ПРИЛОЖЕНИЕ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ИМЕНИ. В.Ф. УТКИНА

Кафедра «Вычислительная и прикладная математика»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ «Тестирование программного обеспечения информационных систем»

Направление подготовки

09.03.03 «Прикладная информатика»

Направленность (профиль) подготовки

Прикладная информатика

Квалификация выпускника – бакалавр

Форма обучения – очная, заочная

Рязань 2022 г.

1. ОБЩИЕ ПОЛОЖЕНИЯ

Оценочные материалы – это совокупность учебно-методических материалов и процедур, предназначенных для оценки качества освоения обучающимися данной дисциплины как части основной образовательной программы.

Цель – оценить соответствие знаний, умений и уровня приобретенных компетенций, обучающихся целям и требованиям основной образовательной программы в ходе проведения текущего контроля и промежуточной аттестации.

Основная задача – обеспечить оценку уровня сформированности компетенций, приобретаемых обучающимся в соответствии с этими требованиями.

Контроль знаний обучающихся проводится в форме промежуточной аттестации – зачета в 7-м семестре.

2. ОПИСАНИЕ ПОКАЗАТЕЛЕЙ И КРИТЕРИЕВ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

ПК-1.3, ПК-2.3, ПК-3.3.

Сформированность каждой компетенции в рамках освоения данной дисциплины оценивается по трехуровневой шкале:пороговый уровень является обязательным для всех обучающихся по завершении освоения дисциплины;

1) продвинутый уровень характеризуется превышением минимальных характеристик сформированности компетенций по завершении освоения дисциплины;

2) эталонный уровень характеризуется максимально возможной выраженностью компетенций и является важным качественным ориентиром для самосовершенствования.

Уровень освоения компетенций, формируемых дисциплиной

а) описание критериев и шкалы оценивания тестирования:

Шкала оценивания	**Критерий**
3 балла (эталонный уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 85 до 100%
2 балла (продвинутый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 75 до 84%
1 балл (пороговый уровень)	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 60 до 74%
0 баллов	уровень усвоения материала, предусмотренного программой: процент верных ответов на тестовые вопросы от 0 до 59%

б) описание критериев и шкалы оценивания теоретического вопроса:

Шкала оценивания	**Критерий**
3 балла (эталонный уровень)	выставляется студенту, который дал полный ответ на вопрос, показал глубокие систематизированные знания, смог привести примеры, ответил на дополнительные вопросы преподавателя.
2 балла (продвинутый уровень)	выставляется студенту, который дал полный ответ на вопрос, но на некоторые дополнительные вопросы преподавателя ответил только с помощью наводящих вопросов.
1 балл (пороговый уровень)	выставляется студенту, который дал неполный ответ на вопрос в билете и смог ответить на дополнительные вопросы только с помощью преподавателя.
0 баллов	выставляется студенту, который не смог ответить на вопрос

в) описание критериев и шкалы оценивания практического задания:

Шкала оценивания	**Критерий**
3 балла (эталонный уровень)	Задание решено верно
2 балла (продвинутый уровень)	Задание решено верно, но имеются технические неточности в выполнении
1 балл (пороговый уровень)	Задание решено верно, с дополнительными наводящими вопросами преподавателя
0 баллов	Задание не решено

На зачет выносится: тестовое задание, 1 практическое задание и 1 теоретический вопрос.

Студент может набрать максимум 9 баллов.

Итоговый суммарный балл студента, полученный при прохождении промежуточной аттестации, переводится в традиционную форму по системе «зачтено», «не зачтено».

Оценка «зачтено» выставляется студенту, который набрал в сумме не менее 5 баллов. Обязательным условием является выполнение всех предусмотренных в течение семестра практических заданий и лабораторных работ.

Оценка «не зачтено» выставляется студенту, который набрал в сумме менее 5 баллов, либо имеет к моменту проведения промежуточной аттестации несданные практические, либо лабораторные работы.

3 ПАСПОРТ ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ДИСЦИПЛИНЕ

Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или её части)	Наименование оценочного средства
1. Введение в тестирование	ПК-1.3 ПК-2.3 ПК-3.3	Зачет
2. Тестирование на ранних стадиях разработки. Техники тестирования	ПК-1.3 ПК-2.3 ПК-3.3	Зачет
3. Модульное тестирование. Основы JUnit	ПК-1.3 ПК-2.3 ПК-3.3	Зачет
4. Модульное тестирование. Параметризованные тесты	ПК-1.3 ПК-2.3 ПК-3.3	Зачет
5. Автоматизация тестирования. Основы	ПК-1.3 ПК-2.3 ПК-3.3	Зачет
6. Selenium WebDriver. Поиск элементов	ПК-1.3 ПК-2.3 ПК-3.3	Зачет
7. Нагрузочное тестирование. Введение	ПК-1.3 ПК-2.3 ПК-3.3	Зачет
8. Нагрузочное тестирование. использование JMeter	ПК-1.3 ПК-2.3 ПК-3.3	Зачет

4 ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ

4.1 Промежуточная аттестация (зачет и экзамен)

ПК-1: Способен разрабатывать требования, проектировать и выполнять программную реализацию программного обеспечения
ПК-1.3. Проектирует программное обеспечение и выполняет его программную реализацию

а) типовые тестовые вопросы закрытого типа

1. Укажите вариант с правильной последовательностью и перечнем этапов действий в рамках юнит-теста:

   Подготовка, действие, проверка.

   Действие, выполнение.

   Выполнение функции, оценка результата
2. Какой класс Java служит для проверки соответствие наблюдаемого результата ожидаемому:
   a. **Assert.**
   b. Dessert.
   c. FindElement.
   d. FindElements.
3. Требуется ли сопровождение разработанным тестам?
   a. **Да.**
   b. Нет.
   c. В зависимости от тестируемого приложения.
4. Укажите источники информации для валидации ПО при его системном тестировании:
   a. **Спецификация требований.**
   b. **Предметная область.**
   c. **Стандарты.**
   d. Проектная документация
5. Спецификация требований проверяется тестировщиком на предмет:
   a. **Тестируемости каждого требования.**
   b. Возможности реализации в коде.
   c. Осуществимости.
   d. Соответствия программному коду.
6. Чек-лист представляет собой:
   a. **Перечень проверок, выполняемых в процессе тестирования ПО.**
   b. Набор тест-кейсов для тестирования ПО.
   c. Сценарии тестирования ПО.
   d. Набор тестовых данных.
7. Полнота спецификации требований к ПО определяется:
   a. Полнотой проектной документации.
   b. **Степенью охвата спецификацией требований всех необходимых для разработки ПО сведений предметной области, соответствующих стандартов и сведений, полученных от заинтересованных лиц и пользователей ПО.**
   c. Степенью соответствия предметной области.
   d. Полнотой плана тестирования.
8. Тест-кейс – это

a. **Набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения ПО.**
b. Набор выходных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения ПО.
c. Набор входных и выходных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения ПО.
d. Набор тестовых данных, покрывающих все возможные варианты работы ПО.

