11

# Colocando em prática

Chegamos ao fim de mais um capítulo, e claro a mais a um Colocando em prática. Neste capítulo aprendemos um pouco mais sobre teste e o JUnit, e agora vamos praticar isto tudo. Veja as orientações abaixo.

Seguem as dicas e linhas gerais:

1- Vamos começar criando a classe `NegociacaoTest` no pacote `br.com.alura.argentum.modelo` . Nesta classe de testes vamos implementar o método `naoCriaNegociacaoComDataNula` , que deve **esperar** o lançamento de uma exceção quando uma Negociacao com data nula for criada.

2- Modifique o construtor de Negociacao para que ele jogue uma `IllegalArgumentException` com a mensagem `data não pode ser nula` quando receber uma data nula como argumento .

3- Crie um teste chamado `naoCriaNegociacaoComPrecoNegativo` , que deve esperar uma `IllegalArgumentException.class` quando o preço da negociacao for negativo.

4- Modifique novamente o construtor de Negociacao para que ele jogue uma `IllegalArgumentException` com a mensagem `preco não pode ser negativo` quando recebe um preço negativo como argumento.

5- Crie um teste chamado `naoCriaNegociacaoComQuantidadeZero` , que deve esperar uma `IllegalArgumentException.class` quando a quantidade da negociacao for zero.

6- Modifique outra vez o construtor de Negociacao para que ele jogue uma `IllegalArgumentException` com a mensagem `quantidade deve ser pelo menos 1` quando recebe uma quantidade menor ou igual a zero como argumento.

7- Neste capítulo também aprendems sobre o padrão de projeto Builder, e vamos começar a implementá-lo para facilitar a criação de objetos `Candlestick` . Crie a classe `CandleBuilder` no pacote `br.com.alura.argentum.modelo` . Esta classe deve ter os mesmo atributos que o `Candlestick` .

8- Para cada atributo da classe `CandleBuilder` , crie um método com o nome semântico responsável de receber o valor deste atributo, setá-lo e que retorne o próprio CandleBuilder (Utilize o `this` ). Por exemplo, com o atributo **abertura**, vamos criar o seguinte método:

```
public CandleBuilder comAbertura(double abertura){
    this.abertura = abertura;
    return this;
}
```

Crie um método análogo a este para cada atributo.

9- Crie o método `geraCandle()` que retorna um `Candlestick` com os valores dos atributos de `CandleBuider` .

10- Vamos partir agora para os testes da classe Candlestick. Crie a classe `CandlestickTest` e dentro dela o método `precoMaximoNaoPodeSerMenorQueMinimo()` que deve esperar uma exceção `IllegalArgumentException` caso o preço máximo seja menor que o mínimo. Aproveite o seu novo CandleBuilder para montar o Candlestick.

11- Altere o construtor do `Candlestick` para que ele jogue uma `IllegalArgumentException` com a mensagem "O minimo não pode ser maior que o maximo" caso o valor maximo seja menor que o valor de minimo.

12- Crie outro teste para o nosso Candlestick chamado `quandoAberturaIgualFechamentoEhAlta()` que deve ser responsável por testar o seguinte cenário: Se o Candlestick tiver o mesmo valor de abertura e fechamento , ele deve retornar que é um candlestick de alta quando chamamos o método `isAlta()` . Utilize o método Assert.assertTrue para fazer essa comparação.

13- Altere o método `isAlta()` de Candlestick para que ele reflita o comportamento mencionado no teste.