10

Mãos à obra!

Vamos começar a escrever a classe Avaliador. Portando, vamos até o módulo de domínio e começamos e declará-la:

```
class Avaliador:
```

Essa classe vai nos informar qual o maior e qual o menor lance do leilão. Vamos inicializá-la com esses dois atributos com o menor e com o maior valor para um número de ponto flutuante no sistema:

```
class Avaliador:

    def __init__(self):
        self.maior_lance = sys.float_info.min
        self.menor_lance = sys.float_info.max
```

Bacana! Agora precisamos criar o método avalia. Esse método percorrerá a lista de lances no leilão e verá se o valor deste lance é maior ou menor do que o valor que está nos atributos:

```
class Avaliador:

    def __init__(self):
        self.maior_lance = sys.float_info.min
        self.menor_lance = sys.float_info.max

    def avalia(self, leilao):
        for lance in leilao.lances:
            if lance.valor > self.maior_lance:
                self.maior_lance = lance.valor
            elif lance.valor < self.menor_lance:
                self.menor_lance = lance.valor
```

Vamos voltar ao arquivo `principal.py` e testar essa classe:

```
from src.leilao.dominio import Leilao, Usuario, Lance, Avaliador


# restante do código omitido

avaliador = Avaliador()
avaliador.avalia(leilao_celular)

print(f'Maior lance: {avaliador.maior_lance}')
print(f'Menor lance: {avaliador.menor_lance}')
```

Quando executamos esse código, obtemos a seguinte saída:

```
Lance do usuario Gui com o valor de 200.0
Lance do usuario Yuri com o valor de 100.0
Maior lance: 200.0
Menor lance: 100.0
```

Porém quando trocamos a ordem que os lances são inseridos no sistema obtemos essa saída:

```
Lance do usuario Yuri com o valor de 100.0
Lance do usuario Gui com o valor de 200.0
Maior lance: 200.0
Menor lance: 1.7976931348623157e+308
```

Ou seja, temos um bug no sistema. Nosso código tem uma falha! Com isso, vimos o porquê é importante testar um software. Quando testamos o código, pensamos nos possíveis casos de uso que envolvem aquela regra de negócio. Por isso, garantimos que, para aqueles casos de uso testados, o código funciona corretamente.

A IDE Pycharm já possui alguns atalhos para a criação de testes. Para isso, basta colocar o cursor em cima da classe `Avaliador` e digitar o atalho **Ctrl+Alt+T**. Vamos chamar esse teste de `test_avaliador.py` e selecionar o método `avalia` para ser testado.

O Pycharm já cria um método para a gente com o nome `test_avalia`. Esse prefixo `test_*` o Python utiliza para reconhecer que este é um método de testes.

Vamos escrever o teste do avaliador nesse método. Para isso, vamos instanciar os objetos das classes de Leilão, Usuário, Lance e Avaliador.

Para fazer a asserção dos valores podemos utilizar o método `assertEqual` que herdamos da classe `TestCase`. Vemos que na primeira vez que rodamos o teste ele falha. Vamos arrumar o código na classe `Avaliador`. O problema se encontra na parte que fazemos a checagem dos valores no `if`. Basta trocar o `elif` por um `if` que conseguimos resolver esse problema:

```
    def avalia(self, leilao):
        for lance in leilao.lances:
            if lance.valor > self.maior_lance:
                self.maior_lance = lance.valor
            if lance.valor < self.menor_lance:
                self.menor_lance = lance.valor
```

Quando rodamos o teste agora, ele passa. :)