JSON

[D.01] Producto JSON

Crear, dentro del archivo ***producto.js***, un JSON para representar información acerca de un **producto** (código de barra, nombre, precio y fecha de vencimiento). Asignarle valores a cada una de sus propiedades.
Validar su buen diseño ingresando en http://jsonviewer.stack.hu/.
Una vez validado el JSON, mostrar todas sus propiedades por consola.

[D.02] Colección de productos JSON

Tomando como punto de partida el ejercicio anterior, diseñar, en el archivo ***productos.js***, una colección de tres elementos de tipo **producto**. Asignarle, a cada elemento, valores para cada una de las propiedades.
Validar su buen diseño ingresando en http://jsonviewer.stack.hu/.
Una vez validado el JSON, mostrar todas las propiedades de todos los productos por consola.

[D.03] Mis productos en la Web

Crear el documento HTML ***productos.html*** y vincularle el archivo *productos.js* realizado en el punto anterior.
Nota: Verificar su funcionamiento en la consola del navegador (F12).

[D.04] Agregando y quitando propiedades

Crear el documento HTML ***productos_alterados.html*** y realizar las siguientes tareas:

1. Vincular el archivo *productos.js* realizado en el punto [D.02].
2. Agregar en la etiqueta <script type="text/javascript"> el código necesario para:
    a. agregar la propiedad **moneda** (con el valor ***usd***) a todos los productos cuyo precio sea **impar**.
    b. mostrar todos los productos (tener en cuenta que no todos tendrán la propiedad *moneda*).
    c. eliminar la propiedad *fechaVencimiento* a todos los productos cuyo precio sea **par**.
    d. mostrar todos los productos (tener en cuenta que no todos tendrán la propiedad moneda o la propiedad *fechaVencimiento*).

Nota: Verificar su funcionamiento en la consola del navegador (F12). Utilizar el operador **in** y **delete**.

[D.05] Acumulando jotasones

Crear un documento HTML (***acumulando.html***) y vincular el siguiente archivo: *jotasones.js*.
Guardar el contenido del anterior archivo en el **localStorage** (jotasones), siempre y cuando, no se haya guardado previamente.
Mostrar por **consola** y por **alert** si se guardó o se recuperó del localStorage. En todos los casos, mostrar por consola el contenido completo.
Agregar un nuevo elemento a la colección tomando como valores el incremento en uno del último objeto recuperado.
Informar por **alert** el valor del último id agregado.

Nota: Utilizar las funciones **localStorage.setItem**, **localStorage.getItem**, **JSON.parse** y **JSON.stringify**, según corresponda.

[D.06] Acumulando jotasones II

Tomando como referencia el ejercicio anterior, se pide que se agreguen las instrucciones necesarias para preguntarle al usuario si desea reiniciar el contenido del **localStorage** (si se detecta que hay datos guardados) o continúa normalmente.
Nota: Utilizar la ventana **confirm**.

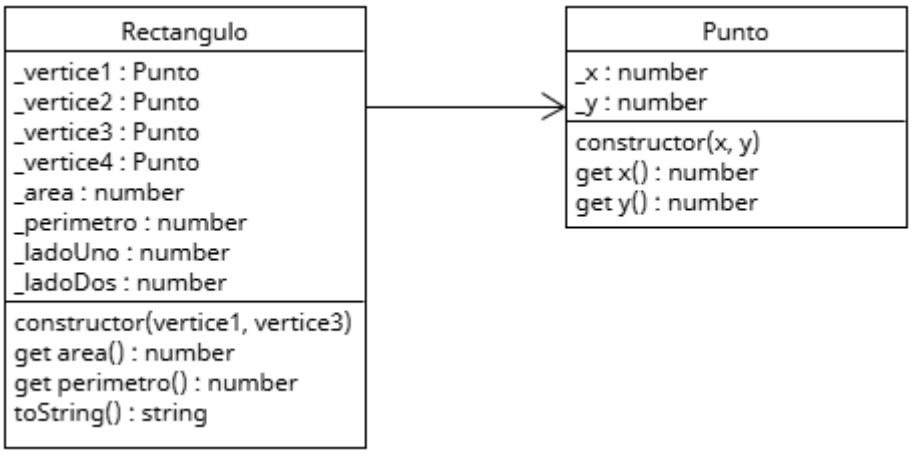
[D.07] Acumulando jotasones III

Tomando como referencia el ejercicio anterior, se pide que se agreguen las instrucciones necesarias para que el usuario ingrese el valor del campo **id** del jotason que desea eliminar de la colección. Informar por **alert**, si se ha eliminado el elemento o no.
Nota: Utilizar la ventana **prompt**.

