

IF32204/IF42204 – Pengujian Perangkat Lunak

BDD- Automated Testing with Selenium and Cucumber

Week/Session	:	14/2
Purpose	:	<ul style="list-style-type: none">▪ Cucumber terpasang di komputer mahasiswa▪ Mahasiswa mampu menuliskan <i>scenario</i> dan <i>step definition</i> pada Cucumber, dengan menguji suatu sistem.▪ Mahasiswa memahami konsep BDD
Deliverable	:	<ul style="list-style-type: none">▪ Laporan → BDD_XXX.pdf▪ Sebuah Java Project → BDD_XXX.zip
Deadline	:	IF32204 Friday, 3 May 2019 at 23.00 p.m. IF42204 Friday, 3 May 2019 at 23.00 p.m.

I. Testing with Cucumber

Cucumber adalah sebuah *testing framework* yang dapat membantu kesenjangan antara *software-developers* dan *business managers*. *Tests* dituliskan dalam bahasa yang sederhana berdasarkan *behavior-driven development* (BDD) *style*, seperti **Given, When, Then**; yang mana dapat dimengerti oleh orang awam. Kemudian, *Test cases* diletakkan pada *feature files* yang mencakup satu atau lebih *test scenario*. Cucumber menterjemahkan *tests* ke dalam Bahasa pemrograman yang ditentukan dan menggunakan Selenium untuk menjalankan *test cases* pada sebuah *browser*. Pada praktikum ini, *Tests* tersebut diterjemahkan ke dalam Bahasa Java.

II. Gherkin Syntax

Gherkin syntax menggunakan sekumpulan *keywords* untuk membuat struktur dan pendefinisian pada *executable specifications*. Setiap keyword diterjemahkan ke dalam banyak bahasa; namun pada praktikum ini kita menggunakan Bahasa Inggris.

Keyword yang dimaksud terdiri atas *primary keywords* dan *secondary keywords*.

1. Primary Keywords
 - Feature
 - Example (or Scenario)
 - Given, When, Then, And, But (steps)
 - Background
 - Scenario Outline (or Scenario Template)
 - Examples
2. Secondary Keywords
 - `"""` (Doc Strings)
 - `|` (Data Tables)
 - `@` (Tags)
 - `#` (Comments)

- a. **Given** provides context for the test scenario about to be executed, such as the point in your application that the test occurs as well as any prerequisite data.
- b. **When** specifies the set of actions that triggers the test, such as user or subsystem actions.
- c. **Then** specifies the expected result of the test.

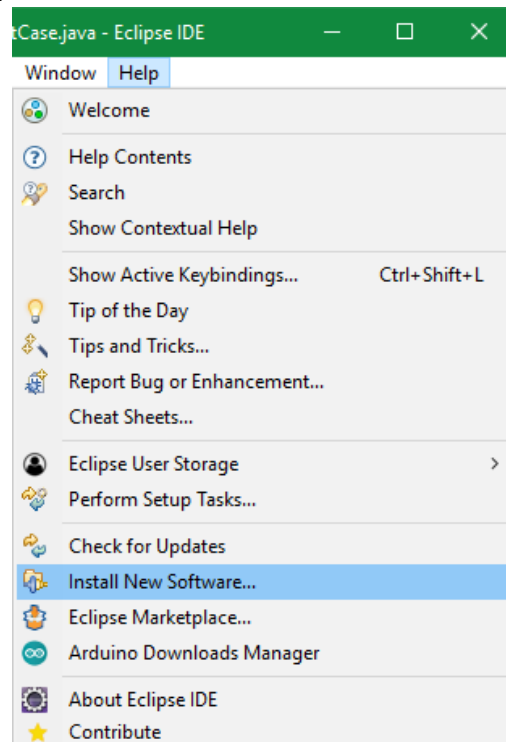
Listing 1. Initial login test

```
1 Feature: Test login
2
3 Scenario Outline: Login Success and Failure
4     Given I navigate to the mock application
5     When I try to login with valid credentials
6     Then I should see that I logged in successfully
```

III. Cucumber Installation

Petunjuk instalasi yang diberikan di sini difokuskan pada instalasi Cucumber pada Windows OS dengan menggunakan eclipse IDE. Langkah-langkahnya adalah sebagai berikut:

