**//Classe principal APS**

**import** java.util.Scanner;

**import** java.util.Random;

**public** **class** Aps {

**public** **static** Scanner *ler*;

**public** **static** **void** main(String[] args) {

**int** op1, op, i;

// instanciando o método Scanner para leitura de dados digitados pelo usuário

*ler* = **new** Scanner(System.***in***);

//instanciando objeto para geração de números aleatórios

Random n = **new** Random();

//Instanciando o objeto Ordenações

Ordenações ord1 = **new** Ordenações();

Ordenações ord2 = **new** Ordenações();

Ordenações ord3 = **new** Ordenações();

//v vai receber o numero que irá ser o tamanho do vetor

**int** v, ret;

**do**{

//Menu de entrada

System.***out***.println("1 - Ler arquivo externo?");

System.***out***.println("2 - Preencher vetor?");

op1 = *ler*.nextInt();

**switch**(op1){

**case** 1:

//instaciando a classe Reader para onde está os comandos para leitura de arquivos externo.

Reader ae = **new** Reader();

// método Leitura sendo chamado na Main

ae.leitura();

**break**;

**case** 2:

//Usuário vai definir o tamanho do vetor, onde v vai receber o que o usuário irá atribuir

System.***out***.println("Digite o números de casas no vetor que deseja?");

v = *ler*.nextInt();

**int** vetor[] = **new** **int**[v];

//menu de pré-introdução de dados para o vetor

System.***out***.println("1 - Deseja digitar os números para Ordenar?");

System.***out***.println("2 - Deseja que o números sejam gerados para Ordenação?");

op = *ler*.nextInt();

**switch**(op){

**case** 1:

**for** (i = 0; i < vetor.length; i++) {

System.***out***.print("Digite o " + (i + 1) + "º valor = ");

vetor[i] = *ler*.nextInt();

}

ord1.bubblesort(vetor.clone());

ord2.apresentaQuick(vetor.clone());

ord3.Cocktail(vetor.clone());

**break**;

**case** 2:

**for** (i = 0; i < vetor.length; i++) {

vetor[i] = n.nextInt();

}

ord1.bubblesort(vetor.clone());

ord2.apresentaQuick(vetor.clone());

ord3.Cocktail(vetor.clone());

**break**;

**default**:

System.***out***.print("Opção inexistente");

**break**;

}

**break**;

}

System.***out***.println("Retornar ao Menu inicial");

ret = *ler*.nextInt();

}

**while**(ret == 1);

}

}

// Classe Ordenações

**public** **class** Ordenações {

**long** tinicio, tfim;

**public** **void** bubblesort(**int** vet[]){

//teste para ver se o vetor recebido está desordenado.

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

System.***out***.print(vet[i] + " ");

}

**int** tam = vet.length;

**int** i, j, aux;

tinicio = System.*nanoTime*();

**for**(i = 0; i < (tam - 1); i++){

**for**(j = 1; j < tam; j++){

**if**(vet[j] < vet[j - 1]){

aux = vet[j];

vet[j] = vet[j - 1];

vet[j - 1] = aux;

}

}

}

tfim = System.*nanoTime*();

// 1 milisegundo é 1000000 nanosegundos

System.***out***.println("Tempo = " + ((tfim - tinicio) / 1000000.0) + " ms");

**for**(i = 0; i < vet.length; i++){

System.***out***.print(vet[i] + " ");

}

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

}

//criação de um método que apresente o Quick, pois com as Recursões dentro do Quicksort o apresentação iria se repetir várias vezes

**public** **void** apresentaQuick(**int** vet[]){

//teste para ver se o vetor recebido está desordenado.

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

System.***out***.print(vet[i] + " ");

}

tinicio = System.*nanoTime*();

Quicksort(vet, 0, vet.length - 1);

tfim = System.*nanoTime*();

System.***out***.println("Tempo = " + ((tfim - tinicio) / 1000000.0) + " ms ");

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

System.***out***.print(vet[i] + " ");

}

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

}

**public** **void** Quicksort(**int** v[], **int** esq, **int** dir){

**int** pivo;

**if**(esq < dir){

pivo = *Separar*(v, esq, dir);

Quicksort(v, esq, pivo - 1);

Quicksort(v, pivo + 1, dir);

}

}

**public** **static** **int** Separar(**int** vet[], **int** esq, **int** dir){

**int** c = vet[esq], i = esq + 1, j = dir, troca;

**while**(i <= j){

**if**(vet[i] <= c)

i++;

**else** **if**(c < vet[j])

j--;

**else**{

troca = vet[i];

vet[i] = vet[j];

vet[j] = troca;

}

}

i = j + 1;

vet[esq] = vet[j];

vet[j] = c;

**return** j;

}

**public** **void** Cocktail(**int** vetor[]){

tinicio = System.*nanoTime*();

**int** esq = 0, dir = vetor.length - 1, cond = 0, troca, i;

**while**(cond == 0 && esq < dir){

cond++;

**for**(i = esq; i < dir; i++){

**if**(vetor[i] > vetor[i + 1]){

troca = vetor[i];

vetor[i] = vetor[i + 1];

vetor[i + 1] = troca;

cond = 0;

}

}

dir = dir - 1;

**for**(i = dir; i > esq; i--){

**if**(vetor[i] < vetor[i - 1]){

troca = vetor[i];

vetor[i] = vetor[i - 1];

vetor[i - 1] = troca;

cond = 0;

}

}

esq = esq + 1;

}

tfim = System.*nanoTime*();

System.***out***.println("Tempo = " + ((tfim - tinicio) / 1000000.0) + " ms");

**for**(i = 0; i < vetor.length; i++){

System.***out***.print(vetor[i] + " ");

}

}

}

//classe Reader

**import** java.nio.file.Files;

//biblioteca de manipulação de arquivos (Path e Paths)

**import** java.nio.file.Path;

**import** java.nio.file.Paths;

**public** **class** Reader {

**public** **void** leitura() {

// **TODO** Auto-generated method stub

// Objeto Path armazena na variável "pasta", todo o caminho hierárquico que a fora especificado na Paths.

Path pasta = Paths.*get*("C:/Users/juniorw2/Documents/Arquivo externo/Arquivo.txt");

// condição caso o caminho esteja errado.

**try**{

//"File.readAllBytes irá ler o arquivo especifica na Path Pasta, onde se encontra o caminho especificado. E irá retornar um Array de bite, onde estará o texto

**byte**[] text = Files.*readAllBytes*(pasta);

//para interpretação do arquivo armazenado no Array "text" é necessário uma String, que fora instanciada para a String "ler" interpretar o que contém na "text"

String ler = **new** String(text);

//Converter string para char

**char**[] revers = ler.toCharArray();

**int** tamanho = ler.length();

**int** vetor[] = **new** **int** [tamanho];

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

// System.out.print(revers[i]);

//}

**for**(**int** j = 0; j < vetor.length; j++){

//convert Char para Int e joga no vetor de inteiros

vetor[j] = (**int**) revers[j];

}

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

System.***out***.print(vetor[i] + " ");

}

}

**catch**(Exception erro){

}

}

}