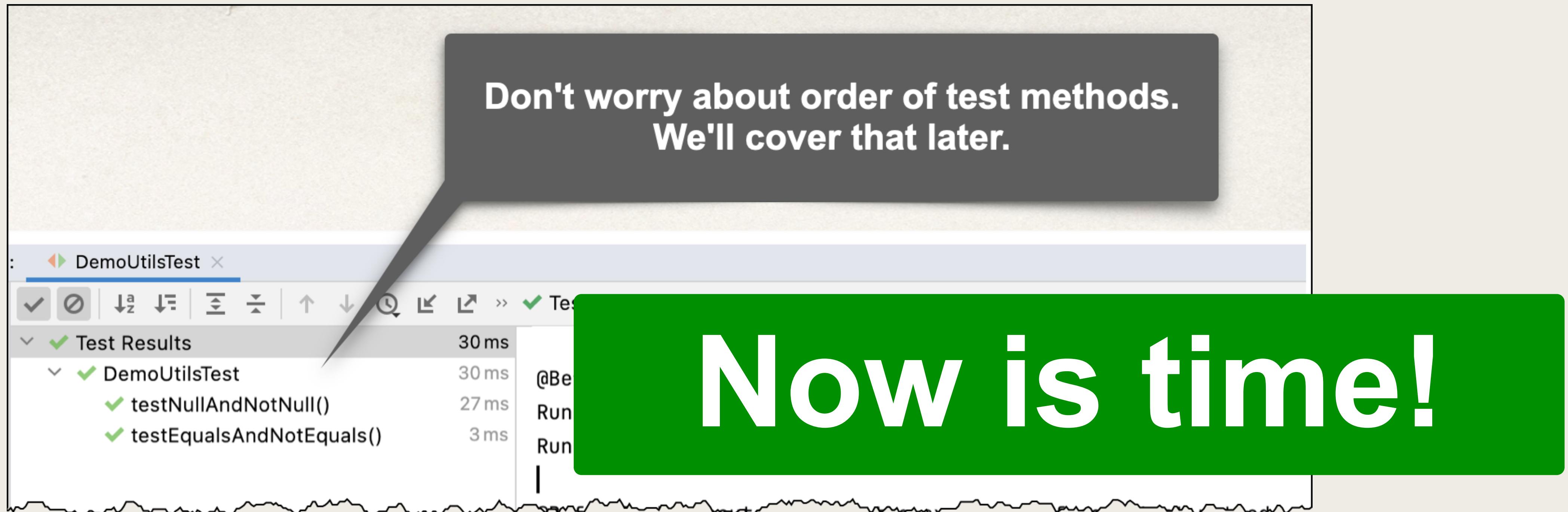


Running Tests in Order



Remember this?



Order???

- In general
 - Order should not be a factor in unit tests
 - There should be no dependency between tests
 - All tests should pass regardless of the order in which they are run

