COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación			CURSO:	1º
	PROTOCOLO:	Boletín de ejercicios	AVAL:		DATA:	
	AUTOR	Francisco Bellas Aláez (Curro)				

Boletín de Arrays y Colecciones

Ejercicios básicos:

1. Realiza los métodos **estáticos** que se citan a continuación. Realiza en el programa principal una prueba de los mismos con un vector de tamaño 10.

a) Función que devuelva un vector con N números enteros con valores aleatorios entre 1000 y 5000. N es un parámetros de la función.

b) Función a la que se le pasa un vector cualquiera de números enteros y muestra sus elementos por pantalla. Debes usar un for mejorado mostrando elemento a elemento.

c) Función a la que se le pasa un vector cualquiera de enteros y devuelve el valor máximo que contiene (dentro de la función NO se muestra por pantalla). Usa for clásico.

d) Función a la que se le pasa un vector cualquiera de enteros y devuelve el valor mínimo que contiene (dentro de la función NO se muestra por pantalla). Usa for mejorado.

e) Función a la que se le pasa el vector cualquiera y dos índices e intercambia los datos que hay en las posiciones indicadas por los índices. Si hay un error de rango devuelve *false* si no devuelve *true*.

2. Repite el ejercicio anterior pero adaptándolo a una matriz de tamaño NxM y que guarde caracteres (char) aleatorios entre 'A' y 'Z'. Pruébalo con una tabla de 3x4. Ten en cuenta que:

- En el apartado (b) debes mostrar el array con formato de tabla, además cada carácter de una fila debe tener una separación de al menos dos espacios con los de al lado.
- El apartado (e) lógicamente tendrás que pasar 4 índices. En el programa principal se probará con un array de 3x4.

3. Repite el primer ejercicio adaptándolo a un ArrayList<Integer>. Pruébalo con una colección de 10 elementos.

Una vez que acabes estos ejercicios básicos, se deben comentar las funciones del resto de ejercicios del boletín.

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	DAM	
	MÓDULO	Programación		CURSO:	1º
	PROTOCOLO:	Boletín de ejercicios	AVAL:	DATA:	
	AUTOR	Francisco Bellas Aláez (Curro)			

Ejercicios estándar:

4. Crea una clase con los siguientes métodos estáticos públicos que serán realizados **usando sólo el método charAt()** y **la propiedad length:**

a) *muestraEnLinea*: Método que muestre cada letra de una cadena que se pasa como parámetro en una línea distinta.

b) *subCadena*: Método al que se le pase una cadena, una posición de inicio y una cantidad de caracteres y devuelve el fragmento indicado. Si se le pasan parámetros no válidos devuelve cadena vacía.

c) *muestraCentrado*: Función que se le pasa un string y lo muestra centrado en a consola (supón consola de 80 caracteres de ancho).

d) *cadenaAVector*: Función a la que se le pasa una cadena y devuelve un vector de char con cada uno de los caracteres de la cadena.

e) *alReves*: Función a la que se le pasa una frase y la devuelve en orden inverso.

f) *pasoAMayusculas*: Función a la que se le pasa un número indeterminado de cadenas como parámetros y las devuelve concatenadas y todo en mayúsculas dentro del ASCII estándar de 7 bits (es decir, no tengas en cuenta vocales con tilde ni diéresis ni la letra Ñ), además si la cadena tiene un guión bajo (_) colocará un espacio en su lugar. Revisa el **Apéndice II** para hacer este apartado.

5. a) Crea una clase denominada Ventas que como mínimo conste de los siguientes miembros:

- Un vector público de 12 posiciones de números enteros que representaran las ventas de cada mes del año.

- Un entero privado que representa el año. Con set y get. El set comprueba que el año sea menor que el actual (busca cómo obtener el actual). Si es mayor o igual se queda con el año pasado.

- Un constructor que inicialice el vector con números aleatorios entre 0 y 1000 y el año con un parámetro.

- Un método denominado *media* que devuelve la media de valores del vector

- Un método denominado *grafica* que muestra en pantalla un gráfico de barras de forma que cada 100 unidades vendidas aparezca un nuevo bloque # aumentado a la barra de ese mes. Será algo así (fíjate que quede bien formateado):

```
Año 2021:
Mes  1  (10): #
Mes  2 (220): ###
Mes  3 (920): ##########
Mes  4 (170): ##
…
```

b) En el programa principal (en otra clase) se le pide al usuario el año, crea un objeto de la clase Ventas del año pedido mediante el constructor, se muestra el gráfico de barras y finalmente se muestra la media con dos decimales. El programa se repetirá hasta que desee el usuario.

c) Modifica el main de forma que si se le pasa desde la línea de comando el parámetro año ya no se lo pida al usuario. Mira el **Apéndice III** de los apuntes para más información.

