Nosso objetivo agora é modificar um código já existente, fazendo com que, quando o usuário tentar propor dois lances seguidos, seja lançada uma \DomainException() com a mensagem "Usuário não pode propor dois lances consecutivos". Além disso, quando o total de lances por usuário for mais do que 5, também lançaremos uma \DomainException(), dessa vez com a mensagem "Usuário não pode propor mais de 5 lances por leilão".

public function recebeLance(Lance $lance)

{

if (!empty($this->lances) && $this->ehDoUltimoUsuario($lance)) {

throw new \DomainException('Usuário não pode propor 2 lances seguidos');

}

$totalLancesUsuario = $this->quantidadeLancesPorUsuario($lance->getUsuario());

if ($totalLancesUsuario >= 5) {

throw new \DomainException('Usuário não propor mais de 5 lances por leilão');

}

$this->lances[] = $lance;

}

Com as exceções sendo lançadas, executaremos os testes no terminal, o que nos retornará dois erros - afinal, já tínhamos testes para esses dois casos, mas eles não lançavam exceções. Portanto, teremos que alterar os métodos testLeilaoNaoDeveReceberLancesRepetidos() e testLeilaoNaoDeveAceitarMaisDe5LancesPorUsuario() de LeilaoTest.

No primeiro deles,incluiremos a função expectException() esperando uma \DomainException, além de um expectExceptionMessage() com a mensagem de erro que estamos mandando ao usuário. Com isso, não conseguiremos mais fazer as verificações com assertCount() e assertEquals(), já que, quando o método lançar uma exceção, esse trecho de código não será executado.

public function testLeilaoNaoDeveReceberLancesRepetidos()

{

$this->expectException(\DomainException::class);

$this->expectExceptionMessage('Usuário não pode propor 2 lances seguidos');

$leilao = new Leilao('Variante');

$ana = new Usuario('Ana');

$leilao->recebeLance(new Lance($ana, 1000));

$leilao->recebeLance(new Lance($ana, 1500));

}

Ao executarmos o teste no terminal, teremos somente um erro, referente ao método de teste que ainda não alteramos. Repetiremos o processo em testLeilaoNaoDeveAceitarMaisDe5LancesPorUsuario(), adicionando as funções expectException() e expectExceptionMessage() e removendo as verificações que fazíamos anteriormente.

Poderíamos utilizar outro tipo de exceção, inclusive uma criada manualmente - algo que, na verdaderia, seria o recomendado. Porém, manteremos sempre a \DomainException para focarmos na criação de testes.

public function testLeilaoNaoDeveAceitarMaisDe5LancesPorUsuario()

{

$this->expectException(\DomainException::class);

$this->expectExceptionMessage('Usuário não propor mais de 5 lances por leilão');

$leilao = new Leilao('Brasília Amarela');

$joao = new Usuario('João');

$maria = new Usuario('Maria');

$leilao->recebeLance(new Lance($joao, 1000));

$leilao->recebeLance(new Lance($maria, 1500));

$leilao->recebeLance(new Lance($joao, 2000));

$leilao->recebeLance(new Lance($maria, 2500));

$leilao->recebeLance(new Lance($joao, 3000));

$leilao->recebeLance(new Lance($maria, 3500));

$leilao->recebeLance(new Lance($joao, 4000));

$leilao->recebeLance(new Lance($maria, 4500));

$leilao->recebeLance(new Lance($joao, 5000));

$leilao->recebeLance(new Lance($maria, 5500));

$leilao->recebeLance(new Lance($joao, 6000));

}

Executando os testes no terminal, todos passarão corretamente, inclusive marcados com um X no arquivo testes-executados.txt.

Alura\Leilao\Tests\Model\Leilao

[x] Leilao nao deve receber lances repetidos

[x] Leilao nao deve aceitar mais de 5 lances por usuario

[x] Leilao deve receber lances with data set "2-lances"

[x] Leilao deve receber lances with data set "1-lance"

Já temos um total de 14 métodos de teste sendo executados, incluindo métodos específicos e utilização dos *data providers*, além de 29 asserções sendo feitas em um projeto pequeno, e ainda poderíamos incluir mais testes, pois não esgotamos as possibilidades.

Por exemplo, em algum momento o leilão precisa ser finalizado, uma funcionalidade que ainda não temos. E o que acontece, ou deveria acontecer, se tentarmos avaliar um leilão finalizado? Conversaremos sobre isso no próximo vídeo.