Laboratório de Estrutura de Dados

Segunda versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

UNIVERSIDADE ESTADUAL DA PARAÍBA

CENTRO DE CIÊNCIAS E TECNOLOGIA

CIÊNCIA DA COMPUTAÇÃO

Lucas de Lucena Siqueira - 201080354

Daniel Xavier Brito de Araújo - 201080303

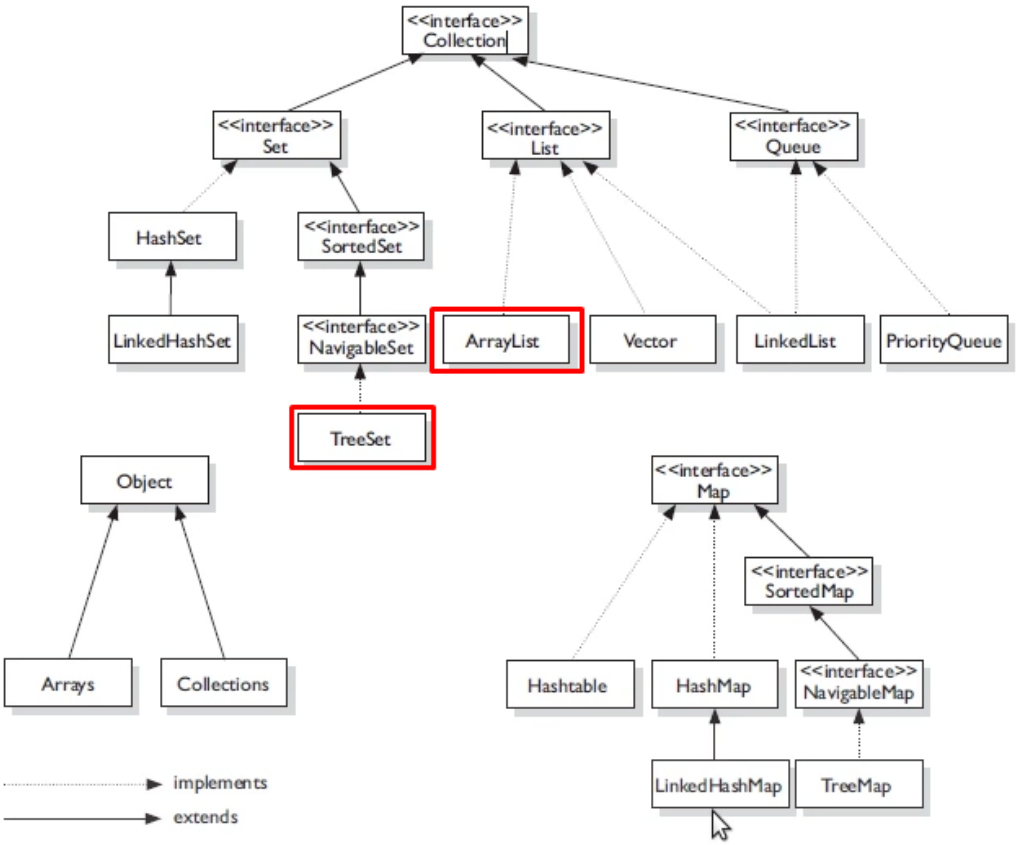
Ian Vasconcelos Bernardo - 201080044

Campina Grande
2022

1. Introdução

Este relatório corresponde ao segundo relato dos resultados obtidos onde mudanças foram realizadas com objetivo de melhorar algumas limitações no projeto da disciplina de Estrutura de Dados. Com um dataset da AirBnB de Berlim, empresa responsável por promover acomodações, foi possível realizar a análise de diversos anúncios contendo informações gerais como nome do dono, vizinhança, latitude, longitude, tipo do lugar, preço, avaliações, dentre outros atributos.

Dessa vez foram feitas algumas novas implementações a partir do uso de outras 2 estruturas de dados (Lista e Árvore). Para a implementação da Lista foi utilizado o “ArrayList”, classe essa que implementa a interface “List”. Já para a implementação da Árvore, foi utilizada a “TreeSet”, classe essa que implementa a interface “NavigableSet”, que por sua vez implementa a interface “SortedSet”.



