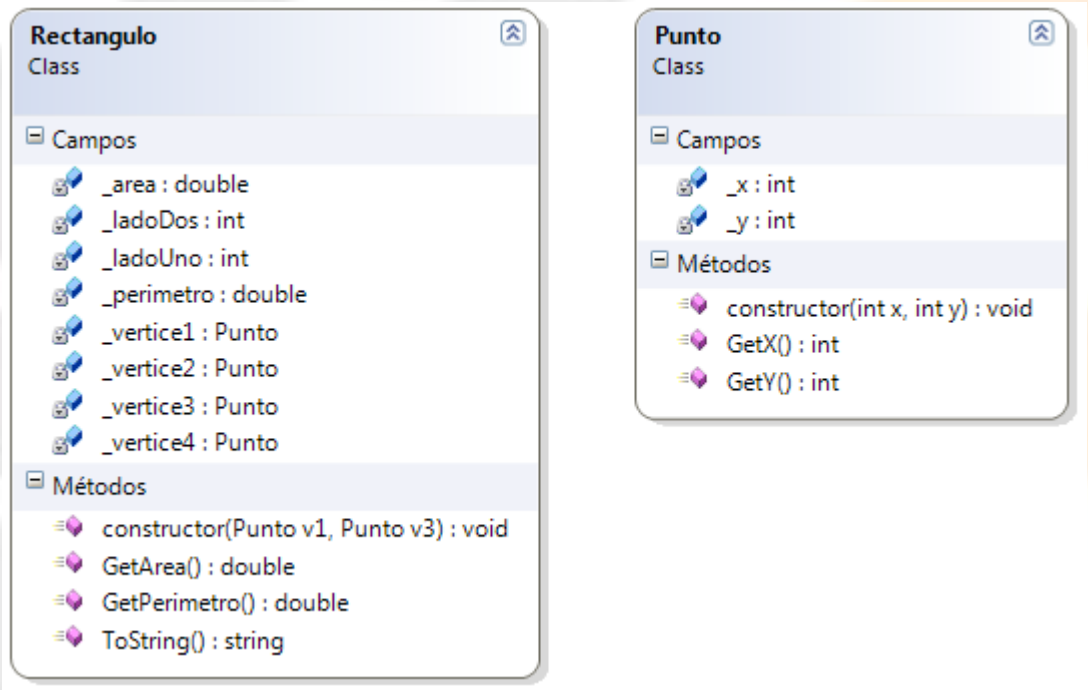
Profesor: *Neiner Maximiliano*

1- Codificar las clases Punto y Rectángulo en TypeScript.

