# Hooks concept-

A yellow rectangular sign with white text

Description automatically generated with medium confidence

Hooks are for technical stakeholders.

Hooks not part of feature file. We can write inside feature files but then other classes which use the common methods won’t be able to access it.

A computer screen with white text

Description automatically generated with low confidence

We can have multiple before and after annotations, but we need to give the order of execution.

Example, first browser launch, second db connection launch can be written in before hooks.

Example for after hooks – first close browser, second disconnect browser.

A screen shot of a computer

Description automatically generated with low confidence

# We can also use tag-hook combination-

A screenshot of a computer

Description automatically generated with medium confidence

In the above examples, only those scenarios with “tag = smoke” are executed in before tag.

# Hooks code with before and after-

Hooks:

|  |
| --- |
| **package** MyHooks;  **import** io.cucumber.java.After;  **import** io.cucumber.java.Before;  **public** **class** AmazonHooks {    @Before  **public** **void** setup() {  System.***out***.println("launching amazon application");  }  @After  **public** **void** tearDown() {  System.***out***.println("close the browser");  }  } |

Amazon search runner file:

|  |
| --- |
| **package** testRunners;  **import** io.cucumber.junit.Cucumber;  **import** io.cucumber.junit.CucumberOptions;  **import** org.junit.runner.RunWith;  @RunWith(Cucumber.**class**)  @CucumberOptions(plugin = { "pretty" }, features = { "src/test/resources/AppFeatures/Search.feature" },  // glue = {"/src/test/java/StepDefinitions/"} //this will also work  glue = { "StepDefinitions", "MyHooks" } // this way of defining also works  )  //in features we can give path till the feature file itself but tomorrow if n number of files are there and we want to run them  //then give the path till the package name  //glue tells where the step definitions are available.  //plugin = pretty means colorful and nice output.  **public** **class** AmazonSearchRunnerTest {  } |

Run runner:

Junit output:

A screenshot of a computer

Description automatically generated with medium confidence

Console output:

A screenshot of a computer code

Description automatically generated with low confidence

Likewise, if there are multiple scenarios, the before and after will run after each of the scenarios.

# Multiple before and after with order number-

Hooks:

|  |
| --- |
| **package** MyHooks;  **import** io.cucumber.java.After;  **import** io.cucumber.java.Before;  **public** **class** AmazonHooks {    @Before(order=1)  **public** **void** setUpBrowser() {  System.***out***.println("launching chrome brower");  }    @Before(order=2)  **public** **void** setUpURL() {  System.***out***.println("launching url");  }  @After(order=2)  **public** **void** tearDownClose() {  System.***out***.println("close the browser");  }    @After(order=1)  **public** **void** tearDownLogout() {  System.***out***.println("logout from the application");  }  } |

Run the runner file:

Junit:

A screenshot of a computer

Description automatically generated with medium confidence

Console:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer code

Description automatically generated with low confidence

# Using scenario as parameter-

Scenario comes from cucumber library and has lot of inbuilt methods.

Hooks code with scenario parameter:

|  |
| --- |
| **package** MyHooks;  **import** io.cucumber.java.After;  **import** io.cucumber.java.Before;  **import** io.cucumber.java.Scenario;  **public** **class** AmazonHooks {    @Before(order=1)  **public** **void** setUpBrowser(Scenario sc) {  System.***out***.println("launching chrome brower");  System.***out***.println(sc.getName());  }    @Before(order=2)  **public** **void** setUpURL() {  System.***out***.println("launching url");  }  @After(order=2)  **public** **void** tearDownClose(Scenario sc) {  System.***out***.println("close the browser");  System.***out***.println(sc.getName());  }    @After(order=1)  **public** **void** tearDownLogout() {  System.***out***.println("logout from the application");  }  } |

Run runner:

A picture containing text, screenshot, font, line

Description automatically generated

A picture containing text, font, screenshot, line

Description automatically generated

Its capturing the scenario name from feature file as seen below:

A screenshot of a computer

Description automatically generated with low confidence

A picture containing text, screenshot, font, line

Description automatically generated

# Hooks with before step and after step added-

|  |
| --- |
| **package** MyHooks;  **import** io.cucumber.java.After;  **import** io.cucumber.java.AfterStep;  **import** io.cucumber.java.Before;  **import** io.cucumber.java.BeforeStep;  **import** io.cucumber.java.Scenario;  **public** **class** AmazonHooks {  @Before(order = 1)  **public** **void** setUpBrowser(Scenario sc) {  System.***out***.println("launching chrome brower");  System.***out***.println(sc.getName());  }  @Before(order = 2)  **public** **void** setUpURL() {  System.***out***.println("launching url");  }  @After(order = 2)  **public** **void** tearDownClose(Scenario sc) {  System.***out***.println("close the browser");  System.***out***.println(sc.getName());  }  @After(order = 1)  **public** **void** tearDownLogout() {  System.***out***.println("logout from the application");  }  @BeforeStep  **public** **void** takeScreenshot() {  System.***out***.println("takes screenshot after every step");  }  @AfterStep  **public** **void** refreshBrowser() {  System.***out***.println("refresh the browser after every step");  }  } |