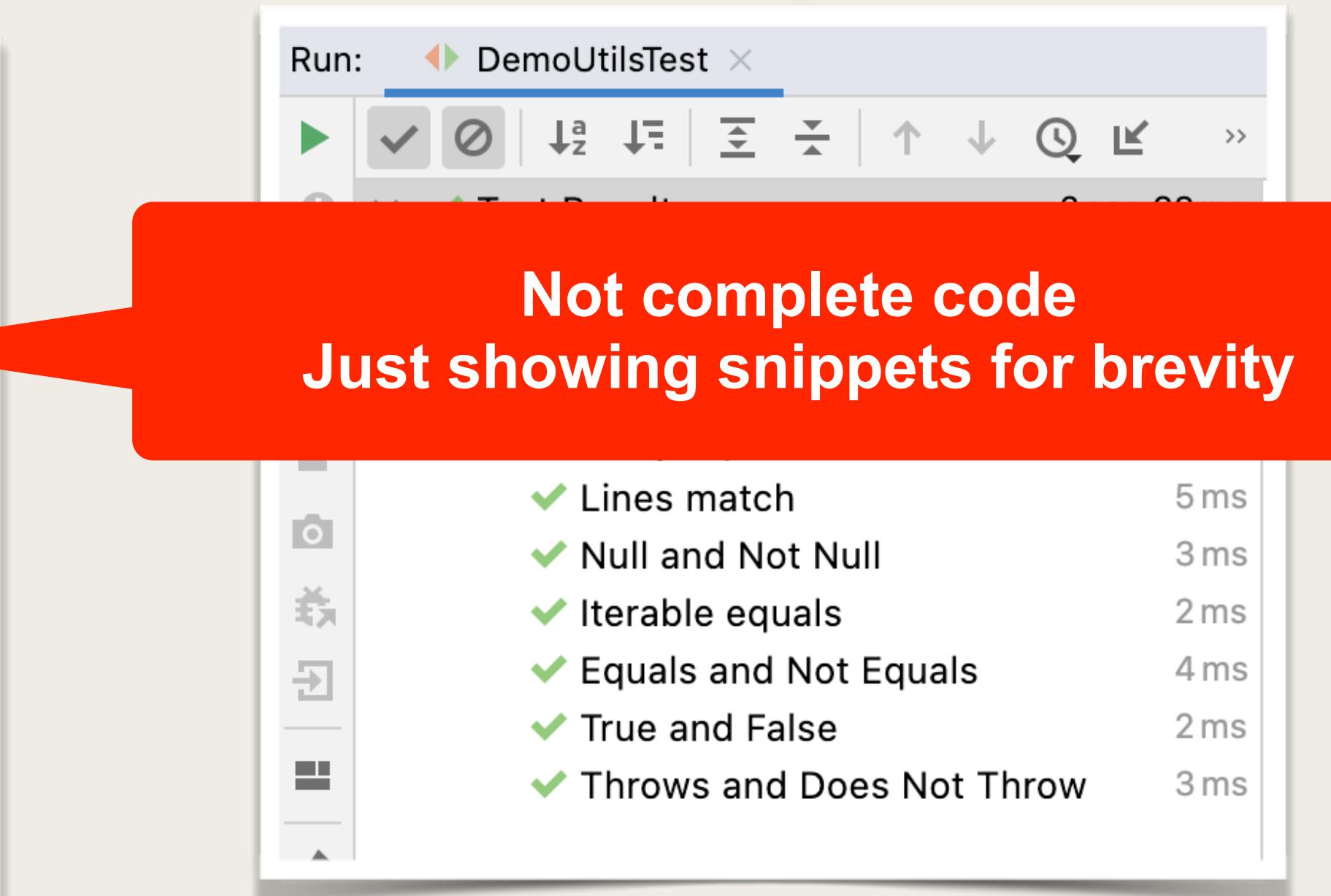
Use Cases

- However, there are some uses cases when you want to control the order
 - You want tests to appear in alphabetical order for reporting purposes
 - Sharing reports with project management, QA team etc...
 - Group tests based on functionality or requirements

Existing ... before any changes

DemoUtilsTest.java

```
class DemoUtilsTest {  
    ...  
  
    @DisplayName("Equals and Not Equals")  
    void testEqualsAndNotEquals()  
  
    @DisplayName("Null and Not Null")  
    void testNullAndNotNull()  
  
    @DisplayName("Same and Not Same")  
    void testSameAndNotSame()  
  
    @DisplayName("True and False")  
    void testTrueFalse()  
  
    @DisplayName("Array Equals")  
    void testArrayEquals()  
  
    @DisplayName("Iterable equals")  
    void testIterableEquals()  
  
    @DisplayName("Lines match")  
    void testLinesMatch()  
  
    @DisplayName("Throws and Does Not Throw")  
    void testThrowsAndDoesNotThrow()  
  
    @DisplayName("Timeout")  
    void testTimeout()  
}
```



Not complete code
Just showing snippets for brevity

JUnit Docs

By default, test classes and methods will be ordered using an algorithm that is deterministic

<https://junit.org/junit5/docs/current/user-guide/>

Annotation

Annotation	Description
@TestMethodOrder	Configures the order / sort algorithm for the test methods

Specify Method Order

Name	Description
MethodOrderer.DisplayName	Sorts test methods alphanumerically based on display names
MethodOrderer.MethodName	Sorts test methods alphanumerically based on method names
MethodOrderer.Random	Pseudo-random order based on method names
MethodOrderer.OrderAnnotation	Sorts test methods numerically based on @Order annotation

