

**ALGORITMOS DE ORDENAÇÃO**

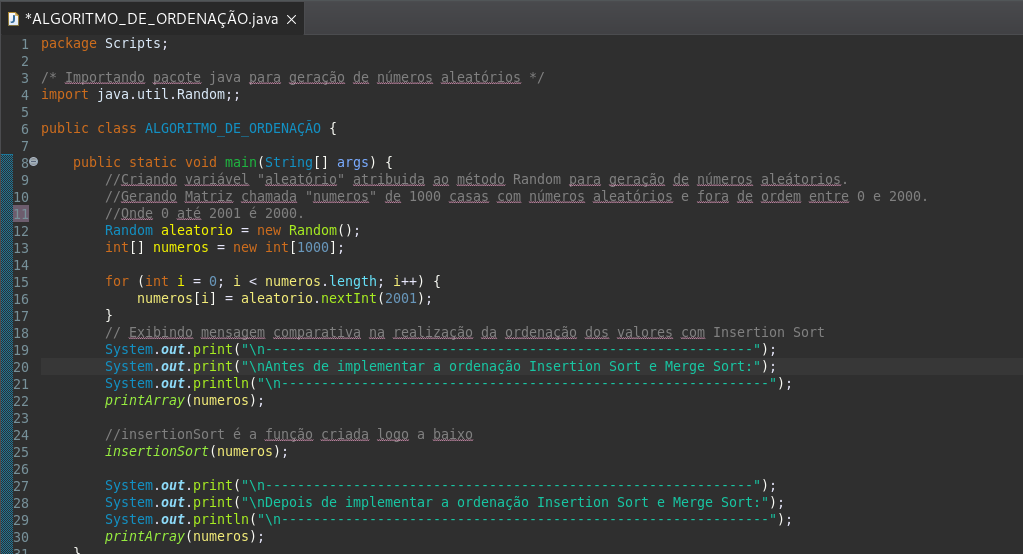
**ESTRUTURAS DE DADOS NÃO LINEARES**

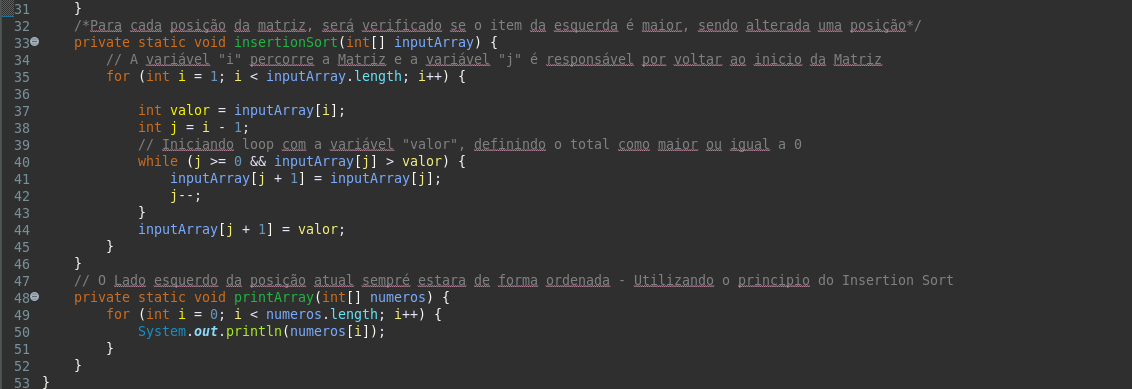
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Nome**: Erick Barbara de Araujo **RA**: 336356   
**Nome**: Thamara da Silva Sousa **RA**: 309814  
**Nome**: Vitor Da Silva Bezerra **RA**: 336459   
**Nome**: Wallace Santos Ribeiro **RA**: 309767

