CICLO FORMATIVO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES WEB *UT5 – INTRODUCCIÓN A LA POO*	EFA MORATALAZ Profesor: **DGC**	Asignatura: **PROGRAMACIÓN** Fecha:	**Nota**:
Alumno: _____			

1) 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): se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
   - 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.

2) 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. Piensa que modificador de acceso es el más adecuado, también su tipo. Si quieres añadir algún atributo puedes hacerlo.

Por defecto, todos los atributos menos el DNI serán valores por defecto según su tipo (0 números, cadena vacía para String, etc.).

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(): 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(): indica si es mayor de edad, devuelve un booleano.
    - comprobarSexo(char sexo): comprueba que el sexo introducido es correcto. Si no es correcto, será H.
- toString(): devuelve toda la información del objeto.
- 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.
- 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.

3) Haz una clase llamada Password que siga las siguientes condiciones:

- Que tenga los atributos longitud y contraseña. Por defecto, la longitud será de 8.
- Los constructores serán los siguiente:
    - Un constructor por defecto.
    - Un constructor con la longitud que nosotros le pasemos. Generará una contraseña aleatoria con esa longitud.
- Los métodos que implementa serán:
    - esFuerte(): devuelve un booleano si es fuerte o no, para que sea fuerte debe tener más de 2 mayúsculas, más de 1 minúscula y más de 5 números.
    - generarPassword(): genera la contraseña del objeto con la longitud que tenga.

- Método get para contraseña y longitud.
- Método set para longitud

Ahora, crea una clase clase ejecutable:

- Crea un array de Passwords con el tamaño que tú le indiques por teclado.
- Crea un bucle que cree un objeto para cada posición del array.
- Indica también por teclado la longitud de los Passwords (antes de bucle).
- Crea otro array de booleanos donde se almacene si el password del array de Password es o no fuerte (usa el bucle anterior).
- Al final, muestra la contraseña y si es o no fuerte (usa el bucle anterior).

4) Crear una clase Libro que contenga los siguientes atributos:
    - ISBN
    - Titulo
    - 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»

En el fichero main, crear 2 objetos Libro (los valores que se quieran) y mostrarlos por pantalla.
Por último, indicar cuál de los 2 tiene más páginas.

5) Vamos a realizar una clase llamada Raices, donde representaremos los valores de una ecuación de 2º grado. Tendremos los 3 coeficientes como atributos, llamémosles a, b y c. Hay que insertar estos 3 valores para construir el objeto. Las operaciones que se podrán hacer son las siguientes:

   - obtenerRaices(): imprime las 2 posibles soluciones
   - obtenerRaiz(): imprime única raíz, que será cuando solo tenga una solución posible.
   - getDiscriminante(): devuelve el valor del discriminante (double), el discriminante tiene la siguiente formula, (b^2)-4*a*c
   - tieneRaices(): devuelve un booleano indicando si tiene dos soluciones, para que esto ocurra, el discriminante debe ser mayor o igual que 0.
   - tieneRaiz(): devuelve un booleano indicando si tiene una única solución, para que esto ocurra, el discriminante debe ser igual que 0.
   - calcular(): mostrara por consola las posibles soluciones que tiene nuestra ecuación, en caso de no existir solución, mostrarlo también.

Formula ecuación 2º grado: (-b±√((b^2)-(4*a*c)))/(2*a)
Solo varia el signo delante de -b

6) Queremos representar con programación orientada a objetos, un aula con estudiantes y un profesor. Tanto de los estudiantes como de los profesores necesitamos saber su nombre, edad y sexo. De los estudiantes, queremos saber también su calificación actual (entre 0 y 10) y del profesor que materia da. Las materias disponibles son matemáticas, filosofía y física.

   Los estudiantes tendrán un 50% de hacer novillos, por lo que si hacen novillos no van a clase pero, aunque no vayan, quedará registrado en el aula (como que cada uno tiene su sitio). El profesor tiene un 20% de no encontrarse disponible (reuniones, baja, etc.)

   Las dos operaciones anteriores deben llamarse igual en Estudiante y Profesor (polimorfismo).
   El aula debe tener un identificador numérico, el número máximo de estudiantes y para que esta destinada (matemáticas, filosofía o física). Piensa que más atributos necesita.

Un aula para que se pueda dar clase necesita que el profesor esté disponible, que el profesor de la materia correspondiente en el aula correspondiente (un profesor de filosofía no puede dar en un aula de matemáticas) y que haya más del 50% de alumnos.

El objetivo es crear un aula de alumnos y un profesor y determinar si puede darse clase, teniendo en cuenta las condiciones antes dichas.

Si se puede dar clase mostrar cuantos alumnos y alumnas (por separado) están aprobados de momento (imaginad que les están entregando las notas).

NOTA: Los datos pueden ser aleatorios (nombres, edad, calificaciones, etc.) siempre y cuando tengan sentido (edad no puede ser 80 en un estudiante o calificación ser 12)

7) Vamos a hacer una baraja de cartas españolas orientado a objetos. Una carta tiene un número entre 1 y 12 (el 8 y el 9 no los incluimos) y un palo (espadas, bastos, oros y copas)

   La baraja estará compuesta por un conjunto de cartas, 40 exactamente.

   Las operaciones que podrá realizar la baraja son:

   - barajar: cambia de posición todas las cartas aleatoriamente
   - siguienteCarta: devuelve la siguiente carta que está en la baraja, cuando no haya más o se haya llegado al final, se indica al usuario que no hay más cartas.
   - cartasDisponibles: indica el número de cartas que aún puede repartir
   - darCartas: dado un número de cartas que nos pidan, le devolveremos ese número de cartas (piensa que puedes devolver). En caso de que haya menos cartas que las pedidas, no devolveremos nada, pero debemos indicárselo al usuario.
   - cartasMonton: mostramos aquellas cartas que ya han salido, si no ha salido ninguna indicárselo al usuario
   - mostrarBaraja: muestra todas las cartas hasta el final. Es decir, si se saca una carta y luego se llama al método, este no mostrara esa primera carta.