

گزارش آزمایشگاه مهندسی نرم افزار

پاییز ۱۴۰۱



گزارش ۳: تبدیل نیازمندی‌ها به موارد آزمون با استفاده از روش ایجاد مبتنی بر رفتار (BDD)

دانشکده مهندسی کامپیوتر

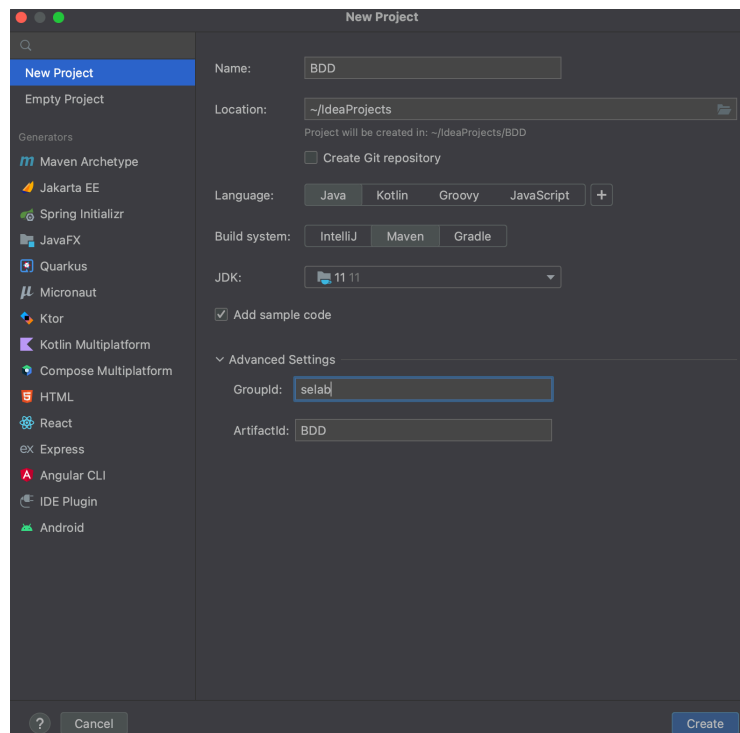
رستا روغنی (۹۷۱۰۵۹۶۳)

مهرانه نجفی (۹۷۱۰۴۷۰۷)

۱ سناریو: جمع دو عدد

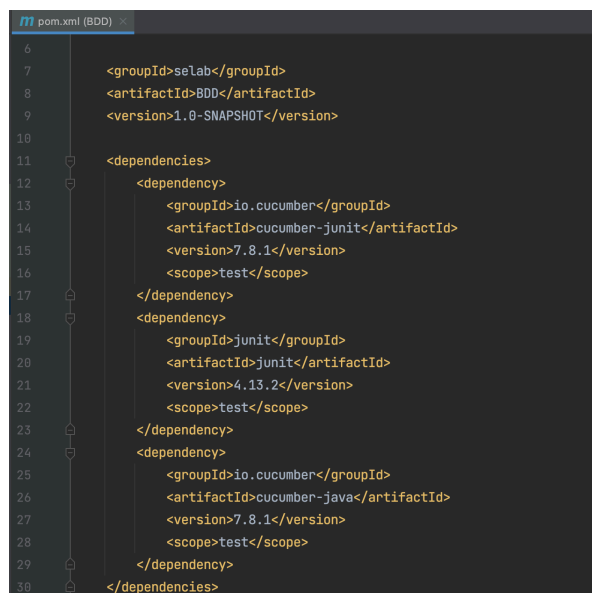
۱.۱ راه اندازی پروژه

۱. یک پروژه‌ی Maven ایجاد می‌کنیم.



