

بسمه تعالی

دستور کار آزمایشگاه مهندسی نرم افزار

دانشگاه صنعتی شریف

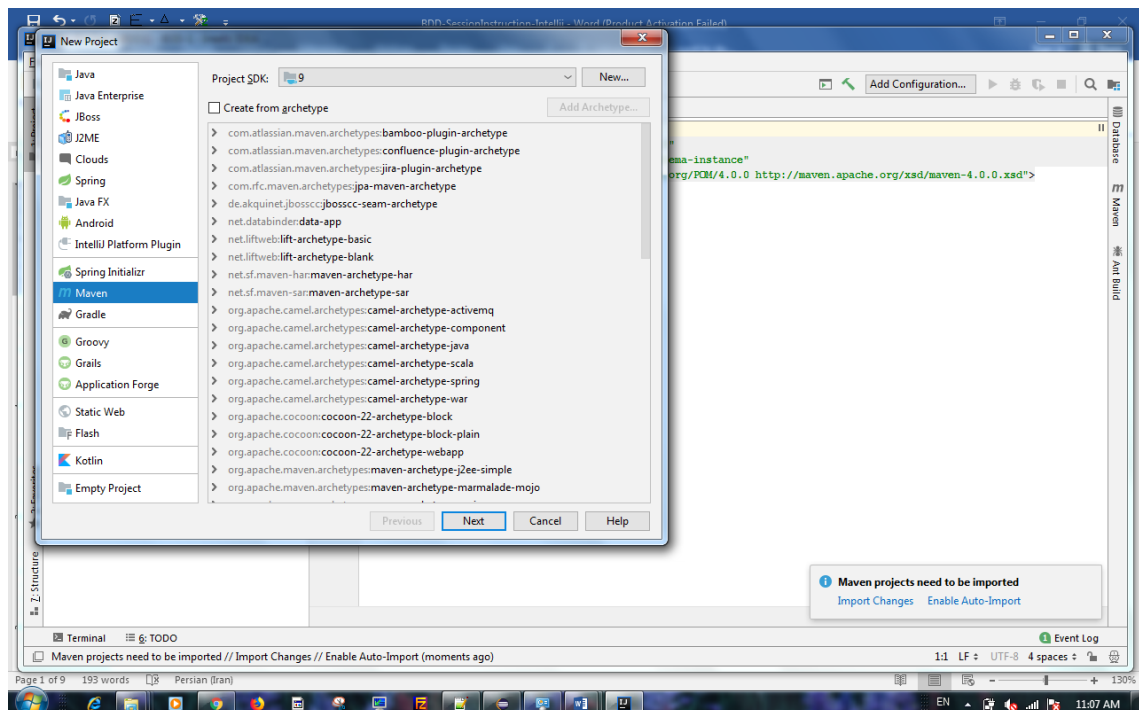
دانشکده مهندسی کامپیوتر

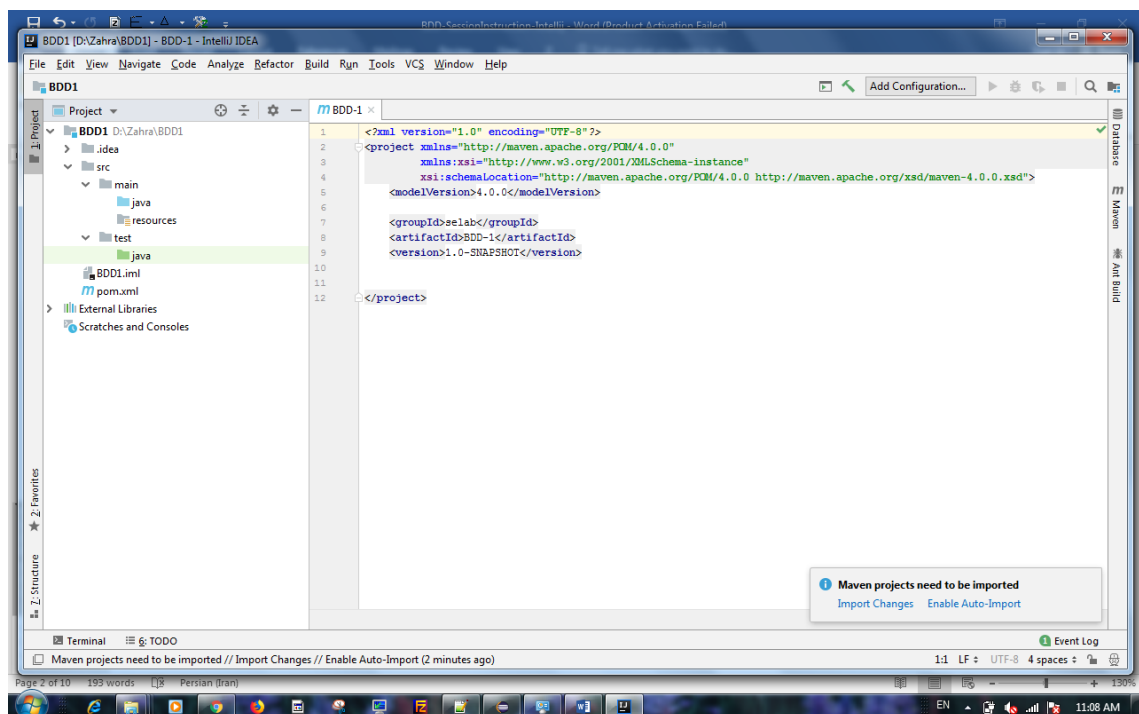
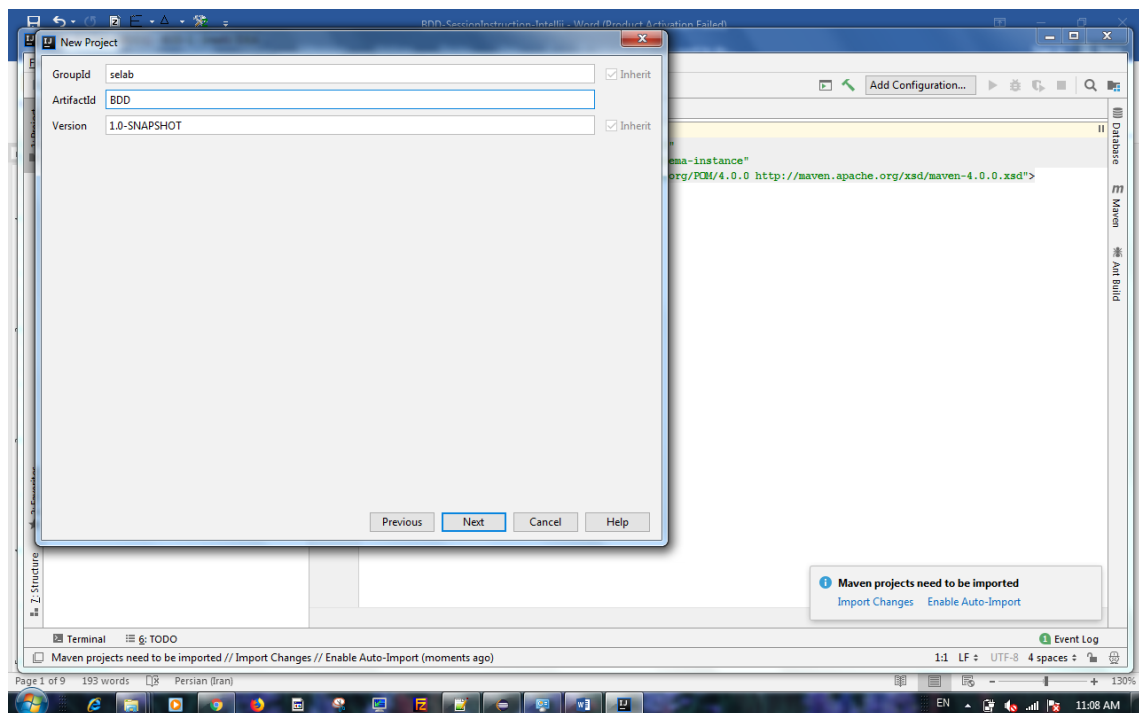
ارائه مثالی از تبدیل نیازمندی ها به موارد آزمون با استفاده از روش ایجاد مبتنی بر رفتار (BDD)

