18. Encapsulación de variables II

1. Elabora un programa que defina la clase llamada Circulo(). La clase tendrá tres métodos:

   - Un constructor para elaborar los objetos de esta clase. El constructor debe utilizar como parámetro el valor del radio.
   - Un método que devuelva el área del círculo.
   - Un método que devuelva el perímetro del circulo.

   Crea un objeto tipo Circulo y muestra por pantalla su área y su perímetro.

   ```
   Radio del círculo:6
   El círculo tiene una superficie: 18.85
   El círculo tiene un perímetro: 37.70
   ```

2. Escribe un programa que contenga una clase a la que llamaremos Rectangulo(). Los objetos de esta clase quedarán definidos por tres propiedades:

   - lados: Número de lados.
   - base: Longitud de la base del rectángulo.
   - altura: Longitud de la altura del rectángulo.
   - Ninguna de estas variables debe estar encapsulada.

   Define los siguientes métodos:

   - Método constructor que crea instancias (objetos) Rectangulo recibiendo como parámetros los valores de la base y la altura.
   - Método que muestra por pantalla las tres propiedades del objeto.
   - Método que calcule el área del rectángulo.
   - Método que calcule el perímetro del rectángulo.

   Crea un objeto rectángulo definiendo la longitud de sus valores a través de ordenes input.

   Muestra el valor de las propiedades número de lados, longitud de la base y longitud de la altura del objeto ejecutando el método que hemos definido para ello.

   Muestra el valor del área y el perímetro del rectángulo ejecutando los métodos creados:.

   ```
   Introduce el valor de la longitud de la base: 5
   Introduce el valor de la longitud de la altura: 6
   Número de lados: 4
   Longitud de la base: 5.0
   Longitud de la altura: 6.0
   El rectángulo tiene una superficie de: 30.0
   El rectángulo tiene un perímetro de: 22.0
   ```

3. Vamos a repasar algo la gestión de errores. Modifica el programa anterior para que las ordenes input solamente puedan aceptar valores numéricos reales mayores de cero. Si se introduce un valor negativo o una cadena de texto el programa continuará pidiendo el valor hasta que se introduzca con el formato correcto.

   ```
   Introduce el valor de la longitud de la base: -5
   La longitud ha de ser un valor numérico mayor que cero
   Introduce el valor de la longitud de la base: 5
   Introduce el valor de la longitud de la altura: siete
   La longitud ha de ser un valor numérico mayor que cero
   Introduce el valor de la longitud de la altura: 7

   Número de lados: 4
   Longitud de la base: 5.0
   Longitud de la altura: 7.0
   ```

```
El rectángulo tiene una superficie de: 35.0
El rectángulo tiene un perímetro de: 24.0
```

4. Modifica el programa anterior de tal forma que el programa te pregunte por el número de lados que tiene el rectángulo. Cambia el valor de la propiedad número de lados de cuatro a tres. Ejecuta ahora el método que muestra la información del rectángulo. ¿Tiene sentido el resultado? ¿Tiene sentido que el usuario pueda cambiar el valor de esa propiedad? ¿Qué solución se te ocurre para este problema?

```
Introduce el valor de la longitud de la base: 5
Introduce el valor de la longitud de la altura: 4
Introduce el número de lados de tu rectángulo:2
El número de lados debe ser un entero mayor de 2
Introduce el número de lados de tu rectángulo:3.5
El número de lados debe ser un entero mayor de 2
Introduce el número de lados de tu rectángulo:3

Número de lados: 3
Longitud de la base: 5.0
Longitud de la altura: 4.0
```

> Hemos podido cambiar esa propiedad a través de una orden del tipo rectan.lados=3. Sin embargo, no tiene sentido que esta orden se pueda programar desde fuera de la clase que define a los objetos rectángulo.

5. La solución al problema que nos hemos encontrado en el ejercicio anterior es encapsular la variable número de lados. Modifica el programa anterior definiendo el número de lados como una variable encapsulada (__lados). Modifica de forma parecida a como lo has hecho en el ejercicio anterior el valor de la propiedad __lados. El programa no dará ningún error, pero observa lo que aparece como resultado al mostrar las propiedades del rectángulo.

```
Introduce el valor de la longitud de la base: 5
Introduce el valor de la longitud de la altura: 4
Introduce el número de lados de tu rectángulo:3

Número de lados: 4
Longitud de la base: 5.0
Longitud de la altura: 4.0
El rectángulo tiene una superficie de: 20.0
El rectángulo tiene un perímetro de: 18.0
```

6. En cinemática se trabaja habitualmente con tres vectores: posición, velocidad y aceleración. Crea un programa con una clase a la que llamaremos Vector(). Cada objeto vector queda definido por tres propiedades:

   - tipo: Esta propiedad solamente podrá tomar uno de los tres siguientes valores:
     - p (para indicar que es un vector de posición)
     - v (si es un vector que indica velocidad)
     - a (si es un vector aceleración)
   - x: Valor tipo float que indica el valor de la coordenada x del vector.
   - y: Valor tipo float que indica el valor de la coordenada y del vector.