Order by Display Name

Sorted alphabetically

DemoUtilsTest.java

```
→ @TestMethodOrder(MethodOrderer.DisplayName.class)
class DemoUtilsTest {
    ...
    @DisplayName("Equals and Not Equals")
    void testEqualsAndNotEquals()

    @DisplayName("Null and Not Null")
    void testNullAndNotNull()

    @DisplayName("Same and Not Same")
    void testSameAndNotSame()

    @DisplayName("True and False")
    void testTrueFalse()

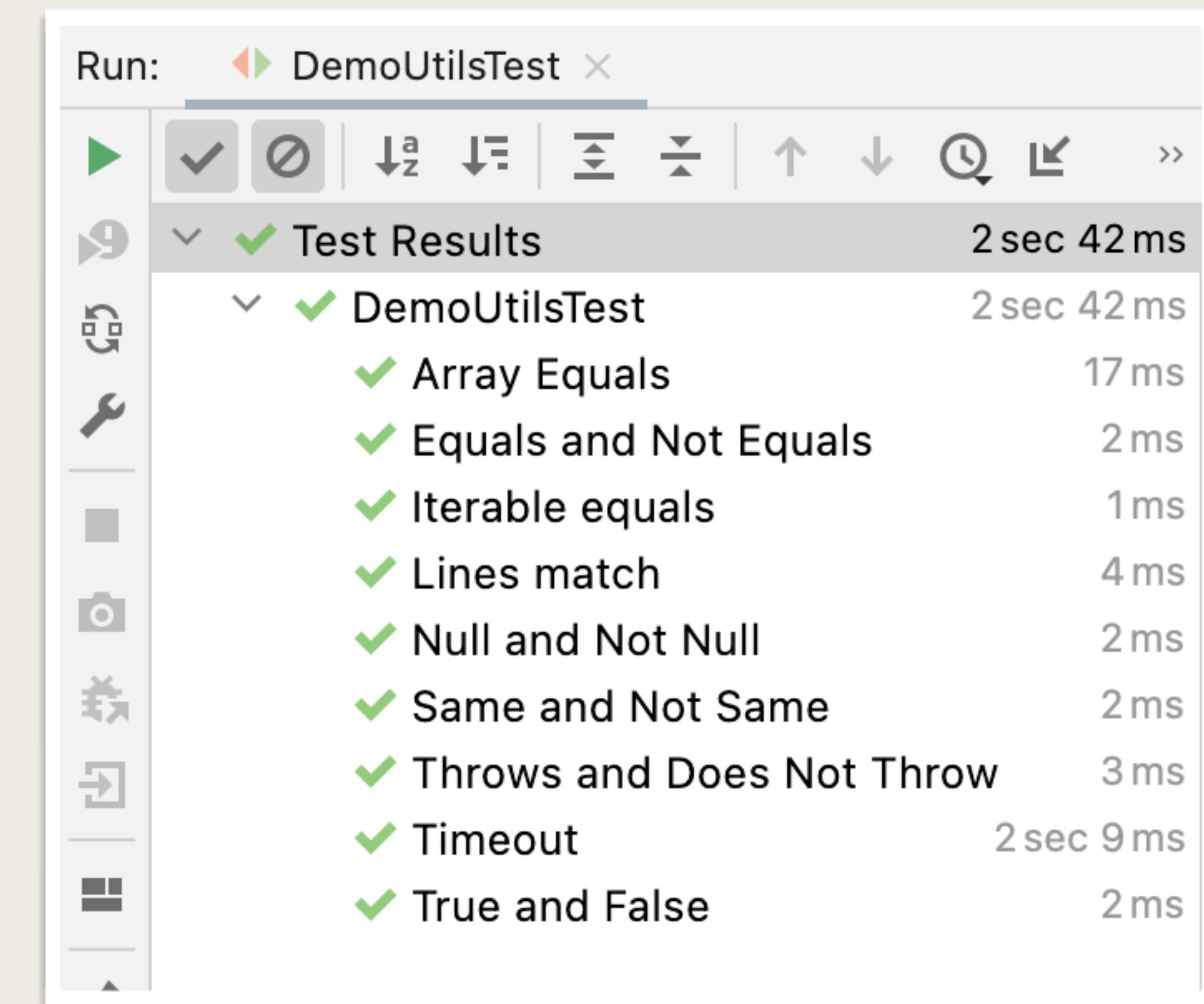
    @DisplayName("Array Equals")
    void testArrayEquals()

    @DisplayName("Iterable equals")
    void testIterableEquals()

    @DisplayName("Lines match")
    void testLinesMatch()

    @DisplayName("Throws and Does Not Throw")
    void testThrowsAndDoesNotThrow()

    @DisplayName("Timeout")
    void testTimeout()
}
```