6. Realizar una clase denominada Matriz con los siguientes miembros:

- Una propiedad que es matriz bidimensional de números enteros.

- Un constructor que se le pase un parámetro entero N de forma que inicialice la matriz a tamaño NxN con valores entre 0 y 10 aleatorios.

- Función **estática** que muestra una matriz que se le pasa como parámetro en formato tabla. Deben aparecer los índices como cabeceras de filas y columnas

- Función suma con las siguientes sobrecargas:
  - Sin parámetros: devuelve la suma de todos los elementos de la matriz
  - Un parámetro boolean: Si está a true devuelve la suma de los elementos de la diagonal principal de la matriz. Si está a false la suma del resto de los elementos (todos menos la diagonal). Trata de hacerlo con solo un bucle para la diagonal y ninguno para los que no son de la diagonal (si no te salo hazlo con varios bucles).
  - Un parámetro entero: suma de los elementos de la fila indicada por el parámetro. Si dicha fila no existe devuelve -1.

- Función denominada *borraFila* a la cual se le pasa un entero y devuelve la matriz pero sin la fila indicada por el número del parámetro. Si el número está fuera del rango válido devuelve la matriz completa.

En el programa principal (en otra clase) se inicializa un objeto de la clase Matriz y se plantea un menú con las opciones que llaman a las funciones correspondientes:

- Mostrar matriz.
- Mostrar suma de todos los elementos.
- Mostrar suma de la diagonal.
- Mostrar suma de los elementos salvo diagonal
- Mostrar suma de elementos de una fila: la fila la escoge el usuario.
- Mostrar matriz sin una fila: la fila la escoge el usuario.
- Salir. No sale del menú hasta que se selecciona esta opción.

(Opcional) Añade una función denominada *borraFilas* a la cual se le pase una colección de números enteros que contiene números de filas: Devuelve un array como el original pero sin las filas indicadas en la colección (si hay números fuera del rango simplemente que no los tenga en cuenta).

7. Diseño de una colección (ArrayList) de videojuegos.

Se parte de una clase muy sencilla denominnada **Videojuego** con los campos Titulo, año y fabricante. En los set de título y fabricante se hará que los textos se guarden en mayúsculas, y en el de año si este es anterior a 1950 se pondrá el año actual.

En otra clase denominada **Coleccion** tendrá una propiedad privada denominada videojuegos que será un ArrayList de objetos Videojuego.

Dispondrá de la función menu en la que se debe permitir al usuario las siguientes opciones:

- Insertar nuevo videojuego (se permite decidir al usuario al principio o al final de la colección si hubiera ya algún elemento).
- Visualizar toda las lista de videojuegos: Se muestra en cada fila un videojuego bien formateado en columnas incluído el índice en la colección y con cabeceras. Si el título o el fabricante ocupa más de 23 caracteres lo trunca a tamaño 20 y añade puntos suspensivos (...).
- Buscar videojuegos: En este punto el usuario mete el principio del título y mostrar todos los títulos que empiecen por dicho fragmento.
- Eliminar videojuego (por posiciones).
- Borrar videojuegos de un año determinado.
- Salir del programa

Al ejecutar el programa se debe inicializar la colección automáticamente al menos con 3 títulos.

Debes hacer otras funciones para modularizar mejor el código y evitar repetir subrutinas.

Finalmente en una tercera clase denominada **Principal** estará el main en el que se instancia un objeto tipo **Coleccion** y llama a su función **menu**.

Por supuesto se debe hacer un programa claro para el usuario informando o salvando los posibles errores que se puedan producir (eliminar titulo no existente, que introduzca con mayúsculas o minúsculas, que introduzca espacios al principio o al final de la cadena, etc.)

