Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информатика»

ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ПРАКТИКУМ
по выполнению лабораторных работ
для студентов специальности 1-40 04 01
«Информатика и технологии программирования»
дневной формы обучения

Гомель 2024

УДК 004.054(075.8)
ББК 32.972.1я73
Т36

Рекомендовано кафедрой «Информатика» ГГТУ им. П. О. Сухого
(*протокол № 11 от 29.06.2022 г.*)

Составитель *В. Н. Шибеко*

Рецензент: проф. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
д-р техн. наук, проф. *И. А. Мурашко*

Тестирование и верификация программного обеспечения : практикум по выполнению лаборатор. работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» днев. формы обучения / сост. В. Н. Шибеко. – Гомель : ГГТУ им. П. О. Сухого, 2024. – 65 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Т36

Рассмотрены вопросы и задания автоматизированного тестирования, включающего подготовку чек-листов, модульное проектирование, документирование результатов.

Для студентов специальности 1-40 04 01 «Информатика и технологии программирования».

УДК 004.054(075.8)
ББК 32.972.1я73

СОДЕРЖАНИЕ

Введение

Основные задачи курса лекций по дисциплине «Тестирование и верификация программного обеспечения»:

– познакомить слушателей с основными понятиями тестирования;

– познакомить слушателей с методами и приемами тестирования программных приложений, а также с критериями выбора тестов;

В данной работе рассмотрены основные виды тестирования программ и те способы тестирования, которые применяются непосредственно в процессе разработки программных продуктов, что является актуальным для будущих программистов.

Структурно пособие разделено на 6 лабораторных работ и 4 практических задания.

Практические задания необходимы для быстрого знакомства с инструментами тестирования для Java.

Лабораторные работы по видам тестирования, документированию результатов, тестированию web-приложений опираются на пособие [3], разработка и тестирование требований - на [1].

Практические задания основаны на данные электронных ресурсов.

ЛАБОРАТОРНАЯ РАБОТА № 1. Виды тестирования.

1.1. Цель работы: изучить классификацию видов тестирования, практически закрепить эти знания путем генерации тестов различных видов, научиться планировать тестовые активности в зависимости от специфики поставляемой на тестирование функциональности.

1.2. Краткие теоретические сведения

Не существует стандартного перечня видов тестирования, однако некоторые из них настолько очевидны, что стали общепринятыми.

Инсталляционное тестирование (installation testing) – проверка всего того, что связано с инсталляцией продукта в систему и удалением продукта из системы.

Регрессионное тестирование (regression testing) – проверка того, что внесённые в приложение изменения не привели к потере работоспособности того, что ранее работало, и/или привели к работоспособности того, что ранее не работало.

Тестирование нового функционала (new feature testing) – проверка того, что заявленный в данной реализации новый функционал работает должным образом.

Конфигурационное тестирование (configuration testing) – проверка того, как приложение работает с различным оборудованием и/или собственными настройками.

Тестирование совместимости (compatibility testing) – проверка того, как приложение взаимодействует с другими приложениями и операционной системой. В случае веб-ориентированных приложений особое внимание уделяется совместимости с различными браузерами.

Тестирование удобства использования (usability testing) – проверка того, насколько пользователю удобно и приятно работать с приложением.

Тестирование интернационализации (internationalisation testing) – проверка готовности продукта к переводу на различные языки.

Тестирование локализации (localisation testing) – проверка качества перевода продукта на конкретный язык.

Позитивное тестирование (positive testing) – проверка того, как приложение работает в заведомо “тепличных условиях” (корректные данные, условия работы и т.п.).

Негативное тестирование (negative testing) – проверка того, как приложение реагирует на различные “неприятности” (пропала сеть, повреждён файл и т.п.).

Исследовательское тестирование (exploratory testing) – редкий вид тестирования, основанный на профессиональной интуиции тестировщика, которая может подсказать опасные и малоисследованные способы работы с приложением.

Покажем на примере полотенца возможный перечень вопросов по каждому из перечисленных выше видов тестирования.

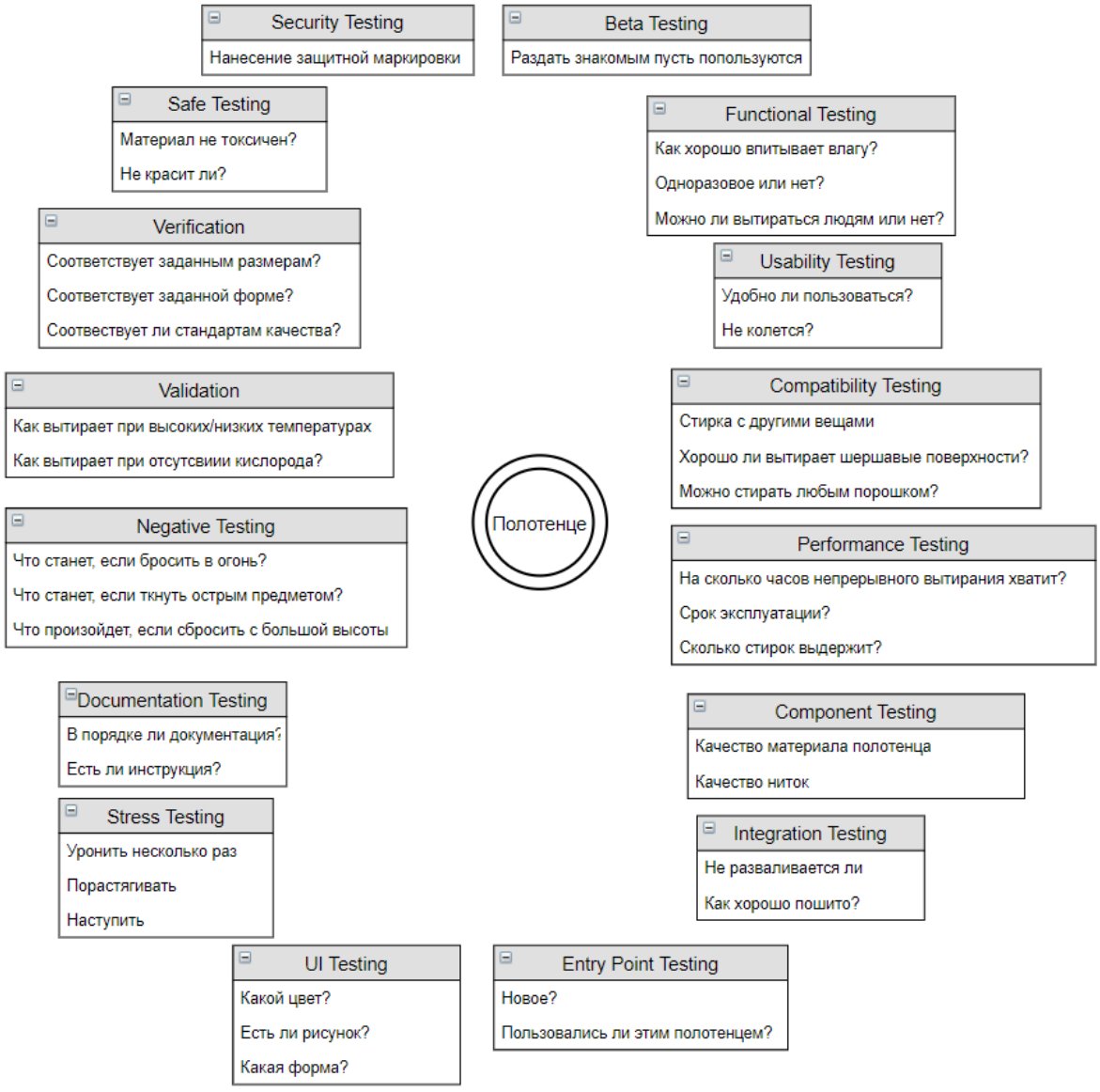


Рис. 1. Пример составления тестов

1.3. Задания на выполнение

Задание 1. Выполнить генерацию тестов (не менее 3 на каждый вид тестирования) различных видов для конкретного объекта реального мира, согласно с вариантом в таблице ниже.

Таблица 1

Варианты заданий

Номер варианта	Предмет
1.	Ручка
2.	Стул
3.	Стол
4.	Точилка
5.	Чашка
6.	Кухонный нож
7.	Вилка
8.	Ложка
9.	Зубная щетка
10.	Щетка для обуви
11.	Ластик
12.	Маркер
13.	Полотенце
14.	Линейка
15.	Циркуль

Пример оформления

Объект тестирования: **Полотенце**		
Вид тестирования	**Тестовые проверки**	**Ожидаемый результат**
Negative Testing	Что станет,если бросить в огонь	Сгорит
	Что станет, сели проткнуть острым предметом	Появится дырка
	Что произойдет, если сбросить с большой высоты	Своства останутся прежними

Задание 2. Взять у преподавателя вариант предметной области.

1) Разработать тесты для первой поставки программного обеспечения (build 1), состоящей из трех модулей (модуль 1, модуль 2, модуль 3). Пользователи – русскоязычное сообщество. Имена модулей выбрать самостоятельно.
2) Разработать тесты для второй поставки программного обеспечения (build 2): исправлены дефекты, доставлена новая функциональность – модуль 4.
3) Разработать тесты для третьей поставки программного обеспечения (build 3): заказчик решил расширять рынки сбыта и просит осуществить поддержку программного обеспечения на английском языке. Это требования и тесты по интернационализации и локализации. Такие требования могут быть направлены на разрешение различий в валютах, форматировании дат, чисел, адресов и телефонных номеров, языках, в том числе различных вариантах одного языка (например, американского и британского вариантов английского), используемых символах и наборах символов, лич-

ных именах и фамилиях, часовых поясах, международных нормативных актах и законах, культурных и политических традициях, размере используемой бумаги, единицах веса и меры, электрическом напряжении и конфигурации электрических разъемов и во многом другом.