سناریو: جمع دو عدد

مراحل انجام کار در محیط IntelliJ:

۱- ایجاد پروژه جدید در IntelliJ:





۲- افزودن dependency های موردنیاز شامل JUnit و Cucumber به فایل pom.xml پروژه:

```
<dependencies>
  <dependency>
    <groupId>info.cukes</groupId>
```

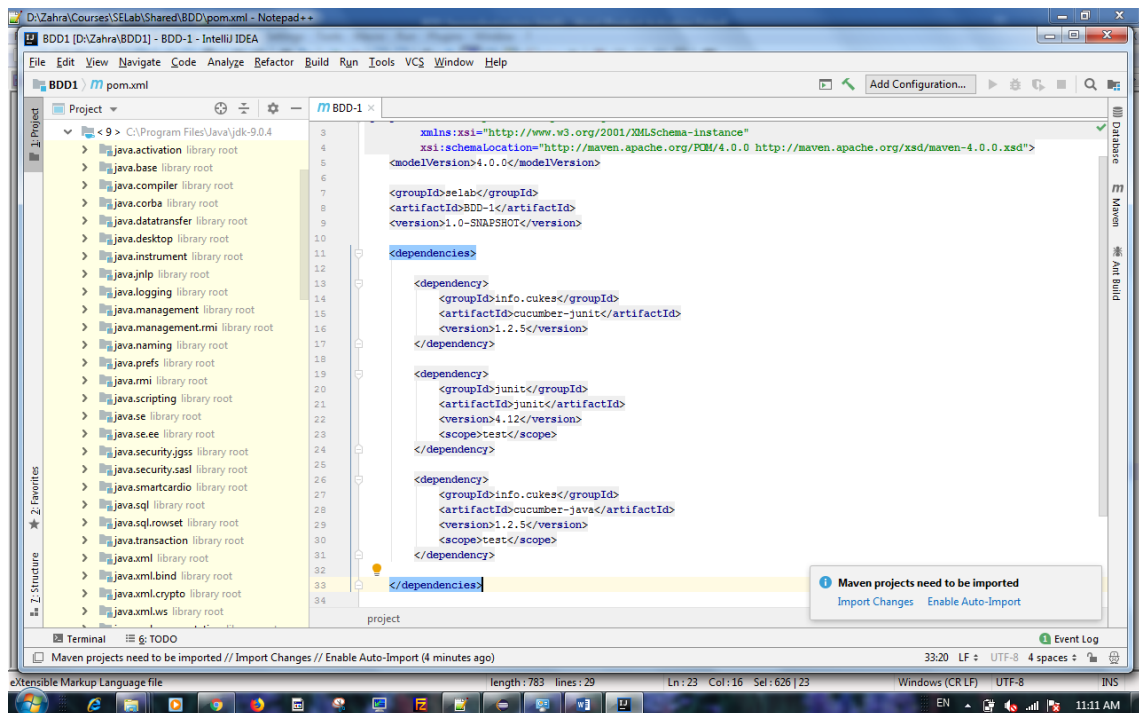
```

    <artifactId>cucumber-junit</artifactId>
    <version>1.2.5</version>
</dependency>

<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>

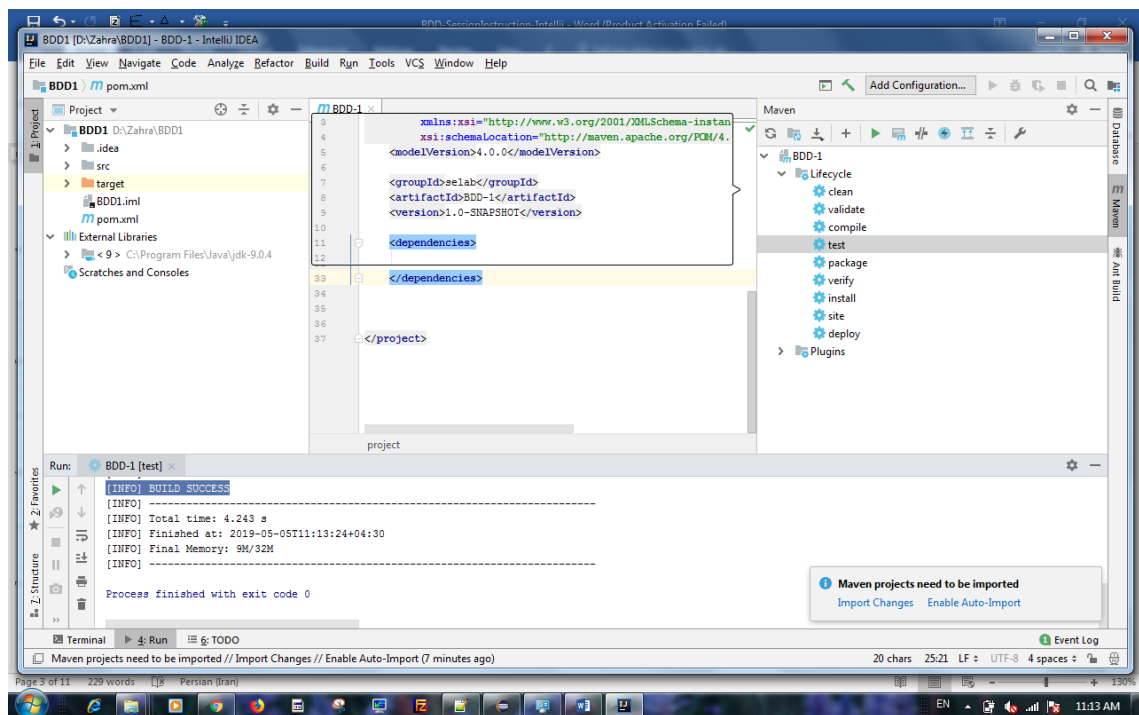
<dependency>
    <groupId>info.cukes</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>1.2.5</version>
    <scope>test</scope>
</dependency>
</dependencies>

```



۳- اجرای Maven->Test از طریق navigation bar سمت راست صفحه و اطمینان از انجام موفق آن با دیدن پیغام BUILD SUCCESS. (در این زمان برای دانلود dependencyها نیاز به اینترنت است و در صورتی که

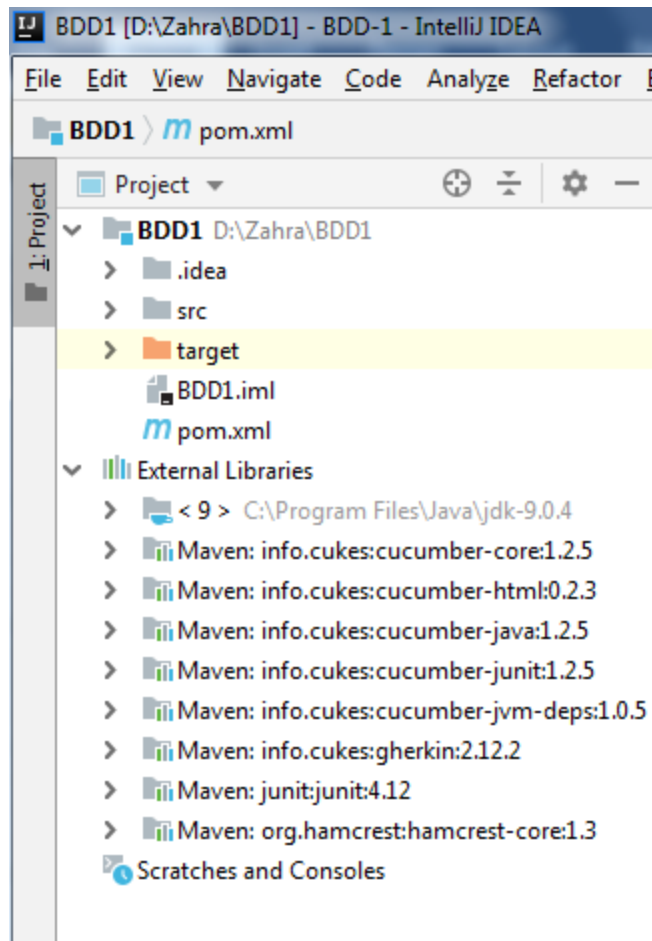
ارتباط برقرار نباشد با خطا مواجه می‌شود).



