
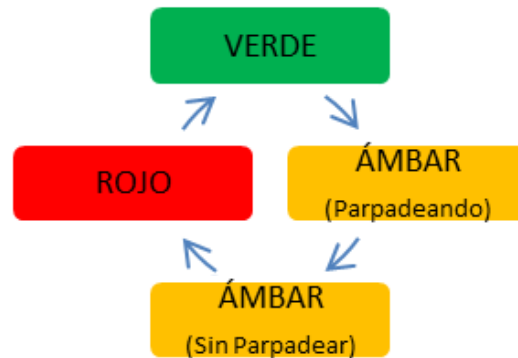
	<p style="text-align: center;"><b>I.E.S. HERMANOS MACHADO</b>  <b>2º CFGS DAW</b>  <b>Desarrollo Web en Entorno Cliente</b>  <b>Unidad 4: Ejercicios</b></p>	<p style="text-align: center;"><b>DEPARTAMENTO DE INFORMÁTICA</b></p>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------


1. Crear una función constructora de objetos *Producto*. Dichos productos tendrán tres atributos, uno de tipo cadena de caracteres llamado *nombre*, otro de tipo entero llamado *unidades* y otro de tipo real llamado *precio*. Aparte de estos tres atributos también contendría los siguientes métodos:
  - *valorEnStock*: que no recibe parámetros y devuelve el importe total de las unidades disponibles, es decir, la multiplicación de las unidades por el precio del producto.
  - *incrementarStock*: que recibe un entero como parámetro y aumenta el número de unidades en la cantidad indicada en el parámetro.
  - *disminuirStock*: que recibe un entero como parámetro y disminuye el número de unidades en la cantidad indicada en el parámetro.
  - Todos los métodos consultores (getter) y modificadores (setter) de los tres atributos descritos anteriormente, teniendo en cuenta que si por error se intentara modificar el atributo *precio* con un valor negativo, este debería guardarse automáticamente como positivo.
2. Desarrolle un programa en el que se instancien varios objetos *Producto* llamando a la función constructora implementada en el ejercicio anterior. Seguidamente modificar el stock de los productos creados, así como su nombre y precio. Implemente un documento web donde se muestre claramente el funcionamiento de todos los métodos disponibles en los objetos *Producto* creados.
3. Implemente la clase Semaforo con un atributo llamado color y otro de tipo booleano llamado parpadeando. El valor del atributo color será 0 cuando el semáforo esté en rojo, 1 cuando esté en ámbar, y 2 cuando esté en verde. El constructor sin parámetros inicializará todos los objetos como semáforos en rojo (color=0) y sin parpadear (parpadeando=false). Incluya los métodos consultores y modificadores estándar para los dos atributos. El código de los métodos modificadores debe asegurar que el atributo color no toma valores no permitidos, y que sólo se puede activar el parpadeo del semáforo cuando el color es ámbar. Añada un método cadenaColor, que no recibe argumentos y devuelve una cadena de caracteres indicando el color actual de semáforo ("ROJO", "ÁMBAR", "VERDE"). Añada un método llamado imprimir, que no recibe ningún argumento ni devuelve ningún valor. Este método imprimirá por pantalla el estado del semáforo usando mensajes del tipo: "Semáforo en ROJO", "Semáforo en AMBAR parpadeando", "Semáforo en AMBAR". Añada un método llamado cambia, que cada vez que es llamado hará cambiar al semáforo del estado en que se encuentre al siguiente estado según el diagrama. Cree un proyecto web donde se instancie un semáforo y demuestre el funcionamiento de la clase implementada, para ello siga la secuencia de operaciones siguiente imprimiendo el semáforo después de cada cambio:
  - Cree un Semáforo.
  - Cambie el color a un color incorrecto.
  - Cambie el color a verde (2).
  - Ponga el atributo parpadeando a cierto.