1. Install plugin Cucumber pada Eclipse.
Pada tab menu, klik **Help** → **Install New Software**

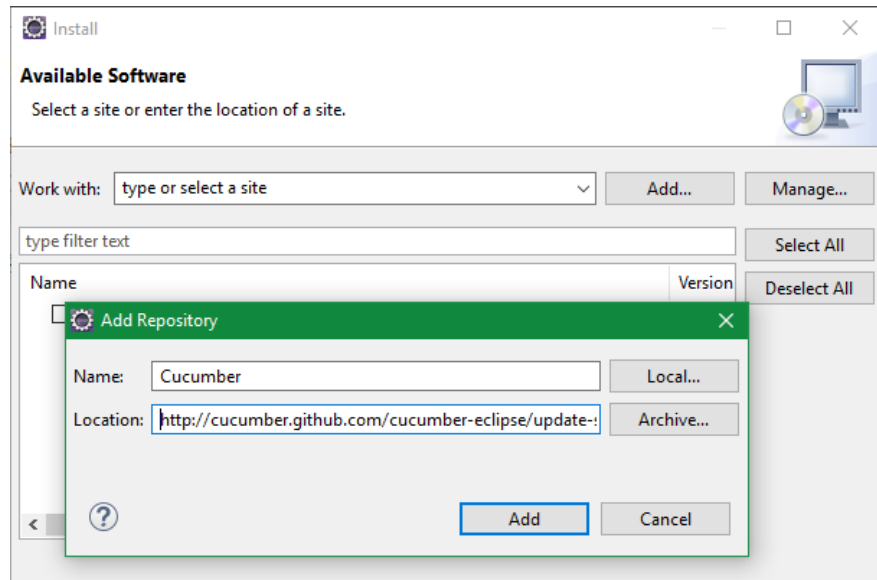


2. Klik **Add**.

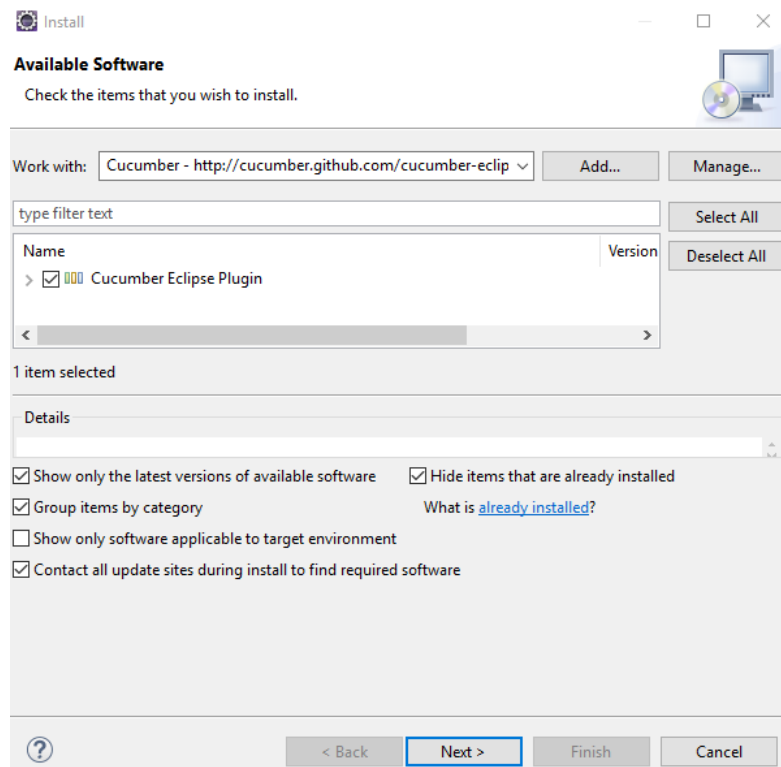
Field Name diisi dengan Cucumber.

Field Location diisi dengan <http://cucumber.github.com/cucumber-eclipse/update-site>

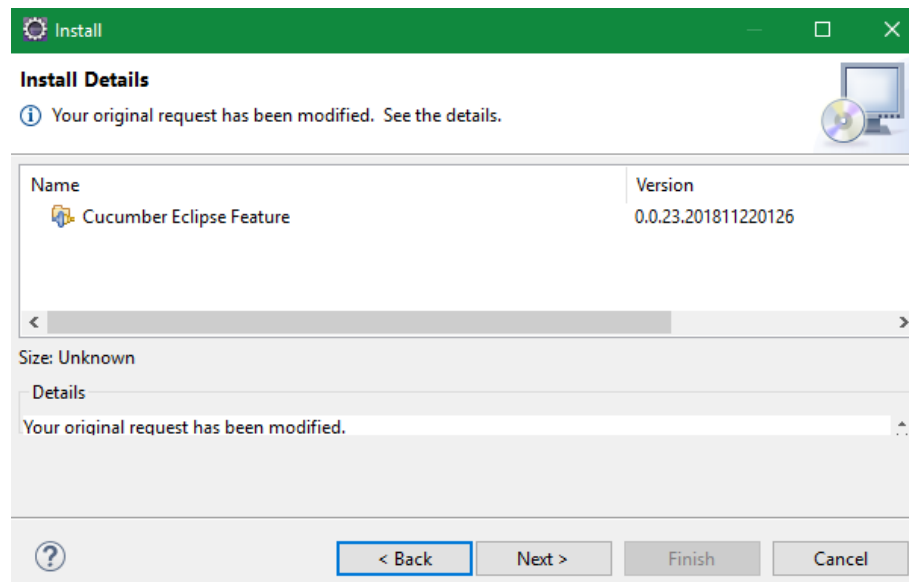
Lalu klik **Add**.



