

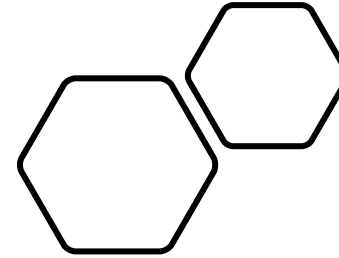


BridgeLabz

Employability Delivered

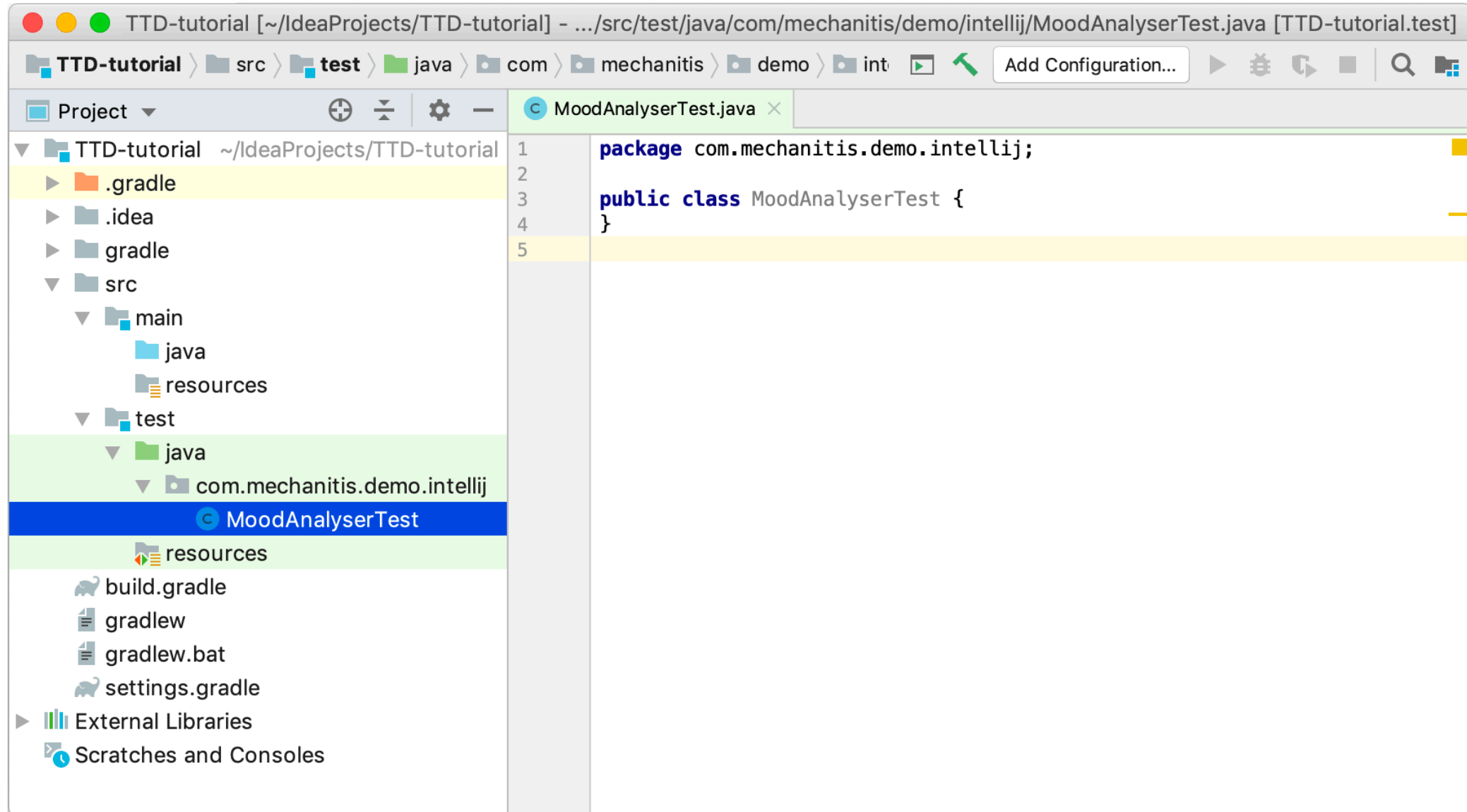
Junit with
IntelliJ Intro

IntelliJ Intro to Junit

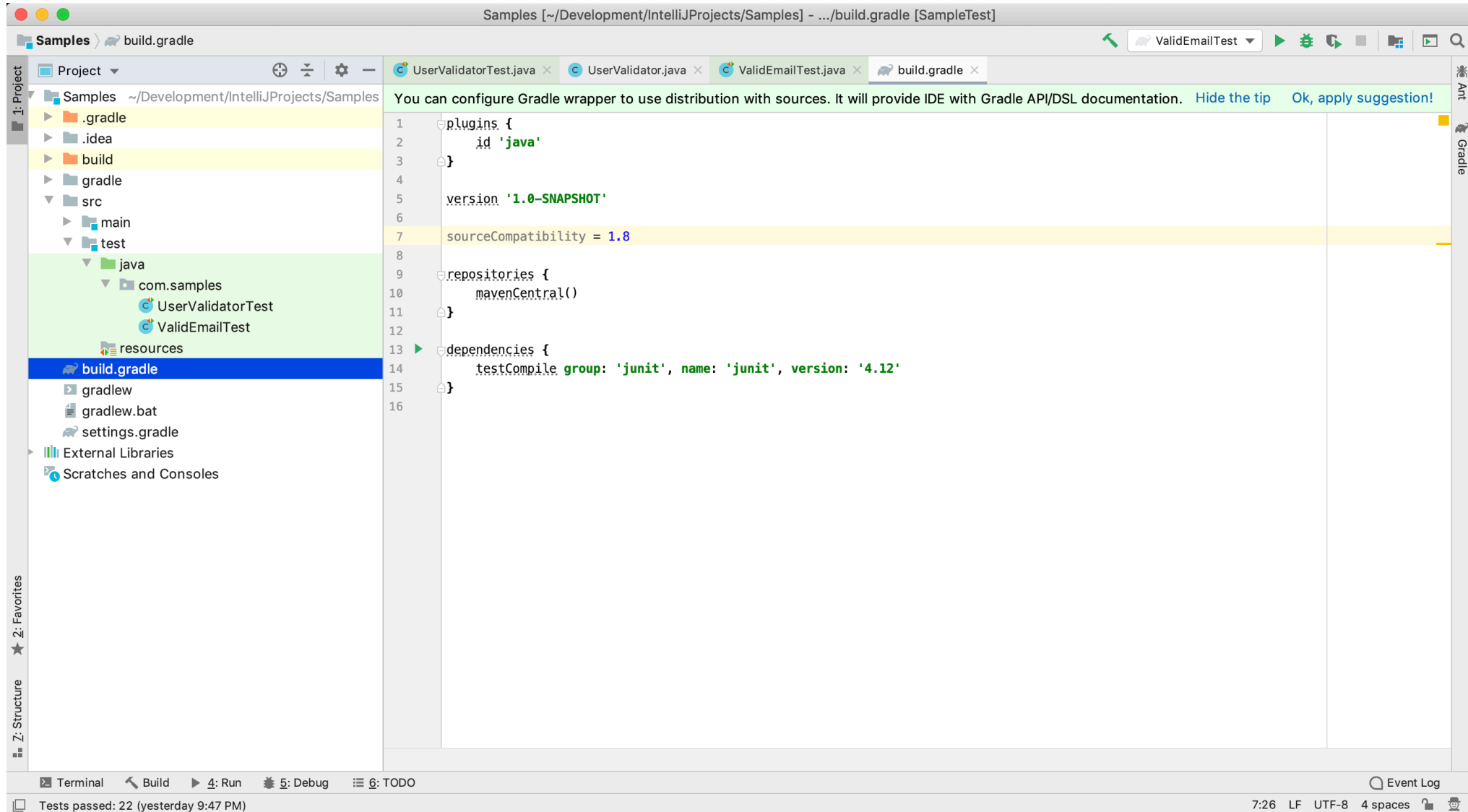


Create your first Junit
Mood Analyser
Application

Intro to IntelliJ – Create your Junit Test File



Intro to IntelliJ – Set Gradle Dependency



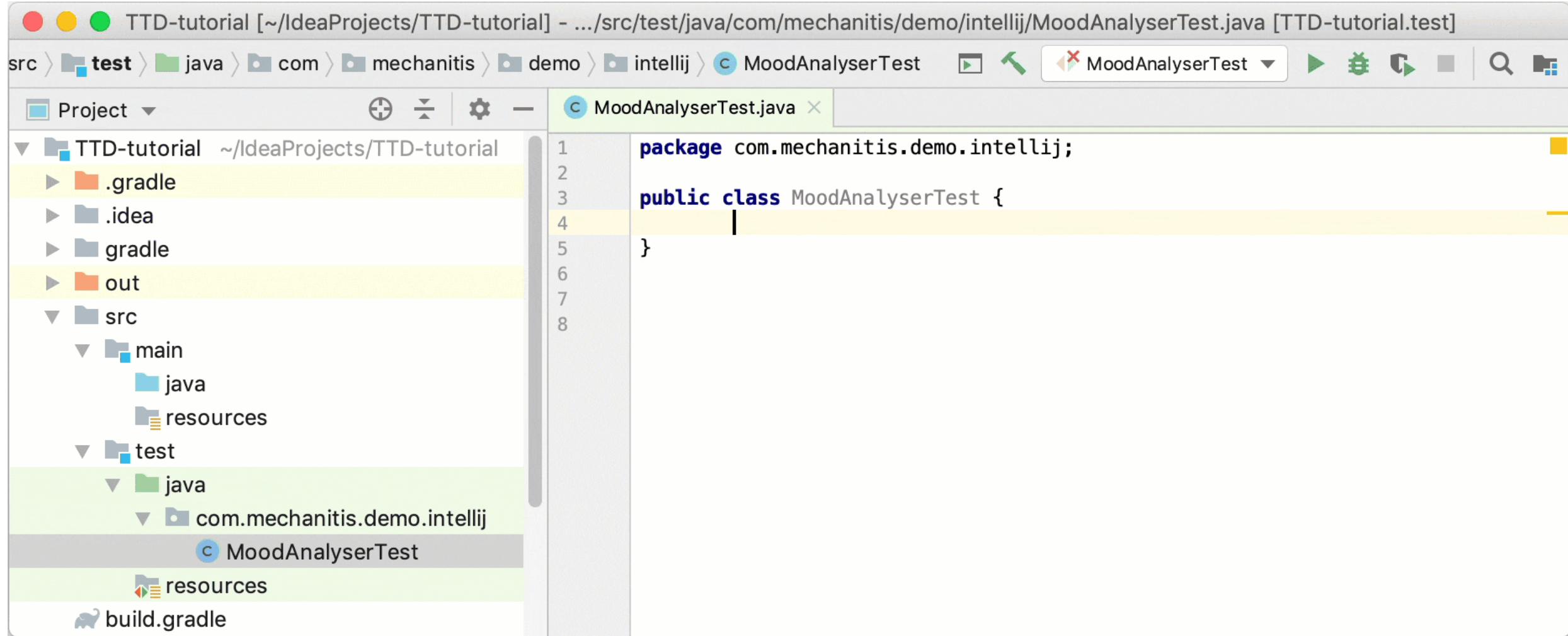
The screenshot shows the IntelliJ IDEA IDE interface. The left sidebar displays the Project Structure view for a project named 'Samples'. The 'build.gradle' file is selected under the 'resources' folder. The main editor window shows the content of 'build.gradle' with the following code:

```
1 plugins {  
2     id 'java'  
3 }  
4  
5 version '1.0-SNAPSHOT'  
6  
7 sourceCompatibility = 1.8  
8  
9 repositories {  
10     mavenCentral()  
11 }  
12  
13 dependencies {  
14     testCompile group: 'junit', name: 'junit', version: '4.12'  
15 }  
16
```