شکل ۱: New Maven Project

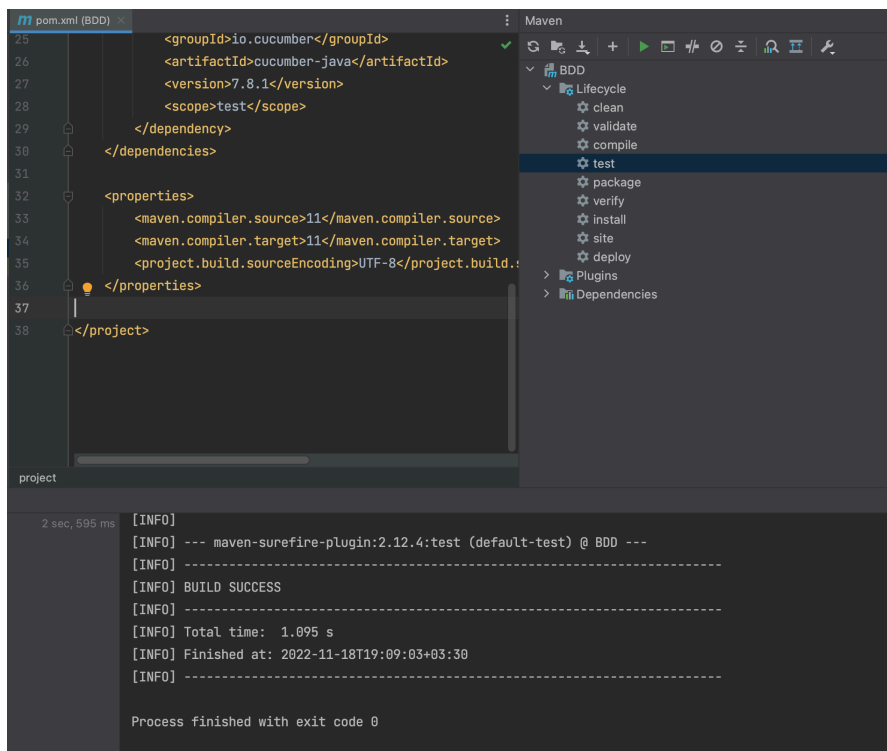
۲. به شکل زیر در فایل pom.xml تغییر ایجاد می‌کنیم و dependency ها را اضافه می‌کنیم. (چون ورژن ما بالاتر بود، بعضی از dependency ها با دستورکار تفاوت دارند)



```
6
7
8 <groupId>selab</groupId>
9 <artifactId>BDD</artifactId>
10 <version>1.0-SNAPSHOT</version>
11
12 <dependencies>
13   <dependency>
14     <groupId>io.cucumber</groupId>
15     <artifactId>cucumber-junit</artifactId>
16     <version>7.8.1</version>
17     <scope>test</scope>
18   </dependency>
19   <dependency>
20     <groupId>junit</groupId>
21     <artifactId>junit</artifactId>
22     <version>4.13.2</version>
23     <scope>test</scope>
24   </dependency>
25   <dependency>
26     <groupId>io.cucumber</groupId>
27     <artifactId>cucumber-java</artifactId>
28     <version>7.8.1</version>
29     <scope>test</scope>
30   </dependency>
31 </dependencies>
```

شکل ۲: Dependencies

۳. Maven->Lifecycle->test را اجرا می‌کنیم و می‌بینیم که به درستی Build می‌شود.



```
25 <groupId>io.cucumber</groupId>
26 <artifactId>cucumber-java</artifactId>
27 <version>7.8.1</version>
28 <scope>test</scope>
29 </dependency>
30 </dependencies>
31
32 <properties>
33   <maven.compiler.source>11</maven.compiler.source>
34   <maven.compiler.target>11</maven.compiler.target>
35   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
36 </properties>
37
38 </project>
```

