Exercises

1. Here in Wipro, we will develop a real program that will be used in all our future Rookies Developer Program: The Rockers. This is a very, very basic program! This is just a register structure! Rockers is constituted by Employee, Team, Contract and User. The employee is a person that is characterized for its name, its Wipro ID, its Client ID (if existent) and its Wipro email. The employees are organized in Teams. Each team have a name. Each team should have at least one employee member. Each employee is under a project type (*ide est*, ESS, Commercial, Non-Commercial) and have an user. User is a virtual entity that have the Employee login and password of LDAP. Ever employee should be able to register itself, edit the information only about itself, and see who are all their colleagues, their Wipro ID and their Wipro e-mail.
2. The Earth is intested of monsters which can be vampires or werewolves. Vampires are beans that feed yourselves of blood and, for this reason, they control their hemoglobin levels. Vampires are characterized by name, born date and they can be or not the relation with their slaves (the people which they bite). Werewolves are bean that feed yourselves of meet and, for this motive, they control their protein level. Werewolves are, too, characterized by name, born date. They should know all the humans they "werewolved". Vampires and werewolves group thereselves and form clans. In vampire clans have only vampires. In werewolves clans have only werewolves. The clans should know all of their members. The monsters should know only their own clan. A monster should have the ability to develop some task in some date. Tasks are of two types: transformation or feed. The transformation task transforms a human bean in a monster by some ways. The feed task kills a human bean. Humans are characterized by their names and born date. Some humans are trained to hunt monsters. The training is performed by another human bean that have the expertise in technique to kill monsters. The kill technique is characterized for an object (used for kill the monster) and the mass of it. Here is an order that groups all hunters. This order is formed by a lot of hunters. This order indicates a mission to the hunter. This mission begins and finish in some date, having a duration time. Each hunter knows the number of monster it kill in some mission. And the mission knows the final score: number of monsters killed minus the number of the humans died. This is the world of a game. Develop in JAVA, the class structure that best wraps this world and a way to play a match in this game.
3. In the SET Team here in Mastercard, we use the JBehave Framework together with The MCFramework for Selenium to make automated tests. Those tests follows the philosophy called BDD (Behavior Driven Development). In this philosophy of programming, all development should be driven by the behavior. Is more than Test Driven, but business driven. One of the main characteristic of this approach is that we have a live documentation because the documentation is compiled together with the code. This documentation is a file with test scenarios described in some syntax that is named as Ghrekin Language. Those are the characteristics of this language:
   A) Keywords: Narrative (mark a set of lines that describes a set of scenarios); Scenario (a keyword that delimiter a scenario block); Given (that is followed by a phrasal description that explains in one line the preconditions of this scenario); When (that is followed by a phrasal description that contains a step condition); Then (that contains the conclusion step: what happens when we perform the actions in steps).

B) Rules:
   a. Narrative block should be ignored
   b. The “#” character should delimit the line of comment
   c. The Scenario line should be ignored
   d. The When line should be parsed
   e. When clause could receive one or more parameters
   f. Line that begins with “Meta:” string should be ignored
   g. Lines that begin with “@” character should be ignored

C) The challenge: You should create a class structure and a program that should receive in command line the path of a directory, find all .story files and parse them to .java code following the examples followed. The class name generated should be the name of story file but in camel-case. The JAVA file should be saved in a subdirectory with the name of current directory concatenated with the string “-JAVA”. The system should map and throws the expected exceptions and a new type of exception: the **StoryFileWrongFormatException** that should extend from Exception.

The story file:

```
# story-file-name.story

Meta:

@TAG1

@TAG2


Narrative:

As an Employee User

I want to login on the Rockers System

So that the login should be done with success


Scenario: Login in the system

Given User is on the login page

When User logs as UserEmployee

Then User is on the dashboard page
```

The JAVA file:

```
public class StoryFileName{
        @Given("User is on the login page")
        public void givenUserIsOnTheLoginPage(){
        //TODO
        }
        @When("User logs as UserEmployee")
        public void whenUserLogsAsUserEmployee(){
        //TODO
        }
        @Then("User is on the dashboard page")
        public void thenUserIsOnTheDashboardPage(){
        //TODO
        }
}
```