CURSO JAVA 1

Programación Orientada a Objetos

Ejercicios Extras

CLASE SERVICIO

egg | Argentina programa 4.0

Ejercicios extras

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre, puedes continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por último, recuerda que la prioridad es ayudar a los compañeros de tu equipo y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

1. 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 a través de un método constructor. Luego, en RaicesServicio las operaciones que se podrán realizar son las siguientes:

   a) Método getDiscriminante(): devuelve el valor del discriminante (double). El discriminante tiene la siguiente fórmula: (b^2)-4*a*c

   b) Método tieneRaices(): devuelve un booleano indicando si tiene dos soluciones, para que esto ocurra, el discriminante debe ser mayor o igual que 0.

   c) Método tieneRaiz(): devuelve un booleano indicando si tiene una única solución, para que esto ocurra, el discriminante debe ser igual que 0.

   d) Método obtenerRaices(): llama a tieneRaíces() y si devolvió true, imprime las 2 posibles soluciones.

   e) Método obtenerRaiz(): llama a tieneRaiz() y si devolvió true imprime una única raíz. Es en el caso en que se tenga una única solución posible.

   f) Método calcular(): esté método llamará tieneRaices() y a tieneRaíz(), y mostrará por pantalla las posibles soluciones que tiene nuestra ecuación con los métodos obtenerRaices() o obtenerRaiz(), según lo que devuelvan nuestros métodos y en caso de no existir solución, se mostrará un mensaje.

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

2. Dígito Verificador. Crear una clase NIF que se usará para mantener DNIs con su correspondiente letra (NIF). Los atributos serán el número de DNI (entero largo) y la letra (String o char) que le corresponde. En NIFService se dispondrá de los siguientes métodos:

a) Métodos getters y setters para el número de DNI y la letra.

b) Método crearNif(): le pide al usuario el DNI y con ese DNI calcula la letra que le corresponderá. Una vez calculado, le asigna la letra que le corresponde según

c) Método mostrar(): que nos permita mostrar el NIF (ocho dígitos, un guion y la letra en mayúscula; por ejemplo: 00395469-F).

La letra correspondiente al dígito verificador se calculará a traves de un método que funciona de la siguiente manera: Para calcular la letra se toma el resto de dividir el número de DNI por 23 (el resultado debe ser un número entre 0 y 22). El método debe buscar en un array (vector) de caracteres la posición que corresponda al resto de la división para obtener la letra correspondiente. La tabla de caracteres es la siguiente:

POSICIÓN	LETRA
0	T
1	R
2	W
3	A
4	G
5	M
6	Y
7	F
8	P
9	D
10	X
11	B

12	N
13	J
14	Z
15	S
16	Q
17	V
18	H
19	L
20	C
21	K
22	E