9. Тест кейсы по уровням классифицируются на:
    a. **Высокоуровневые и низкоуровневые.**
    b. Модульные, интеграционные и системные.
    c. Позитивные и негативные.
    d. Одноуровневые и многоуровневые.
10. Базовый набор состояний тест-кейса включает в себя следующие состояния:
    a. **Создан, запланирован, выполняется, пройден успешно, закрыт.**
    b. Создан, не выполнен, пропущен, закрыт.
    c. Создан, запланирован, пропущен, закрыт.
    d. Создан, не выполнен, провален, закрыт.

б) типовые тестовые вопросы открытого типа

1. Что является объектом юнит-тестирования? (Классы и их составляющие)
2. Какие инструменты применимы для разработки юнит-тестов в C#? (NUnit)
3. Какие инструменты применимы для разработки юнит-тестов в Java? (JUnit)
4. Какие инструменты применимы для разработки юнит-тестов в JavaScript? (Mocha)
5. Что помечает аннотация @Test в юнит-тестировании на Java? (Тест)
6. Что помечает аннотация @After в юнит-тестировании на Java? (Действие после выполнения теста)
7. Что помечает аннотация @Before в юнит-тестировании на Java? (Действие перед выполнением теста)
8. Что помечает аннотация @BeforeAll в юнит-тестировании на Java? (Действие перед выполнением всех тестов)
9. Что помечает аннотация @AfterAll в юнит-тестировании на Java? (Действие после выполнения всех тестов)
10. Для чего нужна аннотация @Tag в юнит-тестировании на Java? (Помечать тесты)?

ПК-2: Способен выполнять проектирование информационных систем среднего и крупного масштаба сложности
ПК-2.3. Сопровождает приемочные испытания и ввод в эксплуатацию информационной системы

а) типовые тестовые вопросы закрытого типа

1. Приемосдаточное тестирование имеет целью проверку работы ПО:
    a. **С точки зрения конечного пользователя.**
    b. С точки зрения программно-аппаратного окружения.
    c. С точки зрения заказчика.
    d. С точки зрения всех заинтересованных в разработке лиц.
2. Суть сопровождения приемочных испытаний заключается в:
    a. **Поддержке процесса выполнения тестов и тестовой инфраструктуры.**

b. Выполнении тестов в рамках приемочных испытаний.
c. Подготовке тестовых данных и выполнении приемочных испытаний ПО.
d. Сопровождении инфраструктуры проведения тестовых испытаний.

3. Подвидами приемочного тестирования являются:
   a. **Производственное приемочное тестирование.**
   b. **Операционное приемочное тестирование.**
   c. **Итоговое приемочное тестирование.**
4. Производственное приемочное тестирование выполняется:
   a. **Проектной командой.**
   b. Заказчиком.
   c. Сторонними лицами.
   d. Конечными пользователями.
5. Цель производственного приемочного тестирования ПО заключается в:
   a. **Определении готовности ПО к передаче заказчику.**
   b. Определении его работоспособности.
   c. Оценке качества разработанного ПО.
   d. Обнаружении дефектов в базовой функциональности ПО.
6. Операционное приемочное тестирование направлено на тестирование:
   a. **Инсталляции ПО, потребления ресурсов, совместимости ПО с программно-аппаратной платформой.**
   b. Использования ПО ОЗУ.
   c. Использования ПО ресурсов вычислительных средств.
   d. Использование ПО ресурсов процессора.
7. Итоговое приемочное тестирование ПО осуществляется:
   a. **Представителями заказчика в реальных условиях эксплуатации.**
   b. Командой тестирования разработчика.
   c. Экспертной комиссией, сформированной разработчиком.
   d. Профессиональными тестировщиками.
8. Целью итогового приемочного тестирования является определение:
   a. **Готовности ПО к эксплуатации.**
   b. Готовности ПО к сопровождению.
   c. Готовности ПО к опытной эксплуатации.
   d. Соответствия ПО заявленным показателям качества.
9. Приемочное тестирование направлено на:
   a. Выявление дефектов, не позволяющих проводить дальнейшее тестирование ПО.
   b. Выявление дефектов в программно-аппаратной среде выполнения ПО.
   c. Выявление наиболее опасных дефектов.
   d. Проверку работоспособности ПО.
   e. **Нет правильного ответа.**
10. Приемочное тестирование соответствует:
    a. Альфа-тестированию.
    b. **Бета-тестированию.**
    c. Регрессионному тестированию.
    d. Тестированию производительности.

б) типовые тестовые вопросы открытого типа

1. Что является предметом регрессионного тестирования? (выявление регресса в ПО)

2. Кто выполняет бета-тестирование? (пользователи ПО)
3. Кто выполняет альфа-тестирование? (разработчик ПО)
4. Что является дефектом в ПО? (несоответствие наблюдаемого и ожидаемого поведения ПО)
5. Что определяет статус дефекта? (этап его жизненного цикла)
6. Что определяет значимость дефекта? (возможность работы ПО при его наличии)
7. Что такое параметризованные тесты? (один сценарий, много данных)
8. Что выступает в роли параметров в параметризованных тестах? (тестовые данные)
9. Что остается инвариантом в параметризованных тестах? (тестовый сценарий)
10. В чем суть статического тестирования? (анализ кода и документации)

ПК-3: Способен выполнять работы и управление работами по созданию и сопровождению информационных систем
ПК-3.3. Организует и руководит тестированием информационной системы

а) типовые тестовые вопросы закрытого типа

1. План тестирования определяет:
    a. **Сроки, ресурсы, содержание работ по тестированию ПО**
    b. Сроки, ресурсы, стоимость работ по разработке тестов для тестирования ПО
    c. Сроки, ресурсы, стоимость работ по разработке тестов для испытаний ПО
    d. Сроки, объем, стоимость работ по разработке тестов для тестирования ПО
2. Набор тест-кейсов – это:
    a. **Совокупность тест-кейсов, выбранных с некоторой общей целью или по некоторому общему признаку.**
    b. Набор чек-листов, выбранных с некоторой общей целью или по некоторому общему признаку.
    c. Система наборов тестовых данных, выбранных с некоторой общей целью или по некоторому общему признаку.
    d. Набор чек-листов, выбранных с в произвольном порядке для тестирования отдельно взятого функционала ПО.
3. Тест-кейсы по последовательности выполнения бывают:
    a. **Свободные и последовательные.**
    b. Зависимые и независимые.
    c. Атомарные и составные.
    d. Последовательные и параллельные.
4. Тест-кейсы по изолированности друг от друга бывают:
    a. **Изолированные и обобщенные.**
    b. Зависимые и независимые.
    c. Атомарные и составные.
    d. Последовательные и параллельные.
5. Обобщенные тест-кейсы имеют общую стадию:
    a. **Приготовления.**
    b. Выполнения.
    c. Разработки.
    d. Обработки результатов тестирования.
6. Изолированные тест-кейсы характеризуются:
    a. **Независимыми этапами подготовки и выполнения.**
    b. Зависимостью по данным на этапе выполнения.

c. Зависимостью на этапе подготовки.
d. Изолированностью от данных.

