12

Para saber mais: Strategy na API Java

O padrão *Strategy* é muito utilizado em projetos e também na API padrão Java. O exemplo clássico do padrão *Strategy* é a interface `java.util.Comparator`. Através dela podemos definir uma *estratégia* de ordenação.

Por exemplo, pode fazer sentido ordenar as nossas negociações pela *quantidade*. Veja o código abaixo que cria e imprime a quantidade de 4 negociações, ainda sem ordenação:

```
public class TesteOrdenacao {

    public static void main(String[] args) {

        LocalDateTime hoje = LocalDateTime.now();

        Negociacao negociacao1 = new Negociacao(40.0, 120, hoje);
        Negociacao negociacao2 = new Negociacao(35.0, 900, hoje);
        Negociacao negociacao3 = new Negociacao(45.0, 50, hoje);
        Negociacao negociacao4 = new Negociacao(20.0, 200, hoje);

        List<Negociacao> lista = Arrays.asList(negociacao1, negociacao2, negociacao3, negociaca

        //aqui queremos ordenar antes de imprimir

        for (Negociacao negociacao : lista) {
            System.out.println(negociacao.getQuantidade());
        }
    }

}
```

Para nossa ajuda, a lista possui um método `sort` que pode receber uma *estratégia* de ordenação. Então vamos criar uma classe dedicada que sabe definir quando uma negociação é *maior* ou *menor* do que a outra. Essa classe deve implementa a interface `Comparator`:

```
import java.util.Comparator;

public class QuantidadeComparator implements Comparator<Negociacao> {

    @Override
    public int compare(Negociacao n1, Negociacao n2) {
        return n1.getQuantidade() - n2.getQuantidade();
    }
}
```

Repare que o `Comparator` define apenas um método `compare` como a nossa interface `Indicador` também só definiu um método. Dentro do método `compare` estamos comparando duas negociações (obvio!). Pode parecer estranho, mas basta devolve um `int` positivo para dizer que `n1` é maior do `n2`, ou um `int` negativo para dizer que `n2` é maior de `n1` e devolver o valor `0` para definir que `n1` e `n2` são iguais.

Retorno do método `compare` : - int positivo: `n1 > n2` - 0: `n1 == n2` - int negativo: `n2 > n1`

Sabendo disso só falta chamar o método `sort` antes de imprimir, *passando a estratégia* como objeto:

```
public class TesteOrdenacao {

    public static void main(String[] args) {

        LocalDateTime hoje = LocalDateTime.now();

        Negociacao negociacao1 = new Negociacao(40.0, 120, hoje);
        Negociacao negociacao2 = new Negociacao(35.0, 900, hoje);
        Negociacao negociacao3 = new Negociacao(45.0, 50, hoje);
        Negociacao negociacao4 = new Negociacao(20.0, 200, hoje);

        List<Negociacao> lista = Arrays.asList(negociacao1, negociacao2, negociacao3, negociacao

        //ordenando com nossa estrategia de comparacao
        lista.sort(new QuantidadeComparator());

        for (Negociacao negociacao : lista) {
            System.out.println(negociacao.getQuantidade());
        }
    }
}
```

O resultado é:

```
50
120
200
900
```

Faça o teste!