Cucumber - runner class

To **create a runner class using `@CucumberOptions`** in a Cucumber + TestNG or JUnit framework (commonly for BDD-style testing), you typically follow these steps:

✅ Purpose of `@CucumberOptions`:

The **`@CucumberOptions`** annotation is used to configure the Cucumber test execution, such as:

- Path to feature files
- Path to step definitions
- Tags to filter scenarios
- Plugins for reporting
- Other execution options

✅ Example Runner Class Using `@CucumberOptions` (with JUnit)

```
java


import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",         // path to your .feature files
    glue = "stepDefinitions",                          // package containing step definition
classes
    plugin = {"pretty", "html:target/cucumber-reports.html"}, // report formats
    tags = "@SmokeTest",                               // run scenarios with this tag
    monochrome = true,                                 // readable console output
    dryRun = false                                     // true = checks mapping without running
steps
)
public class TestRunner {
    // No methods needed. Acts as a test trigger.
}
```

✅ Fields in `@CucumberOptions`

Option	Description
`features`	Path to **`.feature`** files. Relative to project root.
`glue`	Package(s) containing step definitions and hooks.
`plugin`	Output formats like **`pretty`**, **`html`**, **`json`**, etc.
`tags`	Runs only scenarios with matching tag(s).
`monochrome`	Cleaner console output (true = no color).
`dryRun`	If **`true`**, checks if all steps are defined without actually running. Useful during development.

✅ Example Feature File: `Login.feature`

```gherkin
Feature: Login Feature

  @SmokeTest
  Scenario: Valid login
    Given User is on login page
    When User enters valid credentials
    Then User is navigated to homepage
```

✅ Step Definition Class: `LoginSteps.java`

```java
package stepDefinitions;

import io.cucumber.java.en.*;

public class LoginSteps {

    @Given("User is on login page")
    public void user_is_on_login_page() {
        System.out.println("User navigates to login page");
    }

    @When("User enters valid credentials")
    public void user_enters_valid_credentials() {
        System.out.println("User enters username and password");
    }

    @Then("User is navigated to homepage")
    public void user_is_navigated_to_homepage() {
        System.out.println("User lands on homepage");
    }
}
```

✅ Running It

- Right-click the `TestRunner` class → **Run As > JUnit Test**
- It will pick all `@SmokeTest` scenarios from your features and run them with matching steps.

Would you like the same setup with **TestNG** or **custom tags/multiple plugins**?

To **run feature files using the Runner class**, follow this step-by-step process. We'll use the **JUnit** setup with **Cucumber** as an example:

✅ Step-by-Step: Running Feature Files Using the Runner Class

🔹 1. Create a Feature File

Location: `src/test/resources/features/Login.feature`

```gherkin

Feature: Login Feature

  @SmokeTest
  Scenario: Successful login with valid credentials
    Given User is on login page
    When User enters valid username and password
    Then User is navigated to homepage
```

🔹 2. Create Step Definitions

Location: `src/test/java/stepDefinitions/LoginSteps.java`

```java
package stepDefinitions;

import io.cucumber.java.en.*;

public class LoginSteps {

    @Given("User is on login page")
    public void user_is_on_login_page() {
        System.out.println("Navigated to login page");
    }

    @When("User enters valid username and password")
    public void user_enters_valid_credentials() {
        System.out.println("Entered valid credentials");
    }

    @Then("User is navigated to homepage")
    public void user_is_navigated_to_homepage() {
        System.out.println("Landed on homepage");
    }
}
```

🔹 3. Create Runner Class Using `@CucumberOptions`

Location: `src/test/java/runner/TestRunner.java`

```java
package runner;

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
```

```java
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",       // Location of feature files
    glue = {"stepDefinitions"},                     // Location of step definitions
    tags = "@SmokeTest",                            // Tag to filter scenarios
    plugin = {
        "pretty",                                   // Console output
        "html:target/cucumber-reports.html",        // HTML report
        "json:target/cucumber.json"                 // JSON report
    },
    monochrome = true,
    dryRun = false
)
public class TestRunner {
    // No code needed; Cucumber will pick it up using annotations.
}
```