Maven

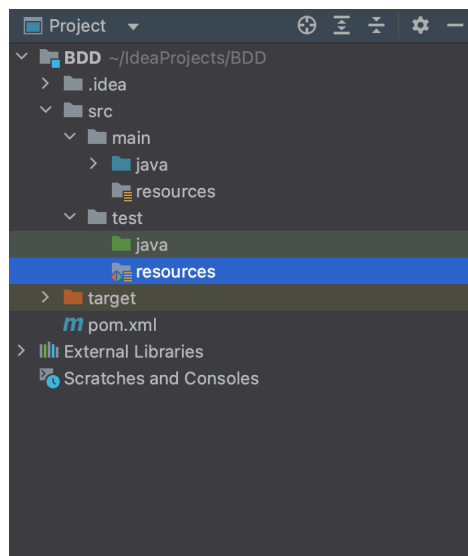
- BDD
 - Lifecycle
 - clean
 - validate
 - compile
 - test
 - package
 - verify
 - install
 - site
 - deploy
 - Plugins
 - Dependencies

project

```
2 sec, 595 ms [INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ BDD ---
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.095 s
[INFO] Finished at: 2022-11-18T19:09:03+03:30
[INFO] -----
Process finished with exit code 0
```

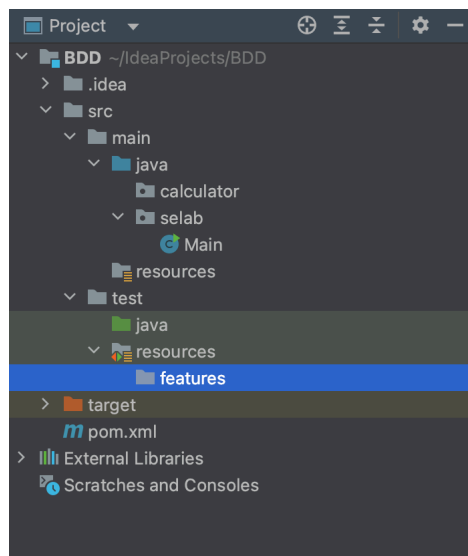
شکل ۳: Successful Build

۴. در پوشه‌ی test یک directory جدید به نام resources درست می‌کنیم و آن را Test Resource Root می‌کنیم.



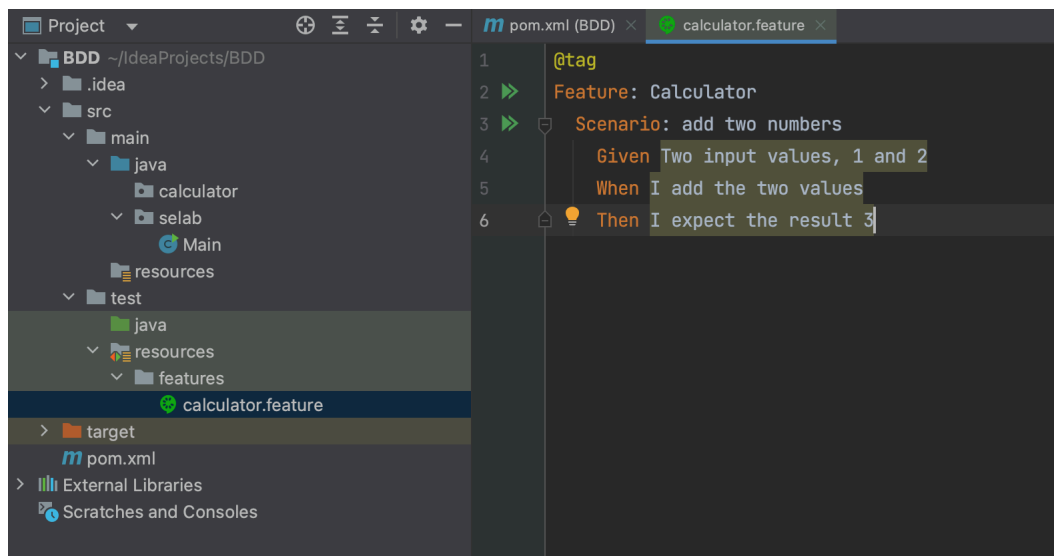
شکل ۴: Test Resource Root

۵. یک package به نام calculator و یک directory به نام features در آدرس‌های گفته‌شده درست می‌کنیم.