4) Разработать тесты для четвертой поставки программного обеспечения (build 4): заказчик хочет убедиться, что программное обеспечение выдержит нагрузку в 2000 пользователей.
Результат выполнения задания оформить в виде таблицы:

Поставка (версия)	**Виды тестирования**	**Возможные тесты**

1.4. Требования к отчету по лабораторной работе

В отчете должны быть отображены следующие пункты:
Задание 1

- Объект тестирования
- Краткие теоретические сведения.
- Не менее 3 вопросов к каждому из перечисленных видов тестирования (пример см. рис. 1). Результат оформить в виде таблицы, приведенной выше.
- Выводы.

Задание 2

- Результат оформленный в виде таблицы.
- Обоснование выбора тестов.

1.5. Контрольные вопросы к лабораторной работе

1. Что такое тестирование?
2. Какие виды тестирования знаете и в чем их суть?
3. Тестовые активности для сформулированных задач
4. Какие существуют виды тестирования в зависимости от знания кода? Дайте характеристику каждому

5. Какая композиция тестов выполняется для первой поставки программного продукта?
6. Какие композиция тестов выполняется для последующих поставок программного продукта?

ЛАБОРАТОРНАЯ РАБОТА № 2. Разработка и тестирование требований

2.1 Цель работы: описать пользовательские требования

2.2. Краткие теоретические сведения

Требование (Requirement) – описание того, какие функции и с соблюдением каких условий должен выполнять программный продукт в процессе решения полезной для пользователя задачи.

Значение требований:

- Позволяют понять, что и с соблюдением каких условий система должна делать.
- Предоставляют возможность оценить масштаб изменений и управлять изменениями.
- Являются основой для формирования плана проекта (в том числе плана тестирования).
- Помогают предотвращать или разрешать конфликтные ситуации.
- Упрощают расстановку приоритетов в наборе задач.
- Позволяют объективно оценить степень прогресса в разработке проекта.

Работа над требованиями включает следующие этапы:

- выявление требований, анализ требований (моделирование бизнес-процессов,
- прототипирование интерфейсов, приоритезация требований, результат этапа – визуализация требований), документирование требований (результат этапа –
- спецификация), тестирование (валидация) требований. Работу с требованиями на этапах выявления, анализа, документирования, как правило, выполняет бизнес-аналитик. Тестирование требований выполняет тестировщик.

В иерархии требований существует три уровня:

- уровень бизнес-требований,
- уровень пользовательских требований,

- уровень продуктовых требований (функциональные и нефункциональные требования).

Бизнес-требования выражают цель, ради которой разрабатывается продукт (зачем он нужен, какая от него ожидается польза).

Пользовательские требования описывают задачи, которые пользователь может выполнять с помощью разрабатываемой системы, и по своей сути представляют собой недетализированные функциональные требования. Поскольку здесь уже появляется описание поведения системы, требования этого уровня могут быть использованы для оценки объёма работ, стоимости проекта, времени разработки. Пользовательские требования оформляются в виде вариантов использования (Use Cases), пользовательских историй (User Stories), пользовательских сценариев (User Scenarios). Функциональные требования описывают поведение системы, т.е. её действия (вычисления, преобразования, проверки, обработку и т.д.). Нефункциональные требования описывают свойства системы (удобство использования, безопасность, надёжность, расширяемость и т.д.), которыми она должна обладать при реализации своего поведения.

Выявление и описание требований: Use Case.

Вариант использования (Use Case) продукта описывает последовательность взаимодействия системы и внешнего действующего лица. Действующим лицом может быть человек, другая система ПО или аппаратное устройство, взаимодействующее с системой для достижения некой цели.

Варианты использования меняют традиционный подход к сбору информации: пользователей не спрашивают, что с их точки зрения, должна делать система, а выясняют, какие задачи собирается с ее помощью решать пользователь.

Цель такого подхода — описать все подобные задачи. До включения каждого варианта использования в утвержденную версию требований заинтересованные в проекте лица проверяют, не вы-

ходит ли он за границы проекта. Теоретически в конечный набор вариантов использования должна входить вся желаемая функциональность системы.

Определение вариантов использования

Определить варианты использования можно несколькими способами:

- сначала определить действующие лица, а затем бизнес-процессы, в которых каждое лицо участвует;
- выразить бизнес-процессы в терминах определенных сценариев, обобщить сценарии в варианты использования и определить действующие лица для каждого варианта; определить внешние события, на которые система должна реагировать, а затем соотнести эти события с участвующими лицами и определенными вариантами использования;
- определить вероятные варианты использования на основе
- функциональных требований; если какие-либо требования невозможно проследить до какого-либо варианта использования, необходимо задуматься, нужны ли они.

Как правило, пользователи сначала определяют самые важные варианты использования, поэтому порядок предлагаемых тем позволит получить представление о приоритетах.

Преимущества применения вариантов использования состоят в том, что каждый вариант сосредоточен на поставленной задаче и пользователе. Тщательное изучение этапов взаимодействия лица и системы помогает еще на ранних стадиях разработки выявить неясности и неточности, а также позволяет составить варианты тестирования на основе вариантов использования. Способ с применением вариантов использования позволяет выявить функциональные требования, с помощью которых пользователи будут выполнять конкретные задачи. Кроме прочего варианты использования облегчают расстановку приоритетов требований. Высшим приоритетом обладают те функциональные требования, которые созданы на основе вариантов использования с

высшим приоритетом. Высший приоритет назначается по следующим причинам:

- варианты использования описывают один из основных бизнес-процессов, активизируемых системой;
- многие пользователи часто обращаются к ним;
- их запросил привилегированный класс пользователей;
- они предоставляют возможности, необходимые для соответствия требованиям;
- функции других систем зависят от их наличия. Существуют также и преимущества технического характера. С помощью варианта использования можно выявить некоторые важные объекты предметной области и их взаимоотношения.

Разработчики, использующие объектно-ориентированные методы проектирования, могут преобразовать варианты использования в объектные модели, такие, как диаграммы классов и диаграммы последовательностей.

2.3. Задание на выполнение

1. Получить у преподавателя задание, содержащее идею и бизнес-цели подлежащего разработке программного продукта.

2. Определить действующие лица и сформулировать наиболее вероятные варианты использования подлежащего разработке программного продукта.

3. Полностью описать три варианта использования подлежащего разработке программного продукта.

4. Для каждого варианта использования указать

- уникальный идентификатор;
- имя в формате «глагол + объект»;
- краткое текстовое описание; предварительные условия;
- выходные условия;
- пронумерованный список действий нормального направления развития.

5. Для каждого варианта использования при необходимости указать пронумерованный список действий альтернативного направления (направлений) развития.

6. Для каждого варианта использования при необходимости указать исключения.

7. Оформить отчет и защитить лабораторную работу.

2.4. Содержание отчета

- Цель работы.
- Описание вариантов использования, подлежащего разработке программного продукта.
- Выводы по работе.

2.5. Контрольные вопросы

- Что такое требование?
- Какие значения имеют требования на проекте?
- Какие существуют этапы работы над требованиями?
- Кто выполняет работу с требованиями?
- Какие существуют уровни требований?
- Что такое вариант использования?
- Для чего нужен вариант использования?
- Какие элементы входят в состав описания варианта использования?
- Что такое основной сценарий варианта использования?
- Что такое альтернативный сценарий варианта использования?

11. Что описывают в исключениях варианта использования?

12. В чем отличие альтернативного сценария от исключения в описании варианта использования?

13. Какие существуют преимущества у вариантов использования как одного из способов описания требований?

ЛАБОРАТОРНАЯ РАБОТА № 3. Подготовка тестов.

3.1. Цель работы: разработка рабочей тестовой документации для тестирования приложения (Чек-листы).

3.2. Краткие теоретические сведения

Любое тестирование, в том числе тестирование ПО - это поиск багов. Баг - это отклонение фактического результата (неких действий) от ожидаемого.

Для чего необходимо тестирование?

Чтобы найти баги до того, как их найдут пользователи, то есть до выпуска продукта. Чтобы проводить тестирование, необходимо:

1. Узнать ожидаемый результат
2. Узнать фактический результат
3. Сравнить эти результаты

Например, нам нужно протестировать сложение целых чисел. При сложении двух чисел 2 и 5 мы ожидаем, что результат будет равен 5 (ожидаемый результат). Чтобы протестировать, мы должны проверить это ожидание, вызвав метод сложения и подав на него указанные числа. Полученный (фактический результат) сравним его с ожидаемым: если они не совпадают, то значит, найден баг.

Таким образом, для тестирования нужно еще выбрать некоторое действие (в данном случае –метод сложения целых чисел), получаем, что процесс тестирования состоит из четырех стадий:

1. Выбрать действие
2. Узнать ожидаемый результат этого действия
3. Узнать фактический результат
4. Сравнить эти результаты

При подготовке тестов необходимо учитывать, что каждый тест должен проверять конкретное предположение, а набор тестов быть дос-

таточно полным, чтобы уверенным в корректной работе приложения. При тестировании сложения целых чисел, мы должны быть убеждены что сложение чисел будет корректно на всем множестве целых чисел. Конечно же, надежно было бы проверить для каждого элемента данного множества. Но даже взяв 32-битное представление – это 2**32 элементов. А взяв это для двух аргументов, получим 2**33 тестов. И это только тестирование корректных данных (позитивное тестирование). А если необходимо учитывать некорректные данные, которые также характеризуются элементами соответствующих множеств (негативное тестирование), то объемы тестирования значительно возрастают. И здесь нам помогают техники тестирования. В рамках лабораторной работы рассмотрим следующие:

- граничные значения
- эквивалентные множества

Метод эквивалентных множеств позволяет минимизировать число тестов, не создавая сценарий для каждого возможного значения, а выбрав только одно значение из целого класса и приняв за аксиому, что для всех значений в данной группе результат будет аналогичным. Например, для сложения целых чисел, эквивалентным множеством будет множество целых чисел. Для проверки достаточно одного значения из него.