Order by Method Name

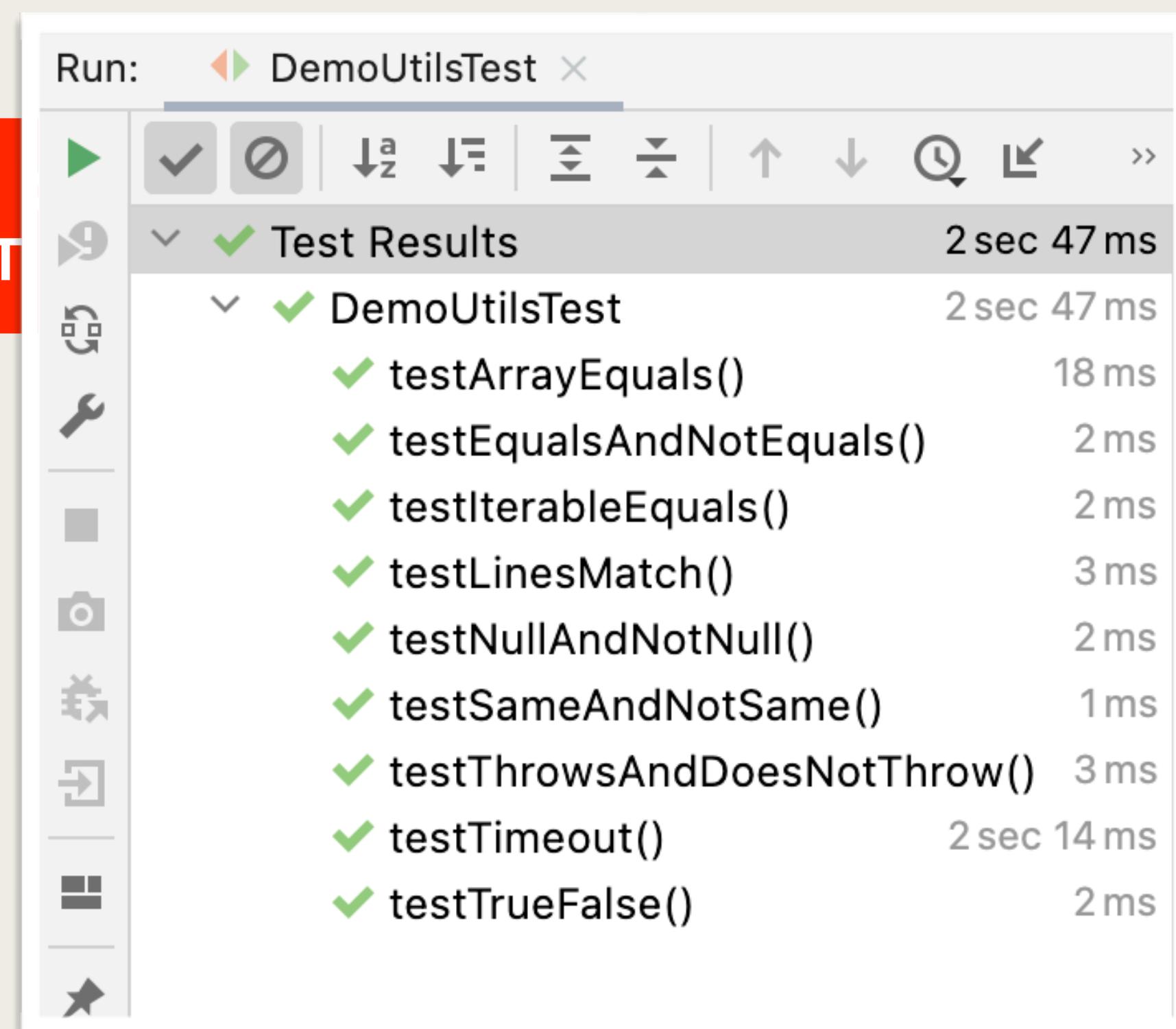
Sorted alphanumerically

DemoUtilsTest.java



```
@TestMethodOrder(MethodOrderer.MethodName.class)
class DemoUtilsTest {
    ...
    // @DisplayName("Equals and Not Equals")
    void testEqualsAndNotEquals()

    void testNullAndNotNull()
    void testSameAndNotSame()
    void testTrueFalse()
    void testArrayEquals()
    void testIterableEquals()
    void testLinesMatch()
    void testThrowsAndDoesNotThrow()
    void testTimeout()
}
```



