Outside-in Development

- What?
- How?
- The Gherkin language
- Example
- Exercise

Outside-In: What?

It’s a development mindset, focused by default on the business view, bridging the gap between development and business

This mindset derives from the Outside-In TDD approach, also called **London school** or **mockist**, an alternative approach to TDD, leading us to first consider a point of view external of our class, starting with the design of the public interface

Outside-In: How?

Since it's an approach more centered in the "user", we need to express the user's needs in the shape of a *Acceptance test*, which is a formalization of an *acceptance criteria*

Acceptance tests are organized similar to the arrange-act-assert sections of a unit test. However, in acceptance tests, these sections are usually known as given-when-then.
(Given) some context, (When) some action is carried out, (Then) a set of consequences should happen

Outside-In: Gherkin Language

Acceptance tests can be constructed with your usual testing framework like this:

```
1  class CarShould
2  {
3    [Test]
4    void decrease_speed_when_brakes_are_applied()
5    {
6        var car = new Car();
7        car.Accelerate(10);
8        var beforeBreakingVelocity = car.Speed();
9
10       car.Break(5);
11
12       Assert.That(car.Speed(), Is.LessThan(beforeBreakingVelocity));
13   }
14 }
```

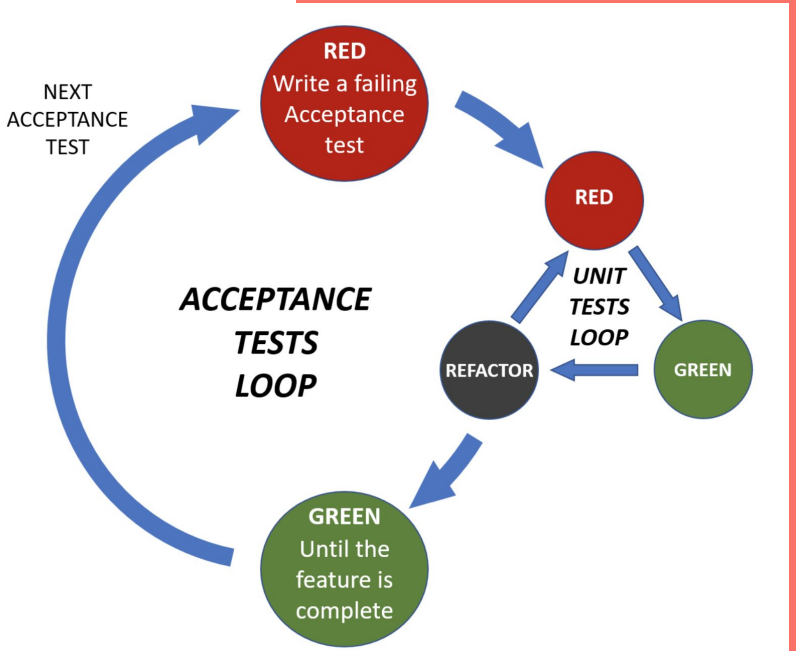
Or, you could use the Gherkin Language, which is more user/business friendly in order to specify your acceptance test (or scenario). The implementation still needs to happen somewhere, it's just that a BA or someone with more business knowledge can specify it for you

For more details on Gherkin syntax, you can visit: https://cucumber.io/docs/gherkin/reference/

Example:
Given **I have a car**
And **I accelerate for 10 seconds**
And **I break for 5 seconds**
When **I measure the velocity**
Then **the velocity should be lower than the one measured before breaking**

Outside-In: Double loop TDD

Double loop TDD introduces the concept of acceptance tests in the *Red->Green->Refactor* classic TDD loop. The idea is to have a high-level test that covers a business requirement. As the name implies, we add a second loop to our TDD loop. This is a outer loop and also has *Red->Green->Refactor* stages. In this acceptance test, with the outside loop we are effectively creating an **executable definition of done**



Revisit: Bank Kata

Try one more time this Kata, with an Outside-In approach. Can you spot the Acceptance criteria?