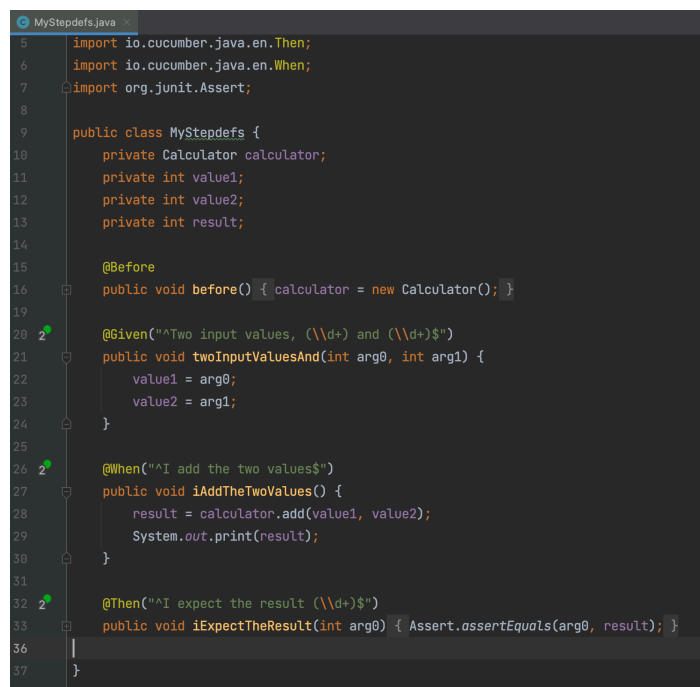
شکل ۵: adding calculator and features to project

۶. در بخش features یک فایل به نام calculator.feature اضافه می‌کنیم و در آن سناریوی جمع دو عدد را می‌نویسیم:



شکل ۶: calculator.feature

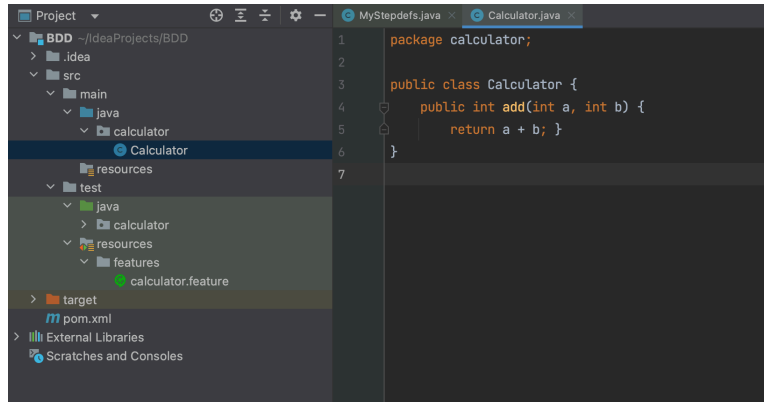
در آدرس test->java یک directory با نام calculator ایجاد می‌کنیم و پس از آن برای هر خط از سناریوی یک step definition درست می‌کنیم. فایل MyStepdefs تولید می‌شود که در آن تغییرات گفته‌شده را اعمال می‌کنیم تا به شکل زیر دربیاید:



شکل ۷: MyStepdefs

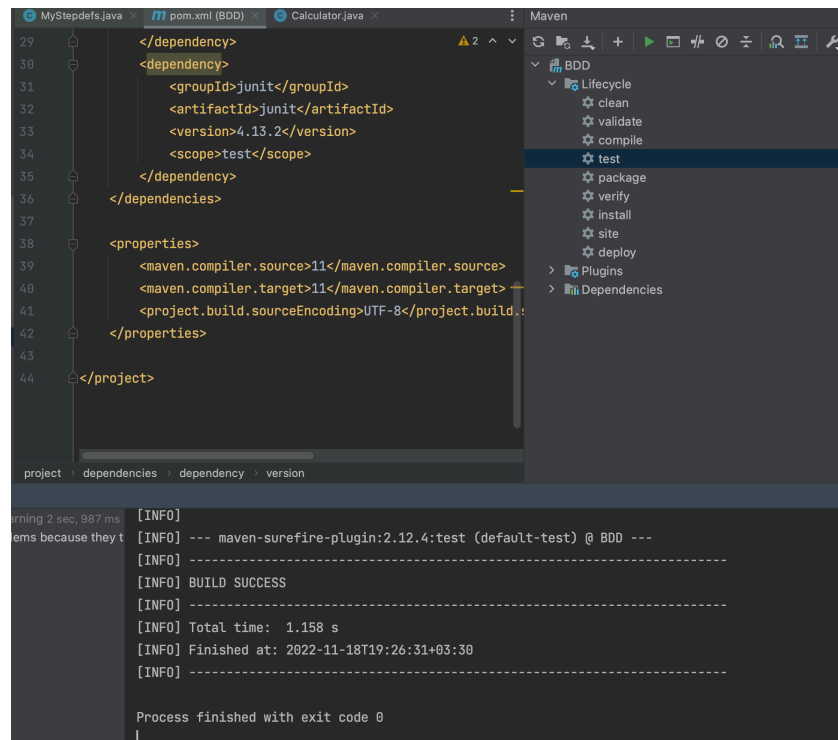
در ابتدا این فایل خطاهایی دارد که برای برطرف کردن‌شان، در مسیر src->main->java->calculator فایل به نام

Calculator را درست می‌کنیم.



شکل ۸: Calculator

خطوط گفته‌شده را به فایل pom.xml اضافه می‌کنیم (البته با توجه به اینکه ما از ورژن ۱۱ استفاده می‌کنیم، به جای ۱.۸ عدد ۱۱ را در خطوط قرار می‌دهیم). و در نهایت دوباره تست را اجرا می‌کنیم تا به درستی build شود.



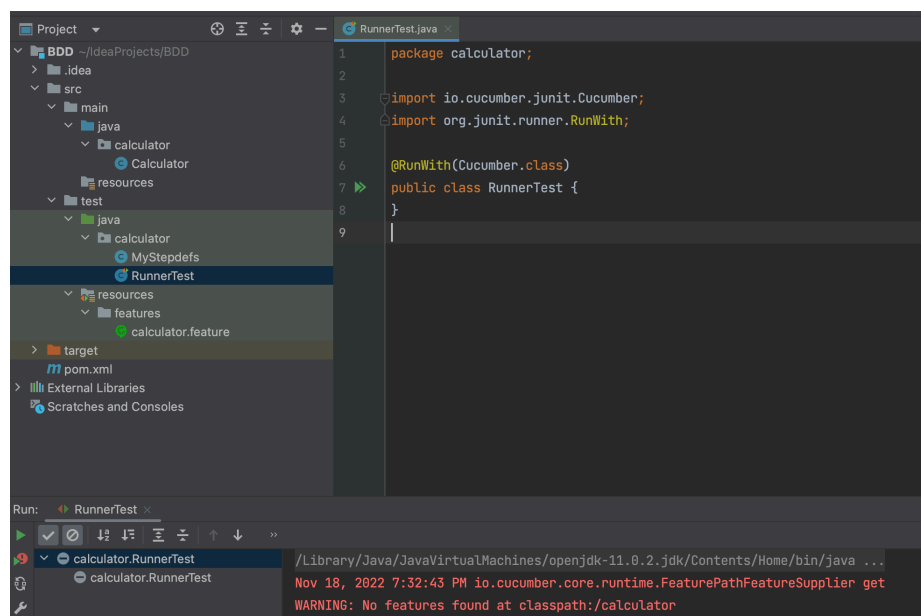
شکل ۹: Successful Build after adding properties to pom.xml