Run: DemoUtilsTest

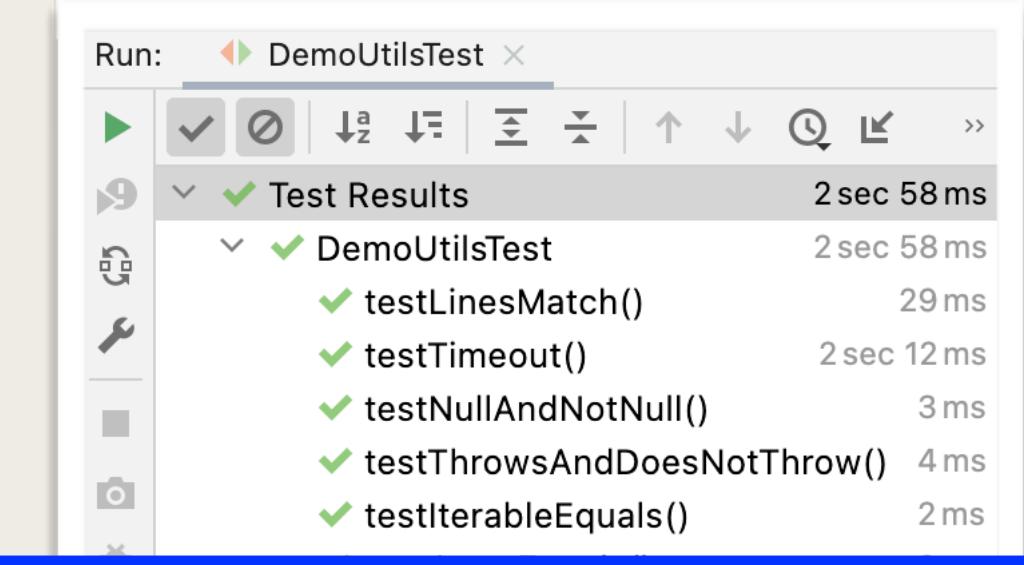
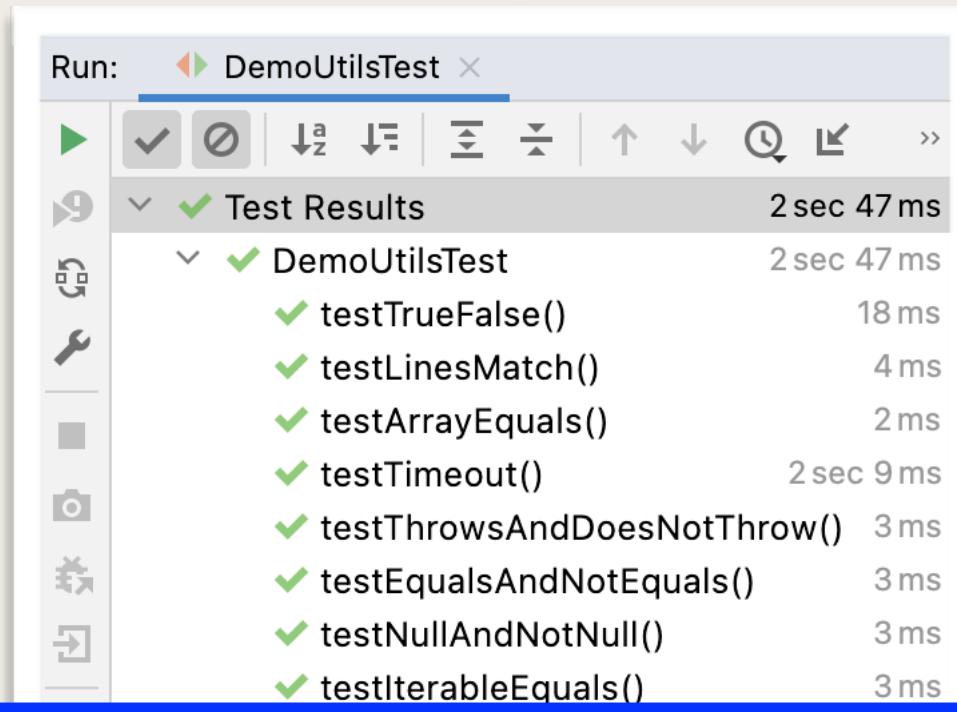
Method	Time
testArrayEquals()	18 ms
testEqualsAndNotEquals()	2 ms
testIterableEquals()	2 ms
testLinesMatch()	3 ms
testNullAndNotNull()	2 ms
testSameAndNotSame()	1 ms
testThrowsAndDoesNotThrow()	3 ms
testTimeout()	2 sec 14 ms
testTrueFalse()	2 ms