Техника граничных значений основана на предположении, что большинство ошибок может возникнуть на границах эквивалентных классов. Она тесно связана с вышеописанной техникой эквивалентного разбиения, из-за чего часто используется с ней в паре. Тогда для примера из предыдущего пункта границами будут являться значения плюс, минус бесконечность. Это особые значения, которые необходимо контролировать. Для проверки на основе граничных значений требуется каждое слагаемое проверить на границы интервалов. Таким образом, мы получим 8 тестов, так как каждое слагаемое может быть особым значением или корректным.

Каждый тест, связанный с конкретным предположением, является тестовым случаем или тест-кейсом. Собранные в один список тест-кейсы образуют чек-лист.

3.3. Задание на лабораторную работу

1. Для заданной ранее предметной области составить чек-лист (check list) для тестирования валидности данных.
    - Чек – лист должен содержать описание тестов модульного тестирования слоя.
    - Чек – лист должен содержать тесты проверки корректности введенных данных
2. Формат таблицы чек-листа

Id	Пункт ТЗ, модуль класс, метод	Краткое описание теста	Подробное описание	Шаги по воспроизведению	Ожидаемый результат	Фактический результат
						При создании чек-листа не заполняется

3.4. Содержание отчета

- Цель работы.
- Задание
- Краткие теоретические сведения по построению чек-листов
- Чек-лист для тестирования в виде таблицы, описание тестов и его описание.
- Обязательно: Каждый тест (test-case) должен иметь ссылку на пункт ТЗ (либо сценарий использования)
- Выводы

3.5. Контрольные вопросы

1. Для чего необходимо тестировать приложения?
2. Что такое чек-лист?
3. Что такое тест-кейсы?
4. Объясните чек-листа с модулем и его функциями (классом и его методами)
5. Объясните связь чек-листа со сценарием использования, требованием
6. Что такое тест?
7. Техники тест-дизайна
    - Классы эквивалентости
    - Анализ граничных значений
    - Метод причинно-следственных связей
    - Предугадывание ошибки
    - Исчерпывающее тестирование
    - Попарное сравнение

ЛАБОРАТОРНАЯ РАБОТА № 4. Unit-тестирование

4.1. Цель работы: изучить технологии модульного тестирования.

4.2. Краткие теоретические сведения

Unit testing **(юнит тестирование или модульное тестирование)** – заключается в изолированной проверке каждого отдельного элемента путем запуска тестов в искусственной среде. Для этого необходимо использовать драйверы и заглушки. Поэлементное тестирование – первейшая возможность реализовать исходный код. Оценивая каждый элемент изолированно и подтверждая корректность его работы, точно установить проблему значительно проще чем, если бы элемент был частью системы.

Unit (Элемент) – наименьший компонент, который можно скомпилировать.

JUnit – это фреймворк регрессионного тестирования, в основном используемый опытными разработчиками ПО для выполнения юнит-тестирования, ускорения программирования и повышения качества кода в Java. Основная его цель – позволить Java-разработчикам писать скрипты и реализовывать повторяющиеся тест-кейсы.

Это один из наиболее часто используемых фреймворков модульного тестирования Java.

Чаще всего JUnit применяется разработчиками для небольших фрагментов кода. Также можно выполнять автоматизированные тесты сайтов, комбинируя Selenium WebDriver с JUnit для автоматизированного тестирования Java. При добавлении любого свежего кода вам будет предложено перезагрузить тест-кейсы и убедиться в отсутствии неполадок.

Особенности

- Наборы тестов;
- Фикстуры;

- Классы JUnit;
- Test Runners.

Преимущества. Среди других фреймворков тестирования, JUnit имеет ряд особенностей:

- Благодаря поддержке функций Java 8, он может использоваться как для интеграционных, так и для юнит-тестов;
- Программисты и разработчики, работающие в этой тестовой среде, находят ее крайне полезной. Легче читать код и исследовать наличие уязвимостей;
- Выявление ошибок на ранней стадии и удобочитаемость повышают надежность кода;
- Практически все основные IDE, такие как Eclipse, NetBeans, Maven, Ant и IntelliJIDEA, совместимы с фреймворком. Так что вы можете легко написать и запустить модульное тестирование прямо из них.
- Используя обновленную версию JUnit (версия 5), можно легко проверить наличие исключений. Также можно выполнять тест-кейсы, написанные на предыдущей версии JUnit.
- Фреймворк юнит-тестирования Java можно использовать с Java 5 или другими версиями.

Недостатки. Фреймворк JUnit не может выполнять тесты зависимостей. Для этого нам нужен TestNG.

Итоги по работе с JUnit:

- JUnit – это широко используемый фреймворк на основе Java. И TestNG и JUnit выполняют аналогичную работу, и их ключевые характеристики в чем-то одинаковы, за исключением возможностей тестирования зависимостей. Процедура выполнения параметризованного тестирования также отличается в обоих фреймворках.
- так как JUnit используется очень давно, он имеет большую поддержку Java-сообщества. Он также определен как стандартная платформа модульного тестирования путем интеграции Selenium WebDriver для приложений на основе Java. Несмотря на то, что у TestNG еще мало пользователей и сообщество постоянно растет, выбор между фреймворками

JUnit и TestNG зависит от потребностей конкретных приложений.

Список аннотаций для Junit:

- @Test – определяет что метод method() является тестовым.
- @Before – указывает на то, что метод будет выполнятся перед каждым тестируемым методом @Test.
- @After – указывает на то что метод будет выполнятся после каждого тестируемого метода @Test
- @BeforeClass – указывает на то, что метод будет выполнятся в начале всех тестов,а точней в момент запуска тестов(перед всеми тестами @Test).
- @AfterClass – указывает на то, что метод будет выполнятся после всех тестов.
- @Ignore – говорит, что метод будет проигнорирован в момент проведения тестирования.
- (expected = Exception.class) – указывает на то, что в данном тестовом методе вы преднамеренно ожидаете Exception.
- (timeout = 100) – указывает, что тестируемый метод не должен занимать больше чем 100 миллисекунд.

Основные методы класса Assert:

- fail(String) – указывает на то что бы тестовый метод завалился при этом выводя текстовое сообщение.
- assertTrue([message], **boolean** condition) – проверяет, что логическое условие истинно.
- assertsEquals([String message], expected, actual) – проверяет, что два значения совпадают.
- Примечание: для массивов проверяются ссылки, а не содержание массивов.
- assertNull([message], object) – проверяет, что объект является пустым **null**.
- assertNotNull([message], object) – проверяет, что объект не является пустым **null**.

- assertSame([String], expected, actual) – проверяет, что обе переменные относятся к одному объекту.
- assertNotSame([String], expected, actual) – проверяет, что обе переменные относятся к разным объектам.

4.3. Задание

- На основе чек-листов из лабораторной работы №3 разработать unit-тесты.
- Выполнить модульное тестирование

4.4. Содержание отчета

- Цель работы.
- Задание
- Краткие теоретические сведения по применению класса Assert
- Коды классов тестирования и пояснения к ним
- Результаты тестирования
- Выводы

4.5. Контрольные вопросы

- Назначение Unit-тестирования
- Основные методы класса Assert
- Аннотации

ЛАБОРАТОРНАЯ РАБОТА № 5. Документирование результатов тестирования

5.1. Цель работы: составить итоговый отчет о результатах тестирования приложения.

5.2. Краткие теоретическое сведения

Дефект (Defect, Bug Report) – это документ, в котором описано неправильное поведение тестируемой системы или ПО. Не нужно путать дефект с ошибкой\багом. В терминологии тестирования ПО ошибка или баг – это непосредственно проблема ПО (например, ошибка в коде), а дефект – это описание данной проблемы.

Для чего необходимо описывать дефекты?

1. Описывая ошибку, мы даем возможность другому человеку воспроизвести тот же результат и убедиться в наличии проблемы. Такое описание необходимо аналитикам, чтобы убедиться в том, что требования действительно не выполняются а также разработчикам, чтобы понять что и как им исправлять.
2. Когда мы получаем исправленное ПО для повторной проверки, детальное описание ошибки дает возможность ничего не забыть и убедиться, что все действительно исправлено.
3. Имея описание ошибки (даже уже исправленной ранее) мы можем повторно проверить, что она не повторяется в новой версии ПО.

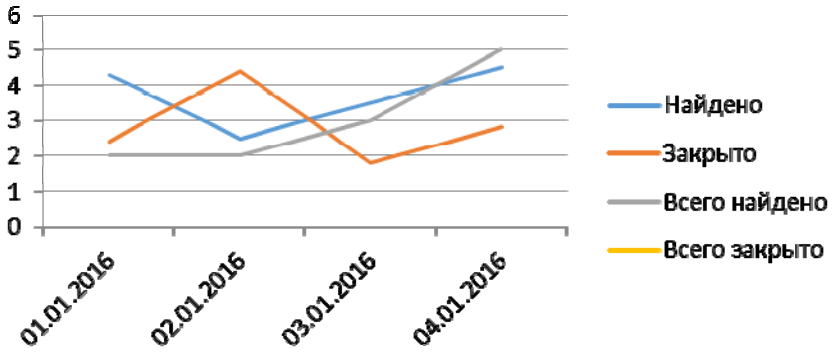
5.3. Задание

Составить отчет о результатах тестирования (test result report).