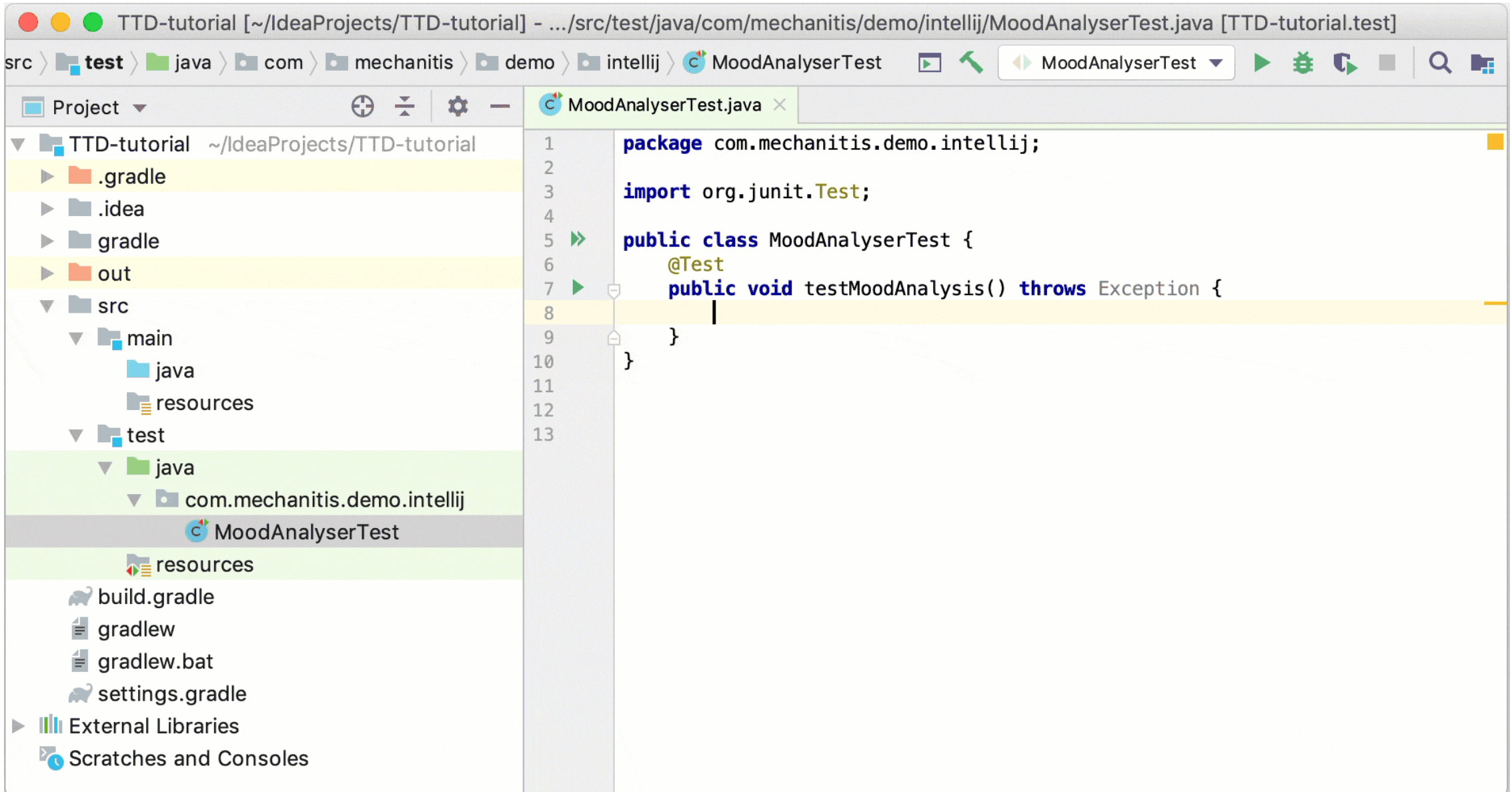
A tip banner at the top of the editor states: "You can configure Gradle wrapper to use distribution with sources. It will provide IDE with Gradle API/DSL documentation. [Hide the tip](#) [Ok, apply suggestion!](#)".

The bottom status bar shows "Tests passed: 22 (yesterday 9:47 PM)" and "7:26 LF UTF-8 4 spaces".

Intro to IntelliJ – Create Test Method



Intro to IntelliJ – Write Test Method Body



The screenshot shows the IntelliJ IDEA IDE interface. The top toolbar includes icons for running, debugging, and other IDE functions. The breadcrumb navigation shows the path: src > test > java > com > mechanitis > demo > intellij > MoodAnalyserTest. The left sidebar displays the Project view, showing the directory structure of the TTD-tutorial project, with the current file MoodAnalyserTest.java selected. The main editor window shows the code for MoodAnalyserTest.java, which includes package, import, and class declarations, with the testMoodAnalysis() method body being actively edited.

```
1 package com.mechanitis.demo.intellij;
2
3 import org.junit.Test;
4
5 public class MoodAnalyserTest {
6     @Test
7     public void testMoodAnalysis() throws Exception {
8         |
9     }
10 }
11
12
13
```


Intro to IntelliJ – Extract Variable

TTD-tutorial [~/IdeaProjects/TTD-tutorial] - .../src/test/java/com/mechanitis/demo/intellij/MoodAnalyserTest.java [TTD-tutorial.test]

