Ejercicios de colecciones de objetos (I): Arrays

1. Hacer un programa que pida por teclado números enteros (un máximo de 30, o hasta que se introduzca el cero). En ese momento el programa mostrará todos los números introducidos, separados por comas (excepto el último cero, si se hubiera introducido).
2. Igual que 1, pero quitando de la lista final, el máximo y el mínimo, y mostrando también la media aritmética de todos ellos (incluyendo el máximo y el mínimo).
   PISTA: Hacer métodos que calculen el máximo, el mínimo y la media aritmética de un array hasta una posición determinada

3. Hacer un programa que pida por teclado las dos coordenadas de un par de vectores planos, y muestre a continuación su suma vectorial y su producto escalar
   Ejemplo:
   Introduce x1: **-1**
   Introduce y1: **-3**
   Introduce x2: **1**
   Introduce y2: **-1**
   (-1,-3)+(1,-1)=(0,-4)
   (-1,-3).(1,-1)= -2

4. **Mensajes secretos**: Hacer un programa que pida por teclado palabras hasta que se introduzca la palabra “FIN” en mayúsculas, o hasta introducir un máximo de 30. Una vez hecho esto, mostrará una palabra formada por la primera letra de cada palabra introducida (excepto la palabra “FIN”, si se hubiera introducido), una detrás de otra en la misma línea.
   Ejemplo:
   Palabra 1: **S**eguimos
   Palabra 2: **e**scribiendo
   Palabra 3: **c**osas
   Palabra 4: **r**ecurrentes
   Palabra 5: **e**n
   Palabra 6: **t**iempos
   Palabra 7: **o**scuros
   Palabra 8: FIN

   Palabra secreta: **Secreto**

5. Hacer un programa que muestre sólo los elementos de casillas impares de un array.
6. Hacer un programa que dado un número “n” muestre el primer elemento de un array y todos los que pueda del resto, saltando de n en n elementos.
7. Hacer un programa que haga lo mismo que el anterior, pero que obtenga a través de “args” de main, el número “n”, así como los elementos a mostrar.
   Ejemplo de argumento de entrada: 3 a b c d e f g h i j k l m n o

Salida del programa: a d g j m

8. Hacer una programa Calculadora que funcione de la siguiente manera "**java Calculadora <operacion> <op1> <op2>**". Funcionará para las operaciones "suma" "resta" "mul" y "div" y operandos decimales.
9. Hacer un programa que implemente la clase Persona, con atributos nombre y apellido y método saludar() que muestre "Hola, soy <nombre> <apellido>". Coleccionar datos de Persona's a través de args, indicando en el primer parámetro el número de personas a coleccionar y en el resto sus nombres y apellidos, y al finalizar, hacer que todas ellas saluden.
10. Hacer un programa que reciba por argumento de entrada dos valores "nTiradas" y "nCaras", y simule a continuación "nTiradas" de un dado de "nCaras", mostrando el número el número de veces que ha salido cada una de las caras con respecto al total, así como el porcentaje que éste representa.
11. Implementar el algoritmo de ordenación por el método de la burbuja, en un método estático cuyo prototipo es **burbuja(int[ ]):int[ ]**
12. Probar el algoritmo anterior desde un programa que recibe mediante los argumentos de entrada de main los elementos a ordenar.
13. Crear una clase, llamada **Matriz2**, que "envuelva" un atributo de tipo **int[ ][ ]** que contenga los datos de una matriz bidimensional.
14. Para la clase **Matriz2**, crear un constructor **create(fila:int, col:int, String[ ])** , que inicialice la matriz con las dimensiones **fila** y **col** indicadas, y con los datos proporcionados mediante un array lineal de String. Utilizar este constructor en combinación con "args" de main, para inicializar una matriz. Hacer una versión en la que se le proporcionen las dimensiones de la matriz también a traves de "args".
15. Para la clase **Matriz2**, reescribir el método **toString():String**, que devuelva un String que muestre el contenido de la matriz, tal como visualmente se mostraría en matemáticas (por filas y columnas). NOTA: Utilizar los caracteres de escape **\t** y **\n** para tabular y cambiar de línea
16. Para la clase **Matriz2**, crear un constructor **create(int[ ] [ ]),** igual que el del ejercicio 14, pero los datos vienen ya dados en una matriz bidimensional de enteros. Ojo. No igualar el array proporcionado al atributo envuelto. Se debe crear una copia de todos sus valores.
17. Para la clase **Matriz2**, crear un método **sumar(Matriz2):Matriz2**, que devuelva el resultado de la suma matricial de la matriz pasada por argumento con la matriz donde se ejecuta el método. Para poder realizar este método, impleméntense un par de métodos **get(f,c):int** y **put(f,c,int)**, que hagan de getters y setters para los valores de la matriz en su posición f (fila) y c (columna). Crear una excepción *Matriz2Exception* que se lance cuando las dimensiones de la matriz a sumar no coincida.
18. Para la clase **Matriz2**, crear un método **mult(Matriz2):Matriz2**, que devuelva el resultado de la multiplicación matricial de la matriz pasada por argumento con la matriz donde se ejecuta el método. El método debe lanzar una excepción *Matriz2Exception* cuando las dimensiones de las matrices involucradas no cuadren.

19. Hacer un método **ordenar(String):String**, que dado un String (p.ej: “abzbdf”) me devuelva otro String en el que aparecen todas sus letras (que sólo pueden ser las letras minúsculas, de la “a” a la “z”), pero ordenadas alfabéticamente (es decir, en el ejemplo anterior “abbdfz”). El número máximo de caracteres de cada palabra será 20.
20. Hacer un programa que utilice el método anterior para probarlo, requiriendo que se introduzcan por teclado el número de palabras que se quiere procesar, y a continuación cada una de esas palabras. El programa mostrará por pantalla cada una de las palabras anteriores, pero cuyas letras han sido ordenadas

    EJEMPLO de ENTRADA
    6
    abc
    cbd
    zzz
    y
    abcdef
    zyx

    EJEMPLO de SALIDA
    abc
    bcd
    zzz
    y
    abcdef
    xyz

21. Utilizar la redirección de la entrada/salida estándar para probar que el programa funciona para todas las posibles combinaciones de palabras de 2 letras