(Imagem 01 - Diagrama com heranças da Interface Collection)

Então foram aplicadas as seguintes implementações no código:

- Ordenação pelo nomes dos lugares (campo name) em ordem alfabética, a qual foi realizada com o “TreeSet”, tendo se mostrado muito mais eficiente que os algoritmos utilizados anteriormente.
- Ordenação pelos preços (campo price) dos lugares do menor para o maior preço, a qual foi realizada com o auxílio do ArrayList no SelectionSort, implementação essa que diminuiu significativamente a quantidade de linhas do algoritmo de ordenação.

Cada algoritmo foi testado com a mesma base de dados e também nas mesmas condições de ambiente. Quanto às ferramentas utilizadas, a implementação foi feita a partir da linguagem JAVA.

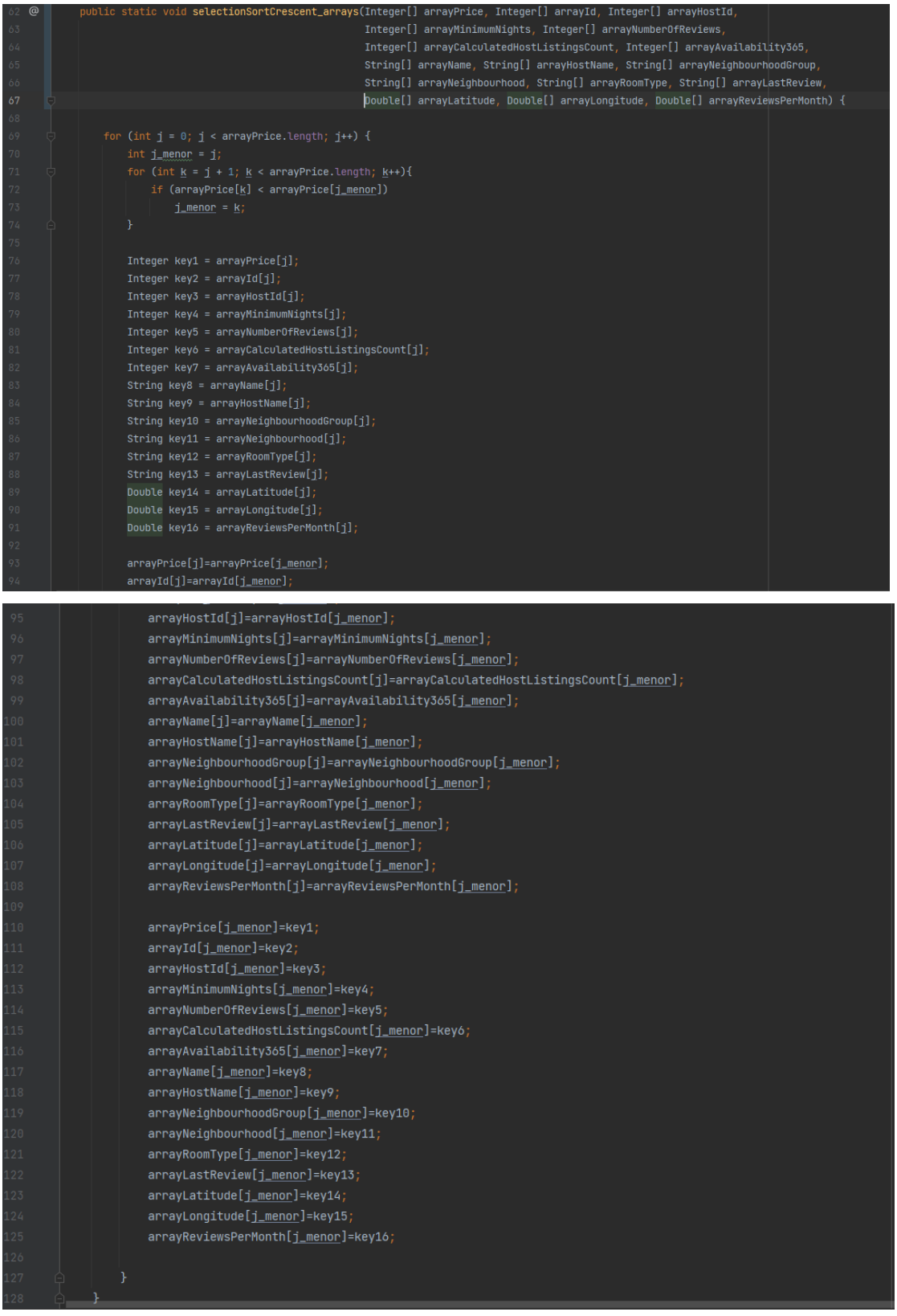
2. Descrição geral sobre o método utilizado

