Cucumber

Context

Cucumber is a tool that supports ***BDD***. The tests are written in plain (human) language (***ubiquitous language***) so that non-technical stakeholders can understand them. Cucumber combines requirements specifications, automated tests and living documentation. The syntax is called Gherkin and is available in a lot of languages, including ***english***. Cucumber is available in many programming languages, including the most used : ***Java***, ***C#***, ***JS***, ***Python*** and ***Ruby***.

Description

Gherkin

Gherkin uses a set of special **keywords** to give **structure** and meaning to executable specifications. That structure is used to define **human-readable** and **machine-interpretable** scenarios.

Here are the most used keywords :

- **Feature** : Name and describre the feature you are testing
- **Scenario** : Name and describre the scenario you are testing
- **Given** / **When** / **Then** / **And** : Those keywords are the actual steps of each scenario. They are the ones where there is an implementation code behind it. There is no difference on the way they work, but they have to be used in a logical way to make sense when the scenario is read

Example

```
Feature: Acknowledgments status

  Scenario: View a acknowledgment status from one of your sent messages

    Given user is logged in

    When user is consulting the SENTBOX

    And user reads the basic message

    And user requests the acknowledgments status

    Then the acknowledgments are shown correctly
```

Step definition

The step definitions is where the **implementation** is written. Each method is preceded with an **annotation** referring to the keyword of the step and a **regular expression** to match the step.

Example

```
@When("^user is consulting the (.*?)$")
public void user_is_consulting_the_box(String box) {
    //Write implementation here
}
```

Test runner

The test runner is a class making the link between the feature files and the implementation code. There is a possibility to configure it with Cucumber annotations and parameters.

Example

```
@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/java/be/imec/hie/features/eHealthBox",
        glue = {"be.imec.hie.steps"},
        plugin = {"json:target/cucumber-report/report.json",
                "pretty",
                "io.qameta.allure.cucumber4jvm.AllureCucumber4Jvm"},
        strict = true,
        tags = {"not @ignore"}
)

public class EHealthBoxTestRunner {
    // Additional configuration code
}
```

Integration within the test automation framework

Cucumber has been integrated to the framework using **maven dependencies** and used for the **SOAP** integration testing.

Within the project, you can find the code using **Cucumber** in the **SOAP** project, under :

- *src/test/java/be/imec/hie/features* for the feature files
- *src/test/java/be/imec/hie/steps* for the steps definition
- *src/test/java/be/imec/hie/runners* for the feature runners

Further information

Cucumber has their own documentation following this link : https://cucumber.io/docs