— Одинаковые условия измерения указаны явно.

Вопросы для самоконтроля

1. Почему максимум и среднее отвечают на разные вопросы?

2. Как избежать сравнения разных сцен?

Развёрнутая лабораторная работа. Автоматические и ручные проверки модуля урона

Средства выполнения

Python с модулем unittest из стандартной библиотеки, файл test_damage.py. Ручные проверки документируются в таблице.

Задание

Проверить контракт damage и доказать, что набор обнаруживает хотя бы один намеренно внесённый дефект. Сопоставить результаты ручных расчётов и автоматического запуска.

Последовательность выполнения

1. Сохраните файл test_damage.py и выполните python -m unittest -v test_damage. В исходной версии должны пройти три тестовых метода. Сохраните вывод как исходный протокол проверки.

2. Перепишите контракт функции своими словами: оба аргумента неотрицательны; результат — неотрицательная разность; при отрицательном аргументе ожидается ValueError. Отдельно укажите область типов данных, рассматриваемую в учебной работе.

3. Составьте классы входов: атака больше защиты, равна защите, меньше защиты; обе величины нулевые; отрицательна атака; отрицательна защита. Для каждого выберите конкретную пару и рассчитайте ожидаемый результат вручную.

4. Добавьте отдельные проверки равенства аргументов, отрицательной защиты и большого допустимого значения. Избегайте одного огромного теста, в котором ранний сбой скрывает остальные случаи.

5. В учебной копии временно замените max(0,attack-defense) на attack-defense. Повторите запуск. Случай (5,12) должен показать нарушение контракта. Зафиксируйте диагностическое сообщение, затем верните правильную реализацию.

6. В другой учебной копии удалите проверку отрицательных аргументов. Подтвердите, что тест ожидаемого исключения обнаруживает этот дефект. После проверки восстановите исходное условие.

7. Подготовьте один ручной тест интеграции с интерфейсом: ввод пары, запуск расчёта, показ результата и обработка ошибки. Разделите проверку математической функции и корректность сообщения пользователю.

8. Оформите отчёт о каждом внесённом учебном дефекте: исходная версия, изменение, воспроизведение, ожидаемое и фактическое поведение, доказательство обнаружения. Не выдавайте внесённый дефект за неизвестную ошибку продукта.

9. После восстановления реализации выполните полный набор. Все тесты должны пройти; оставленные ожидаемые падения не подменяют работоспособную итоговую версию.

10. Подготовьте заключение о покрытии: перечислите проверенные классы и то, что не проверялось, например интерфейс вне одного сценария, другой тип данных или производительность. Число passed не является доказательством отсутствия всех дефектов.

Контрольные сценарии и ожидаемые результаты

Сценарий	Ожидаемый результат
(12,5)	7
(5,12)	0
(5,5)	0
(0,0)	0
(−1,0)	ValueError
(1,−1)	ValueError
Учебная мутация без max	Набор должен обнаружить дефект

Код	Профессиональная компетенция
Восстановленная версия	Все проверки завершаются успешно

Усложнённый вариант

Подготовьте аналогичный набор для удаления предмета из инвентаря: используйте отдельное состояние для каждого теста, проверьте частичное и полное удаление, отказ при нехватке и неизменность другого предмета. Обоснуйте, какие проверки являются модульными, а какие интеграционными.

4. Комплексное учебное задание

Подготовьте план тестирования игрового инвентаря, 12 тест-кейсов, таблицу решений, три учебных отчёта о дефектах, регрессионный набор и итоговое заключение. Реализуйте автоматическую проверку хотя бы одной чистой функции и покажите успешный и неуспешный запуск.

Этапы и предъявляемые материалы

Этап	Результат
1	План и тестовые данные
2	Тест-кейсы, дефекты и автоматические проверки
3	Итоговый отчёт с ограничениями

Выполненная работа представляется с исходными данными, объяснением принятых решений и проверками. Изменения исходного условия согласуются до выполнения расчётов или реализации. При защите обучающийся воспроизводит один основной и один граничный либо отрицательный сценарий и объясняет полученный результат.

Контроль готовности

— Ожидаемый результат не выводится из текущего поведения ошибочной программы.

— Побочные изменения проверены вместе с сообщением.

— Другой участник повторяет дефект.

— Набор связан с изменением.

— Приведение типа выполняется до деления.

— Сырые данные сохранены.

5. Термины и понятия

Термин	Содержание
Тест-кейс	Условия, действия и ожидаемый результат проверки.
Дефект	Несоответствие установленному требованию.
Регрессия	Повторная проверка ранее работавших функций после изменения.
Покрытие	Отношение выполненных проверок к выбранному основанию.

6. Исходные данные и учебные примеры

Пример содержит функцию расчёта урона и автоматические тесты для практической работы по тестированию.

test_damage.py

```
import unittest

def damage(attack, defense):
    if attack < 0 or defense < 0:
        raise ValueError("negative argument")
    return max(0, attack-defense)

class DamageTests(unittest.TestCase):
    def test_normal(self):
        self.assertEqual(damage(12,5),7)
    def test_zero(self):
        self.assertEqual(damage(5,12),0)
        self.assertEqual(damage(0,0),0)
    def test_negative(self):
        with self.assertRaises(ValueError):
            damage(-1,0)

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

Выполните тесты для обычного случая, нулевого результата и недопустимого аргумента. Результаты внесите в протокол тестирования.

7. Использованные источники

1. Святослав Куликов. Тестирование программного обеспечения. Базовый курс. (3- е издание). Версия книги 3.2.6 от 24.05.2024. https://svyatoslav.biz/software_testing_book

2. Журнал «Тестирование и качество» для ИТ-специалистов

3. Электронное издание на основе: Плаксин М.А. Тестирование и отладка программ для профессионалов будущих и настоящих [Электронный ресурс] / М. А. Плаксин. -2-е изд. (эл.). -М.: БИНОМ. Лаборатория знаний, 11 2013.-167 с.: ил. ISBN 978-5-9963-0946-7. http://www.studentlibrary.ru/book/ISBN9785996309467.html

4. Электронное издание на основе: ATDD – разработка программного обеспечения через приемочные тесты. Пер. с англ. Слинкин А. А. - М.: ДМК Пресс, 2013. - 232 с.: ил. - ISBN 978-5-94074-572-3. http://www.studentlibrary.ru/book/ISBN9785940745723.html

5. Unity Technologies. Unity User Manual 2022.3. Creating and Using Scripts. URL: https://docs.unity3d.com/2022.3/Documentation/Manual/CreatingAndUsingScripts.html