02

Capturando argumentos recebidos pelo Mock

Tivemos mais uma mudança em nosso sistema. Precisamos agora de um batch job que pegue leilões encerrados e gere um pagamento associado ao valor desse leilão. Para isso, vamos criar a classe `Pagamento` e a interface `RepositorioDePagamentos`, que representa o contrato de acesso aos pagamentos já armazenados:

```
public class Pagamento {

    private double valor;
    private Calendar data;

    public Pagamento(double valor, Calendar data) {
        this.valor = valor;
        this.data = data;
    }
    public double getValor() {
        return valor;
    }
    public Calendar getData() {
        return data;
    }
}

public interface RepositorioDePagamentos {
    void salva(Pagamento pagamento);
}
```

Agora vamos a classe que gera a lista de pagamentos de acordo com os leilões encerrados:

```
public class GeradorDePagamento {

    private final RepositorioDePagamentos pagamentos;
    private final RepositorioDeLeiloes leiloes;
    private final Avaliador avaliador;

    public GeradorDePagamento(RepositorioDeLeiloes leiloes,
            RepositorioDePagamentos pagamentos,
            Avaliador avaliador) {
        this.leiloes = leiloes;
        this.pagamentos = pagamentos;
        this.avaliador = avaliador;
    }

    public void gera() {

        List<Leilao> leiloesEncerrados = leiloes.encerrados();
        for(Leilao leilao : leiloesEncerrados) {
            avaliador.avalia(leilao);

            Pagamento novoPagamento =
                    new Pagamento(avaliador.getMaiorLance(), Calendar.getInstance());
            pagamentos.salva(novoPagamento);
```

```
        }
    }
}
```

Veja que a regra de negócio é bem simples: ela pega todos os leilões encerrados, avalia o leilão para descobrir o maior lance, gera um novo pagamento com o valor e a data corrente e salva esse pagamento no repositório.

Agora, precisamos testar essa nossa nova classe. Vamos lá:

```
public class GeradorDePagamentoTest {

    @Test
    public void deveGerarPagamentoParaUmLeilaoEncerrado() {

        RepositorioDeLeiloes leiloes = mock(RepositorioDeLeiloes.class);
        RepositorioDePagamentos pagamentos = mock(RepositorioDePagamentos.class);
        Avaliador avaliador = mock(Avaliador.class);

        Leilao leilao = new CriadorDeLeilao()
            .para("Playstation")
            .lance(new Usuario("José da Silva"), 2000.0)
            .lance(new Usuario("Maria Pereira"), 2500.0)
            .constroi();

        when(leiloes.encerrados()).thenReturn(Arrays.asList(leilao));
        when(avaliador.getMaiorLance()).thenReturn(2500.0);

        GeradorDePagamento gerador =
            new GeradorDePagamento(leiloes, pagamentos, avaliador);
        gerador.gera();

        // como fazer assert no Pagamento gerado?
    }
}
```

O problema é como fazer o assert no `Pagamento` que é gerado pela classe `GeradorDePagamento`. Afinal, ele é instanciado internamente e não temos como recuperá-lo no nosso método de teste.

Mas repare que a instância é passada para o `RepositorioDePagamentos`, que é um mock! Então, podemos pedir ao Mock para guardar esse objeto para que possamos o recuperar a fim de realizar as asserções! A classe do Mockito que faz isso é chamada de `ArgumentCaptor`, ou seja, capturador de argumentos.

Para a utilizarmos, precisamos instanciá-la, passando qual a classe que será recuperada. No nosso caso, é a classe `Pagamento`. Em seguida, fazemos uso do `verify()` e checamos a execução do método que recebe o atributo. Como parâmetro, passamos o método `capture()` do ArgumentCaptor:

```
// criamos o ArgumentCaptor que sabe capturar um Pagamento
ArgumentCaptor<Pagamento> argumento = ArgumentCaptor.forClass(Pagamento.class);
// capturamos o Pagamento que foi passado para o método salvar
verify(pagamentos).salvar(argumento.capture());
```

Simples assim! Agora o `argumento` contém a instância de `Pagamento` criada! Basta pegarmos a instância do Pagamento através do método `argumento.getValue()` :

```
Pagamento pagamentoGerado = argumento.getValue();
```

Pronto! Agora temos tudo para escrever o teste:

```
@Test
public void deveGerarPagamentoParaUmLeilaoEncerrado() {

    RepositorioDeLeiloes leiloes = mock(RepositorioDeLeiloes.class);
    RepositorioDePagamentos pagamentos = mock(RepositorioDePagamentos.class);
    Avaliador avaliador = mock(Avaliador.class);

    Leilao leilao = new CriadorDeLeilao()
        .para("Playstation")
        .lance(new Usuario("José da Silva"), 2000.0)
        .lance(new Usuario("Maria Pereira"), 2500.0)
        .constroi();

    when(leiloes.encerrados()).thenReturn(Arrays.asList(leilao));
    when(avaliador.getMaiorLance()).thenReturn(2500.0);

    GeradorDePagamento gerador =
        new GeradorDePagamento(leiloes, pagamentos, avaliador);
    gerador.gera();

    ArgumentCaptor<Pagamento> argumento = ArgumentCaptor.forClass(Pagamento.class);
    verify(pagamentos).salva(argumento.capture());
    Pagamento pagamentoGerado = argumento.getValue();
    assertEquals(2500.0, pagamentoGerado.getValor(), 0.00001);
}
```

Veja que o `ArgumentCaptor` possibilita capturar a instância que foi passada para o Mock. Isso é especialmente útil em situações como essas: nossa classe de produção instancia um novo objeto, que é passado para uma das dependências. Assim, conseguimos garantir que o objeto criado está correto.