ФГБОУ ВО «Уфимский Университет Науки и Технологий»

Отчет по Лабораторной работе №2

«Настройка и использование библиотеки Junit для модульного тестирования программного обеспечения»

Вариант: «7»

Выполнили:

Ст. гр ПИ-101Бзу

Мурхайдарова Р.Э.

Романец Н.В.

Сафиуллин А.Ф.

Проверил:

Казанцев А.В.

Уфа - 2024

**Лабораторная работа №2.**

**Настройка и использование библиотеки Junit для модульного тестирования программного обеспечения.**

**Цель работы:** изучение библиотеки Junit для тестирования программного кода и создания веток, тэгов и выполнения слияния в Git.

Создание ветки

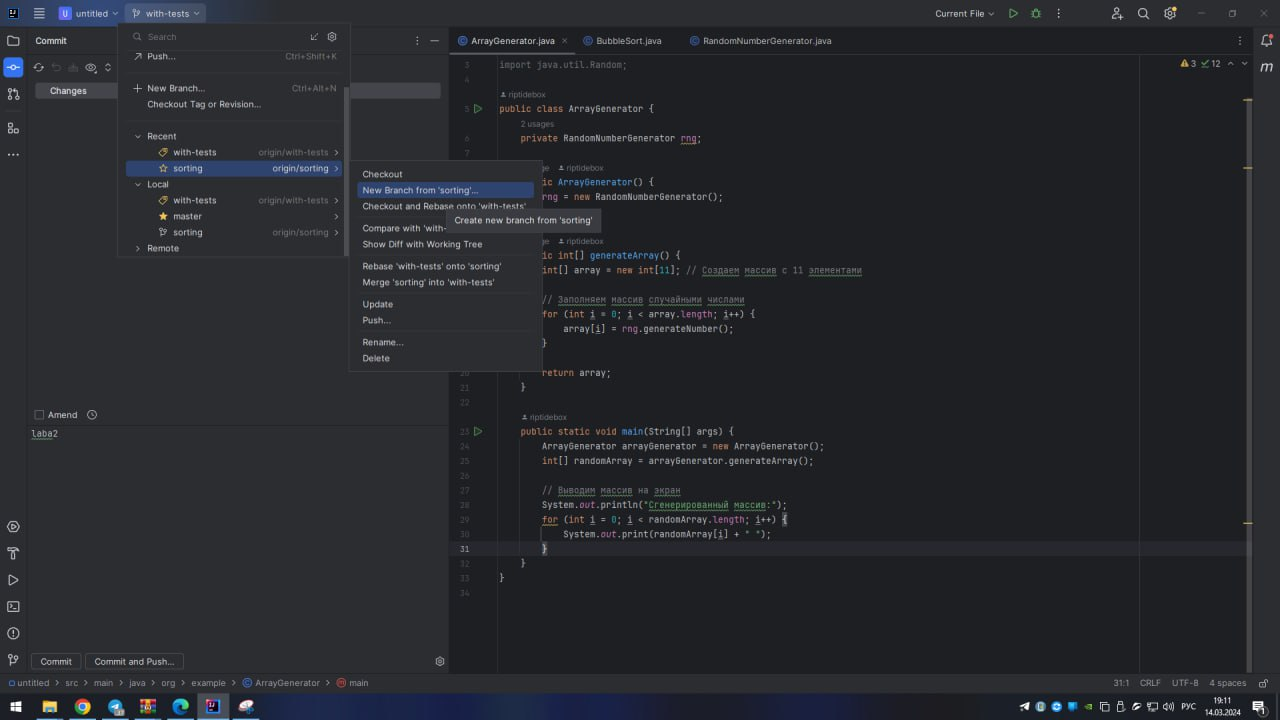


Рис 1 – Создание ветки

Добавление библиотеки JUnit 4.12

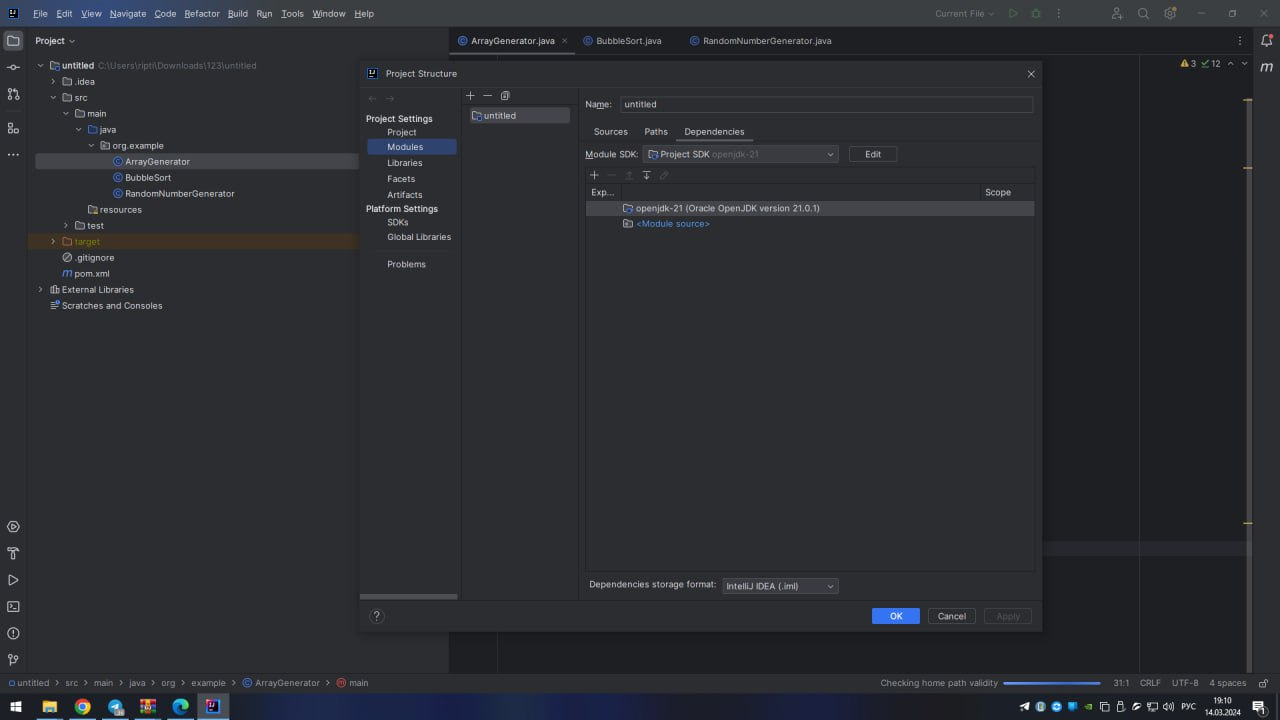


Рис 2 – Добавление библиотеки

Создать класс и соответствующий метод: генерирует случайные числа в диапазоне от 100 до 500.