	<p align="center"><b>I.E.S. HERMANOS MACHADO</b>  <b>2º CFGS DAW</b>  <b>Desarrollo Web en Entorno Cliente</b>  <b>Unidad 4: Ejercicios</b></p>	<p align="center"><b>DEPARTAMENTO DE INFORMÁTICA</b></p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------

- Cambie el color a ámbar (3).
- Ponga el atributo parpadeando a cierto.
- Llame 5 veces al método cambio (use un bucle)
- Cree un nuevo semáforo y asigne los valores de los atributos del primer semáforo al nuevo.



4. Implemente una clase *Figura* que represente de forma genérica a cualquier figura geométrica, debe tener un atributo *color* con sus correspondientes consultores y modificadores. Tendrá un constructor que recibirá el color como único parámetro e inicializará el atributo del mismo nombre. Además tendrá un método llamado *imprimir* que devolvería una cadena como esta si el atributo color fuera "rojo": "Soy una figura de color rojo". Crear una clase *Rectangulo* que herede de *Figura*. Los objetos de esta nueva clase tendrán los atributos *base* y *altura*, con sus modificadores y consultores. El método constructor recibe los parámetros de color, base y altura. De *Figura* también hereda la clase *Circulo*, que tendrá el atributo *radio* y sus métodos consultores y modificadores correspondientes. Ambas clases hijas tendrán un método *calcularArea*, que devolverá el área de la figura representada. Las dos redefinirán el método *imprimir* de manera que exprese qué figura es, cuál es su área y su color, por ejemplo: "Soy un rectángulo rojo de 20 cm<sup>2</sup>". Instanciar varios objetos *Figura*, *Circulo* y *Rectangulo* y comprobar si tienen un funcionamiento correcto.
5. Desarrolla un programa que permita modelar el funcionamiento del sistema de título de abono del transporte público. Para ello, parta de una clase llamada *Bonobus* que dispondrá de un método llamado *picarViaje*, que recibirá como argumento un número entero indicando la línea de autobús, tres enteros para representar el día, mes y año de la fecha actual, y dos enteros para representar la hora y los minutos del momento en el que se pica el billete. Este método devolverá un valor booleano que será verdadero cuando se haya podido picar el viaje y falso cuando el bono ya no sea válido. Hay que implementar dos clases hijas de *Bonobus* llamadas *BonoDiezViajes* y *BonoTarifaPlana*, que representan respectivamente un bono con 10 viajes y un bono válido durante un periodo de tiempo. La clase *BonoDiezViajes* permitirá realizar 10 viajes, tras los cuales quedará invalidado el bono. Para controlar

	<p align="center"><b>I.E.S. HERMANOS MACHADO</b>  <b>2º CFGS DAW</b>  <b>Desarrollo Web en Entorno Cliente</b>  <b>Unidad 4: Ejercicios</b></p>	<p align="center"><b>DEPARTAMENTO DE INFORMÁTICA</b></p>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------

el número de viajes restantes la clase dispondrá de un atributo privado de tipo entero para el cual se implementará sus métodos consultores y modificadores. De la clase anterior heredaré otra llamada *BonoDiezViajesConTrasbordo*, que permitirá realizar un trasbordo durante la hora posterior a la cancelación del billete siempre que se utilice en otra línea de autobús diferente a aquella en la que se realizó la cancelación. Esta clase deberá disponer de un atributo de tipo entero para controlar la línea del último autobús en el que se picó el último viaje, tres atributos de tipo entero para controlar la fecha de dicho viaje, y dos de tipo entero para controlar la hora. Todos estos atributos dispondrán de su correspondiente método consultor y modificador. La clase *BonoTarifaPlana* dispondrá de los atributos necesarios, con sus métodos consultores y modificadores, para representar la fecha en la que caduca la tarjeta. Esta clase implementará el método *picarViaje* de tal forma que permita realizar viajes mientras que la fecha de validez no sea anterior a la fecha actual. Heredando de la clase anterior, cree las clases necesarias para representar los bonos con validez de un día, tres días, un mes y un año. Estas clases sobrescribirán el método *picarViaje* de forma tal que se establezca la fecha de caducidad a partir de la primera vez que se cancele un viaje del bono, y se use la implementación de la clase padre para realizar las comprobaciones pertinentes. Desarrolle instanciaciones de todas las clases anteriormente descritas para comprobar que funcionan correctamente.