Random by Method Name

DemoUtilsTest.java

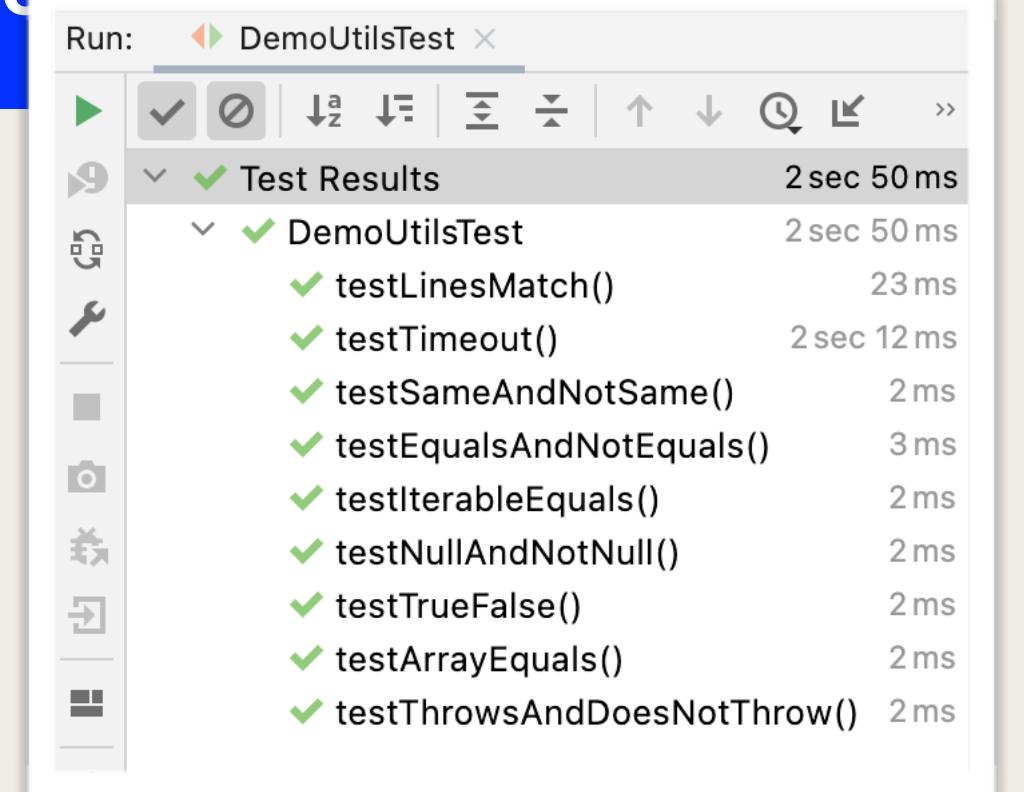
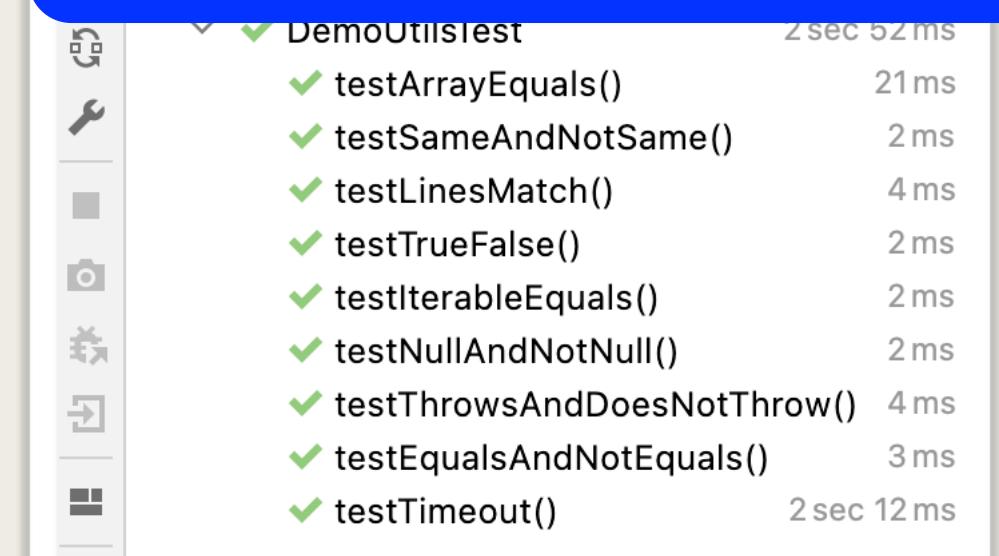
```
@TestMethodOrder(MethodOrderer.Random.class)
class DemoUtilsTest {
    ...
    void testEqualsAndHashCode()
    void testNullAndNotNull()
    void testSameAndNotSame()
    void testTrueFalse()
}
```

