Practice Tasks + Solutions for Cucumber + Selenium + JUnit

---

1. Installations & Configurations (Maven Project)

Task

- Create a **Maven Cucumber Project**
- Add dependencies (`selenium-java`, `cucumber-java`, `cucumber-junit`)
- Setup **JUnit runner class**

Solution

1. Run:
2. `mvn archetype:generate -DgroupId=com.practice -DartifactId=CucumberPractice \`
3. `-DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false`
4. Add dependencies in `pom.xml` (same as earlier).
5. Create `RunCucumberTest.java` with `@RunWith(Cucumber.class)`.

---

2. Simple Scenario

Task

Automate a **Google Search**:

- Open Google
- Search for “Cucumber BDD”
- Verify results page contains "Cucumber".

Feature File – `google.feature`

```
Feature: Google Search

  Scenario: Search for Cucumber BDD
    Given User is on Google Home Page
    When User searches for "Cucumber BDD"
    Then Results should contain "Cucumber"
```

Step Definition – `GoogleSteps.java`

```
WebDriver driver;

@Given("User is on Google Home Page")
public void user_is_on_google_home_page() {
    driver = new ChromeDriver();
    driver.get("https://www.google.com");
}

@When("User searches for {string}")
public void user_searches_for(String query) {
    driver.findElement(By.name("q")).sendKeys(query);
    driver.findElement(By.name("q")).submit();
}

@Then("Results should contain {string}")
public void results_should_contain(String keyword) {
    assertTrue(driver.getPageSource().contains(keyword));
    driver.quit();
}
```

---

3. Scenario Outline

Task

Test **multiple login attempts**:

- Valid user → success
- Invalid user → failure

Feature File – `login.feature`

```
Feature: Login Test

  Scenario Outline: Multiple login checks
    Given User is on Login Page
    When User enters username "<username>" and password "<password>"
    Then Login should be "<status>"

  Examples:
    | username | password   | status  |
    | admin    | admin123   | success |
    | wrong    | test123    | failure |
```

Step Definition

```
@Then("Login should be {string}")
public void login_should_be(String status) {
    if (status.equals("success")) {
        assertTrue(driver.getTitle().contains("Home"));
    } else {
        assertTrue(driver.getPageSource().contains("Invalid"));
    }
    driver.quit();
}
```

---

4. Data-Driven with Data Tables

Task

Login with multiple users using **DataTable**.

Feature File – `datatable.feature`

```
Feature: Login with multiple users

  Scenario: Enter multiple credentials
    Given User is on Login Page
    When User logs in with
      | username | password |
      | admin    | admin123 |
      | user1    | pass123  |
    Then Verify login results
```

Step Definition

```
@When("User logs in with")
public void user_logs_in_with(DataTable dataTable) {
    List<Map<String, String>> users = dataTable.asMaps();
    for (Map<String, String> user : users) {
        driver.findElement(By.id("username")).clear();
        driver.findElement(By.id("username")).sendKeys(user.get("username"));
        driver.findElement(By.id("password")).clear();
        driver.findElement(By.id("password")).sendKeys(user.get("password"));
        driver.findElement(By.id("loginBtn")).click();
    }
}
```

---

5. Data-Driven with Lists

Task

Search for multiple products on an e-commerce site.

Feature File – `products.feature`

```
Feature: Product Search

  Scenario: Search multiple products
    Given User is on Search Page
    When User searches for
      | Laptop   |
      | Mobile   |
      | Headset  |
    Then Products should be displayed
```

Step Definition

```
@When("User searches for")
public void user_searches_for(List<String> items) {
    for (String item : items) {
        driver.findElement(By.id("searchBox")).clear();
        driver.findElement(By.id("searchBox")).sendKeys(item);
        driver.findElement(By.id("searchBtn")).click();
        System.out.println("Searched: " + item);
    }
}
```

---

6. Dependency Injection (PicoContainer)

Task

Use **Shared WebDriver** across multiple step files.

SharedDriver.java

```
public class SharedDriver {
    private WebDriver driver;

    public SharedDriver() {
        driver = new ChromeDriver();
    }

    public WebDriver getDriver() {
        return driver;
    }
}
```

Step File

```
public class SearchSteps {
    private SharedDriver shared;

    public SearchSteps(SharedDriver shared) {
        this.shared = shared;
    }

    @Given("User is on Search Page")
    public void user_is_on_search_page() {
        shared.getDriver().get("https://example.com/search");
    }
}
```

---

7. Parallel Execution

Task

Run **Login** and **Search features in parallel**.

Configure `pom.xml`

```
<configuration>
    <parallel>classes</parallel>
    <threadCount>2</threadCount>
</configuration>
```

Two Runners

```
@RunWith(Cucumber.class)
@CucumberOptions(features="src/test/resources/features/login.feature",
                 glue="stepdefinitions")
public class RunLoginTest {}

@RunWith(Cucumber.class)
@CucumberOptions(features="src/test/resources/features/products.feature",
                 glue="stepdefinitions")
public class RunProductTest {}
```

Now both will run in **parallel**.

---

8. Feature Level Parallelism

Task

Run **each feature file separately in parallel** using multiple runners (as above).
Already covered by using **Surefire parallel=classes**.

---

9. Reporting – HTML + Extent

Task

Generate **HTML + JSON + Extent Reports**.

Runner with Reports

```
@RunWith(Cucumber.class)
@CucumberOptions(
  features = "src/test/resources/features",
  glue = "stepdefinitions",
  plugin = {
    "pretty",
    "html:target/cucumber-html-report",
    "json:target/cucumber.json",
    "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"
  },
  monochrome = true
)
public class RunCucumberTest {}
```

Output

- `target/cucumber-html-report/index.html`
- `target/cucumber.json`
- `Extent Report` under `/test-output/`