- Implementação da Lista (ArrayList):

  Para a implementação da Lista, foi necessário analisar em qual algoritmo seria mais interessante realizar tal implementação. Após a primeira análise, foi possível notar que a forma mais interessante de aplicar ela seria através do Selection Sort, que usa como princípio duas trocas principais, trocas essas que podemos realizar com o uso da função “swap”.
  A primeira coisa realizada foi alterar os parâmetros do método de ordenação, que anteriormente eram diversos arrays, cada um contendo um atributo diferente, porém após essa implementação, passou a ser apenas a lista de objetos “listings_review_date”, que contém todos os objetos fornecidos pelo dataset.

  Segue abaixo a discrepante diferença na quantidade de linhas de código utilizadas entre utilizar apenas arrays e utilizar um ArrayList:

- Método de Ordenação Selection Sort **sem** a implementação do ArrayList:



```java
public static void selectionSortCrescent_arrays(Integer[] arrayPrice, Integer[] arrayId, Integer[] arrayHostId,
                                                Integer[] arrayMinimumNights, Integer[] arrayNumberOfReviews,
                                                Integer[] arrayCalculatedHostListingsCount, Integer[] arrayAvailability365,
                                                String[] arrayName, String[] arrayHostName, String[] arrayNeighbourhoodGroup,
                                                String[] arrayNeighbourhood, String[] arrayRoomType, String[] arrayLastReview,
                                                Double[] arrayLatitude, Double[] arrayLongitude, Double[] arrayReviewsPerMonth) {

    for (int j = 0; j < arrayPrice.length; j++) {
        int j_menor = j;
        for (int k = j + 1; k < arrayPrice.length; k++){
            if (arrayPrice[k] < arrayPrice[j_menor])
                j_menor = k;
        }

        Integer key1 = arrayPrice[j];
        Integer key2 = arrayId[j];
        Integer key3 = arrayHostId[j];
        Integer key4 = arrayMinimumNights[j];
        Integer key5 = arrayNumberOfReviews[j];
        Integer key6 = arrayCalculatedHostListingsCount[j];
        Integer key7 = arrayAvailability365[j];
        String key8 = arrayName[j];
        String key9 = arrayHostName[j];
        String key10 = arrayNeighbourhoodGroup[j];
        String key11 = arrayNeighbourhood[j];
        String key12 = arrayRoomType[j];
        String key13 = arrayLastReview[j];
        Double key14 = arrayLatitude[j];
        Double key15 = arrayLongitude[j];
        Double key16 = arrayReviewsPerMonth[j];

        arrayPrice[j]=arrayPrice[j_menor];
        arrayId[j]=arrayId[j_menor];
        arrayHostId[j]=arrayHostId[j_menor];
        arrayMinimumNights[j]=arrayMinimumNights[j_menor];
        arrayNumberOfReviews[j]=arrayNumberOfReviews[j_menor];
        arrayCalculatedHostListingsCount[j]=arrayCalculatedHostListingsCount[j_menor];
        arrayAvailability365[j]=arrayAvailability365[j_menor];
        arrayName[j]=arrayName[j_menor];
        arrayHostName[j]=arrayHostName[j_menor];
        arrayNeighbourhoodGroup[j]=arrayNeighbourhoodGroup[j_menor];
        arrayNeighbourhood[j]=arrayNeighbourhood[j_menor];
        arrayRoomType[j]=arrayRoomType[j_menor];
        arrayLastReview[j]=arrayLastReview[j_menor];
        arrayLatitude[j]=arrayLatitude[j_menor];
        arrayLongitude[j]=arrayLongitude[j_menor];
        arrayReviewsPerMonth[j]=arrayReviewsPerMonth[j_menor];

        arrayPrice[j_menor]=key1;
        arrayId[j_menor]=key2;
        arrayHostId[j_menor]=key3;
        arrayMinimumNights[j_menor]=key4;
        arrayNumberOfReviews[j_menor]=key5;
        arrayCalculatedHostListingsCount[j_menor]=key6;
        arrayAvailability365[j_menor]=key7;
        arrayName[j_menor]=key8;
        arrayHostName[j_menor]=key9;
        arrayNeighbourhoodGroup[j_menor]=key10;
        arrayNeighbourhood[j_menor]=key11;
        arrayRoomType[j_menor]=key12;
        arrayLastReview[j_menor]=key13;
        arrayLatitude[j_menor]=key14;
        arrayLongitude[j_menor]=key15;
        arrayReviewsPerMonth[j_menor]=key16;

    }
}
```

