Behavior-Driven Development (BDD)

Formalizing user stories

```
Scenario: Guess a word
    Given that the Wordmaker starts the game
     When the Wordmaker has started the game
     Then the Wordmaker waits for the player to join
```

Formalizing user stories: Steps

```
Given <context>
When <event>
Then <expected outcome>
```

Given user is not logged in...

Given player has at least 1 life left...

Formalizing user stories: Steps

```
Given <context>
When <event>
Then <expected outcome>
```

When user edits profile...

When player steps on bomb...

Formalizing user stories: Steps

```
Given <context>

When <event>

Then <expected outcome>
```

Then user gets an error message

Then number of lives decreases by 1

Formalizing user stories: Scenarios

Scenario: All done

Given I am out shopping

Given I have eggs

Given I have milk

Given I have butter

When I check my list

Then I smile because I don't need anything

Formalizing user stories: Features

```
Feature: User Login
  Scenario: Successful login
    Given user navigates to polyfood.ch website
    When user logs in using Username "USER" and Password "goodPASSWORD"
    Then login should be successful
  Scenario: Failed login
    Given user navigates to polyfood.ch website
    When user logs in using Username "USER" and Password "badPASSWORD"
    Then error message should be thrown
```

Formalizing user stories: Features

```
Feature: Multiple site support
  Only blog owners can post to a blog, except administrators, who can post to all blogs.

  Background:
    Given a global administrator named "Greg"
    And a blog named "Greg's anti-tax rants"
    And a customer named "Dr. Bill"
    And a blog named "Expensive Therapy" owned by "Dr. Bill"

  Scenario: Dr. Bill posts to his own blog
    Given I am logged in as Dr. Bill
    When I try to post to "Expensive Therapy"
    Then I should see "Your article was published."

  Scenario: Dr. Bill tries to post to somebody else's blog, and fails
    Given I am logged in as Dr. Bill
    When I try to post to "Greg's anti-tax rants"
    Then I should see "Hey! That's not your blog!"

  Scenario: Greg posts to a client's blog
    Given I am logged in as Greg
    When I try to post to "Expensive Therapy"
    Then I should see "Your article was published."
```

https://github.com/1c-syntax/turbo-gherkin-parser/blob/master/src/test/resources/correct/example3.feature

Scenarios → Acceptance Tests

```
Feature: User Login

  Scenario: Successful Login
    Given user navigates to polyfood.ch website
    When user logs in using Username as "jane_doe" and Password "password123"
    And clicks the Submit button
    Then Home page should be displayed
    And login should be successful
```

login.feature

```
class LoginSteps {
    @Given("user navigates to polyfood.ch website")
    fun navigatePage() {
        println("Cucumber executed Given statement")
        // Add actual navigation logic
    }

    @When("user logs in using Username as {string} and Password {string}")
    fun login(username: String, password: String) {
        println("Username is: $username")
        println("Password is: $password")
        // Add login logic here
    }

    @When("clicks the Submit button")
    fun clickTheSubmitButton() {
        println("Executing When statement")
        // Add button click logic here
    }

    @Then("Home page should be displayed")
    fun validatePage() {
        println("Executing 1st Then statement")
        // Add home page validation logic
    }

    @Then("login should be successful")
    fun validateLoginSuccess() {
        println("Executing 2nd Then statement")
        // Add login success validation logic
    }
}
```

LoginSteps.kt

Piecing it all together

```
Feature: User Login

  Scenario: Successful Login
    Given user navigates to polyfood.ch website
    When user logs in using Username as "jane_doe" and Password "password123"
    And clicks the Submit button
    Then Home page should be displayed
    And login should be successful
```

login.feature

```
class LoginSteps {
    @Given("user navigates to polyfood.ch website")
    fun navigatePage() {
        println("Cucumber executed Given statement")
        // Add actual navigation logic
    }

    @When("user logs in using Username as {string} and Password {string}")
    fun login(username: String, password: String) {
        println("Username is: $username")
        println("Password is: $password")
        // Add login logic here
    }

    @When("clicks the Submit button")
    fun clickTheSubmitButton() {
        println("Executing When statement")
        // Add button click logic here
    }

    @Then("Home page should be displayed")
    fun validatePage() {
        println("Executing 1st Then statement")
        // Add home page validation logic
    }

    @Then("login should be successful")
    fun validateLoginSuccess() {
        println("Executing 2nd Then statement")
        // Add login success validation logic
    }
}
```

LoginSteps.kt

- Keep features independent
- Ideally keep scenarios independent

1

As a vacation traveler,
I want to be able to get a refund anytime before my arrival,
so that I can get my money back in the event of an emergency.

`user story`

2

```
Feature: Vacation Traveler Refund

  Scenario Outline: Refund reservation
    Given the user has a reservation id "<id>"
    When the user clicks on the "refund" button
    Then a refund order is made for the reservation id "<id>"

  Examples:
    | id    |
    | 12345 |
    | 67890 |
```

`refund.feature`

3

```
class RefundSteps {

    private lateinit var reservationId: String

    @Given("the user has a reservation id {string}")
    fun theUserHasAReservationId(id: String) {
        reservationId = id
        // ... simulate retrieving the reservation by id ...
        println("Retrieved reservation with id: $reservationId")
    }

    @When("the user clicks on the {string} button")
    fun theUserClicksOnTheButton(buttonName: String) {
        // ... simulate clicking on the refund button ...
        println("User clicked on the $buttonName button")
    }

    @Then("a refund order is made for the reservation id {string}")
    fun aRefundOrderIsMadeForTheReservationId(id: String) {
        // ... simulate making a refund order for the reservation ...
        assert(reservationId == id)
        // ... more asserts to verify the actual refund logic check ...
        println("A refund order has been made for reservation id: $id")
    }
}
```

`RefundSteps.kt`

4

Implementation

Principles of BDD