**Báo cáo bài thực hành Junit5x**

**Học phần: Kiểm thử phần mềm**

**Họ tên:** Vũ Đức Vượng

**Msv:** 2017604374

1. **Junit 5x**

• Nghiên cứu JUnit5 và sự khác nhau giữa JUnit5 với JUnit4

• Các Annotation của JUnit5 và ví dụ

• Thực hiện Junit 5 trong Eclipse, NetBeans, IntelliJ, and Maven (pom.xml )

1. **Kiểm thử với Mockito và TestNG (Nghiên cứu thêm)**

**Bài làm**

1. **JUnit5x**

* Nghiên cứu JUnit5 và sự khác nhau giữa JUnit5 với JUnit4

- JUnit là một java testing framework được sử dụng rộng rãi trong các dự án java. JUnit 5 là phiên bản mới của JUnit với mục đích hỗ trợ các tính năng mới ra mắt từ Java 8 trở về sau.

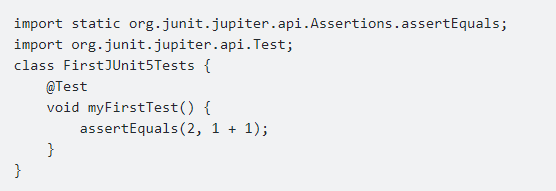
- JUnit 5 gồm có 3 module khác nhau từ 3 sub-project khác nhau:

* JUnit platform
* JUnit Jupiter
* JUnit Vintage

- Sự khác nhau giữa JUnit5 với JUnit4:

|  |  |  |
| --- | --- | --- |
| **ĐẶC TÍNH** | **JUNIT 4** | **JUNIT 5** |
| Khai báo một phương pháp kiểm tra | @Test | @Test |
| Thực thi trước tất cả các phương thức kiểm tra trong lớp hiện tại | @BeforeClass | @BeforeAll |
| Thực hiện sau khi tất cả các phương thức kiểm tra trong lớp hiện tại | @AfterClas | @AfterAll |
| Thực hiện trước mỗi phương pháp kiểm thử | @Before | @BeforeEach |
| Thực hiện sau mỗi phương pháp kiểm thử | @After | @AfterEach |
| Vô hiệu hóa một phương thức / lớp kiểm tra | @Ignore | @Disabled |
| Thử nghiệm các bài kiểm tra năng động | NA | @TestFactory |
| Các kiểm thử lồng nhau | NA | @Nested |
| Gắn thẻ và lọc | @Category | @Tag |
| Đăng ký tiện ích mở rộng tùy chỉnh | NA | @ExtendWith |