۷. گزینه‌ی 'Feature: calculator' را Run می‌زنیم تا سناریو اجرا شود که نتیجه‌ی آن به شکل زیر است:

```
✓ Done: Scenarios 1 of 1 (917 ms) ⚠
/Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java ...
Testing started at 19:27 ...
3
1 Scenarios (1 passed)
3 Steps (3 passed)
0m0.271s
```

شکل ۱۰ : Successful Scenario

۸. برای مشاهده جزئیات اجرا توسط JUnit یک کلاس جاوا به نام RunnerTest در آدرس test->java->calculator درست می‌کنیم و پس از قرار دادن کدها گفته‌شده، آن را اجرا می‌کنیم که به خطا می‌خوریم.



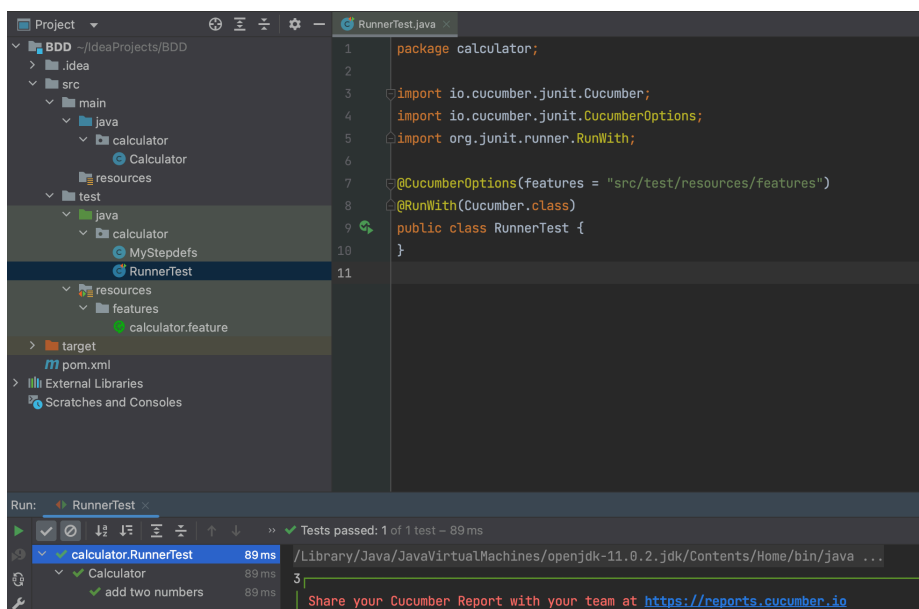
```
Project: BDD -/ideaProjects/BDD
src
  main
    java
      calculator
        Calculator
        resources
      test
        java
          calculator
            MyStepdefs
            RunnerTest
            resources
              features
                calculator.feature
    target
    pom.xml
  External Libraries
  Scratches and Consoles

RunnerTest.java
1 package calculator;
2
3 import io.cucumber.junit.Cucumber;
4 import org.junit.runner.RunWith;
5
6 @RunWith(Cucumber.class)
7 public class RunnerTest {
8 }
9

Run: RunnerTest
calculator.RunnerTest
/Library/Java/JavaVirtualMachines/openjdk-11.0.2.jdk/Contents/Home/bin/java ...
Nov 18, 2022 7:32:43 PM io.cucumber.core.runtime.FeaturePathFeatureSupplier get
WARNING: No features found at classpath:/calculator
```