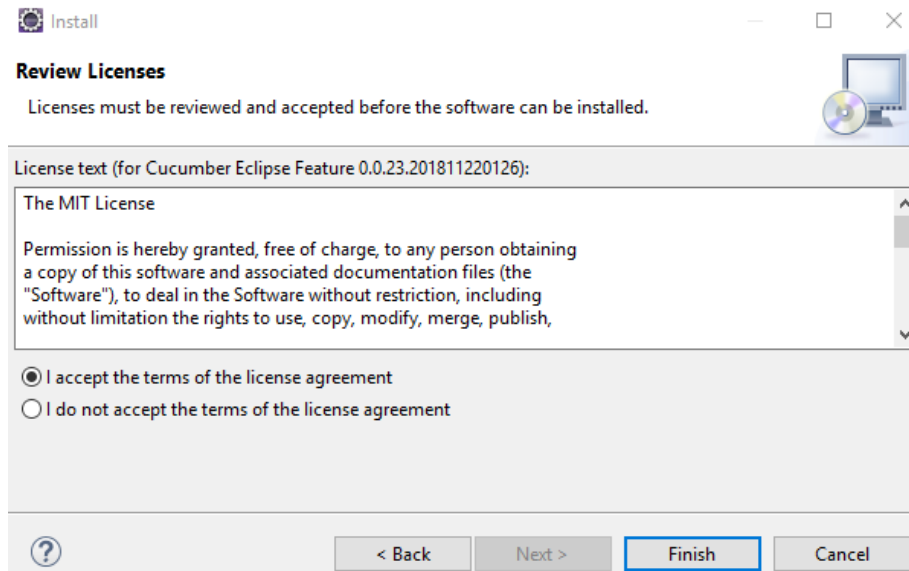
3. Akan ditampilkan seperti gambar di bawah ini. Centang **Cucumber Eclipse Plugin**. Lalu klik **Next**. Tunggu proses *download plugin*.



4. Akan ditampilkan seperti gambar di bawah ini. Klik Next.



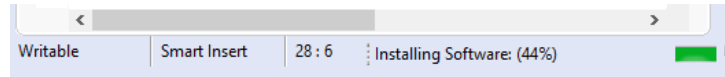
5. Akan ditampilkan seperti gambar di bawah ini. Pilih **"I accept the terms of license agreement"**.
Lalu klik **Finish**.



Jika muncul pop-up seperti gambar di bawah ini, klik **OK** atau **Install Anyway**.

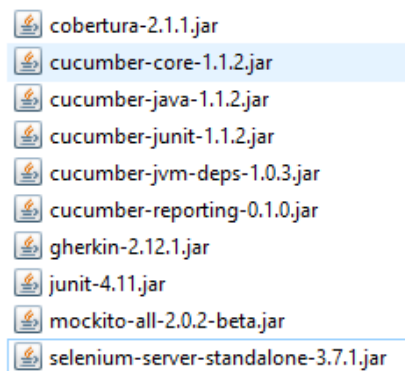


Eclipse akan mengeksekusi proses instalasi plugin. Tunggu sampai selesai.



6. Saat instalasi plugin selesai, Anda perlu **restart Eclipse**.

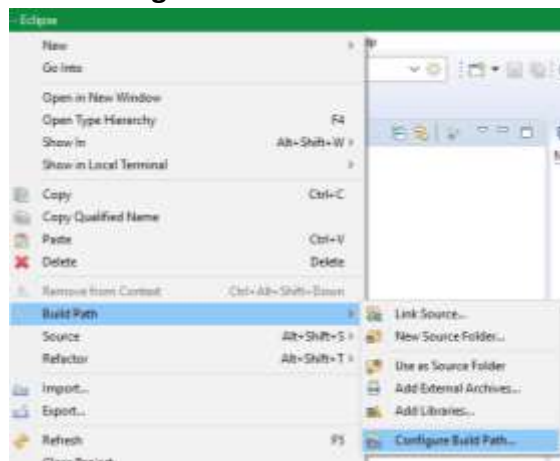
7. Selanjutnya, *Download* beberapa **jars** yang dibutuhkan untuk menggunakan Cucumber (<https://mvnrepository.com/>). Selain itu, **Selenium jars** juga dibutuhkan karena Cucumber akan digunakan bersamaan dengan Selenium.