۴- زدن گزینه‌های Import Changes و Enable Auto-Import از پیغام گوشه سمت راست پایین صفحه

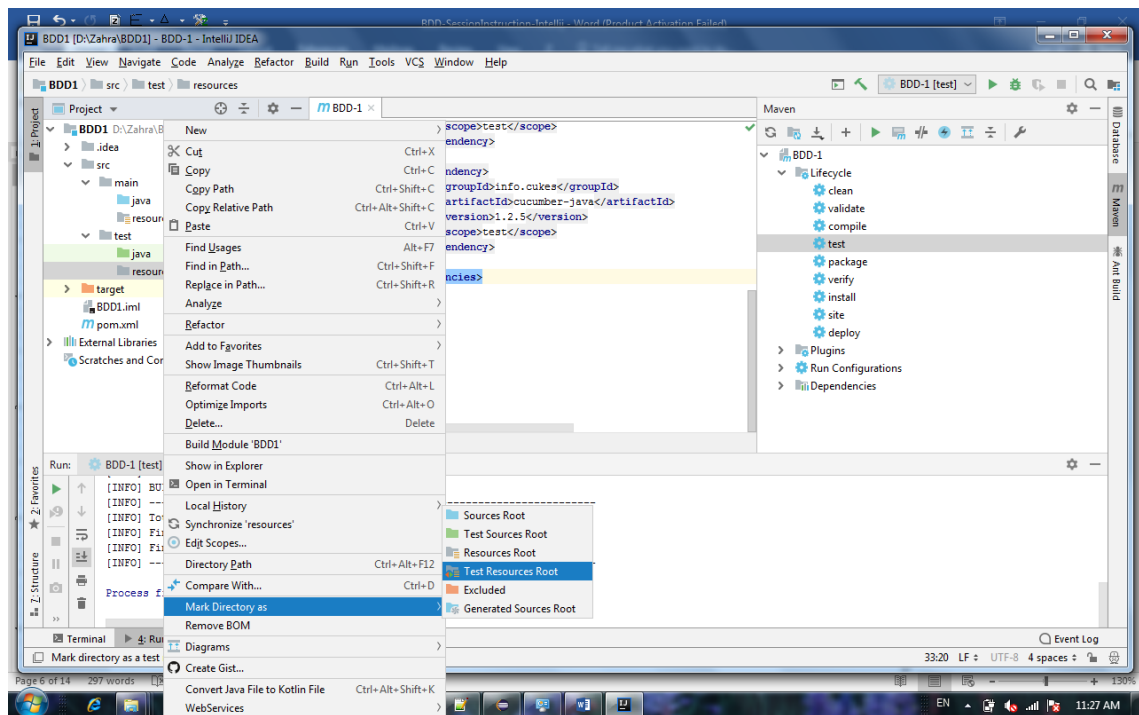
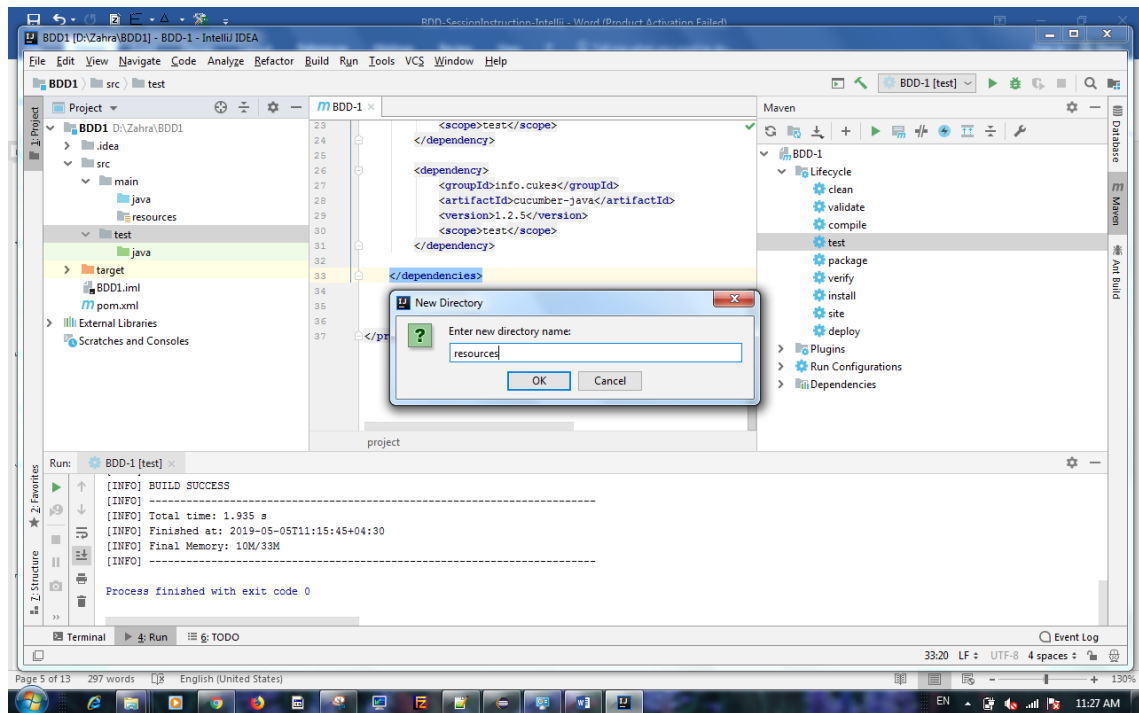
جهت Maven projects need to be imported

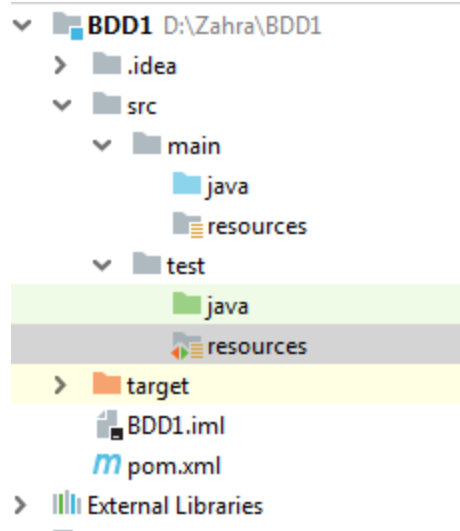
۵- با انجام این مراحل، ملزومات راه‌اندازی می‌شود:



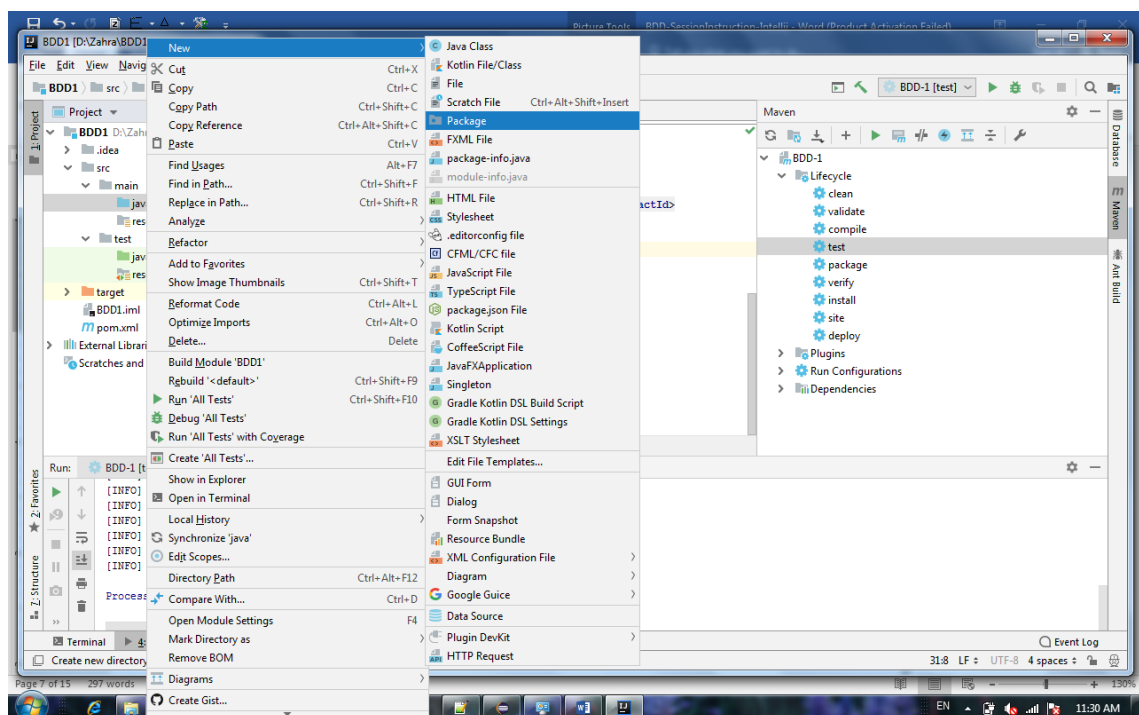
۶- ایجاد یک پکیج جدید به نام resources در پکیج test پروژه و راست کلیک بر روی آن و انتخاب Mark