   Un vector nunca podrá cambiar de tipo una vez que se definido el mismo. Por lo tanto, esa propiedad deberá estar encapsulada.

   Crea un objeto Vector() siguiendo las normas anteriores. Y muestra sus propiedades:

```
Tipo de vector (p,v,a):p
Coordenada x:2.3
Coordenada y:-7.13
tipo de vector: p
```

```
x= 2.3
y= -7.13
```

7. Partiendo del programa anterior intenta modificar las propiedades "tipo" y "coordenada x" de un objeto Vector() utilizando ordenes sencillas similares a miVector.tipo="p", miVector.x=33. Observa el resultado. ¿Qué conclusiones obtienes?

```
Tipo de vector (p,v,a):p
Coordenada x:3
Coordenada y:2
tipo de vector: p
x= 3.0
y= 2.0
Nuevo tipo para el vector:a
Nuevo valor para la coordenada x:5
tipo de vector: p
x= 5.0
y= 2.0
```

El valor de x ha cambiado sin problemas, el del tipo no. No lo ha hecho porque es una propiedad encapsulada y solamente será posible cambiar su valor a través de un método que esté definido dentro de la clase.

8. Modifica el programa anterior para que se pueda modificar el valor de la propiedad tipo de un vector a través de un método de la clase. Este método utilizará como parámetro el valor del nuevo tipo del vector.

```
Tipo de vector (p,v,a):p
Coordenada x:3
Coordenada y:2
tipo de vector: p
x= 3.0
y= 2.0
Nuevo tipo para el vector:a
Nuevo valor para la coordenada x:5
tipo de vector: a
x= 5.0
y=2.0
```

9. Partiendo de la clase definida en el ejercicio anterior. Crea un método que calcule el módulo del vector y luego muestra su valor por pantalla.

```
Introduce las coordenadas del primer vector:
Tipo de vector (p,v,a):p
Coordenada x:3
Coordenada y:2
Módulo del vector: 3.61
```

10. Crea un programa que permita sumar vectores. Debes crear el vector suma, asignarle por ejemplo el tipo "a" y mostrar sus valores por pantalla.

```
Introduce las coordenadas del primer vector:
Tipo de vector (p,v,a):p
Coordenada x:3
Coordenada y:6
Introduce las coordenadas del segundo vector:
Tipo de vector (p,v,a):p
Coordenada x:1
Coordenada y:3
El resultado de la suma es:
tipo de vector: a
x= 4.0
y= 9.0
```

11. **Desafío:** Modifica el programa anterior de tal forma que cuando los tipos de dos vectores sean diferentes y los intentemos sumar, el código nos informe de la imposibilidad de realizar la suma. A continuación, nos tienen que ofrecer la opción de cambiar el tipo de uno de los vectores y realizar la suma.

    - Otros aspectos a tener en cuenta:
        - El tipo solo puede ser "p", "v" o "a". En caso de solicitar un tipo diferente el programa debe de informar de ello y volver a solicitar el tipo.
        - Las coordenadas han de ser valores tipo float, en caso de introducir una cadena de texto el programa debe informar de ello y volver a solicitar el valor.

> La dificultad de este ejercicio es conseguir leer, de alguna manera, el valor del tipo de vector desde fuera del código de definición de la clase.

Resultado cuando los vectores son del mismo tipo:

```
Introduce las coordenadas del primer vector:
Tipo de vector (p,v,a):p
Coordenada x:3
Coordenada y:2.5
Introduce las coordenadas del segundo vector:
Tipo de vector (p,v,a):p
Coordenada x:-2.5
Coordenada y:1
El resultado de la suma es:
tipo de vector: a
x= 0.5
y= 3.5
```

Si los vectores son de distinto tipo:

```
Introduce las coordenadas del primer vector:
Tipo de vector (p,v,a):v
Coordenada x:12
Coordenada y:10
Introduce las coordenadas del segundo vector:
Tipo de vector (p,v,a):a
Coordenada x:-5
Coordenada y:-1
El primer vector es de tipo v y el segundo es de tipo a.
No es posible sumar dos vectores de dos tipos diferentes.
¿Quieres cambiar el tipo del primer vector a tipo a? (S)S
El resultado de la suma es:
tipo de vector: a
x= 7.0
y= 9.0
```