Отчет должен содержать следующие разделы:

- краткое описание (summary);
- команда тестировщиков (test team);
- описание процесса тестирования (testing process description);
- расписание (timetable):
- статистика по новым дефектам (new defects statistics):
- список новых дефектов (new defects list):

• статистика по всем дефектам (overall defects statistics) (см. статистику по новым дефектам);



• рекомендации (recommendations);
• приложения (appendixes);

Метрики:

Успешное прохождение тест-кейсов:

$$T^{SP} = \frac{T^{Success}}{T^{Total}} \cdot 100\%,$$

где

T^{SP} — процентный показатель успешного прохождения тест-кейсов,

$T^{Success}$ — количество успешно выполненных тест-кейсов,

T^{Total} — общее количество выполненных тест-кейсов.

Минимальная границы значений:

1. Начальная фаза проекта: 10%.
2. Основная фаза проекта: 40%.
3. Финальная фаза проекта: 80%.

Выполнение тест-кейсов:

$$T^{E} = \frac{T^{Executed}}{T^{Planed}} \cdot 100\%,$$

где

T^E – процентный показатель выполнения тест-кейсов,

$T^{Executed}$ – количество выполненных тест-кейсов,

T^{Planed} – количество тест-кейсов, запланированных к выполнению.

Уровни (границы):

- Минимальный уровень: 80 %.
- Желаемый уровень: 95–100 %.

Общее устранение дефектов:

$$D_{Level}^{FTP} = \frac{D_{Level}^{Closed}}{D_{Level}^{Found}} \cdot 100\%,$$

где D_{Level}^{FTP} – процентный показатель устранения дефектов уровня важности за время существования проекта,

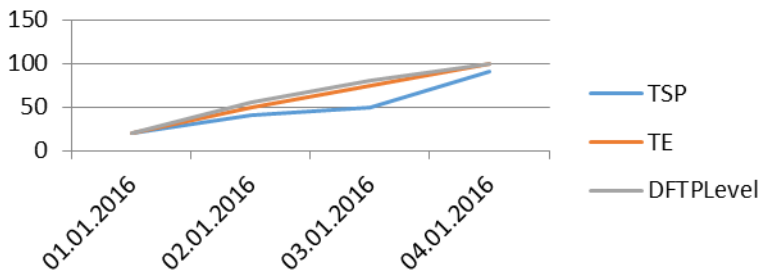
D_{Level}^{Closed} – количество устранённых за время существования проекта дефектов уровня важности,

D_{Level}^{Found} – количество обнаруженных за время существования проекта дефектов уровня важности.

Таблица 1

Минимальные границы значений

		Важность дефекта			
		низкая	средняя	высокая	критическая
Фаза проекта	Начальная	10%	40%	50%	80%
	Основная	15%	50%	75%	90%
	Финальная	20%	60%	100%	100%



5.4. Требования к отчету

В отчете должны быть отображены следующие пункты:

- Задание.
- Краткие теоретические сведения.
- Провести исследования согласно заданию. По результатам составить диаграмму.
- Выводы.

5.5. Контрольные вопросы

- Что такое дефект?
- Для чего необходимо описывать дефекты?
- Что такое жизненный цикл дефекта?

ЛАБОРАТОРНАЯ РАБОТА № 6. Автоматизированное тестирование Web-приложений

6.1 Цель работы: получить навыки автоматизированного тестирования web-приложений с использованием *SeleniumIDE*.

6.2. Краткие теоретические сведения

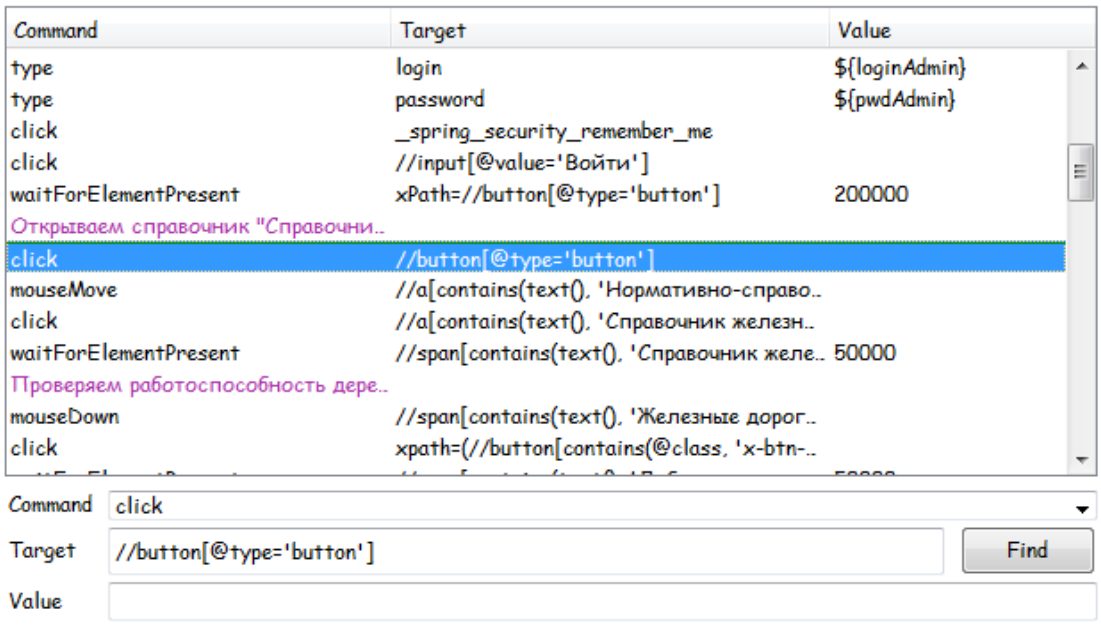
Selenium IDE (*Integrated Development Environment*, интегрированная среда разработки) – это инструмент, используемый для разработки тестовых сценариев (записи определенной последовательности действий). Он представляет собой простое в использовании дополнение к браузеру *Firefox* и, в целом, является наиболее эффективным способом разработки тестовых сценариев. Дополнение среди прочего содержит контекстное меню, позволяющее пользователю сначала выбрать любой элемент интерфейса на отображаемой браузером в данный момент странице, а затем выбрать команду из списка команд *Selenium* с параметрами, предустановленными в соответствии с выбранным элементом. Это не только экономит время, но и дает замечательную возможность для изучения языка команд *Selenium*.

Во время записи *Selenium IDE* автоматически вставляет команды в тестовый сценарий, основываясь на действиях пользователя. Обычно это команды:

- при нажатии на ссылку – команды *click* или *clickAndWait*;
- при вводе данных – команда *type*;
- при выборе опции из выпадающего списка – команда *select*;
- при нажатии на чекбокс или переключатель – команда *click.*

В тестовых сценариях бывает необходимо выполнить проверку параметров веб-страницы. Для этого необходимы команды *assert* и *verify*.

При включенном в *Selenium IDE* режиме записи, переключитесь на браузер с тестируемым веб-приложением и щелкните правой кнопкой мыши в любом месте на странице. Появится контекстное меню с командами *verify* и/или *assert*.



6.3. Задания

Вариант 1

1. Зайти на главную страницу сайта vilka.by.
2. Проверить, что главная страница содержит текст “Календарь событий” и “Сегодня с вами работает”.
3. Проверить, что главная страница содержит ссылку с названием “Аксессуары” на страницу с аксессуарами.
4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).
5. Создать переменную со значением “золотая рыбка”. На главной странице сайта сделать активным поле ввода “Поиск”, в поле для поиска внести значение переменной и нажать кнопку найти. Посчитать и вывести количество результатов поиска (использовать команду storeXpathCount).
6. Если на главной странице присутствует кнопка/ссылка “Вход”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует кнопка “Выйти”, нажать кнопку “Обо мне” и на открытой странице выбрать аватар и сохранить изменения.
7. Проверить работу кнопки/ссылки “Выйти”.

Вариант 2

1. Зайти на главную страницу сайта http://kartaby.by/.
2. Проверить, что главная страница содержит текст “Последнее на форуме” и “Дорогие Друзья”.
3. Проверить, что главная страница содержит ссылку с названием “Форум” на соответствующую страницу.
4. Проверить, что на главной странице есть хотя бы одна кнопка (button).
5. Перейти по ссылкам “Форум”-> “Поиск”. Создать переменную со значением “дорога”. Ввести в поле “Ключевые слова” значение переменной и нажать кнопку “Найти”. Посчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).
6. Если на главной странице присутствует кнопка/ссылка“Войти”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует кнопка “Покинуть”, перейти по ссылке “Входящие”в верхнем левом углу и проверить, что на открытой странице присутствует текст “В папке “Входящие” нет сообщений”.
7. Проверить работу кнопки/ссылки“Покинуть”.

Вариант 3

1. Зайти на главную страницу сайта onliner.by.
2. Проверить, что главная страница содержит текст “Мобильные телефоны” и “Вакансии”.
3. Проверить, что главная страница содержит ссылку с названием “Каталог” на страницу с каталогом.
4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).
5. Создать переменную со значением “велосипед”. На главной странице сайта сделать активным поле ввода для поиска, в поле для поиска внести значение переменной.Посчитать и вывести количество результатов поиска на первой странице (использовать команду storeXpathCount).

6. Если на главной странице присутствует кнопка/ссылка“Вход”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует кнопка/ссылка “Выйти”, сохранить в новую переменную номер вашей учетной записи и вывести его в Log.

7. Проверить работу кнопки/ссылки “Выйти”.

