Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

**ЛАБОРАТОРНА РОБОТА № 4**

з дисципліни «Математичні методи та технології тестування та верифікації програмного забезпечення»

Тема: «Вивчення модульного тестувания (unit testing) на прикладі фреймворку JUnit.»

Виконав студент 2 курсу

групи КС-21

Безрук Юрій Русланович

Перевірив:

Доцент Нарєжній О. П.

Харків – 2020

Целью данной работы является изучение фреймворка для модульного тестировани JUnit, сборщиков проектов, написание автоматизированных тестов для программного обеспечения.

# ХОД РАБОТЫ

После установки IntelliJ IDEA и скачивания фреймворка JUnit, создаём проект Java и тестируем подключение библиотеки. Для этого, например, можно подключить ее к сборщику проектов Maven.

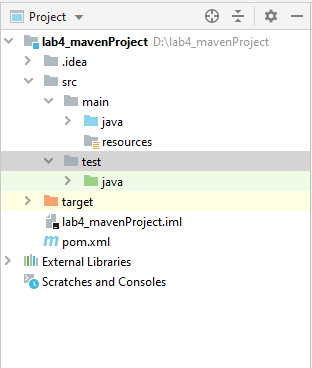


Рисунок 1 - дерево проекта Maven

Что бы подключить JUnit, необходимо подтянуть его в зависимости (те библиотеки, которые непосредственно используются в проекте для компиляции кода или его тестирования). В файле pom.xml прописываем зависимости:



Рисунок 2 - файл pom.xml

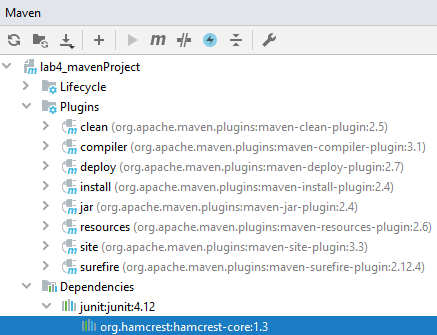
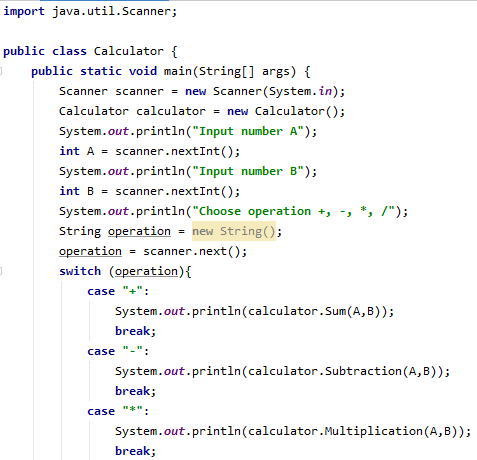


Рисунок 3 - зависимость JUnit

Для проверки возможностей фреймворка запустим пример калькулятора из задания.



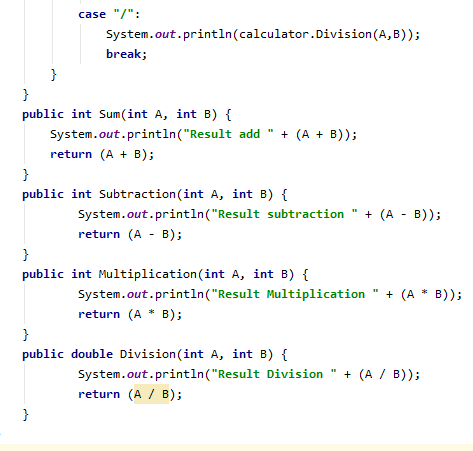


Рисунок 4 - код калькулятора

Для тестирования работы программы напишем тестовый класс.

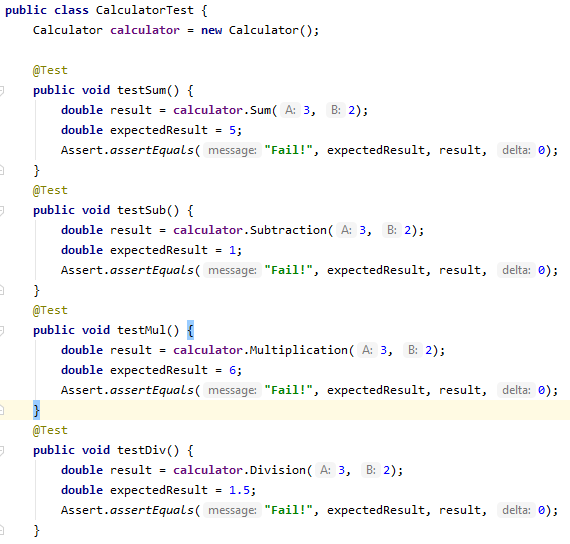


Рисунок 5 - тестовый класс

После запуска тестов обнаруживаем, что все методы прошли тест, кроме деления, т.к. в делении участвовали только целые числа, было выполнено целочисленное деление, поэтому получаем 3/2=1, вместо ожидаемого результата 1,5.



Рисунок 6 - результат теста калькулятора

Для выполнения некоторых действий перед вызовом метода-теста или даже класса-теста, в JUnit существуют соответственные аннотации: @Before, @After, @BeforeClass, @AfterClass.