7. Обобщенные свободные тест-кейсы характеризуются:
   a. **Общим этапом приготовления и независимостью от порядка выполнения.**
   b. Строго определенным порядком выполнения.
   c. Различными этапами приготовления.
   d. Независимостью от данных.
8. Обобщенные последовательные тест-кейсы характеризуются:
   a. **Общим этапом приготовления и строгим порядком выполнения.**
   b. Индивидуальными этапами приготовления и строгим порядком выполнения.
   c. Общим этапом приготовления и произвольным порядком выполнения.
   d. Индивидуальными этапами приготовления и произвольным порядком выполнения.
9. Разработка тест-кейсов возможна на основе:
   a. **Чек-листов.**
   b. Потребностей заказчиков.
   c. Результатов тестирования.
   d. Результатов разработки программного кода.
10. Эффективные тест-кейсы обеспечивают повышение:
    a. **Качества ПО.**
    b. Программных интерфейсов.
    c. Качества спецификации требований.
    d. Аппаратных интерфейсов.

б) типовые тестовые вопросы открытого типа

1. Что такое тестовые данные? (входные данные для тестирования)
2. На основе какого документа, разработанного аналитиками, возможно создание тест-кейсов? (спецификации требований к ПО)
3. Что повышает автоматизация тестирования с точки зрения процесса? (скорость выполнения тестов)
4. Можно ли автоматизировать тестирование удобства и простоты использования? (нет)
5. На что направлено системное тестирование? (на проверку всего приложения)
6. Что является предметом нагрузочного тестирования ПО? (способность сохранять заданные показатели качества при нагрузке)
7. На что направлена техника тестирования классов эквивалентности? (уменьшение количества тестовых данных)
8. Что содержит отчет о дефекте? (описание дефекта с указанием его приоритета)
9. Какие стадии жизненного цикла имеет отчет о дефекте в случае упрощенного варианта жизненного цикла дефекта? (обнаружен, назначен, исправлен, проверен, закрыт)
10. Что такое воспроизводимость дефекта? (проявление дефекта при воспроизведении шагов сценария)

4. Типовые задания промежуточного контроля

1. Осуществить проверку прилагаемого фрагмента спецификации требований (пункт 9.1).
2. Составить план тестирования к прилагаемому фрагменту спецификации

требований (пункт 9.1).

3. Сформировать чек-лист для тестирования приложения «Текстовый редактор».
4. Воспользовавшись чит-листом, сформировать набор тестовых сценариев для тестирования прилагаемой формы (пункт 9.2).
5. Составить набор тестовых сценариев для тестирования прилагаемой формы (пункт 9.2), воспользовавшись техникой тестирования всех пар.
6. Составить набор тестовых сценариев для тестирования прилагаемой формы (пункт 9.2), воспользовавшись техникой тестирования разбиение на классы эквивалентности.
7. Сформировать набор тестовых сценариев для тестирования любых трех из прилагаемых (пункт 9.1) вариантов использования. Учесть альтернативные и исключительные течения.
8. Сформировать наборы тестовых данных для тестирования любых трех из прилагаемых (пункт 9.1) вариантов использования. Учесть альтернативные и исключительные течения.
9. Осуществить тестирование методов класса (пункт 9.3) с использованием библиотеки модульного тестирования.
10. Осуществить тестирование конструкторов класса (пункт 9.3) с использованием библиотеки модульного тестирования.
11. Разработать набор тестовых сценариев для тестирования методов класса (пункт 9.3) с покрытием по функционалу. Составить отчет по результатам тестирования.
12. Разработать набор тестовых сценариев для тестирования методов класса (пункт 9.3) с покрытием по данным. Тестовый набор разработать как параметризованные тесты.
13. Для класса (пункт 9.3) сформировать набор возможных метрик кода в среде MS Visual Studio. Кратко охарактеризовать полученные значения.
14. Для класса (пункт 9.3) сформировать набор возможных метрик кода в среде Eclipse. Кратко охарактеризовать полученные значения.
15. Для класса (пункт 9.3) сформировать набор возможных метрик кода в среде Idea IntelliJ. Кратко охарактеризовать полученные значения.
16. Настроить инфраструктуру для автоматизированного тестирования с использованием среды MS Visual Studio для актуальных версий браузеров IE, Google Chrome, Mozilla FireFox.
17. Настроить инфраструктуру для автоматизированного тестирования с использованием среды Eclipse для актуальных версий браузеров IE, Google Chrome, Mozilla FireFox.
18. Настроить инфраструктуру для автоматизированного тестирования с использованием среды Idea IntelliJ для актуальных версий браузеров IE, Google Chrome, Mozilla FireFox.
19. Осуществить запуск пробных автоматизированных тестов, написанных на языках C#, Java, Ruby, Python, JavaScript.
20. Продемонстрировать работу с объектом Capabilities (настройки браузера) применительно к актуальным версиям браузеров IE, Google Chrome, Mozilla FireFox.
21. Продемонстрировать работу с Cookies актуальных версий браузеров IE, Google Chrome, Mozilla FireFox.
22. Осуществить запуск десяти браузеров в различных потоках. Язык программирования – Java.
23. Осуществить запуск десяти браузеров в различных потоках. Язык программирования – C#.
24. На основе DOM произвольно интернет-страницы страницы продемонстрировать поиск элементов на основе CSS.