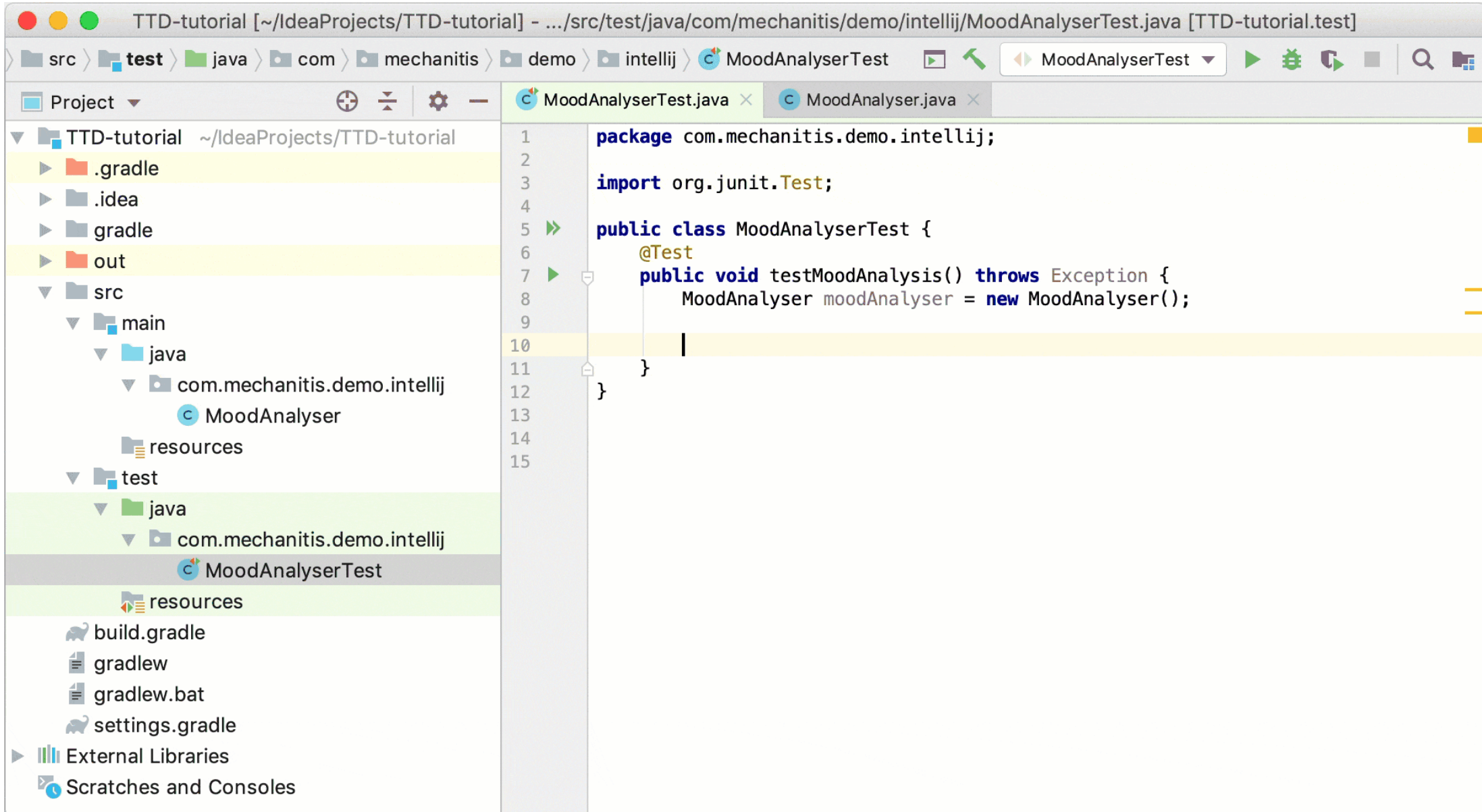
src > test > java > com > mechanitis > demo > intellij > MoodAnalyserTest

Project ▾

- TTD-tutorial ~/IdeaProjects/TTD-tutorial
 - .gradle
 - .idea
 - gradle
 - out
 - src
 - main
 - java
 - com.mechanitis.demo.intellij
 - MoodAnalyser
 - resources
 - test
 - java
 - com.mechanitis.demo.intellij
 - MoodAnalyserTest
 - resources

```
1 package com.mechanitis.demo.intellij;
2
3 import org.junit.Test;
4
5 public class MoodAnalyserTest {
6     @Test
7     public void testMoodAnalysis() throws Exception {
8         new MoodAnalyser
9     }
10 }
11
12
13
```