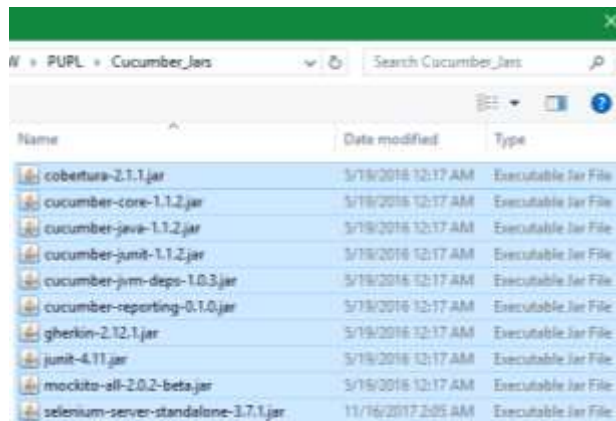
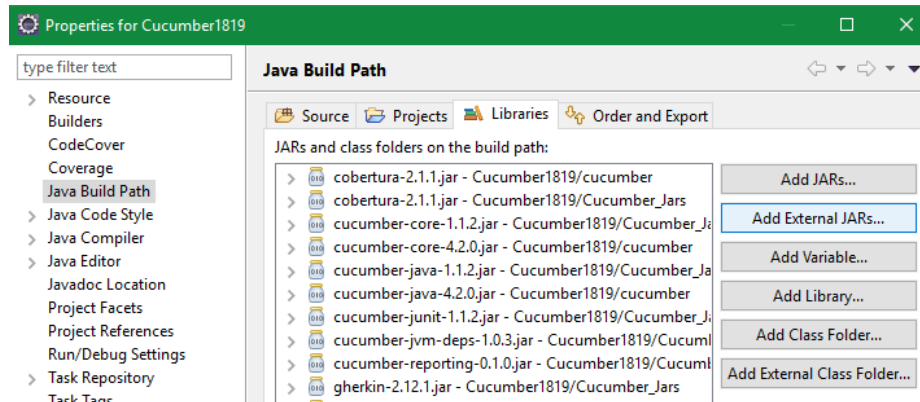


8. Buka Eclipse dan buat **New Java Project**.

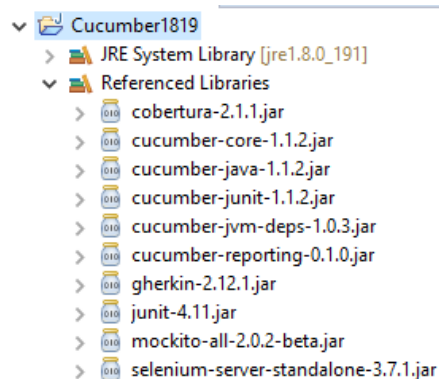
9. Tambahkan 10 External Jars yang disebutkan sebelumnya.

Klik Kanan → Build Path → Configure Build Path → Add External Jars →





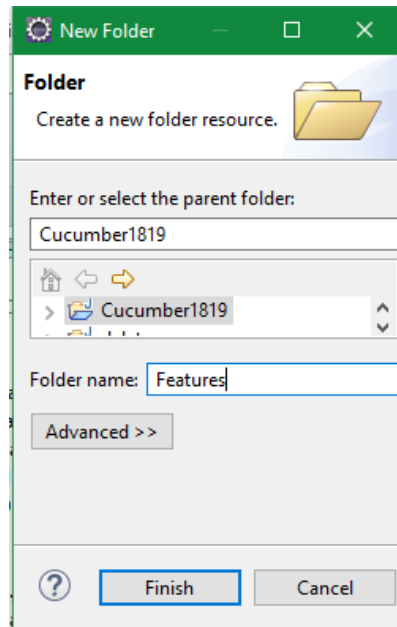
10. Anda dapat memastikan keberadaan Jars yang telah ditambahkan pada folder Referenced Library.