Вариант 4

1. Зайти на главную страницу сайта edu.gstu.by.

2. Проверить, что главная страница содержит текст “Учебный портал ГГТУ имени П.О. Сухого” и “Календарь”.

3. Проверить, что главная страница содержит ссылку с названием “edu.gstu.by” на главную страницу.

4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).

5. Создать переменную со значением “Тестирование и верификация ПО”. Внизу главной странице сайта сделать активным поле ввода “Поиск курса”, в поле для поиска внести значение переменной и нажать кнопку “Применить”. Убедиться, что отображаются результаты поиска. Сохранить в новую переменную категорию курса и вывести его в Log.

6. Если на главной странице присутствует кнопка/ссылка“Вход”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствуют имя и фамилия пользователя, т.е. пользователь авторизован,посчитать и вывести количество курсов у авторизованного пользователя (использовать команду storeXpathCount).

7. Проверить работу кнопки/ссылки “Выход”.

Вариант 5

1. Зайти на главную страницу сайта mail.ru.

2. Проверить, что главная страница содержит текст “Одноклассники” и “Спорт”.

3. Проверить, что главная страница содержит ссылку с названием “Ответы” на соответствующую страницу.

4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).

5. Создать переменную со значением “велосипед”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменной, выбрать категорию “приложения” и нажать кнопку “Найти”. Подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).

6. Если на главной странице присутствует кнопка/ссылка “Войти”/”Вход”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным. Если на главной странице присутствует кнопка “Выход”, т.е. пользователь авторизован, вывести количество входящих писем в его почтовом ящике.

7. Проверить работу кнопки/ссылки “Выход”.

Вариант 6

1. Зайти на главную страницу сайта rambler.ru.

2. Проверить, что главная страница содержит текст “как открыть сберегательный вклад”.

3. Проверить, что главная страница содержит ссылку с названием “Пророчества” на соответствующую страницу.

4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).

5. Создать переменную со значением “велосипед”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменной, выбрать категорию “поиск по сайту” и нажать кнопку “Найти”. Подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).

6. Если на главной странице присутствует кнопка/ссылка “Войти”/“Вход”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует имя пользователя, т.е. пользователь авторизован, вывести текущую температуру в городе, указанную на главной странице.

7. Проверить работу кнопки/ссылки “Выйти”.

Вариант 7

1. Зайти на главную страницу сайта yandex.by.
2. Проверить, что главная страница содержит текст “Телепрограмма” и“Афиша”.
3. Проверить, что главная страница содержит ссылку с названием “Маркет” на соответствующую страницу.
4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).
5. Создать переменную со значением “велосипед”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменнойи нажать кнопку “Найти”. Подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).
6. Если на главной странице присутствует кнопка/ссылка “Войти”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует имя пользователя/почта, т.е. пользователь авторизован, вывести количество писем, отображающихся во входящих на первой странице (в почтовом ящике).
7. Проверить работу кнопки/ссылки “Выйти”.

Вариант 8

1. Зайти на главную страницу сайта shop.by.
2. Проверить, что главная страница содержит текст “торговый портал”.
3. Проверить, что главная страница содержит ссылку с названием “Авто” на соответствующую страницу.
4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).
5. Создать переменную со значением “велосипед”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменнойи нажать кнопку “Найти”. Подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).

6. Если на главной странице присутствует кнопка/ссылка “Войти”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует имя пользователя/почта, т.е. пользователь авторизован, зайти в раздел “Мои отзывы на товары” и проверить, что значение заголовкаоткрытой страницы совпадает с “МойShop.by – Мои отзывы на товары”.

7. Проверить работу кнопки/ссылки “Выйти”.

Вариант 9

1. Зайти на главную страницу сайта http://www.ikea.com/ru/ru/.
2. Проверить, что главная страница содержит текст “Добро пожаловать в ИКЕА Россия!”.
3. Проверить, что главная страница содержит ссылку с названием “Список покупок” на соответствующую страницу.
4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).
5. Создать переменную со значением “ковер”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменной и нажать кнопку “Найти”. Подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).
6. Если на главной странице присутствует кнопка/ссылка “Вход”, осуществить вход в систему со своими логином и паролем. Убедиться, что при навигации по сайту пользователь остается авторизованным. Если на главной странице присутствует кнопка/ссылка“Выход”, т.е. пользователь авторизован, добавить в список покупок любой товар и после этого проверить его наличие в списке покупок.
7. Проверить работу кнопки/ссылки “Выход”.

Вариант 10

1. Зайти на главную страницу сайта https://www.21vek.by/.
2. Проверить, что главная страница содержит текст “онлайн-гипермаркет”.
3. Проверить, что главная страница содержит ссылку с названием “детям” на соответствующую страницу.

4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).

5. Создать переменную со значением “стиральная машина”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменнойи нажать кнопку “Найти”. В результатах поиска выбрать категорию “Samsung”, подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).

6. Если на главной странице присутствует кнопка/ссылка “Войти”, осуществить вход в систему. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует кнопка/ссылка “Мое”, т.е. пользователь авторизован, добавить в корзину любой товар и после этого проверить его наличие в корзине.

7. Проверить работу кнопки/ссылки “Выход”.

Вариант 11

1. Зайти на главную страницу сайта http://5element.by/.

2. Проверить, что главная страница содержит текст “единый мобильный номер”.

3. Проверить, что главная страница содержит ссылку с названием “Клуб покупателей”на соответствующую страницу.

4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).

5. Создать переменную со значением “стиральная машина”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменнойи запустить поиск. Подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).

6. Если на главной странице присутствует кнопка/ссылка “Войти в личный кабинет”, осуществить вход в систему. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует кнопка/ссылка “Личный кабинет”, т.е. пользователь авторизован, добавить в корзину любой товар и после этого проверить его наличие в корзине.

7. Проверить работу кнопки/ссылки “Выйти”.

Вариант 12

1. Зайти на главную страницу сайта http://zht.by/.
2. Проверить, что главная страница содержит текст “Служба поддержки”.
3. Проверить, что главная страница содержит ссылку с названием “бытовая химия” на соответствующую страницу.
4. Проверить, что на главной странице сайта есть хотя бы одно поле ввода (input) и одна кнопка (button).
5. Создать переменную со значением “шампунь”. Сделать активным поле ввода для поиска, в поле для поиска внести значение переменнойи запустить поиск. Подсчитать и вывести количество результатов на первой странице (использовать команду storeXpathCount).
6. Если на главной странице присутствует кнопка/ссылка “Вход”, осуществить вход в систему. Убедиться, что при навигации по сайту пользователь остается авторизованным.Если на главной странице присутствует кнопка/ссылка “Личный кабинет”/”Выход”, т.е. пользователь авторизован, добавить в корзину любой товар и после этого проверить его наличие в корзине.
7. Проверить работу кнопки/ссылки “Выход”.

6.4. Требования к отчету

В отчете должны быть отображены следующие пункты:

- Задание.
- Краткие теоретические сведения.
- Описать выполнение каждого пункта задания варианта с прикреплением результатов в виде скриншотов.
- Выводы.

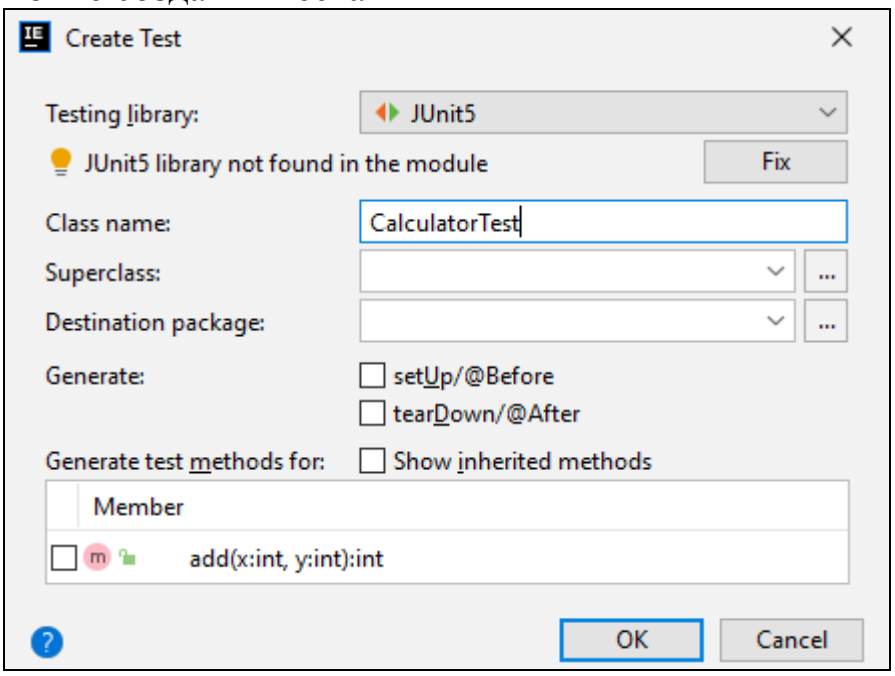
6.5. Контрольные вопросы

- Что такое SeleniumIDE, для чего нужен
- Способы создания тестовых сценариев в SeleniumIDE.

Практическое задание 1. Создание простого приложения с Unit-тестом

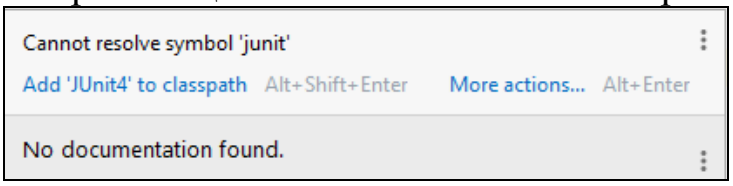
1. Создаем maven – проект
2. Создаем класс Calculator в папку /src/main/java/
3. Добавляем в класс метод сложения целых чисел:
   int add(int x, **int** y) {**return** x+y;}

4. Стать курсором на класс Calculator и набрать комбинацию Ctrl+Shift+T для создания теста и выбрать пункт Create New Test.
5. В окне создания теста



Поставить галочку для генерации теста для метода add и нажат ОК.