Intro to IntelliJ – Quick Action to Create Method

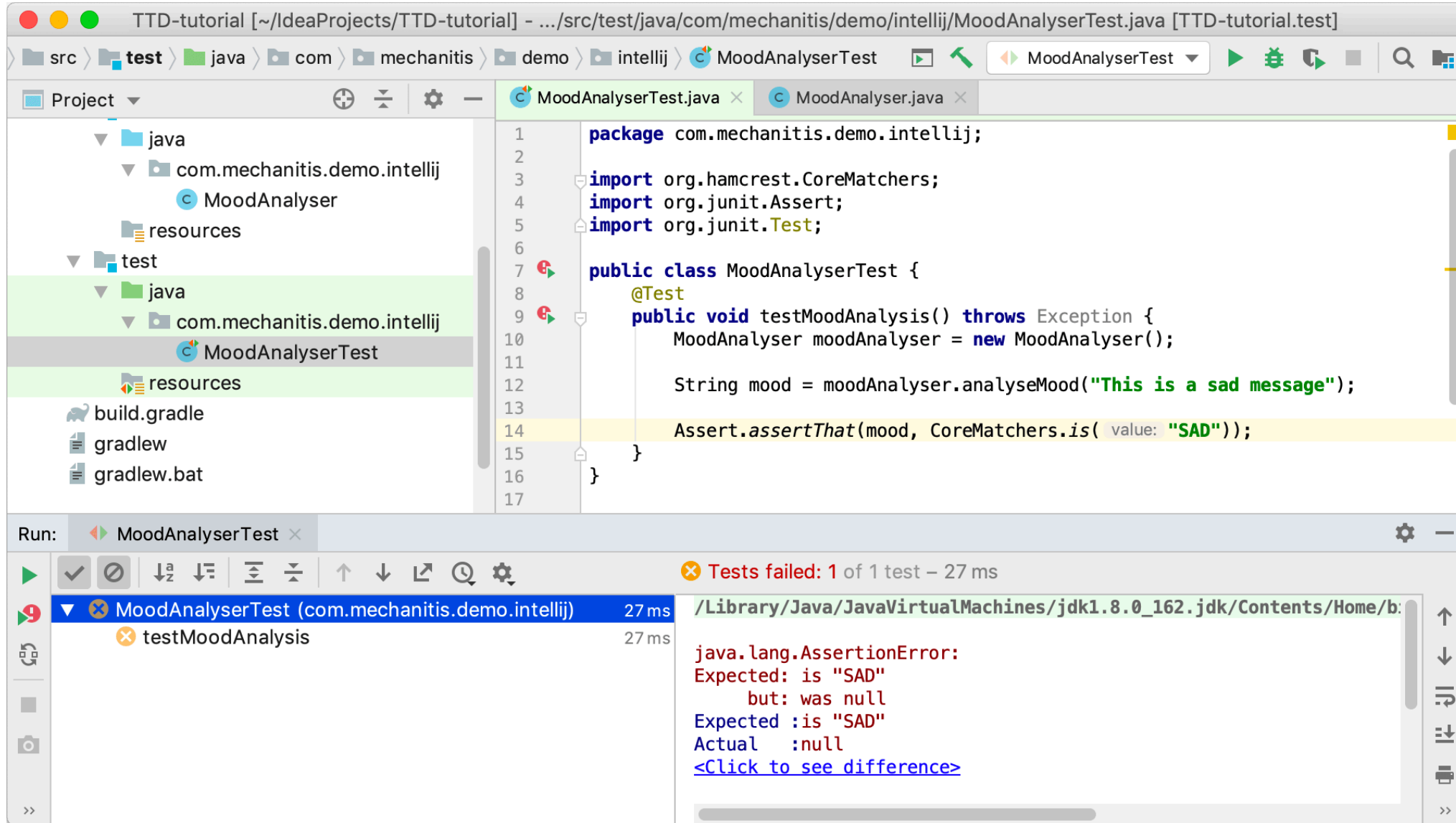


The screenshot shows the IntelliJ IDEA IDE interface. The left sidebar displays the project structure for 'TTD-tutorial'. The main editor window shows the file 'MoodAnalyserTest.java' with the following code:

```
1 package com.mechanitis.demo.intellij;  
2  
3 import org.junit.Test;  
4  
5 public class MoodAnalyserTest {  
6     @Test  
7     public void testMoodAnalysis() throws Exception {  
8         MoodAnalyser moodAnalyser = new MoodAnalyser();  
9  
10    }  
11 }  
12  
13  
14  
15
```

The code is partially highlighted in yellow, and a vertical cursor is visible on line 10. The IDE's title bar and toolbar are also visible.

Intro to IntelliJ – Run the Test Cases



The screenshot shows the IntelliJ IDEA IDE interface. The top toolbar includes icons for running tests (a green play button with a bug icon) and debugging. The left sidebar displays the project structure, with the 'test' directory expanded to show the 'MoodAnalyserTest' class. The main editor window shows the source code of 'MoodAnalyserTest.java'. The code includes imports for Hamcrest and JUnit, and a test method 'testMoodAnalysis' that uses 'Assert.assertThat' to verify the output of 'MoodAnalyser.analyseMood'. The test fails because the actual result is null, while the expected result is 'SAD'. The bottom panel shows the 'Run' configuration for 'MoodAnalyserTest' and a detailed error message in the 'Test Results' tab.

```
package com.mechanitis.demo.intellij;

import org.hamcrest.CoreMatchers;
import org.junit.Assert;
import org.junit.Test;

public class MoodAnalyserTest {
    @Test
    public void testMoodAnalysis() throws Exception {
        MoodAnalyser moodAnalyser = new MoodAnalyser();

        String mood = moodAnalyser.analyseMood("This is a sad message");

        Assert.assertThat(mood, CoreMatchers.is("SAD"));
    }
}
```