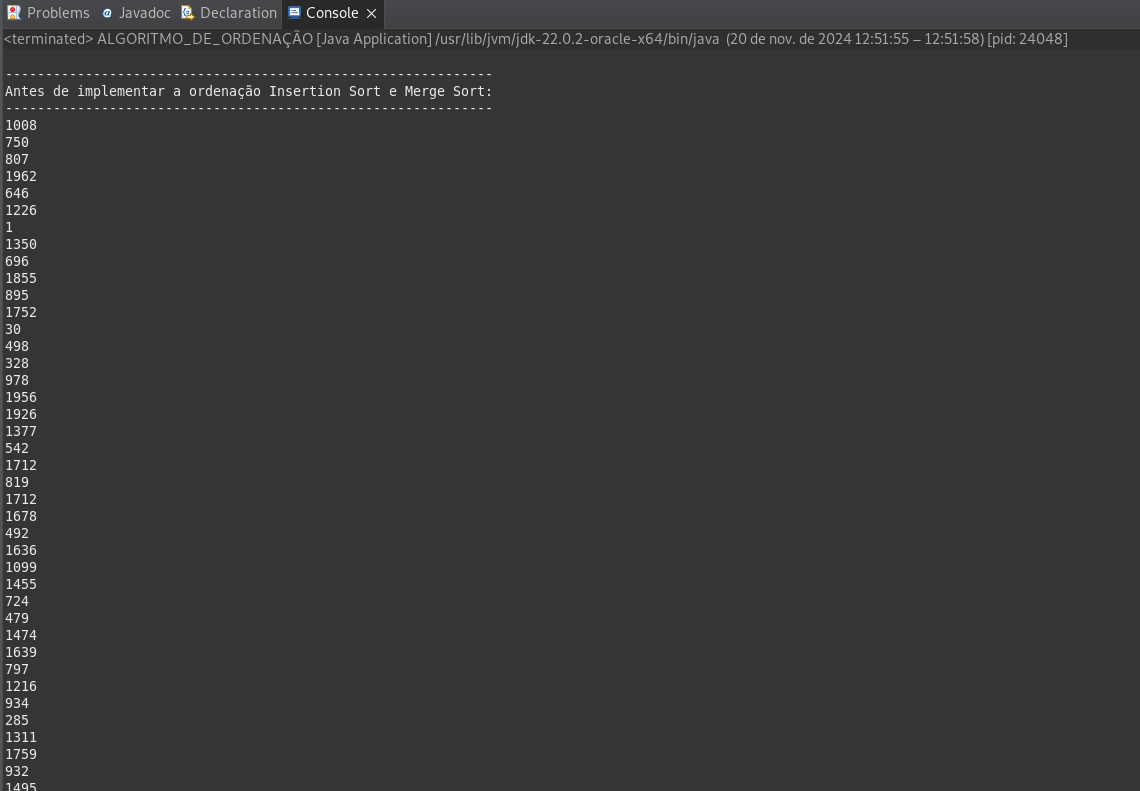
A atividade semestral consiste em:  
  
1) gerar 1000 números aleatórios e fora de ordem entre 0 e 1999.  
2) escolher um dos algoritmos de ordenação sequencial (Bubble Sort ou Insertion Sort ou Selection Sort).  
3) ordenar os números gerados através de um programa escrito na linguagem escolhida pelo grupo, a partir do algoritmo escolhido em 3), desde que seja possível gerar na saída o tempo de execução da ordenação.  
4) escolher um dos algoritmos de ordenação: Merge Sort ou Quick Sort.  
5) Ordenar os números gerados através de um programa escrito na linguagem escolhida pelo grupo, a partir do algoritmo escolhido em 4), desde que seja possível gerar na saída o tempo de execução da ordenação.  
6) comparar os tempos de execução obtidos em 3 e 5 e discutir os resultados obtidos.  
  
O grupo deverá entregar o trabalho em formato Word com a introdução, onde explicará os algoritmos escolhidos, o código com fonte monoespaçada e as telas de saída do programa.  
Antes de executar o programa, fechar o máximo de tarefas possíveis no computador. Executar os itens 3 e 5 no mesmo computador.

