assertTimeoutPreemptively method

This lesson demonstrates how to use assertTimeoutPreemptively method in JUnit 5 to assert timeout conditions.

We'll cover the following

assertTimeoutPreemptively() method

Assertions API provide static `assertTimeoutPreemptively()` method. It is used to test long-running tasks. If given task inside the test case takes more than the specified duration, then the test will fail.

The executable provided to the test case will be executed in the `different` thread as that of the calling code. Also, the execution of the executable will be preemptively aborted if the timeout is exceeded.

Demo

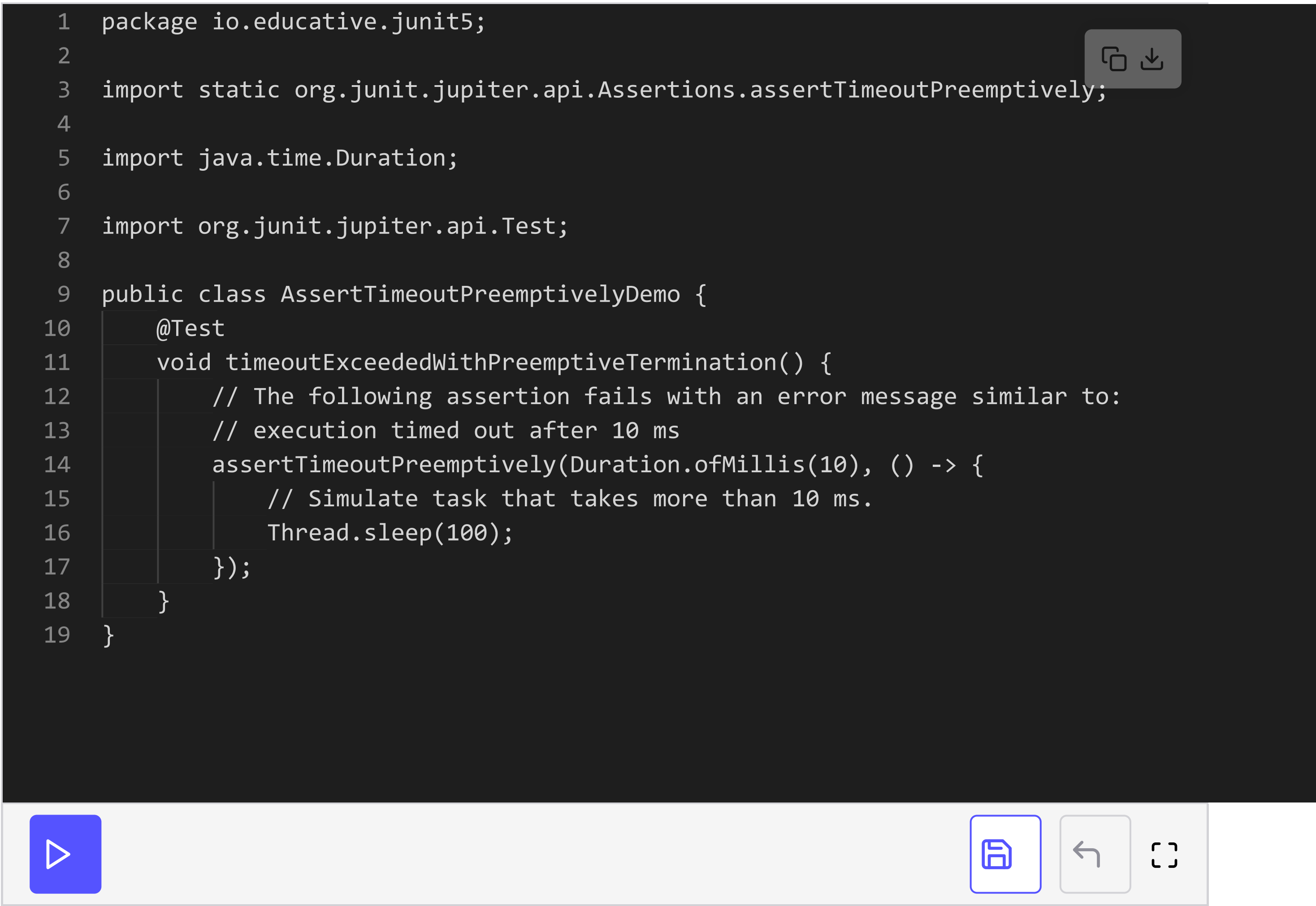
Let’s look into the usage of the above methods:-

Java Unit Testing with JUnit 5

JUnit 5 Assertions – assertTimeoutPreemptively() method

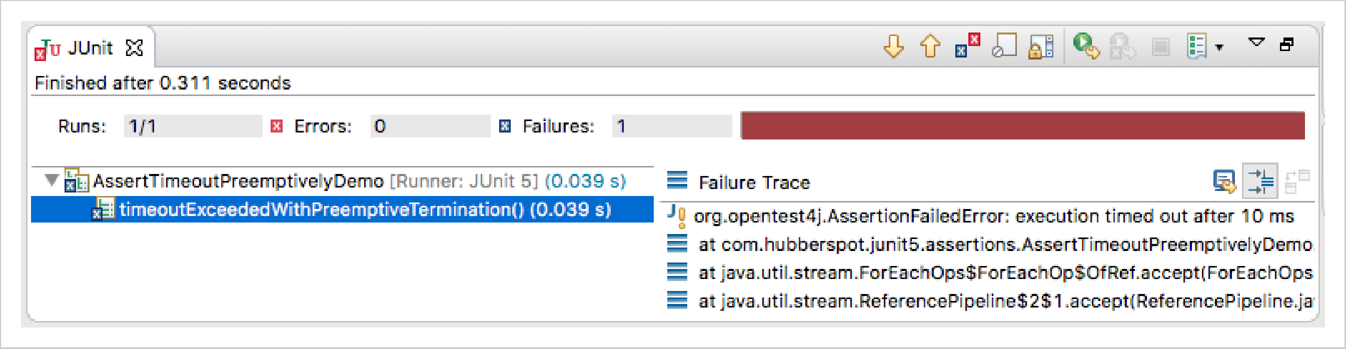
Dinesh Varyani
https://www.hubberspot.com

assertTimeoutPreemptively method



```java
package io.educative.junit5;

import static org.junit.jupiter.api.Assertions.assertTimeoutPreemptively;

import java.time.Duration;

import org.junit.jupiter.api.Test;

public class AssertTimeoutPreemptivelyDemo {
    @Test
    void timeoutExceededWithPreemptiveTermination() {
        // The following assertion fails with an error message similar to:
        // execution timed out after 10 ms
        assertTimeoutPreemptively(Duration.ofMillis(10), () -> {
            // Simulate task that takes more than 10 ms.
            Thread.sleep(100);
        });
    }
}
```

Run AssertTimeoutPreemptivelyDemo class as JUnit Test.



Finished after 0.311 seconds

Runs: 1/1 Errors: 0 Failures: 1

```
AssertTimeoutPreemptivelyDemo [Runner: JUnit 5] (0.039 s)
    timeoutExceededWithPreemptiveTermination() (0.039 s)
```

```
org.opentest4j.AssertionFailedError: execution timed out after 10 ms
at com.hubberspot.junit5.assertions.AssertTimeoutPreemptivelyDemo
at java.util.stream.ForEachOps$ForEachOp$OfRef.accept(ForEachOps
at java.util.stream.ReferencePipeline$2$1.accept(ReferencePipeline.ja
```

In the next chapter, we will discuss about JUnit Test Lifecycle callbacks.