Run: MoodAnalyserTest

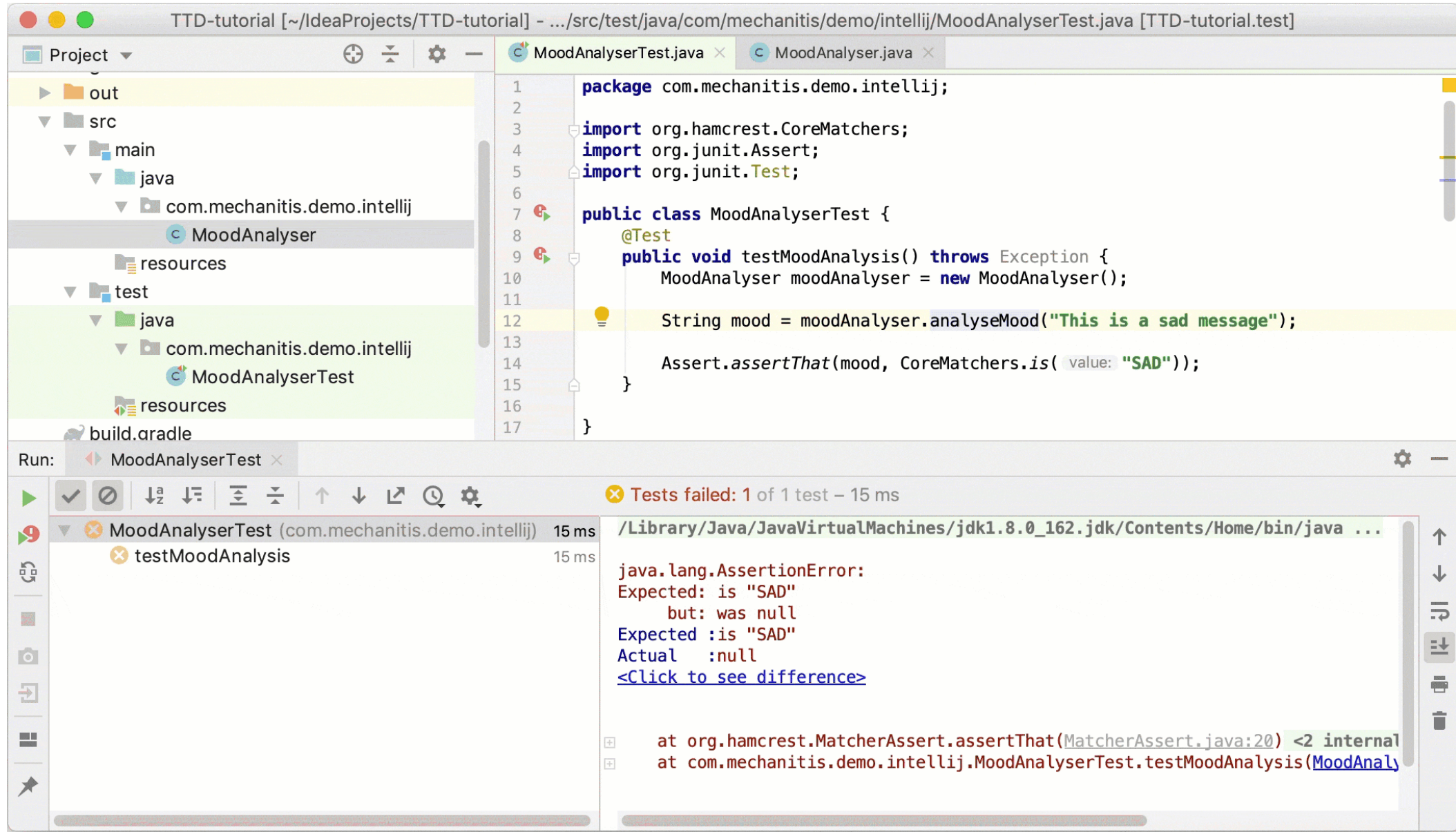
Tests failed: 1 of 1 test – 27 ms

MoodAnalyserTest (com.mechanitis.demo.intellij) 27 ms

testMoodAnalysis 27 ms

java.lang.AssertionError:
Expected: is "SAD"
but: was null
Expected :is "SAD"
Actual :null
[<Click to see difference>](#)

Intro to IntelliJ – Implement Code to Make it Work



The screenshot displays the IntelliJ IDEA IDE interface. The top toolbar shows standard window and editor controls. The left sidebar contains the Project tool window, showing a directory structure with 'src/main/java/com.mechanitis.demo.intellij' and 'test/java/com.mechanitis.demo.intellij'. The 'MoodAnalyserTest' class is selected in the test directory. The main editor window shows the source code of 'MoodAnalyserTest.java'. The code includes package declarations, imports for Hamcrest and JUnit, and a test method 'testMoodAnalysis' that creates a 'MoodAnalyser' instance and calls 'analyseMood' with the string 'This is a sad message'. The test fails at line 12, where 'mood' is null, but the assertion expects 'SAD'. The bottom panel shows the Run tool window with a failed test result for 'testMoodAnalysis' and a detailed error message: 'java.lang.AssertionError: Expected: is "SAD" but: was null'. The error stack trace points to the 'assertThat' method in Hamcrest and the 'testMoodAnalysis' method in 'MoodAnalyserTest'.

```
package com.mechanitis.demo.intellij;

import org.hamcrest.CoreMatchers;
import org.junit.Assert;
import org.junit.Test;

public class MoodAnalyserTest {
    @Test
    public void testMoodAnalysis() throws Exception {
        MoodAnalyser moodAnalyser = new MoodAnalyser();

        String mood = moodAnalyser.analyseMood("This is a sad message");

        Assert.assertThat(mood, CoreMatchers.is(value: "SAD"));
    }
}
```

Run: MoodAnalyserTest

Tests failed: 1 of 1 test – 15 ms

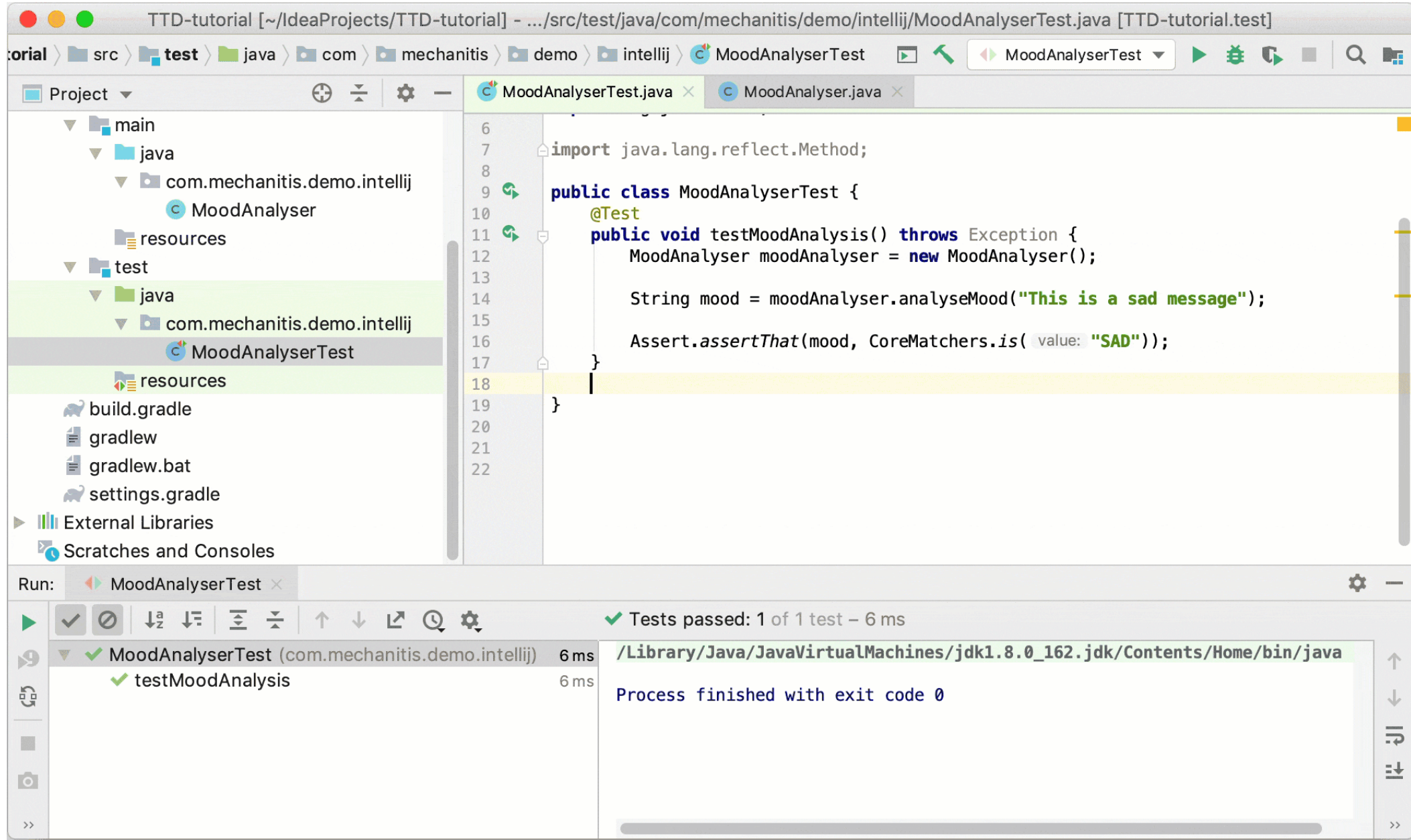
MoodAnalyserTest (com.mechanitis.demo.intellij) 15 ms