(Imagem 02 - Selection Sort com arrays apenas)

- Método de Ordenação Selection Sort **com** a implementação do ArrayList:

```java
public static void selectionSortCrescent_List(List<AirBnbListings> listings_review_date) {

    for (int j = 0; j < listings_review_date.size(); j++) {
        int j_menor = j;
        for (int k = j + 1; k < listings_review_date.size(); k++){
            if (listings_review_date.get(k).getPrice() < listings_review_date.get(j_menor).getPrice())
                j_menor = k;
        }

        int key = listings_review_date.get(j).getPrice();
        Collections.swap(listings_review_date, j, j_menor);
        Collections.swap(listings_review_date, j_menor, key);
    }
}

```

(Imagem 03 - Selection Sort com ArrayList)

● Implementação da Árvore (TreeSet)

Para a implementação da TreeSet, em primeira instância foi necessário utilizar a interface “NavigableSet” para o “AirBnbListings” que vai implementar um “TreeSet”. Porém uma condição necessária a ser cumprida para que um elemento possa ser inserido em um “TreeSet” é que ele implemente a interface “Comparable”. Para tanto, foi realizado tal procedimento, como é demonstrado na imagem a seguir:

```java
class AirBnbListingsComparator implements Comparator<AirBnbListings> {

    @Override
    public int compare(AirBnbListings airBnbListings1, AirBnbListings airBnbListings2) {
        return airBnbListings1.getName().compareTo(airBnbListings2.getName());
    }
}
```

(Imagem 04 - Implementação da interface Comparable)

```java
        NavigableSet<AirBnbListings> airBnbListingsNavigableSet = new TreeSet<>(new AirBnbListingsComparator()); //Arvore
        airBnbListingsNavigableSet.addAll(listings_review_date);
```

(Imagem 05 - Inserção dos elementos na TreeSet)

Como está representado nas imagens, a classe compara dois apartamentos (AirBnbListings) utilizando o nome como parâmetro. A vantagem de utilizar o TreeSet diz respeito à inserção dos elementos já ser de forma ordenada, ou seja, com poucas linhas de código, alterando apenas o referencial da ordenação, é possível realizar o procedimento de ordenação com muita eficiência.

Descrição do ambiente de testes

- Processador AMD Ryzen 5 1600x (6 núcleos e 12 threads)
- Placa de vídeo NVIDEA GTX 1050TI 4GB GDDR5
- Memória RAM 16GB DDR4 2800MHz (2x8)
- Sistema Operacional Windows 10 Pro

3. Resultados e Análise

● Utilização da Lista (ArrayList):

