Ejercicios propuestos.

1. Crea un proyecto que contenga una clase llamada Utils. Dentro de esta clase debemos programar los siguientes métodos estáticos:

    - Diseña una función que reciba una cadena y muestre cada carácter de la misma indicando si es dígito, letra u otro.
    - Diseña una función que reciba un array de cadenas y devuelva otro array en el que figuren las cadenas del primer array que finalizan con un patrón concreto. La función recibe el array y el patrón.
    - Diseña una función que reciba una cadena de palabras separadas por un mismo carácter (punto y coma, coma, etc.) y devuelva un array de String con cada una de esas palabras. La función también debe llevar como parámetro el separador.
    - Diseña una función que reciba un array de enteros y un valor. La función ordenará el array, lo mostrará y devolverá la posición de ese valor en el array o -1 si no lo encuentra.
    - Diseña una función que reciba un array de fechas (java.time.LocalDate) y las muestre ordenadas de forma ascendente.
    - Diseña una función que reciba un array de fechas (java.time.LocalDate), un mes y un año y devuelva cuántas de esas fechas pertenecen a ese mes y año.
    - Diseña una función que reciba un array de horas (java.time.LocalTime) y una hora y devuelva cuantas de las horas del array son anteriores a la pasada como parámetro.
2. Crea **un proyecto Maven** e incluye en el mismo la dependencia a la API JodaTime. Este proyecto debe contener una función que reciba como parámetro un array de fechas (JodaTime) y las muestre en pantalla en formato largo: 30 de septiembre de 2018. Puedes usar estos métodos: getDayOfMonth(), monthOfYear().getAsText() y getYear().
3. En este caso, programaremos un proyecto al que pasaremos en el array de argumentos del método main una lista de años. La aplicación comprobará si esos años son bisiestos o no.
4. **Ordenar por fecha de nacimiento, un array de objetos Persona utilizando la interfaz Comparable.**
    a. Crearemos una clase Persona (id, nombre y fecha de nacimiento). Esta clase debe implementar la interfaz Comparable.
    b. Al implementar esta interfaz debemos sobreescribir el método compareTo. En este método indicamos la forma en que se comparan dos objetos, que es lo tendrá en cuenta el método Arrays.sort().
5. **Ordenar un array de objetos Persona utilizando la interfaz Comparator. Ordenar el array de varias formas: por id, por nombre o por fecha.**
    a. En este caso crearemos la clase Persona igual que en el ejercicio anterior.
    b. Ordenaremos el array mediante sort, pero pasándole un objeto Comparator<Persona>.
    c. En la implementación del método compareTo indicaremos la forma de ordenar.

```java
//ordenar por nombre
Arrays.sort(personas, new Comparator<Persona>() {

    @Override
    public int compare(Persona arg0, Persona arg1) {
        // TODO Auto-generated method stub
        return arg0.getNombre().compareTo(arg1.getNombre());
    }

});
```

6. **Ordenar un array de objetos Persona utilizando la interfaz Comparator. En esta ocasión ordenaremos por nombre y si coinciden por fecha.**
7. Diseña una aplicación Java que:
    a. Solicite al usuario un valor entero “n”, que indique con cuántos números va a trabajar la propia aplicación.
    b. Solicite al usuario estos “n” números que serán de tipo String.
    c. Mostrar los valores en orden descendente según su valor numérico. Ojo, si se teclean dos o más valores idénticos numéricamente, como .1 y 0.1, deben mostrarse en el mismo orden en el que fueron tecleados.
    d. El valor de “n” debe estar comprendido entre 1 y 100.
    e. Cada cantidad pasada, tendrá como máximo 100 dígitos.
    f. La clase BigDecimal puede trabajar con números grandes.

Ejemplos de entrada (verde) y salida (negro).

```
5
12
3
.3
0.3
122
```

```
122
12
3
.3
0.3
```