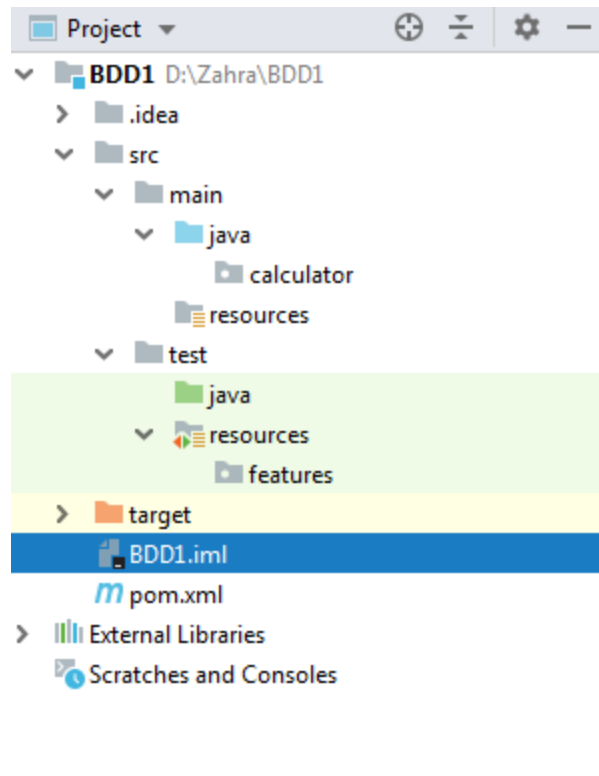
Directory as Test Resource Root



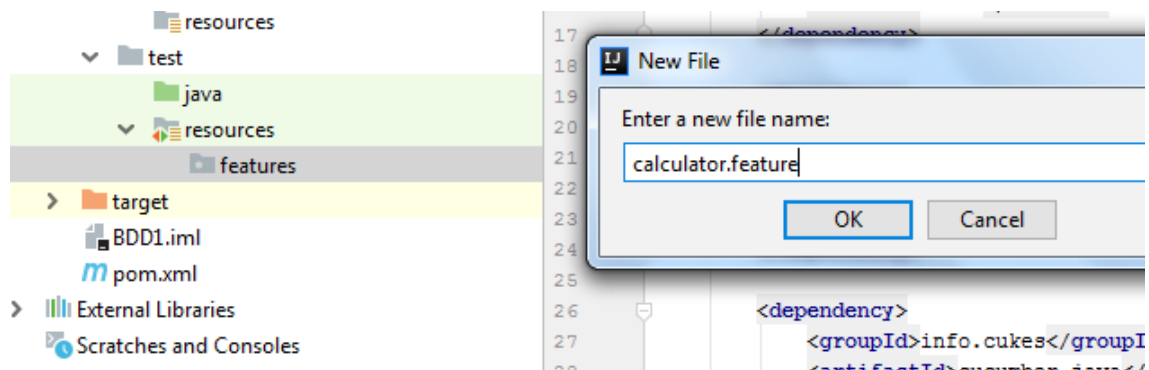


۷- ایجاد پکیج جدید با نام calculator در مسیر src->main->java و ایجاد دایرکتوری جدید با نام features در مسیر test->resources





۸- ایجاد فایل جدید با عنوان calculator.feature در زیر دایرکتوری feature و نوشتن سناریوی جمع دو عدد در آن



خطوط زیر به عنوان سناریوی جمع در این فایل نوشته می‌شوند:

@tag

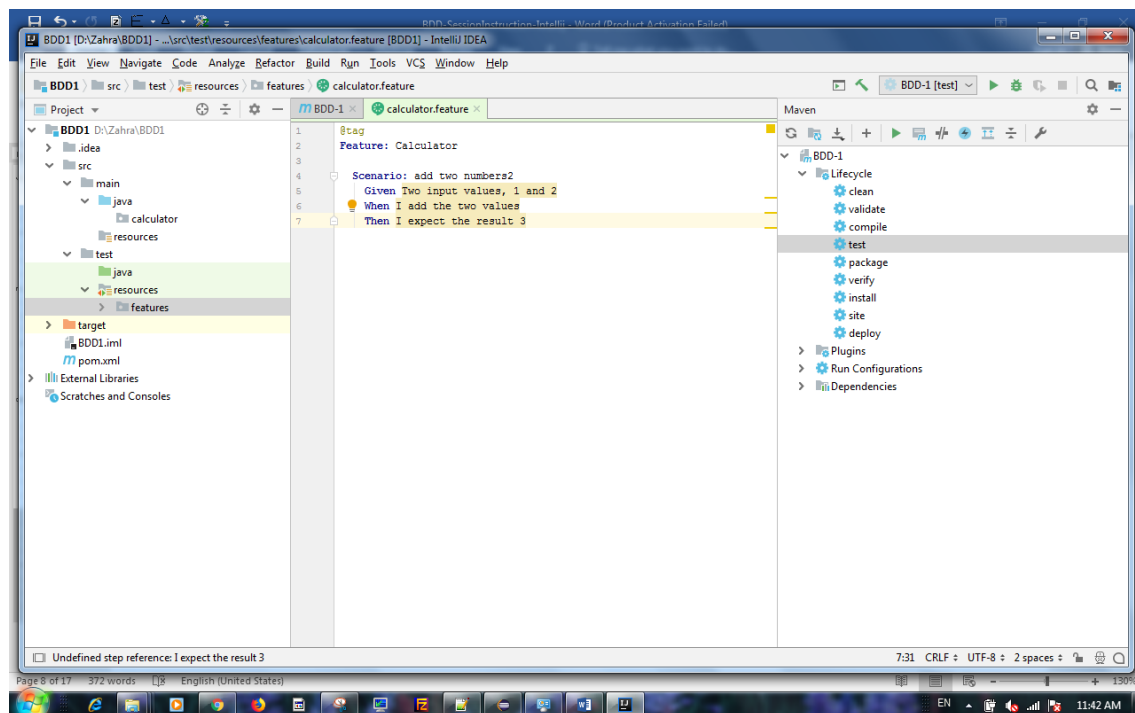
Feature: Calculator

Scenario: add two numbers

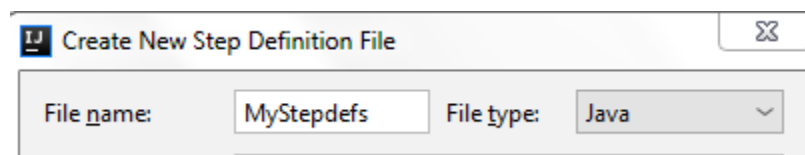
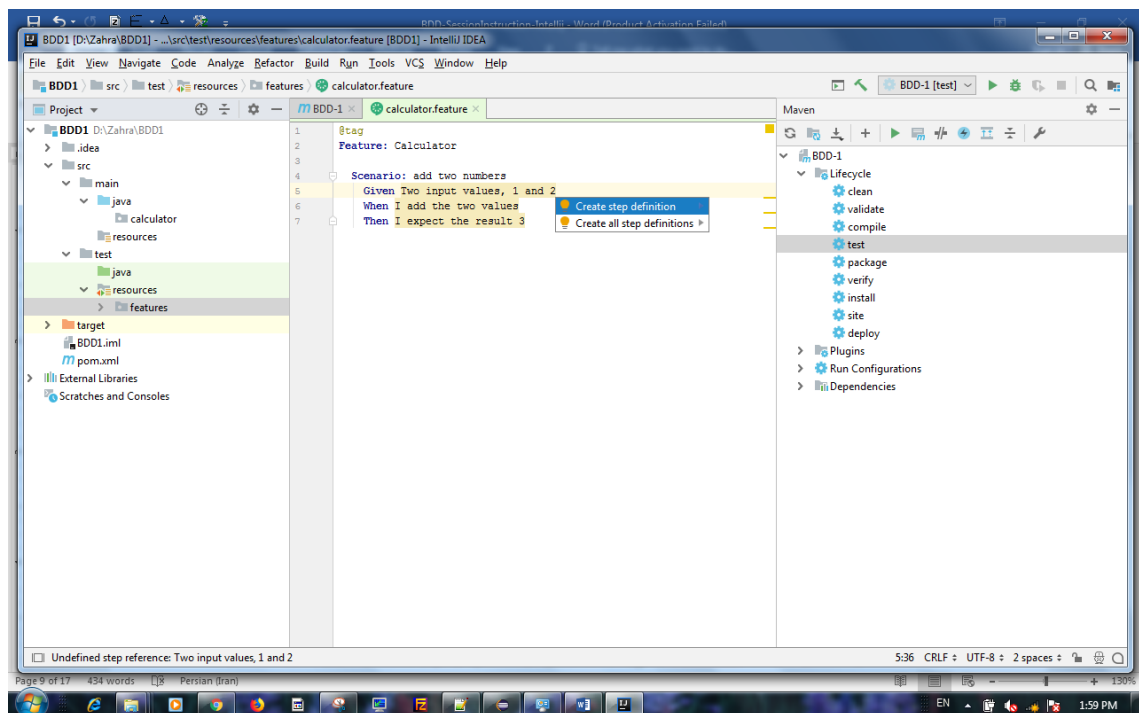
Given Two input values, 1 and 2

When I add the two values

Then I expect the result 3



زبان نوشتن featureها Gherkin است که باید توسط cucumber ترجمه و اجرا شود. با ذخیره فایل، برخی از خطوط به رنگ زرد درمی آیند زیرا مشخص نیست باید چگونه تفسیر شوند. برای حل این مشکل، به انتهای هر یک از خطوط Given، When و Then رفته و کلید Alt+Enter را می زنیم تا یک فایل جهت تعیین مراحل و stepهای اجرای سناریو ایجاد شود. قبل از این کار یک دایرکتوری با نام calculator در مسیر test->java ایجاد می کنیم تا در آن ساخته شود:



متدهای بدون بدنه به صورت زیر در فایل MyStepdefs ایجاد می شوند:

```
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;

public class MyStepdefs {
    @Given("^Two input values, (\\d+) and (\\d+)$")
    public void twoInputValuesAnd(int arg0, int arg1) {
    }

    @When("^I add the two values$")
    public void iAddTheTwoValues() {
    }

    @Then("^I expect the result (\\d+)$")
    public void iExpectTheResult(int arg0) {
    }
}
```

حال لازم است بدنه این متدها به صورتی که موردنظر است پیاده سازی شود؛ ابتدا Given، When، Then و در نهایت

.Before

```
import cucumber.api.java.Before;
import cucumber.api.java.en.Given;
import cucumber.api.java.en.Then;
import cucumber.api.java.en.When;
import org.junit.Assert;

public class MyStepdefs {
    private Calculator calculator;
    private int value1;
    private int value2;
    private int result;

    @Before
    public void before() {
        calculator = new Calculator();
    }

    @Given("^Two input values, (\\d+) and (\\d+)$")
    public void twoInputValuesAnd(int arg0, int arg1) {
        value1 = arg0;
        value2 = arg1;
    }

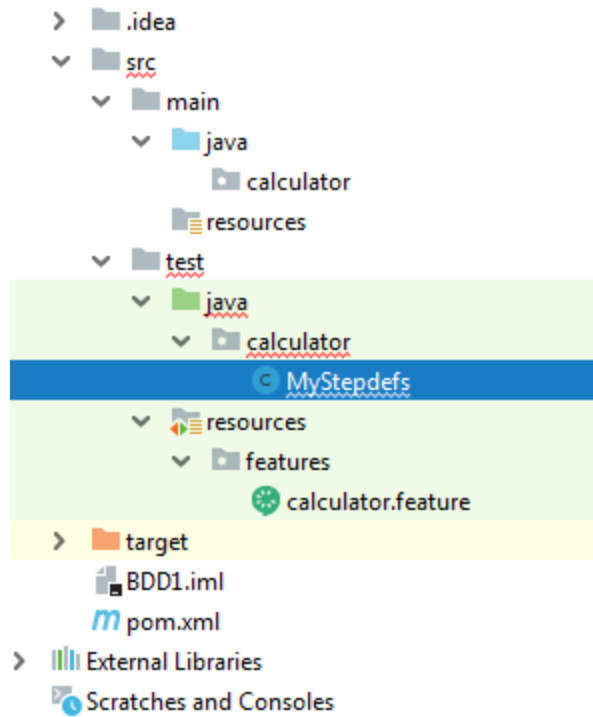
    @When("^I add the two values$")
    public void iAddTheTwoValues() {
        result = calculator.add(value1, value2);
        System.out.print(result);
    }

    @Then("^I expect the result (\\d+)$")
    public void iExpectTheResult(int arg0) {
        Assert.assertEquals(arg0, result);
    }
}
```

با ذخیره این فایل، برخی فایل های پروژه به compile error برمی خورند؛ زیرا Calculator هنوز تعریف نشده است.

این کلاس باید پس از ایجاد دایرکتوری calculator، در مسیر زیر ساخته شود:

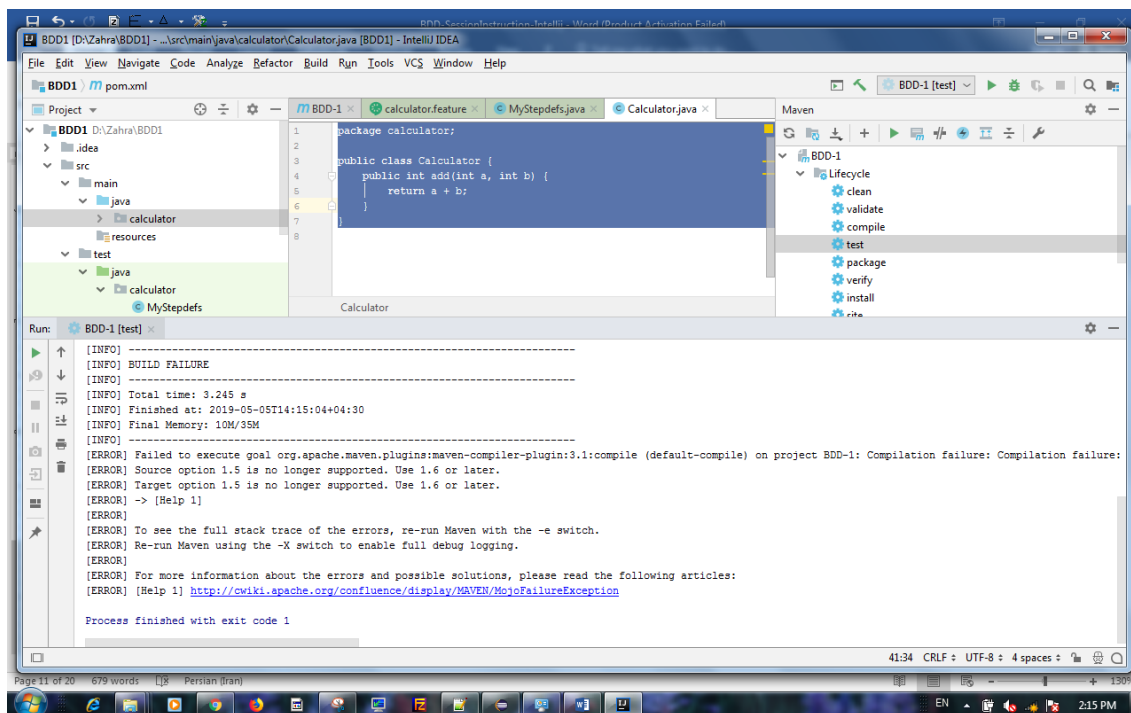
src->main->java->calculator



```
package calculator;
```