25. На основе DOM произвольно интернет-страницы страницы продемонстрировать поиск элементов на основе XPath.
26. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе CSS по атрибутам id, name и class.
27. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе CSS с проверкой значения атрибута.
28. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе CSS с использованием комбинации условий.
29. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе CSS с применением движения по дереву элементов.
30. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе XPath по атрибутам.
31. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе XPath с проверкой значения атрибута.
32. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе XPath с использованием комбинации условий.
33. Осуществить поиск элемента на произвольной интернет-странице с использованием локатора на основе XPath с применением движения по дереву элементов.
34. Осуществить поиск группы элементов на произвольной интернет-странице с использованием локатора на основе CSS.
35. Осуществить поиск группы элементов на произвольной интернет-странице с использованием локатора на основе XPath.
36. Получить атрибуты любого элемента произвольной интернет-страницы. Убедиться в правильности полученных атрибутов.
37. Получить текст любого элемента произвольной интернет-страницы. Убедиться в правильности полученного текста.
38. Продемонстрировать определение видимости произвольного элемента произвольной интернет-страницы.
39. Получить стили любого элемента произвольной интернет-страницы. Убедиться в правильности полученных стилей.
40. Получить размер и положение любого элемента произвольной интернет-страницы. Убедиться в правильности полученных значений.
41. Продемонстрировать клик левой кнопкой мыши.
42. Осуществить ввод текста в поле для ввода, сделать все необходимые проверки. Очистить поле и убедиться в том, что это произошло.
43. Advanced Interactions API.
44. Работа со ложными и невидимыми элементами.
45. Ожидание появления элемента, проверка наличия и отсутствия элемента.
46. Ожидание исчезновения элемента, ожидание видимости элемента.
47. Особые условия ожидания.
48. Продемонстрировать ожидание загрузки страницы.
49. Обработка особых ситуаций.
50. Продемонстрировать работу с окнами и фреймами при поиске элемента.
51. Продемонстрировать работу с Selenium Server.
52. Получить доступ к логам сервера.
53. Продемонстрировать перехват трафика.

5. Контрольные вопросы текущего контроля

1. Определение тестирования.
2. Что такое дефект?

3. Что такое сбой?
4. Какие стадии возможны у любого дефекта?
5. Какие виды тестирования существуют?
6. Какие техники тестирования существуют?
7. Какие принципы формирования наборов тестовых данных существуют?
8. Какие уровни тестирования существуют?
9. Как связано тестирование с другими этапами разработки ПО?
10. Сформулируйте общие положения обеспечения качества ПО.
11. Какие атрибуты качества у требований выделяют?
12. Что такое тестируемость требования?
13. Что такое корректность требования?
14. Что такое однозначность требования?
15. Что такое трассируемость требования?
16. Что такое надежность ПС и как его можно протестировать?
17. Что такое масштабируемость ПС и как его можно протестировать?
18. Что такое безопасность ПС и как его можно протестировать?
19. Что такое защита ПС и как его можно протестировать?
20. Что такое доступность ПС и как его можно протестировать?
21. Что такое удобство установки ПС и как его можно протестировать?
22. Что такое проверяемость ПС и как его можно протестировать?
23. Что такое устойчивость ПС и как его можно протестировать?
24. Что такое удобство использования ПС и как его можно протестировать?
25. Что такое переносимость ПС и как его можно протестировать?
26. Что такое возможность повторного использования ПС и как его можно протестировать?
27. Что такое модульное тестирование?
28. Какие инструменты применимы для разработки модульные тестов в C#?
29. Какие инструменты применимы для разработки модульных тестов в Java?
30. Какие инструменты применимы для разработки модульных тестов в JavaScript?
31. Что такое аннотация в модульном тестировании?
32. Что такое фикстура в модульном тестировании?
33. Как осуществить тестирование исключительных ситуаций?
34. Как можно управлять процессом выполнения модульных тестов в Java?
35. Какие инструменты можно использовать для визуализации результатов модульного тестирования?
36. Что такое параметризованный модульный тест? Привести пример.
37. Как осуществляется тестирование конструктора?
38. Как осуществляется тестирование метода?
39. Каковы атрибуты тестового сценария?
40. Как на основе вариантов использования разработать тестовые сценарии?
41. Как осуществляется тестирование альтернативных течений?
42. Как осуществляется тестирование исключительных течений?
43. Каковы этапы жизненного цикла дефекта?
44. Как установить желаемое поведение системы?
45. Каковы атрибуты дефекта?
46. Какова инфраструктура автоматизированного тестирования на C#?
47. Какова инфраструктура автоматизированного тестирования на Java?
48. Какова инфраструктура автоматизированного тестирования на JavaScript?
49. Какова инфраструктура автоматизированного тестирования на Python?
50. Какова инфраструктура автоматизированного тестирования на Ruby?
51. Какие способы подключения библиотеки автоматизированного тестирования существуют?

52. Каковы основные команды библиотеки Selenium WebDriver?
53. Какие команды существуют для перехода по URL?
54. Какие команды существуют для поиска элементов?
55. Каким образом осуществляется инициализация драйвера?
56. Как и зачем формируются ожидания появления элемента?
57. Что такое DOM и как это помогает при поиске элементов?
58. Что такое локатор?
59. Каковы особенности локаторов на основе CSS?
60. Каковы особенности локаторов на основе XPath?
61. Как осуществляется поиск на странице при помощи JavaScript?
62. Какие команды существуют для работы с Cookies?
63. Какие команды существуют для работы с Capabilities?
64. Как осуществляется работа с невидимыми элементами?
65. Как осуществляется ожидание исчезновения элемента?
66. Как осуществляется работа с окнами и фреймами
67. Что такое качество ПС?
68. Какие метрики качества ПС выделяют?
69. Какие типы тестирования ПС существуют?
70. Каковы основные направления обеспечения качества ПС?
71. Что такое модульные тесты?
72. Что такое исследовательское тестирование?
73. Что такое интеграционное тестирование?
74. Что такое нагрузочное тестирование?
75. Что такое регрессионное тестирование?
76. Что такое приёмочное тестирование?
77. Как подготовить данные для нефункционального тестирования?

6. Контрольные задания итогового контроля

1. Осуществить анализ фрагмента спецификации требований (пункт 9.1).
2. Сформировать наборы тестовых данных для тестирования формы авторизации.
3. Сформировать тестовые сценарии для произвольных трёх вариантов использования (пункт 9.1).
4. Осуществить модульное тестирование класса (пункт 9.3).
5. Осуществить модульное тестирование класса (пункт 9.3). Использовать параметризованные тесты.
6. Осуществить автоматизированное тестирование страницы Ya.ru: протестировать переходы по любым 10 ссылкам стартовой страницы.
7. Осуществить автоматизированное тестирование страницы Ya.ru: протестировать авторизацию в почте.
8. Осуществить автоматизированное тестирование страницы Ya.ru: протестировать работу произвольного поискового запроса.
9. Осуществить автоматизированное тестирование страницы Ya.ru: протестировать работу произвольных пяти операций экранной клавиатуры.
10. Осуществить автоматизированное тестирование страницы Ya.ru: протестировать просмотр погоды на 10 дней.

7. Вопросы к зачёту по дисциплине

1. Основные понятия тестирования ПО.
2. Основные понятия обеспечения качества ПО.
3. Модели и характеристики качества ПО.
4. Процессы управления качеством ПО.
5. Требования к качеству ПО.