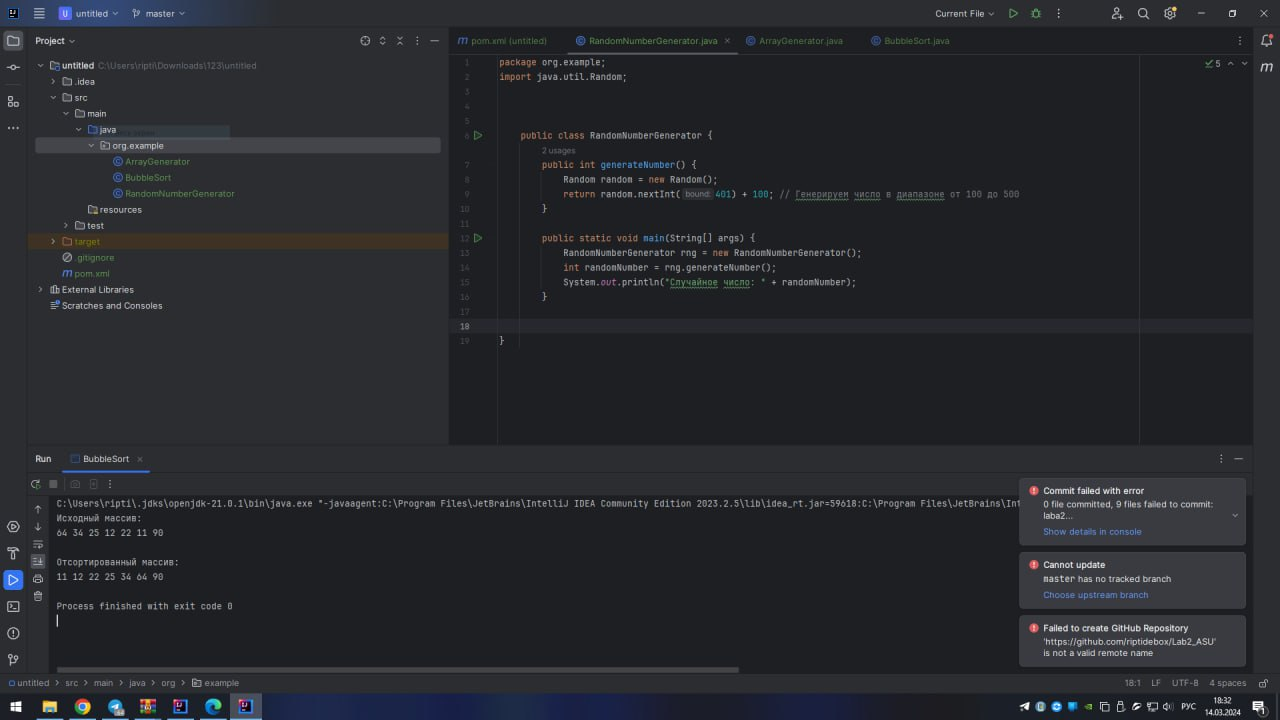


Рис 3 – Генерация случайных чисел в диапазоне

Создать класс и соответствующий метод: реализует создание одномерного массива с 11 элементами, которые генерируется случайно, используя созданный класс разработчика 1.

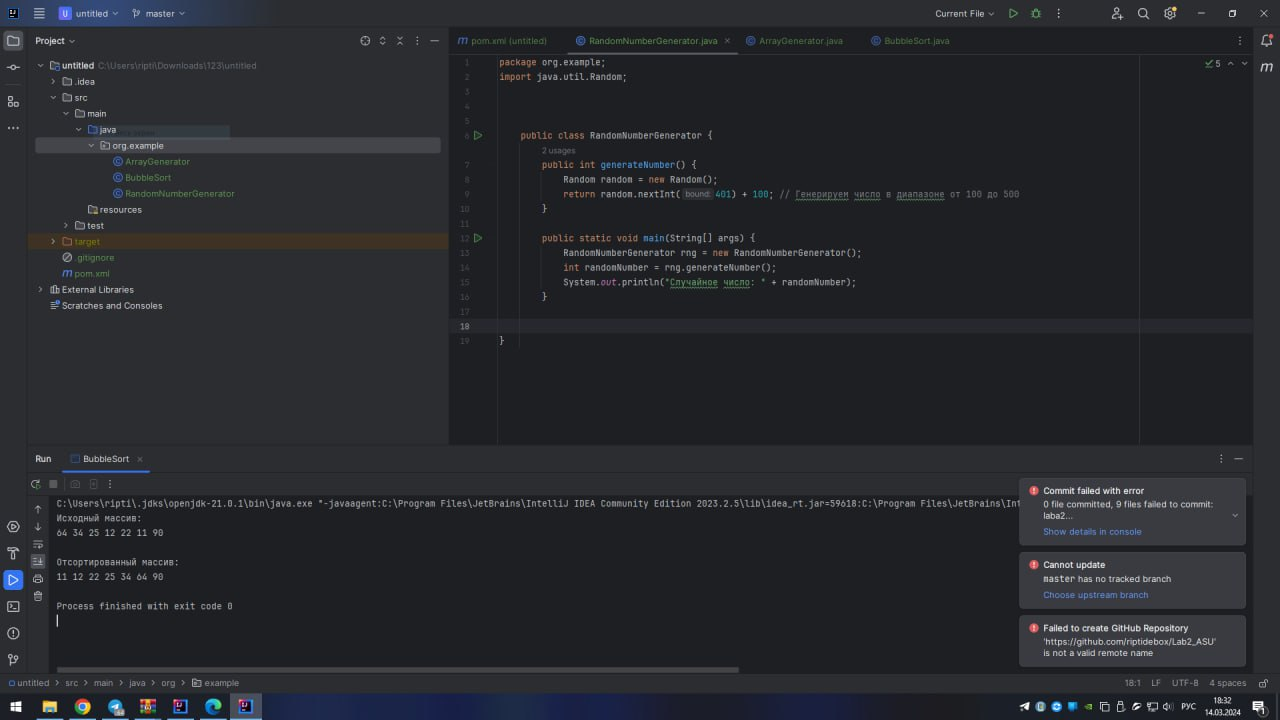


Рис 4 – Создание одномерного массива

Создать класс и соответствующий метод: реализует пузырьковую сортировку одномерного массива.