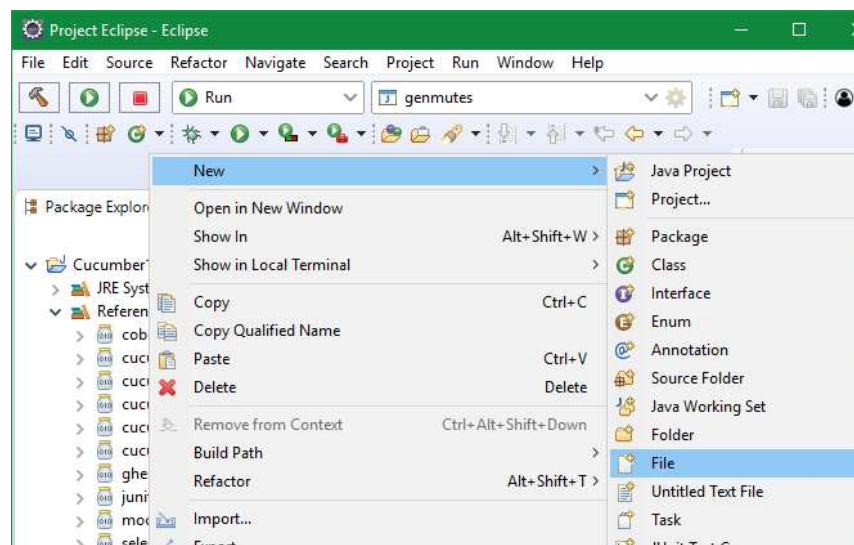
IV. Create Feature File Using Cucumber

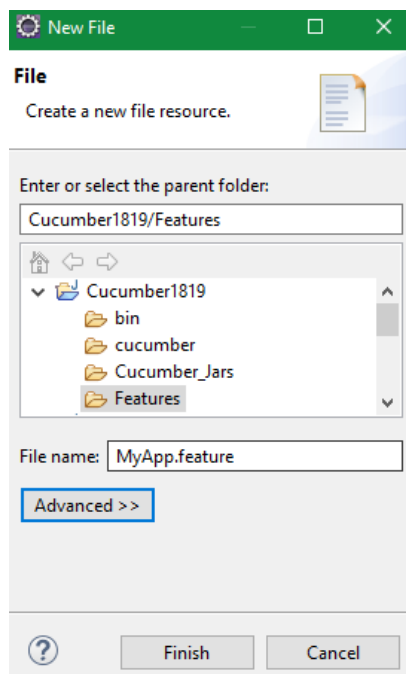
Feature File digunakan untuk menguji seluruh kebutuhan *user* dapat dipenuhi dengan menggunakan Gherkin Syntax. Satu Feature File dapat berisi beberapa *scenarios test*.