Can also use DisplayName
since everything is random
... order doesn't matter



Great scenario!!!!

Make sure all of your tests pass regardless of order
Confirms no dependencies between tests



Annotation

Annotation	Description
@Order	<p>Manually specify the order with an <code>int</code> number</p> <ul style="list-style-type: none">- Order with lowest number has highest priority- Negative numbers are allowed

@Order

Remember
Lowest number has highest priority

```
DemoUtilsTest.java
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class DemoUtilsTest {
    ...
    @DisplayName("Equals and Not Equals")
    @Order(3)
    void testEqualsAndNotEquals()

    @DisplayName("Null and Not Null")
    @Order(1)
    void testNullAndNotNull()

    @DisplayName("Same and Not Same")
    void testSameAndNotSame()

    @DisplayName("True and False")
    void testTrueFalse()

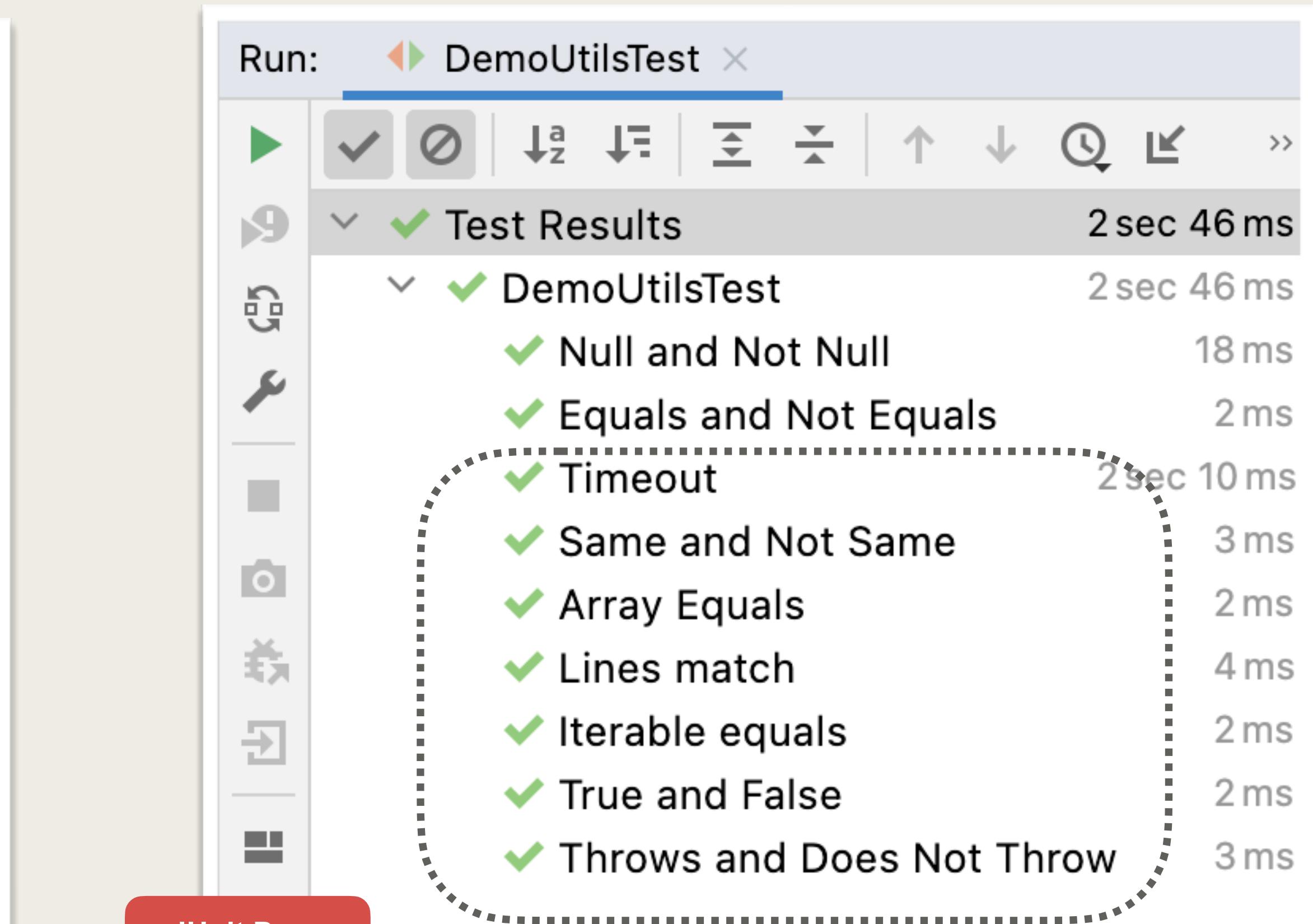
    @DisplayName("Array Equals")
    void testArrayEquals()

    @DisplayName("Iterable equals")
    void testIterableEquals()

    @DisplayName("Lines match")
    void testLinesMatch()

    @DisplayName("Throws and Does Not Throw")
    void testThrowsAndDoesNotThrow()

    @DisplayName("Timeout")
    void testTimeout()
}
```



