Guía de ejercicios - POO

1. Crea una clase **Cilindro** que tenga los atributos **privados** radio y altura. Define un método calcularVolumen() que retorne el volumen del cilindro (usa la fórmula: π * radio^2 * altura). En el **main**, crea un objeto Cilindro y calcula su volumen a partir de los datos ingresados por el usuario.
2. Crea una clase **Empleado** que tenga los atributos nombre, edad y salario. Define un método mostrarDatos() que imprima los datos del empleado. Debe implementar un constructor para inicializar los atributos y también un constructor predeterminado. En el **main**, crea un arreglo de objetos de varios elementos y luego llama al método mostrarDatos() para cada uno.
3. Crea una clase **Producto** con los atributos nombre, precio y cantidad. Implementa métodos para:
   a) Establecer y obtener los valores de los atributos.
   b) Un método calcularTotal() que retorne el precio total (precio * cantidad).

   En el **main**, crea un objeto **Producto**, establece sus valores, y muestra el total calculado.
4. Crea una clase **Vector2D** que represente un vector en 2D con los atributos x e y. Implementa métodos para:
   a) Calcular la longitud del vector.
   b) Sumar y restar otro vector.
   c) Multiplicar el vector por un escalar.

   En el **main**, crea varios vectores, realiza operaciones entre ellos, y muestra los resultados.
5. Crea una clase **Temperatura** que almacene la temperatura en grados Celsius. Implementa métodos para:
   a) Convertir la temperatura a Fahrenheit.
   b) Convertir la temperatura a Kelvin.
   c) Mostrar la temperatura en Celsius, Fahrenheit y Kelvin.

   En el **main**, crea un objeto **Temperatura**, establece un valor en Celsius, y muestra sus equivalentes en las otras escalas.
6. Para Arduino, crear una clase **Entrada**, la misma debe implementar:
   a) Un constructor el cual debe indicarse el pin a utilizar.
   b) getEstado: que devuelva el estado actual de la entrada
   c) getFlancoAsc: que devuelva si se ha detectado un flanco ascendente
   d) getFlancoDes: que devuelva si se ha detectado un flanco descendente

e) Sobrecarga getEstado para que, de manera no bloqueante (con millis), implemente un tiempo antirrebote el cual debe ser pasado por parámetro y tener un valor defecto.

7. Implementar una clase, **Tarea**, en Arduino para manejar tareas periódicas de manera no bloqueante. La clase debe permitir configurar un intervalo de tiempo y verificar si ha pasado ese tiempo para ejecutar la tarea.

   a) El constructor debe recibir el período de tiempo.

   b) Debe implementar el método esTiempo(), que devolverá true si se ha cumplido el tiempo correspondiente.

   Realizar un ejemplo encendiendo dos Leds con intervalos diferente.