6. Характеристики дефектов.
7. Техники управления качеством ПО.
8. Количественная оценка качества ПО.
9. Основные вопросы тестирования ПО в гибких методологиях разработки ПО.
10. Техники и виды тестирования.
11. Модульное тестирование. Аннотации, фикстуры.
12. Модульное тестирование. Организация тестов в группы. Обработка событий при выполнении модульных тестов.
13. Тестирование классов.
14. Инфраструктура автоматизированного тестирования.
15. Использование CSS для поиска элементов.
16. Использование XPath для поиска элементов.
17. Поиск внутри элемента.
18. Поиск нескольких элементов.

8. Типовые задания для самостоятельной работы

1. Паттерны проектирования: Builder, Factory Method и их тестирование.
2. Структурные паттерны: Composite, Flyweight и их тестирование.
3. Паттерны поведения: Chain of Responsibility, Strategy, Memento и их тестирование.
4. Гибкие методологии разработки ПС и тестирование.
5. Тестирование аналитических и проектных моделей.
6. Тестирование классов.
7. Тестирование взаимодействия компонентов.
8. Тестирование иерархий классов.
9. Тестирование распределенных объектов.
10. Использование контрольных таблиц при модульном тестировании.
11. Нагрузочное тестирование форума.
12. Нагрузочное тестирование интернет-магазина.
13. Бесплатные инструменты нагрузочного тестирования.
14. Бесплатные инструменты функционального тестирования.

9. Варианты заданий для промежуточного и итогового контроля

9.1. Фрагмент спецификации требований к программной системе тестирования знаний по английскому языку

1. Назначение документа

Документ предназначен для описания требований к системе проведения срезового тестирования по английскому языку

2. Общее описание процесса

Процесс состоит из следующих этапов:

Подготовка теста

Тренер создает тест, варианты теста и добавляет к ним вопросы. После этого тренер составляет список участников теста. Далее тренер создает расписание, где указывается время, место проведения теста, а так же количество вакантных мест. После этого сотрудники которые есть в с писке участников теста могут записаться на тест при наличии вакантных мест. Записать сотрудников на тест так же может и тренер.

Проведение теста

Сотрудники приходят на тест в указанное время. Тест состоит из двух частей – письменной и устной (Speech). На тест отводится ограниченное время. По завершению теста сотрудникам выставляются результаты. За письменную часть результаты выставляются автоматически, за устную часть баллы выставляет тренер.

Выставление и редактирование результатов за тест

Тренер выставляет баллы за устную часть, но так же у него есть возможность редактировать баллы за каждый вопрос письменной части. Так же тренер может изменять ответы сотрудников.

После выставления результатов сотрудник может ознакомиться с ними.

Просмотр результатов теста

Сотрудники могут просмотреть свои результаты, выставленные им за каждую часть теста, а так же общий балл за тест, а так же результаты всех предыдущих тестов, в которых он участвовал. Так же существует возможность просматривать правильные/неправильные ответы, если тренер предоставил такую возможность. Тренер может просматривать результаты всех сотрудников. Ресурс-менеджер так же может просматривать результаты всех сотрудников, но по умолчанию видит лишь результаты сотрудников, которые ему подчинены.

3. Роли пользователей системы.

В системе есть три вида пользователей:

1. Сотрудник, проходящий тест (Respondent)
   Сотрудник может записываться на тест, заполнять тест, просматривать свои результаты
2. Ресурс-менеджер (ResourceManager)
   Имеет все возможности сотрудника, а так же может просматривать результаты всех сотрудников. По умолчанию ему отображаются результаты только тех сотрудников, которые ему подчинены.
3. Тренер (Instructor)
   Тренер имеет доступ ко всем возможностям системы и может устанавливать роли для других сотрудников.

4. Описание функциональности системы

4.1. Подготовка теста

UC1. Авторизация

Действующие лица:	• Тренер • Сотрудник • Ресурс-менеджер
Краткое описание:	Пользователь заходит в систему и авторизуется
Предусловия:	Пользователь решает зайти в систему
Постусловия:	Система открыла форму в соответствии с ролью пользователя
Нормальное течение:	1. Пользователь заходит по ссылке на сайт системы тестирования и попадает на форму авторизации. 2. Пользователь вводит логин (поле Login) и пароль (поле Password) 3. При успешной проверке логина и пароля система открывает форму, соответствующую роли: • Для тренера - форма «Список тестов» (Tests) • Для сотрудника и ресурс-менеджера – форма «Тест»

Альтернативные течения:	Шаг 2. 1. Введены неверные логин и/или пароль 2. Система снова запрашивает логин и пароль
Приоритет (Критично | Важно | Желательно):	Важно
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Всегда
Дополнительные требования	Логин и пароль сотрудника из PMC (или логин и пароль из данных сотрудника для студентов)

1.1.1. Создание теста

UC2. Просмотр списка тестов

Действующие лица:	Тренер
Краткое описание:	Тренер может просмотреть список тестов
Предусловия:	Тренер желает ознакомиться со списком всех тестов
Постусловия:	Открыт список тестов
Нормальное течение:	1. На экране выведена форма списка тестов. Для каждого теста на форме указаны: • Название теста (Test) • Дата теста (Start Date) • Статус (Status) • Комментарии (Comments) Данные в таблице можно сортировать и фильтровать по всем полям, кроме комментариев. 2. Система предоставляет тренеру следующие возможности: • Просмотреть информацию о конкретном тесте (см. UC4) • Создать тест (Create Test)(см. UC3) • Просмотреть свойства теста (Test Properties) • Импортировать старые результаты теста (Load XML) (см. UC29)
Альтернативные течения:	-
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто |	Всегда

Иногда | Редко | Один раз):	

UC3. Создать тест

Действующие лица:	Тренер
Краткое описание:	Тренер создает тест
Предусловия:	Открыт список тестов
Постусловия:	Тест создан
Нормальное течение:	1. Тренер выбирает создание нового теста (Create Test). 2. Тренер попадает на форму создания нового теста и должен ввести следующие данные: • Название (Title) • Дату начала тестирования (Start date) • Дату окончания тестирования (End date) • Комментарий (Comment) • Флаг Разрешить просмотр ответов (Allow View results) • Флаг Тест для студентов (Test for students) Поля “Title”, Start date и “End date” обязательны для заполнения 3. Тренер подтверждает сохранение нового теста. 4. Система сохраняет новый тест в списке тестов. 5. Статус созданного теста – Draft
Альтернативные течения:	Шаг 3. Тренер решает не подтверждать сохранение тестирования, новый тест не создается.
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Часто
Дополнительные требования	

UC4. Просмотр информации о тесте

Действующие лица:	Тренер
Краткое описание:	Тренер может просмотреть интересующий его тест
Предусловия:	Тренер желает ознакомиться с определенным тестом и находится на форме со списком тестов

