TP2B – ORDENADORA - CASOS DE PRUEBA

**PARA TODOS LOS CASOS DE PRUEBA SE GENERARON CLASES DE TEST, CON JUNIT (ver package “test”).**

**01 – ORDENAR NUMEROS ENTEROS**

Recibe un array de elementos enteros para ordenar.

|  |  |
| --- | --- |
| Entrada | {1, 2, 3 , 4, -1 , 10, 3, 9} |
| Salida Ordenada | {-1, 1, 2, 3, 3, 4, 9, 10} |

**02 – ORDENAR NUMEROS DOUBLE CASI IGUALES**

Recibe un array de elementos double los cuales difieren por decimales

|  |  |
| --- | --- |
| Entrada | {-0.99, 0.99, 1.00, 0.999998, 0.089, 1.01, 0.09999999, -0.9998} |
| Salida Ordenada | {-0.9998, -0.99, 0.089, 0.09999999, 0.99, 0.999998, 1.0, 1.01} |

**03 – ORDENAR NUMEROS DOUBLE CHICOS**

Recibe un array de elementos double en notación científica con exponentes negativos.

|  |  |
| --- | --- |
| Entrada | {10E-3, 10E+0, 10E-7, 10E-5, 10E-4, 10E-6, 10E-2, 10E-1} |
| Salida Ordenada | {10E-7, 10E-6, 10E-5, 10E-4, 10E-3, 10E-2, 10E-1, 10E+0} |

**04 – ORDENAR NUMEROS DOUBLE GRANDES**

Recibe un array de elementos double en notación científica con exponentes positivos elevados.

|  |  |
| --- | --- |
| Entrada | {8E+100, 1E+100, 1.28E+102, 3.2E+101, 1.6E+101, 6.4E+101, 4E+100, 2E+100} |
| Salida Ordenada | {1E+100, 2E+100, 4E+100, 8E+100, 1.6E+101, 3.2E+101, 6.4E+101, 1.28E+102} |

**05– ORDENAR PALABRAS**

Recibe un array de elementos String.

|  |  |
| --- | --- |
| Entrada | {"Hola","Chau","Ala","Aca","Abap","Zorro","Banco"} |
| Salida Ordenada | {"Abap","Aca","Ala","Banco","Chau","Hola","Zorro"} |

**06 – ORDENAR “CIRCULOS”**

Recibe un array de elementos “CIRCULO”, clase desarrollada durante el taller de los días sábados. Los mismos se construyen a partir de un “radio” recibido. Y se compararan las áreas de las figuras para determinar quién es más grande. La clase “Circulo” implementa la interfaz “Comparable”.

|  |  |
| --- | --- |
| Entrada | { Circulo(5), Circulo(5.01), Circulo(4.99), Circulo(2), Circulo(4), Circulo(1), Circulo(10)} |
| Salida Ordenada | { Circulo(1), Circulo(2), Circulo(4), Circulo(4.99), Circulo(5), Circulo(5.01), Circulo(10)} |

**07 – ORDENAR “NUMEROS COMPLEJOS”**

Recibe un array de elementos “COMPLEJO”, clase desarrollada durante las clases. Los mismos se construyen a partir de un parte real y otra imaginaria. Y se compararan los módulos de los mismos para determinar quién es mayor. La clase “Complejo” implementa la interfaz “Comparable”.

|  |  |
| --- | --- |
| Entrada | {Complejo(5,5), Complejo(1.2,1.5), Complejo(1,2),Complejo(3,3),Complejo(1.19,1.489),Complejo(2,2)} |
| Salida Ordenada | {Complejo(1.19,1.489), Complejo(1.2,1.5), Complejo(1,2),Complejo(2,2),Complejo(3,3),Complejo(5,5)} |

**08 – FATIGA, ORDENAR ENTEROS DE DIMENSION 10.000**

Recibe un array de 10.000 elementos enteros, con el fin de probar la robustez del algoritmo. El array contiene números consecutivos del 1 al 10000. Y lo recibe en orden inverso.

|  |  |
| --- | --- |
| Entrada | IN/08\_Fatiga10mil\_enteros.in |
| Salida Ordenada | OUT/08\_Fatiga10mil\_enteros.out |

**09 – ORDENAR CON NUMEROS REPETIDOS**

Recibe un array de elementos enteros para ordenar, con número repetidos.

|  |  |
| --- | --- |
| Entrada | {1,1,5,1,1} |
| Salida Ordenada | {1,1,1,1,5} |

**10 – UN SOLO NUMERO**

Recibe un array de elementos enteros con un único número.

|  |  |
| --- | --- |
| Entrada | {5} |
| Salida Ordenada | {5} |

**11 – NUMEROS Y PALABRAS**

Recibe un array de elementos String para ordenar, con letras y números.

|  |  |
| --- | --- |
| Entrada | {“hola”,”5”,”perro”,”chau”,”2”} |
| Salida Ordenada | {“2”, “5”, “chau”, “hola”, “perro”} |

**12 – ORDEN INVERSO**

Recibe un array de elementos enteros para ordenar, en orden inverso.

|  |  |
| --- | --- |
| Entrada | {10, 8, 5, 4, 3, 1, -4} |
| Salida Ordenada | {-4, 1, 3, 4, 5, 8, 10} |

**12 – ORDENADOS**

Recibe un array de elementos enteros para ordenar, en orden.

|  |  |
| --- | --- |
| Entrada | {-3, 0, 1, 2, 3, 4, 5, 6, 15} |
| Salida Ordenada | {-3, 0, 1, 2, 3, 4, 5, 6, 15} |