Para a primeira análise, será comparada a utilização do ArrayList para armazenar os objetos referentes aos AirBnbListings e ordenar todos os objetos com o Selection Sort. Retomando à versão anterior do código, é possível comparar precisamente a diferença referente à quantidade de linhas de código necessárias para realizar a ordenação.

```
62 @    public static void selectionSortCrescent_arrays(Integer[] arrayPrice, Integer[] arrayId, Integer[] arrayHostId,
63                                                     Integer[] arrayMinimumNights, Integer[] arrayNumberOfReviews,
64                                                     Integer[] arrayCalculatedHostListingsCount, Integer[] arrayAvailability365,
65                                                     String[] arrayName, String[] arrayHostName, String[] arrayNeighbourhoodGroup,
66                                                     String[] arrayNeighbourhood, String[] arrayRoomType, String[] arrayLastReview,
67                                                     Double[] arrayLatitude, Double[] arrayLongitude, Double[] arrayReviewsPerMonth) {
68
69          for (int j = 0; j < arrayPrice.length; j++) {
70              int j_menor = j;
71              for (int k = j + 1; k < arrayPrice.length; k++){
72                  if (arrayPrice[k] < arrayPrice[j_menor])
73                      j_menor = k;
74              }
75
76              Integer key1 = arrayPrice[j];
77              Integer key2 = arrayId[j];
78              Integer key3 = arrayHostId[j];
79              Integer key4 = arrayMinimumNights[j];
80              Integer key5 = arrayNumberOfReviews[j];
81              Integer key6 = arrayCalculatedHostListingsCount[j];
82              Integer key7 = arrayAvailability365[j];
83              String key8 = arrayName[j];
84              String key9 = arrayHostName[j];
85              String key10 = arrayNeighbourhoodGroup[j];
86              String key11 = arrayNeighbourhood[j];
87              String key12 = arrayRoomType[j];
88              String key13 = arrayLastReview[j];
89              Double key14 = arrayLatitude[j];
90              Double key15 = arrayLongitude[j];
91              Double key16 = arrayReviewsPerMonth[j];
92
93              arrayPrice[j]=arrayPrice[j_menor];
94              arrayId[j]=arrayId[j_menor];
```

```
                arrayHostId[j]=arrayHostId[j_menor];
                arrayMinimumNights[j]=arrayMinimumNights[j_menor];
                arrayNumberOfReviews[j]=arrayNumberOfReviews[j_menor];
                arrayCalculatedHostListingsCount[j]=arrayCalculatedHostListingsCount[j_menor];
                arrayAvailability365[j]=arrayAvailability365[j_menor];
                arrayName[j]=arrayName[j_menor];
                arrayHostName[j]=arrayHostName[j_menor];
                arrayNeighbourhoodGroup[j]=arrayNeighbourhoodGroup[j_menor];
                arrayNeighbourhood[j]=arrayNeighbourhood[j_menor];
                arrayRoomType[j]=arrayRoomType[j_menor];
                arrayLastReview[j]=arrayLastReview[j_menor];
                arrayLatitude[j]=arrayLatitude[j_menor];
                arrayLongitude[j]=arrayLongitude[j_menor];
                arrayReviewsPerMonth[j]=arrayReviewsPerMonth[j_menor];

                arrayPrice[j_menor]=key1;
                arrayId[j_menor]=key2;
                arrayHostId[j_menor]=key3;
                arrayMinimumNights[j_menor]=key4;
                arrayNumberOfReviews[j_menor]=key5;
                arrayCalculatedHostListingsCount[j_menor]=key6;
                arrayAvailability365[j_menor]=key7;
                arrayName[j_menor]=key8;
                arrayHostName[j_menor]=key9;
                arrayNeighbourhoodGroup[j_menor]=key10;
                arrayNeighbourhood[j_menor]=key11;
                arrayRoomType[j_menor]=key12;
                arrayLastReview[j_menor]=key13;
                arrayLatitude[j_menor]=key14;
                arrayLongitude[j_menor]=key15;
                arrayReviewsPerMonth[j_menor]=key16;

            }
        }
```

(Código 01 - Selection Sort com arrays)

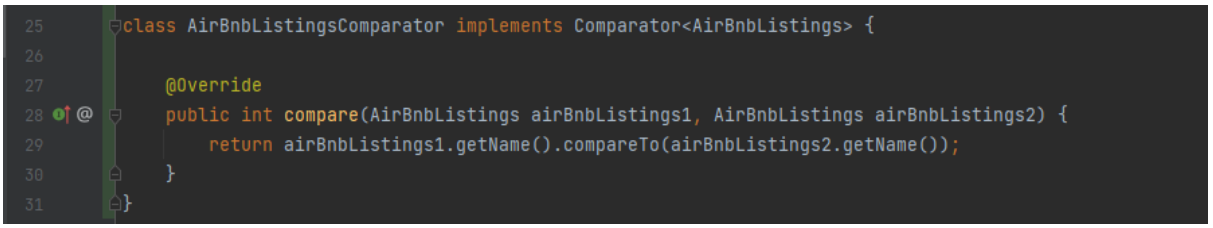
Já observando o código refatorado com o acréscimo do ArrayList, podemos observar que a quantidade de linhas diminuiu significativamente -algo em torno de 52 linhas de um total de 66-. Segue abaixo o novo método de ordenação com a utilização do ArrayList:

```
    public static void selectionSortCrescent_List(List<AirBnbListings> listings_review_date) {

        for (int j = 0; j < listings_review_date.size(); j++) {
            int j_menor = j;
            for (int k = j + 1; k < listings_review_date.size(); k++){
                if (listings_review_date.get(k).getPrice() < listings_review_date.get(j_menor).getPrice())
                    j_menor = k;
            }

            int key = listings_review_date.get(j).getPrice();
            Collections.swap(listings_review_date, j, j_menor);
            Collections.swap(listings_review_date, j_menor, key);
        }
    }

