Tarea 014 - JavaScript- Ejercicio de Ventas

Los resultados de todas las tareas incorporarán, además del código fuente, los comentarios precisos y necesarios para su fácil compresión. No escatimes esfuerzos en comentar el código, es una buena práctica para aprender, además de ser muy útil para modificaciones o reutilizaciones futuras.

Sería muy buena práctica añadir el título y el enunciado del ejercicio (como comentarios) al principio del código fuente.

Añade documentos en formato Word con capturas de la salida por pantalla (al ejecutar la página) si consideras que queda más clara la resolución del ejercicio.

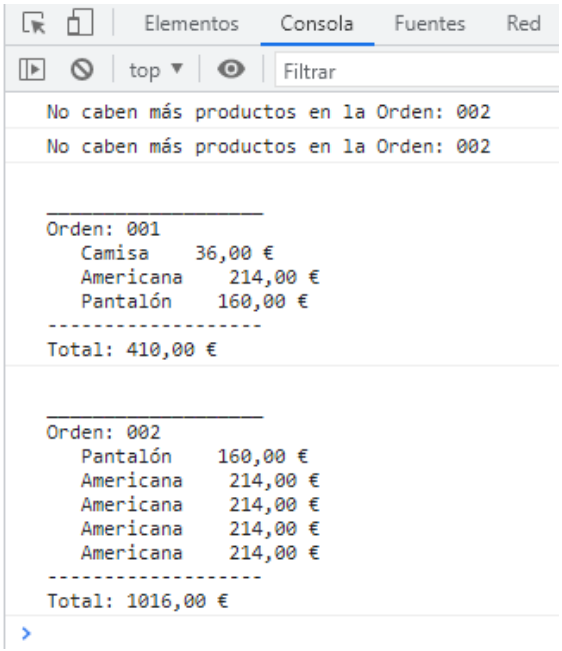
Deberás entregar dos versiones del resultado:

1. **Un único documento en Word sin comprimir**: que contenga todas las líneas de código, y por orden, primero el código **html**, luego el código **css** y por último el código **js**. Este archivo servirá para que el profesor haga anotaciones y/o correcciones a los alumnos.
2. **Los archivos de código fuente**: en un único archivo, ya sea de extensión: **html**, **js**, o **css**. Si precisas entregar varios archivos, comprímelos en un único **zip**. (No admito rar).

El nombre del archivo entregado comenzará por tu nombre seguido por TareaXXX. Ejemplo: **federicoTarea014.zip**

014 Clases con relación de agregación

Se pide crear una serie de clases y objetos que funcionen como las pruebas que se observan en la salida por consola siguiente:



Se trata de un sistema de ventas que consta de:

- Una clase Producto para crear los objetos que se venden.
- Una calase Orden para crear cada ticket de venta.
- Conseguir que se impriman los tikets tal como se muestra arriba.

La salida por consola es la que corresponde con el siguinete código de ejecución:

ventas.js index.html

ventas.js > ...

```
15  > class Producto {···
42    }
43
44  > class Orden {···
86    }
87
88    let p1=new Producto('Camisa', 36,5);
89    let p2=new Producto('Chaqueta', 214,99);
90    let p3=new Producto('Pantalón', 67);
91    //console.log(p1.toString());
92    let o1=new Orden();
93    o1.agregarProducto(p1);
94    o1.agregarProducto(p2);
95    o1.agregarProducto(p3);
96    let orden2=new Orden();
97    p3.precio=160;;
98    orden2.agregarProducto(p3);
99    p2.nombre='Americana';
100   orden2.agregarProducto(p2);
101   orden2.agregarProducto(p2);
102   orden2.agregarProducto(p2);
103   orden2.agregarProducto(p2);
104   orden2.agregarProducto(p2);
105   orden2.agregarProducto(p2);
106   console.log(o1.mostrarOrden());   ... _________ Orden: 001 · Camisa 36,00 € ·
107   console.log(orden2.mostrarOrden());   ... ricana 214,00 € · Americana 214,00 €
```