1. Tambahkan Folder baru pada proyek yang telah dibuat sebelumnya untuk menyimpan semua *feature files*.

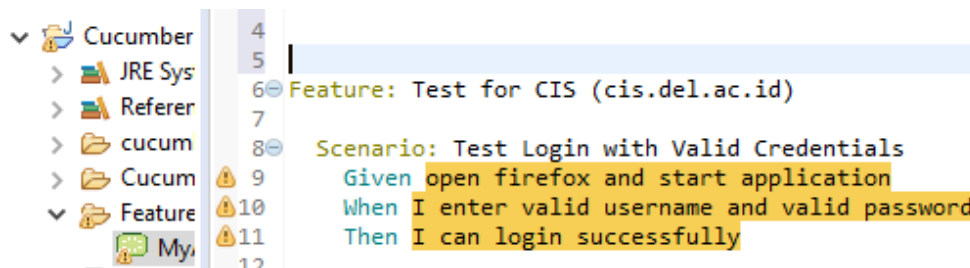


2. Pada folder yang dibuat sebelumnya, tambahkan sebuah file baru dengan ekstensi **.feature**.



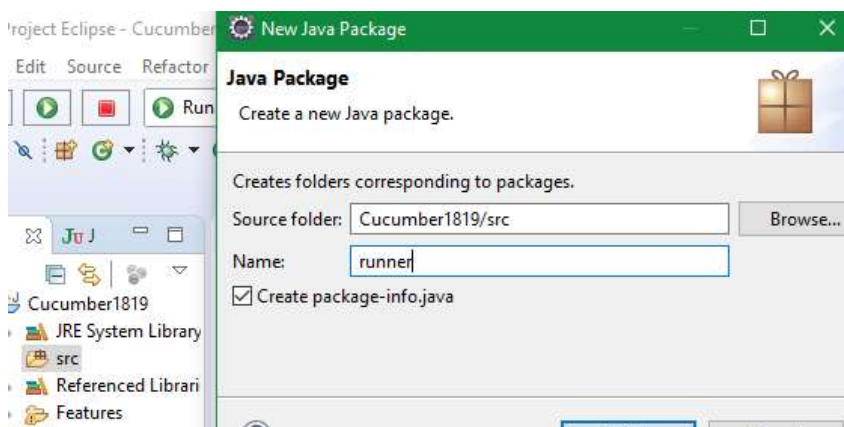


3. Pada feature tersebut, ketikkan requirement dengan menggunakan Gherkin Syntax, seperti *snippet* di bawah ini.

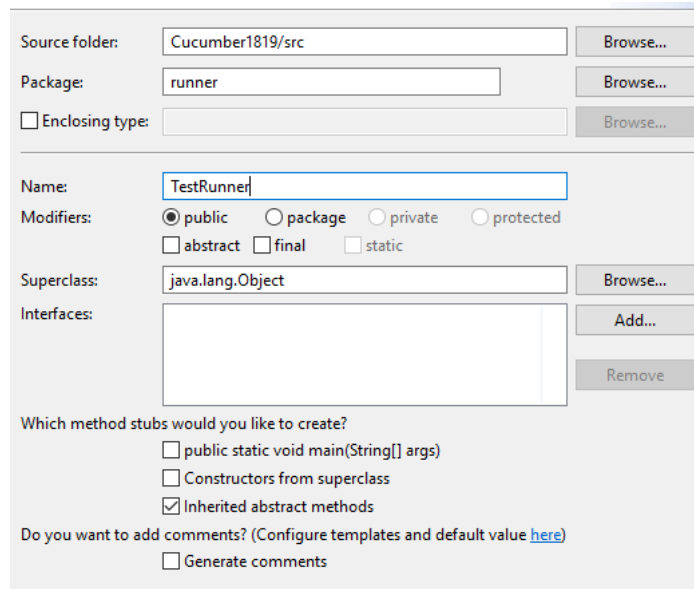


Snippet di atas menunjukkan Scenario yang akan digunakan untuk menguji fungsi Login pada aplikasi CIS IT Del.

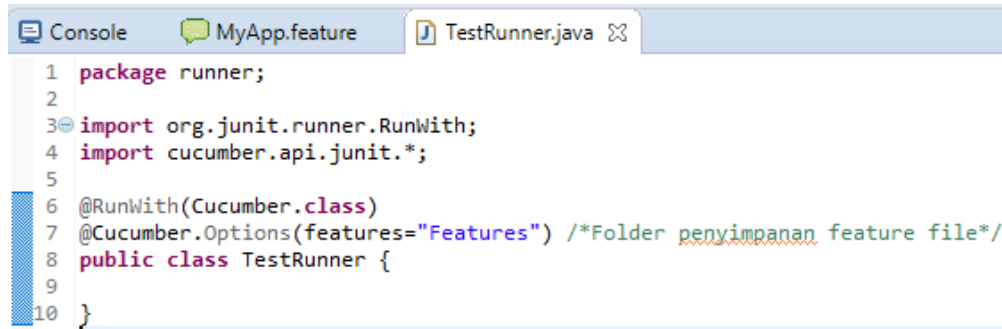
4. Untuk mengeksekusi feature file, Anda harus membuat *test runner (main method)*. Buat *package* baru pada Project dengan nama 'runner'.



5. Pada package runner, buat sebuah class Java baru dengan nama TestRunner.

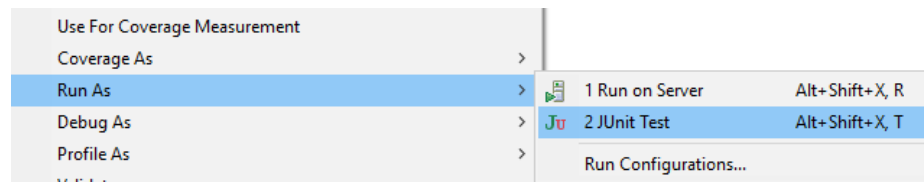


6. Pada class yang menjadi test runner, ketikkan code berikut.

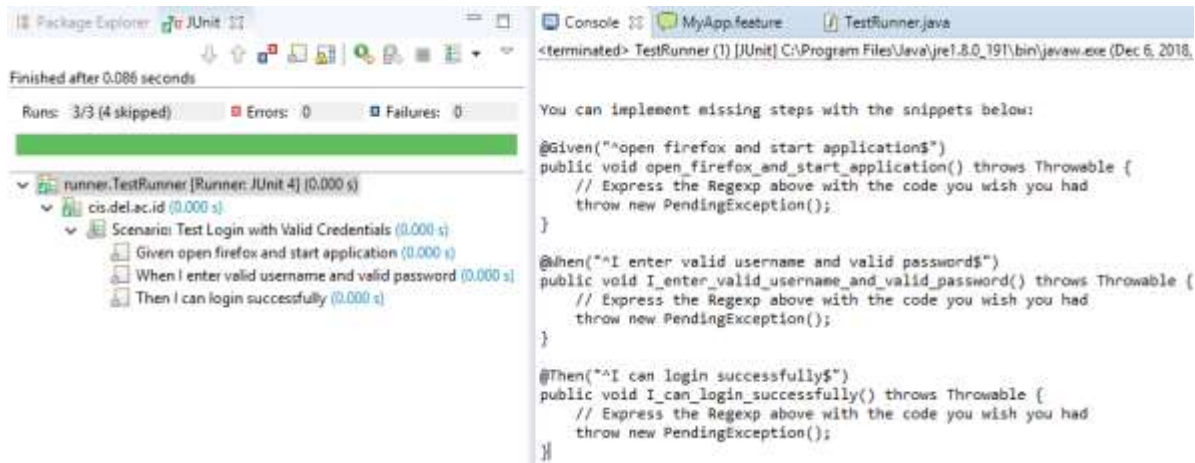


```
1 package runner;
2
3 import org.junit.runner.RunWith;
4 import cucumber.api.junit.*;
5
6 @RunWith(Cucumber.class)
7 @Cucumber.Options(features="Features") /*Folder penyimpanan feature file*/
8 public class TestRunner {
9
10 }
```

