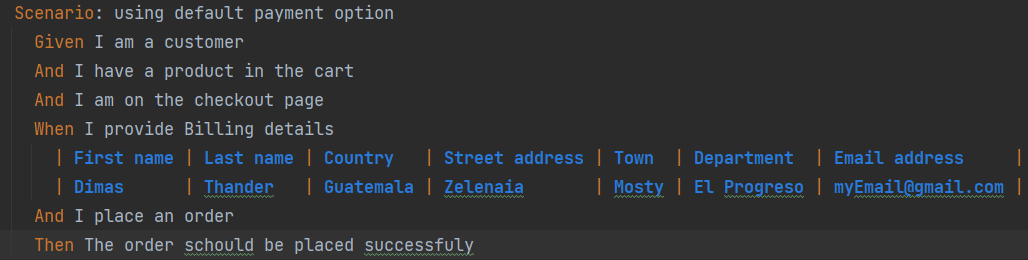
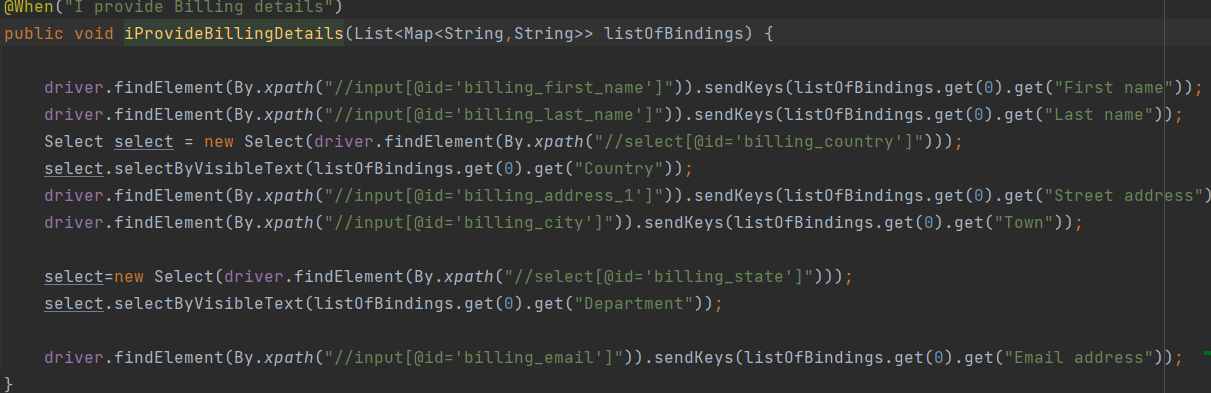
The main idea is the question of how we can share the states (arguments from feature fail) in steps definition.

For example we have scenario:

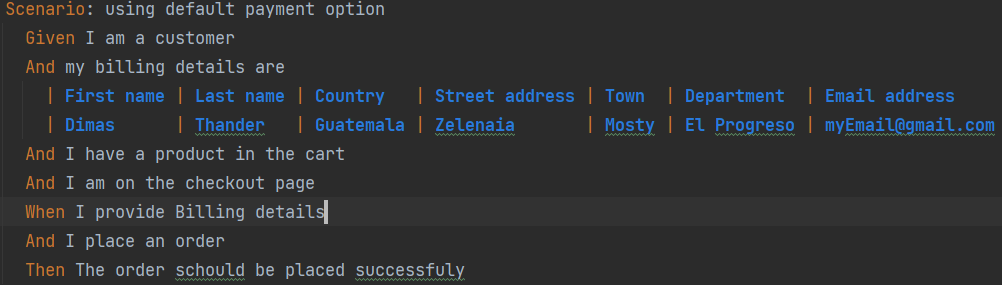


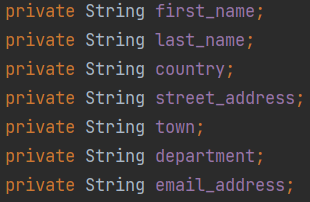
And in that scenario in a step ‘When’ we have multiple arguments (billing details) that we need to use in the step definition. This particular step definition looks in this way.



It is ok if we don't need to use these credentials any more in other steps in that scenario.

Then we have got changes in the scenario, and now it looks like this:



Notice that we have got 1 more step ‘And’ and the data table has been moved there. It means that the step definition of that step will use these arguments. But the step definition of the step ‘When’ still needs these arguments as well. So in this case we need some mechanism that allows us to share these arguments between steps definition. It's bad practice to repeat yourself and we are not going to define these arguments in every step definition that needs it. So we can define these arguments as a class variables (instance variables) in a step definition class, in that case we will have the opportunity to use them in any step definition. For example:

The problem with that approach is:

1)- we use a lot of code: to define the variables, to get the values from .feature fail, to assign the values to the variables, and all that code is in step definition file.

2)-We are not using domain object principle.

3)-As a result the code is technical and hard to read.

4)-Working with variables we can end up with adding static words, and by doing so we lose the parallel support.

5)-In the future there might be some more information from .feature fail (more arguments like this) and we will need to define more variables (+10 or +20) which is difficult to maintain. It doesn't make sense to create so many variables in a class. And what if we need to share those variables between the step definition classes? Right now we are able to share these states (variables) between steps inside of one step definition class, but we are not able to share it between different step definition classes.

So this is not a good practice to share the states between the steps and this is where the **dependency injection** framework comes into picture. But before it we need to create domain objects.

Lets create package domainObjects and in this package we create class BillingDitails. So we put all the variables from our step definition class to the BillingDitails class as fields, then we create getters, setters, constructor there. So it is gonna be our domain object, that we can define as a class variable in our step definition class, and then we can initialize it in one of our step definition method, and use all the fields of that object in any methods in a step definition class.

**Custom parameter**

But we need to remember that we have to accept a lot of arguments in order to initialize our BillingDitails object since it has a lot of fields. So instead of doing this in step definition class, we can use a custom parameter. So lets create package ‘customType’. then create a class CustomDataTableType. In this class we create method

@DataTableType

public BillingDitails billingDitailsInit(Map<String,String> map){

return new BillingDitails(map.get("First name"),

map.get("Last name"),

map.get("Country"),

map.get("Street address"),

map.get("Town"),

map.get("Department"),

map.get("Email address"));

The method gets arguments from .feature fail as a map (it is automatically recognized by cucumber) and used as a parameters to create a new object of our BillingDitails class and returns it.

You may ask why we need this CustomDataTableType class? We need it to make a step definition class looks more clear with less code and more readable. By using custom parameters it is easy to create domain objects, it takes less lines of code in step definitions class.