6. После нажатия ОК нажать Alt+Enter и в появившемся окне выбрать опцию “Add Unit<xx> to classpath”



Результат – класс Junit<xx> (xx- версия класса) будет добавлен в проект. Аналогично добавляем все требуемые и запрашиваемые классы (jupiter).

7. В методе add класса теста CalculatorTest создаем объект функционального класса Calculator, вызовем метод add функционального класса. А затем выполним сверку результата исполнения с ожидаемым, используя метод assertEquals класса Assert. Код теста приведен ниже:

```
class CalculatorTest {

    @org.junit.jupiter.api.Test
    void add() {
        Calculator calculator = new Calculator();
        int excpected=calculator.add( x: 2, y: 3);
        int actual=6;
        Assert.assertEquals(excpected,actual);
    }
}
```

8. Для запуска теста выбрать в окне проекта строку с классом теста, затем клик правой кнопкой и выбрать пункт «Выполнить CalculatorTest». Так как ожидаемый и фактический результаты не совпали, результат – тест не прошел:

```
java.lang.AssertionError: expected:<5> but was:<6>
Expected :5
Actual   :6
<Click to see difference>

<1 internal line>
    at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>
    at CalculatorTest.add(CalculatorTest.java:12) <31 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
```

9. Изменим actual значение на правильное – 5 и повторно запустим тест. Тест прошел успешно.

```
Process finished with exit code 0
```

Практическое задание 2. Аннотации jUnit

1. Используем проект из практического задания № 1
2. Вставим нижеприведенный код в код класса CalculatorTest.

```
@BeforeAll
static void beforeAll() {
   System.out.println("Before all tests (once)");
}

@BeforeEach
void beforeEach() {
   System.out.println("Runs before each test");
}

@Test
void standardTest() {
   System.out.println("Test is running");
}
@DisplayName("My #2 JUnit5 test")
@Test
void testWithCustomDisplayName() {
   System.out.println("Test is running");
}
@DisplayName("Tagged JUnit5 test ")
@Tag("cool")
@Test
void tagged() {
   System.out.println("Test is running");
}
@Disabled("Failing due to unknown reason")
@DisplayName("Disabled test")
@Test
void disabledTest() {
   System.out.println("Disabled, will not show up");
}
@DisplayName("Repeated test")
@RepeatedTest(value = 2, name = "#{currentRepetition} of {totalRepe-
titions}")
void repeatedTestWithRepetitionInfo() {
```

```
    System.out.println("Repeated test");
}

@AfterEach
void afterEach() {
    System.out.println("Runs after each test");
}
@org.junit.jupiter.api.Test
public void add() {
    Calculator calculator = new Calculator();
    int excpected = calculator.add(2, 3);
    int actual = 5;
    Assert.assertEquals(excpected, actual);
    System.out.println("@Test");
}
```

3. Как и в практическом задании 1 разрешим maven импорт необходимых классов. Фактическое значение результата исправим со значения 5 на 6.
4. Как и в приложении 1 запустим класс теста на исполнение. Фрагмент вывода в консоль показал, что тест не прошел.

```
java.lang.AssertionError: expected [6] but found [5]

    at org.testng.Assert.fail(Assert.java:110)
    at org.testng.Assert.failNotEquals(Assert.java:1413)
    at org.testng.Assert.assertEqualsImpl(Assert.java:149)
    at org.testng.Assert.assertEquals(Assert.java:131)
    at org.testng.Assert.assertEquals(Assert.java:1240)
    at org.testng.Assert.assertEquals(Assert.java:1274)
    at CalcTest.add(CalcTest.java:57) <31 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>
```

Дополнительная выдача в консоль показывает порядок прохождения выполнения новых методов.

5. Исправим ожидаемый результат на 5. Получим успешное прохождение теста.

Практическое задание 3. Подключение пакета TestNG

Используем проект из практического задания № 1.

1. Добавим в файл описания класса теста строку:

```
import org.testng.annotations.Test;
```

2. Вставим нижеприведенный код в код класса CalculatorTest.

```
import org.testng.Assert;
import org.testng.annotations.Test;

import static org.testng.Assert.*;
import org.testng.annotations.Test;

public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calculator = new Calculator();
        int expected = calculator.add(2, 3);
        int actual = 5;
        Assert.assertEquals(expected, actual);
        System.out.println("@Test");
    }
}
```

3. Как и в практическом задании 1 разрешим maven импорт необходимых классов. Фактическое значение результата исправим со значения 5 на 6.

4. Как и в приложении 1 запустим класс теста на исполнение.

   Фрагмент вывода в консоль показал, что тест не прошел.

```
java.lang.AssertionError:
Expected :6
Actual   :5
<Click to see difference>
```

```
================================================
Default Suite
Total tests run: 3, Passes: 2, Failures: 1, Skips: 0
================================================
```

5. Исправим ожидаемый результат на 5. Получим успешное прохождение теста.

```
================================================
Default Suite
Total tests run: 3, Passes: 3, Failures: 0, Skips: 0
================================================
```

Практическое задание 4. Аннотации TestNG

Используем проект из практического задания № 1.

1. Добавим в файл описания класса теста следующие строки:

```
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterGroups;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.AfterSuite;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeGroups;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;
```

2. Вставим нижеприведенный код в код класса CalculatorTest.

```
/**
 * @BeforeSuite method is executed before
 * any tests in the TestNG suite.
 */
@BeforeSuite
public void beforeSuite(){
   System.out.println("In Before Suite method");
}

/**
 * @AfterSuite method gets executed after
 * execution of all the tests in a TestNG suite.
 */
@AfterSuite
public void afterSuite(){
   System.out.println("In After Suite method");
}

/**
```

```
 * @BeforeTest method is executed before the first
 * test-method mentioned in each test inside the '<test>'
 * tag in test TestNG suite.
 */
@BeforeTest
public void beforeTest(){
   System.out.println("In Before Test method");
}

/**
 * @AfterTest method gets executed after
 * the last test-method
 */
@AfterTest
public void afterTest(){
   System.out.println("In After Test method");
}

/**
 * @BeforeClass method gets executed before
 * any of the test-methods inside a test class.
 */
@BeforeClass
public void beforeClass(){
   System.out.println("In Before Class method");
}

/**
 * @AfterClass method gets executed after
 * all of the test-methods inside a test class gets executed.
 */
@AfterClass
public void afterClass(){
   System.out.println("In After Class method");
}

/**
 * @BeforeGroups method is executed before executing any of
 * the tests belonging to the group in the 'groups'
```

```
 * attribute.
 * This method is executed before execution of the
 * test-method belonging to the group "groupOne".
 */
@BeforeGroups(groups={"groupOne"})
public void beforeGroupOne(){
   System.out.println("In Before Group One Test method");
}

/**
 * @AfterGroups method is executed after executing all the
 * tests belonging to the group as mentioned in the 'groups'
 * attribute.
 * This method gets executed after execution of the
 * test-methods belonging to group "groupOne".
 */
@AfterGroups(groups={"groupOne"})
public void afterGroupOne(){
   System.out.println("In After Group One Test method");
}

/**
 * This method is executed before execution of the
 * test-method belonging to group "groupTwo".
 */
@BeforeGroups(groups={"groupTwo"})
public void beforeGroupTwo(){
   System.out.println("In Before Group Two Test method");
}

/**
 * The following method gets executed after execution of the
 * test-methods belonging to group "groupTwo".
 */
@AfterGroups(groups={"groupTwo"})
public void afterGroupTwo(){
   System.out.println("In After Group Two Test method");
}
```

```
/**
 * @BeforeMethod gets executed before each test-method.
 */
@BeforeMethod
public void beforeMethod(){
    System.out.println("In Before Method");
}

/**
 * @AfterMethod gets executed after each test-method.
 */
@AfterMethod
public void afterMethod(){
    System.out.println("In After Method");
}

/**
 * @Test method belongs to the group "groupOne".
 */
@Test(groups={"groupOne"})
public void testMethod1(){
    System.out.println("In Test one method");
}

/**
 * @Test method belongs to the group "groupTwo".
 */
@Test(groups={"groupTwo"})
public void testMethod2(){
    System.out.println("In Test two method");
}
```

3. Как и в практическом задании 1 разрешим maven импорт необходимых классов. Фактическое значение результата исправим на 6.

4. Как и в приложении 1 запустим класс теста на исполнение.

   Фрагмент вывода в консоль показал, что тест не прошел.

```
java.lang.AssertionError:
Expected :6
Actual   :5
<Click to see difference>
```

```
================================================
Default Suite
Total tests run: 3, Passes: 2, Failures: 1, Skips: 0
================================================
```

Дополнительная выдача в консоль показывает порядок прохождения выполнения новых методов.