```

(Código 02 - Selection Sort com ArrayList)

- Utilização da Árvore (TreeSet):

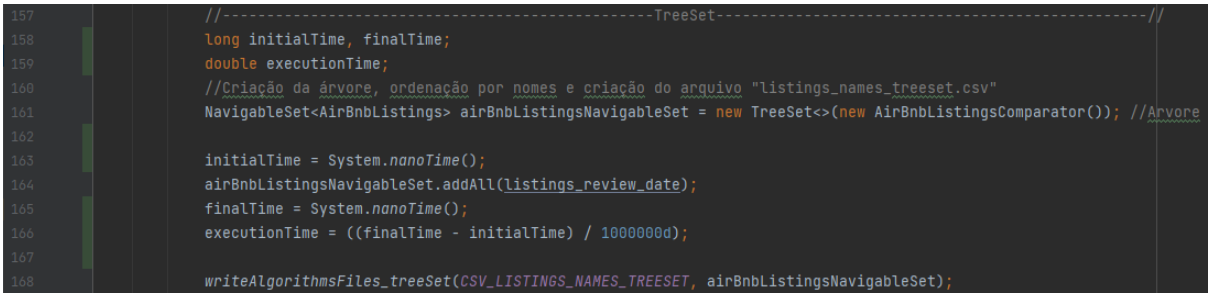
Já a segunda análise diz respeito à utilização da TreeSet, que permitiu com que a ordenação dos elementos fosse muito mais simples e eficiente, se tornando um dos métodos mais eficientes dentre todos já apresentados anteriormente. O acréscimo da estrutura TreeSet permitiu com que fosse realizada a ordenação dos objetos apenas com a implementação da interface “Comparable”, como é apresentado a seguir:



```
25  class AirBnbListingsComparator implements Comparator<AirBnbListings> {
26
27      @Override
28  @   public int compare(AirBnbListings airBnbListings1, AirBnbListings airBnbListings2) {
29          return airBnbListings1.getName().compareTo(airBnbListings2.getName());
30      }
31  }
```

(Código 03 - Implementação da Interface Comparable)

A próxima análise a ser realizada é referente ao método utilizado para realizar o procedimento de ordenação e inserção dos elementos na TreeSet, que ocorre de forma simultânea. Para tanto, bastou com que a árvore fosse instanciada com a utilização da classe “AirBnbListingsComparator”, que foi apresentada anteriormente e que todos os elementos presentes no ArrayList com todos os objetos “listings_review_date” fossem adicionados na TreeSet, e apenas com esses processos, a árvore com todos os elementos do ArrayList “listings_review_date” já se encontraria preenchida e ordenada.



```
157         //-----------------------------------------------TreeSet-----------------------------------------------//
158         long initialTime, finalTime;
159         double executionTime;
160         //Criação da árvore, ordenação por nomes e criação do arquivo "listings_names_treeset.csv"
161         NavigableSet<AirBnbListings> airBnbListingsNavigableSet = new TreeSet<>(new AirBnbListingsComparator()); //Arvore
162
163         initialTime = System.nanoTime();
164         airBnbListingsNavigableSet.addAll(listings_review_date);
165         finalTime = System.nanoTime();
166         executionTime = ((finalTime - initialTime) / 1000000d);
167
168         writeAlgorithmsFiles_treeSet(CSV_LISTINGS_NAMES_TREESET, airBnbListingsNavigableSet);
```

(Código 04 - Utilização da TreeSet)

Ainda vale acrescentar que o tempo de execução com a utilização do TreeSet é muito eficiente, necessitando de apenas **26.6826ms** para a realização de todo o processo.