* Các Annotation của JUnit5 và ví dụ
* @Test

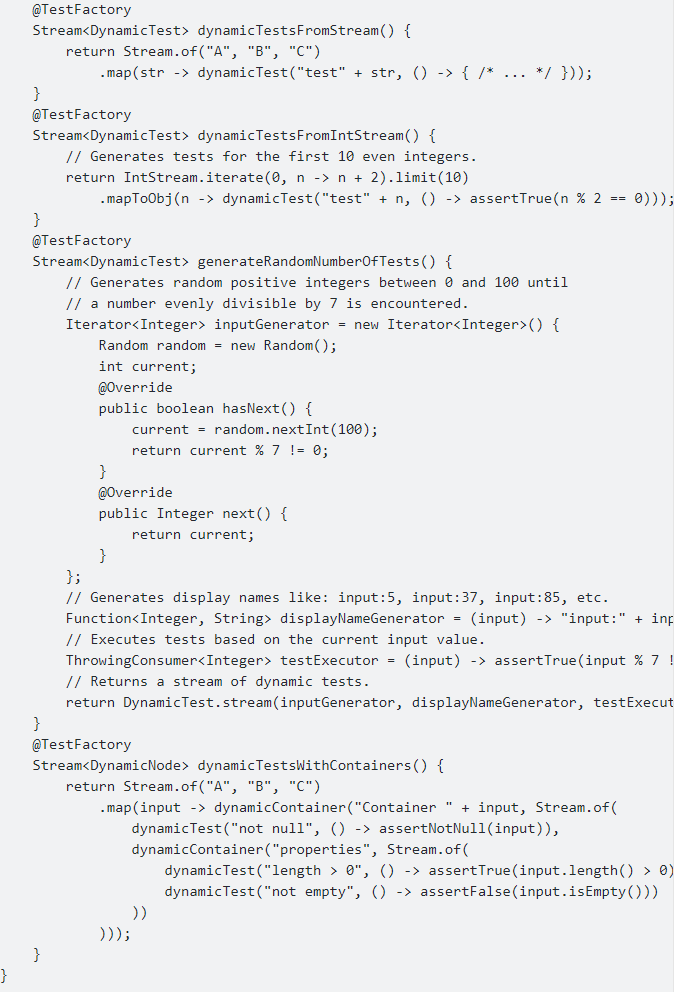


* @ParameterizedTest



* @RepeatedTest
* @TestFactory

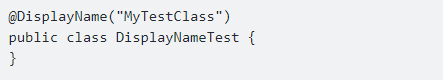




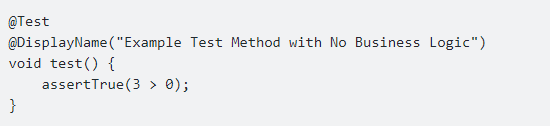
* @TestInstance: Annotation này dùng để cấu hình vòng đời cho một đối tượng test cụ thể. Ví dụ, nếu bạn muốn JUnit Jupiter thực hiện tất cả các phương thức test trên cùng một đối tượng test cụ thể, chỉ cần chú thích class test của bạn với @TestInstance (Lifecycle.PER\_CLASS).
* @TestTemplate



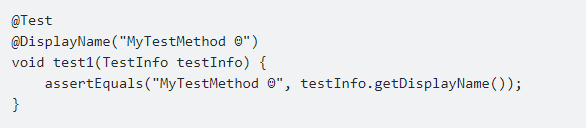
* @DisplayName
* Tùy biến hiển thị tên Class:



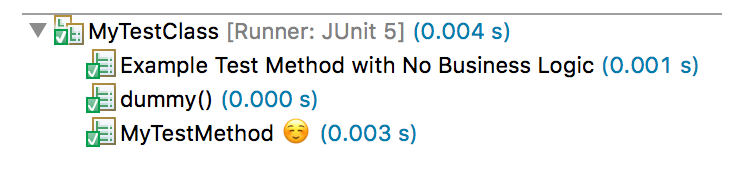
* Tùy biến hiển thị tên Method:



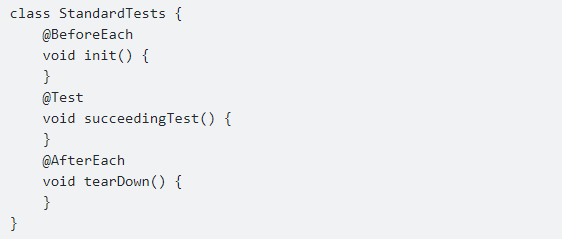
* Tùy biến hiển thị tên method và emojis:



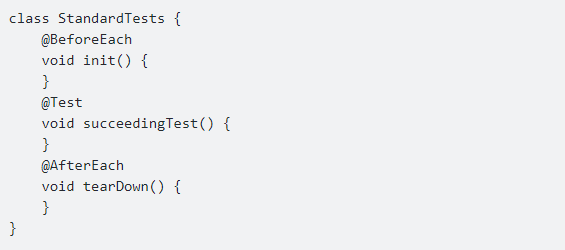
* Tên tùy biến được hiển thị trong Test Report:



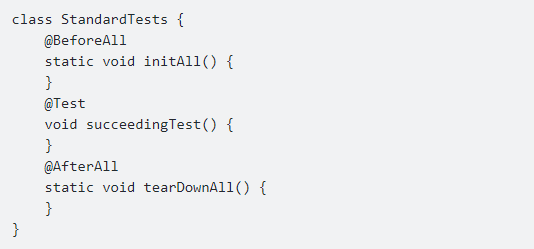
* @BeforeEach



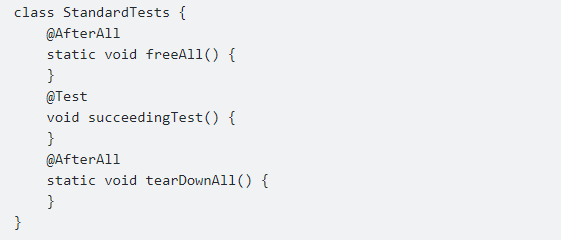
* @AfterEach



* @BeforeAll



* @AfterAll

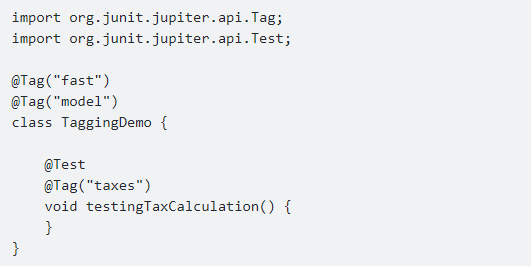


* @Nested

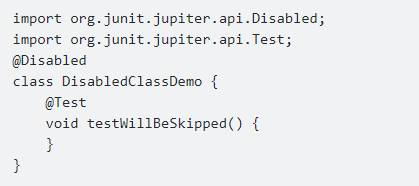




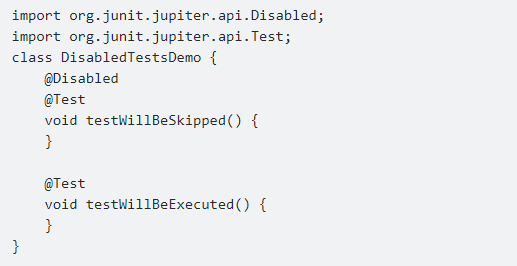
* @Tag



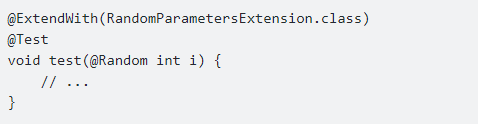
* @Disabled
  + Vô hiệu hóa Class Test:



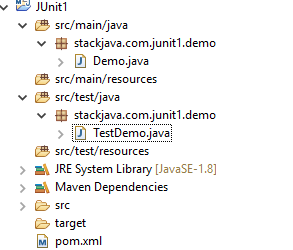
* + Vô hiệu hóa Method Test:



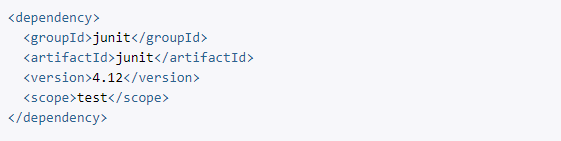
* @ExtendWith



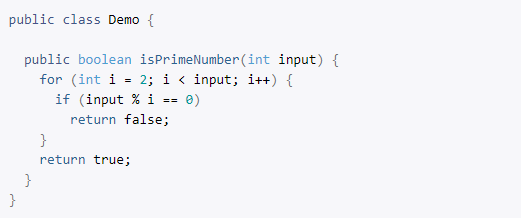
* **Thực hiện Junit 5 trong Eclipse, NetBeans, IntelliJ, and Maven (pom.xml )**
* Thực hiện Junit 5 trong Eclipses và Maven:
  + Tạo Maven Project:

[](https://stackjava.com/wp-content/uploads/2018/04/junit-0.png)

* + Khai báo:



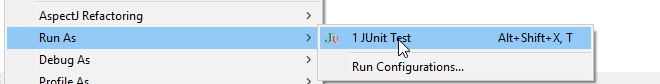
* + Code Ví dụ Demo kiểm tra số nguyên tố:



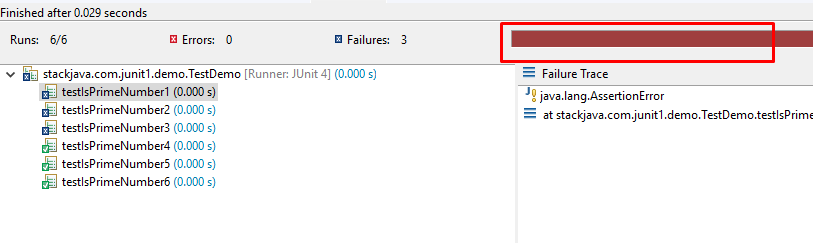
* + Code test: 6 Test Case tương ứng 6 Method:
* Hàm thứ nhất đầu vào là số âm ‘-1’ nên kết quả mong đợi sẽ là false nên mình dùng assertFalse
* Hàm thứ hai đầu vào là số 0, 0 không phải là số nguyên tố nên kết quả mong đợi sẽ là false
* Hàm thứ ba đầu vào là số 1, 1 không phải là số nguyên tố nên kết quả mong đợi sẽ là false
* Hàm thứ tư đầu vào là số ‘2’ , 2 là số nguyên tố nên kết quả mong đợi sẽ là true nên mình dùng assertTrue
* Hàm thứ năm đầu vào là số  ‘4’, 4 không phải là số nguyên tố nên kết quả mong đợi sẽ là false
* Hàm thứ sáu đầu vào là số ‘5’, 5 là số nguyên tố nên kết quả mong đợi sẽ là true



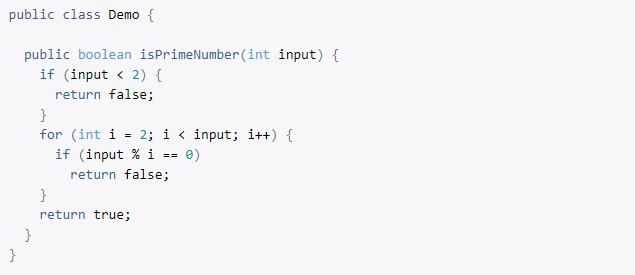
* + Chạy các Test Case:

[](https://stackjava.com/wp-content/uploads/2018/04/junit-3.png)

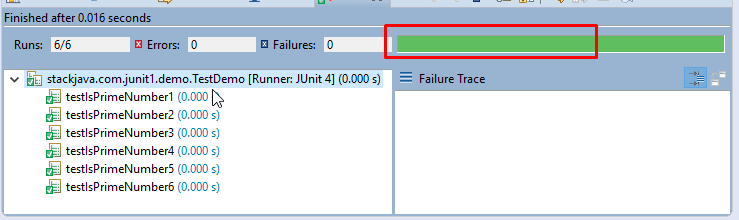
* + Kết quả:

[](https://stackjava.com/wp-content/uploads/2018/04/junit-1.png)

* + 3 Test Case thất bại chứng tỏ Method đang bị sai, ta sửa lại:



* + Chạy lại kết quả - Đúng:

[](https://stackjava.com/wp-content/uploads/2018/04/junit-2.png)

1. **Kiểm thử với Mockito và TestNG (Nghiên cứu thêm)**
   1. **Kiểm thử với Mockito:**

- JUnit là một framework dùng cho việc tạo các test case để kiểm thử các method của các đối tượng trong chương trình Java.

- Trong nhiều trường hợp, method được kiểm thử gọi đến những phương thức của một interface nào đó mà interface đấy chưa được implements bới bất kỳ class nào. Khi đó, trong Unit Test, ta cần giả lập các hàm của interface được sử dụng. Và Mockito là một framework thường dùng để giả lập kiểm thử, tạo giả một đối tượng cài đặt interface, giả lập lời gọi hàm đến đối tượng giả này và chỉ định kết quả trả lại cho lời gọi hàm giả lập đó.

* 1. **Kiểm thử với TestNG:**

- TestNG là một framework thử nghiệm lấy cảm hứng từ JUnit và NUnit nhưng đem đến một số tính năng mới làm cho nó mạnh hơn và dễ sử dụng hơn.

- Đây là một framework thử nghiệm mã nguồn mở tự động; trong đó NG của TestNG có nghĩa là Next Generation - Thế hệ tiếp theo. TestNG tương tự như JUnit nhưng mạnh hơn Junit, nhưng nó vẫn được lấy cảm hứng từ JUnit. Nó được thiết kế tốt hơn so với JUnit, đặc biệt là khi kiểm tra các lớp tích hợp.

- TestNG loại bỏ hầu hết các hạn chế của khuôn khổ cũ hơn và cho phép nhà phát triển khả năng viết các bài kiểm tra linh hoạt và mạnh mẽ hơn với sự trợ giúp của các chú thích đơn giản, nhóm, sắp xếp và parametrizing.