Постусловия:	Открыт нужный тест
Нормальное течение:	1. Тренер просматривает список тестов и выбирает необходимый ему тест 2. Система выводит на экран форму теста. По умолчанию открывается закладка со списком участников теста (вкладка Participants). 3. Тренер может перейти на любую вкладку формы (Participants, Schedule, Version, Statistics) и просматривать информацию на них с использованием фильтров и сортировки. 4. Тренеру доступны следующие возможности • Обновить список участников теста (Refresh) • Добавить и редактировать список участников теста из списка сотрудников (Add Participant) • Импортировать список участников теста из UPSA • Разослать нотификации • Установить тест на паузу/продолжить(см. UC16) • Остановить выполнение теста • Добавить устный балл • Просмотреть ответы сотрудника(View User Answers) • Установить/изменить вариант теста сотруднику • Отредактировать ответы (Edit User Answers) • Отредактировать баллы за письменный ответ(Edit User Rates)
Альтернативные течения:	
Приоритет (Критично | Важно | Желательно):	Важно
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Часто

4.1.2. Работа со списком сотрудников

UC5. Просмотр списка сотрудников

Действующие лица:	Тренер
Краткое описание:	Тренер может просмотреть список всех сотрудников
Предусловия:	Тренер желает ознакомится со списком сотрудников
Постусловия:	Открыт список сотрудников
Нормальное течение:	1. Система отображает форму со списком сотрудников(Users and Roles). Для каждого сотрудника отображается:

	• Логин (Login) • Имя (First name) • Фамилия (Last name) • E-mail • Дата создания (Create date) • Офис (Office) • Роль (Role) Список можно сортировать по всем полям 2. Система предоставляет тренеру следующие возможности: • Добавить нового сотрудника (Create New User) • Редактировать информацию о сотруднике (Edit User) • Переместить сотрудника в архив (Move to Archive)
Альтернативное течение:	
Исключительное течение	
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Всегда

UC6. Добавление нового сотрудника

Действующие лица:	Тренер
Краткое описание:	Тренер добавляет нового сотрудника
Предусловия:	Открыт список сотрудников (Users and Roles)
Постусловия:	Сотрудник добавлен
Нормальное течение:	1. Тренер выбирает добавление нового сотрудника (Create New User) 2. Тренер попадает на форму добавления нового сотрудника и должен ввести следующие данные: • Логин (Login) • Пароль (Password) • E-mail • Имя (First Name) • Фамилия(Second Name) • Офис(Office) • Роль (Role)

	• Ресурс-менеджер (RM) 3. Тренер подтверждает добавление нового сотрудника 4. Система сохраняет нового сотрудника в списке сотрудников
Альтернативные течения:	Шаг 3. Тренер решает не подтверждать добавление сотрудника, новый сотрудник не добавляется.
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Часто
Дополнительные требования	

UC1. Добавить и редактировать список участников теста

Действующие лица:	Тренер
Краткое описание:	Тренер хочет добавить или отредактировать список участников теста, которые будут проходить очередное тестирование.
Предусловия:	Создан тест
Постусловия:	Установлен список сотрудников, которые будут проходить тестирование
Нормальное течение:	1. Тренер переходит на форму работы с тестом. Система отображает закладку со списком участников теста (вкладка «Participants»). Для нового недавно созданного теста - Список пуст. 2. Тренер выбирает вариант создания списка с помощью выбора из существующего списка сотрудников (Add Participant), либо с помощью импорта из системы UPSA(Import Participant from UPSA Search)(см. Альтернативное течение). 3. Система отображает окно для выбора сотрудников из списка сотрудников. 4. Тренер выбирает из этого списка сотрудников, которых требуется добавить в список участников теста, и подтверждает свой выбор. 5. Система открывает форму теста на закладке со списком участников теста (вкладка «Participants») на которой отображены сотрудники записанные на данный. Вновь добавленным сотрудникам присваивается статус “Draft” (см. Описание статусов сущностей). 6. Тренер может отредактировать список сотрудников

	путем добавления/удаления сотрудников, при условии, что статус удаляемого сотрудника “Draft” или “Assigned”(см. Описание статусов сущностей). 7. Система открывает форму теста на закладке со списком участников теста (вкладка «Participants») на которой отображает только сотрудников, имеющих роль respondent 8.
Альтернативное течение:	Шаг 2 1. Тренер решил не пользоваться выбором сотрудников из существующего списка сотрудников, а сделать импорт сотрудников из системы UPSA. 2. Система предлагает тренеру выбрать файл для импорта. 3. Тренер выбирает файл и подтверждает добавление. 4. Процесс переходит на шаг 5 основного процесса
Исключительное течение	Шаг 4 1. Тренер решает не подтверждать добавление выбранных сотрудников. Добавление сотрудников в список участников теста не происходит.
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Всегда

UC2. Поиск по участникам теста

TBD

UC3. Рассылка нотификаций

Действующие лица:	• Тренер • Сотрудник
Краткое описание:	Тренер рассылает сотрудникам различные нотификации
Предусловия:	Выбран требуемый тест
Постусловия:	Результаты доведены до сотрудников и им разрешен просмотр результатов на сайте.
Нормальное течение:	1. Тренер заходит на форму теста, во вкладку участников теста и отмечает сотрудников, которым необходимо выслать нотификации. 2. Тренер выбирает из списка требуемую нотификацию:

	• о начале срезового тестирования по английскому языку (CHOSE_TIME); • о времени и дате сессии на которую записался сотрудник (ASSIGN_NOTIFICATION); • с сообщением результатов теста (CHECK_RESULTS); • с сообщением неверных ответов (CHECK_WRONGS); Описание содержимого нотификаций см. в разделе «Перечень нотификаций» 3. Система отправляет через Outlook выбранную нотификацию сотрудникам.
Альтернатив-ные течения:	
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Всегда.

4.1.3. Работа с расписанием

Определить расписание прохождения тестирования

Действующие лица:	Тренер
Краткое описание:	Тренер определяет даты/часы/место (сессии), которые составят расписание тестирования
Предусловия:	Решено проводить тестирование
Постусловия:	Составлено расписание прохождения тестирования (сессий)
Нормальное течение:	1. Тренер начинает составление расписания проведения тестирования. 2. Тренер на форме просмотра теста, во вкладке управления расписанием (вкладка «Schedule») выбирает добавить сессию(Add Schedule). Для каждой сессии требуется ввести: • Дату начала(Start date) • Время начала • Продолжительность • Место проведения • Число вакантных мест 3. Тренер подтверждает добавление сессии 4. Система отображает расписание на экране.
Альтернативные течения:	Шаг 3. 1. Тренер решает не подтверждать создание сессии.