JUnit Docs

By default, test classes and methods will be ordered using an algorithm that is deterministic but intentionally nonobvious.

@Order - Negative Numbers

Remember
Lowest number has highest priority

The screenshot shows the IntelliJ IDEA interface. On the left, the code editor displays `DemoUtilsTest.java` with various test methods annotated with `@Order`. A red arrow points from the code editor towards the run configuration. Three annotations are highlighted with red circles and numbers: `@Order(3)` for `testEqualsAndNotEquals()`, `@Order(1)` for `testNullAndNotNull()`, and `@Order(-7)` for `testArrayEquals()`.

```
class DemoUtilsTest {
    ...
    @DisplayName("Equals and Not Equals")
    @Order(3)
    void testEqualsAndNotEquals()

    @DisplayName("Null and Not Null")
    @Order(1)
    void testNullAndNotNull()

    @DisplayName("Same and Not Same")
    void testSameAndNotSame()

    @DisplayName("True and False")
    void testTrueFalse()

    @DisplayName("Array Equals")
    @Order(-7)
    void testArrayEquals() -7

    @DisplayName("Iterable equals")
    void testIterableEquals()

    @DisplayName("Lines match")
    void testLinesMatch()

    @DisplayName("Throws and Does Not Throw")
    void testThrowsAndDoesNotThrow()

    @DisplayName("Timeout")
    void testTimeout()
}
```

On the right, the "Run" interface shows the "Test Results" section for the `DemoUtilsTest` class. The results are listed as follows:

Test Method	Time
Array Equals	21ms
Null and Not Null	1ms
Equals and Not Equals	2ms
Timeout	2 sec 10 ms
Same and Not Same	2ms
Lines match	5ms
Iterable equals	2ms
True and False	2ms
Throws and Does Not Throw	3ms

@Order - Duplicate Order Values

If duplicate @Order values
then ...

JUnit Docs

By default, test classes and methods will be ordered using an algorithm that is deterministic but intentionally nonobvious.

Additional Features

- If you have multiple test classes, you can order the classes
- Define custom order implementation
- Configure default order in properties file

<https://junit.org/junit5/docs/current/user-guide/>