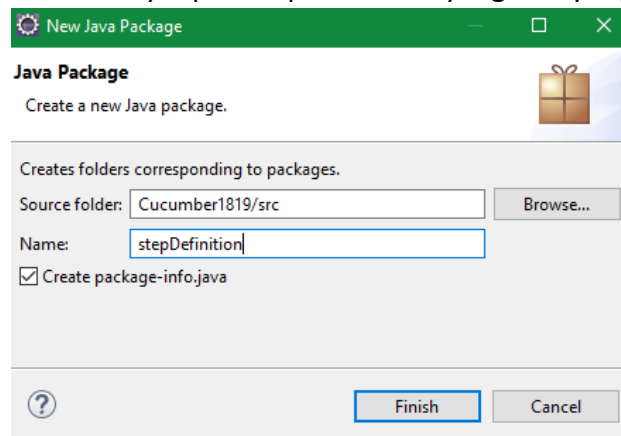
7. Run TestRunner.java dengan menggunakan Junit Test dan akan menampilkan hasil seperti *snippet* di bawah ini.



Hasil tersebut menunjukkan bahwa Scenario tidak dapat dieksekusi, karena feature file masih berisi plain text English, sehingga membutuhkan kode yang sesuai.



8. Buat package baru, untuk menyimpan step definition yang menyimpan *script test*.



9. Buat class baru pada package tersebut.



10. Copy perintah pada *console* saat me-run TestRunner pada class yang baru anda buat sebelumnya.

```
1 package stepDefinition;
2
3 public class SmokeTest {
4
5     @Given("^open firefox and start application$")
6     public void open_firefox_and_start_application() throws Throwable {
7         // Express the Regexp above with the code you wish you had
8         throw new PendingException();
9     }
10
11     @When("^I enter valid username and valid password$")
12     public void I_enter_valid_username_and_valid_password() throws Throwable {
13         // Express the Regexp above with the code you wish you had
14         throw new PendingException();
15     }
16
17     @Then("^I can login successfully$")
18     public void I_can_login_successfully() throws Throwable {
19         // Express the Regexp above with the code you wish you had
20         throw new PendingException();
21     }
22 }
```

11. Import semua *library* yang dibutuhkan dan hapus semua *dummy code*.

```
1 package stepDefinition;
2
3
4 import org.openqa.selenium.WebDriver;
5 import cucumber.api.java.en.Given;
6 import cucumber.api.java.en.Then;
7 import cucumber.api.java.en.When;
8
9 public class SmokeTest {
10     WebDriver driver;
11
12     @Given("^open firefox and start application$")
13     public void open_firefox_and_start_application() throws Throwable {
14
15     }
16
17     @When("^I enter valid username and valid password$")
18     public void I_enter_valid_username_and_valid_password() throws Throwable {
19
20     }
21
22     @Then("^I can login successfully$")
23     public void I_can_login_successfully() throws Throwable {
24
25     }
26 }
```

12. Tambahkan codes berikut ini.

```

1 package stepDefinition;
2
3
4+ import org.openqa.selenium.By;
11
12 public class SmokeTest {
13     WebDriver driver;
14
15     @Given("^open firefox and start application$")
16     public void open_firefox_and_start_application() throws Throwable {
17         System.setProperty("webdriver.gecko.driver", "C:\\Program Files\\geckodriver-v0.19.1-win64\\geckodriver.exe");
18
19         driver = new FirefoxDriver();
20         driver.manage().window().maximize();
21         driver.get("https://cis.del.ac.id/user/login");
22     }
23
24     @When("^I enter valid username and valid password$")
25     public void I_enter_valid_username_and_valid_password() throws Throwable {
26         driver.findElement(By.id("loginform-username")).sendKeys("budi.hariono");
27         driver.findElement(By.id("loginform-password")).sendKeys("budi.hariono");
28     }
29
30     @Then("^I can login successfully$")
31     public void I_can_login_successfully() throws Throwable {
32         driver.findElement(By.cssSelector(".btn")).click();
33     }
34 }
35

```

Perhatikan *scenario* yang telah dibuat sebelumnya. Scenario tersebut menjelaskan *step definition* yang diubah dalam bentuk Java pada class SmokeTest. Jika Anda menemukan error, maka Anda perlu melakukan *update* untuk **gecko.driver**.

