BOLETIN5_Ejercicios Básicos de Programación Orientada a Objetos

1) Crea una clase **Contador** con un atributo entero "contador" y con los métodos para **incrementar y decrementar** el contador. La clase contendrá un constructor por defecto, un constructor con parámetros y los métodos getters y setters.

2) Crea una clase llamada **Cuenta** que tendrá los siguientes atributos: titular y cantidad (puede tener decimales).

- El titular será obligatorio y la cantidad es opcional. Crea dos constructores que cumpla lo anterior
- Crea sus métodos get, set y toString.
- Tendrá dos métodos especiales:
    - **ingresar(double cantidad): void**. se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
    - **retirar(double cantidad): void.** se retira una cantidad a la cuenta, si restando la cantidad actual a la que nos pasan es negativa, la cantidad de la cuenta pasa a ser 0.

3) Crea una clase **Fecha**. La clase contendrá

- Además de constructores, métodos set y get y el método **toString.**
- Método para comprobar si la fecha es correcta.
- Método para modificar la fecha actual por la del día siguiente.

4) Escribir una clase que represente un **reloj** que señale la hora el minuto, y el segundo. La clase dispondrá de dos constructores, uno sin parámetros, que pone el reloj a 0:0:0, y otro al que se le pasa la hora, los minutos y los segundos. Además, habrá que realizar los siguientes métodos:

- **dameHora(): String**. Uno que da la hora, los minutos y los segundos, separados por el carácter ":", en una cadena.
- **dameHora(String tipo).** Dos tipos: "24hrs" y "12hrs". Otro que también da la hora, pero en formato 24 horas (como el anterior) o en formato de 12 horas, en cuyo caso debe distinguir entre "am" (para horas de 0 a 11) o "pm", para horas de 12 a 23, también en una cadena.

5) Haz una clase llamada **Persona** que siga las siguientes condiciones:

- Sus atributos son: nombre, edad, DNI, sexo (H hombre, M mujer), peso y altura. No queremos que se accedan directamente a ellos.
- Por defecto, todos los atributos menos el DNI serán valores por defecto según su tipo (0 números, cadena vacía para String, etc.). Sexo sera hombre por defecto, usa una constante para ello.

Se implantarán varios constructores:

- Un constructor por defecto.
- Un constructor con el nombre, edad y sexo, el resto por defecto.
- Un constructor con todos los atributos como parámetro.

Los métodos que se implementaran son:

- calcularIMC():boolean. calculara si la persona está en su peso ideal (peso en kg/(altura^2 en m)), si esta fórmula devuelve un valor menor que 20, la función devuelve un -1, si devuelve un número entre 20 y 25 (incluidos), significa que está por debajo de su peso ideal la función devuelve un 0 y si devuelve un valor mayor que 25 significa que tiene sobrepeso, la función devuelve un 1. Te recomiendo que uses constantes para devolver estos valores.
- esMayorDeEdad(): boolean. indica si es mayor de edad, devuelve un booleano.
- comprobarSexo(char sexo): boolean. comprueba que el sexo introducido es correcto. Si no es correcto, será false.
- toString(): String. devuelve toda la información del objeto.
- generaDNI(): void. genera un número aleatorio de 8 cifras, genera a partir de este su número su letra correspondiente. Este método será invocado cuando se construya el objeto. Puedes dividir el método para que te sea más fácil. No será visible al exterior.

Métodos set de cada parámetro, excepto de DNI.

Ahora, crea una clase ejecutable que haga lo siguiente:

- Pide por teclado el nombre, la edad, sexo, peso y altura.
- Crea 3 objetos de la clase anterior, el primer objeto obtendrá las anteriores variables pedidas por teclado, el segundo objeto obtendrá todos los anteriores menos el peso y la altura y el último por defecto, para este último utiliza los métodos set para darle a los atributos un valor.
- Para cada objeto, deberá comprobar si está en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
- Indicar para cada objeto si es mayor de edad.
- Por último, mostrar la información de cada objeto.

6) Vamos a crear una clase para trabajar con una pila. Una pila es una estructura de datos que nos permite guardar un conjunto de variables. La característica fundamental es que el último elemento que se añade al conjunto es el primero que se puede sacar.

Para representar una pila vamos a crear una clase llamada “Pila” y en su interior utilizar un arreglo (vector) de cadena de caracteres con tamaño 10, por lo tanto, la pila no podrá tener más de 10 elementos. Vamos a crear varias funciones para trabajar con la pila:

- inicializarPila(): void. Como tenemos un arreglo de 10 elementos de cadenas tenemos que inicializarlo e introducir un carácter (por ejemplo, un * que indique que ese elemento del arreglo no corresponde con un dato de la pila. Esta función inicializa el vector con ese carácter.
- LongitudPila(): void. Función devuelve el número de elementos que tiene la pila (elementos en el array no vacio “*”).
- EstaLlenaPila(): boolean. Función que devuelve si la pila está llena.
- AddPila(String): void: función que recibe una cadena de caracteres y añade la cadena a la pila, si no está llena. Si está llena muestra un mensaje de error.
- SacarDeLaPila(): String: Función que devuelve el último elemento añadido y lo borra de la pila. Si la pila está vacía muestra un mensaje de error y devuelve “None”.
- MostrarPila(): void Método que muestra en pantalla los elementos de la pila.

Realiza un programa principal que nos permita usar las funciones anteriores, que nos muestre un menú, con las siguientes opciones:

1. Añadir elemento a la pila
2. Sacar elemento de la pila
3. Longitud de la pila
4. Mostrar pila
5. Salir

7) Queremos desarrollar una aplicación que nos ayude a gestionar las notas de un centro educativo. Cada alumno tiene los siguientes atributos: Nombre, Apellidos, Edad, Nota.

Dispondremos de dos clases que representaremos con un array y cada clase tendrá 4 alumnos.

Crear un algoritmo que solicite los datos para cada uno de los alumnos de salvo la nota, tanto para la clase 1 como para la clase 2.

Posteriormente, realiza las siguientes funciones:

- mostrarAlumnos (Alumno[]): void. Crear una función que dado el Array de alumnos de una clase muestre por orden de posición/ingreso los datos de cada alumno.
- mostrarDatosAlumno(Alumno[], int): void. Función que muestre los datos de un alumno situado en una posición determinada de la clase.
- ponerNotaAlumno(Alumno[], int, int): void. Cambiar la nota de un alumno por una nota proporcionada como argumento.
- incrementarNotaAlumno(Alumno[], int, int): void. Incrementar la nota del alumno según el valor proporcionado por argumento. Ojo. Si la nota supera un 10, esta se queda en un 10.
- decrementarNotaAlumno(Alumno[], int, int): void. Decrementar la nota del alumno según el valor proporcionado por argumento. Ojo. Si la nota es menor que un 0, esta se queda en un 0.
- obtenerAlumnoMasJoven(Alumno[]) : String. Crea una función que proporcione el nombre del alumno de uno de los alumnos más jóvenes (el alumno más joven y en caso de disponer de dos alumnos con la misma edad el primero descubierto).
- obtenerAlumnoMasJoven(Alumno[]) : int. Crea una función que proporcione el índice del alumno más joven (en caso de disponer de dos alumnos con la misma edad devolverá el primer alumno descubierto).
- obtenerAlumnoMasJoven(Alumno[]) : int[]. Crea una función que retorne los índices de los alumnos más jóvenes.
- calcularNotaMedia(Alumno[]) : int. Función que retorna la nota media de todos los alumnos de una clase.
- obtenerAlumnosAprobados(Alumno[]) : Alumno[]. Obtener un Array con los objetos de los alumnos que han aprobado.