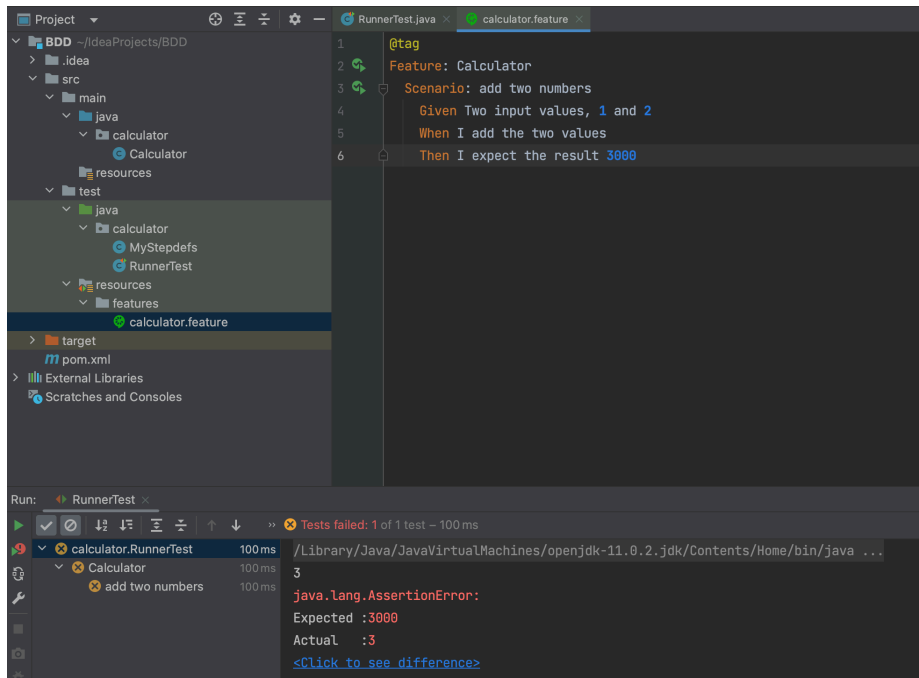
شکل ۱۱ : RunnerTest.java

۹. تغییرات گفته شده را می دهیم تا مشکل برطرف شود.



شکل ۱۲: RunnerTest.java 2.0

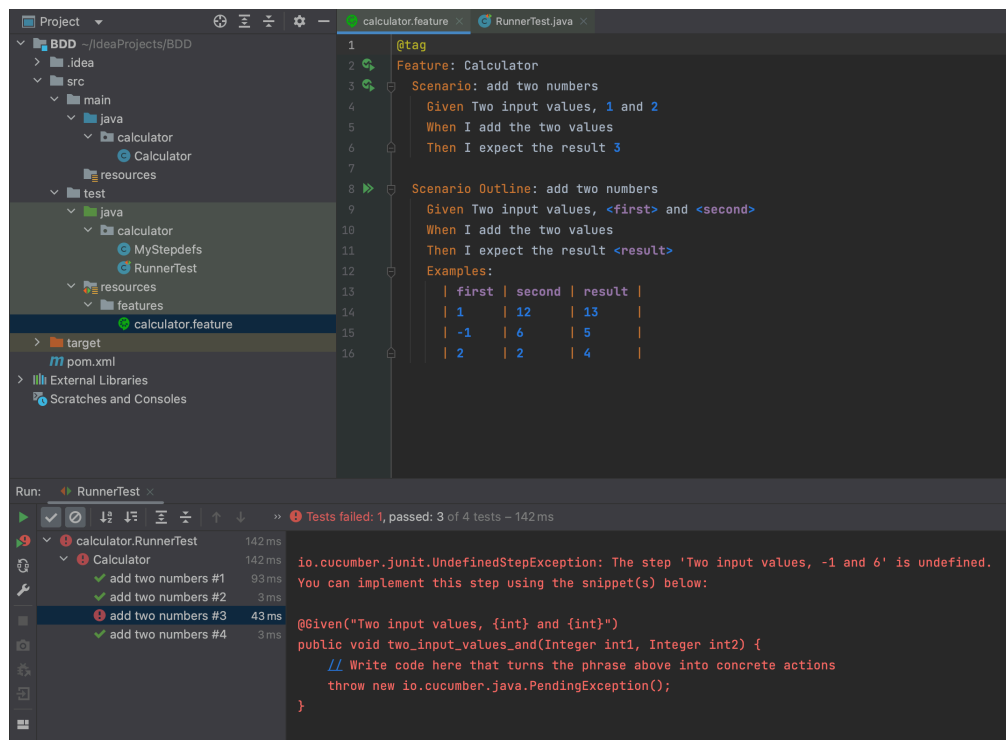
همچنین می بینیم که اگر در سناریو، حاصل جمع را به جای ۳ عدد ۳۰۰۰ قرار بدهیم، تست پاس نمی شود.