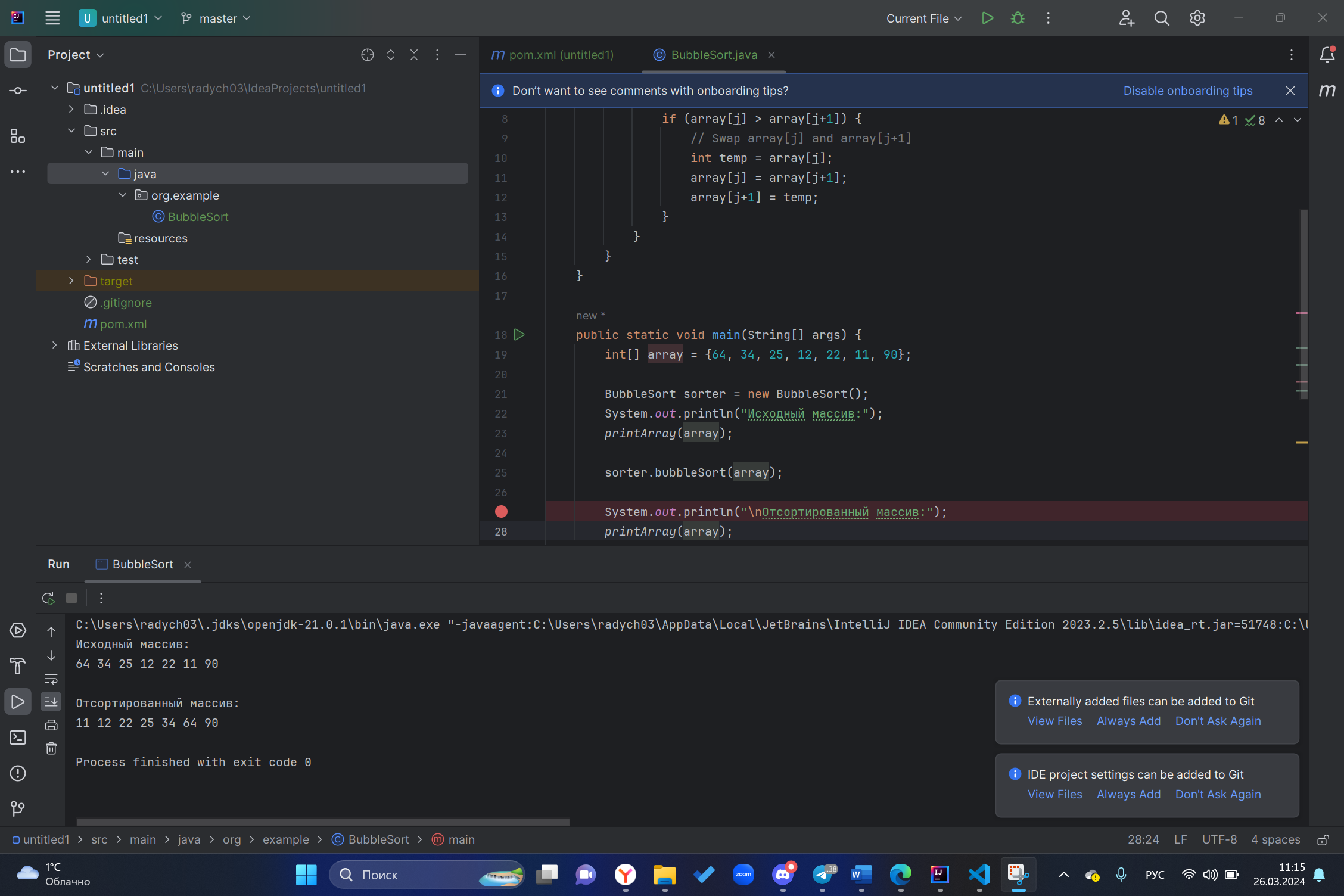


Рис 5 – Реализация пузырьковой сортировки

**Листинг программы Unit-теста**

@BeforeClass

public static void setUp() {

// Создание массива и сортировка перед выполнением всех тестов

ArrayGenerator arrayGenerator = new ArrayGenerator();

array = arrayGenerator.generateArray();