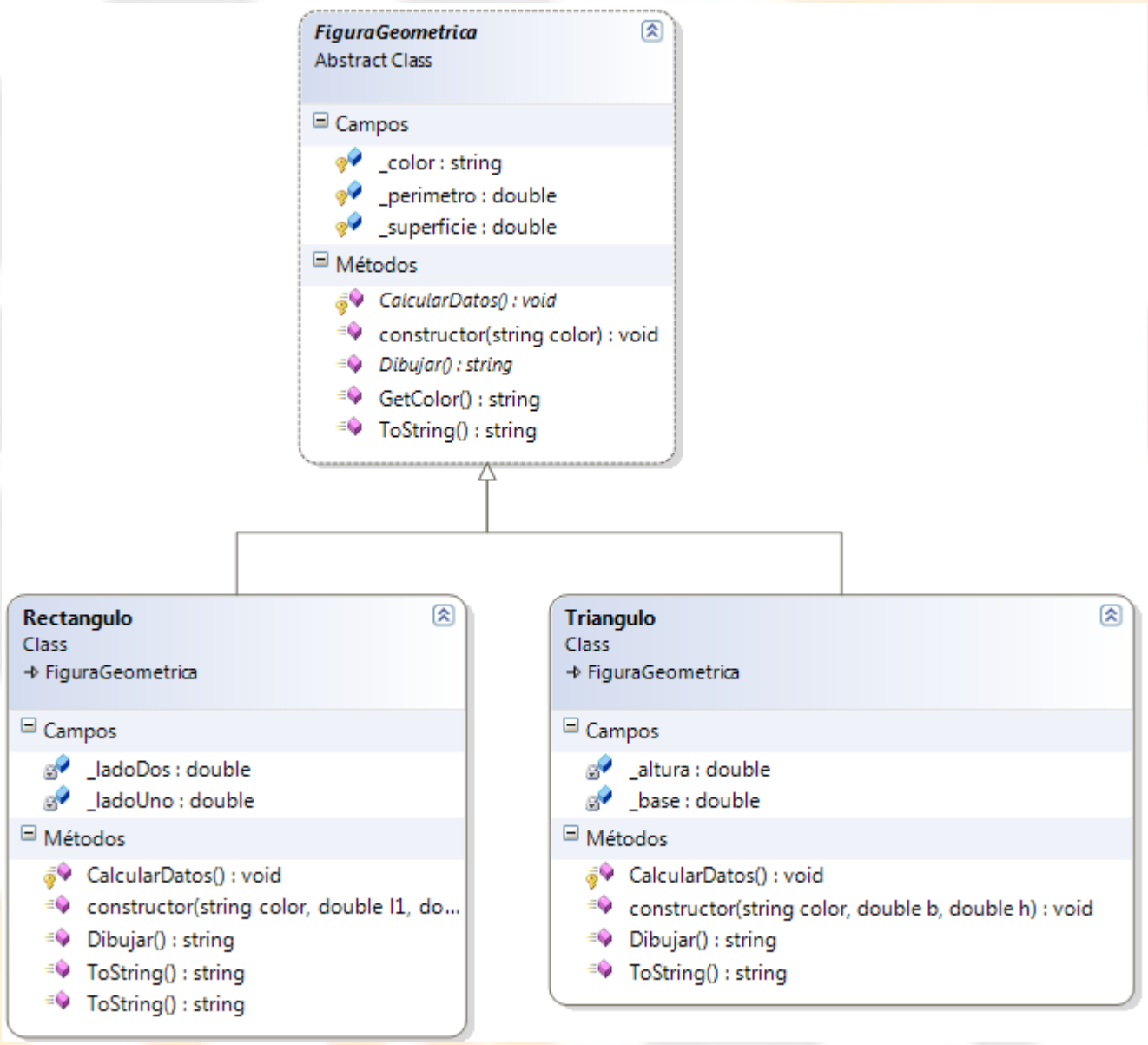


La clase **Punto** ha de tener dos atributos privados con acceso de sólo lectura (sólo con **getters**), que serán las coordenadas del punto. Su constructor recibirá las coordenadas del punto.
La clase **Rectangulo** tiene los atributos privados de tipo **Punto** *_vertice1*, *_vertice2*, *_vertice3* y *_vertice4* (que corresponden a los cuatro vértices del rectángulo).
La base de todos los rectángulos de esta clase será siempre horizontal. Por lo tanto, debe tener un constructor para construir el rectángulo por medio de los vértices 1 y 3.
Los atributos *_ladoUno*, *_ladoDos*, *_area* y *_perimetro* se deberán inicializar una vez construido el rectángulo.
Desarrollar una aplicación que muestre, a través de su método ToString, todos los datos del rectángulo por consola y en la página.