شکل ۱۳: Failed Scenario

۲.۱ اجرای scenario outline

سناریو را به شکل زیر در ادامه‌ی فایل calculator.feature تعریف می‌کنیم و با اجرای RunnerTest می‌بینیم که تست سوم (که ورودی‌های آن -۱ و ۶ هستند) پاس نمی‌شوند و به مشکل undefined می‌خورند.



شکل ۱۴: Test 3 Failed

علت این مشکل این است که برای بخش Given به شکل

@Given("Two input values, (\\d+)and(\\d+)\$")

نوشته شده است. بنابراین عددی منفی یا اعدادی که با علامت‌های + و - شروع می‌شوند، با این رجکس میچ نمی‌شوند (که در این تست عدد -۱ دچار این مشکل می‌شود و چون علامت - را نمی‌توان با هیچ بخشی از نوشته‌ی بالا میچ کرد، ارور می‌گیریم). پس در همین زمینه باید تغییر ایجاد کنیم. کد جدید به شکل زیر است:


```

9      public class MyStepdefs {
10         private Calculator calculator;
11         private int value1;
12         private int value2;
13         private int result;
14
15         @Before
16         public void before() {
17             calculator = new Calculator();
18         }
19
20         @Given("Two input values, {int} and {int}")
21         public void twoInputValuesAnd(int arg0, int arg1) {
22             value1 = arg0;
23             value2 = arg1;
24         }
25
26         @When("I add the two values")
27         public void iAddTheTwoValues() {
28             result = calculator.add(value1, value2);
29             System.out.print(result);
30         }
31
32         @Then("I expect the result {int}")
33         public void iExpectTheResult(int arg0) {
34             Assert.assertEquals(arg0, result);
35         }
36     }

```

شکل ۱۵: New MyStepdefs

این بار به جای فرمت d+ که فقط تعداد یک یا بیشتر رقم را دربر می‌گیرد، از int استفاده می‌کنیم که همه‌ی اعداد صحیح را (مثبت و منفی) می‌کند. پس از اجرای دوباره‌ی RunnerTest می‌بینیم که این بار همه‌ی تست‌ها پاس می‌شوند.

The screenshot shows a test runner interface. On the left, a tree view shows the test results: 'calculator.RunnerTest' (196 ms) is expanded, showing four sub-items: 'Calculator' (196 ms), 'add two numbers #1' (171 ms), 'add two numbers #2' (12 ms), and 'add two numbers #3' (6 ms). All tests are marked with green checkmarks. On the right, a configuration panel for Cucumber publishing is visible, showing the following settings:

```

src/test/resources/cucumber.properties: cucumber.publish.enabled=true
src/test/resources/junit-platform.properties: cucumber.publish.enabled=true
Environment variable: CUCUMBER_PUBLISH_ENABLED=true
JUnit: @CucumberOptions(publish = true)

```

شکل ۱۶: All Tests Passed

کدهای این قسمت در پوشه‌ای با نام BDD پیوست شده است.

۲

<https://github.com/Miraneh/SoftwareLab/HW3>