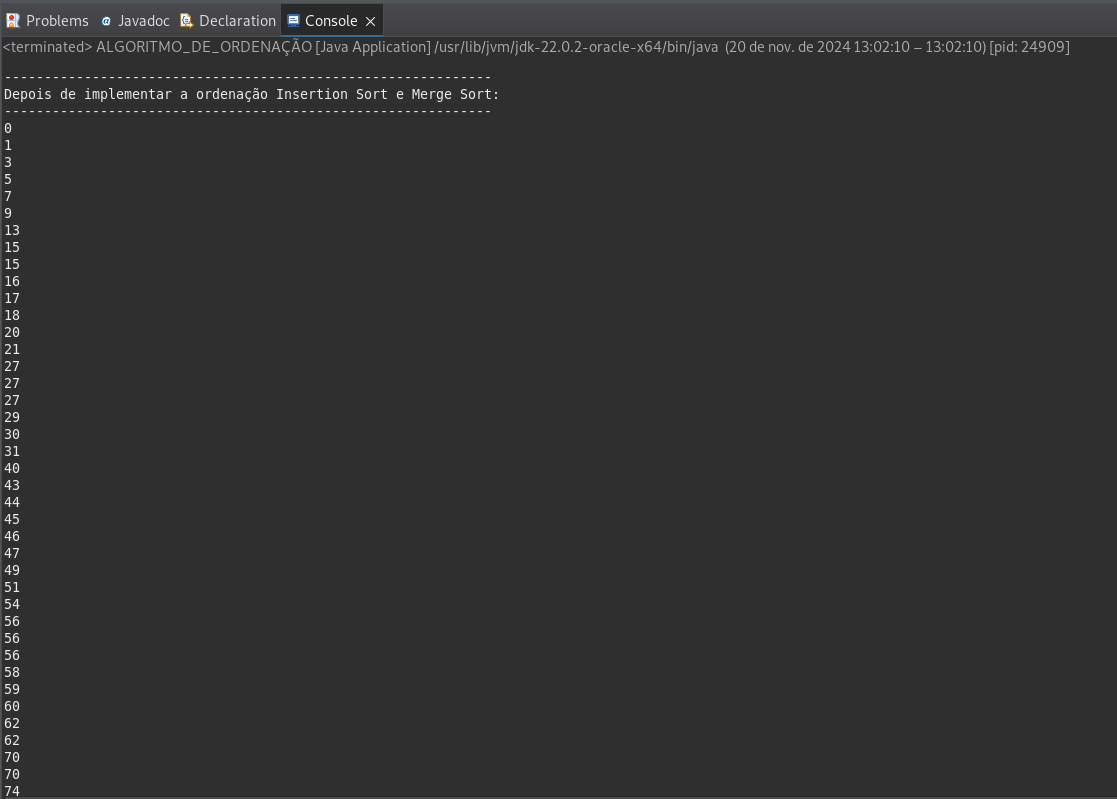
* Utilizando a Linguagem **Java** através da ferramenta **Eclipse**, foi desenvolvido base do código utilizando o algoritmo **Insertion Sort** e **Merge Sort** para ordenação da sequencia de 1000 números aleatórios através da função **Random**.





* Execução do código:







Fim do código, exibindo a ordenação dos números aleatórios em sequencia até 2000 com Insert Sort e Merge Sort, tempo de execução total: 19 milissegundos.



* Exibição do Código:

package Scripts;

/\* Importando pacote java para geração de números aleatórios \*/

import java.util.Random;;

public class ALGORITMO\_DE\_ORDENAÇÃO {

public static void main(String[] args) {

long tempoInicial = System.*currentTimeMillis*();

//Criando variável "aleatório" atribuida ao método Random para geração de números aleátorios.

//Gerando Matriz chamada "numeros" de 1000 casas com números aleatórios e fora de ordem entre 0 e 2000.

//Onde 0 até 2001 é 2000.

Random aleatorio = new Random();

int[] numeros = new int[1000];

for (int i = 0; i < numeros.length; i++) {

numeros[i] = aleatorio.nextInt(2001);

}

// Exibindo mensagem comparativa na realização da ordenação dos valores com Insertion Sort

System.***out***.print("\n-------------------------------------------------------------");

System.***out***.print("\nAntes de implementar a ordenação Insertion Sort e Merge Sort:");

System.***out***.println("\n-------------------------------------------------------------");

*printArray*(numeros);

//insertionSort é a função criada logo a baixo

*insertionSort*(numeros);

System.***out***.print("\n-------------------------------------------------------------");

System.***out***.print("\nDepois de implementar a ordenação Insertion Sort e Merge Sort:");

System.***out***.println("\n-------------------------------------------------------------");

*printArray*(numeros);

// Registrando tempo de execução

long tempoFinal = System.*currentTimeMillis*();

// Calculando o tempo total de execução

long duracao = tempoFinal - tempoInicial;

System.***out***.println("Tempo de execução: " + duracao + " milissegundos");

}

/\*Para cada posição da matriz, será verificado se o item da esquerda é maior, sendo alterada uma posição\*/

private static void insertionSort(int[] inputArray) {

// A variável "i" percorre a Matriz e a variável "j" é responsável por voltar ao inicio da Matriz

for (int i = 1; i < inputArray.length; i++) {

int valor = inputArray[i];

int j = i - 1;

// Iniciando loop com a variável "valor", definindo o total como maior ou igual a 0

while (j >= 0 && inputArray[j] > valor) {

inputArray[j + 1] = inputArray[j];

j--;

}

inputArray[j + 1] = valor;

}

}

// O Lado esquerdo da posição atual sempré estara de forma ordenada - Utilizando o principio do Insertion Sort

private static void printArray(int[] numeros) {

for (int i = 0; i < numeros.length; i++) {

System.***out***.println(numeros[i]);

}

}

}