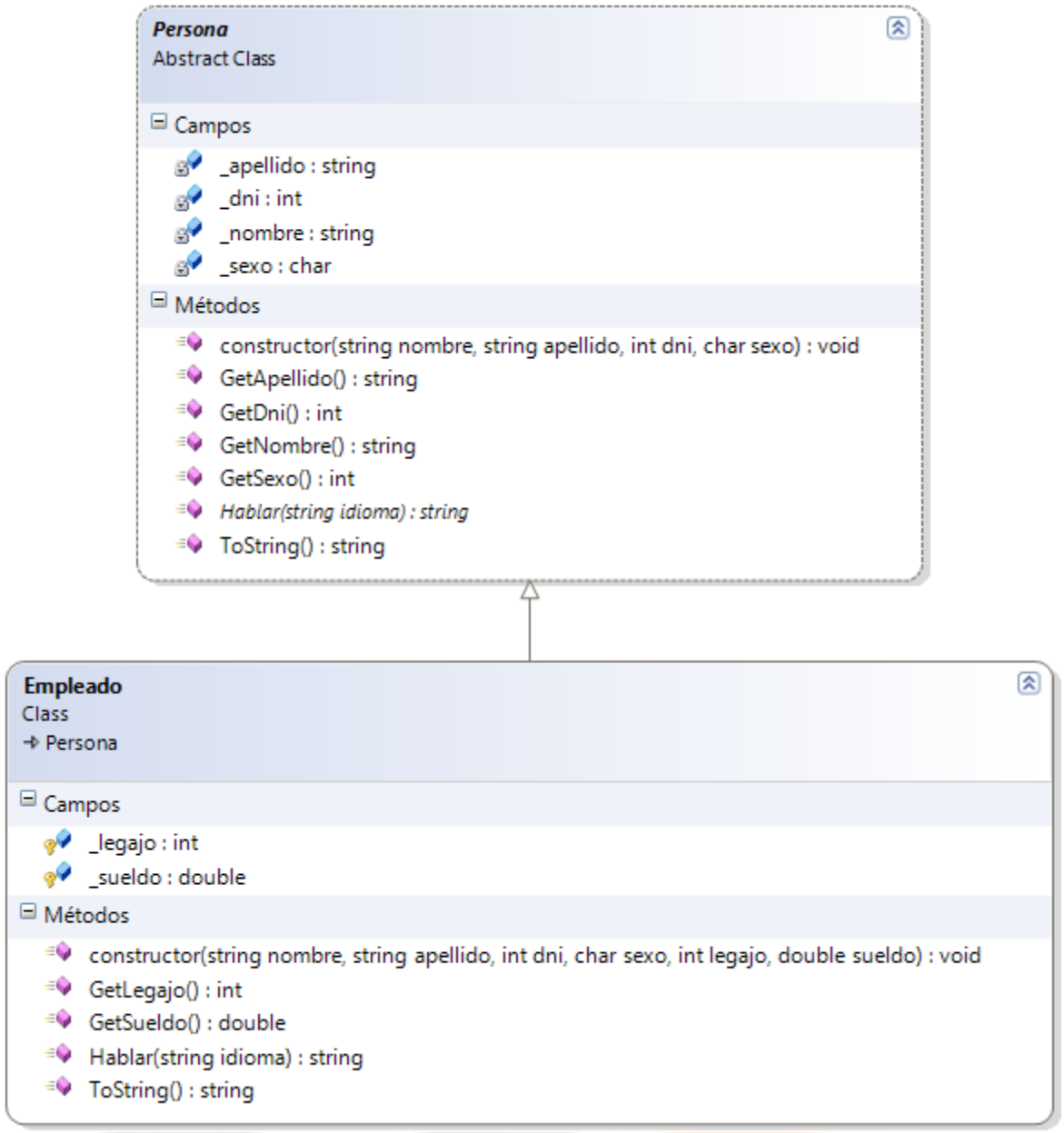
2.- La clase ***FiguraGeometrica*** posee: todos sus atributos protegidos, un constructor con un parámetro, un método getter para el atributo ***_color***, un método virtual (***ToString***) y dos métodos abstractos: ***Dibujar*** (público) y ***CalcularDatos*** (protegido).
CalcularDatos será invocado en el constructor de la clase derivada que corresponda, su funcionalidad será la de inicializar los atributos _superficie y _perimetro.
Dibujar, retornará un string (con el color que corresponda) formando la figura geométrica del objeto que lo invoque (retornar una serie de asteriscos que modele el objeto).
Ejemplo:

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

Utilizar el método ToString para obtener toda la información completa del objeto, y luego dibujarlo por pantalla y mostrarlo por consola.
Jerarquía de clases:



3.- Codificar el siguiente diagrama de clases en TypeScript. Cada clase deberá ser codificada en un archivo distinto. *Ejemplo: persona.ts, empleado.ts, etc.*



Detalles técnicos

<u>Clase Persona</u>

Atributos (todos privados)
Constructor (inicializa los atributos de la clase)
Métodos (de instancia)

- **Hablar** (abstracto). Retorna un string.
- **ToString**. Retorna un string mostrando todos los datos de la persona, separados por un guión medio (-).
- **getters** para cada uno de los atributos.

<u>Clase Empleado</u>

Atributos (todos protegidos)

Constructor (inicializa los atributos de la clase)
Métodos (de instancia)

- **Hablar** (polimorfismo). Retorna un string con el formato "El empleado habla *Español*", siendo Español, el valor recibido por parámetro.
- **ToString** (polimorfismo). Retorna un string mostrando todos los datos del empleado, separados por un guión medio (-).
- **getters** para cada uno de los atributos.