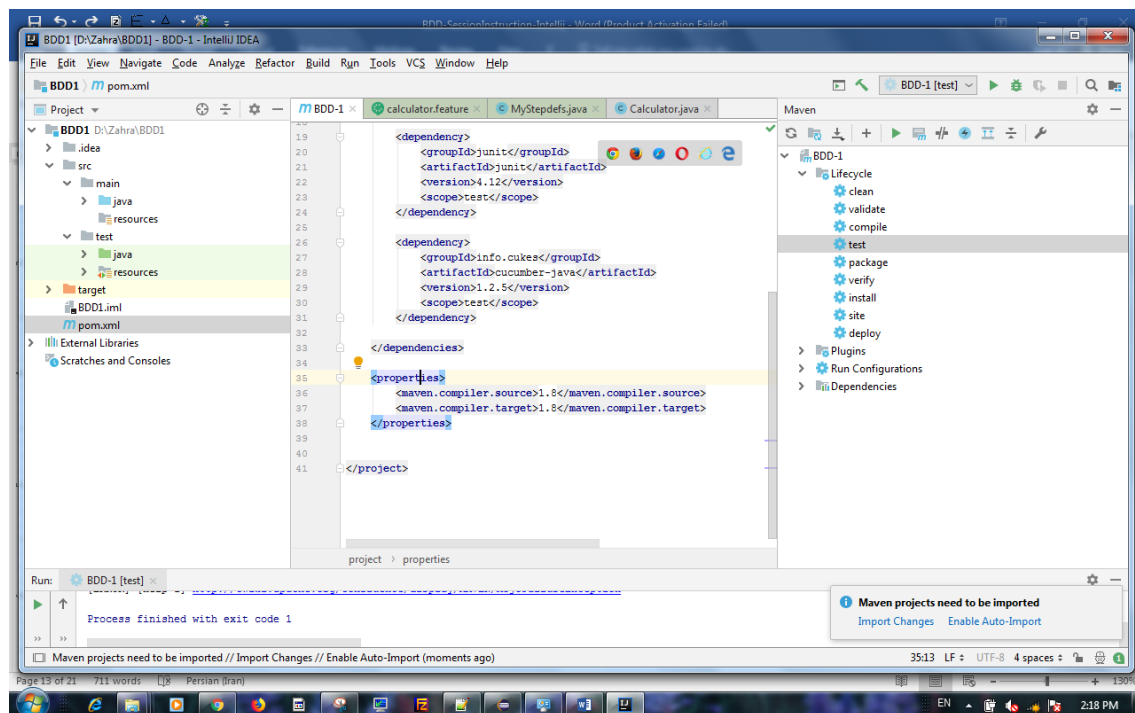
```
public class Calculator {  
    public int add(int a, int b) {  
        return a + b;  
    }  
}
```

۹- زدن گزینه Maven test که در این صورت به ERROR برمی خوریم:

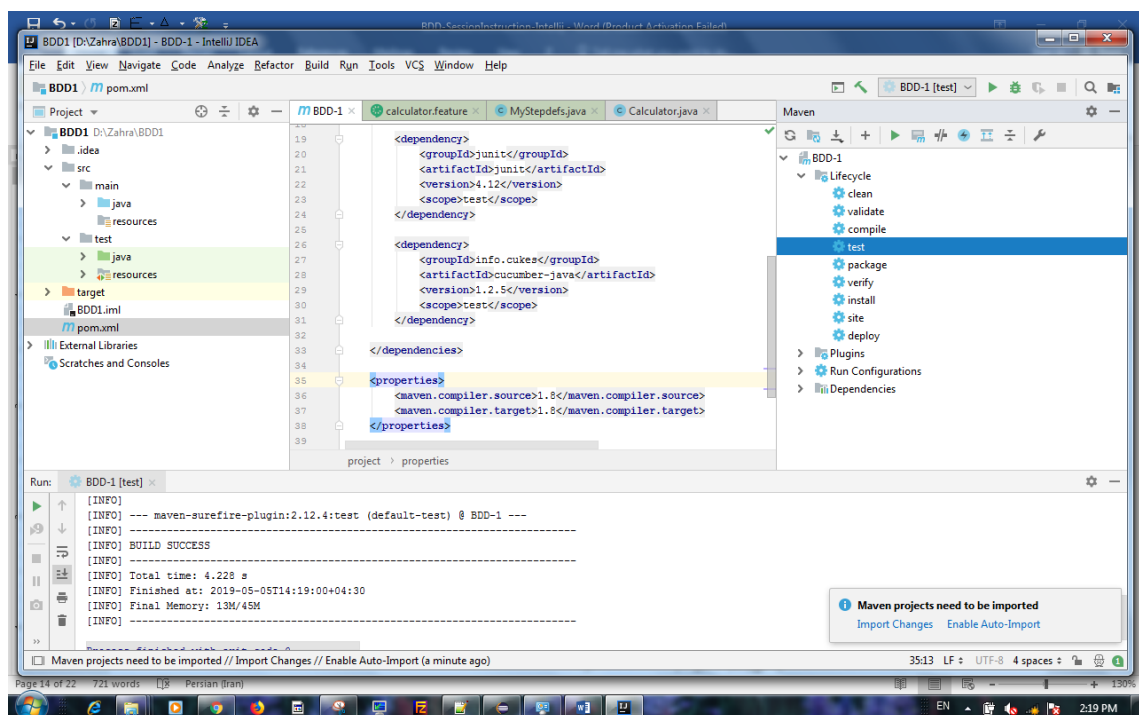


برای رفع این خطا لازم است نسخه بالاتر از Maven را در فایل pom.xml معرفی کنیم. بنابراین خطوط زیر را پس از پایان dependencies به این فایل اضافه می‌کنیم:

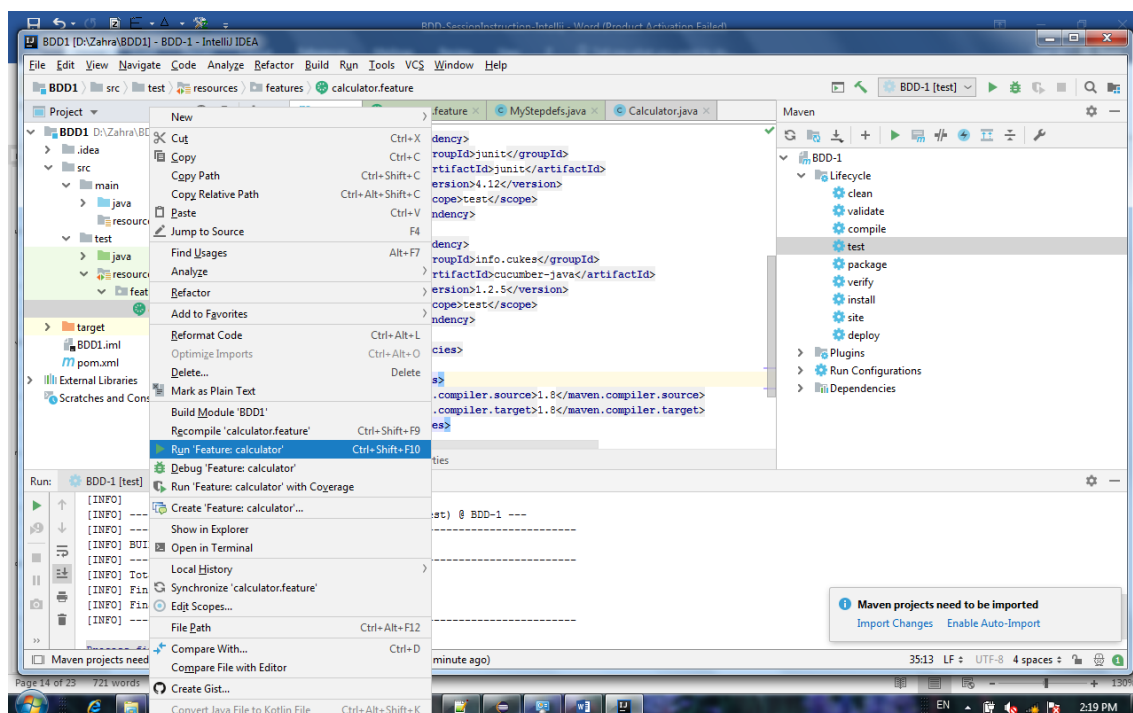
```
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```



مجدداً Maven test را می‌زنیم تا این بار SUCCESS شود:



۱۰- بر روی فایل calculator.feature کلیک راست کرده و گزینه Run را انتخاب می‌کنیم تا سناریوی این feature اجرا شود:



نتیجه اجرا باید به صورت زیر باشد که نشان می دهد یک سناریو با موفقیت تست شده است:

```

✓ Done: Scenarios 1 of 1 (3 s 447 ms)

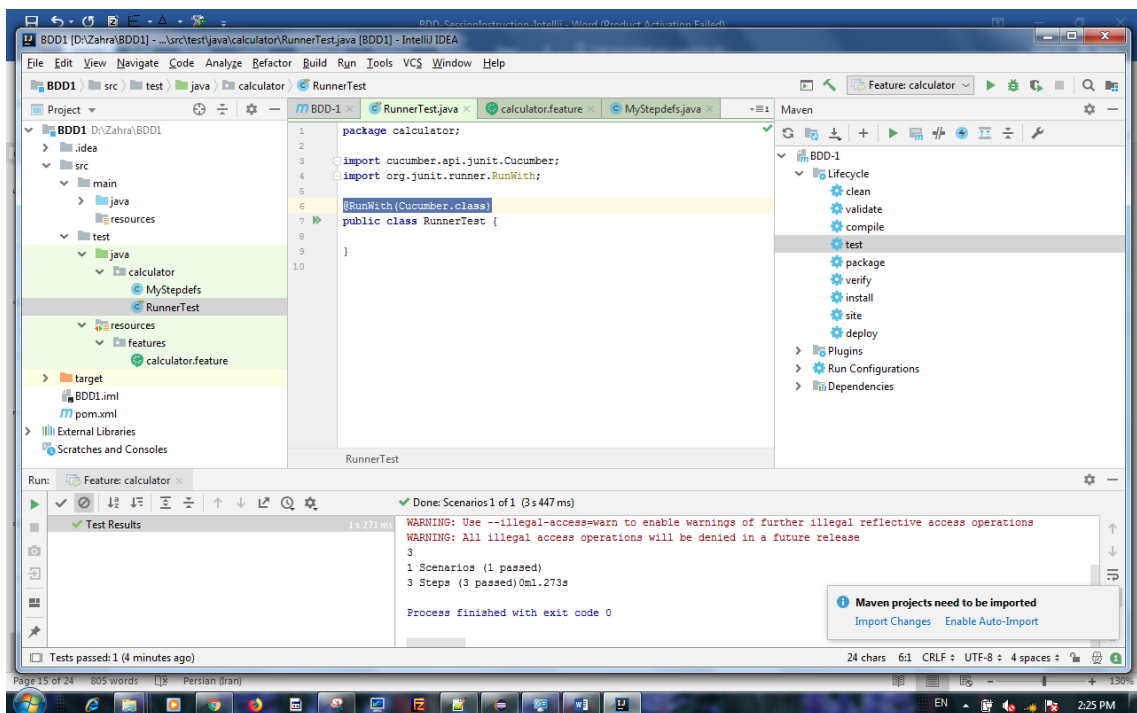
WARNING: Use --illegal-access=warn to enable warnings of
WARNING: All illegal access operations will be denied in
3
1 Scenarios (1 passed)
3 Steps (3 passed) 0m1.273s

Process finished with exit code 0

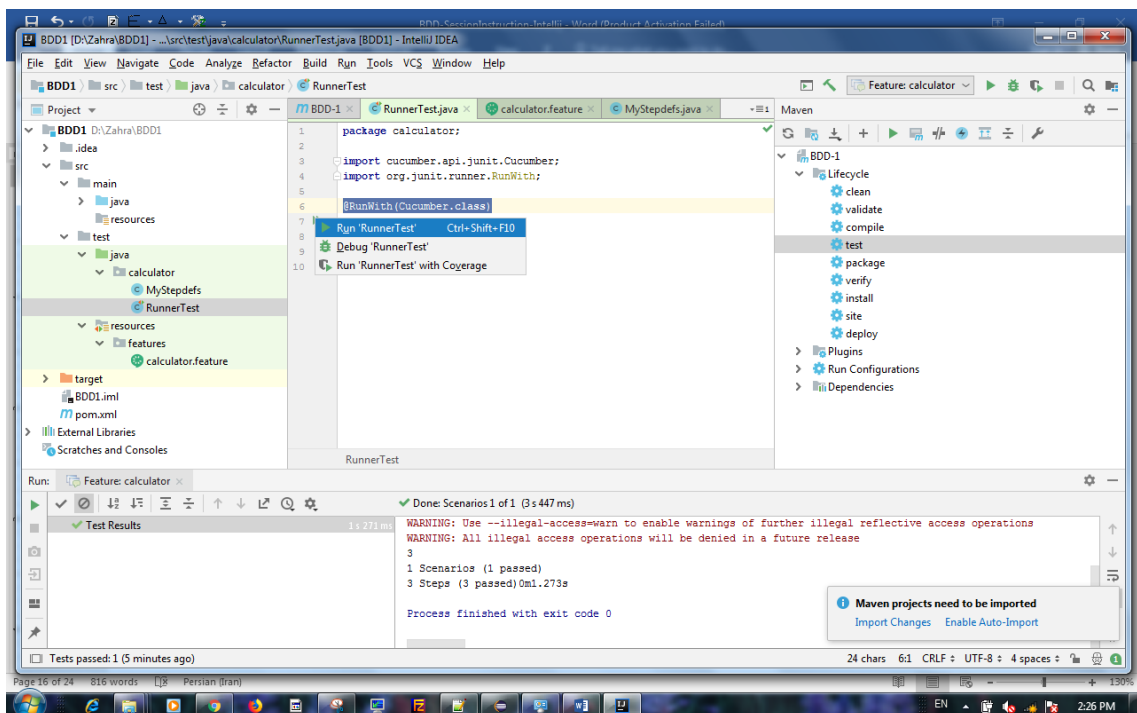
```

۱۱- برای مشاهده جزئیات اجرا از طریق JUnit به این صورت عمل می کنیم که یک کلاس جدید به نام RunnerTest.java در مسیر calculator->java->test (کنار فایل MyStepdefs) ایجاد می نماییم و بدنه آن را به صورت زیر پر می کنیم که بالای سر امضای متد، عبارت زیر را قرار می دهیم:

```
@RunWith(Cucumber.class)
```



با زدن فلش سبز رنگ کنار نام کلاس، این فایل اجرا می شود ولی این بار به یک خطای جدید برمی خورد:

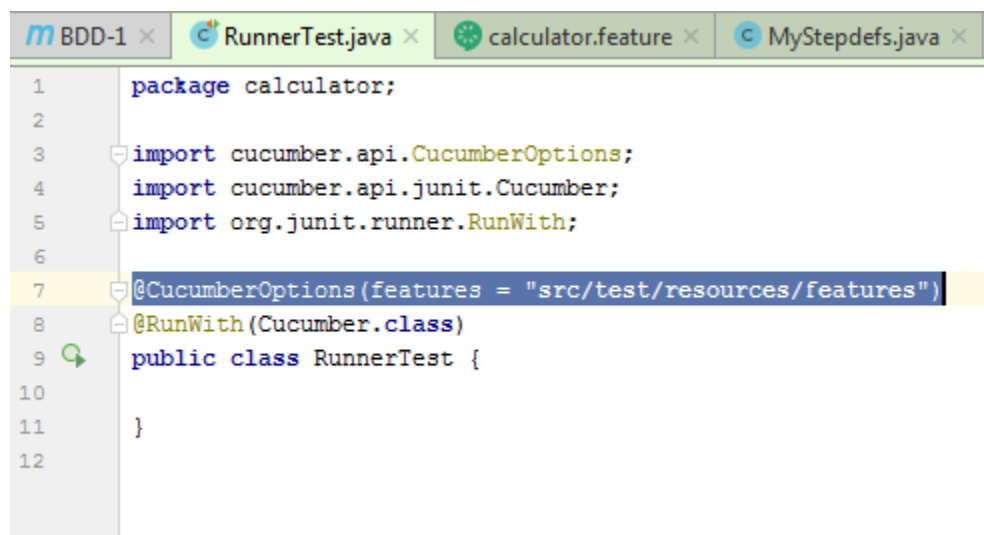