13. Edit file Test Runner dengan menambahkan Step Definition pada file features.

```

1 package runner;
2
3 import org.junit.runner.RunWith;
4 import cucumber.api.junit.*;
5
6 @RunWith(Cucumber.class)
7 @Cucumber.Options(features="Features", glue="stepDefinition") /*Folder penyimpanan feature file*/
8 public class TestRunner {
9
10 }

```

14. Untuk menjalankan scenario, Run SmokeTest.java dengan Junit Test dan akan memunculkan hasil berikut.

```

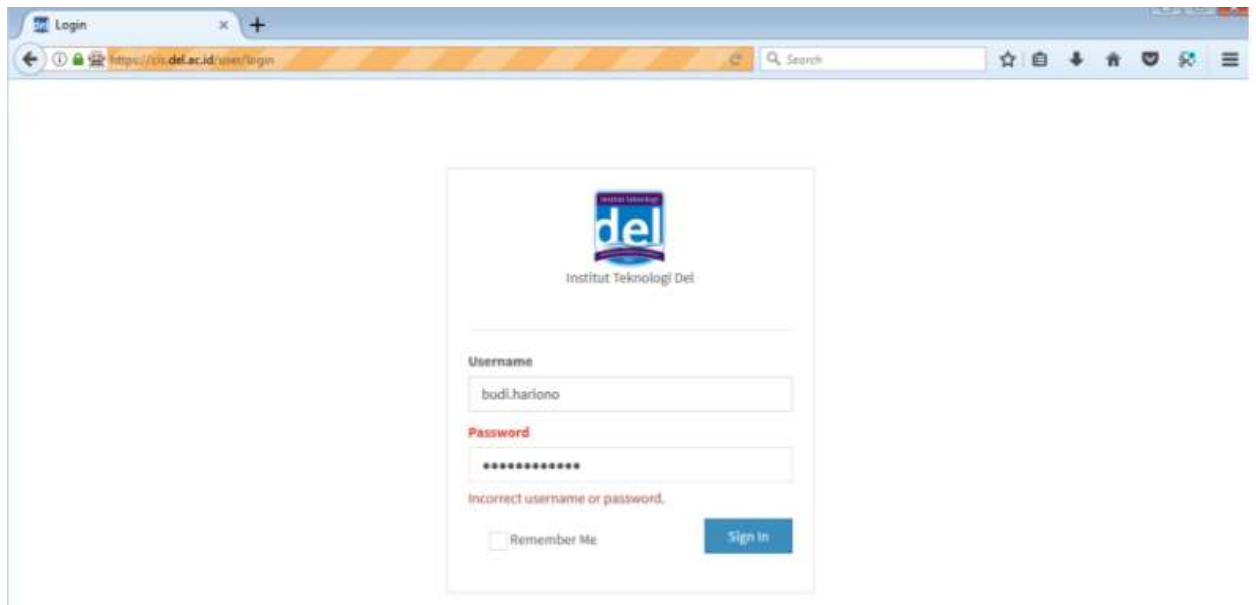
Finished after 13.168 seconds

Runs: 3/3      Errors: 0      Failures: 0

runner.TestRunner [Runner: JUnit 4] (0.957 s)
└─ Feature: Test Cis.del.ac.id smoke scenario (0.957 s)
   └─ Scenario: Test login with valid credentials (0.957 s)
      └─ Given open firefox and start application (0.112 s)
      └─ When I enter valid username and valid password (0.844 s)
      └─ Then user should be able login successfully (0.001 s)

```

15. Selenium akan membuka driver secara Otomatis.



V. References

Bowers, Alan and Bell, James. Automated Testing with Selenium and Cucumber. IBM: [online]. Available at: <https://www.ibm.com/developerworks/library/a-automating-ria/index.html> [Accessed in 3 December 2018].

-. Gherkin Reference. Available at: <https://docs.cucumber.io/gherkin/reference/> [Accessed in 3 December 2018].

VI. Task

1. Tuliskan scenario dan step definition dari **dua fungsi (minimal)** yang dapat Anda gunakan pada aplikasi yang ada di IT Del.

Note:

Jika Anda mencoba menguji aplikasi CIS (cis.del.ac.id), login tidak diperhitungkan.

2. Buat laporan yang berisi scenario dan step definition dari masing-masing fungsi yang diuji, serta screenshot pengujian yang dilakukan.

~EOF~