POO

[D.08] Rectángulo y punto

La clase **Punto** ha de tener dos atributos privados con acceso de sólo lectura (sólo con getters), que serán las coordenadas del punto. Su constructor recibirá las coordenadas del punto.
La clase **Rectangulo** tiene los cuatro atributos privados de tipo **Punto** (_vertice1, _vertice2, _vertice3 y _vertice4, que corresponden a los cuatro vértices del rectángulo).



Los atributos _ladoUno, _ladoDos, _area y _perimetro se deben inicializar una vez construido el rectángulo. Desarrollar una aplicación que muestre, a través de su método ***toString***, todos los datos del rectángulo por consola.

Nota: La base de todos los rectángulos de esta clase será siempre horizontal. Por lo tanto, el constructor para construir el rectángulo se realizará por medio de los vértices 1 y 3.

[D.09] Dibújame la figurita

La clase **FiguraGeometrica** posee todos sus atributos protegidos, un constructor sin parámetros y tres métodos: **toString** (público), **dibujar** (público) y **calcularDatos** (protegido).

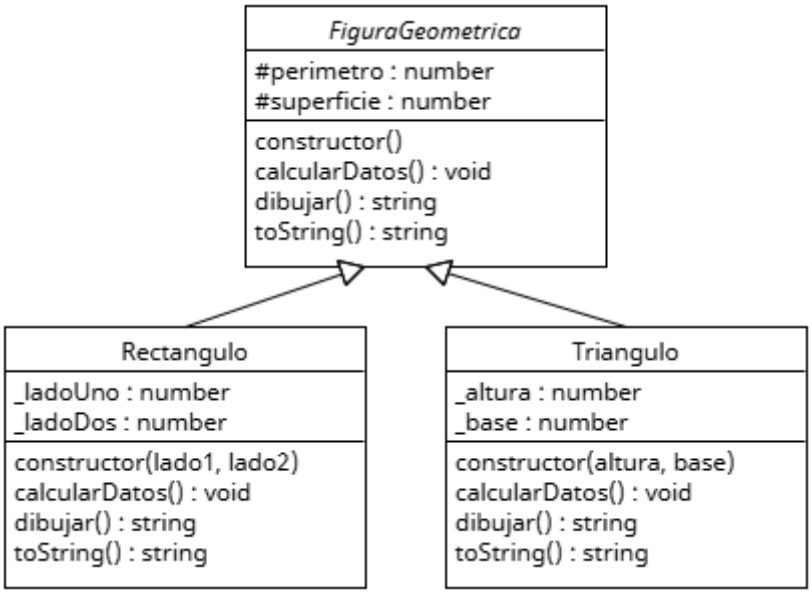
- **calcularDatos** será invocado en el constructor de la clase derivada que corresponda, su funcionalidad será la de inicializar los atributos #superficie y #perimetro.
- **dibujar**, retorna un string formando la figura geométrica del objeto que lo invoque (retornar una serie de asteriscos que modele el objeto).

Ejemplo:

```
*          *******
***        *******
*****      *******
```

- **toString** retorna toda la información completa del objeto, que luego será mostrada por consola.

Jerarquía de clases:



Nota: Hacer que la clase FiguraGeometrica no se pueda instanciar.

Map, filter y reduce

[D.10] Acá el que se aburre es porque quiere...

Crear un documento HTML (***mapeado.html***) y vincular el siguiente archivo: *MOCKDATA.js*.
A partir del array **usuarios**, se pide realizar, en un archivo (***map_filter_reduce.js***), las siguientes tareas:

1. Retornar y mostrar todos los nombres de los trabajos de los usuarios.
2. Retornar y mostrar todos los nombres de los países de los usuarios.
3. Retornar y mostrar un array de objetos de aquellos usuarios cuyo país sea China.
4. Retornar y mostrar una array de objetos de todos los usuarios menores a 21 años.
5. Retornar y mostrar la cantidad de usuarios con sexo masculino (Male).
6. Retornar y mostrar la cantidad de usuarios con sexo femenino (Female).
7. Retornar y mostrar una array de strings (el nombre de los usarios de sexo femenino (Female)).
8. Retornar y mostrar una array de strings (el email de los usarios de sexo masculino (Male)).
9. Retornar y mostrar un array de objetos que solo contengan los nombres, apellidos y ciudades de todos los usuarios.
10. Retornar y mostrar un array de objetos que solo contengan los nombres, apellidos y ciudades de todos los usuarios masculinos mayores de 35 años.
11. Retornar y mostrar el promedio de edad de todos los usuarios.
12. Retornar y mostrar el promedio de edad de los usuarios masculinos.
13. Retornar y mostrar el promedio de edad de los usuarios egipcios (Egypt).