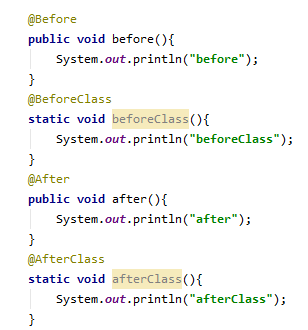


Рисунок 7 - методы с аннотациями

Результат выполнения теста приведен ниже.

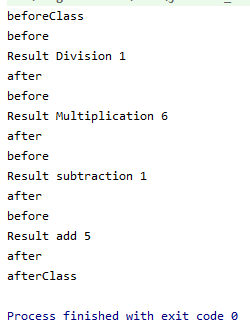


Рисунок 8 - результат работы методов с аннотациями

Далее приступаем к выполнению индивидуального задания, чётко сформулированные требования:

Программа должна уметь сортировать массив целых чисел типа int любого размера по возрастанию, в пределах размера целочисленного типа данных. Размер массива и сам массив вводятся с консоли. Тип возвращаемого значения - int[], не должен равняться null. В случае, если массив не отсортирован, после выполнения метода он точно должен измениться.

Программа была реализована через класс со статическими методами.

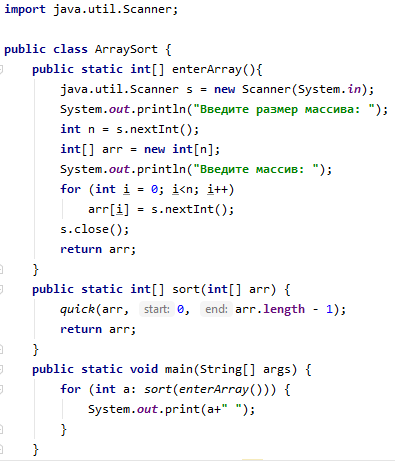


Рисунок 9 - методы ввода и сортировки

Сама сортировка производилась при помощи алгоритма QuickSort.

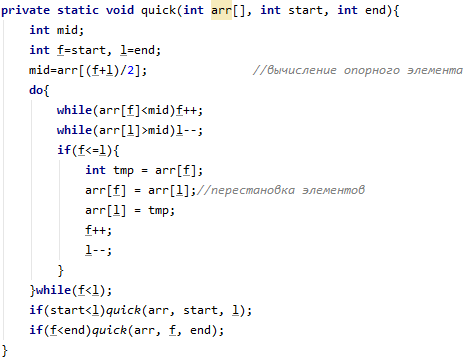


Рисунок 10 – «быстрая сортировка»

Помимо различных исключений, обрабатываемых Java, у программы может быть несколько семантических проблем. Для тестирования некоторых ситуаций создадим класс ArraySortTest.

Среди его методов-проверок:

Проверка на правильность сортировки. Выполняется при помощи сравнения с встроенным алгоритмом сортировки.

Проверка что возвращаемое значение не null.

Проверка на тип возвращаемого значения при помощи механизмов рефлексии Java.

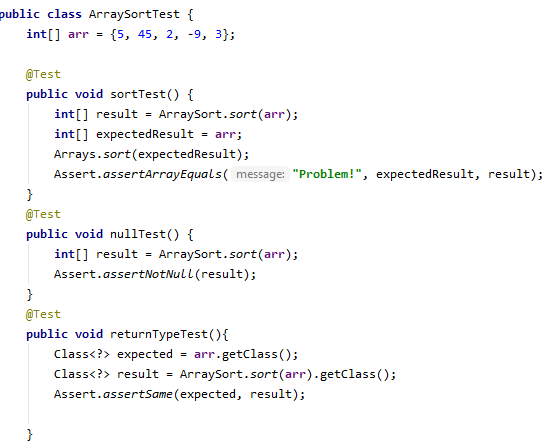


Рисунок 11 - класс-тест

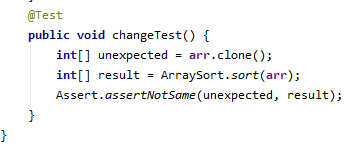


Рисунок 12 - класс-тест (продолжение)

В методах в качестве тестируемого массива используют заранее оглашенный, чтобы исключить зависимость от консоли.

В ходе тестирования не было обнаружено проблем с несоответствием требованиям.

# ВЫВОДЫ

Итак, модульное тестирование позволяет проводить тесты всей программы или ее отдельных компонентов (units). Такой уровень тестирования предполагает написание автоматизированных тестовых сценариев, которые выявляют дефекты на стадии разработки программного приложения и позволяют разработчику исправить ошибки сразу же без их занесения в базу дефектов. Библиотека JUnit используется для написания модульных тестов, которые повторяются. Она обеспечивает перегрузки методов для всех примитивных типов, объектов и массивов. Используемые параметры: ожидаемое значение (expectedResult) и актуальное значение (result). Возможно также вывод сообщения в случае неудачи теста.

Таким образом, в ходе выполнения лабораторной работы было изучено фрейморк для модульного тестирования JUnit, сборников проектов, написание автоматизированных тестов для программного обеспечения, а также написана программы «Калькулятор» и «Сортировщик»