EJERCICIOS de Programación Orientada a Objetos - PROGRAMACIÓN I

Ejercicio 1:

Crea una clase llamada Cuenta que tendrá los siguientes atributos:

1. titular y cantidad (puede tener decimales). Crear sus métodos get, set.
2. Crear dos constructores que permitan crear una cuenta para un titular y la cantidad.

Tendrá dos métodos especiales:

1. ingresar(double cantidad): se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
2. retirar(double cantidad): 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.

Ejercicio 2:

Crear una clase Libro que contenga los siguientes atributos:

1. ISBN, Título, Autor, Número de páginas

Crear sus respectivos métodos get y set correspondientes para cada atributo. Crear el método toString() para mostrar la información relativa al libro con el siguiente formato:
“El libro con ISBN creado por el autor tiene páginas”

Crear una aplicación que permita crear 2 objetos Libro (los valores que se quieran) y mostrarlos por pantalla. Por último, indicar cuál de los 2 tiene más páginas.

Ejercicio 3:

Crear una clase Fracción, que cuente con dos atributos: dividendo y divisor, que se asignan en el constructor, y se imprimen como X/Y en el método toString().

1. Crear un método sumar que recibe otra fracción y devuelve una nueva fracción con la suma de ambas.
2. Crear un método multiplicar que recibe otra fracción y devuelve una nueva fracción con el producto de ambas.

Ejercicio 4:

Crear una clase Vector, que en su constructor reciba una lista de elementos que serán sus coordenadas. En el método toString() se imprime su contenido con el formato [x,y,z].

1. Crear un método que reciba un número y devuelva un nuevo vector (objeto de la clase Vector), con los elementos multiplicados por ese número.
2. Crear un método sumar que recibe otro vector, verifica si tienen la misma cantidad de elementos y devuelve un nuevo vector con la suma de ambos. Si no tienen la misma cantidad de elementos entonces imprimir un mensaje de error y la función debe retornar el vector que se pasó como parámetro.

Ejercicio 5:

Escribir una clase Papel que contenga un atributo texto, un método escribir, que reciba una cadena de caracteres para agregar al texto, y el método toString() que imprima el contenido del texto.
Escribir una clase Birome que contenga un atributo cantidad de tinta, y un método escribir, que reciba un texto y un papel (objeto de la clase Papel) sobre el cual escribir. Cada letra escrita debe reducir la cantidad de tinta contenida. Cuando la tinta se acabe, debe imprimir un mensaje de error indicando que la tinta se ha acabado.

Ejercicio 6:

Escribir una clase Corcho, que contenga los atributos

1. bodega (cadena con el nombre de la bodega) y enBotella, booleano que indique si esta puesto o no en la botella.

Escribir una clase Botella que contenga un atributo:

1. corcho (user la clase Corcho para definir el atributo de la clase) con una referencia al corcho que la tapa (objeto de la clase Corcho).

Escribir una clase Sacacorchos que contenga un atributo:

1. tieneCorcho que indica si el sacacorchos ha sido usado previamente

Y los métodos

1. destapar() que recibe un objeto botella. El método sirve para sacar el corcho de la botella (cambia el atributo del objeto corcho que forma parte de la clase botella). Debe imprimir un mensaje de error en el caso en que la botella ya está destapada. Antes de destapar una botella, debe verificar si el objeto Sacacorchos ha sido previamente usado o no. En caso de que esté usado, imprima un mensaje de error y no destape la botella.
2. limpiar() que permite quitar el corcho del sacacorchos o en caso de que el sacacorchos no tenga un corcho, entonces imprima un mensaje de error indicando esto.

Escriba una aplicación en donde se crean los objetos corcho, botella y sacacorcho y se pueda sacar el corcho de la botella usando el sacacorchos.

Ejercicio 7:

Crear una clase Mate que describa el funcionamiento de la conocida bebida tradicional local. La clase debe contener como miembros:

1. Un atributo para la cantidad de cebadas restantes hasta que se lava el mate (representada por un número).
2. Un atributo para el estado (lleno o vacío).
3. El constructor debe recibir como parámetro n, la cantidad máxima de cebadas en base a la cantidad de yerba vertida en el recipiente.
4. Un método cebar(), que llena el mate con agua. Si se intenta cebar con el mate lleno, se debe imprimir un mensaje de error 'Cuidado! Te quemaste!'
5. Un método beber(), que vacía el mate y le resta una cebada disponible. Si se intenta beber un mate vacío, se debe imprimir un mensaje de error 'El mate está vacío !'
6. Es posible seguir cebando y bebiendo el mate aunque no haya cebadas disponibles. En ese caso la cantidad de cebadas restantes se mantendrá en 0, y cada vez que se intente beber se debe imprimir un mensaje de aviso: 'Advertencia: el mate está lavado.'.