Run the runner:

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text, screenshot, font, document

Description automatically generated

# Even before steps and after steps can have order-

A picture containing text, screenshot, font

Description automatically generated

# Hooks with @-

|  |
| --- |
| **package** MyHooks;  **import** io.cucumber.java.After;  **import** io.cucumber.java.AfterStep;  **import** io.cucumber.java.Before;  **import** io.cucumber.java.BeforeStep;  **import** io.cucumber.java.Scenario;  **public** **class** AmazonHooks {  @Before("@Smoke")  **public** **void** setUpBrowser(Scenario sc) {  System.***out***.println("launching chrome brower");  System.***out***.println(sc.getName());  }  // @Before(order = 2)  // public void setUpURL() {  // System.out.println("launching url");  //// }  @After("@Regression")  **public** **void** tearDownClose(Scenario sc) {  System.***out***.println("close the browser");  System.***out***.println(sc.getName());  }  // @After(order = 1)  // public void tearDownLogout() {  // System.out.println("logout from the application");  // }  // @BeforeStep  // public void takeScreenshot() {  // System.out.println("takes screenshot after every step");  // }  //  // @AfterStep  // public void refreshBrowser() {  // System.out.println("refresh the browser after every step");  // }  } |

Feature file:

|  |
| --- |
| #Author: your.email@your.domain.com  #Keywords Summary :  #Feature: List of scenarios.  #Scenario: Business rule through list of steps with arguments.  #Given: Some precondition step  #When: Some key actions  #Then: To observe outcomes or validation  #And,But: To enumerate more Given,When,Then steps  #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>  #Examples: Container for s table  #Background: List of steps run before each of the scenarios  #""" (Doc Strings)  #| (Data Tables)  #@ (Tags/Labels):To group Scenarios  #<> (placeholder)  #""  ## (Comments)  #Sample Feature Definition Template  Feature: Amazon Search  #this is the feature for which we want to write code and requirements  #one feature file can have multiple scenarios  @Smoke  Scenario: Search for a product apple macbook air  #here we give the scenario name  #below scenario (given, when, then, and) etc are known as steps  Given I have a search field on Amazon page  #given can be considered as pre-condition  When I search for product with name "apple" and price is 1000  #string is in double quotes  Then Product with name "apple" should be displayed  Then order id is 12345 and username is "naveen"  @Regression  Scenario: Search for a product iphone  Given I have a search field on Amazon page  When I search for product with name "iphone" and price is 2000  Then Product with name "iphone" should be displayed  Then order id is 5677 and username is "naveen automation" |

Run runner:

A screenshot of a computer code

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with low confidence

If we see the output above, for smoke, the before hook was executed and not after hook, and for regression the after hook was executed and not before book, as the @ have been defined in such a way for before and after for smoke and regression etc.

# Project structure-

A screenshot of a computer

Description automatically generated with medium confidence

# Codes used for this lesson-

Hooks:

|  |
| --- |
| **package** MyHooks;  **import** io.cucumber.java.After;  **import** io.cucumber.java.AfterStep;  **import** io.cucumber.java.Before;  **import** io.cucumber.java.BeforeStep;  **import** io.cucumber.java.Scenario;  **public** **class** AmazonHooks {  @Before("@Smoke")  **public** **void** setUpBrowser(Scenario sc) {  System.***out***.println("launching chrome brower");  System.***out***.println(sc.getName());  }  // @Before(order = 2)  // public void setUpURL() {  // System.out.println("launching url");  //// }  @After("@Regression")  **public** **void** tearDownClose(Scenario sc) {  System.***out***.println("close the browser");  System.***out***.println(sc.getName());  }  // @After(order = 1)  // public void tearDownLogout() {  // System.out.println("logout from the application");  // }  // @BeforeStep  // public void takeScreenshot() {  // System.out.println("takes screenshot after every step");  // }  //  // @AfterStep  // public void refreshBrowser() {  // System.out.println("refresh the browser after every step");  // }  } |