COLEXIO VIVAS S.L.						
	RAMA:	Informática	CICLO:	DAM		
	MÓDULO	Programación			CURSO:	1º
	PROTOCOLO:	Boletín de ejercicios	AVAL:		DATA:	
	AUTOR	Francisco Bellas Aláez (Curro)				

8. En un instituto se desea realizar una serie de estadísticos con las notas de los alumnos. Para ello y como fase inicial de un futuro proyecto se pide la realización de un programa de simulación de notas de dicho instituto. Se creará entonces una tabla de 4 materias y 15 alumnos. Para llenarla se usarán notas aleatorias entre 0 y 10 sin decimales. Se deben ponderar de forma que los porcentajes de notas sean: 0, 1 y 2 → 5% cada una; 3 → 10%; 4, 5 y 6 → 15% cada una; 7 y 8 → 10% cada una; 9 y 10 → 5% cada una.

Habrá un menú con las opciones siguientes:

- Ver tabla de notas.
- Ver notas de un alumno.
- Ver notas de una materia.
- Calcular la media de notas global.
- Media de un alumno.
- Media de una materia.
- Nota máxima y mínima de una asignatura.
- Modificación de una nota.
- Salir

En todos los casos pertinentes se mostrarán los datos con nombres de alumnos y/o materias.

La estructura del programa será orientada a objetos de forma que habrá una clase ***Aula*** que contiene como propiedades una matriz de notas privado y dos vectores de cadenas, uno con los nombres de los alumnos y otro con los nombres de las materias.

El array de notas es privado, por tanto tiene tiene *get* normal y el *set* se le pasan 3 datos: la posición y el valor a introducir. Los vectores tienen *get* pero no *set*, son por tanto de solo lectura.

Tendrá un constructor que inicializará los tres *arrays* indicando el tamaño como parámetros (aunque luego sea de 15x4 debe ser modificable). También tendrá todos los métodos necesarios que actúen sobre el array a nivel de proceso es decir, sin interfaz de usuario: medias, máximos, mínimos, …

Por tanto, en esta clase Aula no habrá **nada de interfaz de usuario** (ni entrada ni salida)**.**

Otra clase será ***Menu*** donde estará todo el interfaz de usuario. Debería tener al menos una propiedad que sea un objeto de la clase Aula, un método que implemente el menú y un método por cada una de las opciones del menú que serán llamadas desde el switch.

Cuando pidas un alumno o asignatura, muestra la lista de alumnos/asignaturas y que se seleccione con índice.

Desde el programa principal sólo se llamará al objeto *Menu*.

9. Realiza una simulación de aciertos en el juego de la lotería primitiva usando colecciones. Para ello haz las siguientes funciones:

Realiza una función (denominada rellenaCol) que tenga como parámetro una colección de enteros. Debe limpiarla y rellenarla con 6 valores aleatorios distintos entre 1 y 49 ambos inclusive.

Otra función (denominada compara) a la que se le pasan dos colecciones de enteros y comprueba cuantos elementos de una colección están en la otra. Devuelve dicho valor.

En el main el usuario introducirá 6 números diferentes entre 1 y 49 separados por comas y se guardarán en una colección (se deben hacer las comprobaciones pertinentes). Si quieres haz alguna función más para modularizar esta parte.

A continuación slanza un millón de loterías mediante la función rellenaCol y por cada una comprueba los aciertos que ha tenido el usuario mediante compara. Usa un array de 7 posiciones para ir incrementando cada una de ellas según los aciertos. Es decir, si compara devuelve 0 incrementas la posición 0 de ese array, si compara devuelve 3 pues rellenas la posición 3. De esa forma al final de la simulación tienes la cantidad de veces que han salido 0 aciertos, 1 acierto, 2, aciertos, etc. Muestra estos datos por pantalla y comprueba que efectivamente no vale la pena jugar

10. Realiza un método que devuelva la letra (tipo char) de un DNI que se le pasa como parámetro (tipo String). El algoritmo de hallar dicha letra es el siguiente:

- Se calcula el resto de dividir el DNI entre 23
- Sale un número entre 0 y 22 que corresponde por posición a una letra de la siguiente **cadena** (ojo, no lo puedes meter en un vector): TRWAGMYFPDXBNJZSQVHLCKE

Si el DNI no es un número válido (o no es un número) devolverá el carácter asterisco. Dicho DNI puede pasarse con o sin puntos (en las posiciones adecuadas) de separación de millares: Es decir 33444555 ó 33.444.555

En el programa principal habrá un menú con tres opciones numéricas: Obtener letra del DNI, comprobar DNI (el usuario introducirá el DNI completo con letra justo después del número, usa substring para quedarte con el DNI) y salir.