testMoodAnalysis 15 ms

java.lang.AssertionError:
Expected: is "SAD"
but: was null
Expected :is "SAD"
Actual :null
<Click to see difference>

at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20) <2 internal>
at com.mechanitis.demo.intellij.MoodAnalyserTest.testMoodAnalysis(MoodAnaly

Intro to IntelliJ – Write Second Test to Fail First



The screenshot displays the IntelliJ IDEA IDE interface. The top toolbar shows the Run button (a green play icon) and a dropdown menu with 'MoodAnalyserTest' selected. The left sidebar shows the project structure with the following hierarchy:

- main
 - java
 - com.mechanitis.demo.intellij
 - MoodAnalyser
- test
 - java
 - com.mechanitis.demo.intellij
 - MoodAnalyserTest
- resources
- build.gradle
- gradlew
- gradlew.bat
- settings.gradle
- External Libraries
- Scratches and Consoles

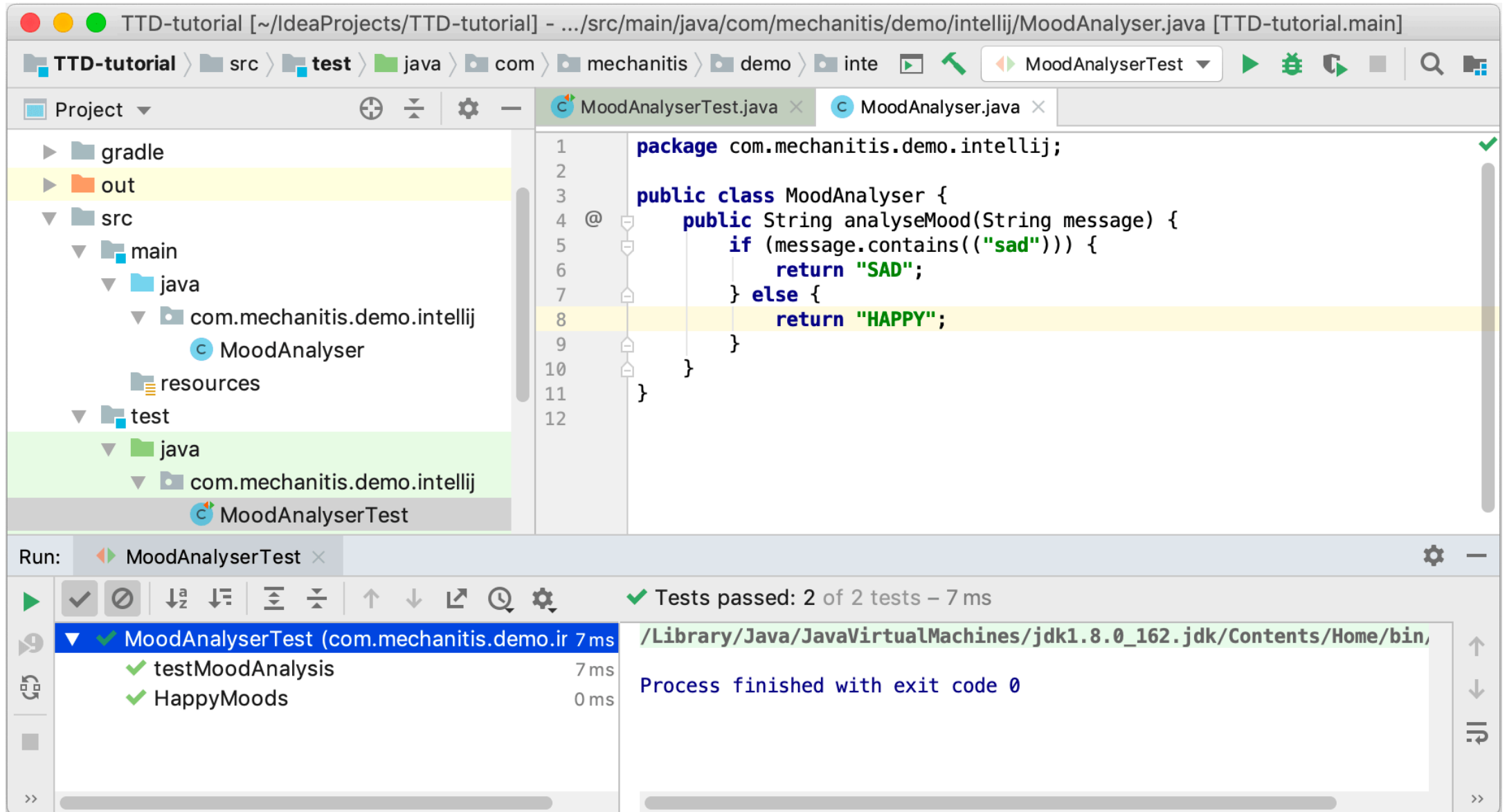
The main editor window shows the code for `MoodAnalyserTest.java`:

```
6
7 import java.lang.reflect.Method;
8
9 public class MoodAnalyserTest {
10     @Test
11     public void testMoodAnalysis() throws Exception {
12         MoodAnalyser moodAnalyser = new MoodAnalyser();
13
14         String mood = moodAnalyser.analyseMood("This is a sad message");
15
16         Assert.assertThat(mood, CoreMatchers.is( value: "SAD"));
17     }
18
19 }
20
21
22
```

