07

Para saber mais - Utilizando Data Builders para preparar ambiente

Durante a implementação do teste para impedir lances de usuários que já deram cinco lances, fizemos a seguinte implementação de cenário:

```
@Test
public void naoDeve_AdicionarLance_QuandoUsuarioDerCincoLances(){
    CONSOLE.propoe(new Lance(ALEX,  100.0));
    final Usuario FRAN = new Usuario("Fran");
    CONSOLE.propoe(new Lance(FRAN,  200.0));
    CONSOLE.propoe(new Lance(ALEX,  300.0));
    CONSOLE.propoe(new Lance(FRAN,  400.0));
    CONSOLE.propoe(new Lance(ALEX,  500.0));
    CONSOLE.propoe(new Lance(FRAN,  600.0));
    CONSOLE.propoe(new Lance(ALEX,  700.0));
    CONSOLE.propoe(new Lance(FRAN,  800.0));
    CONSOLE.propoe(new Lance(ALEX,  900.0));
    CONSOLE.propoe(new Lance(FRAN,  1000.0));
    CONSOLE.propoe(new Lance(ALEX,  1100.0));
    CONSOLE.propoe(new Lance(FRAN,  1200.0));

    int quantidadeLancesDevolvida = CONSOLE.quantidadeLances();

    assertEquals(10, quantidadeLancesDevolvida);
}
```

Perceba que para preparar o objeto do tipo `Leilao` , foi necessário chamar várias vezes o método `propoe()` criando várias instâncias, deixando o código cada vez mais verboso.

Para casos como esses, é bem comum criar classes conhecidas como *Data Builder* que utiliza o Design Pattern Builder (https://pt.wikipedia.org/wiki/Builder). Dado esse exemplo, poderíamos criar o seguinte Data Builder:

```
package br.com.alura.leilao.builder;

import br.com.alura.leilao.model.Lance;
import br.com.alura.leilao.model.Leilao;
import br.com.alura.leilao.model.Usuario;

public class LeilaoBuilder {

    private final Leilao leilao;

    public LeilaoBuilder(String descricao) {
        this.leilao = new Leilao(descricao);
    }

    public LeilaoBuilder lance(Usuario usuario, double valor) {
        this.leilao.propoe(new Lance(usuario, valor));
        return this;
    }
```

```
    public Leilao build() {
        return leilao;
    }

}
```

Então, poderíamos construir o leilão da seguinte maneira:

```
@Test
public void naoDeve_AdicionarLance_QuandoUsuarioDerCincoLances() {
    final Usuario FRAN = new Usuario("Fran");

    final Leilao console = new LeilaoBuilder("Console")
            .lance(ALEX, 100.0)
            .lance(FRAN, 200.0)
            .lance(ALEX, 300.0)
            .lance(FRAN, 400.0)
            .lance(ALEX, 500.0)
            .lance(FRAN, 600.0)
            .lance(ALEX, 700.0)
            .lance(FRAN, 800.0)
            .lance(ALEX, 900.0)
            .lance(FRAN, 1000.0)
            .lance(ALEX, 1100.0)
            .lance(FRAN, 1200.0)
            .build();

    int quantidadeLancesDevolvida = console.quantidadeLances();

    assertEquals(10, quantidadeLancesDevolvida);
}
```

Temos exatamente o mesmo teste, porém, de uma maneira mais enxuta e concisa. Essa técnica é bastante útil em objetos que são complexos para serem preparados.