	Сессия не добавлена в расписание.
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Всегда
Дополнительные требования	• Тренер может вносить изменения в расписание, добавляя новые сессии или удаляя уже существующие. • Тренер может добавлять в сессии сотрудников из числа сотрудников, назначенных на прохождение тестирования • Тренер не может удалять сессии, если в них записаны сотрудники • Нельзя добавлять сотрудников в уже начавшуюся сессию. • Система определяет кол-во минимально необходимых мест для тестирования исходя из списка сотрудников, выбранного для теста.

Запись на тестирование

Действующие лица:	• Сотрудник • Тренер
Краткое описание:	Сотрудник или тренер выбирают дату прохождения тестирования.
Предусловия:	Расписание проведения тестирования создано
Постусловия:	Уточненное расписание проведения тестирования
Нормальное течение:	1. Сотрудник переходит по ссылке, пришедшей в уведомлении (см. «Рассылка уведомлений о дате прохождения тестирования»), вводит логин и пароль и попадает на форму расписания. Или Сотрудник авторизуется на сайте и попадает на форму с расписанием. 2. Сотрудник видит все сессии теста и количество оставшихся свободных мест и выбирает нужную ему сессию проведения тестирования. Сотрудник подтверждает свой выбор, 3. Система вносит сотрудника в расписание на выбранную им сессию. 4. Сессия сотрудника приобретает статус Assigned. 5. Если сотрудник решил записаться на другую дату ранее, чем за сутки до наступления времени сдачи теста, он может это сделать. (см. UC12) 6. За сутки до наступления даты прохождения теста система запрещает запись на все сессии этой даты для

	сотрудников, (тренер может редактировать список сотрудников).
Альтернативные течения:	1. Тренер сам вносит необходимого сотрудника на определенную дату/время/место. 2. Система вносит сотрудника в расписание на выбранную тренером сессию. 3. Сессия сотрудника приобретает статус Assigned и отправляет сотруднику уведомление о том, что тренер назначил для него определенную дату тестирования. 4. Возврат на шаг 5 основного процесса
Приоритет (Критично | Важно | Желательно):	Критично
Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Всегда
Дополнительное требование	• Не разрешается запись на сессию, в которой отсутствуют свободные места.

Изменить дату прохождения тестирования

Действующие лица:	• Тренер • Сотрудник
Краткое описание:	Сотрудник изменяет дату проведения своего тестирования или тренер переносит сотрудника по своему усмотрению.
Предусловия:	Предыдущие дата/время тестирования сотрудника
Постусловия:	Уточненное расписание проведения тестирования
Нормальное течение:	1. Сотрудник заходит на сайт, вводит логин и пароль и видит форму просмотра расписания. 2. Сотрудник выбирает другую сессию проведения тестирования (старая освобождается). 5. Система вносит сотрудника в расписание на выбранную им сессию.
Альтернативные течения:	1. Тренер заходит на сайт и переходит к форме предстоящего тестирования, на вкладку просмотра графика (вкладка «Schedule» для тренера). 2. Тренер может записать нужного сотрудника в выбранную дату. При этом в его старой сессии одно место становится свободным.
Приоритет (Критично | Важно | Желательно):	Критично

Частота использования (Всегда | Часто | Иногда | Редко | Один раз):	Часто

9.2. Форма для разработки тестов

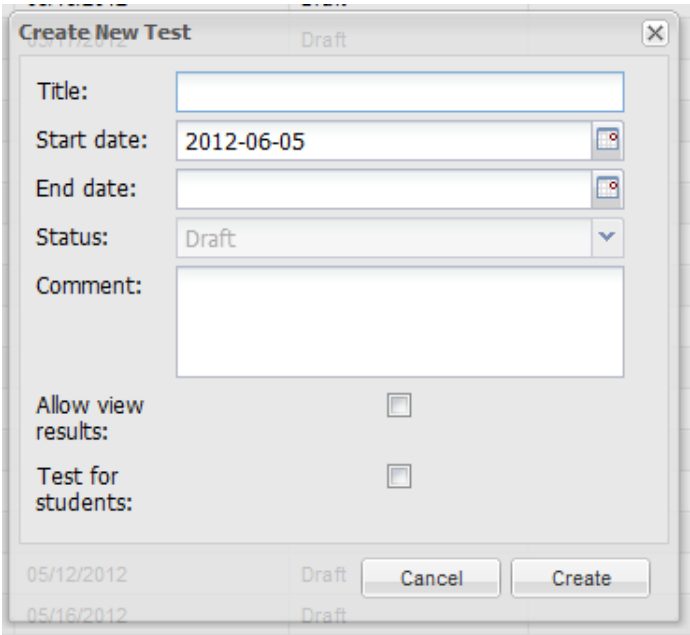

Рис. 9.1. Форма Создание нового теста

Таблица 9.1. Элементы формы «Создание нового теста».

Поле	Тип	Заполнение	Описание
Title	Text	Обязательно	Поле ввода названия теста
Start date	Date (dd/mm/yyyy)	Обязательно	Выбор даты начала тестирования
End date	Date (dd/mm/yyyy)	Обязательно	Выбор даты окончания тестирования
Status	Combo-box		Статус, в котором в данный момент находится тест, т.к. вновь созданный тест имеет статус “Started” поле недоступно.
Comment	Text-area	Не обязательно	Поле для введения тренером каких либо примечаний по поводу теста
Allow view results	Check-box		При выборе данной опции сотрудники могут просматривать свои неверные ответы
Test for students	Check-box		При выборе данной опции, создается тест для студентов
Cancel	Button		Отмена создания нового теста

Поле	Тип	Заполнение	Описание
Create	Button		Подтверждение создания теста с выбранными параметрами