Runner:

|  |
| --- |
| **package** testRunners;  **import** io.cucumber.junit.Cucumber;  **import** io.cucumber.junit.CucumberOptions;  **import** org.junit.runner.RunWith;  @RunWith(Cucumber.**class**)  @CucumberOptions(plugin = { "pretty" }, features = { "src/test/resources/AppFeatures/Search.feature" },  // glue = {"/src/test/java/StepDefinitions/"} //this will also work  glue = { "StepDefinitions", "MyHooks" }, // this way of defining also works  tags = "@Smoke or @Regression")  //in features we can give path till the feature file itself but tomorrow if n number of files are there and we want to run them  //then give the path till the package name  //glue tells where the step definitions are available.  //plugin = pretty means colorful and nice output.  **public** **class** AmazonSearchRunnerTest {  } |

Feature file:

|  |
| --- |
| #Author: your.email@your.domain.com  #Keywords Summary :  #Feature: List of scenarios.  #Scenario: Business rule through list of steps with arguments.  #Given: Some precondition step  #When: Some key actions  #Then: To observe outcomes or validation  #And,But: To enumerate more Given,When,Then steps  #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>  #Examples: Container for s table  #Background: List of steps run before each of the scenarios  #""" (Doc Strings)  #| (Data Tables)  #@ (Tags/Labels):To group Scenarios  #<> (placeholder)  #""  ## (Comments)  #Sample Feature Definition Template  Feature: Amazon Search  #this is the feature for which we want to write code and requirements  #one feature file can have multiple scenarios  @Smoke  Scenario: Search for a product apple macbook air  #here we give the scenario name  #below scenario (given, when, then, and) etc are known as steps  Given I have a search field on Amazon page  #given can be considered as pre-condition  When I search for product with name "apple" and price is 1000  #string is in double quotes  Then Product with name "apple" should be displayed  Then order id is 12345 and username is "naveen"  @Regression  Scenario: Search for a product iphone  Given I have a search field on Amazon page  When I search for product with name "iphone" and price is 2000  Then Product with name "iphone" should be displayed  Then order id is 5677 and username is "naveen automation" |

Step def:

|  |
| --- |
| package StepDefinitions;  import io.cucumber.java.en.Given;  import io.cucumber.java.en.Then;  import io.cucumber.java.en.When;  import junit.framework.Assert;  import amazonImplementation.Product;  import amazonImplementation.Search;  public class SearchFeatureStepDef {  Product product;  Search search;  @Given("I have a search field on Amazon page")  public void i\_have\_a\_search\_field\_on\_amazon\_page() {  System.out.println("step 1 - i am on search page");  }  @When("^I search for product with name \"([^\"]+)\" and price is (\\d+)$")  public void i\_search\_for\_product\_with\_name\_and\_price\_is(String productName, Integer price) {  System.out.println("step 2 - search product with name " + productName + " and price is " + price);  product = new Product(productName, price);  }  @Then("Product with name {string} should be displayed")  public void product\_with\_name\_should\_be\_displayed(String productName) {  System.out.println("step 3 - product with " + productName + " : is displayed");  search = new Search();  String productNameReturned = search.displayProductName(product);  System.out.println("returned product is " + productNameReturned);  Assert.assertEquals(product.getProductName(), productNameReturned);  }  @Then("order id is {int} and username is {string}")  public void order\_id\_is\_and\_username\_is(Integer orderId, String userName) {  // Write code here that turns the phrase above into concrete actions  System.out.println("order id is " + orderId + " " + "and username is " + userName);  }  } |

Search java:

|  |
| --- |
| **package** amazonImplementation;  **public** **class** Search {  **public** String displayProductName(Product product) { // here we will take the product name returned from  // product.java class  **if** (product.getProductList().contains(product.getProductName())) {  **return** product.getProductName();  } **else** {  **return** **null**;  }  // or we can simply write, because if above return is satisfied that will be the  // thing which will be returned to method  // return null;  }  } |

Product java:

|  |
| --- |
| package amazonImplementation;  import java.util.ArrayList;  import java.util.List;  public class Product {  private String productName;  private int price;  public Product(String productName, int price) {  this.productName = productName;  this.price = price;  }  public String getProductName() {  return productName;  }  public void setProductName(String productName) {  this.productName = productName;  }  public int getPrice() {  return price;  }  public void setPrice(int price) {  this.price = price;  }  // this method will list of products in the form of string  public List<String> getProductList() {  List<String> prodList = new ArrayList<>();  prodList.add("apple");  prodList.add("hp");  prodList.add("samsung");  prodList.add("bourbon");  prodList.add("iphone");  return prodList;  }  } |