Crear una aplicación en donde se cree un objeto Mate y se puedan tomar mates.

Ejercicio 8:

Crear una clase llamada Persona que siga las siguientes atributos: nombre, edad, DNI, sexo (H hombre, M mujer), peso y altura. No se debe permitir acceder directamente a estas.

Se implementarán varios constructores:

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

Los métodos que se implementarán son:

1. calcularIMC(): 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.
2. esMayorDeEdad(): indica si es mayor de edad, devuelve un booleano.
3. comprobarSexo(char sexo): comprueba que el sexo introducido es correcto. Si no es correcto, será H. No será visible al exterior.
4. toString(): devuelve toda la información del objeto.

5. generaDNI(): 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.
6. Métodos set de cada parámetro, excepto de DNI.

Crea una aplicación que haga lo siguiente:

1. Pide por teclado el nombre, la edad, sexo, peso y altura.
2. 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.
3. Para cada objeto, deberá comprobar si está en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
4. Indicar para cada objeto si es mayor de edad.
5. Por último, mostrar la información de cada objeto.

Ejercicio 9:

Crear una clase llamada Password que siga las siguientes condiciones:

1. Que tenga los atributos longitud y contraseña . Por defecto, la longitud sera de 8.

Los constructores serán los siguiente:

1. Un constructor por defecto.
2. Un constructor con la longitud. Generar una contraseña aleatoria con esa longitud.

Los métodos que implementa serán:

1. esFuerte(): devuelve un booleano si es fuerte o no, para que sea fuerte debe tener mas de 2 mayúsculas, mas de 1 minúscula y mas de 5 números.
2. generarPassword(): genera la contraseña del objeto con la longitud que tenga.
3. Método get para contraseña y longitud.
4. Método set para longitud.

Crear una aplicación que haga lo siguiente:

1. Crea un array de Passwords con el tamaño que tu le indiques por teclado.
2. Crea un bucle que cree un objeto para cada posición del array.
3. Indica también por teclado la longitud de los Passwords (antes de bucle).
4. Crea otro array de booleanos donde se almacene si el password del array de Password es o no fuerte (usa el bucle anterior).
5. Al final, muestra la contraseña y si es o no fuerte (usa el bucle anterior). Usa este simple formato:

contraseña1 valor_booleano1
contraseña2 valor_bololeano2

Ejercicio 10:

Crear una clase llamada Serie con las siguientes características:

1. Sus atributos son título, número de temporadas, entregado, género y creador. Por defecto, el número de temporadas es de 3 temporadas y entregado false. El resto de atributos serán valores por defecto según el tipo del atributo.

Los constructores que se implementarán serán:

1. Un constructor por defecto.
2. Un constructor con el título y creador. El resto por defecto.
3. Un constructor con todos los atributos, excepto de entregado.

Los métodos que se implementara serán:

1. Métodos get de todos los atributos, excepto de entregado.
2. Métodos set de todos los atributos, excepto de entregado.

Crearemos una clase Videojuego con las siguientes características:

1. Sus atributos son titulo, horas estimadas, entregado, genero y compañia.
   Por defecto, las horas estimadas serán de 10 horas y entregado false. El resto de atributos serán valores por defecto según el tipo del atributo.

Los constructores que se implementarán serán:

1. Un constructor por defecto.
2. Un constructor con el título y horas estimadas. El resto por defecto.
3. Un constructor con todos los atributos, excepto de entregado.

Los métodos que se implementara serán:

1. Métodos get de todos los atributos, excepto de entregado.
2. Métodos set de todos los atributos, excepto de entregado.

Las clases anteriores tienen los siguientes métodos:

1. entregar(): cambia el atributo prestado a true.
2. devolver(): cambia el atributo prestado a false.
3. isEntregado(): devuelve el estado del atributo prestado.

Implementa los anteriores métodos en las clases Videojuego y Serie.

Crear una aplicación que realiza lo siguiente:

1. Crea dos arrays, uno de Series y otro de Videojuegos, de 5 posiciones cada uno.
2. Crea un objeto en cada posición del array, con los valores que desees, puedes usar distintos constructores.
3. Entrega algunos Videojuegos y Series con el método entregar().

4. Cuenta cuantos Series y Videojuegos hay entregados. Al contarlos, retornar por pantalla el resultado.

Por último, indica el Videojuego tiene más horas estimadas y la serie con más temporadas. Mostrarlos en pantalla con toda su información.