The bottom panel shows the Run configuration and results. The Run tab is active, showing the configuration 'MoodAnalyserTest'. The Run button is highlighted. The Run Results tab shows the following output:

```
✓ Tests passed: 1 of 1 test – 6 ms
✓ MoodAnalyserTest (com.mechanitis.demo.intellij) 6 ms
  ✓ testMoodAnalysis 6 ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java
Process finished with exit code 0
```


Intro to IntelliJ – Implement the Actual Code



The screenshot displays the IntelliJ IDEA IDE interface. The top toolbar shows the 'Run' button (a green play icon) and the 'Test' button (a green bug icon). The breadcrumb navigation at the top indicates the current file is `MoodAnalyser.java` within the `com.mechanitis.demo.intelli` package.

The left sidebar shows the Project view with the following structure:

- gradle
- out
- src
 - main
 - java
 - com.mechanitis.demo.intelli
 - MoodAnalyser
 - test
 - java
 - com.mechanitis.demo.intelli
 - MoodAnalyserTest

The main editor window shows the code for `MoodAnalyser.java`:

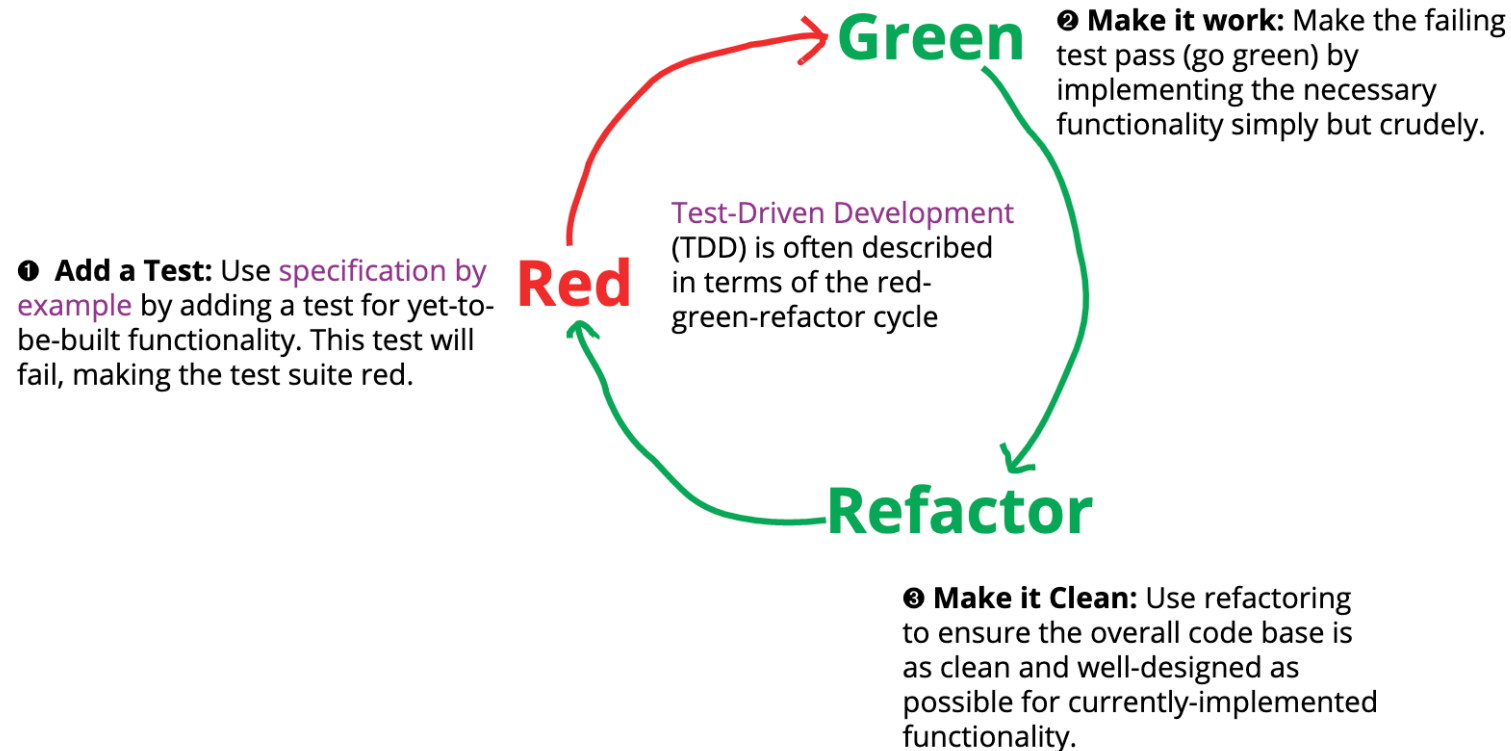
```
1 package com.mechanitis.demo.intelli;  
2  
3 public class MoodAnalyser {  
4     @ public String analyseMood(String message) {  
5         if (message.contains("sad")) {  
6             return "SAD";  
7         } else {  
8             return "HAPPY";  
9         }  
10    }  
11 }  
12
```

The bottom panel shows the Run configuration for `MoodAnalyserTest`. The Run tab displays the following output:

```
✓ Tests passed: 2 of 2 tests – 7 ms  
✓ MoodAnalyserTest (com.mechanitis.demo.ir 7 ms  
  ✓ testMoodAnalysis 7 ms  
  ✓ HappyMoods 0 ms  
/Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin,  
Process finished with exit code 0
```

Intro to IntelliJ – Red Green Refactor

Refactoring is often taught in the context of TDD





BridgeLabz

Employability Delivered

Thankyou