🔹 4. Run the Test

- **Right-click `TestRunner.java` → Run As → JUnit Test**
- This will:
  - Locate the feature files
  - Match steps in **`glue`** package
  - Execute only the scenarios tagged with **`@SmokeTest`**
  - Generate reports in **`/target`**

✅ Output in Console

You'll see:

```pgsql
Navigated to login page
Entered valid credentials
Landed on homepage
BUILD SUCCESSFUL
```

📝 Notes

- You can use **`@RegressionTest`**, **`@Sanity`**, or combine tags like **`@SmokeTest and @RegressionTest`**.
- Reports can also be generated using **`extent`**, **`allure`**, or other plugins.

Would you like:

- To run **multiple feature files**?
- To use **TestNG instead of JUnit**?
- Or generate **Extent/Allure reports**?

Let me know your use case.

Great! Since you said **"yes"**, I'll show you how to run **multiple feature files** using the **Runner class**, with examples of:

1. ✅ Running multiple `.feature` files
2. ✅ Using multiple tags
3. ✅ Optional: Running with **TestNG** (if you prefer TestNG instead of JUnit)
4. ✅ Generating **Extent** or **Allure** reports (optional)

✅ 1. Running Multiple Feature Files (with JUnit)

Use a common folder for all `.feature` files, e.g.,:

```swift
src/test/resources/features/
├── Login.feature
├── Register.feature
└── Dashboard.feature
```

Sample `Login.feature`

```gherkin
@SmokeTest
Feature: Login Feature

  Scenario: Valid login
    Given User is on login page
    When User enters valid credentials
    Then User is navigated to homepage
```

Sample `Register.feature`

```gherkin
@RegressionTest
Feature: Registration Feature

  Scenario: Register with valid data
    Given User is on registration page
    When User provides all valid info
    Then User gets success message
```

✅ Runner Class

```java
package runner;

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
```

```
@CucumberOptions(
    features = "src/test/resources/features",       // Includes ALL .feature files in the folder
    glue = {"stepDefinitions"},                     // Step definitions package
    plugin = {
        "pretty",
        "html:target/html-report.html",
        "json:target/cucumber.json"
    },
    tags = "@SmokeTest or @RegressionTest",         // Runs both tagged scenarios
    monochrome = true,
    dryRun = false
)
public class TestRunner {
}
```

✅ 2. Use Multiple Tags with Logic

You can customize which scenarios to run using:

Tag Expression	Description
`@SmokeTest`	Only smoke test scenarios
`@SmokeTest and @RegressionTest`	Scenarios with both tags
`@SmokeTest or @RegressionTest`	Scenarios with either tag
`not @WIP`	Exclude scenarios marked as WIP

✅ 3. Optional: Use TestNG Instead of JUnit

Use the Cucumber TestNG runner if you prefer TestNG:

Add Dependency in `pom.xml` (Maven)

```
xml

<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-testng</artifactId>
    <version>7.14.0</version>
    <scope>test</scope>
</dependency>
```

Create TestNG Runner Class

```
java

package runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;

@CucumberOptions(
    features = "src/test/resources/features",
    glue = "stepDefinitions",
    plugin = {
        "pretty",
        "html:target/testng-cucumber-report.html"
    },
    tags = "@SmokeTest",
```

```
    monochrome = true
)
public class TestNGRunner extends AbstractTestNGCucumberTests {
}
```

To run: Right-click on **`TestNGRunner`** → Run As → TestNG Test

✅ 4. Optional: Generate Extent or Allure Reports

You can integrate:

For Extent Reports:

- Use **`com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:`** plugin
- Add to **`@CucumberOptions`**:

```
java

plugin = {
  "pretty",
  "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"
}
```

Make sure to add required dependencies and config file (**`extent-config.xml`**).

Would you like me to help you **set up Extent Reports** or **convert your runner to TestNG**?