PROBLEMAS SALIDA TERMINAL CONSOLA DE DEPURACIÓN

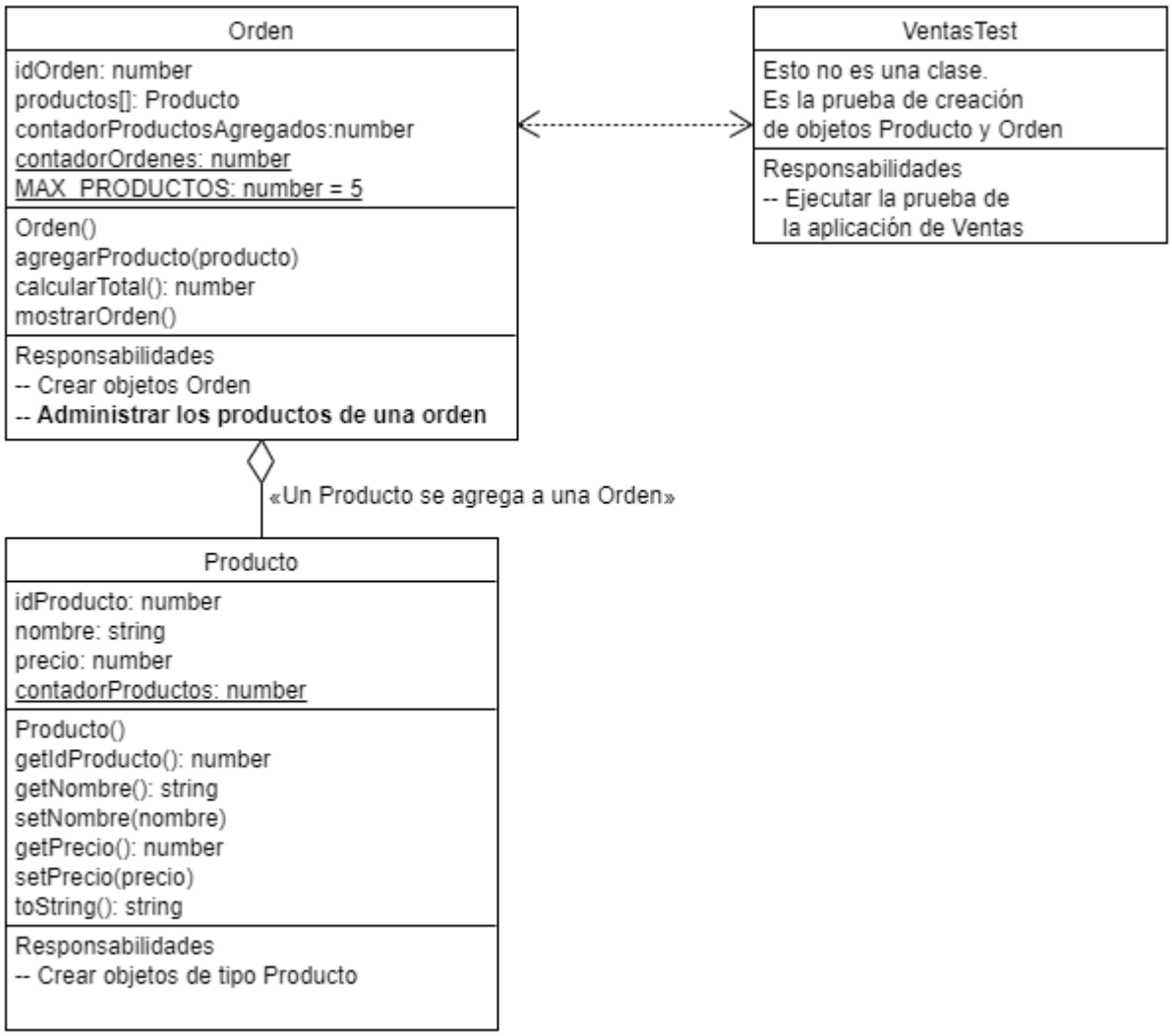
```
____________________
Orden: 001
   · Camisa 36,00 €
   · Americana 214,00 €
   · Pantalón 160,00 €
--------------------
Total: 410,00 €
  at o1.mostrarOrden() ventas.js:106:0

Reveal in value explorer


____________________
Orden: 002
   · Pantalón 160,00 €
   · Americana 214,00 €
```

Debido a que los objetos de la clase **Producto** PUEDEN EXISTIR INDEPENDIENTEMENTE de la clase **Orden**, tenemos una relación de **agregación**.

Las propiedades y métodos a crear se muestran en el siguiente diagrama UML.



Otras consideraciones:

- El número máximo de productos (líneas) en una orden (ticket), establecido como constante de la clase Orden, tendrá el valor 5.
- La impresión del ticket usará el método **toString**() de la clase Producto.
- El número de orden comenzará en 1 (auto incremental) y el formato de impresión será de 3 dígitos fijos.
- Los importes tendrán 2 decimales. El símbolo de moneda y separador de miles serán los del formato local.