06

Consolidando o seu conhecimento

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Crie a classe `Alura\Leilao\Service\EnviadorEmail`, com o seguinte conteúdo:

```php
<?php

namespace Alura\Leilao\Service;

use Alura\Leilao\Model\Leilao;

class EnviadorEmail
{
    public function notificaTerminoLeilao(Leilao $leilao)
    {
        $sucesso = mail(
            'usuario@example.com',
            'Leilão finalizado',
            "Leilão para {$leilao->recuperarDescricao()} finalizado."
        );

        if (!$sucesso) {
            throw new \DomainException('Erro ao enviar e-mail');
        }
    }
}
```

2) Garanta que a sua classe `Alura\Leilao\Service\Encerrador` receba o enviador de e-mail como dependência, e a utilize ao encerrar, da seguinte forma:

```php
<?php

namespace Alura\Leilao\Service;

use Alura\Leilao\Dao\Leilao as LeilaoDao;

class Encerrador
{
    private $dao;
    private $enviadorEmail;

    public function __construct(LeilaoDao $dao, EnviadorEmail $enviadorEmail)
    {
        $this->dao = $dao;
        $this->enviadorEmail = $enviadorEmail;
    }

    public function encerra()
    {
        $leiloes = $this->dao->recuperarNaoFinalizados();
```

```
            foreach ($leiloes as $leilao) {
                if ($leilao->temMaisDeUmaSemana()) {
                    $leilao->finaliza();
                    $this->dao->atualiza($leilao);
                    $this->enviadorEmail->notificaTerminoLeilao($leilao);
                }
            }
        }
    }
```

3) Crie, na classe `EncerradorTest` , um teste que garanta que ambos os leilões sejam atualizados e encerrados, mesmo caso o enviador de e-mails lance uma exceção. Não se esqueça de cuidar do código de testes:

```
<?php

namespace Alura\Leilao\Tests\Domain;

use Alura\Leilao\Dao\Leilao as LeilaoDao;
use Alura\Leilao\Model\Leilao;
use Alura\Leilao\Service\Encerrador;
use Alura\Leilao\Service\EnviadorDeEmail;
use PHPUnit\Framework\TestCase;

class EncerradorTest extends TestCase
{
    private $encerrador;
    private $enviadorDeEmailMock;

    protected function setUp(): void
    {
        $leilaoFiat = new Leilao('Fiat 147 0Km', new \DateTimeImmutable('8 days ago'));
        $leilaoVariante = new Leilao('Variante 0Km', new \DateTimeImmutable('10 days ago'));

        $leilaoDaoMock = $this->createMock(LeilaoDao::class);
        $leilaoDaoMock->method('recuperarFinalizados')
            ->willReturn([$leilaoFiat, $leilaoVariante]);

        $leilaoDaoMock->expects(self::once())
            ->method('recuperarNaoFinalizados')
            ->willReturn([$leilaoFiat, $leilaoVariante]);

        $leilaoDaoMock->expects(self::exactly(2))
            ->method('atualiza')
            ->withConsecutive(
                [$leilaoFiat],
                [$leilaoVariante]
            );


        $this->enviadorDeEmailMock = $this->getMockBuilder(EnviadorDeEmail::class)->getMock();

        $this->encerrador = new Encerrador($leilaoDaoMock, $this->enviadorDeEmailMock);
    }

    public function testDeveEncerrarLeiloesComMaisDeUmaSemana()
    {
        $this->encerrador->encerra();
```

```
        $leiloesEncerrados = $this->leilaoDao->recuperarFinalizados();

        static::assertCount(2, $leiloesEncerrados);
        static::assertTrue($leiloesEncerrados[0]->estaFinalizado());
        static::assertTrue($leiloesEncerrados[1]->estaFinalizado());
        static::assertEquals(
            'Fiat 147 0Km',
            $leiloesEncerrados[0]->recuperarDescricao()
        );
        static::assertEquals(
            'Variante 0Km',
            $leiloesEncerrados[1]->recuperarDescricao()
        );
    }

    public function testDeveContinuarOProcessoamentoAoEncontrarErroAoEnviarEmail()
    {
        $e = new \DomainException('Erro ao enviar e-mail');

        $this->enviadorDeEmailMock->expects(self::exactly(2))
            ->method('notificaTerminoLeilao')
            ->willThrowException($e);

        $this->encerrador->encerra();
    }
}
```

4) Execute os testes e garanta que o código está falhando. A exceção que está sendo lançada ainda não foi capturada.

5) Envolva o código do `Encerrador` em um bloco `try catch`:

```
...
public function encerra()
{
    $leiloes = $this->dao->recuperarNaoFinalizados();

    foreach ($leiloes as $leilao) {
        if ($leilao->temMaisDeUmaSemana()) {
            try {
                $leilao->finaliza();
                $this->dao->atualiza($leilao);
                $this->enviadorDeEmail->notificaTerminoLeilao($leilao);
            } catch (\DomainException $e) {
                error_log($e->getMessage());
            }
        }
    }
}
...
```

6) Execute novamente os testes. Garanta que os testes passam, e que o *log* de erro está sendo exibido na saída do comando.