```
"C:\Program Files\Java\jdk-9.0.4\bin\java.exe" ...
No features found at [classpath:calculator]
0 Scenarios
0 Steps
0m0.000s

Process finished with exit code 0
```

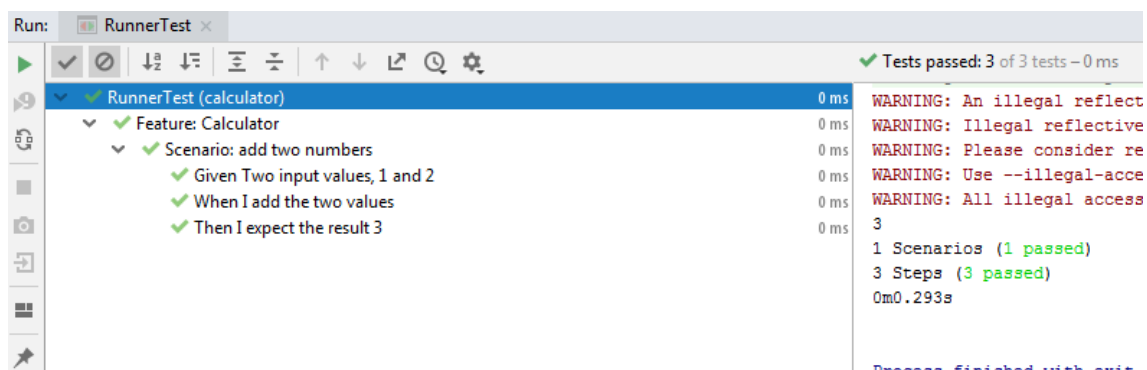
این خطا نشان می‌دهد که فایل feature را نمی‌تواند پیدا کند. برای رفع آن باید مسیر این فایل را در ابتدای کلاس RunnerTest با کمک `@CucumberOptions` مشخص کنیم:

```
@CucumberOptions(features = "src/test/resources/features")
```



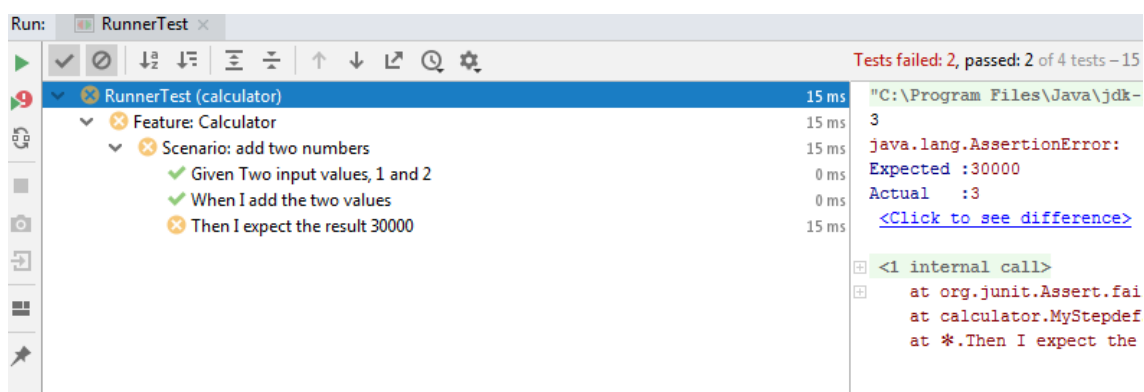
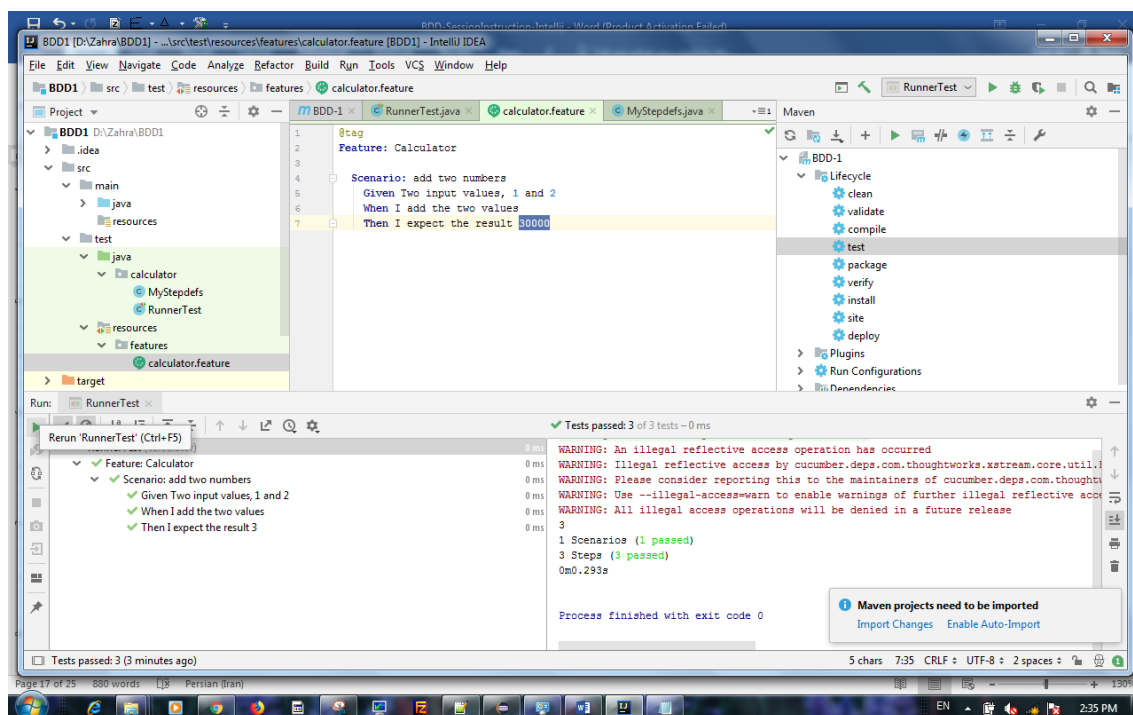
```
1 package calculator;
2
3 import cucumber.api.CucumberOptions;
4 import cucumber.api.junit.Cucumber;
5 import org.junit.runner.RunWith;
6
7 @CucumberOptions(features = "src/test/resources/features")
8 @RunWith(Cucumber.class)
9 public class RunnerTest {
10
11 }
12
```

۱۲- با اجرای مجدد RunnerTest، خروجی به صورت زیر نمایش داده می‌شود:



```
Run: RunnerTest x
Tests passed: 3 of 3 tests - 0 ms
RunnerTest (calculator) 0 ms
  Feature: Calculator 0 ms
    Scenario: add two numbers 0 ms
      Given Two input values, 1 and 2 0 ms
      When I add the two values 0 ms
      Then I expect the result 3 0 ms
1 Scenarios (1 passed)
3 Steps (3 passed)
0m0.293s
Process finished with exit
```

۱۳- با تغییر مقادیر فایل feature و اعداد می توان نتیجه تست را مشاهده کرد:



حال نوبت به تعریف نوع دیگری از سناریو به نام scenario outline است که آن را در ادامه فایل feature به صورت زیر تعریف می کنیم:

Scenario Outline: add two numbers
 Given Two input values, <first> and <second>
 When I add the two values
 Then I expect the result <result>

Examples:

first	second	result
1	12	13
-1	6	5
2	2	4

RunnerTest را دوباره اجرا می‌کنیم. این بار برخی تست‌ها به مشکل undefined برمی‌خورند. این موارد تست کدامند؟ علت بروز مشکل چیست؟