5. Исправим ожидаемый результат на 5. Получим успешное прохождение теста.

```
================================================
Default Suite
Total tests run: 3, Passes: 3, Failures: 0, Skips: 0
================================================
```

Список использованных источников

1. Святослав Куликов «Тестирование программного обеспечения. Базовый курс». 11.10.2017г.
2. Р. Ошеров. Искусство автономного тестирования с примерами на C#. Москва, 2014г.
3. В. Ф. Велесницкий. ПРАКТИКУМ по выполнению лабораторных работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения
4. М. М. Меженная, Т. В. Гордейчук, М. М. Борисик, О. С. Медведев, И.Ф. Киринович Тестирование, оценка программного обеспечения. Минск БГУИР 2016. – 64с.
5. Романькова Т.Л. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ. КУРС ЛЕКЦИЙ по одноименной дисциплине для слушателей специальности 1-40 01 73 «Программное обеспечение информационных систем» заочной формы обучения
6. Электронный ресурс Junit, User Guide:

   https://junit.org/junit5/docs/current/user-guide/
7. Электронный ресурс Юнит 5 – Основы:

   https://coderlessons.com/articles/java/iunit-5-osnovy
8. Электронный ресурс «Введение в Unit 5»:

   https://russianblogs.com/article/82541594978/
9. Электронный ресурс TestNG: https://testng.org/doc/
10. Электронный ресурс: TestNG – Краткое руководство https://coderlessons.com/tutorials/java-tekhnologii/uchitsia-testng/testng-kratkoe-rukovodstvo

Приложение. Предметные области

Таблица 1

Варианты предметных областей при выполнении заданий

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
1. Разработка ИС «Ведомость начисления зарплаты»	1. Цех, участок 2. Ф.И.О. 3. Объем выполнен. работы 4. Расценки на единицу продукции 5. Начисляемый заработок	Объем выполненной работы	Ф.И.О., совпадение
2. Разработка ИС «Список оборудования»	1. Кафедра 2. Наименование 3. Количество 4. Стоимость единицы продукции 5. Общая стоимость	Количество	Стоимость, принадлежность диапазону
3. Разработка ИС «Библиотечный каталог»	1. УДК (универсальный десятичный код) 2. Ф.И.О. автора 3. Наименование 4. Стоимость 5. Количество	Стоимость	Наименование, вхождение подстроки в строку

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
4. Разработка ИС «Железнодорожное расписание»	1. Номер поезда 2. Наименование 3. Время прихода 4. Время отправления 5. Категория(скорый пассажирский)	Номер поезда	Номер поезда, поиск в отсортированном массиве
5. Разработка ИС «Протокол соревнований»	1. Организация 2. Ф.И.О. 3. Год рождения 4. Результат 5. Занятое место	Год рождения	Ф.И.О., совпадение
6. Разработка ИС «Записная книжка-еженедельник»	1. День недели 2. Время суток 3. Мероприятие 4. Ф.И.О. 5. Телефон	Мероприятие	Время суток, принадлежность диапазону

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
7. Разработка ИС «Географический справочник»	1. Страна-название 2. Площадь, в млн. кв.км 3. Население, в млн. чел. 4. Континент 5. Столица	Население	Столица, вхождение подстроки в строку
8. Разработка ИС «Телефонный справочник»	1. Ф.И.О. 2. Адрес-улица 3. Номер дома 4. Организация 5. Телефон	Телефон	Телефон, поиск в отсортированном массиве
9. Разработка ИС «Нефтяные емкости»	1. Типовое название емкости 2. Дата измерения 3. Уровень в метрах 4. Вид топлива 5. Объем топлива	Объем топлива	Вид топлива, совпадение

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
10. Разработка ИС «Справочник автоинспектора»	1. Марка автомобиля 2. Номер 3. Цвет 4. Год выпуска 5. Пробег в км	Год выпуска	Пробег в км, принадлежность диапазону
11. Разработка ИС «Товары на складе»	1. Дата 2. Вид товара 3. Количество 4. Цена за единицу 5. Общая стоимость	Цена за единицу	Вид товара, вхождение подстроки в строку
12. Разработка ИС «Справочник агронома»	1. Название культуры 2. Название удобрения 3. Норма внесения на 100 $м^2$ 4. Дата внесения (по месяцам) 5. Содержание в ед. объема (в одном метре)	Норма внесения	Норма внесения, поиск в отсортированном массиве

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
13. Разработка ИС «Правонарушения»	1. Фамилия субъекта 2. Год рождения 3. Вид правонарушения 4. Дата нарушения 5. Сумма вознаграждения	Год рождения	Фамилия субъекта., совпадение
14. Разработка ИС «Протокол технического эксперимента»	1. Дата проведения эксперимента 2. Вид оборудования 3. Время начала эксперимента 4. Время окончания эксперимента	Вид оборудования	Время начала эксперимента, принадлежность диапазону
15. Разработка ИС «Справочник кинолога»	1. Порода собаки 2. Отец 3. Мать 4. Дата рождения 5. Рост	Рост	Порода собаки, вхождение подстроки в строку

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
16. Разработка ИС «Памятка туристу»	1. Район маршрута 2. Вид туризма 3. Категория (от 1 до 6) 4. Стоимость проезда 5. Вес снаряжения на одного участника	Стоимость проезда	Стоимость проезда , поиск в отсортированном массиве
17. Разработка ИС «Памятка дачнику-овощеводу»	1. Вид овощей 2. Название сорта 3. Дата посадки 4. Дата уборки урожая 5. Урожай - в кг на кв.м.	Вид овощей	Название сорта., совпадение
18. Разработка ИС «Памятка преподавателю»	1. Название дисциплины 2. Объем лекций, в часах 3. Объем лабораторных занятий, в часах 4. Вид контроля - зачет, экзамен 5. Количество студентов	Объем лекций, в часах	Объем лекций, в часах, принадлежность диапазону

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
19. Разработка ИС «Дневник метеонаблюдений»	1. Дата 2. Температура 3. Давление 4. Обланость (ясно, слабая, сильная, дождь) 5. Направление ветра (азимут)	Давление	Направление ветра, вхождение подстроки в строку
20. Разработка ИС «Справочник по транзисторам»	1. Тип транзистора 2. Напряжение питания 3. Допустимый ток 4. Коэффициент усиления 5. Стоимость	Стоимость	Стоимость, поиск в отсортированном массиве
21. Разработка ИС «Справочник по оборудованию»	1. Наименование 2. Страна-изготовитель 3. Стоимость 4. Вес, в кг 5. Объем, в куб. дм	Вес	Стоимость, совпадение

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
22. Разработка ИС «Великие люди»	1. Ф.И.О. 2. Область деятельности(ученый, полководец, общ. деятель, поэт, художник и т.д.) 3. Год рождения 4. Страна 5. Продолжительность жизни	Год рождения	Продолжительность жизни, принадлежность диапазону
23. Разработка ИС «Памятка любителю музыки»	1. Название группы 2. Страна 3. Фамилия руководителя 4. Количество записанных дисков 5. Общий тираж дисков	Общий тираж дисков	Наименование, вхождение подстроки в строку

Продолжение табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
24. Разработка ИС «Великие даты»	1. Дата 2. Страна 3. Вид события (война, революция, образование государства и т.д.) 4. Фамилия видного деятеля 5. Примерное число жертв	Примерное число жертв	Примерное число жертв, поиск в отсортированном массиве
25. Разработка ИС «Памятка альпинисту»	1. Название вершины 2. Высота 3. Страна расположения 4. Год покорения 5. Количество Восхождений	Объем выполнен. работы	Ф.И.О., совпадение
26. Разработка ИС «История персональных ЭВМ»	1. Тип микропроцессора 2. Дата появления 3. Фирма-изготовитель 4. Тактовая частота 5. Объем адресуемой памяти, в К байтах	Тактовая частота	Объем адресуемой памяти, принадлежность диапазону

Окончание табл. 1

Вариант	**Наименование полей**	**Ключ сортировки**	**Виды поиска, фильтра**
27. Разработка ИС «Живая планета»	1. Наименование животного 2. Рост (длина) в м 3. Вес (средний) в кг 4. Скорость передвижения (км/час) 5. Континент океан)	Скорость передвижения	Наименование животного, вхождение подстроки в строку
28. Разработка ИС «Медицинская карта»	1. Ф.И.О. 2. Год рождения 3. Рост 4. Вес 5. Группа крови	Год рождения, выбором	Год рождения, поиск в отсортированном массиве
29. Разработка ИС «Химические элементы»	1. Наименование 2. Атомный вес 3. Валентность 4. Плотность 5. Цвет	Атомный вес	Наименование, совпадение
30. Разработка ИС «Материалы»	1. Наименование 2. Плотность 3. Агрегатное состояние (твердое, жидкое, газообразное) 4. Электропроводность 5. Модуль упругости	Плотность,	Плотность, принадлежность диапазону

Таблица 2

Требования к данным при выполнении редактирования

Вариант ИС	Наименование полей	Требованию к данным
Оборудование кафедры	1. Кафедра 2. Наименование оборудования 3. Количество 4. Стоимость единицы продукции 5. Общая стоимость	1. C(4,10) 2. A(Принтер, ПК, Ноутбук) 3. I, I>=0 и I<=10 4. F, 0<= F<10000 5. F, 0<= F<100000
Разработка ИС «Библиотечный каталог»	1.Тип 2. Ф.И.О. автора 3. Наименование 4. Стоимость 5. Количество	1. A (художественная, техническая, публицистика) 2. C(10,30) 3. C(5,50) 4. F, 0<F<1000 5. I, 0<I<100
Разработка ИС «Железнодорожное расписание»	1. Номер поезда 2. Наименование 3. Время прихода 4. Время отправления 5. Категория (скорый пассажирский)	1. S(4,10) 2. C(10,30) 3. T 4. T 5. A (скорый, пассажирский)
Разработка ИС «Протокол соревнований»	1. Тип соревнований 2. Ф.И.О. 3. Год рождения 4. Результат 5. Занятое место	1. A (легкая атлетика, тяжелая атлетика, водное многоборье) 2. C(5,16) 3. I, 1990<I< 2010 4. F, F<100 5. I, I<100
Разработка ИС «Протокол соревнований»	1. Тип соревнований 2. Ф.И.О. 3. Год рождения 4. Результат 5. Занятое место	6. A (легкая атлетика, тяжелая атлетика, водное многоборье) 7. C(5,16) 8. I, 1990<I< 2010 9. F, F<100 10.I, I<100