Amplía con la posibilidad de pasar desde línea de comando al main un dni sin letra. Si no se le pasa nada se lanza el menú tal cual se describió. Si se le pasa un parámetro y es un DNI sin letra, simplemente muestra la letra y acaba. Si es otra cosa cualquiera da un mensaje de error y acaba. Mira el **Apéndice III** para más información.

Ejercicios opcionales:

11. Realiza una aplicación para trabajo con matrices matemáticas y que se pueda realizar distintas tareas con las mismas: multiplicaciones, sumas, determinantes, transposiciones, inversiones, etc.

12. Juego campo de minas. El jugador parte de la esquina inferior izquierda de una región de la pantalla. Tiene que llegar a la parte superior derecha. En la pantalla hay un número de minas distribuidas de forma aleatoria y que no se ven. La cantidad dependerá del nivel de dificultad elegido. En pantalla, el jugador dispondrá de un rudimentario radar que indica el número de minas que tiene alrededor. Dispondrá de tres vidas. Los movimientos son arriba, abajo, izquierda y derecha.

Una vez que llegue a su destino, el juego comenzará de nuevo con una mina más.

Se puntúa algo por cada movimiento y un bonus por llegar al final de fase (La fase se debe indicar en pantalla). También se realizará una bonificación por tiempo.

La estructura principal será un array del tamaño de la pantalla. Utiliza cierta codificación para indicar lo que hay en cada casilla, por ejemplo: 0 vacía, 1 humano, 2 mina,... incluso valores superiores para otros objetos buenos o malos que se te ocurran. Luego el interfaz de usuario es dibujar distintas cosas en pantalla según el contenido del array.

13. Realizar el juego hundir la flota (1 jugador contra el ordenador). El ordenador colocará en un tablero de 8x8 cinco barcos: 1 de dos casillas, 3 de 3 casillas, 1 de 4 casillas en horizontal o vertical. Todo esto de forma aleatoria. También el usuario podrá colocar los barcos en su tablero. Una vez terminada la colocación ordenador y jugador irán disparando coordenadas por turnos hasta que uno de los dos haya hundido todos los barcos del otro. Si uno acierta en un barco del otro, podrá repetir tirada.

14. Realizar algún juego de tablero para 2 ó más personas (Ajedrez, damas, oca, othelo, etc.)

15. Realiza una aventura conversacional sencilla. Establecerás un array bidimensional de objetos tipo "escenario" de forma que definas en cada escenario colecciones de objetos y personajes además de una descripción y distintas tareas que puedes usar. Puedes omar como referencia algún juego conversacional clásico como el Cobra's Arc, Zork o La guerra de las Vajillas. A continuación unos enlaces con videos o en el caso del Zork también el juego on-line.

Cobra's Arc: https://www.youtube.com/watch?v=fNY4B9KdKxQ

La guerra de las vajillas: https://www.youtube.com/watch?v=GKEACSpW8Ks

Zork: https://www.youtube.com/watch?v=64-VQjygTGc

Zork y otros para jugar on-line: http://www.web-adventures.org

16. Juego Mastermind. El ordenador saca una combinación de 4 símbolos de 7 posibles y el usuario tiene que adivinarla. En cada turno el usuario da una combinación y el ordenador le dice cuantos aciertos de símbolos hay y cuantos

aciertos de símbolo y posición hay. Usa vectores de 4 elementos. Se deben guardar todas las tiradas en una colección para realizar un informe final: número de tiradas, momento de mayor acierto (salvo el final), momento de menor acierto y otras que se te ocurran. Más información en: http://es.wikipedia.org/wiki/Mastermind

17. Realiza el juego del ahorcado. Las palabras inicialmente estarán en un vector de 20 posiciones. Trabaja sólo con mayúsculas. Debe aparecer el dibujo correspondiente a cada fallo.

18. Realiza un programa para hacer cálculos básicos desde la línea de comando. A dicho programa se le pasará como primer argumento un símbolo de operación (+, - , *, /) y a continuación una ristra de números. En el caso de la suma y la multiplicación operará con todos los números que se le pasa. En el caso de la resta al primer número le resta los siguientes y en el caso de la división divide el primer numero entre los siguientes. En el caso de que aparezca un 0 en alguna posición de la división simplemente lo ignora y pasa al número siguiente.

Si lo deseas puedes ampliarlo con otras operaciones como raiz cuadrada, potencias… diseña tú el formato que debe tener la línea de comando.

Ejemplo: *$ java calcula + 5 6 4 10*

y muestre el 25. Por supuesto la lista de números será indeterminada.