```
//Pom.xml

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schema
  <modelVersion>4.0.0</modelVersion>
  <groupId>my.unit.test</groupId>
  <artifactId>unit-test-assignment</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
    <java.version>17</java.version>
    <project.build.sourceEncoding>utf-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>30.1.1-jre</version>
    </dependency>

    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>5.7.2</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.assertj</groupId>
      <artifactId>assertj-core</artifactId>
      <version>3.20.2</version>
      <scope>test</scope>
    </dependency>

   <dependency>
      <groupId>org.mockito</groupId>
      <artifactId>mockito-junit-jupiter</artifactId>
      <version>3.11.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>${java.version}</source>
          <target>${java.version}</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>



//TestDemo.java

package com.promineotech;

import java.util.*;

public class TestDemo {

    public static int addPositive(int a, int b) {
        if(a > 0 && b >0) {
            return a+b;
        }
        else {
            throw new IllegalArgumentException("Both parameters must be positive!");
        }

    }
    public static int expontential(int a, int b){
        if (b > 0) {
            int result = a;
            for (int i = 1; i < b; i++) {
                result *= a;
            }
            return result;
        }
        else {
            throw new IllegalArgumentException("Both parameters must be positive!");
        }
    }

    public int getRandomInt() {
        Random random = new Random();
        return random.nextInt(10) + 1;
    }


    public int randomNumberSquared() {
        int number = getRandomInt();
        return number*number;
    }

}
```

```
// TestDemoJUnitTest.java

package com.promineotech;

import static org.assertj.core.api.Assertions.assertThat;
import static org.assertj.core.api.Assertions.assertThatThrownBy;
import static org.mockito.Mockito.spy;
import static org.mockito.Mockito.doReturn;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.params.ParameterizedTest;
import java.util.stream.Stream;
import org.junit.jupiter.params.provider.Arguments;
import static org.junit.jupiter.params.provider.Arguments.*;
import org.junit.jupiter.params.provider.MethodSource;


class TestDemoJUnitTest {

    private TestDemo testDemo;

    static Stream<Arguments> argumentsForAddPositive(){
        //The method will throw an exception if the any of the value is less than 1
        // @formatter:off
        return Stream.of(
                arguments(4,5,9,false),
                arguments(40,50,90,false),
                arguments(5,4,9,false),
                arguments(0,5,35,true),
                arguments(-1,18,23,true)
                );
        // @formatter:on
    }

    static Stream<Arguments> argumentsForExponentFunction(){
        return Stream.of(
                arguments(3,2,9),
                arguments(5,3,125),
                arguments(-2,7,-128),
                arguments(10,4,10000),
                arguments(-25,3,-15625)
                );
    }
```

```java
    @BeforeEach
    void setUp() throws Exception {
        testDemo = new TestDemo();
    }

    @ParameterizedTest
    @MethodSource("com.promineotech.TestDemoJUnitTest#argumentsForAddPositive")
    void assertThatTwoPositiveNumbersAreAddedCorrectly(int a, int b, int expected, boolean expectedException) {
        if(!expectedException) {
            assertThat(testDemo.addPositive(a, b)).isEqualTo(expected);
        }
        else {
            assertThatThrownBy(() ->
        testDemo.addPositive(a, b))
            .isInstanceOf(IllegalArgumentException.class);
        }

    }

    @Test
    void assertThatPairsOfPositiveNumbersAreAddedCorrectly() {
        assertThat(testDemo.addPositive(4,5)).isEqualTo(9);
        assertThat(testDemo.addPositive(40,50)).isEqualTo(90);
        assertThat(testDemo.addPositive(5,9)).isEqualTo(14);
        assertThat(testDemo.addPositive(12,23)).isEqualTo(35);
        assertThat(testDemo.addPositive(5,18)).isEqualTo(23);

    }


    @Test
    void assertThatPairOfPositiveNumbersAreExponentiallyRaisedCorrectly() {
        assertThat(testDemo.expontential(3,2)).isEqualTo(9);
        assertThat(testDemo.expontential(5,3)).isEqualTo(125);
        assertThat(testDemo.expontential(2,7)).isEqualTo(128);
        assertThat(testDemo.expontential(10,4)).isEqualTo(10000);
        assertThat(testDemo.expontential(25,3)).isEqualTo(15625);
    }

    @ParameterizedTest
    @MethodSource("com.promineotech.TestDemoJUnitTest#argumentsForExponentFunction")
    void assertThatPairOfPositiveNumbersAreExponentiallyRaisedCorrectlyParameterized(int a, int b, int expected) {
            assertThat(testDemo.expontential(a, b)).isEqualTo(expected);
    }


    //Mocking problem
    @Test
    void assertThatNumberSquaredIsCorrect() {
        TestDemo mockDemo = spy(testDemo);
        doReturn(5).when(mockDemo).getRandomInt();
        int fiveSquared = mockDemo.randomNumberSquared();
        assertThat(fiveSquared).isEqualTo(25);

    }

}



//REREFERENCES
//YOUTUBE
//https://youtu.be/gvRpJe0UJGc

//GIT HUB
https://github.com/fmd5045/Week12UnitTesting.git
```