9.3.Фрагменты кода для тестирования

Класс для тестирования на C#

```
    public class Matrix
    {
        private const string format = "{0:0,00}}";
        private readonly double[,] elements;
        public int Rows
        {
            get { return elements.GetRows(); }
        }
        public int Columns
        {
            get { return elements.GetColumns(); }
        }
        public  static  void  CheckNull(object  param,  string
paramName)
        {
            if (param == null)
            {
                throw new ArgumentNullException(paramName);
            }
        }
        public  static  void  CheckNull(object  paramA,  string
paramAName, object paramB, string paramBName)
        {
            CheckNull(paramA, paramAName);
            CheckNull(paramB, paramBName);
        }
        public static void CheckSize(double[,] elements, string
objectName)
        {
            CheckNull(elements, objectName);
            if (elements.Length < 1)
            {
                throw new MatrixException(objectName);
            }
        }
        public Matrix(double[,] elements)
        {
            CheckNull(elements, nameof(elements));
            CheckSize(elements, nameof(elements));
            this.elements = (double[,])elements.Clone();
        }
        public double this[int rowIndex, int columnIndex]
        {
            get
            {
                return this.elements[rowIndex, columnIndex];
            }
        }
```

```
        public static void CheckSameSizes(Matrix matrixA, Matrix
matrixB)
        {
            CheckNull(matrixA,      nameof(matrixA),      matrixB,
nameof(matrixB));
            if     ((matrixA.Rows     !=     matrixB.Rows)     ||
(matrixA.Columns != matrixB.Columns))
            {
                throw      new      MatrixException(matrixA.Rows,
matrixA.Columns, matrixB.Rows, matrixB.Columns);
            }
        }
        public static void CheckSizesToMultiply(Matrix matrixA,
Matrix matrixB)
        {
            CheckNull(matrixA,      nameof(matrixA),      matrixB,
nameof(matrixB));
            if (matrixA.Columns != matrixB.Rows)
            {
                throw      new      MatrixException(matrixA.Rows,
matrixB.Columns);
            }
        }
        public static Matrix operator +(Matrix matrixA, Matrix
matrixB)
        {
            CheckNull(matrixA,      nameof(matrixA),      matrixB,
nameof(matrixB));
            CheckSameSizes(matrixA, matrixB);
            double[,]    array    =    new    double[matrixA.Rows,
matrixA.Columns];
            for (int i = 0; i < array.GetRows(); i++)
            {
                for (int j = 0; j < array.GetColumns(); j++)
                {
                    array[i, j] = matrixA[i, j] + matrixB[i, j];
                }
            }
            return new Matrix(array);
        }
        public static Matrix operator -(Matrix matrixA, Matrix
matrixB)
        {
            CheckNull(matrixA,      nameof(matrixA),      matrixB,
nameof(matrixB));
            CheckSameSizes(matrixA, matrixB);
            double[,]    array    =    new    double[matrixA.Rows,
matrixA.Columns];
            for (int i = 0; i < array.GetRows(); i++)
            {
                for (int j = 0; j < array.GetColumns(); j++)
                {
                    array[i, j] = matrixA[i, j] - matrixB[i, j];
```

```
                }
            }
            return new Matrix(array);
        }
        public static Matrix operator *(Matrix matrixA, Matrix
matrixB)
        {
            CheckNull(matrixA,     nameof(matrixA),     matrixB,
nameof(matrixB));
            CheckSizesToMultiply(matrixA, matrixB);
            double[,]    array    =    new    double[matrixA.Rows,
matrixB.Columns];
            for(int i = 0; i < array.GetRows(); i++)
            {
                for(int j = 0; j < array.GetColumns(); j++)
                {
                    array[i, j] = 0;
                    for(int k = 0; k < matrixB.Rows; k++)
                    {
                        array[i,   j]   +=   matrixA[i,   k]   *
matrixB[k, j];
                    }
                }
            }
            return new Matrix(array);
        }
        public  static  Matrix  operator  *(Matrix  matrix,  int
value)
        {
            CheckNull(matrix, nameof(matrix));
            double[,]    array    =    new    double[matrix.Rows,
matrix.Columns];
            for (int i = 0; i < matrix.Rows; i++)
            {
                for (int j = 0; j < matrix.Columns; j++)
                {
                    array[i, j] = matrix[i, j] * value;
                }
            }
            return new Matrix(array);
        }
        public  static  bool  operator  ==(Matrix  matrixA,  Matrix
matrixB)
        {
            if((object.ReferenceEquals(matrixA,     null)     &&
(object.ReferenceEquals(matrixB, null))))
            {
                return true;
            }
            CheckNull(matrixA,     nameof(matrixA),     matrixB,
nameof(matrixB));
            if((matrixA.Rows      !=      matrixB.Rows)      ||
(matrixA.Columns != matrixB.Columns))
```

```
            {
                return false;
            }
            bool equality = true;
            for(int i = 0; i < matrixA.Rows; i++)
            {
                for(int j = 0; j < matrixA.Columns;j++)
                {
                    if(matrixA[i,j] != matrixB[i,j])
                    {
                        equality = false;
                    }
                }
            }
            return equality;
        }
        public static bool operator !=(Matrix matrixA, Matrix
matrixB)
        {
            CheckNull(matrixA,     nameof(matrixA),     matrixB,
nameof(matrixB));
            CheckSameSizes(matrixA, matrixB);
            return !(matrixA == matrixB);
        }
        public override bool Equals(object matrix)
        {
            return this == (Matrix)matrix;
        }
        public void CheckResultMatrix()
        {
            for(int i = 0; i < this.elements.GetRows(); i++)
            {
                for(int j = 0; j < this.elements.GetColumns();
j++)
                {
                    if(this[i,j] > 100)
                    {
                        throw new MatrixException();
                    }
                }
            }
        }
        public override string ToString()
        {
            StringBuilder result = new StringBuilder();
            for (int i = 0; i < this.Rows; i++)
            {
                for (int j = 0; j < this.Columns; j++)
                {
                    result.AppendFormat(this[i, j] + "  ");
                }
                result.AppendLine();
            }
```

```
            return result.ToString();
        }
    }
}
```

Класс для тестирования на Java

```
    package calculation;

    public interface IscholarshipCalculator {

        public double scholarshipCalculate(double
stepUpCoefficient);

    }
```

Листинг 1. Интерфейс IscholarshipCalculator

```
    public class ScholarshipCalculatorImpl implements
IscholarshipCalculator

        {

            public static final double BASIC_SCHOLARSHIP = 100;

            public double scholarshipCalculate(double
stepUpCoefficient) {

            return BASIC_SCHOLARSHIP * stepUpCoefficient;

            }

            public double stepUpCoefficientCalculate(int
averageMark) throws NotSuchMarkException {

                double stepUpCoefficient;

                switch (averageMark) {

                case 3:

                    stepUpCoefficient = 1;

                    break;

                case 4:

                    stepUpCoefficient = 1.3;

                break;

                case 5:

                    stepUpCoefficient = 1.5;

                    break;

                default:

                    throw new NotSuchMarkException("There is no
mark: " + averageMark);
```

```
            }
        return stepUpCoefficient;
        }
}
```

Оценочные материалы составлены в соответствии с Федеральным государственным образовательным стандартом высшего образования по направлению подготовки 09.03.04 «Программная инженерия» (квалификация выпускника – бакалавр, форма обучения – очная, срок обучения – 4 года.

Оценочные материалы составил:
к.ф.-м.н., доцент кафедры

«Вычислительная и прикладная математика» А. А. Бубнов

Программа рассмотрена и одобрена на заседании кафедры «Вычислительная и прикладная математика» (протокол № ____ от _____).

Заведующий кафедрой
«Вычислительная и прикладная математика»
д.т.н., профессор Г.В. Овечкин