Продолжение табл. 2

Вариант ИС	**Наименование полей**	**Требованию к данным**
Разработка ИС «Записная книжка-еженедельник»	1. День недели 2. Время начала 3. Мероприятие 4. Ф.И.О. 5. Стоимость	1. A (Пн, Вт, Ср, Чт, Пт, Сб.Вс) 2. T, 8.00<=T<=24.00 3. C (20,50) 4. C (4,20) 5. F, 0<=F<1000
Разработка ИС «Географический справочник»	1. Страна-название 2. Площадь, в млн. кв.км 3. Население, в млн.чел. 4. Континент 5. Столица	1. C (4, 50) 2. F, 0<F<100 3. F, 0<F<10000 4. A (Африка, Америка, ЕвраАзия, Австралия) 5. C (4,50)
Разработка ИС «Телефонный справочник»	1. Ф.И.О. 2. Адрес-улица 3. Номер дома 4. Организация 5. Телефон	1. C (4, 50) 2. C (10,50) 3. I, 0<I<1000 4. C (6,20) 5. I (7,10)
Разработка ИС «Нефтяные емкости»	1. Типовое название емкости 2. Дата измерения 3. Уровень в метрах 4. Вид топлива 5. Объем топлива	1) C (4, 50) 2) D, 2019.01.01<D<2019.12.12 3) I, 0<I<10 4) C(6,20) 5) I(10,100)
Разработка ИС «Справочник автоинспектора»	1. Марка автомобиля 2. Номер 3. Цвет 4. Год выпуска 5. Пробег в км	1. S (4,20) 2. S (6) 3. A (красный, синий, и т.д.) 4. I,1960<I<2020 5. I, 0<=I<=1000000
Разработка ИС «Товары на складе»	1. Тип товара 2. Наименование 3. Количество 4. Цена за единицу 5. Общая стоимость	1. A (продовольствие, одежда) 2. C (4,10) 3. I, 0<=I<1000 4. F, 0<F<1000 5. F, 0<=F<=1000000

Продолжение табл. 2

Вариант ИС	**Наименование полей**	**Требованию к данным**
Разработка ИС «Справочник агронома»	1. Название культуры 2. Название удобрения 3. Норма внесения на 100 м кв. 4. Дата внесения 5. Содержание в ед. объема (в одном метре)	1. A (рожь, пшеница, овес) 2. C(4,10) 3. F, 0<F<10.99 4. 2020.04.01<D<2020.06.01 5. F, 0<=F<=1.99
Разработка ИС «Правонарушения»	1. Фамилия субъекта 2. Год рождения 3. Вид правонарушения 4. Дата нарушения 5. Сумма штрафа	1) C (10,20) 2) 1900<I<2010 3) A (скорость, неисправность, прочее) 4) D, 2019<=Y<2021 5) F, 0<F<=260
Разработка ИС «Правонарушения»	1. Фамилия субъекта 2. Год рождения 3. Вид правонарушения 4. Дата нарушения 5. Сумма штрафа	1) C (10,20) 2) 1900<I<2010 3) A (скорость, неисправность, прочее) 4) D, 2019<=Y<2021 5) F, 0<F<=260
Разработка ИС «Протокол технического эксперимента»	1. Дата проведения 2. Тип оборудования 3. Оборудования 4. Время начала 5. Время окончания	1) D, 2019<=Y<2021 2) A (простое, сложное) 3) C (10,20) 4) 12.01<T<20.55 5) 12.01<T<20.55 Завершение позже чем начало
Разработка ИС «Справочник кинолога»	1. Порода собаки 2. Отец 3. Мать 4. Дата рождения 5. Рост	1) A(дог, овчарка, прочее) 2) C(5,20) 3) C(5,20) 4) D, 2010<=Y<2019 5) 10<I<200

Продолжение табл. 2

Вариант ИС	**Наименование полей**	**Требованию к данным**
Разработка ИС «Памятка туристу»	1. Район маршрута 2. Вид туризма 3. Категория (от 1 до 6) 4. Стоимость проезда 5. Вес снаряжения на одного участника	1) C (5,20) 2) A (горный, пеший, водный) 3) 1<=I<=6 4) 0.00<F<=100 5) 1<F<=25
Разработка ИС «Памятка дачнику-овощеводу»	1. Вид овощей 2. Название сорта 3. Дата посадки 4. Дата уборки урожая 5. Ожидаемый урожай - в кг на кв.м.	1) A (картофель, свекла) 2) C (5,20) 3) 2020.04.01<D<2020.05.31 4) 2020.06.01<D<2020.10.31 5) 0.01<F<100
Разработка ИС «Памятка преподавателю»	1. Название дисциплины 2. Объем лекций, в часах 3. Объем лабораторных занятий, в часах 4. Вид контроля зачет, экзамен 5. Количество студентов	1) C (4,20) 2) 12<I<64 3) 12<I<64 4) A (Зачет, Экзамен) 5) 0<I<40
Разработка ИС «Дневник метеонаблюдений»	1. Дата 2. Температура 3. Давление 4. Облачность (ясно, слабая, сильная, дождь) 5. Направление ветра (азимут)	1) 2019.12.12<D<2020.12.12 2) -40<I<40 3) 400<I<600 4) A (ясно, облачно, дождь) 5) 0<=I<360
Разработка ИС «Справочник по транзисторам»	1. Тип транзистора 2. Наименование 3. Напряжение питания 4. Допустимый ток 5. Коэффициент усиления	1) A (полевой, биполярный) 2) S (4,30) 3) 0.01<F<12 4) 0.01<F<1 5) 1<F<10

Продолжение табл. 2

Вариант ИС	**Наименование полей**	**Требованию к данным**
Разработка ИС «Справочник по оборудованию»	1. Наименование 2. Страна-изготовитель 3. Стоимость 4. Вес, в кг 5. Объем, в куб.дм	1) C (10,20) 2) A (РБ, Россия, Казахстан) 3) 1<F<100000 4) 0<F<1800 5) 0<F<2600
Разработка ИС «Великие люди»	1. Ф.И.О. 2. Область деятельности 3. Год рождения 4. Страна 5. Продолжительность жизни	1) C (10,40) 2) A(ученый, полководец, поэт, художник) 3) 1000<I<2000 4) C(4,20) 5) 20<I<100
Разработка ИС «Памятка любителю музыки»	1. Название группы 2. Страна 3. Фамилия руководителя 4. Количество записанных дисков 5. Общий тираж дисков	1) C (10,40) 2) A (ученый, полководец, поэт, художник) 3) 1000<I<2000 4) C(4,20) 5) 20<I<100
Разработка ИС «Великие даты»	1. Дата 2. Страна 3. Вид события 4. Фамилия видного деятеля 5. Примерное число жертв	1) 1000<D<2020 2) C (10,20) 3. A (война, революция, образование государства) 4) C(4,30) 5) 0<I<100000
Разработка ИС «Памятка альпинисту»	1. Название вершины 2. Высота 3. Страна расположения 4. Год покорения 5. Количество восхождений	1) C (10,20) 2) 7000<I<12000 3) A (Россия, Китай, Аргентина) 4) 1000<I<2020 5) 0<I<1000

Продолжение табл. 2

Вариант ИС	**Наименование полей**	**Требованию к данным**
Разработка ИС «История персональных ЭВМ»	1. Тип микропроцессора 2. Год появления 3. Фирма-изготовитель 4. Тактовая частота 5. Объем адресуемой памяти, в мегабайтах	1) C (10,20) 2) 1970<I<2020 3) A (США, Китай, Япония) 4) 1000<I<10000 5) 0.1<I<20000
Разработка ИС «Живая планета»	1. Наименование животного 2. Рост (длина) в м 3. Вес(средний) в кг 4. Скорость передвижения (км/час) 5. Среда обитания	1)C(10,20) 2) 0<I<99 3) 0<I<2000 4) 1<I<100 5) A(Континент/океан)
Разработка ИС «Медицинская карта»	1. Ф.И.О. 2. Год рождения 3. Рост 4. Вес 5. Группа крови	1)C(10,20) 2) 2000<I<2020 3) 0<I<200 4) 1<I<100 5) A(I, II, III, IV)
Разработка ИС «Химические элементы»	1. Тип 2. Наименование 3. Атомный вес 4. Валентность 5. Плотность	1) A(металлы, газы, прочие) 2) C(10,20) 3) 0<I<200 4) 0<I<20 5) 1<I<100
Разработка ИС «Материалы»	1. Наименование 2. Плотность 3. Агрегатное состояние (твердое, жидкое, газообразное) 4. Электропроводность 5. Модуль упругости	1) C (10,20) 2) 0<I<200 3) A (твердое, жидкое, газообразное) 4) 1<I<100 5) 0.01<F<10

Окончание табл. 2

«Ведомость начисления зарплаты»	1. Цех, участок 2. Ф.И.О. 3. Объем выполненной работы 4. Расценки на единицу продукции 5. Начисляемый заработок	1) A (Штамповочный, Сборочный) 2) C (10,50) 3) 1<I<100 4) 0.1<F<100 5) 0.1<F<10000

ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Практикум
по выполнению лабораторных работ
для студентов специальности 1-40 04 01
«Информатика и технологии программирования»
дневной формы обучения

Составитель **Шибеко** Виктор Николаевич

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 05.02.24.

Рег. № 67Е.
http://www.gstu.by