Testear la jerarquía de clases

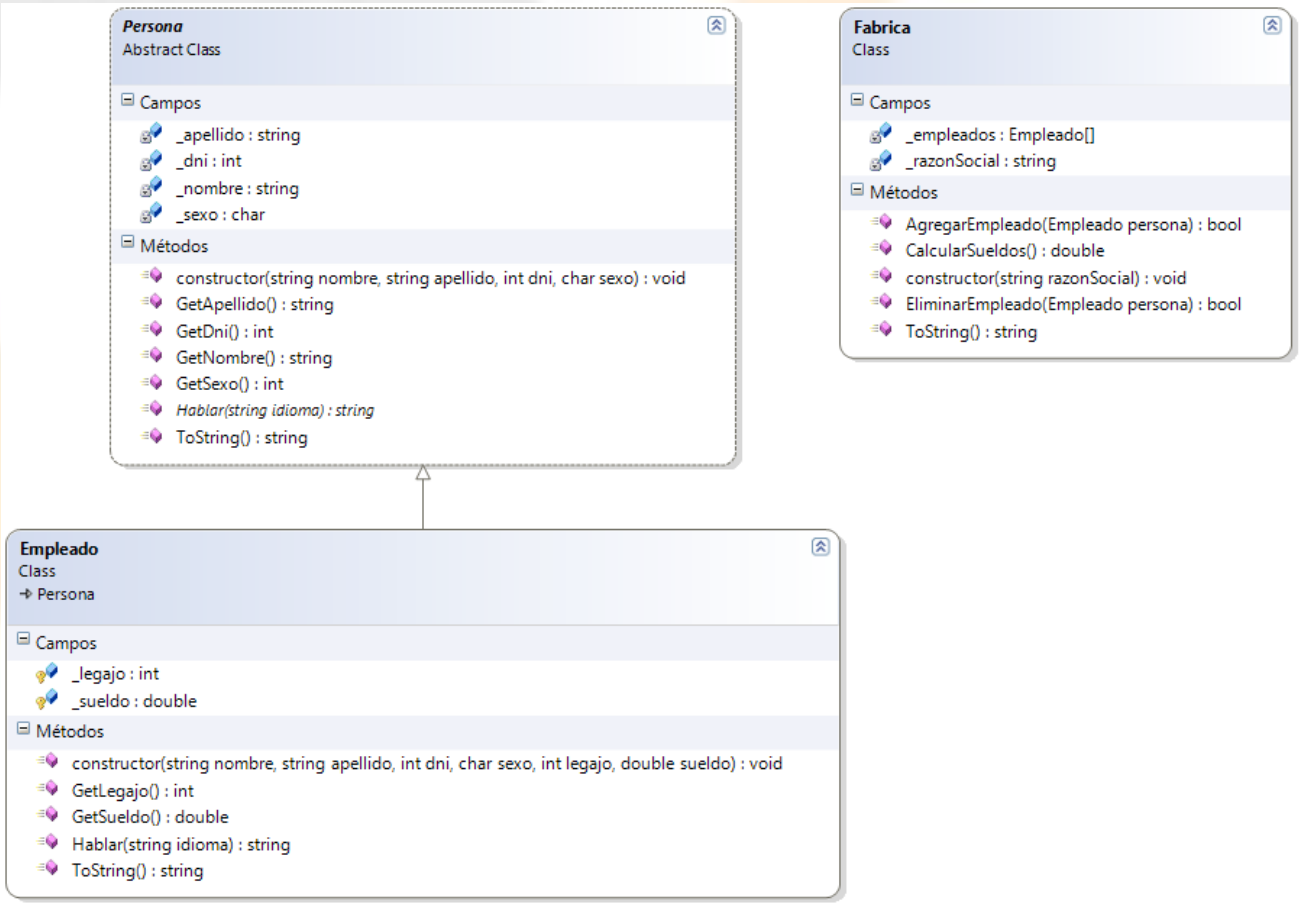
Para probar el buen funcionamiento de la jerarquía anteriormente descrita, en la consola integrada, se deberá instanciar y probar todos los métodos.

Nota: *Recordar incluir las referencias a los archivos a ser utilizados desde el main.ts.*

Creación de formulario de entrada de datos

Teniendo en cuenta el diagrama de clases anteriormente descrito, se pide generar una página Web que posea un formulario de ingreso para los datos correspondientes a un empleado.

Diagrama de clases



Derivación de los datos

Una vez realizado el ingreso de los datos correspondientes a un empleado, se deberá enviar dicha informacion (por **POST**) hacia la página ***administracion.php***, donde se realizará una pequeña validación informando si se evnviaron datos vacios, para lo cual se generará un link (**<a>**) que nos dirija hacia la página principal.
Si se enviaron datos, se creará un objeto de tipo Empleado y se lo mostrará por medio del métdo ***ToString()***.