Tarea 2 – Composición y encapsulación

Enunciado

1. Usando composición has de crear un total de 5 clases de la siguiente forma:
    a. Crea una clase llamada `Lampara` con tres variables de instancia: `estilo` de tipo String, `batería` de tipo booleano y `calificación` de tipo entero. Todas las variables deben ser privadas. La clase se debe construir con los tres parámetros. Añade 4 métodos: `encender` sin tipo de retorno y debe imprimir el mensaje de que la lámpara se ha encendido, `getEstilo` devuelve el estilo de la lámpara, `hayBateria` devuelve el booleano de batería y `getCalificacion` que devuelve la calificación de la lámpara.
    b. Crea una clase llamada `Cama` con 5 atributos privados: `estilo` String, `almohadas`, `altura`, `sabanas` y `colcha` de tipo entero. El constructor debe ser implementado con esos 5 atributos. Agrega 6 métodos: `hacer` no devuelve nada e imprime que se ha hecho la cama, y los `getters` de estilo, almohadas, altura, sábanas y colcha.
    c. Añade una clase llamada `Techo` con dos variables: `altura` y `color` de tipo entero. Constructor con todos los parámetros. Y los dos métodos getter.
    d. Crea una clase llamada `Pared` con una variable `dirección` de tipo String, su constructor con una variable y el método getter.
    e. La última clase será `Habitación` y contendrá 8 variables de instancia: `nombre` de tipo String, `pared1`, `pared2`, `pared3` y `pared4` de tipo Pared, `techo` de tipo Techo, `cama` de tipo Cama y `lámpara` de tipo Lámpara. El constructor debe contener las 8 variables. La clase debe tener dos métodos: `getLampara` que devuelve el objeto lámpara y `hacerLaCama` que imprime el mensaje se está haciendo la cama y llama al método `hacer` en la clase Cama.
    f. Crea una clase `Main` con un método `main` que pruebe el código.
    g. Añade código y métodos para probar el método `encender` de la clase lámpara.

2. Ahora piensa tú un ejemplo de composición. Usa las palabras “tiene un” / “está compuesto de” para ayudarte.
    a. Por un lado, crea métodos para acceder desde el objeto TODO a las partes que lo componen a partir de los getters.
    b. Ahora cambia la visibilidad de los métodos getters a `private` e implementa métodos que no utilicen getters para acceder a los métodos de las partes como hemos visto en clase.

3. Teniendo en cuenta el principio de encapsulación usado en POO crea una clase llamada Impresora, que simulará una impresora real. Debe tener los campos de nivel de tóner, número de páginas impresas y si es una impresora con impresión a doble cara. Añade:
    a. Un método para añadir una cantidad de tóner al nivel actual, este método actualizará el nivel de tóner (hasta un máximo 100%), la cantidad de tóner que se va a añadir debe estar entre 0-100. Comprobar también que al añadir la cantidad de tóner no se supere el nivel de tóner. Si cualquier condición falla debe devolver -1 en caso contrario se devuelve el nuevo nivel de tóner.
    b. Otro método que simule la impresión de páginas (el cuál debe actualizar el número de páginas impresas). Se le pasarán las páginas a imprimir. También se tendrá en cuenta si la impresora es a doble cara, ya que entonces se habrá de calcular cuántas páginas se imprimen. El método debe devolver las páginas a imprimir y actualizará el valor del número de páginas impresas.
    c. Método getter que devuelve el número de páginas impresas.

   d. Decide que visibilidad van a tener los campos, si es necesario un constructor o no y cualquier cosa que creas necesaria.
   e. Crea una clase `MainEncapsulacion` con un `main` con código de prueba.

4. Piensa en algún código de ejemplo para mostrar el mal y buen uso de encapsulación como hicimos en la teoría.

Entrega

- Copia y pega el código en el PDF.