CLASES *String* y *StringBuffer*

1. Crea un programa en Java que solicite al usuario la introducción de una cadena de caracteres y devuelva esta cadena invertida. Haz dos versiones, una con String y otra con StringBuffer.
2. Crea un programa que reciba una cadena de caracteres y la devuelva invertida con efecto espejo, este es, se concatena a la palabra original su inversa, compartiendo la última letra, que hará de espejo, por lo que la palabra obtenida se lee igual hacia adelante que hacia atrás. Por ejemplo, al introducir "teclado" devolverá "tecladodalcet" y al introducir "hola" devolverá "holaloh". Haz dos versiones, una con String y otra con StringBuffer.
3. Diseña un programa en Java que solicite al usuario una cadena en la que buscará y otra que será la cadena buscada. El programa indicará cuántas veces aparece la segunda cadena en la primera.
4. Crea un programa en Java que solicite al usuario dos cadenas de caracteres y que devuelva la primera cadena, pero transformando en mayúsculas la parte que coincide con la segunda cadena introducida. Por ejemplo, si se introducen las cadenas "Esta es mi amiga Ana" y "amiga" devolverá "Esta es mi AMIGA Ana".
5. Diseña un programa en Java que solicite al usuario una cadena de caracteres y muestre por pantalla un conteo de cuántas vocales, consonantes y espacios en blanco contiene la cadena introducida.
6. Crea un método que determine si una cadena es un palíndromo, o sea, se lee igual en los dos sentidos.
7. Investigar StringBuilder. Hacer un ejercicio con esa clase. Por qué es mejor y cuándo se recomienda.

ALGORITMOS DE ORDENACIÓN

1. Realiza un programa que cree un array de 50 posiciones cargado con valores aleatorios. Los valores aleatorios deberán estar entre el 1 y el 100. Una vez cargado el vector, deberá ordenarlo mediante el método de la ***burbuja*** y mostrarlo ordenado por pantalla. Para el método de la burbuja deberás crear una función que reciba un array de números y lo devuelva ordenado.

2. Mejora el método de la ***burbuja*** explicado anteriormente y utiliza una variable a modo de centinela o *flag*, de tal manera que ésta se active cuando hay algún intercambio. En el momento que no haya ningún intercambio, el algoritmo debería parar puesto que el vector ya está ordenado.

3. Realiza un método que tome como parámetros de entrada dos arrays de enteros y devuelva como salida un único array con los dos elementos de los anteriores arrays ordenados de forma ascendente.

4. Realiza un programa que cree un vector de 50 posiciones con número aleatorios entre 1 y 50. Una vez creado el vector, el programa deberá mostrar el mayor, el menor y la media de los valores almacenados en el array.

5. Realiza un programa que cree 100 números aleatorios entre 1 y 1000 y que muestre los 10 mayores.

6. Realiza un programa que cree un vector de 100 posiciones con números aleatorios entre 1 y 100. Una vez creado el vector, el programa deberá ordenarlo y mostrar los números entre 1 y 100 que no han sido almacenados. Ten en cuenta a la hora de buscar un número en un array que no tienes que comparar con todo el array puesto que ya está ordenado.