

## Ensinando como um objeto mockado deve reagir:

O mockito **cria um objeto que responde como se fosse o objeto real, porém ele não sabe o que fazer**. Sendo assim **cabe a nós ensinar como um objeto mockado deve reagir a cada pergunta feita para ele**, para que durante a execução do teste ele se comporte conforme exatamente a classe real se comportaria.

Vamos ver isso na prática **implementando uma nova regra, "não devemos alugar filmes para pessoas que estão devendo"**. Vamos imaginar que temos que pesquisar se a pessoa é uma devedora no spc, então criamos uma interface que irá pesquisar isso, com um único método abstrato que retorna um boolean caso o usuario esteja negativado:

```
1 package br.ce.wcaquino.servicos;
2
3 import br.ce.wcaquino.entidades.Usuario;
4
5 public interface SPCService {
6
7     public boolean possuiNegativacao(Usuario usuario);
8
9 }
10
```

Pronto, agora no método de alugar filme faremos a consulta, para isso precisamos de uma **variável de instância para usar como referência para o SPCService**:

```
public class LocacaoService {

    private LocacaoDAO dao;
    private SPCService spcService;
```

Agora usamos o spcService para criar nossa regra que lançará a exceção:

```
for(Filme filme : filmes) {
    if(filme.getEstoque() == 0) {
        throw new FilmeSemEstoqueException();
    }
}

if(spcService.possuiNegativacao(usuario)) {
    throw new LocadoraException("Usuário esta negativado no SPC");
}
```

E por fim um método para injetar o spcService:

```
public void setSpcService(SPCService spcService) {
    this.spcService = spcService;
}
```

## O teste:

Inicializamos nosso SPCService de forma mockada, e então o injetamos através do método setSpcService( ):

```
@Before
public void setUp() {
    servicoDeLocacao = new LocacaoService();
    LocacaoDAO dao = Mockito.mock(LocacaoDAO.class);
    servicoDeLocacao.setLocacaoDAO(dao);

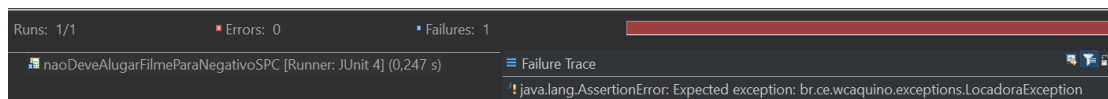
    SPCService spcService = Mockito.mock(SPCService.class);
    servicoDeLocacao.setSpcService(spcService);
}
```

Escrevemos o teste esperando a exceção pelo usuário estar negativado:

```
@Test(expected = LocadoraException.class)
public void naoDeveAlugarFilmeParaNegativoSPC() throws FilmeSemEstoqueException, LocadoraException {
    //GIVEN/DADOQUE
    Usuario caraTentandoAlugar = UsuarioBuilder.umUsuario().agora();
    List<Filme> listaDeFilmes = Arrays.asList(
        FilmeBuilder.umFilme().agora(),
        FilmeBuilder.umFilme().agora());

    //WHEN/QUANDO
    servicoDeLocacao.alugarFilme(caraTentandoAlugar, listaDeFilmes);
}
```

E se executarmos então temos um:



Isso acontece pq a **exceção só será lançada se o retorno do método do SPC for true**, porém o **valor padrão de um boolean é "false"**, então **precisamos alterar esse comportamento para que o SPC mockado retorne true**. Podemos fazer isso no próprio cenário do teste(GIVEN) usando o mockito:

```

@Test(expected = LocadoraException.class)
public void naoDeveAlugarFilmeParaNegativoSPC() throws FilmeSemEstoqueException, LocadoraException {
    //GIVEN/DADOQUE
    Usuario caraTentandoAlugar = UsuarioBuilder.umUsuario().agora();
    List<Filme> listaDeFilmes = Arrays.asList(
        FilmeBuilder.umFilme().agora(),
        FilmeBuilder.umFilme().agora());

    Mockito.when(spcService.possuiNegativacao(caraTentandoAlugar)).thenReturn(true);

    //WHEN/QUANDO
    servicoDeLocacao.alugarFilme(caraTentandoAlugar, listaDeFilmes);
}

```

Agora podemos dizer para nosso mock que **quando(when)** o `spcService.possuiNegativacao()` **chamar o usuário negativado** ele **então retornará true**. Um ponto a se observar é a legibilidade do teste que está fluente.

E também uma pequena mudança nas **variáveis que recebem o mock**, tornamos elas globais para que **todos os métodos possam acessá-las e alterar seu comportamento** e no **@Before** apenas as instanciamos como mock:

```

private SPCService spcService;
private LocacaoDAO dao;

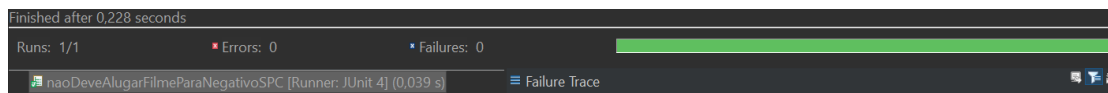
@Before
public void setUp() {
    servicoDeLocacao = new LocacaoService();

    dao = Mockito.mock(LocacaoDAO.class);
    servicoDeLocacao.setLocacaoDAO(dao);

    spcService = Mockito.mock(SPCService.class);
    servicoDeLocacao.setSpcService(spcService);
}

```

Agora vamos executar o teste novamente e o resultado é:



## Por que os outros métodos que usam o SPCService não quebraram com a modificação:

O mock apenas alterou a situação do método para o usuário especificado no teste que era o usuário **"caraTentandoAlugar"**, dessa maneira para os outros testes que o comportamento dele não foi alterado ele retornou o valor **padrão que é "false"**.

