Guía de Ejercicios de Programación Orientada a Objetos

Encapsulamiento

#	Ejercicio.
1	Crea una clase llamada `Rectangulo` que represente un rectángulo. La clase debe tener dos atributos correspondientes a la base y altura. Implementa las siguientes metodos: • Getters y Setter de cada atributo. • `calcularArea()`: Devuelve el área del rectángulo. • `calcularPerimetro()`: Devuelve el perímetro del rectángulo.
2	Crea una clase llamada `Dado` que simule el comportamiento de un dado de seis caras. La clase debe contener los siguientes atributos: • `valor` (int): Almacena el valor actual del dado (un número entre 1 y 6). Implementa los siguientes métodos públicos: • `Dado()`: Constructor que inicializa el dado con un valor aleatorio entre 1 y 6. • `lanzar()`: Simula el lanzamiento del dado, asignando un nuevo valor aleatorio entre 1 y 6 al atributo `valor`. • `getValor()`: Devuelve el valor actual del dado. • `esMaximo()`: Devuelve `true` si el valor del dado es 6, y `false` en caso contrario. • `esMinimo()`: Devuelve `true` si el valor del dado es 1, y `false` en caso contrario.

3 Crea una clase llamada `CuentaBancaria` que represente una cuenta bancaria. La clase debe tener los siguientes atributos:

- Número de cuenta (entero)
- Saldo actual (float)

Implementa los siguientes métodos:

- Un constructor que me permita establecer el número de cuenta y el saldo.
- Un método `depositar(float monto)` que incremente el saldo.
- Un método `retirar(float monto)` que disminuya el saldo si hay fondos suficientes, caso contrario no hace nada.
- Un método `obtenerSaldo()` que devuelva el saldo actual.

4 Crea una clase llamada `Triangulo` que represente un triángulo. La clase debe contener un vector de 3 elementos, donde cada elemento corresponde a la longitud de un lado del triángulo. Implementa los siguientes métodos:

- `getLado(int numero)`: Devuelve la longitud del valor del lado correspondiente al número proporcionado (1, 2, o 3). Si el número es incorrecto (fuera del rango 1-3), devuelve cero.
- `setLado(int numero, float valor)`: Establece el valor del lado correspondiente al número proporcionado (1, 2, o 3). Si el número es incorrecto (fuera del rango 1-3), no realiza ninguna acción.
- `getTipo()`: Devuelve el tipo de triángulo según sus lados:
  - 1 para un triángulo equilátero (todos los lados iguales).
  - 2 para un triángulo isósceles (dos lados iguales).
  - 3 para un triángulo escaleno (todos los lados diferentes).
- `isEscaleno()`: Devuelve `true` si el triángulo es escaleno, `false` en caso contrario.
- `isIsosceles()`: Devuelve `true` si el triángulo es isósceles, `false` en caso contrario.

- `isEquilatero()`: Devuelve `true` si el triángulo es equilátero, `false` en caso contrario.

5 Crea una clase llamada `Termometro` que represente un termómetro digital. La clase debe contener los siguientes atributos:

- `temperatura` (float): Almacena la temperatura actual medida por el termómetro.
- `unidad` (char): Almacena la unidad de medida de la temperatura ('C' para Celsius, 'F' para Fahrenheit).

Implementa los siguientes métodos:

- `Termometro(float tempInicial, char unidadInicial)`: Constructor que inicializa la temperatura y la unidad de medida.
- get y set de temperatura.
- `cambiarUnidad(char nuevaUnidad)`: Cambia la unidad de medida entre Celsius y Fahrenheit. Si la nueva unidad es diferente de la actual, convierte la temperatura a la nueva unidad.
    - Fórmula de conversión de Celsius a Fahrenheit: `(C * 9/5) + 32`
    - Fórmula de conversión de Fahrenheit a Celsius: `(F - 32) * 5/9`
- `getUnidad()`: Devuelve la unidad actual de medida.

6 Crea una clase llamada `Punto` que represente un punto en un plano cartesiano. La clase debe contener los siguientes atributos:

- `x` (float): Almacena la coordenada en el eje X.
- `y` (float): Almacena la coordenada en el eje Y.

Implementa los siguientes métodos públicos:

- `Punto(float xInicial, float yInicial)`: Constructor que inicializa las coordenadas `x` y `y` del punto.
- Getters y Setters para cada atributo.

- `calcularDistancia(Punto otroPunto)`: Devuelve la distancia entre el punto actual y otro punto dado. La fórmula para calcular la distancia entre dos puntos `(x1, y1)` y `(x2, y2)` es: `sqrt((x2 - x1)^2 + (y2 - y1)^2)`.
- `mover(float deltaX, float deltaY)`: Mueve el punto sumando `deltaX` a `x` y `deltaY` a `y`.

7 **Crea una clase llamada `Usuario` que represente a un usuario en un sistema.** La clase debe tener los siguientes atributos:

- Nombre (string)
- Clave (string)
- Rol (string): Puede ser "admin" o "user".

Implementa los siguientes métodos:

- **Usuario(string nombre, string clave, string rol)**: Constructor que inicializa los atributos.
- getter y setter de cada atributo

Desarrolla un programa que realice lo siguiente:

1. **Cargar en el sistema una lista de 5 usuarios** utilizando un array de objetos `Usuario` (esto debe estar hardcodeado en el programa).
2. **Solicitar al usuario que ingrese su nombre y contraseña** al iniciar el programa.
3. **Verificar si las credenciales ingresadas coinciden con alguno de los usuarios cargados** en el sistema utilizando una función que reciba el array de usuarios, la cantidad de usuarios, el nombre y la contraseña. Esta función debe devolver el índice donde se encuentra el usuario en el array, o -1 si el usuario no existe.
4. **Si se encuentra un usuario con las credenciales correctas, permitir el acceso al sistema** mostrando el rol al que pertenece con un saludo amigable. Utiliza una función que reciba un objeto `Usuario` y muestre el saludo con el rol específico.

5. **Si el usuario ingresa credenciales incorrectas, permitir un máximo de 3 intentos**. Si se agotan los intentos, el programa debe finalizar indicando que se han agotado los intentos.