Guía VI de Objetos: HashMaps

Ejercicios varios

Comentario:

- Recuerden utilizar las distintas implementaciones de Map para resolver los problemas y almacenar valores.

1) 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.). Sexo sera hombre por defecto, usa una constante para ello.

- Se implantaran 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 esta 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 esta 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, sera H. No sera visible al exterior.
    - 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 sera 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 esta en su peso ideal, tiene sobrepeso o por debajo de su peso ideal con un mensaje.
- Utilizar el dni como clave del hashmap

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

Que tenga los atributos longitud y contraseña . Por defecto, la longitud sera de 8.
Los constructores serán los siguiente:
Un constructor por defecto.
Un constructor con la longitud que nosotros le pasemos. Generara 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 main ejecutable:

- Crea un array de Passwords con el tamaño que tu le indiques por teclado.
- Crea un bucle que cree un objeto para cada posición del array. Utilizar hashcode como clave del hash.
- 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).
- Crear un método para compara dos contraseñas de dos instancias diferentes. Y retornar un valor lógico si es o no igual. (pista: compareTo)
- Al final, muestra la contraseña y si es o no fuerte (usa el bucle anterior).

3) Crearemos una clase llamada Serie con las siguientes características:

- Sus atributos son titulo, numero de temporadas, entregado, genero y creador.
- Por defecto, el numero 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 implementaran serán:
    - Un constructor por defecto.
    - Un constructor con el titulo y creador. El resto por defecto.
    - Un constructor con todos los atributos, excepto de entregado.

Los métodos que se implementara serán:

- Métodos get de todos los atributos, excepto de entregado.
- Métodos set de todos los atributos, excepto de entregado.
- Sobrescribe los métodos toString.

Crearemos una clase Videojuego con las siguientes características:

- 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 implementaran serán:
    - Un constructor por defecto.
    - Un constructor con el titulo y horas estimadas. El resto por defecto.
    - Un constructor con todos los atributos, excepto de entregado.

Los métodos que se implementara serán:

- Métodos get de todos los atributos, excepto de entregado.
- Métodos set de todos los atributos, excepto de entregado.
- Sobrescribe los métodos toString.

Tanto los videojuegos como las series tienen el siguiente comportamiento. Elegir el mejor diseño de clases para solventar el comportamiento común

- entregar(): cambia el atributo prestado a true.
- devolver(): cambia el atributo prestado a false.
- isEntregado(): devuelve el estado del atributo prestado.
- Método compareTo (Object a), compara las horas estimadas en los videojuegos y en las series el numero de temporadas. Como parámetro que tenga un objeto, no es necesario que implementes la interfaz Comparable. Recuerda el uso de los casting de objetos.

Implementa los anteriores métodos en las clases Videojuego y Serie. Ahora crea una aplicación ejecutable y realiza lo siguiente:

- Crea dos arrays, uno de Series y otro de Videojuegos, de 5 posiciones cada uno.
- Crea un objeto en cada posición del array, con los valores que desees, puedes usar distintos constructores. Pero no pueden existir dos series o videojuegos con el mismo nombre dentro del array. Hacer las validaciones correspondientes utilizando el método equals.
- Entrega algunos Videojuegos y Series con el método entregar().
- Cuenta cuantos Series y Videojuegos hay entregados. Al contarlos, devuélvelos.
- Por último, indica el Videojuego tiene más horas estimadas y la serie con mas temporadas. Muestralos en pantalla con toda su información (usa el método toString()).