BubbleSort sorter = new BubbleSort();

sorter.bubbleSort(array);

}

@AfterClass

public static void tearDown() {

// Проверка корректности сортировки после выполнения всех тестов

for (int i = 0; i < array.length - 1; i++) {

assertTrue("Массив должен быть отсортирован", array[i] <= array[i + 1]);

}

}

@Test

public void testArrayLength() {

assertEquals("Размер массива должен быть 11", 11, array.length);

}

@Test

public void testArrayNotNull() {

assertNotNull("Массив не должен быть null", array);

}

@Test

public void testArrayElementsNotNull() {

for (int i = 0; i < array.length; i++) {

assertNotNull("Элемент массива не должен быть null", array[i]);

}

}

}

**Заключение**

В ходе лабораторной работы по теме сортировки массивов, работы с Git и использования юнит-тестов были получены ценные знания и навыки, необходимые для разработки программного обеспечения. С помощью изучения различных методов сортировки массивов, таких как сортировка пузырьком, сортировка вставками и сортировка выбором, студенты погрузились в основы алгоритмов сортировки и поняли их принципы работы и эффективность.

Работа с Git позволила учащимся овладеть навыками версионного контроля и совместной работы над проектами. Создание веток, тегов и выполнение слияния обогатили понимание процессов разработки программного обеспечения и улучшили координацию работы в команде. Использование юнит-тестов дало студентам возможность обеспечить качество своего кода и убедиться в его корректности. Тестирование позволяет обнаруживать ошибки на ранних стадиях разработки и обеспечивать надежность программного продукта.

Таким образом, лабораторная работа по сортировке массивов, работе с Git и использованию юнит-тестов оказалась не только практически полезной, но и обогатила студентов фундаментальными знаниями и навыками, которые пригодятся им в будущей профессиональной деятельности в области разработки программного обеспечения.

**Контрольные вопросы**

1. Что такое модульное тестирование?

Модульное тестирование — это метод тестирования программного обеспечения, при котором отдельные модули или компоненты программы тестируются независимо от остальной части системы. Целью модульного тестирования является проверка правильности работы отдельных функций, методов или классов программы, чтобы обеспечить их корректное функционирование в изоляции от других компонентов.

1. Что такое JUnit?

JUnit — это популярный фреймворк для модульного тестирования Java-приложений. Он предоставляет набор инструментов и классов для написания и запуска тестовых сценариев, а также автоматической проверки их результата. JUnit позволяет разработчикам создавать и запускать тесты для проверки корректности поведения отдельных частей Java-кода.

1. Как располагаются тесты в JUnit?

В JUnit тесты располагаются в классах, которые содержат методы, помеченные аннотацией @Test. Эти методы представляют собой отдельные тестовые случаи, которые проверяют определенные аспекты функциональности программы. JUnit обнаруживает и запускает тесты автоматически при запуске тестового процесса.

1. Что такое assert и assume?

* assert - это ключевое слово в JUnit, которое используется для проверки утверждений во время выполнения тестов. Если утверждение не выполняется, тест считается проваленным, и соответствующее сообщение об ошибке выводится в консоль.
* assume — это также ключевое слово, но в контексте библиотеки JUnit Jupiter. Оно позволяет предполагать определенные условия перед выполнением теста. Если предположение не выполняется, тест пропускается, а не считается проваленным.

1. В чем отличие ветки от метки?

* Ветка (Branch) в системе контроля версий, такой как Git, представляет собой копию репозитория, которая может быть изменена независимо от остальной части проекта. Ветки используются для разработки новых функций, исправления ошибок или изоляции экспериментального кода.
* Метка (Tag) в системе контроля версий служит для пометки определенной версии или состояния проекта. Метки обычно назначаются на определенные коммиты и используются для обозначения релизов или важных точек в истории проекта. В отличие от веток, метки обычно не изменяются после своего создания и используются для идентификации конкретных состояний проекта.