Ordenação

Quando a gente ordena uma coleção, as operações de busca ficam mais rápidas. Por exemplo, os próprios carros, **se a gente os ordenasse pelo preço de forma crescente**, seria simples pegar os 3 mais baratos *(os 3 primeiros ué)* e os 3 mais caros *(3 últimos).*

O melhor jeito – usar a implementação pronta do java

A melhor forma de fazer isso é usar o **sort()** que recebe um **Comparator** que toda coleção de java já tem:

```
carros.sort(Comparator.comparing(Carro::getPreco));
carros.forEach(carro -> System.out.println("Carro modelo "+ carro.getModelo() + " custa: " + carro.getPreco()));

Carro modelo Brasilia custa: 16.000
Carro modelo Fusca custa: 17.000
Carro modelo Jipe custa: 46.000
Carro modelo Smart custa: 54.000
Carro modelo Lamborghini custa: 1000000
```

Funciona, mas é muito custoso - Selectionsort

Usando uma abordagem diferente (agora pra fins didáticos), podemos ir **trocando os elementos de posição** no array caso eles sejam “menores”.

Basicamente falando, a gente vai **iterar sobre a coleção que queremos ordenar,** para cada elemento da coleção **nós vamos comparar quem tem o menor valor** *(percorrendo toda a lista e pegando o índice do elemento com menor valor).* Então **tendo o elemento atual da iteração** e o **menor elemento,** basta troca-los de lugar:

```
for(int atual = 0; atual < carros.length; atual++){
    int indiceDoMenor = encontrarIndiceDoMenor(carros, atual, carros.length);

    Carro carroAtual = carros[atual];
    Carro carroMaisBarato = carros[indiceDoMenor];

    carros[atual] = carroMaisBarato;
    carros[indiceDoMenor] = carroAtual;
}
```

```
public static int encontrarIndiceDoMenor(Carro[] lista, int inicio, int termino){

    int indiceMenor = inicio;

    for(int posicaoAtual = inicio ; posicaoAtual < termino ; posicaoAtual ++){
        if(lista[posicaoAtual].getPreco().compareTo(lista[indiceMenor].getPreco()) < 0){
            indiceMenor = posicaoAtual;
        }
    }

    return indiceMenor;
}
```

Essa estratégia funciona porque a cada iteração a gente vai mandar a próxima posição (atual) pra ser iterada e ver se tem alguém que vale menos que ela, então sempre vai ser ***”daqui pra cá tem alguém que vale menos?”***

Se tiver a gente simplesmente vai trocar o que vale menos pelo atual, dessa maneira o que vale menos vai ser sempre jogado a esquerda do atual, naturalmente ordenando eles.

Funciona, e é menos custoso – InsertionSort

O **insertionSort** é um pouco mais legível e menos custoso computacionalmente falando, a ideia é que **dado a um array recebido eu possa ordenar ele** fazendo com que o **elemento atual pergunte**

ao **elemento anterior** se **ele** é menor ou não, então se for eu troco de lugar e **continuo fazendo isso** até chegar a 0 (primeira posição) ou **até chegar a alguém que é menor que ele**.

É tipo: ***"se eu sou menor que você então eu vou ficar na sua frente e vou continuar fazendo isso até chegar a alguém menor que eu ou eu estiver na primeira posição":***