Ejercicios adicionales Optativos Arreglos uni y multi dimensionales

Ej 1

Escribir la clase CeldaSudoku, que contendrá una matriz de enteros de 3 posiciones por lado. Crear los siguientes métodos:

- public void mostrar(), que mostrará el contenido de la matriz.
- public boolean agregarValor(int fila, int columna, int valor), que deberá validar que el valor a ingresar (entre 1 y 9) no esté presente en otra posición de la matriz.

Ej 2

Clave4 es un acertijo numérico donde el programa "inventa" un número de cuatro dígitos, sin repetidos, que el jugador debe adivinar ingresando él también sucesivas series de 4 dígitos. Como el usuario puede abandonar ingresando un asterisco, el número de 4 dígitos se carga en un String:

1. Si el String es un asterisco, significa que el jugador abandona y el programa terminará.
2. Si es otra cosa, se procesa utilizando el método procesar(String valores) de Clave4, el cual debe ser completado. Este método, en primera instancia, debe verificar que la cadena ingresada sea válida. La cadena será válida sólo cuando mida exactamente 4 (cuatro) caracteres de longitud y todas sus posiciones (obtener cada una con el método charAt()) sean caracteres entre '0' y '9'.
    a. Si la cadena ingresada es válida, el método devolverá una instancia de la clase Resultado.
    b. Si no es válida, el método devolverá null.
3. El diseño de la clase Resultado puede verse a continuación:

Resultado
- int buenos - int regulares
+ Resultado(int buenos, int regulares) + int getBuenos() + int getRegulares() + int getMalos()

Los tres getters (getBuenos(), getMalos() y getRegulares()) indican el resultado de la comparación de la serie original de dígitos y la ingresada con el usuario). Buenos indica la cantidad de dígitos que coinciden en valor y posición con los inventados por el programa; regulares indica la cantidad de dígitos que aparecen en el número "inventado" por el programa, pero que se encuentran mal ubicados. Finalmente, malos marca la cantidad de dígitos que no aparecen en el original. En el constructor se

cargan los dos vectores a partir de los strings con el método de String toCharArray(). Se cuentan los buenos y los regulares y a partir de estos dos valores se obtiene la cantidad de malos.

En el programa principal, crear una instancia de Clave4 y, dentro de un ciclo, permitir que el usuario ingrese datos hasta que haya acertado o cargue un asterisco. Ejemplo: Si la máquina "inventa" el 4321 (no debe mostrar el número en ningún momento), se podría dar lo que se muestra a continuación:

Valor ingresado	Resultado
1234	0 buenos, 4 regulares, 0 malos
2346	1 buenos, 2 regulares, 1 malos
5678	0 buenos, 0 regulares, 4 malos
4378	2 buenos, 0 regulares, 2 malos
4321	4 buenos, 0 regulares, 0 malos (ganó)

Ej3

Nos piden que implementemos el software para una máquina expendedora de golosinas.

Cada golosina tiene un nombre y un precio para ahorrar tiempo, os paso los datos que tendrá a continuación (copiar y pegar):

```
String[][] nombresGolosinas = {

            {"KitKat", "Chicles de fresa", "Lacasitos", "Palotes"},

            {"Kinder Bueno", "Bolsa variada Haribo", "Chetoos",
"Twix"},

            {"Kinder Bueno", "M&M'S", "Papa Delta", "Chicles de
menta"},

            {"Lacasitos", "Crunch", "Milkybar", "KitKat"}

};



double[][] precio = {

  {1.1, 0.8, 1.5, 0.9},

  {1.8, 1, 1.2, 1},

  {1.8, 1.3, 1.2, 0.8},

```

```
23   {1.5, 1.1, 1.1, 1.1}
24
25};
```

También tendrán una cantidad inicial, que en principio será de 5.

Tendremos un pequeño menú con las siguientes opciones:

- **Pedir golosina**: pedirá la posición de la golosina que quiera. Esta máquina tiene golosinas en cada posición, identificados por su fila y columna, que será lo que introduzca el usuario al pedir una golosina, por ejemplo si el usuario teclea 20 significa que está pidiendo la golosina que está en la fila 2 columna 0. Cuando no haya más golosinas se le indicará al usuario. Solo puede pedir una golosina y supondremos que el usuario siempre tiene dinero al elegir. Recuerda de disminuir la cantidad la pedir.
- **Mostrar golosinas**: mostrara todas las golosinas disponibles. Mostrará el código que debe introducir el usuario, el nombre y el precio. La cantidad no se mostrará.
- **Rellenar golosinas**: esta es una función exclusiva de un técnico por lo que nos pedirá una contraseña, si el usuario escribe "MaquinaExpendedora2017" le pedirá la posición de la golosina y la cantidad.
- **Apagar maquina**: sale del programa, antes de salir mostrara las ventas totales durante la ejecución del programa.

El programa debe ser modularizado, es decir, todas las funciones que veas que sean necesarias debes crearlas, así como todas aquellas acciones que veas que se repitan. Piensa que funciones pueden ser.

Las funciones deben ser lo más genéricas posibles.

Ej 4

Vamos a realizar el típico juego del 3 en raya, donde habrá dos jugadores que tengan que hacer el 3 en raya, los signos serán el X y el O, cuando haya una posición vacía habrá un –

El tablero de juego, será una matriz de 3×3 de char.

El juego termina cuando uno de los jugadores hace 3 en raya o si no hay más posiciones que poner.

El juego debe pedir las posiciones donde el jugador actual quiera poner su marca, esta debe ser validada y por supuesto que no haya una marca ya puesta.

Realízalo de forma modular, como en el anterior, puedes reutilizar funciones o crear nuevas versiones de ellas.

Ej 5

Queremos realizar una encuesta a 10 personas, en esta encuesta indicaremos el sexo (1=masculino, 2=femenino), si trabaja (1=si trabaja, 2= no trabaja) y su sueldo (si tiene un trabajo, sino sera un cero) estará entre 600 y 2000 (valor entero). Los valores pueden ser generados aleatoriamente. Calcula y muestra lo siguiente:

- Porcentaje de hombres (tengan o no trabajo).
- Porcentaje de mujeres (tengan o no trabajo).
- Porcentaje de hombres que trabajan.
- Porcentaje de mujeres que trabajan.
- El sueldo promedio de los hombres que trabajan.
- EL sueldo promedio de las mujeres que trabajan.

Usa todos los métodos que veas necesarios, piensa que es aquello que se repite o que puede ser mejor tenerlo por separado.